Package 'omicsTools'

November 5, 2026

Title Omics Data Processing Toolbox

Version 2.0.0

Description Provides tools for processing and analyzing omics data from genomics, transcriptomics, proteomics, and metabolomics platforms. Includes data import, filtering, imputation, normalization, targeted calibration, spectrum export, visualization, statistical comparisons, and reproducible QC reports. The QC-RLSC (quality control-based robust LOESS signal correction) algorithm is used for normalization. Dunn et al. (2011) <doi:10.1038/nprot.2011.335>.

License AGPL (>= 3)

URL https://github.com/cheemalab/omicsTools

BugReports https://github.com/cheemalab/omicsTools/issues

Depends R (>= 4.1.0)

Imports bs4Dash, cli, dplyr, DT, ggplot2, htmltools, jsonlite, readr, rlang, stats, shiny (>= 1.7.2), tibble, tidyr, utils, tools

Encoding UTF-8

Suggests testthat (>= 3.2.0), knitr, rmarkdown, cmmr, readxl, openxlsx, zip

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Yaoxiang Li [aut], Meth Jayatilake [aut], Anushka Dongaonkar [aut], Rupayan Mallick [aut], Amrita Cheema [cre, aut]

Maintainer Amrita Cheema <metabolomics@georgetown.edu>

Repository CRAN

Date/Publication 2026-11-04 23:10:02 UTC

Contents

calculate_qc_rsd . . . 3
check_and_sort_columns . . . 4
check_match . . . 5
combine_logical_tibbles . . . 5
compare_omics_groups . . . 6
convert_mrm_data . . . 7
detect_duplicates . . . 8
detect_sample_outliers . . . 9
export_omics_results . . . 10
filter_blank_features . . . 10
filter_missing_value . . . 11
filter_qc_rsd . . . 12
fit_targeted_calibration . . . 13
flag_underexpressed_features . . . 14
generate_process_report . . . 15
generate_sample_metadata . . . 16
handle_missing_values . . . 17
impute_half_min . . . 18
internal_standard_normalize . . . 19
load_data . . . 20
load_omics_data . . . 20
load_parse_sciex_txt . . . 21
ms1_annotation . . . 22
omics_example_file . . . 23
omics_gui_capabilities . . . 24
plot_lcms_features . . . 24
plot_lipid_data_summary . . . 25
plot_met_data_summary . . . 26
plot_omics_heatmap . . . 27
plot_omics_pca . . . 28
plot_peak_list . . . 28
plot_targeted_calibration . . . 29
pqn_normalize . . . 29
predict_targeted_calibration . . . 30
process_mrm_duplicates . . . 31
process_omics_data . . . 33
qc_normalize . . . 34
run_app . . . 35
save_msp . . . 36
transpose_df . . . 37
validate_omics_data . . . 38
write_omics_report . . . 39

Index **40**

`calculate_qc_rsd` *Calculate QC Statistics and RSD*

Description

This function calculates the mean, standard deviation, and coefficient of variation for QC data, optionally ignoring NA values. It also calculates the relative standard deviation (RSD) for both Pooled QC and NIST samples. The input data should be a tibble where the first column is the sample ID and all other columns are features.

Usage

```
calculate_qc_rsd(
  data,
  qc_string = "Pooled QC|Pooled_QC",
  nist_string = "NIST",
  id_col = "sample_id",
  ignore_na = TRUE
)
```

Arguments

`data`	A tibble with the first column as sample IDs and other columns as features.
`qc_string`	A string pattern to filter Pooled QC samples. Default is "Pooled QC|Pooled_QC".
`nist_string`	A string pattern to filter NIST samples. Default is "NIST".
`id_col`	The name of the sample ID column. Default is "sample_id".
`ignore_na`	Logical, whether to ignore NA values in calculations. Default is TRUE.

Value

A list containing tibbles with calculated statistics for Pooled QC and NIST samples. If no matching samples are found, returns NULL for that category.

Author(s)

Yaoxiang Li

Examples

```
# Generate example data
example_data <- tibble::tibble(
  sample_id = c("Pooled QC 1", "Pooled QC 2", "NIST 1", "NIST 2", "Sample 1"),
  feature1 = c(10, 15, 20, NA, 5),
  feature2 = c(20, NA, 25, 30, 10),
  feature3 = c(NA, 25, 30, 35, 15)
)
```

```
# Calculate QC statistics and RSD
rsd_stats <- calculate_qc_rsd(example_data)

# Access results
pooled_qc_rsd <- rsd_stats$Pooled_QC_RSD
nist_rsd <- rsd_stats$NIST_RSD
```

`check_and_sort_columns` *Check and Sort Columns, Compare Values*

Description

This function checks if two tibbles have the same column names, sorts the columns of one tibble to match the order of the other, and then checks if all values in both tibbles are the same.

Usage

```
check_and_sort_columns(area_data, area_txt)
```

Arguments

`area_data`	A tibble containing the data to be checked and sorted.
`area_txt`	A tibble containing the reference data for column order and value comparison.

Value

Prints messages indicating whether the tibbles have the same column names and whether all values are the same.

Author(s)

Yaoxiang Li

Examples

```
## Not run:
area_data <- read.delim("path/to/All_txt.txt", check.names = FALSE)
area_txt <- read.delim("path/to/area.txt", check.names = FALSE)
check_and_sort_columns(area_data, area_txt)

## End(Not run)
```

check_match *Check Match*

Description

This function checks if sample IDs and column names match between area data and internal standard area data.

Usage

```
check_match(area_data, istd_area_data)
```

Arguments

area_data	A data frame containing the area data.
istd_area_data	A data frame containing the internal standard area data.

Value

A logical value indicating if the sample IDs and column names match.

