Package 'qmtools'

July 3, 2025

Title Quantitative Metabolomics Data Processing Tools

Version 1.13.0

Description The qmtools (quantitative metabolomics tools) package provides basic tools for processing quantitative metabolomics data with the standard SummarizedExperiment class. This includes functions for imputation, normalization, feature filtering, feature clustering, dimension-reduction, and visualization to help users prepare data for statistical analysis. This package also offers a convenient way to compute empirical Bayes statistics for which metabolic features are different between two sets of study samples. Several functions in this package could also be used in other types of omics data.

License GPL-3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

biocViews Metabolomics, Preprocessing, Normalization, DimensionReduction, MassSpectrometry

Imports rlang, ggplot2, patchwork, heatmaply, methods, MsCoreUtils, stats, igraph, VIM, scales, grDevices, graphics, limma

Suggests Rtsne, missForest, vsn, pcaMethods, pls, MsFeatures, impute, imputeLCMD, nlme, testthat (>= 3.0.0), BiocStyle, knitr, rmarkdown

Depends R (>= 4.2.0), SummarizedExperiment

BugReports https://github.com/HimesGroup/qmtools/issues

URL https://github.com/HimesGroup/qmtools

Config/testthat/edition 3

VignetteBuilder knitr

LazyData false

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

git_branch devel

git_last_commit e6f19a5

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-07-02

Author Jaehyun Joo [aut, cre],
Blanca Himes [aut]

Maintainer Jaehyun Joo <jaehyunjoo@outlook.com>

Contents

`qmtools-package` *qmtools: Quantitative Metabolomics Data Processing Tools*

Description

The qmtools (quantitative metabolomics tools) package provides basic tools for processing quantitative metabolomics data with the standard SummarizedExperiment class. This includes functions for imputation, normalization, feature filtering, feature clustering, dimension-reduction, and visualization to help users prepare data for statistical analysis. This package also offers a convenient way to compute empirical Bayes statistics for which metabolic features are different between two sets of study samples. Several functions in this package could also be used in other types of omics data.

Details

The qmtools package provides six categories of important functions:

- Imputation: the imputeIntensity function performs data imputation on missing values.
- Normalization: the normalizeIntensity function performs data-driven normalization on metabolomics data.
- Feature filtering: the removeFeatures function removes uninformative features based on missing values, QC and blank samples.
- Feature clustering: the clusterFeatures function clusters metabolic features according to their retention time and intensity correlation.
- Dimension-reduction: the reduceFeatures function performs dimensionality reduction.
- Visualization: the plotBox, plotCorr, plotMiss, plotReduced, and plotRTgroup functions help to visualize metabolomics data.

Please refer to the vignette to see how the aforementioned functions work.

Author(s)

Maintainer: Jaehyun Joo <jaehyunjoo@outlook.com>

Authors:

- Blanca Himes <bhimes@pennmedicine.upenn.edu>

See Also

Useful links:

- https://github.com/HimesGroup/qmtools
- Report bugs at https://github.com/HimesGroup/qmtools/issues

---

`clusterFeatures` *Feature clustering*

Description

Function to cluster LC-MS features according to their retention time and intensity correlation across samples with a SummarizedExperiment.

Usage

```
clusterFeatures(
  x,
  i,
  rtime_var = "rtime",
  rt_cut = 10,
  cor_cut = 0.7,
  rt_grouping = c("hclust", "closest", "consecutive"),
  cor_grouping = c("louvain", "SimilarityMatrix", "connected", "none"),
  cor_use = c("everything", "all.obs", "complete.obs", "na.or.complete",
    "pairwise.complete.obs"),
```

```
  cor_method = c("pearson", "kendall", "spearman"),
  log2 = FALSE,
  hclust_linkage = "complete"
)
```

Arguments

x	A SummarizedExperiment object.
i	A string or integer value specifying which assay values to use.
rtime_var	A string specifying the name of variable containing a numeric vector of retention times in rowData(x).
rt_cut	A numeric value specifying a cut-off for the retention-time based feature grouping.
cor_cut	A numeric value specifying a cut-off for the correlation-based feature grouping.
rt_grouping	A string specifying which method to use for the retention-time based feature grouping.
cor_grouping	A string specifying which method to use for the correlation-based feature grouping.
cor_use	A string specifying which method to compute correlations in the presence of missing values. Refer to ?cor for details.
cor_method	A string specifying which correlation coefficient is to be computed. See ?cor for details.
log2	A logical specifying whether feature intensities need to be log2-transformed before calculating a correlation matrix.
hclust_linkage	A string specifying the linkage method to be used when rt_grouping is "hclust".

Details

For soft ionization methods (e.g., LC/ESI-MS) commonly used in metabolomics, one or more ions could be generated from an individual compound upon ionization. The redundancy of feature data needs to be addressed since we typically interested in compounds rather than different ion species. This function attempts to identify a group of features from the same compound with the following steps:

1. Features are grouped by their retention times to identify co-eluting compounds.
2. For each retention time-based group, features are further clustered by patterns of the intensity correlations across samples to identify a subset of features from the same compound.

The retention time-based grouping is performed using either a hierarchical clustering via hclust or the methods available in the **MsFeatures** package via MsFeatures::groupClosest and MsFeatures::groupConsecutive. For the rt_grouping = "hclust", by default, complete-linkage clustering is conducted using the Manhattan distance (i.e., difference in retention times) where the distance between two clusters is defined as the difference in retention times between the farthest pair of elements in the two clusters. Group memberships are assigned by specifying the cut height for the distance metric. Other linkage methods can be specified with hclust_linkage. Please refer to ?hclust for details. For the "closest" and "consecutive", please refer to ?MsFeatures::groupClosest and ?MsFeatures::groupConsecutive for the details of algorithms.

For the correlation-based grouping, cor_grouping = "connected" creates a undirected graph using feature correlations as an adjacency matrix (i.e., correlations serve as edge weights). The edges

whose weights are below the cut-off specified by `cor_cut` will be removed from the graph, separating features into several disconnected subgroups. Features in the same subgroup will be assigned to the same feature cluster. For the "louvain", the function further applies the Louvain algorithm to the graph in order to identify densely connected features via igraph::cluster_louvain. For the "SimilarityMatrix", MsFeatures::groupSimilarityMatrix is used for feature grouping. Please refer to `?MsFeatures::groupSimilarityMatrix` for the details of algorithm.

Value

A SummarizedExperiment object with the grouping results added to columns "rtime_group" (initial grouping on retention times) and "feature_group" in its `rowData`.

References

Johannes Rainer (2022). MsFeatures: Functionality for Mass Spectrometry Features. R package version 1.3.0. 'https://github.com/RforMassSpectrometry/MsFeatures

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre: Fast unfolding of communities in large networks. J. Stat. Mech. (2008) P10008

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. https://igraph.org

See Also

See hclust, cutree, MsFeatures::groupClosest, MsFeatures::groupConsecutive, MsFeatures::groupSimilarityMatrix, and igraph::cluster_louvain for the underlying functions that do work.

See plotRTgroup to visualize the grouping result.

