Package: LIPIDIFy (via r-universe)

November 25, 2026

Type Package

Title Comprehensive Lipidomics Data Analysis with Interactive Visualization

Version 1.0.0

Description Provides a comprehensive toolkit for end-to-end lipidomics data analysis, including missing value imputation, batch effect correction, normalization, differential abundance analysis using limma and edgeR, gene set enrichment analysis, and extensive visualization capabilities. Lipid names are automatically classified by class, subclass, and fatty-acid saturation. Features both an interactive Shiny interface for bench biologists and fully scriptable R functions for bioinformaticians. Supports flexible custom lipid classification schemes and user-defined enrichment sets.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Depends R (>= 4.5.0)

Imports shiny (>= 1.8.0), shinydashboard, ggplot2 (>= 3.5.0), ggrepel, plotly, DT, tidyr, dplyr, scales, gridExtra, grid, pheatmap, limma, edgeR, fgsea, FactoMineR, pls, openxlsx, rmarkdown, stringr, withr, utils, stats, grDevices

Suggests BiocStyle, knitr, testthat (>= 3.0.0), impute, sva, vsn, SummarizedExperiment, tinytex, shinytest2, chromote

VignetteBuilder knitr

biocViews Software, Lipidomics, MassSpectrometry, Normalization, Preprocessing, DifferentialExpression, GeneSetEnrichment, Visualization, ShinyApps, MultipleComparison, BatchEffect, QualityControl, DataImport, Classification

URL https://github.com/fayrouzhammal/LIPIDIFy

BugReports https://github.com/fayrouzhammal/LIPIDIFy/issues

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Config/pak/sysreqs
cmake libfreetype6-dev make libicu-dev libpng-dev libuv1-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:56:59 UTC

RemoteUrl https://github.com/bioc/LIPIDIFy

RemoteRef RELEASE_3_24

RemoteSha fb074d2921632230919b4f9d796163b9e525cb5f

Contents

apply_normalizations . . . 3
classify_lipids . . . 4
correct_batch_effects . . . 4
create_default_contrasts . . . 5
create_enrichment_barplot . . . 6
create_enrichment_dotplot . . . 7
create_heatmap_robust . . . 8
create_lipid_expression_barplot . . . 9
create_pca_plot_with_ellipses . . . 10
create_pipeline_plot . . . 11
create_plsda_plot_with_ellipses . . . 12
create_volcano_plot_labeled . . . 13
determine_saturation . . . 14
export_classification . . . 14
fix_sample_alignment . . . 15
generate_example_data . . . 15
get_imputation_descriptions . . . 16
get_imputation_methods . . . 16
get_normalization_descriptions . . . 17
get_normalization_methods . . . 17
impute_missing_values . . . 18
launch_lipidomics_app . . . 18
load_custom_classification . . . 19
load_custom_enrichment_sets . . . 20
load_lipidomics_data . . . 21
load_lipidomics_data_from_df . . . 22
normalize_log2median . . . 22
normalize_mean . . . 23
normalize_median . . . 24
normalize_pqn . . . 24
normalize_quantile . . . 25
normalize_tic . . . 25
normalize_vsn . . . 26

perform_differential_analysis 27
perform_enrichment_analysis 28
perform_pca 29
perform_plsda 30
test_saturation_classification 30
visualize_raw_data 31
visualize_raw_data_improved 31

Index **33**

`apply_normalizations` *Apply a Sequence of Normalization Methods*

Description

Applies normalization methods in the order supplied, passing the output of each step as the input to the next.

Usage

```
apply_normalizations(data, methods)
```

Arguments

`data`	Numeric matrix with samples in rows and lipids in columns.
`methods`	Character vector of method names as returned by `get_normalization_methods()`.

Details

Method names are matched exactly against `get_normalization_methods()`; an unrecognised name leaves the data unchanged. `"VSN"` and `"Log2Median"` are separate methods – `"VSN"` calls `normalize_vsn` (Bioconductor **vsn**) and `"Log2Median"` calls `normalize_log2median`. If **vsn** is not installed, `"VSN"` raises an error; it never falls back to another method.

Value

Normalized numeric matrix of the same dimensions as `data`.

See Also

`normalize_vsn`, `normalize_log2median`.

Examples

```
m <- matrix(rlnorm(60, 8, 1), nrow = 6, ncol = 10)
apply_normalizations(m, c("TIC", "Log2"))
```

classify_lipids *Classify Lipids Based on Their Names*

Description

Classifies a vector of lipid names into lipid group, type, and saturation category using regular-expression pattern matching.

