Package 'metaboPipe'

May 28, 2024

Title Create a pipeline for metabolomics data analysis

Version 0.0.0.9000

Description
The package provides a pipeline for metabolomics data analysis. It includes functions for data preprocessing like filtering of missing values and outliers, normalization, missing value imputation and batch correction. The pipeline is implemented using the 'targets' package.

License CC BY NC SA 4.0

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Suggests knitr,
rmarkdown

VignetteBuilder knitr

Imports caret,
impute,
imputeLCMD,
missForest,
pcaMethods,
structToolbox,
SummarizedExperiment,
VIM,
tidyverse,
MetaboAnalystR (>= 4.0.0),
tinytex,
HotellingEllipse,
ggforce,
tools,
cowplot,
targets,
tarchetypes,
crew,
pmp,
fst,
shiny

Remotes xia-lab/MetaboAnalystR

Depends R (>= 2.10)

LazyData true

R topics documented:

`batch_correct` *Batch correction*

Description

Batch correction

Usage

```
batch_correct(output_name, input_name, order_col, batch_col, qc_col, qc_label)
```

Arguments

`output_name`	The name of the output target.
`input_name`	The name of the input data.
`order_col`	The order column.
`batch_col`	The batch column.
`qc_col`	The QC column.
`qc_label`	The QC label.

Value

A target to perform batch correction.

See Also

`warper_batch_correction()`

Examples

```
batch_correct(BatchCorrected_experiment, input_experiment, order_col = "Order", batch_col = "Batch", qc_col =
```

batch_plot *Generate a batch plot*

Description

This function generates a batch plot showing the relationship between a feature and run order, stratified by batches and quality control (QC) samples.

Usage

```
batch_plot(
  dataset_experiment,
  order_col,
  batch_col,
  qc_col,
  qc_label,
  colour_by_col,
  feature_to_plot,
  ylab = "Peak area",
  title = "Feature vs run_order"
)
```

Arguments

`dataset_experiment`	The dataset for which the batch plot will be generated.
`order_col`	The column representing run order.
`batch_col`	The column representing batches.
`qc_col`	The column representing QC samples.
`qc_label`	The label for QC samples.
`colour_by_col`	The column used for coloring in the plot.
`feature_to_plot`	The feature to be plotted.
`ylab`	The label for the y-axis.
`title`	The title of the plot.

Value

A ggplot object displaying the batch plot.

Examples

```
batch_plot(dataset_experiment = my_dataset, order_col = "order", batch_col = "batch", qc_col = "qc", qc_label =
```

`ba_plot` *Generate a before-after plot*

Description

This function generates a before-after plot comparing distributions before and after a certain process or treatment.

Usage

```
ba_plot(DE_before, DE_after, factor_name = "sample_type")
```

Arguments

`DE_before`	The dataset before the process or treatment.
`DE_after`	The dataset after the process or treatment.
`factor_name`	The name of the factor variable for stratification.

Value

A ggplot object displaying the before-after plot.

Examples

```
ba_plot(DE_before = before_data, DE_after = after_data, factor_name = "sample_type")
```

`create_experiment` *Create DatasetExperiment object*

Description

Create DatasetExperiment object

Usage

```
create_experiment(
  output_name,
  data,
  experiment_name = "Name",
  experiment_description = "Description"
)
```

Arguments

`output_name`	The name of the output target.
`data`	The target name of the data with the data as data frames.
`experiment_name`	The name of the experiment (default is "Name").
`experiment_description`	The description of the experiment (default is "Description").

Value

A target to create a DatasetExperiment object.

See Also

`warper_createExperiment()`

Examples

```
create_experiment(experiment, data, experiment_name = "Metabolomic Assay for nutrition", experiment_descripti
```

`create_pipeline` *Create Pipeline Function*

Description

This function generates the code for a targets pipeline in an _targets.R file and saves it to the working directory.

