Package 'metabolomicsR'

April 7, 2022

Type Package

Title Tools for Metabolomics Data

Version 1.0.0

Date 2022-01-10

Maintainer 'Xikun Han <hanxikun2017@gmail.com>

URL https://github.com/XikunHan/metabolomicsR

Description Tools to preprocess, analyse, and visualize metabolomics data.
We included a set of functions for sample and metabolite quality control,
outlier detection, missing value imputation, dimensional reduction, normalization,
data integration, regression, metabolite annotation, enrichment analysis,
and visualization of data and results. The package is designed to be a comprehensive R package
that can be easily used by researchers with basic R programming skills.
The framework designed here is versatile and is extensible to other various methods.

License GPL-2

Encoding UTF-8

Depends methods, R (>= 4.1)

Imports ggplot2, data.table, plotROC, utils, stats

Suggests ggthemes, knitr, rmarkdown, testthat (>= 3.0.0), lme4, nlme,
broom, reshape2, impute, M3C, FNN, RColorBrewer, readxl,
survival, future, pbapply, future.apply, progressr, ggrepel,
here, genuMet, ggstatsplot, cowplot, pROC, BiocStyle, MASS,
xgboost

RoxygenNote 7.1.2

biocViews Software, Metabolomics, MassSpectrometry, Regression,
Normalization, QualityControl

LazyData FALSE

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Xikun Han [cre, aut]

R topics documented:

assayData *get assayData*

Description

Accessors for Metabolite object. Get the assayData in the Metabolite object.

Usage

```
assayData(object)

## S4 method for signature 'Metabolite'
assayData(object)
```

Arguments

object	A Metabolite object.

assayData<- *set assayData*

Description

Accessors for Metabolite object. 'assayData<-' will update the assayData in the Metabolite object.

Usage

```
assayData(object) <- value

## S4 replacement method for signature 'Metabolite'
assayData(object) <- value
```

Arguments

object	A Metabolite object.
value	The new assayData.

Value

assayData

batch_norm *batch normalization*

Description

Normalization data by the median value of each batch

Usage

```
batch_norm(
  object,
  feature_platform = "PLATFORM",
  QC_ID_pattern = "MTRX",
  test = FALSE,
  verbose = TRUE
)
```

Arguments

object	A Metabolite object. In the feature annotation slot 'feature', a platform column should be provided for metabolite measurement platform (eg. 'PLATFORM'). The values in the 'PLATFORM' column (eg. 'Neg', 'Polar', 'Pos Early', and 'Pos Late') are column names in the sample annotation 'sample' to determine the batches of samples.
feature_platform	The column name of feature platform for metabolite measurements (eg. 'PLATFORM').
QC_ID_pattern	A character pattern to determine QC samples. Default value: "MTRX".
test	test the function for the first 20 columns.
verbose	print log information.

Value

A Metabolite object after normalization.

See Also

QCmatrix_norm

bridge *bridge different data sets based on conversion factors*

Description

Bridge metabolite data based on a conversion factor file

Usage

```
bridge(
  object,
  conversion_factor_data = NULL,
  QC_ID_pattern = "MTRX",
  verbose = TRUE
)
```

Arguments

object	A Metabolite object. In the 'featureData', 'conversion_factor_ID' column should be created to match with conversion_factor_data.
conversion_factor_data	A data set with columns 'conversion_factor_ID' and 'conversion_factor_value'.
QC_ID_pattern	A character pattern to determine QC samples. Default value: "MTRX". Skip QC samples when rescale (median value is already 1).
verbose	print log information.

Value

A Metabolite object after multiplying by conversion factor.

---

column_missing_rate *column missing rate*

Description

Calculate column missing rate – metabolite missingness.

Usage

```
column_missing_rate(object)

## Default S3 method:
column_missing_rate(object)

## S3 method for class 'Metabolite'
column_missing_rate(object)
```

Arguments

object	An object, data.frame, data.table or Metabolite.

Value

Returns a vector of the missing rate for each column

Examples

```
## Not run:
# for a data.frame or data.table
v <- column_missing_rate(object = df)

# to skip the first column (eg. ID)
v <- column_missing_rate(object = df[, -1])

## End(Not run)

## Not run:

# for a Metabolite object
v <- column_missing_rate(object)


## End(Not run)
```

`correlation` *correlation of features between two Metabolite objects*

Description

Calculate the correlation of features between two Metabolite objects

Usage

```
correlation(
  object_X = NULL,
  object_Y = NULL,
  method = "pearson",
  verbose = TRUE
)
```

Arguments

`object_X`	The first Metabolite object.
`object_Y`	The second Metabolite object.
`method`	a character string to calculate correlation coefficient. One of "pearson" (default), "kendall", or "spearman".
`verbose`	print log information.

