# Package 'omicsTools'

May 9, 2026

**Title** Omics Data Process Toolbox

**Version** 1.1.7

**Date** 2025-12-16

**Description** Processing and analyzing omics data from genomics, transcriptomics, proteomics, and metabolomics platforms. It provides functions for preprocessing, normalization, visualization, and statistical analysis, as well as machine learning algorithms for predictive modeling. 'omicsTools' is an essential tool for researchers working with high-throughput omics data in fields such as biology, bioinformatics, and medicine.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)

**Depends** R (>= 4.1.0)

**Imports** bs4Dash, cli, config (>= 0.3.1), dbscan, dplyr, DT, forcats, ggplot2, ggpubr, ggrepel, ggsci, ggvenn, golem (>= 0.3.5), janitor, magrittr, matrixStats, grDevices, methods, moments, outliers, pheatmap, purrr, RColorBrewer, readxl, readr, rlang, stats, shiny (>= 1.7.2), shinycssloaders, shinyWidgets, stringr, tibble, tidyr, utils, viridis, tools

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Suggests** Biobase, pvca, testthat (>= 3.0.0), knitr, rmarkdown, Boruta, randomForest, caret, recipes, parsnip, workflows, rsample, tune, dials, doParallel, vip, GOplot, openxlsx, UpSetR, zip

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Yaoxiang Li [cre, aut] (ORCID: <https://orcid.org/0000-0001-9200-1016>)

**Maintainer** Yaoxiang Li <liyaoxiang@outlook.com>

**Repository** CRAN

**Date/Publication** 2025-12-16 15:40:02 UTC

# Contents

## calculate_cooks_distance *Calculate Cook's Distance*

### Description

Calculate Cook's Distance

### Usage

```
calculate_cooks_distance(values)
```

### Arguments

| | |
|---|---|
| values | A numeric vector of values for which to calculate Cook's Distance. |

### Value

A numeric vector of Cook's Distance values.

### Examples

```
values <- rnorm(100)
calculate_cooks_distance(values)
```

## calculate_lof *Calculate Local Outlier Factor (LOF)*

### Description

Calculate Local Outlier Factor (LOF)

### Usage

```
calculate_lof(values, k = 5)
```

### Arguments

| | |
|---|---|
| values | A numeric vector of values for which to calculate LOF. |
| k | The number of neighbors to use for calculating LOF. Defaults to 5. |

### Value

A numeric vector of LOF values.

### Examples

```
values <- rnorm(100)
calculate_lof(values)
```

## calculate_measures *Calculate Measures for Each Feature*

### Description

Calculate Measures for Each Feature

### Usage

```
calculate_measures(object)

## S4 method for signature 'OmicsData'
calculate_measures(object)
```

### Arguments

| | |
|---|---|
| object | An OmicsData object. |

### Value

The OmicsData object with calculated measures for each feature.

### Examples

```
omics_data <- createOmicsData()
omics_data <- calculate_measures(omics_data)
```

## 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.

### 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")
```

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)
```

---

`convert_to_binary_matrix` *Convert to Binary Matrix for UpSetR*

---

## Description

Convert to Binary Matrix for UpSetR

## Usage

```
convert_to_binary_matrix(upset_data)

## S4 method for signature 'data.frame'
convert_to_binary_matrix(upset_data)
```

## Arguments

| | |
|---|---|
| `upset_data` | The data frame prepared for UpSet plot. |

## Value

A binary matrix for UpSetR.

## Examples

```
omics_data <- createOmicsData()
omics_data <- calculate_measures(omics_data)
omics_data <- flag_anomalies(omics_data)
upset_data <- prepare_upset_data(omics_data)
upset_matrix <- convert_to_binary_matrix(upset_data)
```

## createOmicsData *Constructor for OmicsData*

### Description

Constructor for OmicsData

### Usage

```
createOmicsData(data = NULL, n_samples = 100, n_features = 2000)
```

### Arguments

| | |
|---|---|
| data | The data frame containing the data. If not provided, synthetic data will be generated. |
| n_samples | The number of samples to generate if data is not provided. Defaults to 100. |
| n_features | The number of features to generate if data is not provided. Defaults to 2000. |

### Value

An OmicsData object.

### Examples

```
omics_data <- createOmicsData()
user_data <- generate_data_with_anomalies()
omics_data <- createOmicsData(data = user_data)
```

## define_thresholds *Define Anomaly Thresholds*

### Description

Define Anomaly Thresholds

### Usage

```
define_thresholds(
  skewness = 2.5,
  kurtosis = 10,
  shapiro_p = 1e-10,
  cooks_distance = 1,
  lof = 15
)
```

### Arguments

| | |
|---|---|
| skewness | The threshold for skewness. Defaults to 2.5. |
| kurtosis | The threshold for kurtosis. Defaults to 10. |
| shapiro_p | The threshold for Shapiro-Wilk test p-value. Defaults to 1e-10. |
| cooks_distance | The threshold for Cook's Distance. Defaults to 1. |
| lof | The threshold for Local Outlier Factor. Defaults to 15. |

### Value

A list of thresholds for anomaly detection.