Usage

```
create_pipeline()
```

Value

Nothing is returned. The function creates an _targets.R file in the specified directory.

Examples

```
create_pipeline()
```

`data.extract` *Function to extract data matrix from a DatasetExperiment object*

Description

This function extracts the data matrix from a SummarizedExperiment object.

Usage

```
data.extract(dataset_exp)
```

Arguments

`dataset_exp`	A DatasetExperiment object.

Value

Data matrix.

Examples

```
data.extract(dataset_exp)
```

`data.modify` *Function to modify data matrix of a DatasetExperiment object*

Description

This function replaces the data matrix in a SummarizedExperiment object with new data.

Usage

```
data.modify(dataset_exp, data)
```

Arguments

`dataset_exp`	A DatasetExperiment object.
`data`	New data matrix.

Value

A DatasetExperiment object with modified data matrix.

Examples

```
data.modify(dataset_exp, data)
```

`distribution_boxplot` *Generate a distribution boxplot*

Description

This function generates a distribution boxplot for a specified factor in the dataset.

Usage

```
distribution_boxplot(dataset_experiment, factor_name, per_class = FALSE)
```

Arguments

`dataset_experiment`	The dataset for which the boxplot will be generated.
`factor_name`	The name of the factor variable.
`per_class`	Logical indicating whether to generate separate boxplots for each class of the factor.

Value

A ggplot object displaying the distribution boxplot.

Examples

```
distribution_boxplot(dataset_experiment = my_dataset, factor_name = "factor", per_class = TRUE)
```

`export_data` *Export Data*

Description

Exports a dataset to a specified directory with a given name.

Usage

```
export_data(dataset_exp, out_dir, out_name)

export_data(dataset_exp, out_dir, out_name)
```

Arguments

`dataset_exp`	A `DatasetExperiment` object
`out_dir`	The output directory
`out_name`	The output name of the files
`output_name`	The name of the exported dataset.
`input_name`	The name of the input dataset to be exported.

Value

A list containing the target for exporting the dataset.

Nothing

Examples

```
export_data("exported_dataset.csv", my_dataset, "output_directory/")
export_data(dataset_exp, out_dir, out_name)
```

`extract_names` *Function to extract names*

Description

Function to extract names

Usage

```
extract_names(data)
```

Arguments

`data`

Value

The variableMetadata dataset for the DatasetExperiment object

Examples

```
extract_names(data)
```

`factorize_cols` *Factorize columns in sample metadata*

Description

Factorize columns in sample metadata

Usage

```
factorize_cols(output_name, input_name, cols)
```

Arguments

`output_name`	The name of the output target.
`input_name`	The name of the input data.
`cols`	The columns to factorize.

Value

A target to factorize columns in sample metadata.

See Also

`warper_factor_sample_col()`

Examples

```
factorize_cols(factorized_experiment, input_name = experiment, cols = c("Col1", "Col2"))
```

`filter_blanks` *Filter Blanks*

Description

Filter blanks from the dataset experiment.

Usage

```
filter_blanks(
  dataset_experiment,
  fold_change = 20,
  blank_label = "blank",
  qc_label = "QC",
  factor_name = "sample_type",
  fraction_in_blank = 0
)
```

Arguments

`dataset_experiment`	The dataset experiment object.
`fold_change`	The fold change threshold for blank filtering.
`blank_label`	The label for blanks.
`qc_label`	The label for quality control samples.
`factor_name`	The factor column name.
`fraction_in_blank`	The fraction of values in blank (default is 0).

Value

A dataset experiment object with blanks filtered out.

Examples

```
filtered_data <- filter_blanks(dataset_experiment, fold_change = 20, blank_label = "blank", qc_label = "QC", fa
```

---

filter_MV *Filter Missing values*

Description

Filter the missing values in a dataset experiment.