Value

A data.table with correlation coefficients.

See Also

`cor`

`create_Metabolite` *Create a Metabolite object*

Description

Create a Metabolite object from three input data sets: 1) metabolite measurements (eg. peak area data or normalized data), and 2) metabolite annotation (eg. chemical annotation) 3) sample annotation (eg. sample meta data)

Usage

```
create_Metabolite(
  assayData,
  featureData,
  sampleData,
  featureID,
  sampleID,
  logs
)
```

Arguments

`assayData`	a data.frame or data.table of metabolite measurements (peak area data or normalized data, sample [row] * feature [column]).
`featureData`	a data.frame or data.table of metabolite annotation (chemical annotation)
`sampleData`	a data.frame or data.table of sample annotation (sample meta data).
`featureID`	a character of the metabolite ID column (in feature file and the column names of data), default: CHEM_ID (provided from Metabolon file).
`sampleID`	a character of the sample ID column (in sample and the first column of data), default: PARENT_SAMPLE_NAME (provided from Metabolon file).
`logs`	Log information.

Value

A Metabolite object with slots: assayData, featureData, and sampleData.

See Also

`Metabolite`, `load_excel`, `load_data`

Examples

```
## Not run:

df <- create_Metabolite(assayData = df_data, featureData = df_feature, sampleData =  df_sample)


## End(Not run)
```

df_plasma *Example data.*

Description

A dataset containing 356 samples and 758 features.

Usage

```
data(df_plasma)
```

Format

An object of class `Metabolite` of length 1.

featureData *get featureData*

Description

Accessors for Metabolite object. Get the featureData in the Metabolite object.

Usage

```
featureData(object)

## S4 method for signature 'Metabolite'
featureData(object)
```

Arguments

`object`	A Metabolite object.

featureData<- *set featureData*

Description

Accessors for Metabolite object. 'featureData<-' will update the featureData in the Metabolite object.

Usage

```
featureData(object) <- value

## S4 replacement method for signature 'Metabolite'
featureData(object) <- value
```

Arguments

object	A Metabolite object.
value	The new featureData.

`filter_column_constant` *filter columns if values are constant*

Description

Remove columns if values are constant

Usage

```
filter_column_constant(object, verbose)

## Default S3 method:
filter_column_constant(object, verbose = TRUE)

## S3 method for class 'Metabolite'
filter_column_constant(object, verbose = TRUE)
```

Arguments

object	An object, data.frame, data.table or Metabolite.
verbose	print log information.

Examples

```
## Not run:

# for a data.frame or data.table
v <- filter_column_missing_rate(object = df)

# if skip the first column (eg. ID)
v <- filter_column_missing_rate(object = df[, -1])


## End(Not run)
```

filter_column_missing_rate

filter columns using missing rate

Description

Remove columns below a specific missing rate threshold.

Usage

```
filter_column_missing_rate(object, threshold, verbose)

## Default S3 method:
filter_column_missing_rate(object, threshold = 0.5, verbose = TRUE)

## S3 method for class 'Metabolite'
filter_column_missing_rate(object, threshold = 0.5, verbose = TRUE)
```

Arguments

object	An object, data.frame, data.table or Metabolite.
threshold	missing rate threshold, default is 0.5. Other values: 0.2, 0.8.
verbose	print log information.

Examples

```
## Not run:

d <- filter_column_missing_rate(object)


## End(Not run)
```

filter_row_missing_rate

filter rows using missing rate

Description

Remove samples below a specific missing rate threshold.

Usage

```
filter_row_missing_rate(object, threshold, verbose)

## Default S3 method:
filter_row_missing_rate(object, threshold = 0.5, verbose = TRUE)

## S3 method for class 'Metabolite'
filter_row_missing_rate(object, threshold = 0.5, verbose = TRUE)
```

Arguments

object	An object, data.frame, data.table or Metabolite.
threshold	missing rate threshold, default is 0.5. Other values: 0.2, 0.8.
verbose	print log information.

