Package: omiprep (via r-universe)

May 12, 2026

Title Omics data preparation and processing pipeline

Version 1.0.0

Description | Reads in raw Metabolon, Nightingale Health, Olink, and SomaLogic xls sheets, or flat text files and aids in data preparation of all metabolomics & proteomics data sets. Formerly known as metaboprep.

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 8.0.0

Imports boot, car, cli, ggplot2, ggpubr, nFactors, pcaMethods, irlba, psych, pwr, readxl, rmarkdown, S7, yaml, OlinkAnalyze, SomaDataIO, reshape2, openxlsx, stringr, shiny, shinycssloaders, bslib, DT, plotly

Suggests knitr, dendextend, kableExtra, remotes, testthat (>= 3.0.0)

Remotes hredestig/pcaMethods

URL https://mrcieu.github.io/omiprep/

Collate 'class_omiprep.R' 'batch_normalise.R' 'continous_power_plot.R' 'cramer_v.R' 'export.R' 'feature_describe.R' 'feature_skewness.R' 'feature_summary.R' 'generate_report.R' 'imbalanced_power_plot.R' 'missingness.R' 'multivariate_anova.R' 'outlier_detection.R' 'pc_and_outliers.R' 'quality_control.R' 'read_metabolon.R' 'read_nightingale.R' 'read_olink.R' 'read_somalogic.R' 'run_metaboprep1.R' 'sample_summary.R' 'shiny_app.R' 'summarise.R' 'summary_omiprep.R' 'total_sum_abundance.R' 'tree_and_independent_features.R' 'utils.R' 'variable_by_factor.R' 'zzz.R'

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libssl-dev xz-utils zlib1g-dev

Repository https://mrcieu.r-universe.dev

Date/Publication 2026-05-12 08:49:13 UTC

RemoteUrl https://github.com/MRCIEU/omiprep

RemoteRef HEAD

RemoteSha 6994838d7f30353b10f9303812ae5c620b024344

Contents

`add_layer` *Add a Layer of Data (internal use)*

Description

This function adds an additional layer of data along the third dimension to an existing 3D array (or 2D matrix/vector) by stacking a new layer of data. It ensures that the dimensions of the new layer match the first two dimensions of the existing array or matrix. If there is a mismatch in row or column names and the 'force' parameter is set to 'TRUE', the function will align the data by filling missing values with 'NA'. It is used internally and not intended for routine user use.

Usage

```
add_layer(current, layer, layer_name, force = FALSE)
```

Arguments

`current`	A vector, matrix, or 3D array representing the current stack of data.
`layer`	A matrix or array that represents the new layer of data to be added. It should match the dimensions of the first two dimensions of 'current'.
`layer_name`	A character string specifying the name of the new dimension for the 3rd axis. This can be used to annotate the new data layer.
`force`	A logical value indicating whether to force the join and create 'NA' values where row or column names do not match between 'current' and 'layer'. Default is 'FALSE'.

Value

A 3D array with the added layer in the third dimension.

`available_data_formats`
List Available Data Formats

Description

Scans the package source files for functions starting with "read_" to determine supported data formats.

Usage

```
available_data_formats()
```

Value

A named character vector of available data formats.

`available_report_templates` *List Available Report Templates*

Description

Scans the package source files for available report templates to write to.

Usage

```
available_report_templates()
```

Value

A character vector of available report templates

`batch_normalise` *Batch Normalisation*

Description

Run batch normalisation based on the platform flag in the features data

Usage

```
batch_normalise(
  omiprep,
  run_mode_col,
  run_mode_colmap,
  source_layer = "input",
  dest_layer = "batch_normalised"
)

## S7 method for class <omiprep::Omiprep>
batch_normalise(
  omiprep,
  run_mode_col,
  run_mode_colmap,
  source_layer = "input",
  dest_layer = "batch_normalised"
)
```

Arguments

omiprep	an object of class Omiprep
run_mode_col	character, column name in features data containing the run mode
run_mode_colmap	named character vector or list, c(mode = "mode col name in samples")
source_layer	character, which data layer to get the data from
dest_layer	character, which data layer to put the the data in to

---

class_omiprep *Omiprep Object*

Description

A 'Omiprep' object is a container for matrices of 'omics data, along with associated metadata. It allows for efficient storage and manipulation of data, supporting quality control, transformations, and various analyses. This object facilitates easy access to data layers, sample and feature summaries, outlier treatment, and more.

Usage

```
Omiprep(
  data,
  samples,
  features,
  exclusions = list(samples = list(user_excluded = character(),
    extreme_sample_missingness = character(), user_defined_sample_missingness =
    character(), user_defined_sample_totalpeakarea = character(),
   user_defined_sample_pca_outlier = character()), features = list(user_excluded =
    character(), extreme_feature_missingness = character(),
   user_defined_feature_missingness = character(), user_defined_feature_skewness =
    character())),
  feature_summary = array(data = NA_real_, dim = c(0, 0, 0)),
  sample_summary = array(data = NA_real_, dim = c(0, 0, 0))
)
```

Arguments

data	numeric matrix, the data matrix containing 'omics values (not to be set directly).
samples	data.frame, a data frame containing sample-related information (not to be set directly).
features	data.frame, a data frame containing feature-related information (not to be set directly).
exclusions	list, holds exclusion codes for data masking (not to be set directly).
feature_summary	numeric matrix, summary statistics for features (not to be set directly).
sample_summary	numeric matrix, summary statistics for samples (not to be set directly).