Usage

```
filter_MV(dataset_exp, threshold = 0.8)
```

Arguments

`dataset_exp`	The dataset experiment object to filter missing values from.
`threshold`	The threshold for filtering missing values (default is 0.8).

Value

A dataset experiment object with missing values filtered out.

Examples

```
filtered_data <- filter_MV(dataset_exp, threshold = 0.8)
```

`filter_outliers` *Filter Outliers*

Description

Filter outliers from the dataset experiment using a Hotelling's T2 distribution ellipse.

Usage

```
filter_outliers(dataset_experiment, nPCs = 5, conf.limit = c("0.95", "0.99"))
```

Arguments

`dataset_experiment`	The dataset experiment object.
`nPCs`	The number of principal components for PCA.
`conf.limit`	The confidence limit for outlier detection. Either 0.95 or 0.99.

Value

A dataset experiment object with outliers filtered out.

Examples

```
filtered_data <- filter_outliers(dataset_experiment, nPCs = 5, conf.limit = "0.95")
```

`filter_step` *Filter data by missing value threshold*

Description

Filter data by missing value threshold

Usage

```
filter_step(
  output_name,
  input_name,
  threshold,
  filter_outliers = TRUE,
  conf.limit = "0.95",
  out_dir
)
```

Arguments

`output_name`	The name of the output target.
`input_name`	The name of the input data.
`threshold`	The threshold for missing values.
`filter_outliers`	Logical indicating whether to filter outliers (default is TRUE).
`conf.limit`	Confidence limit for outlier detection (default is "0.95").
`out_dir`	The directory to save plots (optional).

Value

A list of targets to filter data by missing value threshold.

See Also

`filter_MV()`, `filter_outliers()`, `zero_to_na()`, `missing_values_plot()`, `plot_outliers()`

Examples

```
filter_step(filtered_experiment, input_experiment, threshold = 0.2, filter_outliers = TRUE, conf.limit = "0.95
```

`impute` *Impute missing values*

Description

Impute missing values

Usage

```
impute(output_name, input_name, method, k = 5)
```

Arguments

`output_name`	The name of the output target.
`input_name`	The name of the input data.
`method`	The imputation method.
`k`	The number of neighbors for KNN imputation (default is 5).

Value

A target to impute missing values.

See Also

`impute_warper()`

Examples

```
impute(imputed_experiment, input_experiment, method = "knn", k = 3)
```

`impute_bpca` *Impute BPCA*

Description

Impute missing values in a dataset experiment using BPCA.

Usage

```
impute_bpca(dataset_experiment, nPCs = k, ...)
```

Arguments

`dataset_experiment`	The dataset experiment object.
`nPCs`	The number of principal components for BPCA.

Value

The dataset experiment object with missing values imputed using BPCA.

Examples

```
DE <- ST000284
imputed <- impute_bpca(DE, nPCs = 5)
summary(imputed)
```

`impute_kNN` *Impute kNN*

Description

Impute missing values in a dataset experiment using kNN.

Usage

```
impute_kNN(dataset_experiment, k = k)
```

Arguments

`dataset_experiment`	The dataset experiment object.
`k`	The number of neighbors for kNN imputation.

Value

The dataset experiment object with missing values imputed using kNN.

Examples

```
DE <- ST000284
imputed <- impute_kNN(DE, k = 5)
summary(imputed)
```

impute_mean *Impute Mean*

Description

Impute missing values in a dataset experiment using the mean.

Usage

```
impute_mean(dataset_experiment)
```

Arguments

dataset_experiment	The dataset experiment object.

Value

The dataset experiment object with missing values imputed using the mean.

Examples

```
DE <- ST000284
imputed <- impute_mean(DE)
summary(imputed)
```

impute_median *Impute Median*

Description

Impute missing values in a dataset experiment using the median.

Usage

```
impute_median(dataset_experiment)
```

Arguments

dataset_experiment	The dataset experiment object.

Value

The dataset experiment object with missing values imputed using the median.