Examples

```
## Not run:

d <- filter_row_missing_rate(object)


## End(Not run)
```

---

fit_lm *available regression methods*

Description

'fit_lm': linear regression model `lm`.
'fit_logistic': logistic regression model `glm`.
'fit_poisson': poisson regression model `glm`.
'fit_cox': proportional hazards regression model `coxph`.
'fit_lme': linear mixed-effects model `lme`.
'fit_glmer': logistic linear mixed-effects model `glmer`.
'fit_lmer': linear mixed-effects model `lmer`.

Usage

```
fit_lm(data = NULL, formula = NULL, keep = NULL)

fit_logistic(data = NULL, formula = NULL, keep = NULL)

fit_poisson(data = NULL, formula = NULL, keep = NULL)

fit_cox(data = NULL, formula = NULL, keep = NULL)

fit_lme(data = NULL, formula = NULL, keep = NULL, ...)

fit_glmer(data = NULL, formula = NULL, keep = NULL, ...)

fit_lmer(data = NULL, formula = NULL, keep = NULL, ...)
```

Arguments

data	A data.table with all variables to be fitted.
formula	A "formula" object to be fitted.
keep	Variables to keep regression results.
...	Further arguments passed to regression model.

See Also

regression

---

genuMet_makefeature *distinguish genuine untargeted metabolic features without QC samples*

Description

The makefeature function from genuMet uses a Metabolite object as input. genuMet is an R package used distinguish genuine untargeted metabolic features without quality control samples.

Usage

```
genuMet_makefeature(object, wsize = 100, ssize = 0.5, defswitch = 0.2)
```

Arguments

object	A Metabolite object.
wsize	Window size.
ssize	Slide size.
defswitch	Definition of a switch.

References

<https://github.com/liucaomics/genuMet>

Examples

```
## Not run:
v <- genuMet_makefeature(df)

## End(Not run)
```

---

impute *impute missing values*

Description

impute missing values

Usage

```
impute(object, method)

## S3 method for class 'Metabolite'
impute(object, method = c("half-min", "median", "mean", "zero", "kNN"))

## Default S3 method:
impute(object, method = "half-min")

impute_kNN(object)
```

Arguments

object	An object, a vector, data.frame, data.table or Metabolite.
method	Imputation method, the default method is half the minimum value (‘half-min‘) of the metabolite. Currently support ’half-min’, "median", "mean", "zero", "kNN".

Note

default method is used for a vector

‘impute_kNN‘: Imputation using nearest neighbor averaging (kNN) method, the input is a Metabolite object, assayData was first transposed to row as metabolties and column as samples.

References

Wei, R., Wang, J., Su, M. et al. Missing Value Imputation Approach for Mass Spectrometry-based Metabolomics Data. Sci Rep 8, 663 (2018). https://doi.org/10.1038/s41598-017-19120-0

Examples

```
## Not run:
d <- impute(object)

## End(Not run)
```

---

inverse_rank_transform

rank-based inverse normal transformation

Description

rank-based inverse normal transformation for a metabolite.

Usage

```
inverse_rank_transform(x)
```

Arguments

x	A vector

Examples

```
## Not run:
v <- inverse_rank_transform(x)

## End(Not run)
```

is_outlier *is outlier*

Description

is outlier

Usage

```
is_outlier(object, nSD = 5)
```

Arguments

object	An object, a vector.
nSD	N times of the SD as outliers. Return TRUE or FALSE for a vector.

Examples

```
## Not run:
v <- is_outlier(x)

## End(Not run)
```

load_data *Load metabolite data from three separate files*

Description

Load metabolite data from three separate files (import files using ‘fread‘ from data.table).

Usage

```
load_data(
  data_path = NULL,
  feature_path = NULL,
  sample_path = NULL,
  featureID = "CHEM_ID",
  sampleID = "PARENT_SAMPLE_NAME"
)
```

Arguments

data_path	Path to the metabolite measurements (peak area data or normalized data, sample [row] * feature [column])
feature_path	Path to the metabolite annotation (chemical annotation)
sample_path	Path to the sample annotation (sample meta data)
featureID	a character of the metabolite ID column (in feature file and the column names of data file), default: CHEM_ID (provided from Metabolon file)
sampleID	a character of the sample ID column (in sample file and the first column of data file), default: PARENT_SAMPLE_NAME (provided from Metabolon file).

Value

A Metabolite object with slots: assayData, featureData, and sampleData.