Value

An object of class Omiprep, an S7 class.

Slots

`data` numeric matrix, the 'omics data.

`samples` data.frame, the samples data frame

`features` data.frame, the features data frame

`exclusions` list, exclusion codes (mask for data).

`feature_summary` numeric matrix, feature summary statistics.

`sample_summary` numeric matrix, sample summary statistics.

---

clean_names *Standardize Column or Feature Names*

Description

Cleans a character vector of names by replacing spaces with underscores, removing special characters (‘-‘, ‘.‘), replacing ‘ and converting to lowercase.

Usage

```
clean_names(names)
```

Arguments

`names`	‘character vector‘ A vector of names to be cleaned.

Value

‘character vector‘ A standardized version of the input names.

Examples

```
clean_names(c("Sample ID", "Feature-Name.1", "Concentration %"))
# Returns: c("sample_id", "featurename1", "concentration_pct")
```

continuous_power_plot *continuous trait power analysis plot*

Description

This function (1) identifies an informative distribution of effect and power estimates given your datas total sample size and (2) returns a summary plot.

Usage

```
continuous_power_plot(mydata)
```

Arguments

mydata	Your 'omics data matrix, with samples in rows

Value

a ggplot2 object

Examples

```
ex_data = matrix(NA, 1000, 2)
continuous_power_plot( ex_data )
```

cramerV *Cramer's V (phi)*

Description

Calculates Cramer's V for a table of nominal variables; confidence intervals by bootstrap. Function taken from the rcompanion Rpackage.

Usage

```
cramerV(
  x,
  y = NULL,
  ci = FALSE,
  conf = 0.95,
  type = "perc",
  R = 1000,
  digits = 4,
  bias.correct = FALSE,
  reportIncomplete = FALSE,
```

```
  verbose = FALSE,
  ...
)
```

Arguments

x	Either a two-way table or a two-way matrix. Can also be a vector of observations for one dimension of a two-way table.
y	If x is a vector, y is the vector of observations for the second dimension of a two-way table.
ci	If TRUE, returns confidence intervals by bootstrap. May be slow.
conf	The level for the confidence interval.
type	The type of confidence interval to use. Can be any of "norm", "basic", "perc", or "bca". Passed to boot.ci.
R	The number of replications to use for bootstrap.
digits	The number of significant digits in the output.
bias.correct	If TRUE, a bias correction is applied.
reportIncomplete	If FALSE (the default), NA will be reported in cases where there are instances of the calculation of the statistic failing during the bootstrap procedure.
verbose	If TRUE, prints additional statistics.
...	Additional arguments passed to chisq.test.

Details

Cramer's V is used as a measure of association between two nominal variables, or as an effect size for a chi-square test of association. For a 2 x 2 table, the absolute value of the phi statistic is the same as Cramer's V.

Because V is always positive, if type="perc", the confidence interval will never cross zero. In this case, the confidence interval range should not be used for statistical inference. However, if type="norm", the confidence interval may cross zero.

When V is close to 0 or very large, or with small counts, the confidence intervals determined by this method may not be reliable, or the procedure may fail.

Value

A single statistic, Cramer's V. Or a small data frame consisting of Cramer's V, and the lower and upper confidence limits.

Author(s)

Salvatore Mangiafico, <mangiafico@njaes.rutgers.edu>

References

http://rcompanion.org/handbook/H_10.html

`eval.power.binary.imbalanced` *Estimate power for a binary variable in an imbalanced design*

Description

This function allows you estimate power for a binary variable given a defined number of case samples, control samples, effect size, and significance threshold.

Usage

```
eval.power.binary.imbalanced(N_case, N_control, effect, alpha)
```

Arguments

`N_case`	a numeric vector of sample size of cases
`N_control`	a numeric vector of sample size of controls
`effect`	a numeric vector of effect size
`alpha`	a numeric vector of significance thresholds

Value

a matrix of paramater inputs and power estimates are returned as a matrix

Examples

```
eval.power.binary.imbalanced( N_case = 1000,
 N_control = 1000,
 effect = 0.01,
 alpha = 0.05 )

eval.power.binary.imbalanced( N_case = c(1000, 2000),
 N_control = c(1000, 2000),
 effect = 0.01,
 alpha = 0.05 )
```

eval.power.cont *estimate power for continuous variable*

Description

This function estimates power for a continuous variable given the sample size, effect size, significance threshold, and the degrees of freedom.