Examples

```
data(faahko_se)

se <- clusterFeatures(faahko_se, i = "knn_vsn", rtime_var = "rtmed")
rowData(se)[, c("rtmed", "rtime_group", "feature_group")]
```

`compareSamples` *Sample comparison*

Description

Function to make a comparisons between two groups in study samples with a SummarizedExperiment.

Usage

```
compareSamples(
  x,
  i,
  group,
  class1,
  class2,
```

```
  covariates = NULL,
  confint = TRUE,
  number = nrow(x),
  adjust.method = "BH",
  sort.by = "B",
  resort.by = NULL,
  p.value = 1,
  fc = NULL,
  lfc = NULL,
  ...
)
```

Arguments

x	A SummarizedExperiment object.
i	A string or integer value specifying which assay values to use. The assay is expected to contain log-transformed intensities.
group	A string specifying the name of variable containing a class label of each sample in `colData(x)`.
class1, class2	A string specifying the class label of samples to be compared. Must be one of `group` levels. No need to be specified if `group` has only two levels. This function evaluates the contrast: `class2 - class1`.
covariates	A vector indicating the names of variables to be included in the model as covariates. The covariates must be found in `colData(x)`.
confint	A logical specifying whether 95% confidence intervals of log-fold-change need to be reported. Alternatively, a numeric value between zero and one specifying the confidence level required.
number	The maximum number of metabolic features to list.
adjust.method	A string specifying which p-value adjustment method to use. Options, in increasing conservatism, include "none", "BH", "BY" and "holm". See p.adjust for the complete list of options. A NULL value will result in the default adjustment method, which is "BH".
sort.by	A string specifying which statistic to rank the metabolic features by. Possible values for topTable are "logFC", "AveExpr", "t", "P", "p", "B" or "none" (Permitted synonyms are "M" for "logFC", "A" or "Amean" for "AveExpr", "T" for "t" and "p" for "P").
resort.by	A string specifying statistic to sort the selected metabolic features by in the output. Possibilities are the same as for sort.by.
p.value	A numeric value specifying a cut-off for adjusted p-values. Only metabolic features with lower p-values are listed.
fc	A numeric value specifying a minimum fold-change to be required. If specified, the function output only includes metabolic features with absolute fold-change greater than `fc`.
lfc	A numeric value specifying a minimum log-fold-change required. `fc` and `lfc` are alternative ways to specify a fold-change cut-off and, if both are specified, then `fc` take precedence.
...	Additional arguments passed to limma::eBayes.

Details

This function provides a simplified interface of fitting a linear model to make a comparison of interest using the limma::lmFit, limma::eBayes, and limma::topTable functions. For more flexible model specifications (e.g., interaction model, multi-level model), please use a standard workflow outlined in the `limma` package user's guide.

Value

A data.frame with a row for the metabolic features and the following columns:

- logFC: an estimate of log-fold-change corresponding to the contrast tested
- CI.L: a left limit of confidence interval for `logFC` (if `confint` is enabled)
- CI.R: a right limit of confidence interval for `logFC` (if `confint` is enabled)
- AveExpr: an average log-expression/abundance of metabolic features
- t: a moderated t-statistic
- P.Value: a raw p-value
- adj.P.Value: an adjusted p-value
- B: a log-odds that the metabolic feature is differentially expressed

References

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK. limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res. 2015 Apr 20;43(7):e47. doi: 10.1093/nar/gkv007. Epub 2015 Jan 20. PMID: 25605792; PMCID: PMC4402510.

See Also

See limma::lmFit, limma::eBayes, and limma::topTable for underlying functions that do work.

Examples

```
data(faahko_se)

compareSamples(faahko_se, i = "knn_vsn", group = "sample_group", number = 5)
```

---

faahko_se *FAAH knockout LC/MS data SummarizedExperiment*

Description

A SummarizedExperiment object containing FAAH knockout LC/MS feature intensity data from the **faahKO** package created using the `faahko3` data.

Usage

```
data(faahko_se)
```

Format

An object of class `SummarizedExperiment` with 206 rows and 12 columns.

References

Colin A. Smith (2021). faahKO: Saghatelian et al. (2004) FAAH knockout LC/MS data. R package version 1.32.0. http://dx.doi.org/10.1021/bi0480335

Examples

```
data(faahko_se)
```

`imputeIntensity` *Imputation methods*

Description

Performs a variety of data imputation methods on a matrix-like object or SummarizedExperiment object. The methods include k-Nearest Neighbors (kNN), Random Forest (RF), and many others from the MsCoreUtils::impute_matrix. See the details below.

Usage

```
## S4 method for signature 'ANY'
imputeIntensity(
  x,
 method = c("knn", "rf", "bpca", "QRILC", "MLE", "MinDet", "MinProb", "min", "zero",
    "mixed", "nbavg", "with", "none"),
  ...
)

## S4 method for signature 'SummarizedExperiment'
imputeIntensity(
  x,
 method = c("knn", "rf", "bpca", "QRILC", "MLE", "MinDet", "MinProb", "min", "zero",
    "mixed", "nbavg", "with", "none"),
  i,
  name,
  ...
)
```

Arguments

`x`	A matrix-like object or SummarizedExperiment object.
`method`	A string specifying which imputation method to use.
`...`	Arguments passed to a specific imputation method.
`i`	A string or integer value specifying which assay values to use when `x` is a SummarizedExperiment object.
`name`	A string specifying the name to be used to store the imputed intensities in `x` when `x` is a SummarizedExperiment object. If not specified, a matrix containing the imputed intensities is returned.

Details

The method argument can be one of "knn", "rf", "bpca", "QRILC", "MLE", "MinDet", "MinProb", "min", "zero", "mixed", "nbavg", "with", "none". Please choose one that best describes the nature of missing data. While this function provides several simple imputation methods, they may only work under restrictive assumptions.

- "knn" performs kNN imputation based on the Gower distance or Euclidean distance. See imputeKNN for details.
- "rf" performs random forest imputation using the missForest::missForest, as described in Stekhoven D. J., & Buehlmann, P. (2012). This method is not sensitive to monotonic transformations of the intensity matrix.
- For the other method arguments, please refer to the MsCoreUtils::impute_matrix. Briefly,
  - "bpca": Bayesian PCA missing value imputation.
  - "QRILC": Quantile regression approach for the imputation of left-censored missing data.
  - "MLE": Maximum likelihood-based imputation.
  - "MinDet": Deterministic minimal value approach for the imputation of left-censored data.
  - "MinProb": Stochastic minimal value approach for the imputation of left-censored data.
  - "min": Replace the missing values with the smallest non-missing value in the data.
  - "zero": Replace the missing values with 0.
  - "mixed": Mixed imputation applying two methods.
  - "nbavg": Average neighbour imputation for fractions collected along a fractionation/separation gradient.
  - "with": Replace the missing values with a user-provided value.
  - "none": Reserved for the "mixed" method.

Value

A matrix or SummarizedExperiment object of the same dimension as `x` containing the imputed intensities.