Author(s)

Yaoxiang Li

Examples

```
area_data <- data.frame(sample_id = c("S1", "S2", "S3"), A = 1:3, B = 4:6)
istd_area_data <- data.frame(sample_id = c("S1", "S2", "S3"), A = 1:3, B = 4:6)
check_match(area_data, istd_area_data)
```

combine_logical_tibbles *Combine Multiple Logical Tibbles with Intersection or Union*

Description

Combine Multiple Logical Tibbles with Intersection or Union

Usage

```
combine_logical_tibbles(..., method = c("intersection", "union"))
```

Arguments

...	Multiple tibbles to be combined. Each tibble should have the same dimensions, same column names in the same order, and the first column values should be identical across all tibbles.
method	A string indicating the method to combine the tibbles. Either "intersection" or "union". Default is "intersection".

Value

A combined tibble where each cell is TRUE based on the method: - "intersection": TRUE if all corresponding cells in the input tibbles are TRUE, otherwise FALSE. - "union": TRUE if at least one corresponding cell in the input tibbles is TRUE, otherwise FALSE.

Author(s)

Yaoxiang Li

Examples

```
tibble1 <- tibble::tibble(id = 1:3, A = c(TRUE, FALSE, NA), B = c(TRUE, TRUE, FALSE))
tibble2 <- tibble::tibble(id = 1:3, A = c(TRUE, TRUE, TRUE), B = c(FALSE, TRUE, TRUE))
tibble3 <- tibble::tibble(id = 1:3, A = c(TRUE, FALSE, TRUE), B = c(TRUE, TRUE, NA))
combine_logical_tibbles(tibble1, tibble2, tibble3, method = "intersection")
combine_logical_tibbles(tibble1, tibble2, tibble3, method = "union")
```

---

compare_omics_groups *Compare two groups of samples*

Description

Calculates Welch or Wilcoxon tests on log2 intensities and adjusts p-values using Benjamini-Hochberg. When zeros occur, half the smallest positive value in the two groups is added to that feature before transformation. Log2 fold change is the log2 ratio of group means with the same offset. Untestable features retain an NA p-value and an explanatory status.

Usage

```
compare_omics_groups(
  data,
  group_col,
  reference,
  comparison,
  method = c("welch", "wilcoxon"),
  id_col = "sample_id",
  metadata_cols = NULL
)
```

Arguments

`data`	A samples-by-features data frame including sample IDs.
`group_col`	Group column.
`reference, comparison`	Reference and comparison group labels.
`method`	Welch t-test or Wilcoxon rank-sum test.
`id_col`	Sample ID column.
`metadata_cols`	Columns excluded from intensity processing. NULL also identifies nonnumeric columns as metadata.

Value

A tibble with feature IDs, group means, log2FC, p_value, fdr, and status.

Author(s)

Yaoxiang Li

`convert_mrm_data` *Convert MRM Data to Wide Format*

Description

This function converts a tibble containing MRM data to a wide format based on the specified response column.

Usage

```
convert_mrm_data(
  data,
  response_col,
  sample_name_col = "data_filename",
  sample_id_col = "sample_id",
  component_name_col = "component_name"
)
```

Arguments

`data`	A tibble containing the MRM transition data.
`response_col`	Name of the column containing the response data to be spread.
`sample_name_col`	Name of the column containing sample name information.
`sample_id_col`	Name of the column containing sample ID information.
`component_name_col`	Name of the column containing component name information.

Value

A tibble in wide format with samples as rows and compounds as columns.

Author(s)

Yaoxiang Li

Examples

```
## Not run:
all_txt <- tibble::tribble(
  ~`data_filename`, ~`sample_id`, ~`component_name`, ~Area, ~`IS Area`,
  "Sample1", "ID1", "Compound1", 100, 50,
  "Sample1", "ID2", "Compound2", 200, 75,
  "Sample2", "ID1", "Compound1", 150, 60,
  "Sample2", "ID2", "Compound2", 250, 80
)
area_data <- convert_mrm_data(all_txt, "Area", "data_filename", "sample_id", "component_name")
is_area_data <- convert_mrm_data(all_txt, "IS Area", "data_filename", "sample_id", "component_name")
print(area_data)
print(is_area_data)

## End(Not run)
```

---

detect_duplicates *Detect Duplicate MRM Transitions*

Description

This function adds a column 'MRM_Duplicate_Flag' to the tibble, indicating if a row is a duplicate based on the criteria: same polarity, less than a 1-minute retention time difference, and the same MRM transition (Q1/Q3).

Usage

```
detect_duplicates(
  data,
  polarity_col = "polarity",
  retention_time_col = "retention_time",
  mass_info_col = "Mass Info",
  component_name_col = "component_name"
)
```

Arguments

data	A tibble containing the MRM transition data.
polarity_col	Name of the column containing polarity information.

retention_time_col
: Name of the column containing retention time information.

mass_info_col
: Name of the column containing mass information.

component_name_col
: Name of the column containing component name information.

Value

The original tibble with an added 'MRM_Duplicate_Flag' column.

Author(s)

Yaoxiang Li

Examples

```
## Not run:
sample_data <- tibble::tribble(
  ~polarity, ~`retention_time`, ~`Mass Info`, ~`component_name`,
  "Positive", 1.95, "61.0 / 44.0", "Urea_pos",
  "Positive", 8.34, "206.0 / 189.0", "Lipoamide_pos",
  "Positive", 2.18, "339.1 / 110.0", "AICAR_pos",
  "Positive", 1.76, "175.1 / 70.0", "Arginine_pos",
  "Positive", 1.75, "176.2 / 159.1", "Citrulline_pos",
  "Positive", 8.90, "198.0 / 181.0", "Dopa_pos",
  "Positive", 2.06, "132.1 / 86.0", "Isoleucine_pos",
  "Positive", 1.92, "132.1 / 43.1", "Leucine_pos",
  "Positive", 1.76, "150.1 / 133.0", "Methionine_pos",
  "Positive", 7.79, "166.1 / 103.0", "Phenylalanine_pos"
)
detect_duplicates(sample_data, "polarity", "retention_time", "Mass Info", "component_name")

## End(Not run)
```

---

detect_sample_outliers

Detect multivariate sample outliers

Description

Detect multivariate sample outliers

Usage

```
detect_sample_outliers(data, threshold = NULL)
```

Arguments

data	Numeric samples-by-features matrix or data frame.
threshold	Threshold for robust multivariate distance.