Usage

```
eval.power.cont(N, n_coeff, effect, alpha)
```

Arguments

N	Sample size
n_coeff	degrees of freedom for numerator
effect	effect size
alpha	significance level (Type 1 error)

Examples

```
eval.power.cont(N = 1000, n_coeff = 1, effect = 0.0025, alpha = 0.05)
```

export *Export Data from a Omiprep Object*

Description

Exports all data from a 'Omiprep' object to a structured directory format. For each data layer, the function creates a subdirectory containing: - the primary data matrix ('data.tsv'), - associated feature and sample metadata ('features.tsv', 'samples.tsv'), - feature and sample summaries (if present, 'feature_summary.tsv', 'sample_summary.tsv'), - a serialized feature tree (if present), - and a 'config.yml' file with additional metadata and processing parameters.

Usage

```
export(omiprep, directory, format = "omiprep", ...)

## S7 method for classes <omiprep::Omiprep>, <character>
export(omiprep, directory, format = "omiprep", ...)
```

Arguments

omiprep	A ‘Omiprep‘ object containing the data to be exported.
directory	character, string specifying the path to the directory where the data should be written.
format	character, string specifying the format of the exported data - one of "omiprep", "comets", or "metaboanalyst".
...	Arguments passed on to export_comets, export_metaboanalyst
	layer character, the name of the ‘omiprep@data‘ layer (3rd array dimension) to write out
	group_col character, the column name in the ‘omiprep@samples‘ data identifying the group for one-factor analysis

Value

the ‘Omiprep‘ object, invisibly, for use in pipes

---

export_comets *Export Data to ‘COMETS‘ format*

Description

Export Data to ‘COMETS‘ format

Usage

```
export_comets(omiprep, directory, layer = NULL)

## S7 method for classes <omiprep::Omiprep>, <character>
export_comets(omiprep, directory, layer = NULL)
```

Arguments

omiprep	A ‘Omiprep‘ object containing the data to be exported.
directory	character, string specifying the path to the directory where the data should be written.
layer	character, the name of the ‘omiprep@data‘ layer (3rd array dimension) to write out

Value

the ‘Omiprep‘ object, invisibly, for use in pipes

export_metaboanalyst *Export Data to 'MetaboAnalyst' format*

Description

Export Data to 'MetaboAnalyst' format

Usage

```
export_metaboanalyst(omiprep, directory, layer = NULL, group_col = NULL)

## S7 method for classes <omiprep::Omiprep>, <character>
export_metaboanalyst(omiprep, directory, layer = NULL, group_col = NULL)
```

Arguments

omiprep	A 'Omiprep' object containing the data to be exported.
directory	character, string specifying the path to the directory where the data should be written.
layer	character, the name of the 'omiprep@data' layer (3rd array dimension) to write out
group_col	character, the column name in the 'omiprep@samples' data identifying the group for one-factor analysis

Value

the 'Omiprep' object, invisibly, for use in pipes

export_omiprep *Export Data to 'Omiprep' format*

Description

Export Data to 'Omiprep' format

Usage

```
export_omiprep(omiprep, directory, ...)

## S7 method for classes <omiprep::Omiprep>, <character>
export_omiprep(omiprep, directory, ...)
```

Arguments

omiprep	A 'Omiprep' object containing the data to be exported.
directory	character, string specifying the path to the directory where the data should be written.
...	other parameters passed to `export_omiprep()`, `export_comets()`, or `export_metaboanalyst()`.

Value

the 'Omiprep' object, invisibly, for use in pipes

---

feature_describe *Summary Statistics for Features*

Description

This function allows you to 'describe' 'omics features using the describe() function from the psych package, as well as estimate variance, a dispersion index, the coeficent of variation, and shapiro's W-statistic. The output from `psych::describe()` includes feature-level skewness and kurtosis estimates.

Usage

```
feature_describe(data)
```

Arguments

data	matrix, the 'omics data matrix. Samples in row, features in columns

Value

a data frame of summary statistics for features (columns) of a matrix

---

feature_skewness *Estimate Feature Skewness*

Description

Estimate per-feature skewness and optionally flag features beyond a user-defined skewness threshold.

Usage

```
feature_skewness(data, threshold = NULL, direction = "left")
```

Arguments

data	matrix, a numeric matrix with samples in rows and features in columns.
threshold	numeric, optional skewness threshold. If ‘NULL‘, only skewness is returned and no exclusion flag is calculated.
direction	character, direction of skewness to flag. One of ‘"left"‘, ‘"right"‘, or ‘"both"‘.

Value

data.frame with columns ‘feature_id‘, ‘skew‘, and ‘exclude_by_skewness‘ (logical; ‘NA‘ if ‘threshold = NULL‘).

---

feature_summary *Feature Summary Statistics*

Description

This function estimates feature statistics for samples in a matrix of ’omics features.