### Examples

```
thresholds <- define_thresholds()
thresholds <- define_thresholds(skewness = 3, kurtosis = 8)
```

---

## 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)
```

---

# ensure_enough_sets_for_upset

*Ensure There Are Enough Sets for UpSet Plot*

---

## Description

Ensure There Are Enough Sets for UpSet Plot

## Usage

```
ensure_enough_sets_for_upset(upset_matrix)

## S4 method for signature 'data.frame'
ensure_enough_sets_for_upset(upset_matrix)
```

## Arguments

| | |
|---|---|
| upset_matrix | The binary matrix for UpSetR. |

### Examples

```
omics_data <- createOmicsData()
omics_data <- calculate_measures(omics_data)
omics_data <- flag_anomalies(omics_data)
upset_data <- prepare_upset_data(omics_data)
upset_matrix <- convert_to_binary_matrix(upset_data)
ensure_enough_sets_for_upset(upset_matrix)
```

---

## flag_anomalies *Flag Anomalies*

---

### Description

Flag Anomalies

### Usage

```
flag_anomalies(object)

## S4 method for signature 'OmicsData'
flag_anomalies(object)
```

### Arguments

| | |
|---|---|
| object | An OmicsData object. |

### Value

The OmicsData object with flagged anomalies.

### Examples

```
omics_data <- createOmicsData()
omics_data <- calculate_measures(omics_data)
omics_data <- flag_anomalies(omics_data)
```

---

## 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_data_with_anomalies
*Generate High-Dimensional Data with Anomalies*

## Description

Generate High-Dimensional Data with Anomalies

## Usage

```
generate_data_with_anomalies(n_samples = 100, n_features = 2000)
```

## Arguments

| | |
|---|---|
| n_samples | The number of samples to generate. Defaults to 100. |
| n_features | The number of features to generate. Defaults to 2000. |

## Value

A data frame containing the generated data with anomalies.

## Examples

```
data <- generate_data_with_anomalies()
```

---

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

---

`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)
```

---

# initialize_results_df *Initialize Results Data Frame*

---

## Description

Initialize Results Data Frame

## Usage

```
initialize_results_df(data)
```

## Arguments

| | |
|---|---|
| data | The data frame containing the generated data. |

## Value

A data frame initialized with columns for anomaly measures.

## Examples

```
data <- generate_data_with_anomalies()
anomaly_measures <- initialize_results_df(data)
```

## 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.

### 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_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 *MS1 Annotation*

### Description

MS1 Annotation

### Usage

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

### Arguments

| | |
|---|---|
| AnnotaData | Numeric vector of m/z values (or an object coercible to numeric). |
| masstole | Numeric mass tolerance. Default is 0.05. |
| toleUnit | Unit for tolerance: 1 = Da, 2 = ppm. |
| annotaDB | Annotation database, e.g. "metlin" or "hmdb". |
| ionMode | Ion mode: "pos", "neg", or "neu". |
| adducts | Optional character vector of adduct names to consider. |

### Value

An annotation result object (typically a data frame) for the provided m/z list.

## OmicsData-class *OmicsData Class*

### Description

A class to represent and analyze high-dimensional omics data.

### Slots

`data` The data frame containing the generated data with anomalies.

`anomaly_measures` The data frame containing the calculated anomaly measures.

`thresholds` The list of thresholds for anomaly detection.

perform_batch_assessment *Perform Principal Variance Component Analysis for Batch Effect Assessment*

## Description

This function performs Principal Variance Component Analysis to assess batch effects in the dataset.

## Usage

```
perform_batch_assessment(data_matrix, sample_info, batch_effects, threshold)
```

## Arguments

| | |
|---|---|
| data_matrix | A data frame or matrix where rows represent features and columns represent samples. |
| sample_info | A data frame containing sample information with rows matching the columns of 'data_matrix'. |
| batch_effects | A character vector of column names in 'sample_info' that represent batch effects. |
| threshold | A numeric value between 0 and 1 to specify the PVCA threshold. |

## Value

A PVCA object containing the results of the batch effect assessment.

## Examples

```
# Example data
set.seed(123)
data_matrix <- data.frame(
  sample1 = rnorm(100),
  sample2 = rnorm(100),
  sample3 = rnorm(100),
  sample4 = rnorm(100),
  sample5 = rnorm(100),
  sample6 = rnorm(100),
  sample7 = rnorm(100),
  sample8 = rnorm(100),
  sample9 = rnorm(100),
  sample10 = rnorm(100)
)
rownames(data_matrix) <- paste0("feature", 1:100)

sample_info <- data.frame(
  dose = c(0, 1, 2, 1, 0, 2, 1, 0, 2, 1),
  time = c(-1, 1, 25, -1, 1, 25, -1, 1, 25, -1),
```

```
  batch = rep(c("A", "B", "C"), length.out = 10)
)
rownames(sample_info) <- colnames(data_matrix)

# Perform Batch Effect Assessment
if (requireNamespace("Biobase", quietly = TRUE) &&
    requireNamespace("pvca", quietly = TRUE)) {
  pvca_results <- perform_batch_assessment(
    data_matrix,
    sample_info,
    c("batch", "dose", "time"),
    0.6
  )
}
```

---

`perform_feature_selection` *Perform Feature Selection*

---

## Description

This function performs feature selection using various methods such as LASSO, Elastic Net, Ridge regression, and Boruta. It outputs selected features and variable importance plots.