Value

A tibble with sample identifiers, scores, and outlier flags.

Author(s)

Yaoxiang Li

`export_omics_results` *Export a complete workflow result*

Description

Writes separate intensity, sample, feature, QC, comparison, blank audit, processing-step, and settings files together with a self-contained HTML report. A temporary directory is published only after every output succeeds.

Usage

```
export_omics_results(result, directory)
```

Arguments

`result`	A result from process_omics_data().
`directory`	A new output directory. Existing directories are not overwritten.

Value

Invisibly, the output directory path.

Author(s)

Yaoxiang Li

`filter_blank_features` *Filter features against blank samples*

Description

Positive intensities are averaged separately in blanks and study samples. A feature is retained when its study mean is positive and at least ratio times its blank mean in every batch. The audit table records each decision.

Usage

```
filter_blank_features(
  data,
  sample_type_col,
  blank_value = "Blank",
  study_value = "Sample",
  ratio = 3,
  batch_col = NULL,
  id_col = "sample_id",
  metadata_cols = NULL
)
```

Arguments

data	A samples-by-features data frame including sample IDs.
sample_type_col	Column identifying blanks and study samples.
blank_value, study_value	Labels for blank and study samples.
ratio	Minimum study-to-blank intensity ratio, at least one.
batch_col	Optional batch column. Every batch needs blanks and study samples.
id_col	Sample ID column.
metadata_cols	Columns excluded from intensity processing. NULL also identifies nonnumeric columns as metadata.

Value

A list containing the filtered data and a per-feature, per-batch summary.

Author(s)

Yaoxiang Li

---

filter_missing_value *Filter features by missing-value rate*

Description

Supports data frames and matrices.

Usage

```
filter_missing_value(data, percent = 0.2, metadata_cols = 1L)
```

Arguments

data	A data frame or matrix.
percent	Maximum allowed missing-value proportion per feature.
metadata_cols	Columns in a tabular input that should not be filtered.

Value

The filtered object, preserving the input representation.

Author(s)

Yaoxiang Li

---

filter_qc_rsd *Filter features by quality-control relative standard deviation*

Description

Filter features by quality-control relative standard deviation

Usage

```
filter_qc_rsd(
  data,
  percent = 0.2,
  qc_index = NULL,
  qc_string = "QC|Pooled QC|Pooled_QC",
  id_col = 1L,
  metadata_cols = 1L
)
```

Arguments

data	A data frame or matrix.
percent	Maximum allowed QC relative standard deviation.
qc_index	Logical or numeric indicator identifying QC rows. When omitted for tabular inputs, ‘qc_string‘ is matched in ‘id_col‘.
qc_string	Pattern used to identify QC rows.
id_col	Sample identifier column for tabular data.
metadata_cols	Columns that should not be filtered.

Value

The filtered object, preserving the input representation.

Author(s)

Yaoxiang Li

`fit_targeted_calibration` *Fit a targeted-assay calibration model*

Description

Fit a targeted-assay calibration model

Usage

```
fit_targeted_calibration(
  standards,
  concentration_col = "std_conc",
  response_col = "response",
  model = c("linear", "exponential"),
  response_range = NULL,
  weights = NULL
)
```

Arguments

`standards`	Data frame containing known concentrations and responses.
`concentration_col`	Name of the concentration column.
`response_col`	Name of the instrument-response column.
`model`	Calibration model: ordinary linear regression or an exponential curve fitted by regressing log concentration on response.
`response_range`	Optional two-element response range used to subset standards before fitting.
`weights`	Optional non-negative observation weights.

Value

An object of class ‘omics_calibration‘.

Author(s)

Yaoxiang Li

`flag_underexpressed_features` *Flag Underexpressed Features in Samples Based on Blank Samples*

Description

Flags features in samples based on their abundance in blank samples. If a feature is NA in the first blank sample, all samples for this feature are marked as TRUE. Otherwise, for each sample and feature, if the peak area is at least 10 times the area of the first blank sample, it is marked as TRUE, else FALSE. NA values in the samples remain unchanged.

Usage

```
flag_underexpressed_features(
  data,
  sample_id_col = "sample_id",
  feature_cols,
  threshold = 10
)
```

Arguments

`data`	A tibble containing the MRM transition data.
`sample_id_col`	Name of the column containing sample ID information.
`feature_cols`	A vector of column names representing the features.
`threshold`	A numeric value representing the threshold multiplier (default is 10).

Value

A tibble with the same dimensions and column names as the input data, containing TRUE, FALSE, or NA based on the criteria.

Author(s)

Yaoxiang Li

Examples

```
## Not run:
area_data <- tibble::tibble(
  sample_id = c(
    "011_Blank", "012_sample_002", "013_NIST_Plasma", "014_Blank",
    "015_sample_006", "016_sample_003"
  ),
  `2-Deoxyglucose-6-Phosphate_neg` = c(NA, 345423.96, NA, NA, 125889.80, 323818.25),
  `2-Oxoisoleucine_neg` = c(NA, 53004.06, 124669.80, NA, 23650.90, 118364.36),
 `3-(4-Hydroxyphenyl)propionate_neg` = c(NA, 53004.06, 124669.80, NA, 23650.90, 118364.36)
```

```
)
flagged_data <- flag_underexpressed_features(
  area_data,
  sample_id_col = "sample_id",
  feature_cols = names(area_data)[-1]
)
print(flagged_data)

## End(Not run)
```

`generate_process_report` *Generate Process Report for Sciex 7500/5500 Raw Data*

Description

This function generates a comprehensive process report for Sciex 7500/5500 raw data, including data normalization, missing value imputation, and optional normalization and flagging steps. The results are saved in a temporary directory and then zipped into a file for easy sharing.