Usage

```
feature_summary(
  omiprep,
  source_layer = "input",
  outlier_udist = 5,
  tree_cut_height = 0.5,
  feature_selection = "max_var_exp",
  sample_ids = NULL,
  feature_ids = NULL,
  features_exclude = NULL,
  output = "data.frame",
  cores = NULL,
  fast = FALSE
)

## S7 method for class <omiprep::Omiprep>
feature_summary(
  omiprep,
  source_layer = "input",
  outlier_udist = 5,
  tree_cut_height = 0.5,
  feature_selection = "max_var_exp",
  sample_ids = NULL,
  feature_ids = NULL,
  features_exclude = NULL,
  output = "data.frame",
  cores = NULL,
  fast = FALSE
)
```

Arguments

omiprep	an object of class Omiprep
source_layer	character, the data layer to summarise
outlier_udist	the unit distance in SD or IQR from the mean or median estimate, respectively outliers are identified at. Default value is 5.
tree_cut_height	numeric, the threshold for feature independence in hierarchical clustering. Default is 0.5.
feature_selection	character, either 'max_var_exp' or 'least_missingness', how to select the independent feature within clusters
sample_ids	character, vector of sample ids to work with
feature_ids	character, vector of feature ids to work with
features_exclude	character, vector of feature id indicating features to exclude from the sample and PCA summary analysis but keep in the data
output	character, type of output, either 'object' to return the updated metaboprep object, or 'data.frame' to return the data.
cores	number of cores available for parallelism; the default null will try find the maximum available cores - 1; set to 1 for linear, but potentially slow, computation of the correlation matrix.
fast	If TRUE, accelerates correlation computation by imputing missing values to the column minimum, pre-ranking all columns, and computing Pearson correlation on ranked data (approximating Spearman). Substantially faster than exact Spearman at large feature dimensions ($p > 5000$) but assumes missing data are missing at random. Features with high missingness will have inflated rank ties at the median (ensure these are filtered out appropriately with the missingness option). Default FALSE.

---

find.cont.effect.sizes.2.sim

identify continuos trait effect sizes

Description

This function estimates an appropriate distribution of effect sizes to simulate in a continuous trait power analysis.

Usage

```
find.cont.effect.sizes.2.sim(mydata)
```

Arguments

mydata	Your omics data matrix, with samples in rows

Value

a vector of effect sizes

Examples

```
ex_data = sapply(1:10, function(x){ rnorm(250, 40, 5) })
find.cont.effect.sizes.2.sim(ex_data)
```

`find.PA.effect.sizes.2.sim` *identify effect sizes*

Description

This function estimates an appropriate distribution of effect sizes to simulate in a power analysis.

Usage

```
find.PA.effect.sizes.2.sim(mydata)
```

Arguments

`mydata`	Your 'omics data matrix, with samples in rows

Value

a vector of effect sizes

Examples

```
ex_data = sapply(1:10, function(x){ rnorm(250, 40, 5) })
find.PA.effect.sizes.2.sim(ex_data)
```

`generate_report` *Generate Output Report*

Description

This function writes an output report

Usage

```
generate_report(
  omiprep,
  output_dir,
  output_filename = NULL,
  project = "Project",
  format = "pdf",
  template = "qc_report"
)

## S7 method for class <omiprep::Omiprep>
generate_report(
  omiprep,
  output_dir,
  output_filename = NULL,
  project = "Project",
  format = "pdf",
  template = "qc_report"
)
```

Arguments

`omiprep`	an object of class Omiprep
`output_dir`	character, the directory to save to
`output_filename`	character, default NULL i.e. create from input object
`project`	character, name for the current project
`format`	character, write either 'html' or 'pdf' report
`template`	character, type of report to output only current option is "qc_report"

imbalanced_power_plot *binary trait imbalanced design power analysis plot*

Description

This function (1) estimates an informative distribution of effect and power estimates given your datas total sample size, over a distribution of imbalanced sample sizes and (2) returns a summary plot.

Usage

```
imbalanced_power_plot(mydata)
```

Arguments

mydata	a numeric data matrix with samples in rows and features in columns

Value

a ggplot2 object

Examples

```
ex_data = matrix(NA, 1000, 2)
imbalanced_power_plot( ex_data )
```

missingness *Estimate Missingness*

Description

This function estimates missingness in a matrix of data and provides an option to exclude certain columns or features from the analysis, such as xenobiotics (with high missingness rates) in metabolomics data sets.

Usage

```
missingness(data, by = "row")
```

Arguments

data	matrix, a numeric matrix with samples in rows and features in columns
by	character, whether to calculate missingness by rows (samples) or column (features)

Value

data.frame, a data frame of missingness estimates for each sample/feature.

`multivariate_anova` *multivariate analysis*

Description

This function performs a multivariate analysis over a dependentlresponse and numerous independentlexplanatory variable

Usage

```
multivariate_anova(dep, indep_df)
```

Arguments

`dep`	a vector of the dependent variable
`indep_df`	a data frame of the independent variable

Value

ggplot2 table figure of

Examples

```
## simulate some correlated data
set.seed(1110)
n <- 250
mu <- c(5, 45, 25)
cmat <- matrix(c(1, 0.5, 0.3,
                 0.5, 1, 0.25,
                 0.3, 0.25, 1), nrow = 3, byrow = TRUE)
L <- chol(cmat)
Z <- matrix(rnorm(n * 3), nrow = n)
ex_data <- Z %*% L
ex_data <- sweep(ex_data, 2, mu, "+")
colnames(ex_data) = c("outcome","age","bmi")
multivariate_anova(dep = ex_data[,1], indep_df = ex_data[, 2:3])
```

outlier_detection *Identify indexes of outliers in data*

Description

Given a vector or matrix, this function returns a vector or matrix of 0|1, of the same structure with 1 values indicating outliers.