Examples

```
DE <- ST000284
imputed <- impute_median(DE)
summary(imputed)
```

impute_ppca *Impute PPCA*

Description

Impute missing values in a dataset experiment using PPCA.

Usage

```
impute_ppca(dataset_experiment, nPCs = k, ...)
```

Arguments

`dataset_experiment`	The dataset experiment object.
`nPCs`	The number of principal components for PPCA.

Value

The dataset experiment object with missing values imputed using PPCA.

Examples

```
DE <- ST000284
imputed <- impute_ppca(DE, nPCs = 5)
summary(imputed)
```

impute_QRILC *Impute QRILC*

Description

Impute missing values in a dataset experiment using QRILC.

Usage

```
impute_QRILC(dataset_experiment)
```

Arguments

`dataset_experiment`	The dataset experiment object.

Value

The dataset experiment object with missing values imputed using QRILC.

Examples

```
DE <- ST000284
imputed <- impute_QRILC(DE)
summary(imputed)
```

impute_RF *Impute Random Forest*

Description

Impute missing values in a dataset experiment using random forest.

Usage

```
impute_RF(dataset_experiment)
```

Arguments

dataset_experiment	The dataset experiment object.

Value

The dataset experiment object with missing values imputed using random forest.

Examples

```
DE <- ST000284
imputed <- impute_RF(DE)
summary(imputed)
```

impute_SVD *Impute SVD*

Description

Impute missing values in a dataset experiment using SVD.

Usage

```
impute_SVD(dataset_experiment, nPCs = k, center = TRUE, ...)
```

Arguments

dataset_experiment	The dataset experiment object.
nPCs	The number of principal components for SVD.

Value

The dataset experiment object with missing values imputed using SVD.

Examples

```
DE <- ST000284
imputed <- impute_SVD(DE, nPCs = 5)
summary(imputed)
```

`impute_warper` *Impute Missing Values*

Description

Impute missing values in a dataset experiment using various methods.

Usage

```
impute_warper(dataset_experiment, method, k = 5)
```

Arguments

`dataset_experiment`	The dataset experiment object.
`method`	The imputation method to use.
`k`	The parameter for some imputation methods (default: 5).

Value

The dataset experiment object with missing values imputed.

Examples

```
DE <- ST000284
imputed <- impute_warper(DE, method = "mean")
summary(imputed)
```

`load_data` *Load data from files into data frames*

Description

Load data from files into data frames

Usage

```
load_data(
  output_name,
  dataMatrixFile,
  sampleMetadataFile,
  variableMetadataFile = NULL,
  dataSep = ",",
  sampleSep = ",",
  variableSep = ","
)
```

Arguments

`output_name`	The name of the output target.
`dataMatrixFile`	Path to the data matrix file.
`sampleMetadataFile`	Path to the sample metadata file.
`variableMetadataFile`	Path to the variable metadata file (optional).
`dataSep`	The separator used in the dataMatrixfile (default is ",").
`sampleSep`	The separator used in the sampleMetadataFile (default is ",").
`variableSep`	The separator used in the variableMetadataFile (default is ",").

Value

A list of targets to load and read data matrix and sample metadata.

Examples

```
load_data(data_loaded, "Data/data.csv", "Data/metadata.csv", "Data/variable_metadata.csv", separator = ",")
```

`MetaboAnalyst_load_data` *Function to load the previously saved MetaboAnalystData.csv data into MetaboAnalyst from the TempData directory*

Description

Function to load the previously saved MetaboAnalystData.csv data into MetaboAnalyst from the TempData directory

Usage

```
MetaboAnalyst_load_data()
```

Value

MetaboAnalyst data object (mSet).

Examples

```
MetaboAnalyst_load_data()
```

metaboNorm — *Function to normalize MetaboAnalyst data*

Description

This function performs row-wise normalization, transformation, and scaling of the metabolomic data.