`load_excel` *Load metabolite data from an excel file*

Description

Load metabolite data from an excel file

Usage

```
load_excel(
  path,
  data_sheet = NULL,
  feature_sheet = NULL,
  sample_sheet = NULL,
  featureID = "CHEM_ID",
  sampleID = "PARENT_SAMPLE_NAME"
)
```

Arguments

`path`	Path to the xls/xlsx file.
`data_sheet`	A integer of xlsx sheet number for metabolite measurements (peak area data or normalized data, sample [row] * feature [column])
`feature_sheet`	A integer of xlsx sheet number for metabolite annotation (chemical annotation)
`sample_sheet`	A integer of xlsx sheet number for sample annotation (sample meta data)
`featureID`	a character of the metabolite ID column (in feature file and the column names of data file), default: CHEM_ID (provided from Metabolon file)
`sampleID`	a character of the sample ID column (in sample file and the first column of data file), default: PARENT_SAMPLE_NAME (provided from Metabolon file)

Value

A Metabolite object with slots: assayData, featureData, and sampleData.

Examples

```
file_path <- system.file("extdata", "QMDiab_metabolomics_OrigScale.xlsx",
package = "metabolomicsR", mustWork = TRUE)

df_plasma <- load_excel(path = file_path, data_sheet = 1, feature_sheet = 4, sample_sheet = 8,
sampleID = "QMDiab-ID", featureID = "BIOCHEMICAL")
```

merge_data *merge two Metabolite objects*

Description

Merge two Metabolite objects.

Usage

```
merge_data(object_X = NULL, object_Y = NULL, all = TRUE, verbose = TRUE)
```

Arguments

object_X	The first Metabolite object.
object_Y	The second Metabolite object.
all	logical; all = TRUE: keep all metabolites; all = FALSE, keep common metabolites that were present in both datasets.
verbose	print log information.

Value

A Metabolite object after merging with slots: assayData, featureData, and sampleData.

Examples

```
# to merge two Metabolite objects
# df <- merge_data(df_plasma, df_plasma)
```

Metabolite-class *The Metabolite class*

Description

The Metabolite object is a representation of metabolomic data, metabolomic annotation, and sample annotation.

Slots

assayData a data.frame or data.table of metabolite measurements (peak area data or normalized data, sample [row] * feature [column]).

featureData a data.frame or data.table of metabolite annotation (chemical annotation)

sampleData a data.frame or data.table of sample annotation (sample meta data).

featureID a character of the metabolite ID column (in feature file and the column names of data), default: CHEM_ID (provided from Metabolon file).

sampleID a character of the sample ID column (in sample and the first column of data), default: PARENT_SAMPLE_NAME (provided from Metabolon file).

logs Log information of data analysis process.

miscData Ancillary data.

See Also

Metabolite, load_excel, load_data

---

modelling_norm *LOESS normalization*

Description

Normalization data by machine learning modelling, eg. locally estimated scatterplot smoothing (LOESS) on QC samples in each batch. For each metabolite, the values (eg. raw peak area data) were divided by the median value of QC samples in that batch. QC samples and metabolite batches should be specified (see parameters below).

Usage

```
modelling_norm(
  object,
  method = c("LOESS", "KNN", "XGBoost"),
  feature_platform = "PLATFORM",
  QC_ID_pattern = "MTRX",
  span = 0.75,
  degree = 2,
  k = 3,
  test = FALSE,
  verbose = TRUE
)
```

Arguments

object	A Metabolite object. In the feature annotation slot 'feature', a platform column should be provided for metabolite measurement platform (eg. 'PLATFORM'). The values in the 'PLATFORM' column (eg. 'Neg', 'Polar', 'Pos Early', and 'Pos Late') are column names in the sample annotation 'sample' to determine the batches of samples.
method	Modelling method for the normalization, currently support LOESS and KNN.
feature_platform	The column name of feature platform for metabolite measurements (eg. 'PLATFORM').
QC_ID_pattern	A character pattern to determine QC samples. Default value: "MTRX".
span	default value 0.4
degree	default value 2
k	Number of neighbors in KNN modelling (default value 3)
test	test the function for the first 20 columns.
verbose	print log information.

See Also

batch_norm

Examples

```
## Not run:
d <- QCmatrix_norm(object = df)

## End(Not run)
```

`nearestQC_norm` *nearest QC sample normalization*

Description

Normalization data by the median value of the nearest QC samples. For each metabolite, the values (eg. raw peak area data) were divided by the median value of nearest QC samples (eg. the nearest three QC samples). To identify the nearest QC samples, '@assayData' should be ordered by the injection order.