Usage

```
outlier_detection(data, nsd = 5, meansd = FALSE, by = "column")
```

Arguments

data	a matrix of numerical values, samples in row, features in columns
nsd	the unit distance in SD or IQR from the mean or median estimate, respectively outliers are identified at. Default value is 5.
meansd	set to TRUE if you would like to estimate outliers using a mean and SD method; set to FALSE if you would like to estimate medians and inter quartile ranges. The default is FALSE.
by	character, either 'column' to compute along columns or 'row' to compute across rows. Irrelevant for vectors.

Value

a matrix of 0 (not a sample outlier) and 1 (outlier)

outliers *Identify Outliers*

Description

This function identifies outliers from a vector of data at SD units from the mean.

Usage

```
outliers(x, nsd = 3)
```

Arguments

x	a numerical vector of data
nsd	the number of SD units from the mean to be used as an outlier cutoff.

Value

a list object of length three. (1) a vector of sample indexes indicating the outliers, (2) the lower outlier cuttoff value, (3) the upper outlier cuttoff value.

Examples

```
ex_data = rnbinom(500, mu = 40, size = 5)
outliers(ex_data)
```

`pc_and_outliers` *Principal Component Analysis*

Description

This function performs principal component analysis. In the first, missing data is imputed to the median. Subsequent to the derivation of the PC, the median imputed PC data is used to identify the number of informative or "significant" PC by (1) an acceleration analysis, and (2) a parrallel analysis. Finally the number of sample outliers are determined at 3, 4, and 5 standard deviations from the mean on the top PCs as determined by the acceleration factor analysis.

Usage

```
pc_and_outliers(
  omiprep,
  source_layer = "input",
  sample_ids = NULL,
  feature_ids = NULL
)

## S7 method for class <omiprep::Omiprep>
pc_and_outliers(
  omiprep,
  source_layer = "input",
  sample_ids = NULL,
  feature_ids = NULL
)
```

Arguments

`omiprep`	an object of class Omiprep
`source_layer`	character, type/source of data to use
`sample_ids`	character, vector of sample ids to include, default NULL includes all
`feature_ids`	character, vector of feature ids to include, default NULL includes all

Value

a data.frame

quality_control *Omics Quality Control*

Description

This function is a wrapper function that performs the key quality controls steps on an 'omics data set. Key principles: 1. keep the source underlying data as it is 2. copy the source data to a new data layer called qcing for processing 3. build an exclusion list, accumulating codes for exclusion reasons 4. make any adjustments needed in the destination copy of the data, flag these in the exclusion list 5. copy the final result to a data layer called post_qc 6. return the Omiprep object with the newly populated data layers

Usage

```
quality_control(
  omiprep,
  source_layer = "input",
  sample_missingness = 0.2,
  feature_missingness = 0.2,
  feature_skewness_threshold = NULL,
  feature_skewness_direction = "left",
  total_sum_abundance_sd = 5,
  outlier_udist = 5,
  outlier_treatment = "leave_be",
  winsorize_quantile = 1,
  tree_cut_height = 0.5,
  feature_selection = "max_var_exp",
  pc_outlier_sd = 5,
  max_num_pcs = 10,
  sample_ids = NULL,
  feature_ids = NULL,
  features_exclude_but_keep = NULL,
  cores = NULL,
  fast = FALSE
)

## S7 method for class <omiprep::Omiprep>
quality_control(
  omiprep,
  source_layer = "input",
  sample_missingness = 0.2,
  feature_missingness = 0.2,
  feature_skewness_threshold = NULL,
  feature_skewness_direction = "left",
  total_sum_abundance_sd = 5,
  outlier_udist = 5,
  outlier_treatment = "leave_be",
```

```
  winsorize_quantile = 1,
  tree_cut_height = 0.5,
  feature_selection = "max_var_exp",
  pc_outlier_sd = 5,
  max_num_pcs = 10,
  sample_ids = NULL,
  feature_ids = NULL,
  features_exclude_but_keep = NULL,
  cores = NULL,
  fast = FALSE
)
```

Arguments

omiprep	an object of class Omiprep
source_layer	character, the data layer to summarise
sample_missingness	numeric 0-1, percentage of data missingness which should prompt exclusion of a sample
feature_missingness	numeric 0-1, percentage of data missingness which should prompt exclusion of a feature
feature_skewness_threshold	numeric, optional skewness threshold to exclude features with skewed distributions. Set to ‘NULL‘ to disable.
feature_skewness_direction	character, direction of skewness to apply when ‘feature_skewness_threshold‘ is set. One of ‘"left"‘, ‘"right"‘, or ‘"both"‘.
total_sum_abundance_sd	numeric, number of TSA SD after which a sample would be excluded
outlier_udist	the unit distance in SD or IQR from the mean or median estimate, respectively outliers are identified at. Default value is 5.
outlier_treatment	character, how to handle outlier data values - options ’leave_be’, ’turn_NA’, or ’winsorize’
winsorize_quantile	numeric, quantile to winsorize to, only relevant if ’outlier_treatment’=’winsorize’
tree_cut_height	numeric, the threshold for feature independence in hierarchical clustering. Default is 0.5.
feature_selection	character, either ’max_var_exp’ or ’least_missingness’, how to select the independent feature within clusters
pc_outlier_sd	numeric, number of PCA SD after which a sample would be excluded
max_num_pcs	numeric, the maximum number of PCs to use (look in) when filtering samples on PC outlier SD, default=10, set to NULL to use all informative PCs from the Scree analysis

