Package 'MetaProViz'

May 15, 2026

Type Package

Title METabolomics pre-PRocessing, functiOnal analysis and VIZualisation

Version 4.0.0

Description MetaProViz can analyse standard metabolomics and exometabolomics data (CoRe). It performs pre-processing including feature filtering, missing value imputation, normalisation and outlier detection. It performs functional analysis including differential metabolite analysis (DMA), clustering based on regulatory rules (MCA) and contains different visualisation methods to extract biological interpretable graphs and saves them in a publication ready format.

URL https://saezlab.github.io/MetaProViz

BugReports https://github.com/saezlab/MetaProViz/issues

biocViews Clustering, Metabolomics, Pathways, QualityControl, Software, SystemsBiology, Visualization

License BSD_3_clause + file LICENSE

Encoding UTF-8

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

LazyData false

Depends R (>= 4.4)

Imports broom, ComplexUpset (>= 1.3.3), cosmosR, DBI, dplyr, EnhancedVolcano, factoextra, ggbeeswarm, ggfortify, ggplot2 (>= 3.3.5), ggpubr, ggrepel, grid, gridExtra, gtools, hash, igraph, ggraph, inflection, limma, logger, magrittr, methods, OmnipathR (>= 3.19.12), patchwork, pheatmap, Polychrome, purrr, qcc, qvalue, rappdirs, readr, rlang, rstatix, S4Vectors, stringr, SummarizedExperiment, tibble, tidyr, tidyselect, writexl

Suggests BiocStyle, ggupset, ggVennDiagram, kableExtra, knitr, pkgdown, svglite, testthat (>= 3.1.4)

RoxygenNote 7.3.3

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/MetaProViz

git_branch RELEASE_3_23

git_last_commit 31f4cb3

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-05-14

Author Christina Schmidt [aut, cre, fnd] (ORCID:
<https://orcid.org/0000-0002-3867-0881>),
Denes Turei [aut] (ORCID: <https://orcid.org/0000-0002-7249-9379>),
Dimitrios Prymidis [aut] (ORCID:
<https://orcid.org/0009-0000-0168-3841>),
Macabe Daley [aut] (ORCID: <https://orcid.org/0000-0002-8026-7068>),
Jannik Franken [aut],
Julio Saez-Rodriguez [aut] (ORCID:
<https://orcid.org/0000-0002-8552-8976>),
Christian Frezza [aut] (ORCID: <https://orcid.org/0000-0002-3293-7397>)

Maintainer Christina Schmidt <christina.schmidt.science@gmail.com>

Contents

`alanine_pathways` *alanine_pathways*

Description

Manually curated table for the amino acid alanine toshowcase pathways (wiki, reactome, etc.) and alanine IDs (chebi, hmdb, etc.) included in those pathways

Usage

```
alanine_pathways
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 204 rows and 5 columns.

Examples

```
data(alanine_pathways)
head(alanine_pathways)
```

`biocrates_features` *biocrates_features*

Description

Biocrates kit feature information of the "MxP Quant 500 XL kit" that covers more than 1,000 metabolites with biochemical class information and the exported different metabolite IDs (HMDB, KEGG, etc.). information (INCHI, Key, etc.).

Usage

```
biocrates_features
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1019 rows and 14 columns.

Examples

```
data(biocrates_features)
head(biocrates_features)
```

cellular_meta *cellular_meta*

Description

Metabolomics workbench project PR001418, study ST002226 and ST002224 measured metabolites were assigned HMDB and KEGG IDs as well as one main metabolic pathway. metabolite trivial names. nitrogen supports renal cancer progression , Nature Communications 2022, doi:10.1038/s41467022350364

Usage

```
cellular_meta
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 199 rows and 5 columns.

Examples

```
data(cellular_meta)
head(cellular_meta)
```

checkmatch_pk_to_data *Check and summarize relationship between prixor knowledge to measured*

Description

features

Usage

```
checkmatch_pk_to_data(
  data,
  input_pk,
 metadata_info = c(InputID = "HMDB", PriorID = "HMDB", grouping_variable = "term"),
  save_table = "csv",
  path = NULL
)
```

Arguments

data	dataframe with at least one column with the detected metabolite IDs (e.g. HMDB). If there are multiple IDs per detected peak, please separate them by comma ("," or ", " or chr list). If there is a main ID and additional IDs, please provide them in separate columns.
input_pk	dataframe with at least one column with the metabolite ID (e.g. HMDB) that need to match data metabolite IDs "source" (e.g. term). If there are multiple IDs, as the original pathway IDs (e.g. KEGG) where translated (e.g. to HMDB), please separate them by comma ("," or ", " or chr list).
metadata_info	Colum name of Metabolite IDs in data and input_pk as well as column name of grouping_variable in input_pk. **Default = c(InputID="HMDB", PriorID="HMDB", grouping_variable="term")**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
path	Optional: Path to the folder the results should be saved at. **Default = NULL**

Value

A `list` with three elements:

- `data_summary` — a data frame summarising matching results per input ID, including counts, conflicts, and recommended actions.
- `GroupingVariable_summary` — a detailed data frame showing matches grouped by the specified variable, with conflict annotations.
- `data_long` — a merged data frame of prior knowledge IDs and detected IDs in long format.

Examples

```
## Not run:
data(cellular_meta)
DetectedIDs <- cellular_meta %>%
    dplyr::select("Metabolite", "HMDB") %>%
    tidyr::drop_na()
input_pathway <- translate_id(
    data = metsigdb_kegg(),
    metadata_info = c(
        InputID = "MetaboliteID",
        grouping_variable = "term"
    ),
    from = c("kegg"),
    to = c("hmdb")
)[["TranslatedDF"]] %>% tidyr::drop_na()
Res <- checkmatch_pk_to_data(
    data = DetectedIDs,
    input_pk = input_pathway,
    metadata_info = c(
        InputID = "HMDB",
        PriorID = "hmdb",
        grouping_variable = "term"
```

```
    )
)

## End(Not run)
```

`cluster_ora` *Overrepresentation analysis by cluster*

Description

Uses enricher to run ORA on each of the metabolite cluster from any of the MCA functions using a pathway list

Usage

```
cluster_ora(
  data,
  metadata_info = c(ClusterColumn = "RG2_Significant", BackgroundColumn = "BG_method",
    PathwayTerm = "term", PathwayFeature = "Metabolite"),
  remove_background = TRUE,
  input_pathway,
  pathway_name = "",
  min_gssize = 10,
  max_gssize = 1000,
  save_table = "csv",
  path = NULL
)
```

Arguments

`data`	DF with metabolite names/metabolite IDs as row names. Metabolite names/IDs need to match the identifier type (e.g. HMDB IDs) in the input_pathway.
`metadata_info`	*Optional:* Pass ColumnName of the column including the cluster names that ORA should be performed on (=ClusterColumn). BackgroundColumn passes the column name needed if remove_background=TRUE. Also pass ColumnName for input_pathway including term and feature names. (ClusterColumn= ColumnName data, BackgroundColumn = ColumnName data, PathwayTerm= ColumnName input_pathway, PathwayFeature= ColumnName input_pathway) **c(FeatureName="Metabolite", ClusterColumn="RG2_Significant", BackgroundColumn="BG_method", PathwayTerm= "term", PathwayFeature= "Metabolite")**
`remove_background`	*Optional:* If TRUE, column BackgroundColumn name needs to be in metadata_info, which includes TRUE/FALSE for each metabolite to fall into background based on the chosen Background method for e.g. mca_2cond are removed from the universe. **default: TRUE**

input_pathway	DF that must include column "term" with the pathway name, column "Feature" with the Metabolite name or ID and column "Description" with pathway description.
pathway_name	*Optional:* Name of the pathway list used **default: ""**
min_gssize	*Optional:* minimum group size in ORA **default: 10**
max_gssize	*Optional:* maximum group size in ORA **default: 1000**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt" **default: "csv"**
path	*Optional:* Path to the folder the results should be saved at. **default: NULL**

Value

Saves results as individual .csv files.

Examples

```
KEGG_Pathways <- metsigdb_kegg()
data(intracell_dma) # loads the object into your environment
DMAres <- intracell_dma %>%
    dplyr::filter(!is.na(KEGGCompound)) %>%
    tibble::column_to_rownames("KEGGCompound") %>%
    dplyr::select(-"Metabolite")
RES <- cluster_ora(
    data = DMAres,
    metadata_info = c(
        ClusterColumn = "Pathway",
        PathwayTerm = "term",
        PathwayFeature = "Metabolite"
    ),
    input_pathway = KEGG_Pathways,
    remove_background = FALSE
)
```

---

cluster_pk *Cluster terms in prior knowledge by set overlap*

Description

Cluster terms in prior knowledge by set overlap

Usage

```
cluster_pk(
  data,
  metadata_info = c(metabolite_column = "MetaboliteID", pathway_column = "term"),
  similarity = c("jaccard", "overlap_coefficient", "correlation"),
```

```
  correlation_method = "pearson",
  input_format = c("long", "enrichment"),
  delimiter = "/",
  threshold = 0.5,
  plot_threshold = 0,
  clust = c("components", "community", "hierarchical"),
  hclust_method = "average",
  min = 2,
  plot_name = "ClusterGraph",
  max_nodes = 10000,
  min_degree = 1,
  node_size_column = NULL,
  show_density = FALSE,
  save_plot = "png",
  print_plot = FALSE,
  path = NULL
)
```

Arguments

`data`	Long data frame with one ID per row, or enrichment-style table with a delimited metabolite list per term (see input_format).
`metadata_info`	List with entries `metabolite_column` (metabolite ID column or delimited list column) and `pathway_column` (term column). Defaults to `c(metabolite_column = "MetaboliteID", pathway_column = "term")`.
`similarity`	Similarity measure between term ID sets. Options: "jaccard" (default), "overlap_coefficient", or "correlation". Jaccard similarity is |A intersect B| / |A union B|. Overlap coefficient is |A intersect B| / min(|A|, |B|). Jaccard is stricter for large sets, while overlap_coefficient is more permissive for nested sets.
`correlation_method`	Correlation method when `similarity = "correlation"`. One of "pearson", "spearman", "kendall". Ignored otherwise.
`input_format`	Input format of `data`. Use "long" for one ID per row (default) or "enrichment" for one term per row with a delimited ID list. The `metabolite_column` entry in metadata_info is interpreted accordingly.
`delimiter`	Delimiter for metabolite ID lists when input_format = "enrichment". Ignored for input_format = "long". Default = "/".
`threshold`	Similarity cutoff for keeping edges (applies to all clustering modes). Default = 0.5.
`plot_threshold`	Similarity cutoff for plotting edges in viz_graph. Default = 0 (plot all edges with similarity > 0).
`clust`	Clustering strategy: "components" (connected components on thresholded unweighted graph), "community" (Louvain on thresholded weighted graph), or "hierarchical" (hclust on distance = 1 - similarity).
`hclust_method`	Linkage method for hierarchical clustering. One of "average" (default), "single", "complete", "ward.D", "ward.D2", "mcquitty", "median", "centroid". Used only when clust = "hierarchical".

`min` Minimum cluster size; smaller clusters are relabeled to "None". Default = 2.

`plot_name` *Optional:* String added to output files of the plot. Default = "ClusterGraph".

`max_nodes` *Optional:* Maximum nodes for plotting. If set, keeps nodes from the largest component up to this limit (by degree). Used only for the graph plot. Default = 10000.

`min_degree` *Optional:* Minimum degree filter for graph plotting. Used only for the graph plot. Default = 1.

`node_size_column`
Optional: Numeric column name from `data` used to scale node sizes in the graph. Aggregated per term (mean) when multiple rows map to the same term. Default = NULL.

`show_density` *Optional:* If TRUE, add a hull background per cluster to the graph. Default = FALSE.

`save_plot` *Optional:* Select the file type of output plots. Options are svg, pdf, png or NULL. **Default = "svg"**

`print_plot` *Optional:* If TRUE prints an overview of resulting plots. **Default = FALSE**

`path` Optional: String which is added to the resulting folder name. **default: NULL**

Value

A list with:

`data` Input data with a `cluster` column added.

`cluster_summary`
Summary of cluster sizes and percentages.

`clusters` Named vector of term -> cluster assignment.

`similarity_matrix`
Term-by-term similarity matrix.

`distance_matrix`
Term-by-term distance matrix (1 - similarity).

`node_sizes` Named numeric vector of node sizes used in plotting (or NULL).

`graph_plot` Graph plot returned by viz_graph.

Examples