Usage

```
nearestQC_norm(
  object,
  n_nearest_QCsample = 3,
  feature_platform = "PLATFORM",
  QC_ID_pattern = "MTRX",
  test = FALSE,
  verbose = TRUE
)
```

Arguments

`object`	A Metabolite object. In the feature annotation slot 'feature', a platform column should be provided for metabolite measurement platform (eg. 'PLATFORM'). The values in the 'PLATFORM' column (eg. 'Neg', 'Polar', 'Pos Early', and 'Pos Late') are column names in the sample annotation 'sample' to determine the batches of samples.
`n_nearest_QCsample`	Number of nearest QC samples to calculate the median value. The default value is 3 (an outlier QC sample might be used if only n_nearest_QCsample = 1).
`feature_platform`	The column name of feature platform for metabolite measurements (eg. 'PLATFORM').
`QC_ID_pattern`	A character pattern to determine QC samples. Default value: "MTRX".
`test`	test the function for the first 20 columns.
`verbose`	print log information.

See Also

`batch_norm`, `QCmatrix_norm`

Examples

```
## Not run:
d <- nearestQC_norm(object = df)

## End(Not run)
```

`outlier_rate` *outlier rate*

Description

Calculate outlier rate.

Usage

```
outlier_rate(object, nSD)

## Default S3 method:
outlier_rate(object, nSD = 5)

## S3 method for class 'data.frame'
outlier_rate(object, nSD = 5)

## S3 method for class 'Metabolite'
outlier_rate(object, nSD = 5)
```

Arguments

`object`	An object, vector, data.frame, data.table or Metabolite.
`nSD`	N times of the SD as outliers.

Value

Returns a vector of the outlier rate.

Examples

```
## Not run:
v <- outlier_rate(x)

## End(Not run)

## Not run:

# for a Metabolite object
v <- outlier_rate(object)


## End(Not run)
```

pareto_scale *pareto scale transformation*

Description

pareto scale transformation

Usage

```
pareto_scale(x)
```

Arguments

x	A vector

Examples

```
## Not run:
v <- paretoscale(x)

## End(Not run)
```

plot_injection_order *injection order scatterplot*

Description

Injection order scatterplot. The '@sampleData' should be sorted by injection order, with a new column 'ID' from 1 to N.

Usage

```
plot_injection_order(
  object,
  color = "NEG",
  shape = "NEG",
  size = 0.6,
  ID_order = "ID_injection_order",
  feature_name = NULL,
  random_select = 16
)
```

Arguments

object	A Metabolite object.
color	A column in '@sampleData' to show the color of points.
shape	A column in '@sampleData' to show the shape of points.
size	Point size.

ID_order	Injection ID order in the '@sampleData'.
feature_name	A vector of selected metabolites to plot. If NULL, will randomly select 16 (default) metabolites to plot.
random_select	An integer, number of randomly selected metabolites to plot.

Examples

```
## Not run:
p <- plot_injection_order(df_m_PCA, color = "QC_sample")
p

p <- plot_injection_order(df_m_PCA, color = "QC_sample", feature_name = "X563")
p

## End(Not run)
```

---

plot_Metabolite *plot a Metabolite object*

Description

Plot a Metabolite object including boxplot (more to add.).

Usage

```
plot_Metabolite(
  object,
  plot = "boxplot",
  x = "NEG",
  feature_name = NULL,
  color = "NEG",
  shape = "NEG",
  fill = "NEG",
  random_select = 16,
  size = 0.6,
  n_row = 1,
  n_col = 1,
  ylab = "featureID",
  height = 10,
  width = 10,
  save_to_file = NULL
)
```

Arguments

object	A Metabolite object.
plot	type of plot, current support 'boxplot' and 'betweenstats'.
x	The x-axis coordinate.
feature_name	A vector of selected metabolites to plot. If NULL, will randomly select 16 (default) metabolites to plot.

color	A column in '@sampleData' to show the color of points.
shape	A column in '@sampleData' to show the shape of points.
fill	A column in '@sampleData' to show the 'fill' for histogram.
random_select	An integer, number of randomly selected metabolites to plot.
size	Point size.
n_row	Number of rows of subfigures for 'betweenstats'
n_col	Number of columns of subfigures for 'betweenstats'
ylab	Column name to annotate the y-axis in 'betweenstats' (eg. "BIOCHEMICAL"), default column: "featureID".
height	Height of the figure.
width	Width of the figure.
save_to_file	Path to save the figure.