sample_ids	character, vector of sample ids to retain and work with, all others samples will be excluded
feature_ids	character, vector of feature ids to retain and work with, all other features will be excluded
features_exclude_but_keep	character, vector of feature ids indicating features to exclude from the sample and PCA quality control analysis but keep in the data, OR a name of a logical column in the features data indicating the same
cores	number of cores available for parallelism; the default null will try find the maximum available cores - 1; set to 1 for linear, but potentially slow, computation of the correlation matrix.
fast	If TRUE, accelerates correlation computation by imputing missing values to the column minimum, pre-ranking all columns, and computing Pearson correlation on ranked data (approximating Spearman). Substantially faster than exact Spearman at large feature dimensions ($p > 5000$) but assumes missing data are missing at random. Features with high missingness will have inflated rank ties at the median (ensure these are filtered out appropriately with the missingness option). Default FALSE.

---

read_metabolon *Read Metabolon Data*

Description

Read Metabolon Data

Usage

```
read_metabolon(
  filepath,
  sheet = NULL,
  feature_sheet = NULL,
  feature_id_col = NULL,
  sample_sheet = NULL,
  sample_id_col = NULL,
  return_Omiprep = TRUE
)
```

Arguments

filepath	character, commercial Metabolon excel sheet with extension .xls or .xlsx
sheet	character or integer, the excel sheet name (or index) from which to read.
feature_sheet	character or integer, the excel sheet name (or index) from which to read the feature data.
feature_id_col	character, the excel column containing the feature_id mapping to the data.

sample_sheet	character or integer, the excel sheet name (or index) from which to read the sample data.
sample_id_col	character, the excel column containing the sample_id mapping to the data.
return_Omiprep	logical, if TRUE (default) return a Omiprep object, if FALSE return a list.

Value

list or Omiprep object, list(data = matrix, samples = samples data.frame, features = features data.frame)

Examples

```
# version 1.1 data format
filepath1 <- system.file("extdata", "metabolon_v1.1_example.xlsx", package = "omiprep")
m <- read_metabolon(filepath1, sheet = 2)

# version 1.2 data format (different column names)
filepath2 <- system.file("extdata", "metabolon_v1.2_example.xlsx", package = "omiprep")
m <- read_metabolon(filepath2, sheet = 'OrigScale')


# version 2 data format
filepath3 <- system.file("extdata", "metabolon_v2_example.xlsx", package = "omiprep")
m <- read_metabolon(filepath3,
                    sheet = 'Batch-normalized Data',
                    feature_sheet = 'Chemical Annotation',
                    feature_id_col = 'CHEM_ID',
                    sample_sheet = 'Sample Meta Data',
                    sample_id_col = 'PARENT_SAMPLE_NAME')
```

---

read_nightingale *Read Nightingale Data (format 1)*

Description

Read Nightingale Data (format 1)

Usage

```
read_nightingale(filepath, return_Omiprep = TRUE)
```

Arguments

filepath	character, commercial Nightingale excel sheet with extension .xls or .xlsx
return_Omiprep	logical, if TRUE (default) return a Omiprep object, if FALSE return a list.

Value

list or Omiprep object, list(data = matrix, samples = samples data.frame, features = features data.frame)

Examples

```
# version 1 data format
filepath1 <- system.file("extdata", "nightingale_v1_example.xlsx", package = "omiprep")
m <- read_nightingale(filepath1)

# version 2 data format
filepath2 <- system.file("extdata", "nightingale_v2_example.xlsx", package = "omiprep")
m <- read_nightingale(filepath2)
```

`read_olink` *Read and Process Olink NPX Data File*

Description

This function reads and processes an Olink NPX file in long format. It supports ‘.csv‘, ‘.xls‘, ‘.xlsx‘, ‘.txt‘, ‘.zip‘, and ‘.parquet‘ formats, using Olink’s own OlinkAnalyze::read_NPX() function, and returns a omiprep object or a list of matrices and metadata frames for further analysis.

Usage

```
read_olink(filepath, return_Omiprep = FALSE)
```

Arguments

`filepath`	A string specifying the path to the Olink NPX file.
`return_Omiprep`	logical, if TRUE (default) return a Omiprep object, if FALSE return a list.

Details

The function checks whether the input data is in long format by verifying the presence of duplicate ‘SampleID‘ values. It also accommodates two variants of Olink files:

- Files that include a ‘Sample_Type‘ column with values ‘"SAMPLE"‘ and ‘"CONTROL"‘.
- Files that use the ‘SampleID‘ column to label control samples (e.g., entries containing ‘"CONTROL"‘).