Usage

```
generate_process_report(
  input_file,
  output_file = NULL,
  filter_blank = TRUE,
  blank_string = "Blank|BLANK|blank",
  filter_nist = TRUE,
  nist_string = "NIST|Nist|nist",
  imputation_threshold = 0.25,
  imputation_method = "half_min",
  qc_string = "QC",
  include_is_normalization = TRUE,
  include_qc_rlsc = TRUE,
  include_pqn = TRUE,
  include_qc_rsd = TRUE,
  include_snr_flag = TRUE,
  snr_threshold = 10,
  include_area_flag = TRUE,
  include_height_flag = TRUE,
  id_col = "sample_id",
  ignore_na = TRUE
)
```

Arguments

`input_file`	The path to the input file containing raw data.
`output_file`	The path to the output zip file.

filter_blank	Logical, whether to filter out blank samples (default: TRUE).
blank_string	Character, regular expression pattern to match blank sample IDs (default: 'Blank|BLANK|blank').
filter_nist	Logical, whether to filter out NIST samples (default: TRUE).
nist_string	Character, regular expression pattern to match NIST sample IDs (default: 'NIST|Nist|nist').
imputation_threshold	Numeric, threshold for missing value imputation (default: 0.25).
imputation_method	Character, method for missing value imputation (default: 'half_min').
qc_string	Character, regular expression pattern to match QC sample IDs (default: 'QC').
include_is_normalization	Logical, whether to include internal standard normalization (default: TRUE).
include_qc_rlsc	Logical, whether to include QC-RLSC normalization (default: TRUE).
include_pqn	Logical, whether to include PQN normalization (default: TRUE).
include_qc_rsd	Logical, whether to include QC RSD calculation (default: TRUE).
include_snr_flag	Logical, whether to include Signal-to-Noise ratio flagging (default: TRUE).
snr_threshold	Numeric, threshold for Signal-to-Noise ratio flagging (default: 10).
include_area_flag	Logical, whether to include area threshold flagging (default: TRUE).
include_height_flag	Logical, whether to include height threshold flagging (default: TRUE).
id_col	Character, name of the column containing sample IDs (default: 'sample_id').
ignore_na	Logical, whether to ignore NA values in QC RSD calculation (default: TRUE).

Value

The path to the generated zip file containing the process report.

Author(s)

Yaoxiang Li

---

generate_sample_metadata

Generate sample metadata from file or sample names

Description

Splits sample identifiers into structured fields and preserves acquisition order.

Usage

```
generate_sample_metadata(
  samples,
  separator = "[_-]",
  fields = c("group", "subject", "replicate"),
  strip_extension = TRUE
)
```

Arguments

samples	Character vector of sample or file names.
separator	Regular expression used to split identifiers.
fields	Names assigned to the split components.
strip_extension	Whether to remove a final file extension.

Value

A tibble containing ‘sample_id‘, ‘acquisition_order‘, and requested metadata fields.

Author(s)

Yaoxiang Li

---

handle_missing_values *Handle Missing Values in a Tibble*

Description

This function filters features based on a missing value threshold and imputes missing values using various methods. Metadata columns are specified by the user and are exempt from filtering and imputation.

Usage

```
handle_missing_values(
  data,
  threshold = 0.2,
  imputation_method = "half_min",
  metadata_cols = NULL
)
```

Arguments

data	A tibble containing the data with potential missing values.
threshold	A numeric value between 0 and 1 representing the maximum allowable proportion of missing values in a feature. Default is 0.20.
imputation_method	A character string indicating the method to use for imputation. Valid methods are "mean", "median", "mode", and "half_min". Default is "mean".
metadata_cols	A vector of column names or indices to be treated as metadata, exempt from filtering and imputation. Default is NULL.

Value

A tibble with filtered features and imputed missing values.

Author(s)

Yaoxiang Li

Examples

```
data <- tibble::tibble(
  Feature1 = c(1, 2, NA, 4, 5),
  Feature2 = c(NA, 2, 3, 4, NA),
  Feature3 = c(1, NA, 3, NA, 5),
  Metadata = c("A", "B", "C", "D", "E")
)
imputed_data <- handle_missing_values(
  data,
  threshold = 0.20,
  imputation_method = "half_min",
  metadata_cols = "Metadata"
)
print(imputed_data)
```

---

impute_half_min *Impute missing values with half of a feature minimum*

Description

Impute missing values with half of a feature minimum

Usage

```
impute_half_min(data, metadata_cols = 1L)
```

Arguments

`data`	A data frame or matrix.
`metadata_cols`	Columns in a tabular input that should not be imputed.

Value

The imputed object, preserving the input representation.

Author(s)

Yaoxiang Li

`internal_standard_normalize` *Internal Standard Normalize*

Description

This function performs internal standard normalization.

Usage

```
internal_standard_normalize(area_data, istd_area_data)
```

Arguments

`area_data`	A data frame containing the area data.
`istd_area_data`	A data frame containing the internal standard area data.

Value

A data frame with normalized data.

Author(s)

Yaoxiang Li

Examples

```
area_data <- data.frame(sample_id = c("S1", "S2", "S3"), A = 1:3, B = 4:6)
istd_area_data <- data.frame(sample_id = c("S1", "S2", "S3"), A = 1:3, B = 4:6)
normalized_data <- internal_standard_normalize(area_data, istd_area_data)
```

load_data *Load omics data*

Description

Load omics data

Usage

```
load_data(file_path, format = NULL, ...)
```

Arguments

file_path	Path to the input file.
format	Optional format override. By default the file extension is used.
...	Additional arguments passed to the corresponding reader.

Value

The value returned by [load_omics_data()].

Author(s)

Yaoxiang Li

load_omics_data *Load an omics dataset*

Description

Reads common delimited, Excel, and serialized R formats without attaching additional packages. Tabular inputs are returned as tibbles.

Usage

```
load_omics_data(file_path, format = NULL, ...)
```

Arguments

file_path	Path to the input file.
format	Optional format override. By default the file extension is used.
...	Additional arguments passed to the corresponding reader.