Examples

```
## Not run:
p <- plot_Metabolite(df_m_PCA, plot = "boxplot")
p

## End(Not run)
```

---

plot_PCA *plot PCA*

Description

Plot first two principal components.

Usage

```
plot_PCA(object, color = "NEG", shape = "NEG", size = 1.5)
```

Arguments

object	A Metabolite object.
color	A column in '@sampleData' to show the color of points.
shape	A column in '@sampleData' to show the shape of points.
size	Point size.

plot_ROC *ROC*

Description

Plot Receiver Operating Characteristic (ROC) curve for metabolites with or without covariates

Usage

```
plot_ROC(
  object = NULL,
  y = NULL,
  x = NULL,
  model_a = NULL,
  model_b = NULL,
  lab = NULL
)
```

Arguments

object	A Metabolite object.
y	A column name for the disease (0, 1)
x	One variable name (if x is provided, model_a and model_b should be NULL or vice versa).
model_a	Column names for model a (one or more covariates, as the first model).
model_b	Column names for model b (one or more covariates, as the second model).
lab	Title (eg. "BIOCHEMICAL"), default value is x.

plot_tsne *plot tSNE*

Description

Plot t-distributed stochastic neighbor embedding. See more details in tsne.

Usage

```
plot_tsne(object, color = "NEG", shape = "NEG", size = 1.5)
```

Arguments

object	A Metabolite object.
color	A column in '@sampleData' to show the color of points.
shape	A column in '@sampleData' to show the shape of points.
size	Point size.

Examples

```
## Not run:
p<- plot_tsne(df_2017_QC_norm_PCA, color = "NEG", shape = "QCSample")
p


## End(Not run)
```

`plot_UMAP` *Plot UMAP*

Description

Plot manifold approximation and projection (UMAP). See more details in `umap`.

Usage

```
plot_UMAP(object, color = "NEG", shape = "NEG", size = 1.5)
```

Arguments

`object`	A Metabolite object.
`color`	A column in '@sampleData' to show the color of points.
`shape`	A column in '@sampleData' to show the shape of points.
`size`	Point size.

Examples

```
## Not run:
p<- plot_UMAP(df_2017_QC_norm_PCA, color = "NEG", shape = "QCSample")
p

## End(Not run)
```

`plot_volcano` *volcano plot for regression results*

Description

volcano plot for regression results

Usage

```
plot_volcano(
  fit,
  x = "estimate",
  y = "p.value",
  p.value_log10 = TRUE,
  color = "outcome",
  label = "term",
  highlight = "significant",
  x_lab = "Effect size",
  y_lab = "-log10(P value)"
)
```

Arguments

fit	regression summary results.
x	The x-axis column, eg. effect size.
y	The y-axis column, eg. p value.
p.value_log10	whether to transforme p.value by -log10.
color	A column in fit to show different point colors. Set as NULL to turn off the color argument.
label	A column in fit to label points.
highlight	A column in fit to show the points to highlight. Values as 1 are hightlighted.
x_lab	labels for x-axis.
y_lab	labels for y-axis.

Examples

```
## Not run:
p <- plot_volcano(fit_lm, color = NULL)
p

## End(Not run)
```

QCmatrix_norm *QCmatrix normalization*

Description

Normalization data by the median value of QC samples in each batch. For each metabolite, the values (eg. raw peak area data) were divided by the median value of QC samples in that batch. QC samples and metabolite batches should be specified (see parameters below).

Usage

```
QCmatrix_norm(
  object,
  feature_platform = "PLATFORM",
  QC_ID_pattern = "MTRX",
  test = FALSE,
  verbose = TRUE
)
```

Arguments

object	A Metabolite object. In the feature annotation slot 'feature', a platform column should be provided for metabolite measurement platform (eg. 'PLATFORM'). The values in the 'PLATFORM' column (eg. 'Neg', 'Polar', 'Pos Early', and 'Pos Late') are column names in the sample annotation 'sample' to determine the batches of samples.
feature_platform	The column name of feature platform for metabolite measurements (eg. 'PLATFORM').
QC_ID_pattern	A character pattern to determine QC samples. Default value: "MTRX".
test	test the function for the first 20 columns.
verbose	print log information.