```
# Create toy pathway data in long format
toy_pw <- data.frame(
    MetaboliteID = c("C1", "C2", "C3", "C1", "C2", "C4", "C3", "C4", "C5"),
    term = c("pA", "pA", "pA", "pB", "pB", "pB", "pC", "pC", "pC")
)

r <- cluster_pk(
    toy_pw,
    metadata_info = c(
        metabolite_column = "MetaboliteID",
        pathway_column = "term"
```

```
    ),
    input_format = "long",
    similarity = "jaccard",
    threshold = 0.1,
    clust = "community",
    min = 1,
    save_plot = NULL,
    print_plot = FALSE
)
```

---

compare_pk *Compare Prior Knowledge Resources and/or Columns within a Single Resource*

Description

and Generate an UpSet Plot This function compares gene and/or metabolite features across multiple prior knowledge (PK) resources or, if a single resource is provided with a vector of column names in `metadata_info`, compares columns within that resource. In the multi-resource mode, each element in `data` represents a PK resource (either as a data frame or a recognized resource name) from which a set of features is extracted. A binary summary table is then constructed and used to create an UpSet plot. In the within-resource mode, a single data frame is provided (with `data` containing one element) and its `metadata_info` entry is a vector of column names to compare (e.g., binary indicators for different annotations). In this case, the function expects the data frame to have a grouping column named `"Class"` (or, alternatively, a column specified via the `class_col` attribute in `metadata_info`) that is used for grouping in the UpSet plot.

Usage

```
compare_pk(
  data,
  metadata_info = NULL,
  filter_by = c("both", "gene", "metabolite"),
  plot_name = "Overlap of Prior Knowledge Resources",
  name_col = "TrivialName",
  palette_type = "polychrome",
  save_plot = "svg",
  save_table = "csv",
  print_plot = TRUE,
  path = NULL
)
```

Arguments

`data`	A named list where each element corresponds to a prior knowledge (PK) resource. Each element can be:

- A data frame containing gene/metabolite identifiers (and additional columns for within-resource comparison),
- A character string indicating the resource name. Recognized names include (but are not limited to): "Hallmarks", "Gaude", "MetalinksDB", and "RAMP" (or "metsigdb_chemicalclass"). In the latter case, the function will attempt to load the corresponding data automatically.

metadata_info A named list (with names matching those in data) where each element is either a character string or a character vector indicating the column name(s) to extract features. For multiple-resource comparisons, these refer to the columns containing feature identifiers. For within-resource comparisons, the vector should list the columns to compare (e.g., c("CHEBI", "HMDB", "LIMID")). In within-resource mode, the input data frame is expected to contain a column named "Class" (or a grouping column specified via the class_col attribute). *If no grouping column is found, a default grouping column named "Group" (with all rows assigned the same value) is created.*

filter_by Character. Optional filter for the resulting features when comparing multiple resources. Options are: "both" (default), "gene", or "metabolite". This parameter is ignored in within-resource mode.

plot_name *Optional:* String which is added to the output files of the Upsetplot **Default = ""**

name_col *Optional:* column name including the feature names. Default is "TrivialName".

palette_type Character. Color palette to be used in the plot. Default is "polychrome".

save_plot *Optional:* Select the file type of output plots. Options are svg, png, pdf. **Default = svg**

save_table *Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**

print_plot *Optional:* TRUE or FALSE, if TRUE Volcano plot is saved as an overview of the results. **Default = TRUE**

path *Optional:* Path to the folder the results should be saved at. **Default = NULL**

Value

A list containing two elements:

- summary_table: A data frame representing either:
  - the binary summary matrix of feature presence/absence across multiple resources, or
  - the original data frame (augmented with binary columns and a None column) in within-resource mode.
- upset_plot: The UpSet plot object generated by the function.

Examples

```
## Not run:
## Example 1: Within-Resource Comparison
## (Comparing Columns Within a Single data Frame)
```

```
# biocrates_features is a data frame with columns:
# "TrivialName", "CHEBI", "HMDB", "LIMID", and "Class".
# Here the "Class" column is used as the grouping variable
# in the UpSet plot.
data(biocrates_features)
data_single <- list(Biocft = biocrates_features)
metadata_info_single <- list(Biocft = c("CHEBI", "HMDB", "LIMID"))

res_single <-
compare_pk(
data = data_single, metadata_info = metadata_info_single,
plot_name = "Overlap of BioCrates Columns"
)

## Example 2: Custom data Frames with Custom Column Names

# Example with preloaded data frames and custom column names:
hallmarks_df <- data.frame(
feature = c("HMDB0001", "GENE1", "GENE2"),
stringsAsFactors = FALSE
)
gaude_df <- data.frame(
feature = c("GENE2", "GENE3"),
stringsAsFactors = FALSE
)
metalinks_df <- data.frame(
hmdb = c("HMDB0001", "HMDB0002"),
gene_symbol = c("GENE1", "GENE4"),
stringsAsFactors = FALSE
)
ramp_df <- data.frame(
class_source_id = c("HMDB0001", "HMDB0003"),
stringsAsFactors = FALSE
)
data <- list(
Hallmarks = hallmarks_df, Gaude = gaude_df,
MetalinksDB = metalinks_df, RAMP = ramp_df
)
metadata_info <- list(
Hallmarks = "feature", Gaude = "feature",
MetalinksDB = c("hmdb", "gene_symbol"),
RAMP = "class_source_id"
)
res <- compare_pk(
data = data, metadata_info = metadata_info,
filter_by = "metabolite"
)

## End(Not run)
```

count_id *Count Entries and Generate a Histogram Plot for a Specified Column*

Description

This function processes a data frame column by counting the number of entries within each cell. It considers both NA values and empty strings as zero entries, and categorizes each cell as "No ID", "Single ID", or "Multiple IDs" based on the count. A histogram is then generated to visualize the distribution of entry counts. scale_x_continuous

Usage

```
count_id(
  data,
  column,
  delimiter = ",\n    ",
  fill_colors = c(`No ID` = "#FB8072", `Single ID` = "#B3DE69", `Multiple IDs` =
    "#80B1D3"),
  binwidth = 1,
  title_prefix = NULL,
  save_plot = "svg",
  save_table = "csv",
  print_plot = TRUE,
  path = NULL
)
```

Arguments

data	A data frame containing the data to be analyzed.
column	A string specifying the name of the column in data to analyze.
delimiter	A string specifying the delimiter used to split cell values. Defaults to ",".
fill_colors	A named character vector providing colors for each category. Defaults to c("No ID" = "#FB8072", "Single ID" = "#B3DE69", "Multiple IDs" = "#80B1D3").
binwidth	Numeric value specifying the bin width for the histogram. Defaults to 1.
title_prefix	A string to use as the title of the plot. If NULL (default), the title will be generated as "Number of IDs per Biocrates Cell".
save_plot	*Optional:* Select the file type of output plots. Options are svg, png, pdf. **Default = svg**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
print_plot	*Optional:* TRUE or FALSE, if TRUE Volcano plot is saved as an overview of the results. **Default = TRUE**
path	*Optional:* Path to the folder the results should be saved at. **Default = NULL**

Value

A list with two elements:

`result`	A data frame that includes three additional columns: `was_na` (logical indicator of missing or empty cells), `entry_count` (number of entries in each cell), and `id_label` (a categorical label based on the entry count).
`plot`	A `ggplot` object representing the histogram of entry counts.

Examples

```
data(biocrates_features)
count_id(biocrates_features, "HMDB")
```

`dma` *Differential metabolite analysis*

Description

Performs differential metabolite analysis to obtain Log2FC, p-value, adjusted p-value, and t-value when comparing two or multiple conditions.

Usage

```
dma(
  data,
  metadata_sample = NULL,
 metadata_info = c(Conditions = "Conditions", Numerator = NULL, Denominator = NULL),
  pval = "lmFit",
  padj = "fdr",
  metadata_feature = NULL,
  core = FALSE,
  vst = FALSE,
  shapiro = TRUE,
  bartlett = TRUE,
  transform = TRUE,
  save_plot = "svg",
  save_table = "csv",
  print_plot = TRUE,
  path = NULL
)
```

Arguments

data	SummarizedExperiment or data frame. If SummarizedExperiment, metadata_sample is extracted from colData and metadata_feature from rowData. If data frame, provide unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for undetected metabolites.
metadata_sample	Data frame (optional). Only required if data is not a SummarizedExperiment. Contains metadata information about samples, combined with input data based on unique sample identifiers used as row names. Default: NULL.
metadata_info	Named character vector (optional). Includes conditions column information on numerator or denominator: c(Conditions="ColumnName", Numerator="ColumnName", Denominator="ColumnName"). Denominator and Numerator specify which comparisons are performed (one-vs-one, all-vs-one, all-vs-all). Denominator=NULL and Numerator=NULL selects all conditions and performs multiple all-vs-all comparisons. Log2FC values are obtained by dividing numerator by denominator, thus positive Log2FC values indicate higher expression in numerator. Default: c(conditions="Conditions", numerator=NULL, denumerator=NULL).
pval	Character (optional). Abbreviation of the selected test to calculate p-value. For one-vs-one comparisons choose t.test, wilcox.test, chisq.test, cor.test, or lmFit (limma). For one-vs-all or all-vs-all comparisons choose aov (anova), welch (welch anova), kruskal.test, or lmFit (limma). Default: "lmFit".
padj	Character (optional). Abbreviation of the selected p-value adjustment method for multiple hypothesis testing correction. See ?p.adjust for methods: "BH", "fdr", "bonferroni", "holm", etc. Default: "fdr".
metadata_feature	Data frame (optional). Provides metadata information (e.g., pathway, retention time) for each metabolite. Only used if data is not a SummarizedExperiment. Row names must match metabolite names in data columns. Default: NULL.
core	Logical (optional). Whether consumption/release input is used. Default: FALSE.
vst	Logical. Whether to use variance stabilizing transformation on data when linear modeling is used for hypothesis testing. Default: FALSE.
shapiro	Logical. Whether to perform Shapiro-Wilk test to assess data distribution (normal versus non-normal). Default: TRUE.
bartlett	Logical. Whether to perform Bartlett's test. Default: TRUE.
transform	Logical. If TRUE, data is expected to be non-log2-transformed and log2 transformation will be performed within limma and Log2FC calculation. If FALSE, data is expected to be log2-transformed as this impacts Log2FC calculation and limma. Default: TRUE.
save_plot	Character (optional). File type of output plots: "svg", "png", "pdf". Default: "svg".
save_table	Character (optional). File type for analysis results: "csv", "xlsx", "txt". Default: "csv".
print_plot	Logical (optional). Whether volcano plot is printed as overview of results. Default: TRUE.

path Character (optional). Path to folder where results should be saved. Default: NULL.

Value

List of lists. Depending on parameter settings, returns dma (data frame of each comparison), shapiro (includes data frame and plot), bartlett (includes data frame and histogram), vst (includes data frame and plot), and VolcanoPlot (plots of each comparison).

Examples