Usage

```
classify_lipids(lipid_names)
```

Arguments

lipid_names	Character vector of lipid names.

Value

Data frame with columns `Lipid`, `LipidGroup`, `LipidType`, and `Saturation`.

Examples

```
classify_lipids(c("PC 16:0_18:1", "TG 16:0_18:1_20:4", "Cer 16:0"))
```

correct_batch_effects *Correct Batch Effects from a Normalised Lipidomics Matrix*

Description

Removes known technical batch effects while preserving biological signal. Batch correction should be applied **after** normalisation.

Usage

```
correct_batch_effects(
  data_matrix,
  metadata,
  batch_column,
  group_column = "Sample Group",
  method = "limma"
)
```

Arguments

data_matrix	Numeric matrix with samples as rows and lipids as columns (typically the output of apply_normalizations).
metadata	Data frame of sample metadata aligned with data_matrix (same row order).
batch_column	Character. Name of the column in metadata containing batch labels.
group_column	Character. Name of the biological group column to protect from removal (default "Sample Group"). Pass NULL to skip group protection.
method	One of "limma" (default) or "combat".

Details

Two methods are supported:

"limma" Uses removeBatchEffect. Requires only the limma package (already a dependency). Suitable for most experimental designs.

"combat" Uses sva::ComBat with parametric empirical Bayes adjustment. Requires the **sva** Bioconductor package (BiocManager::install("sva")). Generally more robust when batch effects are large.

Value

Batch-corrected numeric matrix of the same dimensions as data_matrix.

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
d$metadata$Batch <- rep(c("Batch1", "Batch2"), each = 10)
corrected <- correct_batch_effects(norm, d$metadata,
  batch_column = "Batch"
)
dim(corrected)
```

---

create_default_contrasts

Create Default Contrasts

Description

Creates default pairwise contrasts from group levels. If the levels contain special characters that are invalid for limma/edgeR, they should already be sanitized before calling this function.

Usage

```
create_default_contrasts(group_levels)
```

Arguments

group_levels	Character vector of group level names (e.g., c("Control","Treatment","Resistant"))

Value

Character vector of limma-style contrast strings (e.g., "Treatment - Control")

Examples

```
create_default_contrasts(c("A", "B", "C"))
```

---

create_enrichment_barplot

Create an Enrichment Barplot

Description

Create an Enrichment Barplot

Usage

```
create_enrichment_barplot(
  enrichment_data,
  title = "Enrichment Analysis",
  max_pathways = 15
)
```

Arguments

enrichment_data	Data frame of fgsea results.
title	Plot title.
max_pathways	Maximum number of pathways (top by p-value).

Value

A ggplot2 object.

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
cls <- classify_lipids(colnames(norm))
res <- perform_differential_analysis(norm, d$metadata, "Sample Group",
  contrasts_list = NULL, method = "limma"
)
enrich <- perform_enrichment_analysis(res$results, cls, min_set_size = 3)
p <- create_enrichment_barplot(enrich[[1]][["LipidGroup"]])
print(p)
```

`create_enrichment_dotplot` *Create an Enrichment Dotplot*

Description

Create an Enrichment Dotplot

Usage

```
create_enrichment_dotplot(
  enrichment_data,
  title = "Enrichment Analysis",
  max_pathways = 15
)
```

Arguments

`enrichment_data`	Data frame of fgsea results.
`title`	Plot title.
`max_pathways`	Maximum number of pathways (top by p-value).

Value

A `ggplot2` object.

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
cls <- classify_lipids(colnames(norm))
res <- perform_differential_analysis(norm, d$metadata, "Sample Group",
  contrasts_list = NULL, method = "limma"
)
enrich <- perform_enrichment_analysis(res$results, cls, min_set_size = 3)
p <- create_enrichment_dotplot(enrich[[1]][["LipidGroup"]])
print(p)
```

`create_heatmap_robust` *Create a Robust Heatmap of Top Variable Features*

Description

Create a Robust Heatmap of Top Variable Features

Usage

```
create_heatmap_robust(
  data_matrix,
  metadata,
  group_column = "Sample Group",
  top_n = 50,
  classification_data = NULL,
  title = "Heatmap"
)
```

Arguments

`data_matrix`	Numeric matrix (features as rows, samples as columns).
`metadata`	Metadata data frame (samples as rows).
`group_column`	Name of the group column in `metadata`.
`top_n`	Maximum number of features to display.
`classification_data`	Optional classification data frame for row annotation (must have a `Lipid` column).
`title`	Heatmap title.

Value

A `pheatmap` object, or a `ggplot2` error plot.

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
create_heatmap_robust(t(norm), d$metadata, "Sample Group", top_n = 10)
```

`create_lipid_expression_barplot` *Create a Lipid Expression Barplot Ordered by Group*

Description

Produces per-lipid barplots coloured by sample group. Samples are automatically sorted by group (then alphabetically within group) for a cleaner visual.