References

Laurent Gatto, Johannes Rainer and Sebastian Gibb (2021). MsCoreUtils: Core Utils for Mass Spectrometry Data. R package version 1.4.0. https://github.com/RforMassSpectrometry/MsCoreUtils

Stekhoven D. J., & Buehlmann, P. (2012). MissForest - non-parametric missing value imputation for mixed-type data. Bioinformatics, 28(1), 112-118.

See Also

See imputeKNN, missForest::missForest, and MsCoreUtils::impute_matrix for the underlying functions that do work.

Examples

```
data(faahko_se)

## SummarizedExperiment object
se <- imputeIntensity(faahko_se, i = "raw", name = "imp1", method = "knn")
assayNames(se)
```

```
## Matrix
m <- assay(faahko_se, i = "raw")
imputeIntensity(m, method = "min")
```

`imputeKNN` *k-nearest neighbor imputation*

Description

Performs k-nearest neighbor (kNN) imputation on a matrix-like object where rows represent features and columns represent samples. This function finds k-nearest neighbors using either Gower distance or Euclidean distance.

Usage

```
imputeKNN(
  x,
  k = 10,
  type = c("gower", "euclidean"),
  by = c("feature", "sample"),
  scale = FALSE,
  ...
)
```

Arguments

`x`	A matrix-like object.
`k`	An integer specifying the number of nearest neighbors to be used in imputation.
`type`	A string specifying the distance metric to be used. Either "gower" or "euclidean".
`by`	A string specifying whether the imputation is performed by k-nearest features or by k-nearest samples. Either "feature" or "sample".
`scale`	A logical specifying whether `x` needs to be standardized prior to the imputation when Euclidean distance is used. The imputed values are re-transformed so that they are on the original scales.
`...`	Arguments passed to VIM::kNN (Gower distance) or impute::impute.knn (Euclidean distance).

Details

The kNN imputation based on Euclidean distance typically requires standardization of input data to avoid variance-based weighting of variables (make variables on similar scales). When Gower distance is used, the imputation can be done with original units (would get the same result with the standardized input on a different scale). The `type` "gower" utilizes the VIM::kNN and "euclidean" uses the impute::impute.knn.

Value

A matrix of the same dimension as `x` containing the imputed intensities.

References

Trevor Hastie, Robert Tibshirani, Balasubramanian Narasimhan and Gilbert Chu (2021). impute: impute: Imputation for microarray data. R package version 1.66.0.

Alexander Kowarik, Matthias Templ (2016). Imputation with the R Package VIM. Journal of Statistical Software, 74(7), 1-16. doi:10.18637/jss.v074.i07

See Also

See imputeIntensity that provides a SummarizedExperiment-friendly wrapper for this function.

See VIM::kNN and missForest::missForest for the underlying functions that do work.

Examples

```
data(faahko_se)

m <- assay(faahko_se, "raw")
imputeKNN(m)
```

`normalizeIntensity` *Normalization methods*

Description

Performs a few data-driven normalization methods on a matrix-like object or SummarizedExperiment object. The methods include probabilistic quotient normalization (PQN), cyclic loess normalization, feature-based scaling, and many others from the MsCoreUtils::normalize_matrix. See the details below.

Usage

```
## S4 method for signature 'ANY'
normalizeIntensity(
  x,
 method = c("pqn", "div.sum", "div.mean", "div.median", "div.mad", "center.mean",
    "center.median", "diff.median", "cyclicloess", "vsn", "quantiles",
    "quantiles.robust", "feature.scale"),
  ...
)

## S4 method for signature 'SummarizedExperiment'
normalizeIntensity(
  x,
 method = c("pqn", "div.sum", "div.mean", "div.median", "div.mad", "center.mean",
    "center.median", "diff.median", "cyclicloess", "vsn", "quantiles",
    "quantiles.robust", "feature.scale"),
  i,
  name,
  ...
)
```

Arguments

x	A matrix-like object or SummarizedExperiment object.
method	A string specifying which normalization method to use.
...	Arguments passed to a specific normalization method.
i	A string or integer value specifying which assay values to use when x is a SummarizedExperiment object.
name	A string specifying the name to be used to store the normalized intensities in x when x is a SummarizedExperiment object. If not specified, a matrix containing the normalized intensities is returned.

Details

The method argument can be one of "pqn", "cyclicloess", "vsn", "feature.scale", "div.sum", "div.mean", "div.median", "div.mad", "center.mean", "center.median", "diff.median", "quantiles", and "quantiles.robust".

- "pqn" performs probabilistic quotient normalization, as described in Dieterle et al. (2006). See normalizePQN for details.
- "cyclicloess" performs cyclic LOESS normalization using the limma::normalizeCyclicLoess. The input x is expected to contain log-transformed intensities. See Bolstad et al. (2003) and Ballman et al. (2004) for details. Please use type if you want to specify a cyclic loess method due to a name conflict with the existing argument in this function.
- "vsn" performs variance stabilizing normalization (VSN), as described in Huber et al. (2002). It produces normalized intensities based on a glog (generalized logarithm) scale. See the vsn::vsn2 for details.
- "feature.scale" performs feature-based scaling (applied along the rows) as described in van den Berg et al. (2006). See scaleRows for details.
- For "div.sum", "div.mean", "div.median", and "div.mad", the respective sample intensities are divided by the column sums, means, medians, or median absolute deviations. See scaleCols for details.
- "center.mean" and "center.median" center the intensities by subtracting the column means or medians, respectively.
- "diff.median" centers all samples so that they all match the grand median by subtracting the respective columns medians differences to the grand median.
- "quantiles" and "quantiles.robust" perform quantiles normalization, as described in Bolstad et al. (2003). See the preprocessCore::normalize.quantiles and preprocessCore::normalize.quantiles.robust for details.

Value

A matrix or SummarizedExperiment object of the same dimension as x containing the normalized intensities.

References

Laurent Gatto, Johannes Rainer and Sebastian Gibb (2021). MsCoreUtils: Core Utils for Mass Spectrometry Data. R package version 1.4.0. https://github.com/RforMassSpectrometry/MsCoreUtils

Dieterle F, Ross A, Schlotterbeck G, Senn H. Probabilistic quotient normalization as robust method to account for dilution of complex biological mixtures. Application in 1H NMR metabonomics. Anal Chem. 2006 Jul 1;78(13):4281-90. doi: 10.1021/ac051632c. PMID: 16808434.

Ritchie ME, Phipson B, Wu D, Hu Y, Law CW, Shi W, Smyth GK. limma powers differential expression analyses for RNA-sequencing and microarray studies. Nucleic Acids Res. 2015 Apr 20;43(7):e47. doi: 10.1093/nar/gkv007. Epub 2015 Jan 20. PMID: 25605792; PMCID: PMC4402510.

Bolstad BM, Irizarry RA, Astrand M, Speed TP. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. Bioinformatics. 2003 Jan 22;19(2):185-93. doi: 10.1093/bioinformatics/19.2.185. PMID: 12538238.

Ballman KV, Grill DE, Oberg AL, Therneau TM. Faster cyclic loess: normalizing RNA arrays via linear models. Bioinformatics. 2004 Nov 1;20(16):2778-86. doi: 10.1093/bioinformatics/bth327. Epub 2004 May 27.