```
data(intracell_raw_se)
ResI <- dma(
    data = intracell_raw_se,
    metadata_info = c(
        Conditions = "Conditions", Numerator = NULL, Denominator = "HK2"
    )
)

data(intracell_raw)
Intra <- intracell_raw[-c(49:58), ] %>% tibble::column_to_rownames("Code")
ResI <- dma(
    data = Intra[, -c(1:3)],
    metadata_sample = Intra[, c(1:3)],
    metadata_info = c(
        Conditions = "Conditions", Numerator = NULL, Denominator = "HK2"
    )
)
```

---

equivalent_features *equivalent_features*

Description

Manually curated list of aminoacids and aminoacid-related metabolites with corresponding metabolite identifiers (HMDB, KEGG, etc.) irrespective of chirality.

Usage

```
equivalent_features
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 34 rows and 7 columns.

Examples

```
data(equivalent_features)
head(equivalent_features)
```

`equivalent_id` *Find additional potential IDs for "kegg", "pubchem", "chebi", "hmdb"*

Description

Find additional potential IDs for "kegg", "pubchem", "chebi", "hmdb"

Usage

```
equivalent_id(
  data,
  metadata_info = c(InputID = "MetaboliteID"),
  from = "hmdb",
  save_table = "csv",
  path = NULL
)
```

Arguments

`data`	dataframe with at least one column with the detected metabolite IDs (one ID per row).
`metadata_info`	*Optional:* Column name of metabolite IDs. **Default = list(InputID="MetaboliteID")**
`from`	ID type that is present in your data. Choose between "kegg", "pubchem", "chebi", "hmdb". **Default = "hmdb"**
`save_table`	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
`path`	Optional: Path to the folder the results should be saved at. **Default = NULL**

Value

Input DF with additional column including potential additional IDs.

Examples

```
data(cellular_meta)
DetectedIDs <- cellular_meta %>% tidyr::drop_na()
Res <- equivalent_id(
    data = DetectedIDs,
    metadata_info = c(InputID = "HMDB"),
    from = "hmdb"
)
```

gaude_pathways *gaude_pathways*

Description

gaude_pathways

Usage

```
gaude_pathways
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 1932 rows and 3 columns.

Examples

```
data(gaude_pathways)
head(gaude_pathways)
```

get_exclusion_metabolites *Metabolites excluded from prior knowledge resources*

Description

Builds an exclusion table from hard-coded seed identifiers and translates them across ID systems (HMDB, KEGG, CHEBI, PUBCHEM).

Usage

```
get_exclusion_metabolites()
```

Value

A data frame with columns `metabolite_id`, `class`, and `id_type`.

Examples

```
## Not run:
get_exclusion_metabolites()

## End(Not run)
```

hallmarks *hallmarks*

Description

hallmarks

Usage

```
hallmarks
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 7322 rows and 2 columns.

Examples

```
data(hallmarks)
head(hallmarks)
```

intracell_dma *intracell_dma*

Description

Metabolomics workbench project PR001418, study ST002224 where we performed differential metabolite analysis comparing intracellular metabolomics of 786-M1A versus HK2 cells. metabolite values used as input with row names being metabolitetrivial names. nitrogen supports renal cancer progression , Nature Communications 2022, doi:10.1038/s41467022350364

Usage

```
intracell_dma
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 179 rows and 14 columns.

Examples

```
data(intracell_dma)
head(intracell_dma)
```

intracell_raw *intracell_raw*

Description

Metabolomics workbench project PR001418, study ST002224 where we exported integrated raw peak values of intracellular metabolomics of HK2 and ccRCC cell lines 786-O, 786-M1A and 786-M2A. -`Conditions`: Character vector indicating cell line identity - `Analytical_Replicate`: Integer replicate number for analytical replicates -`Biological_Replicate`: Integer replicate number for biological replicates - Additional numeric columns (183 in total) containing raw metabolite intensities nitrogen supports renal cancer progression, Nature Communications 2022, DOI:10.1038/s41467-022-35036-4.

Usage

```
intracell_raw
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 58 rows and 186 columns.

Examples

```
data(intracell_raw)
head(intracell_raw)
```

intracell_raw_se *intracell_raw_se*

Description

Metabolomics workbench project PR001418, study ST002224 where we exported integrated raw peak values of intracellular metabolomics of HK2 and ccRCC cell lines 786-O, 786-M1A and 786-M2A converted into an se object. -`Conditions`: Character vector indicating cell line identity - `Analytical_Replicate`: Integer replicate number for analytical replicates -`Biological_Replicate`: Integer replicate number for biological replicates nitrogen supports renal cancer progression, Nature Communications 2022, DOI:10.1038/s41467-022-35036-4.

Usage

```
intracell_raw_se
```

Format

An object of class `SummarizedExperiment` with 182 rows and 58 columns.

Examples

```
data(intracell_raw_se)
head(intracell_raw_se)
```

`make_gene_metab_set` *Create metabolite sets from existing genesets*

Description

Gene to metabolite translation is based on mappings in Recon-3D (cosmosR).

Usage

```
make_gene_metab_set(
  input_pk,
  metadata_info = c(Target = "gene"),
  pk_name = NULL,
  save_table = "csv",
  path = NULL,
  exclude_metabolites = "all"
)
```

Arguments

`input_pk`	dataframe with two columns for source (=term) and Target (=gene), e.g. Hallmarks.
`metadata_info`	*Optional:* Column name of Target in input_pk. **Default = c(Target="gene")**
`pk_name`	*Optional:* Name of the prior knowledge resource. **default: NULL**
`save_table`	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
`path`	Optional: String which is added to the resulting folder name **default: NULL**
`exclude_metabolites`	Optional metabolite classes to exclude: NULL (exclude nothing), "all" (default), or any combination of c("ions", "small_molecules", "xenobiotics", "atoms").

Value

List of two data frames: "GeneMetabSet" and "MetabSet".

Examples

```
data(hallmarks)
make_gene_metab_set(hallmarks)
```

`mapping_ambiguity` *Create Mapping Ambiguities between two ID types*

Description

Create Mapping Ambiguities between two ID types

Usage

```
mapping_ambiguity(
  data,
  from,
  to,
  grouping_variable = NULL,
  summary = FALSE,
  save_table = "csv",
  path = NULL
)
```

Arguments

`data`	Translated DF from translate_id results or dataframe with at least one column with the target metabolite ID and another MetaboliteID type. One of the IDs can only have one ID per row, the other ID can be either separated by comma or a list. Optional: add other columns such as source (e.g. term).
`from`	Column name of the secondary or translated metabolite identifier in data. Here can be multiple IDs per row either separated by comma " ," or a list of IDs.
`to`	Column name of original metabolite identifier in data. Here should only have one ID per row.
`grouping_variable`	*Optional:* If NULL no groups are used. If TRUE provide column name in data containing the grouping_variable and features are grouped. **Default = NULL**
`summary`	*Optional:* If TRUE a long summary tables are created. **Default = FALSE**
`save_table`	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
`path`	Optional: Path to the folder the results should be saved at. **Default = NULL**

Value

List with at least 4 DFs: 1-3) from-to-to: 1. MappingIssues, 2. MappingIssues summary, 3. Long summary (If summary=TRUE) & 4-6) to-to-from: 4. MappingIssues, 5. MappingIssues summary, 6. Long summary (If summary=TRUE) & 7) Combined summary table (If summary=TRUE)

Examples

```
## Not run:
KEGG_Pathways <- metsigdb_kegg()
InputDF <- translate_id(
    data = KEGG_Pathways,
    metadata_info = c(
        InputID = "MetaboliteID",
        grouping_variable = "term"
    ),
    from = c("kegg"),
    to = c("pubchem")
)[["TranslatedDF"]]
Res <- mapping_ambiguity(
    data = InputDF,
    from = "MetaboliteID",
    to = "pubchem",
    grouping_variable = "term",
    summary = TRUE
)

## End(Not run)
```

---

mca_2cond *Metabolite clustering analysis for two conditions*

Description

Performs metabolite clustering analysis and computes clusters based on regulatory rules between conditions.

Usage

```
mca_2cond(
  data_c1,
  data_c2,
  metadata_info_c1 = c(ValueCol = "Log2FC", StatCol = "p.adj", cutoff_stat = 0.05,
    ValueCutoff = 1),
  metadata_info_c2 = c(ValueCol = "Log2FC", StatCol = "p.adj", cutoff_stat = 0.05,
    ValueCutoff = 1),
  feature = "Metabolite",
  save_table = "csv",
  method_background = "C1&C2",
  path = NULL
)
```

Arguments

data_c1	DF for your data (results from e.g. dma) containing metabolites in rows with corresponding Log2FC and stat (p-value, p.adjusted) value columns.
data_c2	DF for your data (results from e.g. dma) containing metabolites in rows with corresponding Log2FC and stat (p-value, p.adjusted) value columns.
metadata_info_c1	*Optional:* Pass ColumnNames and Cutoffs for condition 1 including the value column (e.g. Log2FC, Log2Diff, t.val, etc) and the stats column (e.g. p.adj, p.val). This must include: c(ValueCol=ColumnName_data_c1,StatCol=ColumnName_data_c1, cutoff_stat= NumericValue, ValueCutoff=NumericValue) **Default=c(ValueCol="Log2FC",StatCol="p. cutoff_stat= 0.05, ValueCutoff=1)**
metadata_info_c2	*Optional:* Pass ColumnNames and Cutoffs for condition 2 includingthe value column (e.g. Log2FC, Log2Diff, t.val, etc) and the stats column (e.g. p.adj, p.val). This must include: c(ValueCol=ColumnName_data_c2,StatCol=ColumnName_data_c2, cutoff_stat= NumericValue, ValueCutoff=NumericValue)**Default=c(ValueCol="Log2FC",StatCol="p. cutoff_stat= 0.05, ValueCutoff=1)**
feature	*Optional:* Column name of Column including the Metabolite identifiers. This MUST BE THE SAME in each of your Input files. **Default="Metabolite"**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt" **Default = "csv"**
method_background	*Optional:* Background method C1|C2, C1&C2, C2, C1 or * **Default="C1&C2"**
path	*Optional:* Path to the folder the results should be saved at. **Default = NULL**

Value

List of two DFs: 1. summary of the cluster count and 2. the detailed information of each metabolites in the clusters.