If neither format is detected, the function stops with an error indicating that the data is likely not from Olink.

Value

Omiprep object or a named list with the following elements:

data A matrix of NPX values with ‘SampleID‘ as rows and ‘OlinkID‘ as columns, containing only sample data.

samples A ‘data.frame‘ containing metadata for samples.

features A ‘data.frame‘ containing feature-level metadata for samples.

controls A matrix of NPX values for control samples.

control_metadata A ‘data.frame‘ containing metadata for control samples.

Examples

```
## Not run:
  filepath <- system.file("extdata", "example_olink_data.txt", package = "omiprep")
  olink_data <- read_olink(filepath)

## End(Not run)
```

`read_somalogic` *Read and Process SomaLogic adat file*

Description

This function reads and processes a commercial SomaLogic '.adat' file. It extracts RFU (Relative Florecent Units) data for samples and controls, along with their respective metadata and feature (protein) metadata. The function returns a structured list suitable for further analysis.

Usage

```
read_somalogic(filepath, return_Omiprep = FALSE)
```

Arguments

`filepath`	A string specifying the path to the SomaLogic '.adat' file.
`return_Omiprep`	logical, if TRUE (default) return a Omiprep object, if FALSE return a list.

Value

A omiprep object or a named list with the following elements:

data A matrix of RFU values for experimental samples, with 'SampleId' as row names and 'SeqId' columns as column names.

samples A data frame containing metadata for experimental samples, with 'sample_id' renamed from 'SampleId'.

features A data frame containing feature-level metadata, including a newly created 'feature_id' column derived from 'SeqId'.

controls A matrix of RFU values for control samples, specifically "Calibrator" samples, with 'SampleId' as row names and 'SeqId' columns as column names. If duplicate control 'SampleId' values are present, control 'SampleId' is replaced with 'paste0(SampleId, "_", SlideId)'.

control_metadata A data frame containing metadata for control samples, specifically "Calibrator" samples, with 'sample_id' renamed from 'SampleId'.

run_metaboprep1 *Metaboprep 1 pipeline*

Description

This function runs the original metaboprep1 pipeline using the old parameter file input format. The function requires access to the internet as the old package (default github commit 'bbe1f85') will be dynamically downloaded and used to process the data.

Usage

```
run_metaboprep1(parameter_file, gitcommit = "bbe1f85", attempt_report = FALSE)
```

Arguments

parameter_file	character, full file path to the metaboprep 1 parameter file
gitcommit	character, Github commit - default pinned to the last stable metaboprep 1 version 'bbe1f85'
attempt_report	logical, whether to attempt metaboprep1 report generation. Default=FALSE as this can lead to errors on some operating systems.

sample_summary *Sample Summary Statistics*

Description

Summarise the sample data

Usage

```
sample_summary(
  omiprep,
  source_layer = "input",
  outlier_udist = 5,
  sample_ids = NULL,
  feature_ids = NULL,
  output = "data.frame"
)

## S7 method for class <omiprep::Omiprep>
sample_summary(
  omiprep,
  source_layer = "input",
  outlier_udist = 5,
  sample_ids = NULL,
```

```
  feature_ids = NULL,
  output = "data.frame"
)
```

Arguments

omiprep	an object of class Omiprep
source_layer	character, the data layer to summarise
outlier_udist	the unit distance in SD or IQR from the mean or median estimate, respectively outliers are identified at. Default value is 5.
sample_ids	character, vector of sample ids to work with
feature_ids	character, vector of feature ids to work with
output	character, type of output, either 'object' to return the updated omiprep object, or 'data.frame' to return the data.

---

shiny_app *Omiprep Shiny App*

Description

Launch a Shiny app to explore the Omiprep object

Usage

```
shiny_app(omiprep)

## S7 method for class <omiprep::Omiprep>
shiny_app(omiprep)
```

Arguments

omiprep	an object of class Omiprep

Value

Runs a Shiny app

summarise *Summary Statistics*

Description

Summarise the sample and feature data

Usage

```
summarise(
  omiprep,
  source_layer = "input",
  outlier_udist = 5,
  tree_cut_height = 0.5,
  feature_selection = "max_var_exp",
  sample_ids = NULL,
  feature_ids = NULL,
  features_exclude = NULL,
  output = "data.frame",
  cores = NULL,
  fast = FALSE
)

## S7 method for class <omiprep::Omiprep>
summarise(
  omiprep,
  source_layer = "input",
  outlier_udist = 5,
  tree_cut_height = 0.5,
  feature_selection = "max_var_exp",
  sample_ids = NULL,
  feature_ids = NULL,
  features_exclude = NULL,
  output = "data.frame",
  cores = NULL,
  fast = FALSE
)
```

Arguments

omiprep	an object of class Omiprep
source_layer	character, the data layer to summarise
outlier_udist	the unit distance in SD or IQR from the mean or median estimate, respectively outliers are identified at. Default value is 5.
tree_cut_height	numeric, the threshold for feature independence in hierarchical clustering. Default is 0.5.