Usage

```
metaboNorm(
  mSet,
  rowNorm = "NULL",
  transNorm = "NULL",
  scaleNorm = "NULL",
  ref = NULL,
  ratio = FALSE,
  ratioNum = 20,
  out_dir
)
```

Arguments

`mSet`	The MetaboAnalyst data object.
`rowNorm`	The row normalization method.
`transNorm`	The transformation normalization method.
`scaleNorm`	The scaling normalization method.
`ref`	Input the name of the reference sample or the reference feature, use " " around the name.
`ratio`	This option is only for biomarker analysis.
`ratioNum`	Relevant only for biomarker analysis.
`out_dir`	The output directory for the plots.

Value

The normalized MetaboAnalyst data object.

Examples

```
metaboNorm(mSet, rowNorm = "NULL", transNorm = "NULL", scaleNorm = "NULL", ref = NULL, ratio = FALSE, ratioNum =
```

`missing_values_plot` *Generate a missing values plot*

Description

This function generates a missing values plot using the VIM package's aggr function and saves the plot to a specified directory with a given filename.

Usage

```
missing_values_plot(dataset_experiment, out_dir, out_name)
```

Arguments

`dataset_experiment`	The dataset for which the missing values plot will be generated.
`out_dir`	The directory where the plot will be saved.
`out_name`	The filename for the saved plot.

Examples

```
missing_values_plot(dataset_experiment = my_dataset, out_dir = "output", out_name = "missing_plot.png")
```

`normalize` *Normalize data*

Description

Normalize data

Usage

```
normalize(
  output_name,
  input_name,
  factor_col,
  sample_id_col,
  rowNorm = "NULL",
  transNorm = "NULL",
  scaleNorm = "NULL",
  ref = NULL,
  out_dir
)
```

Arguments

`output_name`	The name of the output target.
`input_name`	The name of the input data.
`factor_col`	The factor column.
`sample_id_col`	The sample ID column.
`rowNorm`	The row normalization method (optional). One of: `"QuantileNorm"`, `"CompNorm"`, `"SumNorm"`, `"MedianNorm"`, `"SpecNorm"`, or `NULL`.
`transNorm`	The transformation normalization method (optional). One of: `"LogNorm"`, `"CrNorm"`, or `NULL`.
`scaleNorm`	The scaling normalization method (optional). One of: `"MeanCenter"`, `"AutoNorm"`, `"ParetoNorm"`, `"RangeNorm"`, or `NULL`.
`ref`	The reference group for normalization (optional).
`out_dir`	The directory to save plots (optional).

Value

A list of targets to normalize data.

See Also

`normalize_metab()`

Examples

```
normalize(normalized_data, input_data, factor_col = "Group", sample_id_col = "Sample", rowNorm = "CompNorm", t
```

`normalize_csn` *Perform Constant Sum Normalization (CSN)*

Description

Perform Constant Sum Normalization (CSN)

Usage

```
normalize_csn(dataset_experiment, scaling_factor = 1)
```

Arguments

`dataset_experiment`	A `DatasetExperiment` object
`scaling_factor`	Scaling factor for normalization

Value

Normalized A `DatasetExperiment` object

Examples

```
normalize_csn(dataset_experiment, scaling_factor = 1)
```

normalize_metab *Normalize A* `DatasetExperiment` *object using MetaboAnalystR*

Description

Normalize A `DatasetExperiment` object using MetaboAnalystR

Usage

```
normalize_metab(
  dataset_experiment,
  factor_col,
  sample_id_col,
  rowNorm = "NULL",
  transNorm = "NULL",
  scaleNorm = "NULL",
  ref = NULL,
  ratio = FALSE,
  ratioNum = 20,
  out_dir
)
```