Examples

```
data(intracell_raw)
Intra <- intracell_raw %>% tibble::column_to_rownames("Code")
Input <- dma(
    data = Intra[-c(49:58), -c(1:3)],
    metadata_sample = Intra[-c(49:58), c(1:3)],
    metadata_info = c(
        Conditions = "Conditions",
        Numerator = NULL,
        Denominator = "HK2"
    )
)

Res <- mca_2cond(
    data_c1 = Input[["dma"]][["786-O_vs_HK2"]],
    data_c2 = Input[["dma"]][["786-M1A_vs_HK2"]]
)
```

mca_core *Metabolite clustering analysis for core experiments*

Description

Performs metabolite clustering analysis and computes clusters based on regulatory rules between intracellular and culture media metabolomics in core experiments.

Usage

```
mca_core(
  data_intra,
  data_core,
 metadata_info_intra = c(ValueCol = "Log2FC", StatCol = "p.adj", cutoff_stat = 0.05,
    ValueCutoff = 1),
 metadata_info_core = c(DirectionCol = "core", ValueCol = "Log2(Distance)", StatCol =
    "p.adj", cutoff_stat = 0.05, ValueCutoff = 1),
  feature = "Metabolite",
  save_table = "csv",
  method_background = "Intra&core",
  path = NULL
)
```

Arguments

data_intra	DF for your data (results from e.g. dma) containing metabolites in rows with corresponding Log2FC and stat (p-value, p.adjusted) value columns.
data_core	DF for your data (results from e.g. dma) containing metabolites in rows with corresponding Log2FC and stat (p-value, p.adjusted) value columns. Here we additionally require
metadata_info_intra	*Optional:* Pass ColumnNames and Cutoffs for the intracellular metabolomics including the value column (e.g. Log2FC, Log2Diff, t.val, etc) and the stats column (e.g. p.adj, p.val). This must include: c(ValueCol=ColumnName_data_intra,StatCol=ColumnName_ cutoff_stat= NumericValue, ValueCutoff=NumericValue) **Default=c(ValueCol="Log2FC",StatCol="p. cutoff_stat= 0.05, ValueCutoff=1)**
metadata_info_core	*Optional:* Pass ColumnNames and Cutoffs for the consumption-release metabolomics including the direction column, the value column (e.g. Log2Diff, t.val, etc) and the stats column (e.g. p.adj, p.val). This must include: c(DirectionCol= ColumnName_data_core,ValueCol=ColumnName_data_core,StatCol=ColumnName_data_core, cutoff_stat= NumericValue, ValueCutoff=NumericValue)**Default=c(DirectionCol="core", ValueCol="Log2(Distance)",StatCol="p.adj", cutoff_stat= 0.05, ValueCutoff=1)**
feature	*Optional:* Column name of Column including the Metabolite identifiers. This MUST BE THE SAME in each of your Input files. **Default="Metabolite"**

save_table *Optional:* File types for the analysis results are: "csv", "xlsx", "txt" **default: "csv"**

method_background *Optional:* Background method 'Intra|core, Intra&core, core, Intra or * **Default="Intra&core"**

path *Optional:* Path to the folder the results should be saved at. **default: NULL**

Value

List of two DFs: 1. summary of the cluster count and 2. the detailed information of each metabolites in the clusters.

Examples

```
data(intracell_dma)

# Create mock CoRe DMA results with required columns
core_dma <- data.frame(
    Metabolite = intracell_dma$Metabolite[1:50],
    `Log2(Distance)` = runif(50, -2, 2),
    p.adj = runif(50, 0, 0.1),
    core = sample(c("Consumption", "Release"), 50, replace = TRUE),
    check.names = FALSE
)

Res <- mca_core(
    data_intra = as.data.frame(intracell_dma),
    data_core = core_dma,
    save_table = NULL
)
```

---

mca_core_rules *mca_core_rules*

Description

Manually curated table defining the flow of information of the Conusuption-Release and Intracellular metabolomics biological regulatory clusters Regulatory labels from the different grouping methods. columns (RG1-RG3)

Usage

```
mca_core_rules
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 108 rows and 7 columns.

Examples

```
data(mca_core_rules)
head(mca_core_rules)
```

`mca_twocond_rules` *mca_twocond_rules*

Description

Manually curated table defining the flow of information of the two condition biological regulatory clusters Regulatory labels from the different grouping methods. columns (RG1-RG3)

Usage

```
mca_twocond_rules
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 36 rows and 5 columns.

Examples

```
data(mca_twocond_rules)
head(mca_twocond_rules)
```

`medium_raw` *medium_raw*

Description

Metabolomics workbench project PR001418, study ST002226 where we exported integrated raw peak values of intracellular metabolomics of HK2 and cccRCC cell lines 786-O, 786-M1A, 786-M2A, OS-RC-2, OS-LM1 and RFX-631. a numeric column for each measured metabolite (raw data) nitrogen supports renal cancer progression , Nature Communications 2022, doi:10.1038/s41467022350364

Usage

```
medium_raw
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 44 rows and 77 columns.

Examples

```
data(medium_raw)
head(medium_raw)
```

`metadata_analysis` *PCA-based metadata analysis*

Description

Performs PCA analysis on input data and combines it with sample metadata to run ANOVA tests for identifying significant differences between groups.

Usage

```
metadata_analysis(
  data,
  metadata_sample = NULL,
  scaling = TRUE,
  percentage = 0.1,
  cutoff_stat = 0.05,
  cutoff_variance = 1,
  save_table = "csv",
  save_plot = "svg",
  print_plot = TRUE,
  path = NULL
)
```

Arguments

`data`	SummarizedExperiment (se) file including assay and colData. If se file is provided, metadata_sample is extracted from the colData of the se object. Alternatively provide a DF with unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for metabolites that were not detected.
`metadata_sample`	*Optional:* Only required if you did not provide se file in parameter data. Provide DF which contains metadata information about the samples, which will be combined with your input data based on the unique sample identifiers used as rownames. **Default = NULL**
`scaling`	*Optional:* TRUE or FALSE for whether a data scaling is used **Default = TRUE**
`percentage`	*Optional:* percentage of top and bottom features to be displayed in the results summary. **Default = 0.1**
`cutoff_stat`	*Optional:* Cutoff for the adjusted p-value of the ANOVA test for the results summary and on the heatmap. **Default = 0.05**

`cutoff_variance`
: *Optional:* Cutoff for the PCs variance that should be displayed on the heatmap. **Default = 1**

`save_table`
: *Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**

`save_plot`
: *Optional:* Select the file type of output plots. Options are svg, png, pdf. **Default = svg**

`print_plot`
: *Optional:* TRUE or FALSE, if TRUE Volcano plot is saved as an overview of the results. **Default = TRUE**

`path`
: *Optional:* Path to the folder the results should be saved at. **default: NULL**

Value

List of DFs: prcomp results, loadings, top-Bottom features, annova results, results summary

Examples

```
data(tissue_norm)
d <- tissue_norm[1:100, -c(2:14)] %>% tibble::column_to_rownames("Code")
d <- d[, vapply(d, function(x) length(unique(x)) > 1, logical(1))]
Res <- metadata_analysis(
    data = d,
    metadata_sample = tissue_norm[1:100, c(1, 5:6)] %>%
        tibble::column_to_rownames("Code"),
    save_plot = NULL,
    save_table = NULL,
    print_plot = FALSE
)
```

MetaProViz *The MetaProViz package*

Description

MetaProViz (Metabolomics Processing, functional analysis and Visualization), a free open-source R-package that provides mechanistic hypotheses from metabolomics data by integrating prior knowledge from literature with metabolomics. MetaProViz offers an interactive framework consisting of five modules: Processing, differential analysis, prior knowledge access and refactoring, functional analysis and visualization of both intracellular and exometabolomics (consumption-release/core data).

Author(s)

Christina Schmidt and Denes Turei and Dimitrios Prymidis and Macabe Daley and Julio Saez-Rodriguez and Christian Frezza

See Also

Useful links:

- https://saezlab.github.io/MetaProViz
- Report bugs at https://github.com/saezlab/MetaProViz/issues

---

`metaproviz_config_path` *Current config file path of MetaProViz*

Description

Current config file path of MetaProViz

Usage

```
metaproviz_config_path(user = FALSE)
```

Arguments

`user`	Logical: prioritize the user level config even if a config in the current working directory is available.

Value

Character: path to the config file.

Examples

```
metaproviz_config_path()
```

`metaproviz_load_config` *Load the package configuration from a config file*

Description

Load the package configuration from a config file

Usage

```
metaproviz_load_config(path = NULL, title = "default", user = FALSE, ...)
```

Arguments

path	Path to the config file.
title	Load the config under this title. One config file might contain multple configurations, each identified by a title. If the title is not available the first section of the config file will be used.
user	Force to use the user level config even if a config file exists in the current directory. By default, the local config files have prioroty over the user level config.
...	Passed to yaml.load_file.

Value

Invisibly returns the config as a list.

Examples

```
metaproviz_load_config()
```

---

metaproviz_log *Browse the current MetaProViz log file*

Description

Browse the current MetaProViz log file

Usage

```
metaproviz_log()
```

Value

Returns NULL.

See Also

metaproviz_logfile

Examples

```
## Not run:
metaproviz_log()
# then you can browse the log file, and exit with `q`

## End(Not run)
```

metaproviz_logfile *Path to the current MetaProViz log file*

Description

Path to the current MetaProViz log file

Usage

```
metaproviz_logfile()
```

Value

Character: path to the current logfile, or NULL if no logfile is available.

See Also

metaproviz_log

Examples

```
metaproviz_logfile()
# [1] "path/metaproviz/metaproviz-log/metaproviz-20210309-1642.log"
```

metaproviz_reset_config *Restore the built-in default values of all config parameters of MetaProViz*

Description

Restore the built-in default values of all config parameters of MetaProViz

Usage

```
metaproviz_reset_config(save = NULL, reset_all = FALSE)
```

Arguments

save	If a path, the restored config will be also saved to this file. If TRUE, the config will be saved to the current default config path (see metaproviz_config_path).
reset_all	Reset to their defaults also the options already set in the R options.

Value

The config as a list.

See Also

metaproviz_load_config, metaproviz_save_config

Examples

```
# restore the defaults and write them to the default config file:
metaproviz_reset_config()
metaproviz_save_config()
```

`metaproviz_save_config`

Save the current package configuration

Description

Save the current package configuration

Usage

```
metaproviz_save_config(path = NULL, title = "default", local = FALSE)
```

Arguments

`path`	Path to the config file. Directories and the file will be created if don't exist.
`title`	Save the config under this title. One config file might contain multiple configurations, each identified by a title.
`local`	Save into a config file in the current directory instead of a user level config file. When loading, the config in the current directory has priority over the user level config.

Value

Returns `NULL`.

Examples