Value

A tibble for tabular data, or the stored R object for ‘rds‘, ‘rda‘, and ‘rdata‘ files. An R data file containing multiple objects returns a named list.

Author(s)

Yaoxiang Li

`load_parse_sciex_txt` *Load and Parse SCIEX OS Exported LC-MRM-MS2 Data*

Description

Load and Parse SCIEX OS Exported LC-MRM-MS2 Data

Usage

```
load_parse_sciex_txt(
  file_path,
  input_data = NULL,
  return_all_columns = TRUE,
  check_negative_values = TRUE,
  fix_istds = TRUE
)
```

Arguments

`file_path`	File path of the input text file of a complete output of the SCIEX OS results from a sequence. File should be tab-delimited and in the ’long’ format.
`input_data`	Input tibble of raw SCIEX (pre-parsing) text file. If ‘NULL‘ (default value), data will be loaded from ‘file_path‘.
`return_all_columns`	Logical value as to whether to return all columns (‘TRUE‘) or just the necessary columns for downstream machine learning analysis or quality control review (‘FALSE‘). Default value is ‘TRUE‘. When set to false, the columns included in the returned tibble include: ‘"component_name"‘, ‘"component_idx"‘, ‘"precursor_mz"‘, ‘"product_mz"‘, ‘"is_istd"‘, ‘"istd"‘, ‘"retention_time_expected"‘, ‘"data_filename"‘, ‘"data_file_idx"‘, ‘"sample_id"‘, ‘"sample_type"‘, ‘"component_type"‘, ‘"polarity"‘, ‘"component_group"‘, ‘"outlier_reasons"‘, ‘"retention_time_expected_istd"‘, ‘"area"‘, ‘"istd_area"‘, ‘"area_ratio"‘, ‘"height"‘, ‘"istd_height"‘, ‘"height_ratio"‘, ‘"peak_quality"‘, ‘"istd_peak_quality"‘, ‘"retention_time"‘, ‘"retention_time_istd"‘, ‘"rt_error"‘, ‘"rt_delta_min"‘, ‘"rt_start"‘, ‘"istd_rt_start"‘, ‘"rt_end"‘, ‘"istd_rt_end"‘, ‘"peak_width"‘, ‘"istd_peak_width"‘, ‘"fwhm"‘, ‘"istd_fwhm"‘, ‘"signal_noise"‘, ‘"istd_signal_noise"‘, ‘"modified"‘, ‘"relative_rt"‘, ‘"used"‘, ‘"tailing_factor"‘, ‘"asymmetry_factor"‘, ‘points_across_baseline"‘, ‘"points_across_fwhm"‘).

check_negative_values
: Logical value as to whether to check for negative values in the ‘area‘ and ‘height‘ variables (for both components and internal standards). If ‘TRUE‘ (default) and there is at least one negative value in the data, the minimum ‘area‘ or ‘height‘ value will be subtracted from all ‘area‘ and/or ‘height‘ values by ‘component_name‘, and 100 will then be added to avoid having values below 100. ‘area_ratio‘, ‘height_ratio‘, and ‘area_height_ratio‘ values (and their internal standard equivalent variables) will also be re-calculated.

fix_istds
: Logical value (default ‘TRUE‘) to identify internal standards by regular expression of `"(\.IS$)|(_IS$)|(_d[0-9]+_)|(\(d[0-9]+\))"`.

Value

tibble with the fields appropriately renamed.

Author(s)

Yaoxiang Li

Examples

```
## Not run:
data(sciex_mrm_ms_data)
data_tibble <- load_parse_sciex_txt(
  file_path = "path/to/file.txt",
  return_all_columns = FALSE,
  check_negative_values = TRUE
)

## End(Not run)
```

---

ms1_annotation *Annotate MS1 mass-to-charge values*

Description

Search for MS1 annotations.

Usage

```
ms1_annotation(
  AnnotaData,
  masstole = 0.05,
  toleUnit = 1,
  annotaDB = "metlin",
  ionMode = "pos",
  adducts = NULL
)
```

Arguments

AnnotaData	Numeric vector of observed m/z values.
masstole	Non-negative mass tolerance.
toleUnit	Tolerance unit, ‘1‘ or ‘"Da"‘, or ‘2‘ or ‘"ppm"‘.
annotaDB	Database name, ‘"metlin"‘ or ‘"hmdb"‘.
ionMode	Ion mode, ‘"pos"‘, ‘"neg"‘, or ‘"neu"‘ for neutral masses.
adducts	Optional CEU Mass Mediator adduct names. By default all compatible adducts are searched.

Value

A tibble of CEU Mass Mediator matches.

Author(s)

Yaoxiang Li

---

omics_example_file *Locate a bundled plot example*

Description

Locate a bundled plot example

Usage

```
omics_example_file(relative_path)
```

Arguments

relative_path	Path under the package’s example directory.

Value

The path to the example file.

Author(s)

Yaoxiang Li

omics_gui_capabilities *Available omicsTools workflows*

Description

Lists the workflows that are reachable from the ‘omicsTools‘ Shiny application. The registry is used by automated tests so workflows retain their GUI entry point.

Usage

```
omics_gui_capabilities()
```

Value

A tibble with package, workflow, Shiny tab, and supporting exported functions.

Author(s)

Yaoxiang Li

plot_lcms_features *Plot detected LC-MS features by retention time and m/z*

Description

Plot detected LC-MS features by retention time and m/z

Usage

```
plot_lcms_features(
  data,
  rt_col = "rt",
  mz_col = "mz",
  mode_col = "mode",
  title = "Detected LC-MS features"
)
```

Arguments

data	Feature table.
rt_col, mz_col	Column names for retention time and mass-to-charge.
mode_col	Optional polarity/mode column.
title	Plot title.

Value

A ‘ggplot‘ object.