Arguments

`dataset_experiment`	A `DatasetExperiment` object
`factor_col`	Column containing factor information for normalization
`sample_id_col`	Column containing sample IDs
`rowNorm`	Type of row normalization (options: "QuantileNorm", "CompNorm", "SumNorm", "MedianNorm", "SpecNorm", or "NULL")
`transNorm`	Type of transformation normalization (options: "LogNorm", "CrNorm", or "NULL")
`scaleNorm`	Type of scaling normalization (options: "MeanCenter", "AutoNorm", "ParetoNorm", "RangeNorm", or "NULL")
`ref`	Reference feature for 'CompNorm' normalization
`ratio`	Boolean indicating whether to apply ratio normalization
`ratioNum`	Number of samples for ratio normalization
`out_dir`	Output directory for saving files

Value

Normalized A `DatasetExperiment` object

Examples

```
normalize_metab(dataset_experiment, factor_col, sample_id_col, rowNorm = NULL, transNorm = NULL, scaleNorm = N
```

`normalize_pqn` *Perform Probabilistic Quotient normalization (PQN)*

Description

Perform Probabilistic Quotient normalization (PQN)

Usage

```
normalize_pqn(dataset_experiment, qc_label, factor_name)
```

Arguments

`dataset_experiment`	A `DatasetExperiment` object
`qc_label`	Label for quality control samples
`factor_name`	Name of the factor to use for normalization

Value

Normalized A `DatasetExperiment` object

Examples

```
normalize_pqn(dataset_experiment, qc_label, factor_name)
```

`normalize_vln` *Perform Vector Length Normalization (VLN)*

Description

Perform Vector Length Normalization (VLN)

Usage

```
normalize_vln(dataset_experiment)
```

Arguments

`dataset_experiment`	A `DatasetExperiment` object

Value

Normalized A `DatasetExperiment` object

Examples

```
normalize_vln(dataset_experiment)
```

pipePliers *Run Shiny App*

Description

This function launches the Shiny app included with the package.

Usage

```
pipePliers()
```

Examples

```
pipePliers()
```

plot_boxplots *Plot boxplots for multiple columns*

Description

This function generates vertical boxplots for multiple columns of a dataset.

Usage

```
plot_boxplots(data, title = "Boxplot of Columns")
```

Arguments

data	The dataset containing the columns to be plotted.
title	The title of the plot.

Value

A ggplot object displaying the boxplots.

Examples

```
plot_boxplots(data = my_data, title = "Boxplot of Columns")
```

`plot_density_single_with_legend` *Plot density plots for one variable with a legend*

Description

This function generates density plots for one variable, comparing the original and imputed data, with a legend indicating the data source.

Usage

```
plot_density_single_with_legend(original_var, imputed_var)
```

Arguments

`original_var`	The original variable data.
`imputed_var`	The imputed variable data.

Value

A ggplot object displaying the density plots.

Examples

```
plot_density_single_with_legend(original_var = data$original_var, imputed_var = data$imputed_var)
```

`plot_heatmap` *Plot a heatmap of the data*

Description

This function generates a heatmap of the provided dataset.

Usage

```
plot_heatmap(dataset_experiment, na_colour = "#FF00E4")
```

Arguments

`dataset_experiment`	The dataset for which the heatmap will be generated.
`na_colour`	The color to represent missing values in the heatmap.

Value

A ggplot object displaying the heatmap.

Examples

```
plot_heatmap(dataset_experiment = my_dataset, na_colour = "#FF00E4")
```

`plot_hotelling_obs` *Generate a PCA Hotelling's T-squared observations plot*

Description

This function generates a PCA Hotelling's T-squared observations plot showing the T-squared values for each observation.

Usage

```
plot_hotelling_obs(dataset_experiment, nPCs = 5, nPCs_to_plot = 2)
```

Arguments

`dataset_experiment`	The dataset for which the plot will be generated.
`nPCs`	The number of principal components to include in the analysis.
`nPCs_to_plot`	The number of principal components to plot the ellipses for.

Value

A ggplot object displaying the PCA Hotelling's T-squared observations plot.