See Also

batch_norm

Examples

```
## Not run:
d <- QCmatrix_norm(object = df)

## End(Not run)
```

QC_pipeline *quality control pipeline*

Description

This function will run QC steps on a Metabolite object

Usage

```
QC_pipeline(
  object,
  filter_column_constant = TRUE,
  filter_column_missing_rate_threshold = 0.5,
  filter_row_missing_rate_threshold = NULL,
  replace_outlier_method = NULL,
  nSD = 5,
```

```
  impute_method = "half-min",
  verbose = TRUE
)
```

Arguments

object	An object, data.frame, data.table or Metabolite.
filter_column_constant	A logical value, whether to filter columns (features) with a constant value.
filter_column_missing_rate_threshold	A numeric threshold to filter columns (features) below a missing rate, default: 0.5. Other values: 0.2, 0.8. If NULL, then skip this step.
filter_row_missing_rate_threshold	A numeric threshold to filter rows (samples) below a missing rate. Default: NULL, to skip this step. Other values: 0.5, 0.2, 0.8.
replace_outlier_method	Method to replace outlier value, see replace_outlier.
nSD	Define the N times of the SD as outliers.
impute_method	Imputation method, the default method is half the minimum value (‘half-min‘) of the metabolite. Currently support ’half-min’, "median", "mean", "zero".
verbose	print log information.

---

regression *regression analysis*

Description

Run regression models with adjusting for covariates. ‘regression_each‘ is used for one outcome. In ‘regression‘, several outcomes can be specified to run together.

Usage

```
regression(
  object,
  phenoData = NULL,
  model = NULL,
  outcome = NULL,
  covars = NULL,
  factors = NULL,
  feature_name = NULL,
  time = NULL,
  verbose = TRUE,
  ncpus = 1,
  p.adjust.method = "bonferroni",
  ...
)

regression_each(
  object,
```

```
  phenoData = NULL,
  model = NULL,
  formula = NULL,
  outcome = NULL,
  covars = NULL,
  factors = NULL,
  feature_name = NULL,
  time = NULL,
  verbose = TRUE,
  ncpus = 1,
  p.adjust.method = "bonferroni",
  ...
)
```

Arguments

object	A Metabolite object.
phenoData	A data.table with outcome and covariates. If 'phenoData' is NULL, '@sampleData' will be used.
model	Specify a regression model. See fit_lm for more details. 'auto' can be used to infer 'lm' or 'logistic' (with only 0, 1, and NA).
outcome	Column name of the outcome variable.
covars	Column names of covariates.
factors	Variables to be treated as factor.
feature_name	A vector of selected metabolites to run. If both feature_name and random_select are NULL, will run regression for all features.
time	Column name of survival time, used in cox regression, see coxph for more details.
verbose	Print log information.
ncpus	Number of CPUS for parallele job.
p.adjust.method	Adjust for P value method, see p.adjust.
...	Further arguments passed to regression model.
formula	A character or formula object to fit model (only used in 'regression_each')

Value

term estimate std.error statistic p.value n outcome p.value.adj.

Examples

```
data(df_plasma)
fit_lm <- regression(object = df_plasma, phenoData = NULL, model = "lm",
outcome = "BMI", covars = c("AGE", "GENDER", "ETHNICITY"), factors = "ETHNICITY")
```

`replace_outlier` *change outlier values as NA or winsorize*

Description

change outlier values as NA or winsorize

Usage

```
replace_outlier(object, method, nSD)

## Default S3 method:
replace_outlier(object, method = "winsorize", nSD = 5)

## S3 method for class 'data.frame'
replace_outlier(object, method = "winsorize", nSD = 5)

## S3 method for class 'Metabolite'
replace_outlier(object, method = "winsorize", nSD = 5)
```

Arguments

`object`	An object, a vector, data.frame, data.table or Metabolite.
`method`	Replace outlier value method, the default method is 'winsorize': replace the outlier values by the maximum and/or minimum values of the remaining values. 'as_NA': set as NA (do not use this method if using half-min imputation).
`nSD`	Define the N times of the SD as outliers.

Examples

```
## Not run:
d <- replace_outlier(object, method = "winsorize", nSD = 5)

## End(Not run)



## Not run:
d <- replace_outlier(object, method = "winsorize", nSD = 5)

## End(Not run)
```

row_missing_rate *row missing rate*

Description

Calculate row missing rate – sample missingness.