Huber W, von Heydebreck A, Sültmann H, Poustka A, Vingron M. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. Bioinformatics. 2002;18 Suppl 1:S96-104. doi: 10.1093/bioinformatics/18.suppl_1.s96. PMID: 12169536.

van den Berg RA, Hoefsloot HC, Westerhuis JA, Smilde AK, van der Werf MJ. Centering, scaling, and transformations: improving the biological information content of metabolomics data. BMC Genomics. 2006 Jun 8;7:142. doi: 10.1186/1471-2164-7-142. PMID: 16762068; PMCID: PMC1534033.

See Also

See normalizePQN, scaleRows, scaleCols, limma::normalizeCyclicLoess, and MsCoreUtils::normalize_matrix for the underlying functions that do the work.

Examples

```
data(faahko_se)

## SummarizedExperiment object
se <- normalizeIntensity(faahko_se, i = "knn", name = "knn_pqn",
                         method = "pqn")
assayNames(se)

##' ## Matrix
m <- assay(faahko_se, "knn")
normalizeIntensity(m, method = "feature.scale", type = "pareto")
```

---

normalizePQN *Probabilistic quotient normalization (PQN)*

Description

Performs probabilistic quotient normalization (PQN) on a matrix-like object where rows present features and columns represent samples.

Usage

```
normalizePQN(x, ref_samples = NULL, min_frac = 0.5, type = c("median", "mean"))
```

Arguments

x	A matrix-like object.
ref_samples	A vector of sample names or column indices specifying reference samples for the calculation of quotients. Must be a subset of colnames(x) if it is a character vector. If NULL, all samples are used.
min_frac	A numeric value between 0 and 1 specifying a minimum proportion of reference samples for features to be included in the calculation of a reference spectrum.
type	A method to compute a reference spectrum. Either "median" or "mean".

Details

For the calculation of quotients, a reference spectrum needs to be obtained from a median or mean spectrum based on all spectra of the study or a subset of the study. Feature intensities are normalized by the median of quotients. See Dieterle et al. (2006) for details.

Value

A matrix of the same dimension as x containing the normalized intensities.

References

Dieterle F, Ross A, Schlotterbeck G, Senn H. Probabilistic quotient normalization as robust method to account for dilution of complex biological mixtures. Application in 1H NMR metabonomics. Anal Chem. 2006 Jul 1;78(13):4281-90. doi: 10.1021/ac051632c. PMID: 16808434.

See Also

See normalizeIntensity that provides a SummarizedExperiment-friendly wrapper for this function.

Examples

```
data(faahko_se)

m <- assay(faahko_se, "knn")
normalizePQN(m)
```

---

plotBox *Box plot*

Description

Produces a box-and-whisker plot with a SummarizedExperiment or matrix-like object where rows represent features and columns represent samples.

Usage

```
plotBox(x, i, group, log2 = FALSE, violin = FALSE, ylab = "Intensity")
```

Arguments

x	A matrix-like object or SummarizedExperiment object.
i	A string or integer value specifying which assay values to use when x is a SummarizedExperiment object.
group	A discrete variable to visualize the grouping structure.
log2	A logical specifying whether feature intensities needs to be log2-transformed before visualization.
violin	A logical specifying whether a violin plot is shown instead of a boxplot.
ylab	A string specifying the title of y-axis.

Value

A ggplot object.

Examples

```
data(faahko_se)

## Sample group
g <- colData(faahko_se)$sample_group

## SummarizedExperiment object
plotBox(faahko_se, i = "knn", group = g, log2 = TRUE) # before normalization

## Matrix
m <- assay(faahko_se, "knn_vsn")
plotBox(m, group = g) # after normalization
```

---

plotCorr *Correlation plot*

Description

Visualizes correlations between samples or features with a SummarizedExperiment or matrix-like object where rows represent features and columns represent samples. A correlation matrix is visualized using a heatmap with dendrograms.

Usage

```
plotCorr(
  x,
  i,
  type = c("sample", "feature"),
  log2 = FALSE,
  use = c("everything", "all.obs", "complete.obs", "na.or.complete",
    "pairwise.complete.obs"),
  method = c("pearson", "kendall", "spearman"),
  dendrogram = TRUE,
  colors = (scales::viridis_pal())(256),
```

```
  label = FALSE,
  digits = 2,
  widths = c(0.8, 0.2),
  heights = c(0.2, 0.8),
  hide_colorbar = FALSE,
  showticklabels = c(TRUE, TRUE),
  row_dend_left = FALSE,
  k_row = 1,
  k_col = 1,
  ...
)
```

Arguments

x	A matrix-like object or SummarizedExperiment object.
i	A string or integer value specifying which assay values to use when x is a SummarizedExperiment object.
type	A string specifying whether a correlation matrix is computed based on samples or features.
log2	A logical specifying whether feature intensities needs to be log2-transformed before calculating a correlation matrix.
use	A string specifying which method to compute correlations in the presence of missing values. Refer to ?cor for details.
method	A string specifying which correlation coefficient is to be computed. See ?cor for details.
dendrogram	A logical specifying whether dendogram is computed and reordering is performed.
colors	A vector of colors for the heatmap.
label	A logical specifying whether cell values are shown.
digits	A numeric value specifying the desired number of digits when label = TRUE.
widths	A numeric vectors specifying relative widths of heatmap and dendrogram.
heights	A numeric vectors specifying relative heights of heatmap and dendrogram.
hide_colorbar	A logical specifying whether the color bar (legend) is hidden.
showticklabels	A logical vector of length 2 (x-axis, y-axis) specifying whether the ticks are removed from the sides of the plot.
row_dend_left	A logical controlling whether the row dendrogram is placed on the left on the plot.
k_row	A numeric value specifying the desired number of groups by which to color the dendrogram's branches in the rows. If NA, then dendextend::find_k is used to deduce the optimal number of clusters.
k_col	A numeric value specifying the desired number of groups by which to color the dendrogram's branches in the columns. If NA, then dendextend::find_k is used to deduce the optimal number of clusters.
...	Additional arguments passed to heatmaply::heatmaply.

Value

A patchwork object of aligned ggplots.

References

Tal Galili, Alan O'Callaghan, Jonathan Sidi, Carson Sievert; heatmaply: an R package for creating interactive cluster heatmaps for online publishing, Bioinformatics, btx657, https://doi.org/10.1093/bioinformatics/btx657

Examples

```
data(faahko_se)

## Sample group
g <- colData(faahko_se)$sample_group

## SummarizedExperiment object
plotCorr(faahko_se, i = "knn_vsn", method = "spearman", k_col = 4)

## Matrix
m <- assay(faahko_se, "knn_vsn")
plotCorr(m[1:50, ], type = "feature", method = "spearman")
```

---

plotMiss *Missing value plot*

Description

Visualizes missing values with a SummarizedExperiment object or matrix of intensity data where rows represent features and columns represent samples. All values in a data matrix are re-coded (1: missing; 0: non-missing). The left panel displays the amount of missing values in each samples. The right panel displays the pattern of missing values using a heatmap with dendrograms.