Examples

```
plot_hotelling_obs(dataset_experiment = my_dataset, nPCs = 5, nPCs_to_plot = 2)
```

`plot_hotelling_pca` *Generate a PCA Hotelling's T-squared plot*

Description

This function generates a PCA Hotelling's T-squared plot showing the principal component scores and confidence ellipses.

Usage

```
plot_hotelling_pca(dataset_experiment, nPCs = 5)
```

Arguments

`dataset_experiment`	The dataset for which the plot will be generated.
`nPCs`	The number of principal components to include in the analysis.

Value

A ggplot object displaying the PCA Hotelling's T-squared plot.

Examples

```
plot_hotelling_pca(dataset_experiment = my_dataset, nPCs = 5)
```

`plot_outliers` *Generate an outliers plot*

Description

This function generates a plot showing outliers detected using Hotelling's T-squared statistic in PCA.

Usage

```
plot_outliers(dataset_experiment, nPCs = 5, out_dir, out_name)
```

Arguments

`dataset_experiment`	The dataset for which the outliers plot will be generated.
`nPCs`	The number of principal components to include in the analysis.
`out_dir`	The directory where the plot will be saved.
`out_name`	The filename for the saved plot.

Examples

```
plot_outliers(dataset_experiment = my_dataset, nPCs = 5, out_dir = "output", out_name = "outliers_plot.png")
```

`plot_pca` *Generate a PCA plot*

Description

This function generates a PCA plot showing the principal component scores colored by a specified factor.

Usage

```
plot_pca(dataset_experiment, factor_name = "sample_type", nPCs = 5)
```

Arguments

`dataset_experiment`	The dataset for which the PCA plot will be generated.
`factor_name`	The name of the factor variable used for coloring.
`nPCs`	The number of principal components to include in the analysis.

Value

A ggplot object displaying the PCA plot.

Examples

```
plot_pca(dataset_experiment = my_dataset, factor_name = "sample_type", nPCs = 5)
```

`sample.data.extract` *Function to extract sample metadata from a DatasetExperiment object*

Description

This function extracts the sample metadata from a SummarizedExperiment object.

Usage

```
sample.data.extract(dataset_exp)
```

Arguments

`dataset_exp`	A DatasetExperiment object.

Value

Sample metadata dataframe.

Examples

```
sample.data.extract(dataset_exp)
```

`sample.data.modify` *Function to modify sample metadata of a DatasetExperiment object*

Description

This function replaces the sample metadata in a SummarizedExperiment object with new metadata.

Usage

```
sample.data.modify(dataset_exp, sample_meta)
```

Arguments

`dataset_exp`	A DatasetExperiment object.
`sample_meta`	New sample metadata dataframe.

Value

A DatasetExperiment object with modified sample metadata.

Examples

```
sample.data.modify(dataset_exp, sample_meta)
```

`save_metabo` *Function to export MetaboAnalyst data*

Description

Function to export MetaboAnalyst data

Usage

```
save_metabo(mSet)
```

Arguments

`mSet`	The MetaboAnalyst data object

Value

Nothing

Examples

```
save_metabo(mSet)
```

`save_plot` *Function to save plots*

Description

Function to save plots

Usage

```
save_plot(plt, output_dir, output_name)
```

Arguments

`plt`	The plot object
`output_dir`	The output directory
`output_name`	The output name for the plot file

Value

Nothing

Examples

```
save_plot(plt, output_dir, output_name)
```

`sort_by_sample_id` *Sort by sample_id*

Description

Sort by sample_id

Usage

```
sort_by_sample_id(df)
```

Arguments

`df`	A dataframe with a sample_id column.

Value

A data frame sorted by sample_id.

Examples

```
sort_by_sample_id(data)
```

`ST000284` *ST000284 Dataset*

Description

This dataset is used as an example in the package.