```
# restore the defaults and write them to the default config file:
metaproviz_reset_config()
metaproviz_save_config()
```

metaproviz_set_loglevel *Sets the log level for the package logger*

Description

Sets the log level for the package logger

Usage

```
metaproviz_set_loglevel(level, target = "logfile")
```

Arguments

level	Character or class loglevel. The desired log level.
target	Character, either 'logfile' or 'console'

Value

Returns NULL.

Examples

```
metaproviz_set_loglevel(logger::FATAL, target = "console")
```

meta_pk *Meta prior-knowledge*

Description

Meta prior-knowledge

Usage

```
meta_pk(
  data,
  metadata_sample,
  metadata_info = NULL,
  save_table = "csv",
  path = NULL
)
```

Arguments

data	SummarizedExperiment (se) file including assay and colData. If se file is provided, metadata_sample is extracted from the colData of the se object. Alternatively provide a DF with unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for metabolites that were not detected.
metadata_sample	*Optional:* Only required if you did not provide se file in parameter data. Provide DF which contains metadata information about the samples, which will be combined with your input data based on the unique sample identifiers used as rownames. **Default = NULL**
metadata_info	*Optional:* NULL or vector with column names that should be used, i.e. c("Age", "gender", "Tumour-stage"). **default: NULL**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
path	*Optional:* Path to the folder the results should be saved at. **default: NULL**

Value

DF with prior knowledge based on patient metadata

Examples

```
data(tissue_norm_se)
Res <- meta_pk(tissue_norm_se)

data(tissue_norm)
Tissue_Norm <- tissue_norm %>% tibble::column_to_rownames("Code")
Res <- meta_pk(
    data = Tissue_Norm[, -c(1:13)],
    metadata_sample = Tissue_Norm[, c(2, 4:5, 12:13)]
)
```

---

metsigdb_chemicalclass

Metabolite chemical classes from RaMP DB

Description

Metabolite chemical classes from RaMP DB

Usage

```
metsigdb_chemicalclass(
  version = "2.5.4",
  save_table = "csv",
  path = NULL,
  exclude_metabolites = "all"
)
```

Arguments

version	*Optional:* Version of the RaMP database loaded from OmniPathR. **default: "2.5.4"**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
path	Optional: String which is added to the resulting folder name **default: NULL**
exclude_metabolites	Optional metabolite classes to exclude: NULL (exclude nothing), "all" (default), or any combination of c("ions", "small_molecules", "xenobiotics", "atoms").

Value

A data frame containing the Prior Knowledge.

Examples

```
ChemicalClass <- metsigdb_chemicalclass()
```

---

metsigdb_kegg *KEGG pathways*

Description

KEGG pathways

Usage

```
metsigdb_kegg(exclude_metabolites = "all")
```

Arguments

exclude_metabolites	Optional metabolite classes to exclude: NULL (exclude nothing), "all" (default), or any combination of c("ions", "small_molecules", "xenobiotics", "atoms").

Value

A data frame containing the KEGG pathways suitable for ORA.

Examples

```
metsigdb_kegg()
```

`metsigdb_macdb` *Retrieve MACDB metabolite-cancer associations.*

Description

Retrieves metabolite-cancer associations from MACDB via OmnipathR and summarizes them to unique metabolite-cancer associations (by `term` and `Metabolite_PubchemID`).

Usage

```
metsigdb_macdb(exclude_metabolites = "all")
```

Arguments

`exclude_metabolites`
: Optional metabolite classes to exclude: NULL (exclude nothing), "all" (default), or any combination of c("ions", "small_molecules", "xenobiotics", "atoms").

Value

A data frame with one row per unique metabolite-cancer association, collapsed metadata columns, and summary metrics (`evidence_count`, `significance_count`, `association_score`).

`metsigdb_metalinks` *Annotated metabolite-protein interactions from MetalinksDB*

Description

Annotated metabolite-protein interactions from MetalinksDB

Usage

```
metsigdb_metalinks(
  types = NULL,
  cell_location = NULL,
  tissue_location = NULL,
  biospecimen_location = NULL,
  disease = NULL,
  pathway = NULL,
  hmdb_ids = NULL,
  uniprot_ids = NULL,
  save_table = "csv",
```

```
  path = NULL,
  exclude_metabolites = "all"
)
```

Arguments

types	Desired edge types. Options are: "lr", "pd", where 'lr' stands for 'ligand-receptor' and 'pd' stands for 'production-degradation'.**default: NULL**
cell_location	Desired metabolite cell locations. Pass selection using c("Select1", "Select2", "Selectn"). View options setting "?". Options are: "Cytoplasm", "Endoplasmic reticulum", "Extracellular", "Lysosome" , "Mitochondria", "Peroxisome", "Membrane", "Nucleus", "Golgi apparatus" , "Inner mitochondrial membrane". **default: NULL**
tissue_location	Desired metabolite tissue locations. Pass selection using c("Select1", "Select2", "Selectn"). View options setting "?". Options are: "Placenta", "Adipose Tissue","Bladder", "Brain", "Epidermis","Kidney", "Liver", "Neuron", "Pancreas", "Prostate", "Skeletal Muscle", "Spleen", "Testis", "Thyroid Gland", "Adrenal Medulla", "Erythrocyte","Fibroblasts", "Intestine", "Ovary", "Platelet", "All Tissues", "Semen", "Adrenal Gland", "Adrenal Cortex", "Heart", "Lung", "Hair", "Eye Lens", "Leukocyte", Retina", "Smooth Muscle", "Gall Bladder", "Bile", "Bone Marrow", "Blood", "Basal Ganglia", "Cartilage". **default: NULL**
biospecimen_location	Desired metabolite biospecimen locations.Pass selection using c("Select1", "Select2", "Selectn").View options setting "?". "Blood", "Feces", "Saliva", "Sweat", "Urine", "Breast Milk", "Cellular Cytoplasm", "Cerebrospinal Fluid (CSF)", "Amniotic Fluid" , "Aqueous Humour", "Ascites Fluid", "Lymph", "Tears", "Breath", "Bile", "Semen", "Pericardial Effusion".**default: NULL**
disease	Desired metabolite diseases.Pass selection using c("Select1", "Select2", "Selectn"). View options setting "?". **default: NULL**
pathway	Desired metabolite pathways.Pass selection using c("Select1", "Select2", "Selectn"). View options setting "?".**default: NULL**
hmdb_ids	Desired HMDB IDs.Pass selection using c("Select1", "Select2", "Selectn"). View options setting "?".**default: NULL**
uniprot_ids	Desired UniProt IDs.Pass selection using c("Select1", "Select2", "Selectn"). View options setting "?".**default: NULL**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
path	*Optional:* Path to the folder the results should be saved at. **default: NULL**
exclude_metabolites	Optional metabolite classes to exclude: NULL (exclude nothing), "all" (default), or any combination of c("ions", "small_molecules", "xenobiotics", "atoms").

Value

A data frame of metabolite-protein interactions from MetalinksDB.

Examples

```
metsigdb_metalinks()
```

`metsigdb_reactome` *Retrieve Reactome metabolite sets suitable for ORA.*

Description

Queries the OmniPath resource through OmniPathR to obtail Reactome pathway level metabolite sets.

Usage

```
metsigdb_reactome(
  species = "Homo sapiens",
  pathway_ids = NULL,
  out_path = NULL,
  exclude_metabolites = "all"
)
```

Arguments

`species`	String. Optionally specify pathways to query from a species via full name or three letter code. Default = "Homo sapiens". NULL for all species.
`pathway_ids`	String vector. Optionally provide pathway_ids to query. Default NULL to query all pathways.
`out_path`	String. Optionally save results as csv into out_path. Default NULL.
`exclude_metabolites`	Optional metabolite classes to exclude: NULL (exclude nothing), "all" (default), or any combination of c("ions", "small_molecules", "xenobiotics", "atoms").

Value

A tibble in long format containing one row per metabolite for the Reactome pathways.

Examples

```
## Not run:
df <- metsigdb_reactome()
head(df)

## End(Not run)
```

metsigdb_wikipathways *Retrieve WikiPathways metabolite mapping suitable for ORA.*

Description

Retrieves pathway to metabolite mappings from WikiPathways (via `wikipathways_metabolites_sparql()`) via OmnipathR and returns a long-format table with one metabolite identifier per row.

Usage

```
metsigdb_wikipathways(species = "Homo sapiens", exclude_metabolites = "all")
```

Arguments

`species`	Character. Species name. Default is `"Homo sapiens"`.
`exclude_metabolites`	Optional metabolite classes to exclude: NULL (exclude nothing), "all" (default), or any combination of c("ions", "small_molecules", "xenobiotics", "atoms").

Value

A tibble in long format with columns `pathway_id`, `pathway_name`, `pathway_url`, `n_metabolites_in_pathway`, and `metabolite_id`.

pool_estimation *Find metabolites with high variability across total pool samples*

Description

Find metabolites with high variability across total pool samples

Usage

```
pool_estimation(
  data,
  metadata_sample = NULL,
  metadata_info = NULL,
  cutoff_cv = 30,
  save_plot = "svg",
  save_table = "csv",
  print_plot = TRUE,
  path = NULL
)
```

Arguments

data	SummarizedExperiment (se) file including assay and colData. If se file is provided, metadata_sample is extracted from the colData of the se object. Alternatively provide a DF which contains unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for metabolites that were not detected.
metadata_sample	*Optional:* Only required if you did not provide se file in parameter data. Provide DF which contains information about the samples, which will be combined with the input data based on the unique sample identifiers used as rownames. Must contain column with Conditions. If you do not have multiple conditions in your experiment assign all samples into the same condition. **Default = NULL**
metadata_info	*Optional:* NULL or Named vector including the Conditions and PoolSample information (Name of the Conditions column and Name of the pooled samples in the Conditions in the Input_SettingsFile) : c(Conditions="ColumnNameConditions, PoolSamples=NamePoolCondition. If no Conditions is added in the Input_metadata_info, it is assumed that the conditions column is named 'Conditions' in the Input_SettingsFile.). **Default = NULL**
cutoff_cv	*Optional:* Filtering cutoff for high variance metabolites using the Coefficient of Variation. **Default = 30**
save_plot	*Optional:* Select the file type of output plots. Options are svg, png, pdf or NULL. **Default = svg**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt", ot NULL **default: "csv"**
print_plot	*Optional:* If TRUE prints an overview of resulting plots. **Default = TRUE**
path	*Optional:* Path to the folder the results should be saved at. **default: NULL**

Value

List with two elements: DF (including input and output table) and Plot (including all plots generated)

Examples

```
data(intracell_raw_se)
Res <- pool_estimation(
    data = intracell_raw_se,
    metadata_info = c(PoolSamples = "Pool", Conditions = "Conditions")
)

data(intracell_raw)
Intra <- intracell_raw %>% tibble::column_to_rownames("Code")
Res <- pool_estimation(
    data = Intra[, -c(1:3)],
    metadata_sample = Intra[, c(1:3)],
    metadata_info = c(PoolSamples = "Pool", Conditions = "Conditions")
)
```

`processing` *Data preprocessing and normalization*

Description

Applies modular normalization including 80% filtering rule, total-ion count normalization, missing value imputation, and outlier detection using Hotelling's T2 test.

Usage