Usage

```
plotMiss(
  x,
  i,
  group,
  dendrogram_row = TRUE,
  dendrogram_col = FALSE,
  colors = (scales::viridis_pal())(2),
  hide_colorbar = TRUE,
  showticklabels = c(TRUE, FALSE),
  row_dend_left = FALSE,
  k_row = 1,
  k_col = 1,
  ...
)
```

Arguments

x	A matrix-like object or SummarizedExperiment object.
i	A string or integer value specifying which assay values to use when x is a SummarizedExperiment object.

group	A discrete variable to change colors of the barplot by sample groups.
dendrogram_row	A logical specifying whether dendogram is computed and reordering is performed based on rows.
dendrogram_col	A logical specifying whether dendogram is computed and reordering is performed based on columns.
colors	A vector of colors for the heatmap.
hide_colorbar	A logical specifying whether the color bar (legend) in the heatmap is hidden.
showticklabels	A logical vector of length 2 (x-axis, y-axis) specifying whether the ticks are removed from the sides of the heatmap.
row_dend_left	A logical controlling whether the row dendrogram is placed on the left on the heatmap.
k_row	A numeric value specifying the desired number of groups by which to color the dendrogram's branches in the rows. If NA, then dendextend::find_k is used to deduce the optimal number of clusters.
k_col	A numeric value specifying the desired number of groups by which to color the dendrogram's branches in the columns. If NA, then dendextend::find_k is used to deduce the optimal number of clusters.
...	Additional arguments passed to heatmaply::heatmaply.

Value

A patchwork object of aligned ggplots

References

Tal Galili, Alan O'Callaghan, Jonathan Sidi, Carson Sievert; heatmaply: an R package for creating interactive cluster heatmaps for online publishing, Bioinformatics, btx657, https://doi.org/10.1093/bioinformatics/btx657

Examples

```
data(faahko_se)

## Sample group
g <- colData(faahko_se)$sample_group

## SummarizedExperiment object
plotMiss(faahko_se, i = 1, group = g)

## Matrix
m <- assay(faahko_se, i = 1)
plotMiss(m, group = g, dendrogram_col = TRUE)
```

plotReduced *Score plot of dimension-reduced data*

Description

Function to visualize dimension-reduced data matrices mainly produced by reduceFeatures, including `reduced.pca`, `reduced.tsne`, and `reduced.plsda` objects (or a matrix with the same structure).

Usage

```
plotReduced(
  x,
  comp = c(1, 2),
  biplot = FALSE,
  group,
  group_col = NULL,
  point_size = 1.5,
  point_shape_by_group = FALSE,
  label = FALSE,
  label_size = 3.88,
  label_subset = NULL,
  ellipse = FALSE,
  xlab = NULL,
  ylab = NULL,
  title = NULL,
  legend = TRUE,
  arrow_len = 0.2,
  arrow_col = "orange",
  arrow_alpha = 0.3,
  arrow_label = TRUE,
  arrow_label_ext = 1.05,
  arrow_label_size = 3.88,
  arrow_label_col = "orange",
  arrow_label_subset = NULL
)
```

Arguments

`x`	A matrix containing the dimension-reduced data typically produced by reduceFeatures.
`comp`	A numeric vector of length 2 specifying the components to display.
`biplot`	A logical specifying whether visualize an overlay of scores and loadings. Ignored if xis not a `reduced.pca` or `reduced.plsda` object.
`group`	A discrete variable to visualize the grouping structure. For a `reduced.plsda` object, group information used to fit the PLS-DA is used if not specified.
`group_col`	A vector of colors with the same length of unique values in `group`.
`point_size`	A numeric value specifying the size of points.

point_shape_by_group
: A logical specifying whether each group has different shapes of data points. Also can be a numeric vector with the same length of unique values in group to manually set point shapes.

label
: A logical specifying whether score labels are shown instead of points.

label_size
: A numeric value controlling the size of labels.

label_subset
: A character vector specifying a subset of score labels to display.

ellipse
: A logical specifying whether data ellipses are shown.

xlab
: A string specifying the title of x-axis.

ylab
: A string specifying the title of y-axis.

title
: A string specifying the main title of the plot.

legend
: A logical specifying whether the plot legend is shown.

arrow_len
: A numeric value specifying the length of arrow head.

arrow_col
: A string specifying the color of arrows.

arrow_alpha
: A numeric value specifying the transparency of arrow.

arrow_label
: A logical specifying whether text labels for arrows are shown.

arrow_label_ext
: A numeric value specifying the scalar extension for arrow labels.

arrow_label_size
: A numeric value specifying the size of arrow labels.

arrow_label_col
: A string specifying the color of arrow labels.

arrow_label_subset
: A character vector specifying a subset of arrow labels to display.

Value

A ggplot object.

Examples

```
data(faahko_se)

## Sample group
g <- colData(faahko_se)$sample_group

## PCA
pca_res <- reduceFeatures(faahko_se, i = "knn_vsn", method = "pca")

## Visualizes the result
plotReduced(pca_res, group = g)
plotReduced(pca_res, group = g, label = TRUE, ellipse = TRUE)
```

plotRTgroup *Helper to visualize feature grouping*

Description

Visualizes feature grouping results produced by clusterFeatures. A retention-time based feature group is displayed with its sub-groups based on the feature intensity correlations either using a pair plot or graph. Features with the same color indicate that they are in the same group.

Usage

```
plotRTgroup(
  x,
  i,
  group,
  type = c("graph", "pairs"),
  rtime_group_var = "rtime_group",
  feature_group_var = "feature_group",
  cor_cut = 0.7,
  cor_use = c("everything", "all.obs", "complete.obs", "na.or.complete",
    "pairwise.complete.obs"),
  cor_method = c("pearson", "kendall", "spearman"),
  log2 = FALSE
)
```

Arguments

x	A SummarizedExperiment object.
i	A string or integer value specifying which assay values to use. Choose the same value used in the feature grouping.
group	A string specifying the label of retention time-based group to visualize.
type	A string specifying which type of plots to visualize.
rtime_group_var	A string specifying the names of variable containing the retention-time based grouping result in rowData(x).
feature_group_var	A string specifying the names of variable containing the final feature grouping result in rowData(x).
cor_cut	A numeric value specifying a cut-off for the visualizing correlations in a graph as edges. Ignored if type is "pairs".
cor_use	A string specifying which method to compute correlations in the presence of missing values. Refer to ?cor for details. Choose the same value used in the feature grouping. Ignored if type is "pairs".
cor_method	A string specifying which correlation coefficient is to be computed. See ?cor for details. Choose the same value used in the feature grouping. Ignored if type is "pairs".
log2	A logical specifying whether feature intensities needs to be log2-transformed before calculating a correlation matrix. Ignored if type is "pairs". Choose the same value used in the feature grouping.

Value

A graph or pair plot.

See Also

See clusterFeatures for feature grouping.