Usage

```
create_lipid_expression_barplot(
  data_matrix,
  metadata,
  selected_lipids,
  selected_samples = NULL,
  selected_groups = NULL,
  group_column = "Sample Group",
  data_type = "normalized"
)
```

Arguments

`data_matrix`	Numeric matrix (samples as rows, lipids as columns).
`metadata`	Metadata data frame.
`selected_lipids`	Character vector of lipid names to plot.
`selected_samples`	Optional character vector of sample names to retain.
`selected_groups`	Optional character vector of group names to retain.
`group_column`	Name of the group column in `metadata`.
`data_type`	Label for the y-axis subtitle ("raw" or "normalized").

Value

A single `ggplot2` object (one lipid) or a named list of `ggplot2` objects (multiple lipids).

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
p <- create_lipid_expression_barplot(
  d$numeric_data, d$metadata,
  selected_lipids = colnames(d$numeric_data)[1],
  group_column = "Sample Group"
)
print(p)
```

`create_pca_plot_with_ellipses` *Create a PCA Plot with Optional Confidence or Visual Ellipses*

Description

The colour/fill legends are merged so that ellipses do not introduce duplicate legend keys. Sample labels are kept separate from group labels.

Usage

```
create_pca_plot_with_ellipses(
  pca_data,
  variance_explained,
  ellipse_type = "none",
  confidence_level = 0.95,
  title = "PCA Analysis",
  show_sample_labels = FALSE
)
```

Arguments

`pca_data`	Data frame with columns `PC1`, `PC2`, `Group` and (optionally) `Sample` for point labels.
`variance_explained`	Numeric vector of length ≥ 2 with the percent variance explained by PC1 and PC2.
`ellipse_type`	One of `"none"`, `"confidence"`, or `"visual"`.
`confidence_level`	Numeric confidence level for `"confidence"` ellipses (default `0.95`).
`title`	Plot title.
`show_sample_labels`	Logical. If `TRUE`, sample names are shown as text labels next to each point.

Value

A `ggplot2` object.

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
pca_res <- perform_pca(norm, d$metadata, "Sample Group")
p <- create_pca_plot_with_ellipses(pca_res$pca_data, pca_res$variance_explained)
print(p)
```

`create_pipeline_plot` *Quick QC Plot for a Normalized Data Matrix*

Description

Quick QC Plot for a Normalized Data Matrix

Usage

```
create_pipeline_plot(
  data_matrix,
  title = "Normalization Pipeline",
  metadata = NULL,
  group_column = "Sample Group",
  plot_type = "boxplot",
  view_mode = "sample",
  top_n = NULL
)
```

Arguments

`data_matrix`	Numeric matrix or data frame (samples as rows, lipids as columns).
`title`	Plot title string.
`metadata`	Optional metadata data frame (same row order) for group colouring.
`group_column`	Name of the group column in `metadata`.
`plot_type`	One of `"boxplot"`, `"violin"`, or `"density"`. Defaults to `"boxplot"`.
`view_mode`	Either `"sample"` (one box/violin per sample; the default, preserving previous behaviour) or `"lipid"` (one box/violin per lipid). Ignored for `"density"`, which always shows the intensity distribution.
`top_n`	Integer or `NULL`. In `"sample"` view, restrict the plot to the `top_n` most variable samples (`NULL`, the default, shows all). In `"lipid"` view, restrict to the `top_n` most variable lipids (`NULL` defaults to 30).

Value

A `ggplot2` object.

Examples

```
m <- matrix(rlnorm(60, 8, 1), nrow = 6, ncol = 10)
p <- create_pipeline_plot(m, title = "Test Pipeline")
print(p)
```

`create_plsda_plot_with_ellipses` *Create a PLS-DA Plot with Optional Ellipses*

Description

Create a PLS-DA Plot with Optional Ellipses

Usage

```
create_plsda_plot_with_ellipses(
  plsda_data,
  ellipse_type = "none",
  confidence_level = 0.95,
  title = "PLS-DA Analysis",
  show_sample_labels = FALSE
)
```

Arguments

`plsda_data`	Data frame with columns `Comp1`, `Comp2`, `Group`, and (optionally) `Sample`.
`ellipse_type`	One of `"none"`, `"confidence"`, or `"visual"`.
`confidence_level`	Numeric confidence level (default `0.95`).
`title`	Plot title.
`show_sample_labels`	Logical. Show sample name labels if `TRUE`.