Author(s)

Yaoxiang Li

`plot_lipid_data_summary`

Plot and Analyze Lipid Class Data

Description

This function loads a specified data file containing lipid measurements, calculates the summary of missing values and relative standard deviation (RSD) for each lipid class, and generates plots for both summaries. It also plots and saves figures showing the distribution of internal standards across samples.

Usage

```
plot_lipid_data_summary(
  file_path,
  output_xlsx = "lipid_classes_with_names.xlsx",
  missing_plot_path = "missing_values_plot.png",
  rsd_plot_path = "rsd_values_plot.png",
  is_plots_dir = "is_plots",
  blank_pattern = "Blank|Control|Neg",
  pooled_pattern = "Pooled QC|Pooled|Pool|PQ",
  nist_pattern = "NIST Plasma|NIST|nist"
)
```

Arguments

`file_path`	The path to the lipid measurement data file (e.g., "area.txt").
`output_xlsx`	Path to save the lipid classes with names as an Excel file. Default is "lipid_classes_with_names.xlsx".
`missing_plot_path`	Path to save the missing data percentage plot. Default is "missing_values_plot.png".
`rsd_plot_path`	Path to save the RSD percentage plot. Default is "rsd_values_plot.png".
`is_plots_dir`	Directory to save internal standard distribution plots. Default is "is_plots".
`blank_pattern`	Regex pattern to identify blank samples in the "Sample ID" column. Default is "Blank|blank".
`pooled_pattern`	Regex pattern to identify pooled QC samples in the "Sample ID" column. Default is "Pooled QC|Pooled|pool|PQC".
`nist_pattern`	Regex pattern to identify NIST QC samples in the "Sample ID" column. Default is "NIST Plasma|NIST|nist".

Value

Plots and saves summary figures of missing percentages, RSD percentages, and internal standard distributions.

Author(s)

Yaoxiang Li

`plot_met_data_summary` *Plot and Analyze Metabolomics Data Summary*

Description

This function loads a specified data file containing metabolomics measurements, identifies internal standards based on a naming convention, and generates plots for the distribution of internal standards across samples. It also saves the internal standard information in an Excel file.

Usage

```
plot_met_data_summary(
  file_path,
  output_xlsx = "internal_standards.xlsx",
  is_plots_dir = "is_plots",
  blank_pattern = "Blank|Control|Neg",
  pooled_pattern = "Pooled QC|Pooled|Pool|PQ",
  nist_pattern = "NIST Plasma|NIST|nist",
  other_special_pattern = "Special Case|Extra Sample|Other QC"
)
```

Arguments

`file_path`	The path to the metabolomics measurement data file (e.g., "area_TM.txt").
`output_xlsx`	The path where the Excel file with internal standards will be saved (default is "internal_standards.xlsx").
`is_plots_dir`	Directory where plots for each internal standard will be saved (default is "is_plots").
`blank_pattern`	Pattern for identifying blank samples in 'Sample ID' column (default is "Blank|BLANK|blank|Control|Neg Control|BLK|Blk").
`pooled_pattern`	Pattern for identifying pooled QC samples in 'Sample ID' column (default is "Pooled QC|Pooled_QC|Pooled|POOL|Pool|pool|QC Mix|Mix QC|Pool QC|PQ|PQC").
`nist_pattern`	Pattern for identifying NIST QC samples in 'Sample ID' column (default is "NIST Plasma|NIST_Plasma|NIST-QC|Reference Plasma|Plasma Ref|NIST_QC|NIST QC|NIST|nist").
`other_special_pattern`	Pattern for identifying other special cases in 'Sample ID' column (default is "Special Case|Extra Sample|NonStandard QC|Other QC").

Value

Saves internal standard plots and an Excel file with the identified internal standards.

Author(s)

Yaoxiang Li

`plot_omics_heatmap` *Plot a feature intensity heatmap*

Description

Plot a feature intensity heatmap

Usage

```
plot_omics_heatmap(
  data,
  id_col = "sample_id",
  feature_cols = NULL,
  scale_features = TRUE,
  max_features = 50L
)
```

Arguments

`data`	A samples-by-features data frame.
`id_col`	Sample ID column.
`feature_cols`	Numeric feature columns; NULL selects all numeric columns.
`scale_features`	Whether to center and standardize each feature.
`max_features`	Maximum number of features, chosen by decreasing variance.

Value

A ggplot heatmap with creator credit.

Author(s)

Yaoxiang Li

plot_omics_pca *Compute and plot principal components for omics data*

Description

Compute and plot principal components for omics data

Usage

```
plot_omics_pca(data, group = NULL, labels = NULL, center = TRUE, scale. = TRUE)
```

Arguments

data	Numeric matrix/data frame with samples in rows.
group	Optional grouping vector.
labels	Optional sample labels.
center, scale.	Passed to [stats::prcomp()].

Value

A ‘ggplot‘ object with the fitted ‘prcomp‘ object attached as the ‘pca‘ attribute.

Author(s)

Yaoxiang Li

plot_peak_list *Plot a tandem mass spectrum*

Description

Plot a tandem mass spectrum

Usage

```
plot_peak_list(peak_list)
```

Arguments

peak_list	Two-column matrix or data frame containing m/z and intensity.

Value

A ‘ggplot‘ object.

Author(s)

Yaoxiang Li

plot_targeted_calibration *Plot a targeted calibration curve*

Description

Plot a targeted calibration curve

Usage

```
plot_targeted_calibration(calibration)
```

Arguments

calibration	An object returned by [fit_targeted_calibration()].

Value

A 'ggplot' object.

Author(s)

Yaoxiang Li

pqn_normalize *Perform Probabilistic Quotient Normalization for intensities*

Description

Perform Probabilistic Quotient Normalization (PQN) for sample intensities. The PQN method determines a dilution factor for each sample by comparing the distribution of quotients between samples and a reference spectrum, followed by sample normalization using this dilution factor. The reference spectrum in this method is the median spectrum of all samples.

Usage

```
pqn_normalize(data, reference_rows = seq_len(nrow(data)))
```

Arguments

data	A data frame containing the sample data. The first column should contain the sample identifiers, and the rest of the columns contain the peaks to be normalized.
reference_rows	Rows used to calculate the reference spectrum.