Examples

```
data(faahko_se)

## Clustering
se <- clusterFeatures(faahko_se, i = "knn_vsn", rtime_var = "rtmed")

## Graph
plotRTgroup(se, i = "knn_vsn", group = "FG.22")

## Pairwise scatter
plotRTgroup(se, i = 3, group = "FG.22", cor_method = "spearman",
            log2 = TRUE, type = "pairs")
```

---

reduceFeatures *Dimension reduction methods*

Description

Performs dimensionality reduction on a matrix-like object or SummarizedExperiment object.

Usage

```
## S4 method for signature 'ANY'
reduceFeatures(x, method = c("pca", "tsne", "plsda"), ncomp = 2, y, ...)

## S4 method for signature 'SummarizedExperiment'
reduceFeatures(x, method = c("pca", "tsne", "plsda"), ncomp = 2, i, y, ...)
```

Arguments

x	A matrix-like object or SummarizedExperiment object.
method	A string specifying which dimension-reduction method to use.
ncomp	A integer specifying the number of components extract.
y	A factor vector for the information about each sample's group.
...	Arguments passed to a specific dimension-reduction method.
i	A string or integer value specifying which assay values to use when x is a SummarizedExperiment object.

Details

Currently, principal component analysis (PCA), t-distributed stochastic neighbor embedding (t-SNE), and partial least squares-discriminant analysis (PLS-DA) are supported. For the method argument,

`pca` performs PCA using singular value decomposition. If there is any missing value, the non-linear iterative partial least squares (NIPALS) algorithm is used instead using the pcaMethods::nipalsPca. See reducePCA for details.

`tsne` performs t-SNE using the Rtsne::Rtsne. See reduceTSNE for details.

`plsda` performs PLS-DA using a standard PLS model for classification with the pls::plsr. See reducePLSDA for details.

Value

A matrix containing custom attributes related to the dimension-reduction method used.

References

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. In P. R. Krishnajah (Ed.), Multivariate analysis (pp. 391-420). NewYork: Academic Press.

Stacklies, W., Redestig, H., Scholz, M., Walther, D. and Selbig, J. pcaMethods – a Bioconductor package providing PCA methods for incomplete data. Bioinformatics, 2007, 23, 1164-1167

L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008.

L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15(Oct):3221-3245, 2014.

Jesse H. Krijthe (2015). Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation, URL: https://github.com/jkrijthe/Rtsne

Kristian Hovde Liland, Bjørn-Helge Mevik and Ron Wehrens (2021). pls: Partial Least Squares and Principal Component Regression. R package version 2.8-0. https://CRAN.R-project.org/package=pls

See Also

See reducePCA, reduceTSNE, and reducePLSDA for the underlying functions that do the work.

Examples

```
data(faahko_se)

## SummarizedExperiment object
res_pca <- reduceFeatures(faahko_se, i = "knn_vsn", method = "pca")
summary(res_pca)

## Matrix
y <- factor(colData(faahko_se)$sample_group)
m <- assay(faahko_se, i = "knn_vsn")
res_plsda <- reduceFeatures(m, method = "plsda", y = y, ncomp = 3)
summary(res_plsda)
```

`reducePCA` *Principal component analysis (PCA)*

Description

Performs PCA on a matrix-like object where rows represent features and columns represents samples.

Usage

```
reducePCA(x, ncomp = 2, center = TRUE, scale = FALSE, ...)
```

Arguments

`x`	A matrix-like object.
`ncomp`	An integer specifying the number of components to extract.
`center`	A logical specifying whether `x` needs to be mean-centered prior to PCA.
`scale`	A logical specifying whether the unit variance scaling needs to be performed on `x` prior to PCA.
`...`	Additional arguments passed to pcaMethods::nipalsPca. Ignored if `x` has no missing values.

Details

For the data without missing values, PCA is performed with the transpose of `x` via singular value decomposition. Otherwise, PCA is performed with the transpose of `x` using the non-linear iterative partial least squares (NIPALS) algorithm via the pcaMethods::nipalsPca. The function returns a `reduced.pca` object that is a matrix with custom attributes to summarize (via summary) and visualize (via plotReduced) the PCA result. The custom attributes include the following:

- `method`: The method used to reduce the dimension of data.
- `ncomp`: The number of components extracted.
- `R2`: A vector indicating the amount of variance explained by each principal component.
- `R2cum`: A vector of cumulative R2.
- `loadings`: A matrix of variable loadings.
- `sdev`: A vector indicating the standard deviations of the principal components.
- `centered`: A logical indicating whether the data was mean-centered prior to PCA.
- `scaled`: A logical indicating whether the data was scaled prior to PCA.

Value

A reduced.pca object with the same number of rows as `ncol(x)` containing the dimension reduction result.

References

Wold, H. (1966). Estimation of principal components and related models by iterative least squares. In P. R. Krishnajah (Ed.), Multivariate analysis (pp. 391-420). NewYork: Academic Press.

Stacklies, W., Redestig, H., Scholz, M., Walther, D. and Selbig, J. pcaMethods – a Bioconductor package providing PCA methods for incomplete data. Bioinformatics, 2007, 23, 1164-1167

See Also

See reduceFeatures that provides a SummarizedExperiment-friendly wrapper for this function.

See plotReduced for visualization.

See pcaMethods::nipalsPca for the underlying function that does the work.

Examples

```
data(faahko_se)

m <- assay(faahko_se, "knn_vsn")
res <- reducePCA(m, ncomp = 3)
summary(res)
```

---

reducePLSDA *Partial least squares-discriminant analysis (PLS-DA)*

Description

Performs PLS-DA on a matrix-like object where rows represent features and columns represent samples.

Usage

```
reducePLSDA(
  x,
  y,
  ncomp = 2,
  center = TRUE,
  scale = FALSE,
  validation = c("none", "CV", "LOO"),
  return_mvr = FALSE,
  ...
)
```

Arguments

x	A matrix-like object.
y	A factor vector for the information about each sample's group.
ncomp	A integer specifying the number of components to extract.
center	A logical specifying whether x and y matrices need to be mean-centered prior to PLS-DA.
scale	A logical specifying whether the unit variance scaling needs to be performed on x prior to PLS-DA.
validation	A string specifying a validation method to use. See pls::plsr for the details.
return_mvr	A logical indicating whether mvr object is returned. It is recommended for a user to use utilities functions from the pls package for the model fit.
...	Additional arguments passed to pls::plsr.

Details

This function performs standard PLS for classification with the transpose of `x` using the pls::plsr. Since PLS-DA is a supervised method, users must supply the information about each sample's group. Here, `y` must be a factor so that it can be internally converted to an indicator matrix. The function returns a `reduced.plsda` object that is a matrix with custom attributes to summarize (via summary) and visualize (via plotReduced) the PLS-DA result. The custom attributes include the following:

- `method`: The method used to reduce the dimension of data.
- `ncomp`: The number of components extracted.
- `explvar`: A vector indicating the amount of X variance explained by each component.
- `responses`: A vector indicating the levels of factor `y`.
- `predictors`: A vector of predictor variables.
- `coefficient`: An array of regression coefficients.
- `loadings`: A matrix of loadings.
- `loadings.weights`: A matrix of loading weights.
- `Y.observed`: A vector of observed responses.
- `Y.predicted`: A vector of predicted responses.
- `Y.scores`: A matrix of Y-scores.
- `Y.loadings`: A matrix of Y-loadings.
- `projection`: The projection matrix.
- `fitted.values`: An array of fitted values.
- `residuals`: An array of regression residuals.
- `vip`: An array of VIP (Variable Importance in the Projection) coefficients.
- `centered`: A logical indicating whether the data was mean-centered prior to PLS-DA.
- `scaled`: A logical indicating whether the data was scaled prior to PLS-DA.
- `validation`: Results of the validation if requested.