Value

A `ggplot2` object.

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
res <- perform_plsda(norm, d$metadata, "Sample Group")
p <- create_plsda_plot_with_ellipses(res$scores_data)
print(p)
```

`create_volcano_plot_labeled` *Create a Volcano Plot with Optional Classification Colouring*

Description

Significant lipids (adj.P.Val < `pval_threshold` AND |logFC| > `logfc_threshold`) are coloured; non-significant points are grey. When `classification_data` is supplied, significant lipids are coloured by the selected classification column.

Usage

```
create_volcano_plot_labeled(
  results,
  title = "Volcano Plot",
  logfc_threshold = 1,
  pval_threshold = 0.05,
  top_labels = 15,
  classification_data = NULL,
  color_by = NULL
)
```

Arguments

`results`	Data frame of differential analysis results (must have columns `logFC` and `adj.P.Val`; row names = lipid names).
`title`	Plot title.
`logfc_threshold`	Absolute log-fold-change threshold.
`pval_threshold`	Adjusted p-value threshold.
`top_labels`	Number of top significant lipids to label.
`classification_data`	Optional classification data frame with a `Lipid` column.
`color_by`	Column in `classification_data` to use for colouring.

Value

A `ggplot2` object.

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
res <- perform_differential_analysis(norm, d$metadata, "Sample Group",
  contrasts_list = NULL, method = "limma"
)
p <- create_volcano_plot_labeled(res$results[[1]])
print(p)
```

determine_saturation *Determine Fatty-Acid Saturation from a Lipid Name*

Description

Parses standard lipid name notation to count double bonds and classifies the species as SFA (0 double bonds), MUFA (1) or PUFA (>1).

Usage

```
determine_saturation(lipid_name)
```

Arguments

lipid_name	A single character string with the lipid name.

Value

One of "SFA", "MUFA", "PUFA", or "Unclassified".

Examples

```
determine_saturation("PC 16:0_18:1") # "MUFA"
determine_saturation("PE 18:0_18:0") # "SFA"
determine_saturation("TG 16:0_18:1_20:4") # "PUFA"
```

export_classification *Export Lipid Classification to CSV*

Description

Export Lipid Classification to CSV

Usage

```
export_classification(classification, file_path)
```

Arguments

classification	Data frame with lipid classifications.
file_path	Output file path.

Value

Invisibly returns TRUE on success.

Examples

```
cls <- classify_lipids(c("PC 16:0_18:1", "PE 18:0_20:4"))
tmp <- tempfile(fileext = ".csv")
export_classification(cls, tmp)
unlink(tmp)
```

---

fix_sample_alignment *Align Samples Between Data Matrix and Metadata*

Description

Align Samples Between Data Matrix and Metadata

Usage

```
fix_sample_alignment(data_matrix, metadata)
```

Arguments

data_matrix	Numeric matrix (features as rows, samples as columns).
metadata	Metadata data frame.

Value

Named list with aligned `data_matrix` and `metadata`.

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
aligned <- fix_sample_alignment(t(d$numeric_data), d$metadata)
names(aligned)
```

---

generate_example_data *Generate Example Dataset*

Description

Creates a synthetic lipidomics dataset with realistic lipid names and group differences suitable for demonstrating differential analysis capabilities.

Usage

```
generate_example_data()
```

Value

Data frame with simulated lipidomics data including 4 groups with 5 replicates each

Examples

```
example_data <- generate_example_data()
head(example_data[, 1:10])
```

`get_imputation_descriptions` *Return Human-Readable Descriptions of Imputation Methods*

Description

Return Human-Readable Descriptions of Imputation Methods

Usage

```
get_imputation_descriptions()
```

Value

Named character vector (name = method key, value = description).

Examples

```
descs <- get_imputation_descriptions()
cat(descs["half_min"])
```

`get_imputation_methods` *Return Available Imputation Method Names*

Description

Return Available Imputation Method Names

Usage

```
get_imputation_methods()
```

Value

Character vector of imputation method names supported by `impute_missing_values`.

Examples

```
get_imputation_methods()
```

get_normalization_descriptions *Return Human-Readable Descriptions of Normalization Methods*

Description

Used by the Shiny app to populate help text.

Usage