## Usage

```
perform_feature_selection(
  group_info,
  features,
  id_col_group,
  id_col_features,
  group_col,
  outlier_col = NULL,
  outlier_vals = c("No"),
  group_vals = c("No", "Yes"),
  method = c("lasso", "elastic_net", "ridge", "boruta"),
  mixture = 0.5,
  penalty_vals = 50,
  seed = 1234,
  output_dir = "output"
)
```

## Arguments

| | |
|---|---|
| `group_info` | A data frame containing group information. |
| `features` | A data frame containing feature data. |
| `id_col_group` | Column name in ‘group_info‘ to join with ‘features‘. |

| | |
|---|---|
| id_col_features | Column name in ‘features‘ to join with ‘group_info‘. |
| group_col | Column name indicating the group information. |
| outlier_col | (Optional) Column name for identifying outliers. |
| outlier_vals | (Optional) Values indicating non-outliers. |
| group_vals | A vector of length 2 indicating the values for group comparison. |
| method | The feature selection method to use: "lasso", "elastic_net", "ridge", "boruta". |
| mixture | (Optional) The mixture parameter for Elastic Net, default is 0.5. |
| penalty_vals | (Optional) Number of penalty values to try for tuning, default is 50. |
| seed | (Optional) Random seed for reproducibility, default is 1234. |
| output_dir | (Optional) Directory to save output files, default is "output". |

## Value

A tibble containing selected features and variable importances.

---

`pieDraw` *Plot PVCA results (pie chart)*

---

## Description

Plot PVCA results (pie chart)

## Usage

```
pieDraw(pvcaobj)
```

## Arguments

| | |
|---|---|
| pvcaobj | A PVCA object returned by ‘perform_batch_assessment()‘. |

## Value

A ggplot object (invisibly).

plot_distribution_measures
*Plot Distribution Measures*

## Description

Plot Distribution Measures

## Usage

```
plot_distribution_measures(object)

## S4 method for signature 'OmicsData'
plot_distribution_measures(object)
```

## Arguments

| | |
|---|---|
| object | An OmicsData object. |

## Examples

```
omics_data <- createOmicsData()
omics_data <- calculate_measures(omics_data)
omics_data <- flag_anomalies(omics_data)
plot_distribution_measures(omics_data)
```

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\|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_sample_measures *Plot Sample Measures*

---

### Description

Plot Sample Measures

### Usage

```
plot_sample_measures(object)

## S4 method for signature 'OmicsData'
plot_sample_measures(object)
```

### Arguments

| | |
|---|---|
| object | An OmicsData object. |

### Examples

```
omics_data <- createOmicsData()
omics_data <- calculate_measures(omics_data)
omics_data <- flag_anomalies(omics_data)
plot_sample_measures(omics_data)
```

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)
```

## 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. |

## 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

```
# Load the CSV data
data_file <- system.file("extdata", "example2.csv", package = "omicsTools")
data <- readr::read_csv(data_file)

# 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"))
```

---

`prepare_upset_data` *Prepare Data for UpSet Plot*

---

## Description

Prepare Data for UpSet Plot

## Usage

```
prepare_upset_data(object)

## S4 method for signature 'OmicsData'
prepare_upset_data(object)
```

## Arguments

| | |
|---|---|
| `object` | An OmicsData object. |

## Value

A data frame prepared for UpSet plot.

## Examples

```
omics_data <- createOmicsData()
omics_data <- calculate_measures(omics_data)
omics_data <- flag_anomalies(omics_data)
upset_data <- prepare_upset_data(omics_data)
```

---

`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)
```

---

`pvcaDraw` *Plot PVCA results (bar chart)*

---

## Description

Plot PVCA results (bar chart)

## Usage

```
pvcaDraw(pvcaobj)
```

## Arguments

| | |
|---|---|
| `pvcaobj` | A PVCA object returned by 'perform_batch_assessment()'. |

## Value

A ggplot object (invisibly).

---

`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

```
# Load the CSV data
data_file <- system.file("extdata", "example2.csv", package = "omicsTools")
data <- readr::read_csv(data_file)

# 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 Shiny Application*

---

## Description

Run the Shiny 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. |
| ... | arguments to pass to golem_opts. See '?golem::get_golem_options' for more details. |

### Value

No return value, called for launch the application.

---

## transpose_df *Transpose DataFrame*

---

### Description

This function transposes a data frame by converting rows to columns and columns to rows. The first column is assumed to contain the variable names, and the remaining columns contain the values.

### Usage

```
transpose_df(df)
```

### Arguments

| | |
|---|---|
| df | A data frame to be transposed. |

### Value

A transposed data frame.

## Examples

```
# Example usage:
df <- data.frame(
  ID = c("A", "B", "C"),
  Var1 = c(1, 2, 3),
  Var2 = c(4, 5, 6)
)
transpose_df(df)
```

# Index