Value

A reduced.plsda object with the same number of rows as `ncol(x)` containing the dimension reduction result.

References

Kristian Hovde Liland, Bjørn-Helge Mevik and Ron Wehrens (2021). pls: Partial Least Squares and Principal Component Regression. R package version 2.8-0. https://CRAN.R-project.org/package=pls

See Also

See reduceFeatures that provides a SummarizedExperiment-friendly wrapper for this function.

See plotReduced for visualization.

See pls::plsr for the underlying function that does the work.

Examples

```
data(faahko_se)

m <- assay(faahko_se, "knn_vsn")
y <- factor(colData(faahko_se)$sample_group)
res <- reducePLSDA(m, y = y)
summary(res)
```

`reduceTSNE` *t-distributed stochastic neighbor embedding (t-SNE)*

Description

Performs t-SNE on a matrix-like object where rows represent features and columns represent samples.

Usage

```
reduceTSNE(x, ncomp = 2, normalize = TRUE, ...)
```

Arguments

`x`	A matrix-like object.
`ncomp`	A integer specifying the number of components to extract. Must be either 1, 2, or 3.
`normalize`	A logical specifying whether the input matrix is mean-centered and scaled so that the largest absolute of the centered matrix is equal to unity. See Rtsne::normalize_input for details.
`...`	Additional arguments passed to Rtsne::Rtsne.

Details

t-SNE is well-suited for visualizing high-dimensional data by giving each data point a location in a two or three-dimensional map. This function performs t-SNE with the transpose of `x` using Rtsne::Rtsne and returns a `reduced.tsne` object that is a matrix with custom attributes to summarize (via summary) and visualize (via plotReduced) the t-SNE result. The custom attributes include the following:

- `method`: The method used to reduce the dimension of data.
- `ncomp`: The number of components extracted.
- `perplexity`: The perplexity parameter used.
- `theta`: The speed/accuracy trade-off parameter used.
- `normalized`: A logical indicating whether the data was normalized prior to t-SNE.

Value

A reduced.tsne object with the same number of rows as `ncol(x)` containing the dimension reduction result.

References

L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008.

L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15(Oct):3221-3245, 2014.

Jesse H. Krijthe (2015). Rtsne: T-Distributed Stochastic Neighbor Embedding using a Barnes-Hut Implementation, URL: https://github.com/jkrijthe/Rtsne

See Also

See reduceFeatures that provides a SummarizedExperiment-friendly wrapper for this function.

See plotReduced for visualization.

See Rtsne::Rtsne for the underlying function that does the work.

Examples

```
data(faahko_se)

m <- assay(faahko_se, "knn_vsn")
res <- reduceTSNE(m, perplexity = 3)
summary(res)
```

`removeBlankRatio` *Feature Filtering based on QC/blank ratio*

Description

Removes Features with based on QC/blank ratios using the data matrix where rows represent features and columns represent samples. A feature will be retained if there are not enough blank samples to calculate an intensity ratio for a feature (or completely absent in blank samples). Use removeMiss to remove features based on a proportion of missing values. Features with a QC/blank ratio below a cut-off will be discarded.

Usage

```
removeBlankRatio(
  x,
  blank_samples,
  qc_samples,
  cut = 2,
  type = c("median", "mean"),
  blank_min_n = 3
)
```

Arguments

x	A matrix-like object.
blank_samples	A vector of sample names or column indices specifying blank samples for the calculation of ratio. Must be a subset of colnames(x) if it is a character vector.
qc_samples	A vector of sample names or column indices specifying QC samples for the calculation of ratio. Must be a subset of colnames(x) if it is a character vector.
cut	A numeric value greater than 1 specifying a QC/blank ratio cut-off to retain a feature.
type	A method to compute a QC/blank ratio. Either "median" or "mean".
blank_min_n	An integer value specifying the minimum number of blank samples to calculate a ratio.

Value

A matrix containing the filtered features.

See Also

See removeFeatures that provides a SummarizedExperiment-friendly wrapper for this function.

Examples

```
set.seed(1e7)

m_blank <- matrix(rlnorm(200), ncol = 5)
m_qc <- matrix(rlnorm(400, 1), ncol = 10)
m <- cbind(m_blank, m_qc)
colnames(m) <- c(paste0("B", seq_len(5)), paste0("Q", seq_len(10)))

removeBlankRatio(m, blank_samples = paste0("B", seq_len(5)),
                 qc_samples = paste0("Q", seq_len(10)))
```

---

removeFeatures *Feature Filtering methods*

Description

Removes Features based on missing values, QC and blank samples. See the details below.

Usage

```
## S4 method for signature 'ANY'
removeFeatures(x, method = c("missing", "blankratio", "rsd", "icc"), ...)

## S4 method for signature 'SummarizedExperiment'
removeFeatures(x, method = c("missing", "blankratio", "rsd", "icc"), i, ...)
```

Arguments

x	A matrix-like object or SummarizedExperiment object.
method	A string specifying which filtering method to use.
...	Arguments passed to a specific filtering method.
i	A string or integer value specifying which assay values to use when x is a SummarizedExperiment object.

Details

The method argument can be one of "missing", "blankratio", "rsd", "icc".

- "missing" removes features based on proportions of missing values. Users can specify one or more groups in samples. For multiple groups, a feature is retained if there is at least one group with a proportion of non-missing values above a cut-off.
- For "blankratio", QC/blank intensity ratios are calculated for features present at the blank samples. Features with a ratio below a cut-off will be discarded.
- "rsd" calculates a relative standard deviation (also known as coefficient of variation) for each feature using QC samples. Features with a RSD above a cut-off will be removed.
- "icc" calculates an intraclass correlation coefficient (ICC) for each feature using both biological and QC samples to identify how much of the total variation is explained by biological variability, as described in Schiffman, C et al (2019). Features with an ICC below a cut-off will be removed.

Value

A matrix or SummarizedExperiment object.

References

Schiffman, C., Petrick, L., Perttula, K. et al. Filtering procedures for untargeted LC-MS metabolomics data. BMC Bioinformatics 20, 334 (2019). https://doi.org/10.1186/s12859-019-2871-9

See Also

See removeMiss, removeBlankRatio, removeRSD, and removeICC for the underlying functions that do work.