```
get_normalization_descriptions()
```

Value

Named character vector (name = method key, value = description).

Examples

```
descs <- get_normalization_descriptions()
cat(descs["TIC"])
```

get_normalization_methods *Return Available Normalization Method Names*

Description

`"Log2Median"` and `"VSN"` are two distinct methods: `"Log2Median"` applies a fixed `log2(x + 1)` transform plus median centering, while `"VSN"` performs genuine variance-stabilising calibration via `vsn::justvsn()` (requires the Bioconductor **vsn** package).

Usage

```
get_normalization_methods()
```

Value

Character vector of normalization method names supported by `apply_normalizations`.

See Also

`normalize_log2median`, `normalize_vsn`.

Examples

```
get_normalization_methods()
```

impute_missing_values *Impute Missing Values in a Lipidomics Data Matrix*

Description

Replaces NA values using the chosen strategy. Imputation should be applied **before** normalisation so that missing values do not bias per-sample scaling factors.

Usage

```
impute_missing_values(data_matrix, method = "half_min", k = 5L, seed = NULL)
```

Arguments

data_matrix	Numeric matrix with samples as rows and lipids as columns.
method	Imputation method. One of "half_min" (default), "min", "zero", "mean", "median", "knn". See get_imputation_descriptions for details.
k	Integer. Number of nearest neighbours for "knn" (ignored for other methods). Default 5.
seed	Optional integer seed for reproducible results. Only consulted when method = "knn". When NULL (the default) no seed is set and the caller's random-number state is left untouched; supply an integer to run the imputation under withr::with_seed(). Either way the caller's RNG stream is restored on exit. Note that KNN imputation is reproducible regardless of this argument, because impute::impute.knn() seeds its own generator internally via its rng.seed argument; seed controls only the RNG context this call runs in, and does not alter the imputed values.

Value

Imputed numeric matrix of the same dimensions as data_matrix.

Examples

```
m <- matrix(c(1000, NA, 3000, NA, 500, 1500), nrow = 2)
impute_missing_values(m, method = "half_min")
```

launch_lipidomics_app *Launch the LIPIDIFy Shiny Application*

Description

Starts an interactive Shiny dashboard for end-to-end lipidomics analysis, including data upload, lipid classification, normalization, differential analysis, enrichment analysis, and report generation.

Usage

```
launch_lipidomics_app(port = NULL)
```

Arguments

port	Integer. Port number for the Shiny server. NULL lets Shiny pick a free port automatically.

Value

Launches the Shiny application (does not return a value).

Examples

```
if (interactive()) {
  launch_lipidomics_app()
}
```

---

load_custom_classification

Load Custom Lipid Classification from a CSV File

Description

Load Custom Lipid Classification from a CSV File

Usage

```
load_custom_classification(file_path)
```

Arguments

file_path	Path to a CSV file with a Lipid column followed by one or more classification columns.

Value

Data frame with Lipid as the first column.

Examples

```
tmp <- tempfile(fileext = ".csv")
write.csv(
  data.frame(
    Lipid = c("PC 16:0_18:1", "PE 18:0"),
    Class = c("Phospholipid", "Phospholipid")
  ),
  tmp,
  row.names = FALSE
```

```
)
cls <- load_custom_classification(tmp)
unlink(tmp)
```

`load_custom_enrichment_sets` *Load Custom Enrichment Sets from a CSV File*

Description

The CSV must have columns `Lipid` and `Set_Name`. A lipid may appear in multiple rows to belong to multiple sets.

Usage

```
load_custom_enrichment_sets(file_path)
```

Arguments

`file_path`	Path to the CSV file.

Value

Named list of character vectors (one vector per set).

Examples

```
tmp <- tempfile(fileext = ".csv")
write.csv(
  data.frame(
    Lipid = c("PC 16:0_18:1", "PE 18:0"),
    Set_Name = c("Phospholipids", "Phospholipids")
  ),
  tmp,
  row.names = FALSE
)
sets <- load_custom_enrichment_sets(tmp)
unlink(tmp)
```

load_lipidomics_data *Load Lipidomics Data*

Description

Reads a CSV file and separates metadata columns from numeric lipid abundance columns.

Usage

```
load_lipidomics_data(
  file_path,
  metadata_columns = c("Sample Name", "Sample Group", "Tumour ID", "Weight (mg)")
)
```

Arguments

file_path	Path to the lipidomics data file (CSV format).
metadata_columns	Character vector of expected metadata column names.

Value

A named list with components: data (original data frame), metadata (metadata data frame), numeric_data (numeric matrix).

Examples