Value

A data frame with the first column as the sample identifiers and the rest of the columns containing the normalized peak intensities.

Author(s)

Yaoxiang Li

References

Dieterle, F., Ross, A., Schlotterbeck, G., & Senn, H. (2006). Probabilistic quotient normalization as robust method to account for dilution of complex biological mixtures. Application in 1H NMR metabonomics. Analytical chemistry, 78(13), 4281-4290.

Examples

```
data <- data.frame(
  Sample = paste0("S", 1:4),
  feature_a = c(10, 11, 12, 13),
  feature_b = c(20, 18, 16, 14)
)

# Display the first few rows of the original data
print(head(data))

# Apply the pqn_normalize function
normalized_data <- pqn_normalize(data)

# Display the first few rows of the normalized data
print(head(normalized_data))


# Write the normalized data to a new CSV file
readr::write_csv(normalized_data, paste0(tempdir(), "/normalized_data.csv"))
```

---

predict_targeted_calibration

Predict concentrations with a targeted calibration model

Description

Predict concentrations with a targeted calibration model

Usage

```
predict_targeted_calibration(
  calibration,
  new_data,
  response_col = calibration$response_col,
  output_col = "predicted_concentration"
)
```

Arguments

calibration	An object returned by [fit_targeted_calibration()].
new_data	Data frame containing an instrument-response column.
response_col	Optional response-column override.
output_col	Name of the appended concentration column.

Value

'new_data' with predicted concentrations appended.

Author(s)

Yaoxiang Li

---

process_mrm_duplicates

Process All MRM Transitions for Duplicates

Description

This function takes a tibble containing MRM transition data, processes each sample_id separately to detect duplicates, and adds a column 'MRM_Duplicate_Flag' indicating if a row is a duplicate based on the criteria: same polarity, less than a 1-minute retention time difference, and the same MRM transition (Q1/Q3).

Usage

```
process_mrm_duplicates(
  mrm_data,
  sample_name_col = "data_filename",
  sample_id_col = "sample_id",
  polarity_col = "polarity",
  retention_time_col = "retention_time",
  mass_info_col = "Mass Info",
  component_name_col = "component_name"
)
```

Arguments

mrm_data	A tibble containing the MRM transition data.
sample_name_col	Name of the column containing sample name information.
sample_id_col	Name of the column containing sample ID information.
polarity_col	Name of the column containing polarity information.
retention_time_col	Name of the column containing retention time information.
mass_info_col	Name of the column containing mass information.
component_name_col	Name of the column containing component name information.

Value

The original tibble with an added 'MRM_Duplicate_Flag' column.

Author(s)

Yaoxiang Li

Examples

```
## Not run:
mrm_data <- tibble::tribble(
 ~`data_filename`, ~`sample_id`, ~polarity, ~`retention_time`, ~`Mass Info`, ~`component_name`,
  "Sample1", "ID1", "Positive", 1.95, "61.0 / 44.0", "Urea_pos",
  "Sample1", "ID1", "Positive", 8.34, "206.0 / 189.0", "Lipoamide_pos",
  "Sample2", "ID2", "Positive", 2.18, "339.1 / 110.0", "AICAR_pos",
  "Sample2", "ID2", "Positive", 1.76, "175.1 / 70.0", "Arginine_pos")
processed_data <- process_mrm_duplicates(
  mrm_data,
  "data_filename",
  "sample_id",
  "polarity",
  "retention_time",
  "Mass Info",
  "component_name"
)
print(processed_data)

## End(Not run)
```

process_omics_data *Process an omics feature table*

Description

Runs validated preprocessing, optional blank filtering and group comparison, and records the settings and sample/feature counts at each step. QC-RLSC uses injection order within each batch and requires three QC samples per batch. Table order is used when no injection-order column is supplied.

Usage

```
process_omics_data(
  data,
  id_col = "sample_id",
  metadata_cols = NULL,
  sample_info = NULL,
  feature_info = NULL,
  sample_type_col = NULL,
  batch_col = NULL,
  order_col = NULL,
  blank_filter = FALSE,
  blank_value = "Blank",
  study_value = "Sample",
  ratio = 3,
  missing = 0.2,
  imputation = "half_min",
  normalization = "none",
  qc_value = "QC",
  qc_rsd = NULL,
  group_col = NULL,
  reference = NULL,
  comparison = NULL,
  method = "welch"
)
```

Arguments

data	A samples-by-features data frame including sample IDs.
id_col	Sample ID column.
metadata_cols	Columns excluded from intensity processing. NULL also identifies nonnumeric columns as metadata.
sample_info	Optional sample metadata with the same sample IDs.
feature_info	Optional feature metadata with feature_id and optional mz and rt columns. Retention times are in seconds.

`sample_type_col`
: Column identifying blanks and study samples.

`batch_col`
: Optional batch column. Every batch needs blanks and study samples.

`order_col`
: Optional numeric injection-order column, unique within batch.

`blank_filter`
: Whether to filter against blank samples.

`blank_value`, `study_value`
: Labels for blank and study samples.

`ratio`
: Minimum study-to-blank intensity ratio, at least one.

`missing`
: Maximum missing fraction per feature.

`imputation`
: One of none, half_min, mean, median, or mode.

`normalization`
: One of none, qc_rlsc, or pqn.

`qc_value`
: Label identifying QC samples.

`qc_rsd`
: Optional maximum QC relative standard deviation.

`group_col`
: Group column.

`reference`, `comparison`
: Reference and comparison group labels.

`method`
: Welch t-test or Wilcoxon rank-sum test.

Value

A list with original/processed tables, metadata, QC summaries, comparison results, blank audit, processing steps, and settings.

Author(s)

Yaoxiang Li

---

qc_normalize *QC-RLSC Normalize function*

Description

This function performs normalization on the input data matrix using the robust loess signal correction (RLSC) method. Normalization is based on Quality Control (QC) samples in the data.