Usage

```
ST000284
```

Format

A `DatasetExperiment` object

Source

https://www.metabolomicsworkbench.org/data/DRCCMetadata.php?Mode=Study&StudyID=ST000284&StudyType=M

`toMetaboAnalyst` *Function to create a dataSet for MetaboAnalyst*

Description

Function to create a dataSet for MetaboAnalyst

Usage

```
toMetaboAnalyst(
  dataset_exp,
  class_col = "sample_type",
  sample_id = "sample_id"
)
```

Arguments

`dataset_exp`	A DatasetExperiment object.
`class_col`	Column to be used as class.
`sample_id`	Column to be used as sample ID.

Value

Nothing.

Examples

```
toMetaboAnalyst(dataset_exp, class_col = "sample_type", sample_id = "sample_id")
```

`variable.data.extract` *Function to extract variable metadata from a DatasetExperiment object*

Description

This function extracts the variable metadata from a SummarizedExperiment object.

Usage

```
variable.data.extract(dataset_exp)
```

Arguments

`dataset_exp`	A DatasetExperiment object.

Value

Variable metadata dataframe.

Examples

```
variable.data.extract(dataset_exp)
```

`variable.data.modify` *Function to modify variable metadata of a DatasetExperiment object*

Description

This function replaces the variable metadata in a SummarizedExperiment object with new metadata.

Usage

```
variable.data.modify(dataset_exp, variable_meta)
```

Arguments

`dataset_exp`	A DatasetExperiment object.
`variable_meta`	New variable metadata.

Value

A DatasetExperiment object with modified variable metadata.

Examples

```
variable.data.modify(dataset_exp, variable_meta)
```

`warper_batch_correction` *Signal drift and batch correction function*

Description

This function performs signal drift and batch correction on a given DatasetExperiment object using the QC-RSC method.

Usage

```
warper_batch_correction(dataset_exp, order_col, batch_col, qc_col, qc_label)
```

Arguments

`dataset_exp`	A DatasetExperiment object with samples and variables.
`order_col`	Column indicating the order of samples.
`batch_col`	Column indicating batch information.
`qc_col`	Column indicating quality control information.
`qc_label`	Label for quality control.

Value

Corrected DatasetExperiment object.

Examples

```
warper_batch_correction(dataset_exp, order_col, batch_col, qc_col, qc_label)
```

warper_createExperiment

Process the dataset to create the DatasetExperiment object

Description

Process the dataset to create the DatasetExperiment object

Usage

```
warper_createExperiment(
  dataMatrix,
  sampleMetadata,
  variableMetadata,
  experiment_name = "Name",
  experiment_description = "Description"
)
```

Arguments

`dataMatrix`	A matrix with samples as rows and features as columns.
`sampleMetadata`	A data frame with the sample metadata.
`variableMetadata`	A data frame with the variable metadata.
`experiment_name`	The name for the experiment.
`experiment_description`	The description for the experiment.

Value

A `DatasetExperiment` object.

Examples

```
warper_createExperiment(dataMatrix, sampleMetadata, variableMetadata, experiment_name = "Name", experiment_de
```

warper_factor_sample_col

Function to convert sample columns to factors

Description

This function converts specified columns in the sample metadata to factors.

Usage

```
warper_factor_sample_col(dataset_exp, col)
```

Arguments

`dataset_exp`	A DatasetExperiment object with sample metadata.
`col`	Column(s) to be converted to factors.

Value

A DatasetExperiment object with specified columns converted to factors.

Examples

```
warper_factor_sample_col(dataset_exp, col)
```

`zero_to_na` *Make 0 as NA*

Description

Replace 0 values with NA in a dataset experiment.

Usage

```
zero_to_na(dataset_exp)
```

Arguments

`dataset_exp`	The dataset experiment object.

Value

A dataset experiment object with 0 values replaced by NA.

Examples

```
modified_dataset <- zero_to_na(dataset_exp)
```

Index