```
# Write a minimal CSV then load it
tmp <- tempfile(fileext = ".csv")
write.csv(
  data.frame(
    "Sample Name" = c("S1", "S2"), "Sample Group" = c("A", "B"),
    "PC 16:0" = c(1000, 2000), check.names = FALSE
  ),
  tmp,
  row.names = FALSE
)
loaded <- load_lipidomics_data(tmp)
dim(loaded$numeric_data)
unlink(tmp)
```

load_lipidomics_data_from_df *Load Lipidomics Data from Data Frame*

Description

Processes a lipidomics data frame by separating metadata and numeric data columns.

Usage

```
load_lipidomics_data_from_df(
  data_df,
  metadata_columns = c("Sample Name", "Sample Group", "Tumour ID", "Weight (mg)")
)
```

Arguments

data_df	Data frame with lipidomics data
metadata_columns	Vector of metadata column names

Value

List containing data components (data, metadata, numeric_data)

Examples

```
data_df <- generate_example_data()
loaded_data <- load_lipidomics_data_from_df(data_df)
names(loaded_data)
```

normalize_log2median *Log2 Median Centering Normalization*

Description

Log2-transforms the data (log2(x + 1)), then median-centres each sample relative to the global median. This is a simplified variance-stabilising step suitable for mass-spectrometry lipidomics data.

Usage

```
normalize_log2median(data)
```

Arguments

data	Numeric matrix (samples as rows, lipids as columns).

Details

This method is **not** equivalent to the VSN procedure of Huber et al. (2002), which estimates its transformation by maximum likelihood. Log2 Median Centering applies a fixed `log2(x + 1)` transform and a per-sample median shift, and back-transforms to the original scale. For genuine variance-stabilising calibration use the separate `normalize_vsn` function (method name `"VSN"`), which calls `vsn::justvsn()`.

Value

Log2-median-centred numeric matrix.

See Also

`normalize_vsn` for true VSN.

Examples

```
m <- matrix(c(1000, 2000, 3000, 4000, 500, 1500), nrow = 2)
normalize_log2median(m)
```

`normalize_mean` *Mean Normalization*

Description

Scales each sample so its mean equals the global mean across all samples.

Usage

```
normalize_mean(data)
```

Arguments

`data`	Numeric matrix (samples as rows, lipids as columns).

Value

Mean-normalized matrix.

Examples

```
m <- matrix(c(1000, 2000, 3000, 4000, 500, 1500), nrow = 2)
normalize_mean(m)
```

normalize_median *Median Normalization*

Description

Scales each sample so its median equals the global median across all samples.

Usage

```
normalize_median(data)
```

Arguments

data	Numeric matrix (samples as rows, lipids as columns).

Value

Median-normalized matrix.

Examples

```
m <- matrix(c(1000, 2000, 3000, 4000, 500, 1500), nrow = 2)
normalize_median(m)
```

normalize_pqn *PQN Normalization*

Description

Probabilistic Quotient Normalization. Uses the per-feature median across samples as the reference spectrum.

Usage

```
normalize_pqn(data)
```

Arguments

data	Numeric matrix (samples as rows, lipids as columns).

Value

PQN-normalized matrix.

Examples

```
m <- matrix(c(1000, 2000, 3000, 4000, 500, 1500), nrow = 2)
normalize_pqn(m)
```

normalize_quantile *Quantile Normalization*

Description

Forces all samples to share an identical intensity distribution. After this step per-sample boxplots will look nearly identical – that is the intended and correct behaviour of quantile normalization.

Usage

```
normalize_quantile(data)
```

Arguments

data	Numeric matrix (samples as rows, lipids as columns).

Value

Quantile-normalized matrix of the same dimensions.

Examples

```
m <- matrix(c(1000, 2000, 3000, 4000, 500, 1500), nrow = 2)
normalize_quantile(m)
```

normalize_tic *TIC Normalization*

Description

Divides each sample by its total ion current (row sum) and rescales to the global mean TIC.

Usage

```
normalize_tic(data)
```

Arguments

data	Numeric matrix (samples as rows, lipids as columns).

Value

TIC-normalized matrix of the same dimensions.

Examples

```
m <- matrix(c(1000, 2000, 3000, 4000, 500, 1500), nrow = 2)
normalize_tic(m)
```

`normalize_vsn` *VSN (Variance Stabilizing Normalization)*

Description

Applies genuine variance-stabilising normalization using `vsn::justvsn()` from the Bioconductor **vsn** package (Huber et al. 2002). The method fits an arcsinh-based (generalised log) transformation whose parameters are estimated by robust maximum likelihood, simultaneously calibrating between-sample differences and stabilising the variance-versus-mean dependence typical of raw intensity data.

Usage