```
processing(
  data,
  metadata_sample = NULL,
  metadata_info = c(Conditions = "Conditions"),
  featurefilt = "Modified",
  cutoff_featurefilt = 0.8,
  tic = TRUE,
  mvi = TRUE,
  mvi_percentage = 50,
  hotellins_confidence = 0.99,
  core = FALSE,
  save_plot = "svg",
  save_table = "csv",
  print_plot = TRUE,
  path = NULL
)
```

Arguments

`data`	SummarizedExperiment or data frame. If SummarizedExperiment, metadata_sample is extracted from colData. If data frame, provide unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for undetected metabolites.
`metadata_sample`	Data frame (optional). Only required if data is not a SummarizedExperiment. Contains information about samples, combined with input data based on unique sample identifiers used as row names. Must contain Conditions column. If experiment has no multiple conditions, assign all samples to same condition. Default: NULL.
`metadata_info`	Named character vector (optional). Contains names of experimental parameters: c(Conditions="ColumnName", Biological_Replicates="ColumnName"). Column "Conditions" (mandatory) contains sample conditions (e.g., "N"/"T" or "Normal"/"Tumor"), used for feature filtering and PCA color coding. Column "BiologicalReplicates" (optional) contains numerical values. For core=TRUE, must also add core_norm_factor="ColumnName" and core_media="ColumnName". Column core_norm_factor is used for normalization; core_media specifies media controls in Conditions. Default: c(Conditions="Conditions").

`featurefilt` Character (optional). If NULL, no feature filtering is performed. If "Standard", applies 80% filtering rule (Bijlsma et al., 2006) on metabolite features across whole dataset. If "Modified", filtering is done per condition and Conditions column must be provided (Yang et al., 2015). Default: "Standard".

`cutoff_featurefilt` Numeric (optional). Percentage threshold for feature filtering. Default: 0.8.

`tic` Logical (optional). Whether total ion count normalization is performed. Default: TRUE.

`mvi` Logical (optional). Whether missing value imputation (MVI) based on half minimum is performed. Default: TRUE.

`mvi_percentage` Numeric (optional). Percentage (0-100) of imputed value based on minimum value. Default: 50.

`hotellins_confidence` Numeric (optional). Confidence level for outlier identification in Hotelling's T2 test. Default: 0.99.

`core` Logical (optional). Whether consumption-release experiment was performed and core value should be calculated. If TRUE, provide normalization factor column "core_norm_factor" in metadata_sample where Conditions column matches. The normalization factor must be numerical value from growth rate (growth curve) or growth factor (ratio of cell count/protein quantification at start vs. end point). Additionally, control media samples must be available in data and defined as "core_media" in Conditions column of metadata_sample. Default: FALSE.

`save_plot` Character (optional). File type of output plots: "svg", "png", "pdf". If NULL, plots are not saved. Default: "svg".

`save_table` Character (optional). File type of output table: "csv", "xlsx", "txt". If NULL, tables are not saved. Default: "csv".

`print_plot` Logical (optional). Whether to print overview of resulting plots. Default: TRUE.

`path` Character (optional). Path to folder where results should be saved. Default: NULL.

Value

List with two elements: DF (all output tables generated) and Plot (all plots generated).

Examples

```
data(intracell_raw)
Intra <- intracell_raw %>% tibble::column_to_rownames("Code")
ResI <- processing(
    data = Intra[1:30, -c(1:3)],
    metadata_sample = Intra[1:30, c(1:3)],
    metadata_info = c(
        Conditions = "Conditions",
        Biological_Replicates = "Biological_Replicates"
    ),
    save_plot = NULL,
```

```
    save_table = NULL,
    print_plot = FALSE
)
```

`reexports` *Objects exported from other packages*

Description

These objects are imported from other packages. Follow the links below to see their documentation.

magrittr `%>%`

Value

The left-hand side object, passed into the function on the right-hand side.

`replicate_sum` *Merges the analytical replicates of an experiment*

Description

Merges the analytical replicates of an experiment

Usage

```
replicate_sum(
  data,
  metadata_sample = NULL,
  metadata_info = c(Conditions = "Conditions", Biological_Replicates =
    "Biological_Replicates", Analytical_Replicates = "Analytical_Replicates"),
  save_table = "csv",
  path = NULL
)
```

Arguments

`data`	SummarizedExperiment (se) file including assay and colData. If se file is provided, metadata_sample is extracted from the colData of the se object. Alternatively provide a DF which contains unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for metabolites that were not detected.

metadata_sample
: *Optional:* Only required if you did not provide se file in parameter data. Provide DF which contains information about the samples, which will be combined with the input data based on the unique sample identifiers used as rownames. Must contain column with Conditions. If you do not have multiple conditions in your experiment assign all samples into the same condition. **Default = NULL**

metadata_info
: *Optional:* Named vector including the Conditions and Replicates information: c(Conditions="ColumnNameConditions", Biological_Replicates="ColumnName_metadata_sample", Analytical_Replicates="ColumnName_metadata_sample").**Default = NULL**

save_table
: *Optional:* File types for the analysis results are: "csv", "xlsx", "txt", ot NULL **default: "csv"**

path
: *Optional:* Path to the folder the results should be saved at. **default: NULL**

Value

DF with the merged analytical replicates

Examples

```
data(intracell_raw_se)
Res <- replicate_sum(data = intracell_raw_se)

data(intracell_raw)
Intra <- intracell_raw %>% tibble::column_to_rownames("Code")
Res <- replicate_sum(
    data = Intra[-c(49:58), -c(1:3)],
    metadata_sample = Intra[-c(49:58), c(1:3)]
)
```

---

standard_ora *Overrepresentation analysis of metabolite sets in pathways*

Description

Can be applied on the result of differential metabolite analysis (DMA), requires a pathway list (from databases).

Usage

```
standard_ora(
  data,
 metadata_info = c(pvalColumn = "p.adj", percentageColumn = "t.val", PathwayTerm =
    "term", PathwayFeature = "Metabolite"),
  cutoff_stat = 0.05,
  cutoff_percentage = 10,
  input_pathway,
```

```
  pathway_name = "",
  min_gssize = 10,
  max_gssize = 1000,
  save_table = "csv",
  path = NULL
)
```

Arguments

data	DF with metabolite names/metabolite IDs as row names. Metabolite names/IDs need to match the identifier type (e.g. HMDB IDs) in the input_pathway.
metadata_info	*Optional:* Pass ColumnName of the column including parameters to use for cutoff_stat and cutoff_percentage. Also pass ColumnName for input_pathway including term and feature names. (pvalColumn = ColumnName data, percentageColumn= ColumnName data, PathwayTerm= ColumnName input_pathway, PathwayFeature= ColumnName input_pathway) **c(pvalColumn="p.adj", percentageColumn="t.val", PathwayTerm= "term", PathwayFeature= "Metabolite")**
cutoff_stat	*Optional:* p-adjusted value cutoff from ORA results. Must be a numeric value. **default: 0.05**
cutoff_percentage	*Optional:* percentage cutoff of metabolites that should be considered for ORA. Selects top and bottom percentage of selected numeric variable, usually t.val or Log2FC **default: 10**
input_pathway	DF that must include column "term" with the pathway name, column "Metabolite" with the Metabolite name or ID and column "Description" with pathway description that will be depicted on the plots.
pathway_name	*Optional:* Name of the input_pathway used **default: ""**
min_gssize	*Optional:* minimum group size in ORA **default: 10**
max_gssize	*Optional:* maximum group size in ORA **default: 1000**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt" **default: "csv"**
path	*Optional:* Path to the folder the results should be saved at. **default: NULL**

Value

Saves results as individual .csv files.

Examples

```
KEGG_Pathways <- metsigdb_kegg()
data(intracell_dma) # loads the object into your environment
DMAres <- intracell_dma %>%
    dplyr::filter(!is.na(KEGGCompound)) %>%
    tibble::column_to_rownames("KEGGCompound") %>%
    dplyr::select(-"Metabolite")
RES <- standard_ora(
```

```
    data = DMAres,
    input_pathway = KEGG_Pathways
)
```

`tissue_dma` *tissue_dma*

Description

We performed differential metabolite analysis comparing ccRCC tissue versus adjacent normal tissue using median normalised data from the supplementary table 2 of Hakimi et. al.(="Tissue_Norm"). metabolite values used as input with row names being metabolite trivial names. doi:10.1016/j.ccell.2015.12.004

Usage

```
tissue_dma
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 570 rows and 17 columns.

Examples

```
data(tissue_dma)
head(tissue_dma)
```

`tissue_dma_old` *tissue_dma_old*

Description

We performed differential metabolite analysis comparing ccRCC tissue versus adjacent normal tissue of the patient's subset of old patient's (age > 58 years) using median normalised data from the supplementary table 2 of Hakimi et. al.(="Tissue_Norm"). metabolite values used as input with row names being metabolite trivial names. doi:10.1016/j.ccell.2015.12.004

Usage

```
tissue_dma_old
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 570 rows and 17 columns.

Examples

```
data(tissue_dma_old)
head(tissue_dma_old)
```

`tissue_dma_young` *tissue_dma_young*

Description

We performed differential metabolite analysis comparing ccRCC tissue versus adjacent normal tissue of the patient's subset of young patient's (age <42 years) using median normalised data from the supplementary table 2 of Hakimi et. al.(="Tissue_Norm"). metabolite values used as input with row names being metabolite trivial names. doi:10.1016/j.ccell.2015.12.004

Usage

```
tissue_dma_young
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 570 rows and 17 columns.

Examples

```
data(tissue_dma_young)
head(tissue_dma_young)
```

`tissue_meta` *Tissue_Metadata*

Description

In Hakimi et. al. metabolites were assigned to metabolite IDs, pathways, platform, mass and other fetaure metainformation. row names being metabolite trivial names. doi:10.1016/j.ccell.2015.12.004

Usage

```
tissue_meta
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 877 rows and 11 columns.

Examples

```
data(tissue_meta)
head(tissue_meta)
```

`tissue_norm` *tissue_norm*

Description

This is median normalised data from the supplementary table 2 of Hakimi et al with metabolomic profiling on 138 matched clear cell renal cell carcinoma (ccRCC)/normal tissue pairs. measured metabolite (normalised data) doi:10.1016/j.ccell.2015.12.004

Usage

```
tissue_norm
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 276 rows and 584 columns.

Examples

```
data(tissue_norm)
head(tissue_norm)
```

`tissue_norm_se` *tissue_norm_se*

Description

This is median normalised data from the supplementary table 2 of Hakimi et al with metabolomic profiling on 138 matched clear cell renal cell carcinoma (ccRCC)/normal tissue pairs. coldata include patient metadata (e.g. age, gender, sage, etc.) doi:10.1016/j.ccell.2015.12.004

Usage

```
tissue_norm_se
```

Format

An object of class `SummarizedExperiment` with 570 rows and 276 columns.