Usage

```
qc_normalize(data, qc_label = "QC", sample_id_col = "sample_id")
```

Arguments

data	A data frame containing the sample data. The first column should contain the sample identifiers by default named 'sample_id', and the rest of the columns contain the peaks to be normalized. QC samples should be indicated in the sample identifiers with 'QC'.
qc_label	A string indicating the label used for QC samples. Default is 'QC'.
sample_id_col	A string indicating the column name used for sample identifiers. Default is 'sample_id'.

Value

A data frame with the first column as the sample identifiers and the rest of the columns containing the normalized peak intensities.

Author(s)

Yaoxiang Li

Examples

```
data <- data.frame(
  Sample = c("QC_1", "S1", "QC_2", "S2", "QC_3", "QC_4"),
  feature_a = c(10, 11, 12, 13, 14, 15),
  feature_b = c(20, 19, 18, 17, 16, 15)
)

# Display the first few rows of the original data
print(head(data))

# Apply the qc_rlsc_normalize function
normalized_data <- qc_normalize(data, qc_label = "QC", sample_id_col = "Sample")

# Display the first few rows of the normalized data
print(head(normalized_data))


# Write the normalized data to a new CSV file
readr::write_csv(normalized_data, paste0(tempdir(), "/normalized_data.csv"))
```

---

run_app *Run the omicsTools application*

Description

Run the omicsTools application

Usage

```
run_app(
  onStart = NULL,
  options = list(),
  enableBookmarking = NULL,
  uiPattern = "/"
)
```

Arguments

onStart	A function that will be called before the app is actually run. This is only needed for shinyAppObj, since in the shinyAppDir case, a global.R file can be used for this purpose.
options	Named options that should be passed to the runApp call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify width and height parameters which provide a hint to the embedding environment about the ideal height/width for the app.
enableBookmarking	Can be one of "url", "server", or "disable". The default value, NULL, will respect the setting from any previous calls to enableBookmarking(). See enableBookmarking() for more information on bookmarking your app.
uiPattern	A regular expression that will be applied to each GET request to determine whether the ui should be used to handle the request. Note that the entire request path must match the regular expression in order for the match to be considered successful.

Value

A Shiny application.

Author(s)

Yaoxiang Li

---

save_msp *Write a spectrum in MSP format*

Description

Write a spectrum in MSP format

Usage

```
save_msp(
  file_path,
  netcdf_file,
  formula = "Unknown",
  comments = "Generated by omicsTools",
  precursor_ion,
  precursor_type = "Unknown",
  instrument_type = "Unknown",
  instrument = "Unknown",
  peak_list
)
```

Arguments

`file_path`	Output '.MSP' path.
`netcdf_file`	Source raw-file name.
`formula`	Molecular formula.
`comments`	Free-text comments.
`precursor_ion`	Precursor m/z.
`precursor_type`	Precursor adduct.
`instrument_type, instrument`	Instrument metadata.
`peak_list`	Two-column matrix or data frame.

Value

Invisibly, the lines written to 'file_path'.

Author(s)

Yaoxiang Li

`transpose_df` *Transpose a feature table using its first column as identifiers*

Description

Transpose a feature table using its first column as identifiers

Usage

```
transpose_df(df)
```

Arguments

df	Data frame whose first column contains identifiers.

Value

A tibble with original measurement columns as rows and original identifiers as columns.

Author(s)

Yaoxiang Li

---

validate_omics_data *Validate and align omics tables*

Description

Intensity tables have one row per sample and one column per feature. Sample metadata are aligned by ID. Feature metadata use a feature_id column. Missing intensities are allowed; negative or infinite intensities are rejected.

Usage

```
validate_omics_data(
  data,
  id_col = "sample_id",
  metadata_cols = NULL,
  sample_info = NULL,
  feature_info = NULL
)
```

Arguments

data	A samples-by-features data frame including sample IDs.
id_col	Sample ID column.
metadata_cols	Columns excluded from intensity processing. NULL also identifies nonnumeric columns as metadata.
sample_info	Optional sample metadata with the same sample IDs.
feature_info	Optional feature metadata with feature_id and optional mz and rt columns. Retention times are in seconds.

Value

A list containing aligned data, feature and metadata column names, and aligned feature metadata.

Author(s)

Yaoxiang Li

`write_omics_report` *Write a self-contained workflow QC report*

Description

Includes processing settings, counts, missingness, QC RSD, PCA before and after processing, blank-filter decisions, and group comparisons.

Usage

```
write_omics_report(result, file)
```

Arguments

`result`	A result from process_omics_data().
`file`	Output HTML file.

Value

Invisibly, the output file path.

Author(s)

Yaoxiang Li

Index

calculate_qc_rsd, 3
check_and_sort_columns, 4
check_match, 5
combine_logical_tibbles, 5
compare_omics_groups, 6
convert_mrm_data, 7

detect_duplicates, 8
detect_sample_outliers, 9

enableBookmarking(), *36*
export_omics_results, 10

filter_blank_features, 10
filter_missing_value, 11
filter_qc_rsd, 12
fit_targeted_calibration, 13
flag_underexpressed_features, 14

generate_process_report, 15
generate_sample_metadata, 16

handle_missing_values, 17

impute_half_min, 18
internal_standard_normalize, 19

load_data, 20
load_omics_data, 20
load_parse_sciex_txt, 21

ms1_annotation, 22

omics_example_file, 23
omics_gui_capabilities, 24

plot_lcms_features, 24
plot_lipid_data_summary, 25
plot_met_data_summary, 26
plot_omics_heatmap, 27
plot_omics_pca, 28
plot_peak_list, 28
plot_targeted_calibration, 29
pqn_normalize, 29
predict_targeted_calibration, 30
process_mrm_duplicates, 31
process_omics_data, 33

qc_normalize, 34

run_app, 35

save_msp, 36

transpose_df, 37

validate_omics_data, 38

write_omics_report, 39