`feature_selection`
: character, either 'max_var_exp' or 'least_missingness', how to select the independent feature within clusters

`sample_ids`
: character, vector of sample ids to work with

`feature_ids`
: character, vector of feature ids to work with

`features_exclude`
: character, vector of feature id indicating features to exclude from the sample and PCA summary analysis but keep in the data

`output`
: character, type of output, either 'object' to return the updated omiprep object, or 'data.frame' to return the data.

`cores`
: number of cores available for parallelism; the default null will try find the maximum available cores - 1; set to 1 for linear, but potentially slow, computation of the correlation matrix.

`fast`
: If `TRUE`, accelerates correlation computation by imputing missing values to the column minimum, pre-ranking all columns, and computing Pearson correlation on ranked data (approximating Spearman). Substantially faster than exact Spearman at large feature dimensions ($p > 5000$) but assumes missing data are missing at random. Features with high missingness will have inflated rank ties at the median (ensure these are filtered out appropriately with the missingness option). Default `FALSE`.

`summary.Omiprep` *Summary Method for Omiprep Object*

Description

Provides a concise, human-readable summary of a 'Omiprep' object. It reports key dimensions of the data, the presence of metadata columns, the number of data layers, and the status of quality control summaries and exclusions.

Usage

```
## S7 method for class <omiprep::Omiprep>
summary(object, ...)
```

Arguments

`object`
: A 'Omiprep' object.

`...`
: Additional arguments (not used).

Value

Invisibly returns NULL. Prints a formatted summary to the console.

See Also

`class_omiprep`

total_sum_abundance *Estimates total sum abundance*

Description

This function estimates total sum abundance for numeric data in a matrix, for (1) all features and (2) all features with complete data.

Usage

```
total_sum_abundance(data, ztransform = TRUE)
```

Arguments

data	matrix, the 'omics data matrix. Samples in rows, features in columns
ztransform	logical, should the feature data be z-transformed and absolute value minimum, mean shifted prior to summing the feature values. TRUE or FALSE.

Value

a data frame of estimates for (1) total sum abundance and (2) total sum abundance at complete features for each samples

tree_and_independent_features *Identify Independent Features in a Numeric Matrix*

Description

This function identifies independent features using Spearman's rho correlation distances, and a dendrogram tree cut step.

Usage

```
tree_and_independent_features(
  data,
  tree_cut_height = 0.5,
  features_exclude = NULL,
  feature_selection = "max_var_exp",
  cores = NULL,
  fast = FALSE
)
```

Arguments

`data` matrix, the 'omics data matrix. samples in row, features in columns

`tree_cut_height` the tree cut height. A value of 0.2 (1-Spearman's rho) is equivalent to saying that features with a rho >= 0.8 are NOT independent.

`features_exclude` character, vector of feature id indicating features to exclude from the sample and PCA summary analysis but keep in the data

`feature_selection` character. Method for selecting a representative feature from each correlated feature cluster.

`cores` number of cores available for parallelism; the default null will try find the maximum available cores - 1; set to 1 for linear, but potentially slow, computation of the correlation matrix.

`fast` If `TRUE`, accelerates correlation computation by imputing missing values to the column minimum, pre-ranking all columns, and computing Pearson correlation on ranked data (approximating Spearman). Substantially faster than exact Spearman at large feature dimensions ($p > 5000$) but assumes missing data are missing at random. Features with high missingness will have inflated rank ties at the median (ensure these are filtered out appropriately with the missingness option). Default `FALSE`. One of:

- `"max_var_exp"` (Default) Selects the feature with the highest sum of absolute Spearman correlations to other features in the cluster; effectively the feature explaining the most shared variance.
- `"least_missingness"` Selects the feature with the fewest missing values within the cluster.

Value

A list with the following components:

data A 'data.frame' with:

- 'feature_id': Feature (column) names from the input matrix.
- 'k': The cluster index assigned to each feature after tree cutting.
- 'independent_features': Logical indicator of whether the feature was selected as an independent (representative) feature.

tree A 'hclust' object representing the hierarchical clustering of the features based on 1 - |Spearman's rho| distance.

variable_by_factor *ggplot2 violin plot*

Description

This function performs univariate linear analysis of a dependent and an independent variable and generates a viloin or box plot to illustrate the associated structure.

Usage

```
variable_by_factor(
  dep,
  indep,
  dep_name = "dependent",
  indep_name = "independent",
  orderfactor = TRUE,
  violin = TRUE
)
```

Arguments

dep	a vector of the dependent variable
indep	a vector of the independent variable
dep_name	name of the dependent variable
indep_name	name of the independent variable
orderfactor	order factors alphebetically
violin	box plot or violin plot. violin = TRUE is default

Value

a ggplot2 object

Examples

```
x = c( rnorm(20, 10, 2), rnorm(20, 20, 2) )
y = as.factor( c( rep("A", 20), rep("B", 20)  ) )
variable_by_factor(dep = x , indep = y, dep_name = "expression", indep_name = "species" )
```

Index