Examples

```
data(tissue_norm_se)
head(tissue_norm_se)
```

---

tissue_tvn_proteomics *tissue_tvn_proteomics*

Description

The processed proteomics data was downloaded from the supplementary table 3 of Mora & Schmidt et. al., which used the study from Clark et. al. under Proteomics data Commons PDC000127. on their regulation regulation in renal cancer, Genome Medicine 2024, doi:10.1186/s13073024014153 Clark et. al, Integrated proteogenomic characterization of clear cell renal cell carcinoma, Cell 2019, doi:10.1016/j.cell.2019.10.007

Usage

```
tissue_tvn_proteomics
```

Format

An object of class `spec_tbl_df` (inherits from `tbl_df`, `tbl`, `data.frame`) with 8769 rows and 10 columns.

Examples

```
data(tissue_tvn_proteomics)
head(tissue_tvn_proteomics)
```

---

tissue_tvn_rnaseq *tissue_tvn_rnaseq*

Description

The processed transcriptomics data was downloaded from the supplementary table 3 of Mora & Schmidt et. al., which used the study from Clark et. al. under Proteomics data Commons PDC000127. on their regulation regulation in renal cancer, Genome Medicine 2024, doi:10.1186/s13073024-014153 Clark et. al, Integrated proteogenomic characterization of clear cell renal cell carcinoma, Cell 2019, doi:10.1016/j.cell.2019.10.007

Usage

```
tissue_tvn_rnaseq
```

Format

An object of class spec_tbl_df (inherits from tbl_df, tbl, data.frame) with 29283 rows and 10 columns.

Examples

```
data(tissue_tvn_rnaseq)
head(tissue_tvn_rnaseq)
```

---

translate_id *Translate IDs to/from KEGG, PubChem, Chebi, HMDB*

Description

Translate IDs to/from KEGG, PubChem, Chebi, HMDB

Usage

```
translate_id(
  data,
  metadata_info = c(InputID = "MetaboliteID", grouping_variable = "term"),
  from = "kegg",
  to = c("pubchem", "chebi", "hmdb", "cas"),
  summary = FALSE,
  save_table = "csv",
  path = NULL
)
```

Arguments

data	dataframe with at least one column with the target (e.g. metabolite), you can add other columns such as source (e.g. term). Must be "long" DF, meaning one ID per row.
metadata_info	*Optional:* Column name of Target in input_pk. **Default = list(InputID="MetaboliteID" , grouping_variable="term")**
from	ID type that is present in your data. Choose between "kegg", "pubchem", "chebi", "hmdb", "cas". **Default = "kegg"**
to	One or multiple ID types to which you want to translate your data. Choose between "kegg", "pubchem", "chebi", "hmdb", "cas". **Default = c("pubchem","chebi","hmdb","cas")**
summary	*Optional:* If TRUE a long summary tables are created. **Default = FALSE**
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". **Default = "csv"**
path	Optional: Path to the folder the results should be saved at. **Default = NULL**

Value

List with at least three DFs: 1) Original data and the new column of translated ids spearated by comma. 2) Mapping information between Original ID to Translated ID. 3) Mapping summary between Original ID to Translated ID.

Examples

```
## Not run:
KEGG_Pathways <- metsigdb_kegg()
Res <- translate_id(
    data = KEGG_Pathways,
    metadata_info = c(
        InputID = "MetaboliteID",
        grouping_variable = "term"
    ),
    from = c("kegg"),
    to = c("pubchem", "hmdb")
)

## End(Not run)
```

`traverse_ids` *Expand metabolite IDs by traversing RaMP ID mappings*

Description

Traverses pairwise RaMP mappings from `OmnipathR::ramp_id_mapping_table()` across selected metabolite ID types until no new IDs are found.

Usage

```
traverse_ids(
  data,
  id_types = c("HMDB", "KEGG", "CHEBI", "PUBCHEM"),
  delimiter = c(";", ","),
  save_table = "csv",
  path = NULL,
  verbose = FALSE
)
```

Arguments

`data`	Data frame with zero or more of the columns `HMDB`, `KEGG`, `CHEBI`, and `PUBCHEM`. Column names are matched case-insensitively against these exact names.
`id_types`	Character vector of ID types to expand. Choose from `HMDB`, `KEGG`, `CHEBI`, and `PUBCHEM`.

delimiter	Character string indicating whether multiple IDs within one cell are separated by semicolons or commas. Accepted values are ";", ",", "semicolon", or "comma".
save_table	*Optional:* File types for the analysis results are: "csv", "xlsx", "txt". If NULL, no tables are saved. **Default = "csv"**
path	*Optional:* Path to the folder the results should be saved at. **Default = NULL**
verbose	Logical; if TRUE, prints pairwise mapping and edge construction diagnostics to the console.

Value

Named list with three data frames:

ExpandedDF	Input data with appended expanded ID columns and QC summary columns, including all_seed_ids_compatible (logical flag indicating whether all seed IDs in each row are mutually reachable through IDEdges).
ExpandedIDs_Long	Long-format table with row_id, type, id, and is_seed.
IDEdges	Bidirectional ID edge table used for traversal.

Examples

```
input_df <- data.frame(
    name = c(
        "Acetone ; Propanal ; acetone",
        "Acetaldehyde oxime ; HMDB01122",
        "acetate",
        "Urea"
    ),
    all_ids = c(
        "HMDB01659 ; HMDB03366 ; C00207",
        "HMDB03656 ; HMDB01122",
        "C00033",
        "C00086"
     ),
     HMDB = c(
        "HMDB01659; HMDB03366",
        "HMDB03656; HMDB01122",
        NA,
        NA
     ),
     KEGG = c(
        "C00207",
        NA,
        "C00033",
        "C00086"
     ),
     CHEBI = NA,
     stringsAsFactors = FALSE
)
```

```
res <- traverse_ids(input_df)

df_translated <- res$ExpandedDF
head(df_translated)
```

---

viz_graph *Graph visualization for clustered terms*

Description

Graph visualization for clustered terms

Usage

```
viz_graph(
  similarity_matrix,
  clusters,
  plot_threshold = 0.5,
  plot_name = "ClusterGraph",
  max_nodes = NULL,
  min_degree = NULL,
  node_sizes = NULL,
  show_density = FALSE,
  save_plot = "svg",
  print_plot = TRUE,
  path = NULL,
  plot_width = 3000,
  plot_height = 2000,
  plot_unit = "px"
)
```

Arguments

similarity_matrix	Square numeric matrix of term similarity values. Row/column names must match `clusters` names.
clusters	Named vector of cluster labels for each term (e.g., "cluster1").
plot_threshold	Similarity threshold used to define edges. Values below the threshold are removed. Default = 0.5.
plot_name	*Optional:* String added to output files of the plot. Default = "ClusterGraph".
max_nodes	*Optional:* Maximum nodes for plotting. If set, keeps nodes from the largest component up to this limit (by degree).
min_degree	*Optional:* Minimum degree filter for graph plotting.

node_sizes	*Optional:* Named numeric vector of node sizes, with names matching term IDs. Values are scaled for plotting.
show_density	*Optional:* If TRUE, add a hull background per cluster. Default = FALSE.
save_plot	*Optional:* Select the file type of output plots. Options are svg, pdf, png or NULL. **Default = "svg"**
print_plot	*Optional:* If TRUE prints an overview of resulting plots. **Default = TRUE**
path	Optional: String which is added to the resulting folder name. **default: NULL**
plot_width	*Optional:* Plot width passed to save_res. Default = 3000.
plot_height	*Optional:* Plot height passed to save_res. Default = 2000.
plot_unit	*Optional:* Unit for plot dimensions passed to save_res. Default = "px".

Value

Graph plot as a ggplot object.

Examples

```
# Create toy similarity matrix and clusters
sim <- matrix(
    c(1, 0.8, 0.3, 0.1,
      0.8, 1, 0.2, 0.1,
      0.3, 0.2, 1, 0.7,
      0.1, 0.1, 0.7, 1),
    nrow = 4,
    dimnames = list(
        c("Pathway_A", "Pathway_B", "Pathway_C", "Pathway_D"),
        c("Pathway_A", "Pathway_B", "Pathway_C", "Pathway_D")
    )
)
clusters <- c(
    Pathway_A = "1", Pathway_B = "1",
    Pathway_C = "2", Pathway_D = "2"
)
viz_graph(
    sim, clusters,
    plot_threshold = 0.2,
    save_plot = NULL,
    print_plot = FALSE
)
```

---

viz_heatmap *Heatmap visualization*

Description

Heatmap visualization

Usage

```
viz_heatmap(
  data,
  metadata_info = NULL,
  metadata_sample = NULL,
  metadata_feature = NULL,
  plot_name = "",
  scale = "row",
  save_plot = "svg",
  enforce_featurenames = FALSE,
  enforce_samplenames = FALSE,
  print_plot = TRUE,
  path = NULL
)
```

Arguments

data	SummarizedExperiment (se) file including assay and colData. If se file is provided, metadata_sample is extracted from the colData of the se object. metadata_feature, if available, are extracted from the rowData. Alternatively provide a DF with unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for metabolites that were not detected.
metadata_info	*Optional:* NULL or Named vector where you can include vectors or lists for annotation c(individual_Metab= "ColumnName_metadata_feature",individual_Sample= "ColumnName_metadata_sample", color_Metab="ColumnName_metadata_feature", color_Sample= list("ColumnName_metadata_sample", "ColumnName_metadata_sample",...)).**Default = NULL**
metadata_sample	*Optional:* Only required if you did not provide se file in parameter data. Provide DF which contains metadata information about the samples, which will be combined with your input data based on the unique sample identifiers used as rownames. **Default = NULL**
metadata_feature	*Optional:* To provide metadata information for each metabolite. Only used if you did not provide se file in parameter data. Provide DF where the row names must match the metabolite names in the columns of the data. **Default = NULL**
plot_name	*Optional:* String which is added to the output files of the plot
scale	*Optional:* String with the information for scale row, column or none. **Default = row**
save_plot	*Optional:* Select the file type of output plots. Options are svg, pdf, png or NULL. **Default = "svg"**
enforce_featurenames	*Optional:* If there are more than 100 features no rownames will be shown, which is due to readability. You can Enforce this by setting this parameter to TRUE. **Default = FALSE**

enforce_samplenames
: *Optional:* If there are more than 50 sampless no colnames will be shown, which is due to readability. You can Enforce this by setting this parameter to TRUE. **Default = FALSE**

print_plot
: *Optional:* print the plots to the active graphic device.

path
: Optional: String which is added to the resulting folder name **default: NULL**

Value

List with two elements: Plot and Plot_Sized

Examples

```
data(intracell_raw_se)
Res <- viz_heatmap(data = intracell_raw_se)

data(intracell_raw)
Intra <- intracell_raw %>% tibble::column_to_rownames("Code")
Res <- viz_heatmap(data = Intra[, -c(1:3)])
```

---

viz_pca

This script allows you to perform PCA plot visualization using the results of the MetaProViz analysis