Usage

```
row_missing_rate(object)

## Default S3 method:
row_missing_rate(object)

## S3 method for class 'Metabolite'
row_missing_rate(object)
```

Arguments

object	An object, data.frame, data.table or Metabolite.

Value

Returns a vector of the missing rate for each row

Examples

```
## Not run:

# for a data.frame or data.table
v <- row_missing_rate(object = df)

# to skip the first column (eg. ID)
v <- row_missing_rate(object = df[, -1])

## End(Not run)

## Not run:

# for a Metabolite object
v <- row_missing_rate(object)


## End(Not run)
```

RSD *RSD*

Description

calculate RDS (

Usage

```
RSD(x)
```

Arguments

x	A vector

Examples

```
## Not run:
v <- RSD(x)

## End(Not run)
```

run_PCA *Principal Components Analysis*

Description

Performs a principal components analysis on the Metabolite object.

Usage

```
run_PCA(
  object,
  nPCs = 10,
  impute_method = "half-min",
  log = TRUE,
  scale = TRUE,
  addPC = TRUE
)
```

Arguments

object	A Metabolite object.
nPCs	Number of principal components to be calculated. Default value 10.
impute_method	Imputation method, the default method is half the minimum value (‘half-min‘) of the metabolite. Currently support ’half-min’, "median", "mean", "zero". ‘NULL‘ without imputation.
log	Performs natural logarithm transformation before PCA analysis.
scale	scale feature in the PCA calculation.
addPC	If TRUE, merge PCs with ‘@sampleData‘ and return the ‘object‘, else return ‘PC‘.

Examples

```
# skip the first column (eg. ID) to impute missing values
## Not run:
d <- run_PCA(object)

## End(Not run)
```

`sampleData` *get sampleData*

Description

Accessors for Metabolite object. Get the sampleData in the Metabolite object.

Usage

```
sampleData(object)

## S4 method for signature 'Metabolite'
sampleData(object)
```

Arguments

`object`	A Metabolite object.

`sampleData<-` *set sampleData*

Description

Accessors for Metabolite object. 'sampleData<-' will update the sampleData in the Metabolite object.

Usage

```
sampleData(object) <- value

## S4 replacement method for signature 'Metabolite'
sampleData(object) <- value
```

Arguments

`object`	A Metabolite object.
`value`	The new sampleData.

save_data *Save metabolite data*

Description

Save metabolite data in separate txt files

Usage

```
save_data(object, file = "")
```

Arguments

object	A Metabolite object
file	Output file to save the metabolite measurements (suffixes: "_assay.txt", "_feature_annotation.txt", "_sample_annotation.txt", "_logs.txt).

show,Metabolite-method *Print a Metabolite class object*

Description

Print a Metabolite class object

Usage

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

Arguments

object	A Metabolite object.

subset *subset a Metabolite object.*

Description

subset a Metabolite object.

Usage

```
subset(object, subset, select)

## S3 method for class 'Metabolite'
subset(object, subset, select)
```

Arguments

object	An object, data.frame, data.table or Metabolite.
subset	logical expression indicating rows to keep (samples). Expression will be evaluate in the '@sampleData'.
select	expression indicating columns to select (features). See subset. Expression will be evaluate in the '@assayData'.

---

transformation *apply transformation to a Metabolite object*

Description

Apply transformation to Metabolite object

Usage

```
transformation(object, method = "log")
```

Arguments

object	A Metabolite object.
method	Transform method, eg. "log", "pareto_scale", "scale", "inverse_rank_transform". A User defined method is also supported.

Examples

```
## Not run:
d <- transformation(x)

## End(Not run)
```

---

update_Metabolite *Update a Metabolite object*

Description

Update a Metabolite object.

Usage

```
update_Metabolite(object, dataset = NULL, action = NULL)
```

Arguments

object	A Metabolite object
dataset	A vector or data.table used for a specific action mode.
action	Currently support: • "injection_order": '@sampleData' will be updated by the order of sampleID that provided in the injection order data • "keep_feature": feature ID list to keep • "remove_feature": feature ID list to remove • "keep_sample": sample ID list to keep • "remove_sample": sample ID list to remove • "add_sample_annotation": merge data with '@sampleData' • "change_featureID": change the name of featureID (provide the new column name in '@featureData' for dataset)

Examples

```
# df_plasma <- update_Metabolite(df_plasma, dataset = "COMP_IDstr", action = "change_featureID")
```