```
normalize_vsn(data, ...)
```

Arguments

`data`	Numeric matrix or data frame (samples as rows, lipids as columns).
`...`	Further arguments passed to `vsn::justvsn()`, e.g. `minDataPointsPerStratum`.

Details

This is a **different** method from `normalize_log2median`: VSN estimates its transformation from the data, whereas Log2 Median Centering applies a fixed `log2(x + 1)` transform followed by median centering. VSN is not simply another name for a log transformation, and it is not universally the best choice for every lipidomics dataset – compare pipelines before committing to one.

Orientation. The LIPIDIFy normalization contract is samples as rows and lipids as columns, while `vsn::justvsn()` expects features as rows and samples as columns. This function transposes explicitly before and after the call, and restores the original dimension names; the returned matrix has the same orientation, dimensions and dimnames as `data`.

Restrictions. Input must be numeric, finite (no `Inf`/ `-Inf`), and contain at least 2 samples and at least `minDataPointsPerStratum` (42 by default) lipid features. Samples or features consisting entirely of missing values are rejected, as are samples with fewer than two observed values. `NA` values are permitted: vsn fits on the observed values and returns `NA` in the same positions – they are not imputed. Negative values produce a warning because VSN is intended for positive, intensity-like data with a mean-dependent variance; nothing is silently replaced or coerced.

Output is on a generalised-log (roughly log2) scale, so VSN should not be chained with an additional `"Log2"` step.

Value

VSN-calibrated numeric matrix with the same dimensions, orientation and dimnames as `data`.

References

Huber W, von Heydebreck A, Sueltmann H, Poustka A, Vingron M (2002). Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics* **18**, S96–S104.

See Also

`normalize_log2median` for the (separate) log2 median centering method, and `apply_normalizations`.

Examples

```
if (requireNamespace("vsn", quietly = TRUE)) {
  # vsn needs at least 42 features, so use 8 samples x 60 lipids
  m <- outer(1:8, 1:60, function(i, j) exp(6 + 0.05 * j + 0.3 * sin(i + j)))
  rownames(m) <- paste0("Sample_", seq_len(8))
  colnames(m) <- paste0("Lipid_", seq_len(60))
  dim(normalize_vsn(m))
}
```

---

perform_differential_analysis

Perform Differential Analysis

Description

Perform Differential Analysis

Usage

```
perform_differential_analysis(
  data_matrix,
  metadata,
  group_column = "Sample Group",
  contrasts_list = NULL,
  method = "limma",
  design = NULL
)
```

Arguments

`data_matrix`	Normalized data matrix (features as rows, samples as columns)
`metadata`	Metadata data frame. Must contain `group_column` and have a number of rows matching the samples in `data_matrix`.
`group_column`	Column name in metadata containing group information
`contrasts_list`	List of contrasts to perform
`method`	Method to use: "limma" or "edger"

`design` Optional user-supplied design matrix (samples as rows). If `NULL` (the default), a design matrix of the form `~0 + group` is built automatically from `group_column`. When supplying a custom design, `contrasts_list` must also be supplied (limma/edgeR-style contrast strings referencing the design's column names).

Value

List containing results for each contrast

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
res <- perform_differential_analysis(norm, d$metadata, "Sample Group",
  contrasts_list = NULL, method = "limma"
)
names(res)
```

---

perform_enrichment_analysis

Perform Enrichment Analysis

Description

Perform Enrichment Analysis

Usage

```
perform_enrichment_analysis(
  results_list,
  classification_data,
  min_set_size = 5,
  max_set_size = 500,
  custom_sets = NULL
)
```

Arguments

`results_list`	List of differential analysis results
`classification_data`	Lipid classification data frame
`min_set_size`	Minimum pathway set size
`max_set_size`	Maximum pathway set size
`custom_sets`	Optional named list of custom lipid sets

Value

List containing GSEA results

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
cls <- classify_lipids(colnames(norm))
res <- perform_differential_analysis(norm, d$metadata, "Sample Group",
  contrasts_list = NULL, method = "limma"
)
enrich <- perform_enrichment_analysis(res$results, cls, min_set_size = 3)
names(enrich)
```

---

perform_pca *Perform PCA Analysis*

Description

Perform PCA Analysis

Usage