---

Description

PCA plot visualization

Usage

```
viz_pca(
  data,
  metadata_info = NULL,
  metadata_sample = NULL,
  color_palette = NULL,
  scale_color = "discrete",
  shape_palette = NULL,
  show_loadings = FALSE,
  scaling = TRUE,
  pcx = 1,
  pcy = 2,
  theme = NULL,
  plot_name = "",
  save_plot = "svg",
  print_plot = TRUE,
  path = NULL
)
```

Arguments

data	SummarizedExperiment (se) file including assay and colData. If se file is provided, metadata_sample is extracted from the colData of the se object. metadata_feature, if available, are extracted from the rowData. Alternatively provide a DF with unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for metabolites that were not detected.
metadata_info	*Optional:* NULL or Named vector including at least one of those three information : c(color="ColumnName_Plot_SettingsFile", shape= "ColumnName_Plot_SettingsFile"). **Default = NULL**
metadata_sample	*Optional:* Only required if you did not provide se file in parameter data. Provide DF which contains metadata information about the samples, which will be combined with your input data based on the unique sample identifiers used as rownames. **Default = NULL**
color_palette	*Optional:* Provide customiced color-palette in vector format. For continuous scale use e.g. scale_color_gradient(low = "#88CCEE", high = "red") and for discrete scale c("#88CCEE", "#DDCC77","#661100", "#332288")**Default = NULL**
scale_color	*Optional:* Either "continuous" or "discrete" colour scale. For numeric or integer you can choose either, for character you have to choose discrete. **Default = NULL**
shape_palette	*Optional:* Provide customiced shape-palette in vector format. **Default = NULL**
show_loadings	*Optional:* TRUE or FALSE for whether PCA loadings are also plotted on the PCA (biplot) **Default = FALSE**
scaling	*Optional:* TRUE or FALSE for whether a data scaling is used **Default = TRUE**
pcx	*Optional:* Numeric value of the PC that should be plotted on the x-axis **Default = 1**
pcy	*Optional:* Numeric value of the PC that should be plotted on the y-axis **Default = 2**
theme	*Optional:* Selection of theme for plot, e.g. theme_grey(). You can check for complete themes here: https://ggplot2.tidyverse.org/reference/ggtheme.html. If default=NULL we use theme_classic(). **Default = "discrete"**
plot_name	*Optional:* String which is added to the output files of the PCA **Default = ""**
save_plot	*Optional:* Select the file type of output plots. Options are svg, png, pdf or NULL. **Default = svg**
print_plot	*Optional:* TRUE or FALSE, if TRUE Volcano plot is saved as an overview of the results. **Default = TRUE**
path	*Optional:* Path to the folder the results should be saved at. **default: NULL**

Value

List with two elements: Plot and Plot_Sized

Examples

```
data(intracell_raw_se)
Res <- viz_pca(intracell_raw_se)

data(intracell_raw)
Intra <- intracell_raw[, -c(2:4)] %>% tibble::column_to_rownames("Code")
Res <- viz_pca(Intra)
```

`viz_superplot` *This script allows you to perform different visualizations (bar, box, violin plots) using the results of the MetaProViz analysis*

Description

Bar, Box or Violin plot in Superplot style visualization

Usage

```
viz_superplot(
  data,
  metadata_sample = NULL,
  metadata_info = c(Conditions = "Conditions", Superplot = NULL),
  plot_type = "Box",
  plot_name = "",
  plot_conditions = NULL,
  stat_comparison = NULL,
  pval = NULL,
  padj = NULL,
  xlab = NULL,
  ylab = NULL,
  theme = NULL,
  color_palette = NULL,
  color_palette_dot = NULL,
  save_plot = "svg",
  print_plot = TRUE,
  path = NULL
)
```

Arguments

`data`	SummarizedExperiment (se) file including assay and colData. If se file is provided, metadata_sample is extracted from the colData of the se object. metadata_feature, if available, are extracted from the rowData. Alternatively provide a DF with unique sample identifiers as row names and metabolite numerical values in columns with metabolite identifiers as column names. Use NA for metabolites that were not detected.

`metadata_sample`
: *Optional:* Only required if you did not provide se file in parameter data. Provide DF which contains metadata information about the samples, which will be combined with your input data based on the unique sample identifiers used as rownames. **Default = NULL**

`metadata_info`
: Named vector including at least information on the conditions column: c(Conditions="ColumnName_met
Additionally Superplots can be made by adding Superplot ="ColumnName_metadata_sample",
which are usually biological replicates or patient IDs. **Default = c(Conditions="Conditions", Superplot = NULL)**

`plot_type`
: String with the information of the Graph style. Available options are Bar. Box and Violin **Default = Box**

`plot_name`
: *Optional:* String which is added to the output files of the plot.

`plot_conditions`
: Vector with names of selected Conditions for the plot. Can also be used to order the Conditions in the way they should be displayed on the x-axis of the plot. **Default = NULL**

`stat_comparison`
: List of numeric vectors containing Condition pairs to compare based on the order of the plot_conditions vector. **Default = NULL**

`pval`
: *Optional:* String which contains an abbreviation of the selected test to calculate p.value. For one-vs-one comparisons choose t.test or wilcox.test , for one-vs-all or all-vs-all comparison choose aov (=anova) or kruskal.test **Default = NULL**

`padj`
: *Optional:* String which contains an abbreviation of the selected p.adjusted test for p.value correction for multiple Hypothesis testing. Search: ?p.adjust for more methods:"BH", "fdr", "bonferroni", "holm", etc.**Default = NULL**

`xlab`
: *Optional:* String to replace x-axis label in plot. **Default = NULL**

`ylab`
: *Optional:* String to replace y-axis label in plot. **Default = NULL**

`theme`
: *Optional:* Selection of theme for plot, e.g. theme_grey(). You can check for complete themes here: https://ggplot2.tidyverse.org/reference/ggtheme.html. **Default = NULL**

`color_palette`
: *Optional:* Provide customized color_palette in vector format. **Default = NULL**

`color_palette_dot`
: *Optional:* Provide customized color_palette in vector format. **Default = NULL**

`save_plot`
: *Optional:* Select the file type of output plots. Options are svg, pdf, png or NULL. **Default = svg**

`print_plot`
: *Optional:* TRUE or FALSE, if TRUE plots are saved as an overview of the results. **Default = TRUE**

`path`
: *Optional:* Path to the folder the results should be saved at. **Default = NULL**

Value

List with two elements: Plot and Plot_Sized

Examples

```
data(intracell_raw_se)
# only plot the first 2 metabolites
Res <- viz_superplot(data = intracell_raw_se[1:2, , drop = FALSE])

data(intracell_raw)
Intra <- intracell_raw[, c(1:6)] %>% tibble::column_to_rownames("Code")
Res <- viz_superplot(
    data = Intra[, -c(1:3)],
    metadata_sample = Intra[, c(1:3)]
)
```

---

viz_volcano *Volcano plot*

Description

Volcano plot

Usage

```
viz_volcano(
  plot_types = "Standard",
  data,
  metadata_info = NULL,
  metadata_feature = NULL,
  data2 = NULL,
  y = "p.adj",
  x = "Log2FC",
  xlab = NULL,
  ylab = NULL,
  cutoff_x = 0.5,
  cutoff_y = 0.05,
  connectors = FALSE,
  select_label = "",
  plot_name = "",
  subtitle = "",
  name_comparison = c(data = "Cond1", data2 = "Cond2"),
  color_palette = NULL,
  shape_palette = NULL,
  theme = NULL,
  save_plot = "svg",
  path = NULL,
  feature = "Metabolites",
  print_plot = TRUE
)
```

Arguments

plot_types	*Optional:* Choose between "Standard" (data), "Compare" (plot two comparisons together data and data2) or "PEA" (Pathway Enrichment Analysis) **Default = "Standard"**
data	DF with metabolites as row names and columns including Log2FC and stat (p-value, p.adjusted) value columns.
metadata_info	*Optional:* NULL or Named vector including at least one of those three information for Settings="Standard" or "Compare": c(color ="ColumnName_metadata_feature", shape = "ColumnName_metadata_feature", individual="ColumnName_metadata_feature"). For Settings="PEA" a named vector with: PEA_Pathway="ColumnName_data2"=each pathway will be plotted, PEA_score="ColumnName_data2", PEA_stat= "ColumnName_data2"= usually p.adj column, "PEA_Feature="ColumnName_data2"= usually Metabolites), optionally you can additionally include c(color_Metab="ColumnName_metadata_fe shape= "ColumnName_metadata_feature").**Default = NULL**
metadata_feature	*Optional:* DF with column including the Metabolite names (needs to match Metabolite names and Metabolite column name of data) and other columns with required plot_typeInfo. **Default = NULL**
data2	*Optional:* DF to compare to main Input_data with the same column names x and y (Settings="Compare") and metabolites as row names or Pathway enrichment analysis results (Settings="PEA"). **Default = NULL**
y	*Optional:* Column name including the values that should be used for y-axis. Usually this would include the p.adjusted value. **Default = "p.adj"**
x	*Optional:* Column name including the values that should be used for x-axis. Usually this would include the Log2FC value. **Default = "Log2FC"**
xlab	*Optional:* String to replace x-axis label in plot. **Default = NULL**
ylab	*Optional:* String to replace y-axis label in plot. **Default = NULL**
cutoff_x	*Optional:* Number of the desired log fold change cutoff for assessing significance. **Default = 0.5**
cutoff_y	*Optional:* Number of the desired p value cutoff for assessing significance. **Default = 0.05**
connectors	*Optional:* TRUE or FALSE for whether connectors from names to points are to be added to the plot. **Default = FALSE**
select_label	*Optional:* If set to NULL, feature labels will be plotted randomly. If vector is provided, e.g. c("MetaboliteName1", "MetaboliteName2"), selected names will be plotted. If set to default "", no feature names will be plotted. **Default = ""**
plot_name	*Optional:* String which is added to the output files of the plot. **Default = ""**
subtitle	*Optional:* **Default = ""**
name_comparison	*Optional:* Named vector including those information about the two datasets that are compared on the plots when choosing Settings= "Compare". **Default = c(data="Cond1", data2= "Cond2")**
color_palette	*Optional:* Provide customiced color-palette in vector format. **Default = NULL**

shape_palette	*Optional:* Provide customiced shape-palette in vector format. **Default = NULL**
theme	*Optional:* Selection of theme for plot, e.g. theme_grey(). You can check for complete themes here: https://ggplot2.tidyverse.org/reference/ggtheme.html. **Default = NULL**
save_plot	*Optional:* Select the file type of output plots. Options are svg, pdf, png or NULL. **Default = "svg"**
path	Optional: Path to the folder the results should be saved at. **default: NULL**
feature	*Optional:* Name of the feature that are plotted, e.g. "Metabolites", "RNA", "Proteins", "Genes", etc. **Default = "metabolites"**
print_plot	*Optional:* print the plots to the active graphic device.

Value

List with two elements: Plot and Plot_Sized

Examples

```
data(intracell_dma)
Intra <- intracell_dma %>% tibble::column_to_rownames("Metabolite")
Res <- viz_volcano(data = Intra)
```

Index