Examples

```
data(faahko_se)

g <- colData(faahko_se)$sample_group

## SummarizedExperiment object
se <- removeFeatures(faahko_se, i = "raw", method = "missing",
                     group = g, cut = 0.9)

## Matrix
m <- assay(faahko_se, i = "raw")
removeFeatures(m, method = "missing", group = g, levels = "WT", cut = 0.9)
```

`removeICC` *Feature Filtering based on ICC*

Description

Removes Features based on a intraclass correlation coefficient (ICC) using the data matrix where rows represent features and columns represent samples. For each feature, ICC will be calculated using both biological and QC samples to identify how much of the total variation is explained by biological variability, as described in Schiffman, C et al (2019). Informative features are expected to have relatively high variability across the biological samples, compared to QC replicates. Features with an ICC below a cut-off will be removed.

Usage

```
removeICC(
  x,
  qc_samples,
  bio_samples = setdiff(colnames(x), qc_samples),
  cut = 0.4
)
```

Arguments

`x`	A matrix-like object.
`qc_samples`	A vector of sample names or column indices specifying QC samples for the calculation of ICC. Must be a subset of `colnames(x)` if it is a character vector.
`bio_samples`	A vector of sample names or column indices specifying biological samples for the calculation of ICC. Must be a subset of `colnames(x)` if it is a character vector.
`cut`	A numeric value between 0 and 1 specifying a ICC cut-off to retain a feature.

Value

A matrix containing the filtered features.

References

Schiffman, C., Petrick, L., Perttula, K. et al. Filtering procedures for untargeted LC-MS metabolomics data. BMC Bioinformatics 20, 334 (2019). https://doi.org/10.1186/s12859-019-2871-9

See Also

See removeFeatures that provides a SummarizedExperiment-friendly wrapper for this function.

Examples

```
set.seed(1e7)

m_bio_1 <- matrix(rlnorm(600, sdlog = 1), ncol = 20)
m_bio_2 <- matrix(rlnorm(200, sdlog = 0.3), ncol = 20)
m_bio <- rbind(m_bio_1, m_bio_2)
```

```
m_qc <- matrix(rlnorm(400, sdlog = 0.25), ncol = 10)
m <- cbind(m_bio, m_qc)
colnames(m) <- c(paste0("S", seq_len(20)), paste0("Q", seq_len(10)))

removeICC(m, qc_samples = paste0("Q", seq_len(10)),
          bio_samples = paste0("S", seq_len(20)))
```

`removeMiss` *Feature filtering based on proportions of missing values*

Description

Removes Features based on proportions of missing values in the matrix where rows represent features and columns represent samples. Features can be removed based on missing values within a specific group or multiple groups. A feature will be retained, if there is at least one group with a proportion of non-missing values above a cut-off.

Usage

```
removeMiss(x, group, levels = NULL, cut = 0.7)
```

Arguments

`x`	A matrix-like object.
`group`	A character vector for the information about each sample's group.
`levels`	A string or character vector specifying one or more groups for filter filtering based on missing values. If `NULL`, all group levels in `group` will be used.
`cut`	A numeric value between 0 and 1 specifying a minimum proportion of non-missing values to retain a feature.

Value

A matrix containing the filtered features.

See Also

See removeFeatures that provides a SummarizedExperiment-friendly wrapper for this function.

Examples

```
data(faahko_se)
m <- assay(faahko_se, "raw")
g <- colData(faahko_se)$sample_group
table(g)

## Filter based on missing values in "KO" and "WT" groups
removeMiss(m, group = g, cut = 0.9)

## Consider only "KO" group (can be useful for QC-based filtering)
removeMiss(m, group = g, levels = "KO", cut = 0.9)
```

removeRSD *Feature Filtering based on RSD*

Description

Removes Features with low reproducibility based on a relative standard deviation (also known as coefficient of variation) of QC samples using the data matrix where rows represent features and columns represent samples. Features with a RSD above a cut-off will be removed from the data.

Usage

```
removeRSD(x, qc_samples, cut = 0.3)
```

Arguments

x	A matrix-like object.
qc_samples	A vector of sample names or column indices specifying QC samples for the calculation of RSD. Must be a subset of `colnames(x)` if it is a character vector.
cut	A numeric value between specifying a RSD cut-off to retain a feature.

Value

A matrix containing the filtered features.

See Also

See removeFeatures that provides a SummarizedExperiment-friendly wrapper for this function.

Examples

```
set.seed(1e7)

m_bio <- matrix(rlnorm(800, sdlog = 1), ncol = 20)
m_qc <- matrix(rlnorm(400, sdlog = 0.25), ncol = 10)
m <- cbind(m_bio, m_qc)
colnames(m) <- c(paste0("S", seq_len(20)), paste0("Q", seq_len(10)))

removeRSD(m, qc_samples = paste0("Q", seq_len(10)))
```

scaleCols *Scale along columns (samples)*

Description

Function to scale a matrix of intensity data along the columns (samples).

Usage

```
scaleCols(
  x,
  type = c("div.sum", "div.mean", "div.median", "div.mad"),
  restrict = FALSE,
  rescale = FALSE
)
```

Arguments

x	A matrix-like object.
type	A scaling method to use.
restrict	A logical specifying whether only features that are common to all samples are used in the calculation of scaling factors.
rescale	A logical specifying whether the normalized intensities are re-scaled by multiplying the median of normalization factors to make look similar to the original scale.

Details

Sample intensities are divided by the column sums ("div.sum"), means ("div.mean"), medians ("div.median"), or median absolute deviations ("div.mad").

Value

A matrix of the same dimension as x containing the scaled intensities.

See Also

See normalizeIntensity that provides a SummarizedExperiment-friendly wrapper for this function.

Examples

```
data(faahko_se)

m <- assay(faahko_se, "knn")
scaleCols(m)
```

---

scaleRows *Scale along rows (features)*

Description

Function to scale a matrix of intensity data along the rows (features), as described in van den Berg et al. (2006).

Usage

```
scaleRows(
  x,
  type = c("auto", "range", "pareto", "vast", "level", "sum", "max")
)
```

Arguments

x	A matrix-like object.
type	A scaling method to use.

Details

This function will do the following:

- Auto scaling (unit variance scaling): each feature is mean-centered and divided by its standard deviation.
- Range scaling: each feature is mean-centered and divided by its range.
- Pareto scaling: each feature is mean-centered and divided by the square root of its standard deviation.
- Vast scaling (variance stability scaling): it is an extension of auto scaling, using the product of standard deviation and coefficient of variation as a scale factor.
- Level scaling: each feature is mean-centered and divided by its mean.
- Sum scaling: each feature is divided by its sum.
- Max scaling: each feature is divided by its maximum.

Value

A matrix of the same dimension as x containing the scaled intensities.

References

van den Berg RA, Hoefsloot HC, Westerhuis JA, Smilde AK, van der Werf MJ. Centering, scaling, and transformations: improving the biological information content of metabolomics data. BMC Genomics. 2006 Jun 8;7:142. doi: 10.1186/1471-2164-7-142. PMID: 16762068; PMCID: PMC1534033.

See Also

See normalizeIntensity that provides a SummarizedExperiment-friendly wrapper for this function.

Examples

```
data(faahko_se)

m <- assay(faahko_se, "knn")
scaleRows(m, type = "pareto")
```

Index