```
perform_pca(data_matrix, metadata, group_column = "Sample Group")
```

Arguments

data_matrix	Data matrix (samples as rows)
metadata	Metadata data frame
group_column	Group column name

Value

List containing PCA results and plot

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
pca_res <- perform_pca(norm, d$metadata, "Sample Group")
names(pca_res)
```

perform_plsda *Perform PLS-DA Analysis*

Description

Perform PLS-DA Analysis

Usage

```
perform_plsda(data_matrix, metadata, group_column = "Sample Group", n_comp = 2)
```

Arguments

data_matrix	Data matrix (samples as rows)
metadata	Metadata data frame
group_column	Group column name
n_comp	Number of components

Value

List containing PLS-DA results and plot

Examples

```
d <- load_lipidomics_data_from_df(generate_example_data())
norm <- apply_normalizations(d$numeric_data, c("TIC", "Log2"))
res <- perform_plsda(norm, d$metadata, "Sample Group")
names(res)
```

test_saturation_classification *Test Saturation Classification*

Description

Convenience function to verify saturation detection on a set of lipid names.

Usage

```
test_saturation_classification(test_lipids = NULL)
```

Arguments

test_lipids	Optional character vector of lipid names to test. If NULL, a built-in test set is used.

Value

A data frame with columns `Lipid` and `Saturation`, printed to the console and returned invisibly.

Examples

```
test_saturation_classification()
```

`visualize_raw_data` *Visualize Raw Data*

Description

Produces a simple per-sample boxplot, density, or histogram of raw lipidomics intensities.

Usage

```
visualize_raw_data(data_list, plot_type = "boxplot")
```

Arguments

`data_list`	List as returned by `load_lipidomics_data` or `load_lipidomics_data_from_df`, must contain `$numeric_data`.
`plot_type`	One of `"boxplot"`, `"density"`, or `"histogram"`.

Value

A `ggplot2` object.

Examples

```
dl <- load_lipidomics_data_from_df(generate_example_data())
p <- visualize_raw_data(dl, "boxplot")
print(p)
```

`visualize_raw_data_improved`
Visualize Raw or Normalized Data with Sample/Lipid Toggle

Description

Extended visualization that can show data from the sample perspective (one boxplot/violin/density per sample) or the lipid perspective (top variable lipids).

Usage

```
visualize_raw_data_improved(
  data_list,
  plot_type = "boxplot",
  view_mode = "sample",
  top_n = NULL,
  metadata = NULL,
  group_column = "Sample Group"
)
```

Arguments

data_list	List with $numeric_data (samples × lipids matrix).
plot_type	One of "boxplot", "violin", "density", or "histogram".
view_mode	Either "sample" or "lipid".
top_n	Integer or NULL. In lipid mode, the number of most variable lipids to display (defaults to 30 when NULL). In sample mode, the number of most variable samples to display; NULL (the default) shows all samples.
metadata	Optional metadata data frame (same row order as numeric_data) used to colour samples by group when group_column is provided.
group_column	Name of the group column in metadata.

Value

A ggplot2 object.

Examples

```
dl <- load_lipidomics_data_from_df(generate_example_data())
p <- visualize_raw_data_improved(dl, "boxplot", "sample")
print(p)
```

Index

apply_normalizations, 3, *5*, *27*

classify_lipids, 4
correct_batch_effects, 4
create_default_contrasts, 5
create_enrichment_barplot, 6
create_enrichment_dotplot, 7
create_heatmap_robust, 8
create_lipid_expression_barplot, 9
create_pca_plot_with_ellipses, 10
create_pipeline_plot, 11
create_plsda_plot_with_ellipses, 12
create_volcano_plot_labeled, 13

determine_saturation, 14

export_classification, 14

fix_sample_alignment, 15

generate_example_data, 15
get_imputation_descriptions, 16, *18*
get_imputation_methods, 16
get_normalization_descriptions, 17
get_normalization_methods, 17

impute_missing_values, 18

launch_lipidomics_app, 18
load_custom_classification, 19
load_custom_enrichment_sets, 20
load_lipidomics_data, 21
load_lipidomics_data_from_df, 22

normalize_log2median, *3*, *17*, 22, *26, 27*
normalize_mean, 23
normalize_median, 24
normalize_pqn, 24
normalize_quantile, 25
normalize_tic, 25
normalize_vsn, *3*, *17*, *23*, 26

perform_differential_analysis, 27
perform_enrichment_analysis, 28
perform_pca, 29
perform_plsda, 30

removeBatchEffect, *5*

test_saturation_classification, 30

visualize_raw_data, 31
visualize_raw_data_improved, 31