Package 'enrichmet'

November 7, 2026

Type Package

Title enrichmet: R package for Quick and Easy Pathway and Network Analyses in Metabolomics

Version 0.99.18

Description Provides functions to perform metabolite-based pathway enrichment analysis using Fisher's exact test and MetSEA. Computes relative betweenness centrality (RBC) of input metabolites in a pathway-metabolite interaction network. The package generates visualizations including pathway dot plots, MetSEA plots, metabolite centrality plots, and network plots. It supports KEGG-based pathway inputs and generates exportable results.

License AGPL-3 + file LICENSE

Encoding UTF-8

URL https://github.com/biodatalab/enrichmet

BugReports https://github.com/biodatalab/enrichmet/issues

Depends R (>= 4.6.0)

Imports BiocFileCache, methods, circlize, ComplexHeatmap, curl, data.table, dplyr, fgsea, ggplot2, ggraph, ggrepel, igraph, limma, openxlsx, qvalue, rlang, scales, stats, stringr, tibble, tidyr, utils, SummarizedExperiment

Suggests knitr, rmarkdown, BiocStyle, testthat (>= 3.2.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

biocViews Metabolomics, Pathways, Network, NetworkEnrichment, KEGG, Visualization

Config/roxygen2/version 8.1.0

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

git_branch devel

git_last_commit 3141d80

git_last_commit_date 2026-10-09

Repository Bioconductor 3.24

Date/Publication 2026-11-06

Author Yonatan Mekonnen [aut, cre] (ORCID:
<https://orcid.org/0000-0002-7430-9428>),
H. Lee Moffitt Cancer Center & Research Institute [fnd] (Grant number:
P01CA250984)

Maintainer Yonatan Mekonnen <yaayalew@gmail.com>

Contents

calculate_metabolite_centrality . . . 2
create_centrality_plot . . . 3
create_enrichment_plot . . . 4
create_gsea_plot . . . 6
create_heatmap_plot . . . 7
create_impact_plot . . . 9
create_interaction_plot . . . 10
create_membership_plot . . . 11
create_network_plot . . . 13
da_results_from_se . . . 14
enrichmet . . . 16
extract_kegg_ids . . . 20
extract_measured_kegg_ids . . . 21
fetch_kegg_chebi . . . 22
fetch_kegg_compound_lookup . . . 22
fetch_kegg_pathway_metabolites . . . 23
fetch_kegg_reactome . . . 23
fetch_lion_lipid_ontology . . . 24
fetch_reactome_reactions . . . 25
get_cached_file . . . 26
perform_enrichment_analysis . . . 26
perform_enrichment_analysis_bg . . . 27
perform_gsea_analysis . . . 30
prepare_gmt_data . . . 31
run_de . . . 33

Index **36**

`calculate_metabolite_centrality`
Calculate Metabolite Centrality

Description

Computes the relative betweenness centrality of metabolites derived from their co membership within metabolic pathways using the igraph R package.

Usage

```
calculate_metabolite_centrality(PathwayVsMetabolites)
```

Arguments

PathwayVsMetabolites
: A data frame containing pathways and their associated metabolites. Must include columns 'Pathway' and 'Metabolites'.

Value

A data frame with metabolite names and their relative betweenness centrality scores.

Examples

```
# Create example pathway data with valid KEGG metabolite IDs
PathwayVsMetabolites <- data.frame(
  Pathway = c("Glycolysis / Gluconeogenesis",
              "Citrate cycle (TCA cycle)",
              "Pentose phosphate pathway"),
  Metabolites = c("C00031,C00022,C00197,C00221,C00631,C01172",
                  "C00022,C00036,C00024,C00158,C00149,C00311",
                  "C00117,C00257,C00121,C00085,C00118")
)

# Calculate centrality
centrality <- calculate_metabolite_centrality(PathwayVsMetabolites)
head(centrality)
```

---

create_centrality_plot

Create Centrality Plot

Description

Generates a visualization of metabolite centrality results (relative betweenness centrality, RBC).

Usage

```
create_centrality_plot(centrality_results, top_n = 20, kegg_lookup = NULL)
```

Arguments

centrality_results
: Data frame from `calculate_metabolite_centrality()`, typically filtered to the query metabolites. Must contain `Metabolite` and `RBC_Metabolite`; may already contain `Display_Name`.

top_n
: Number of top metabolites to display (default = 20).

kegg_lookup
: Optional data frame with columns `kegg_id` and `name` for axis labels.

Details

Bars show relative betweenness centrality from the pathway–metabolite annotation graph only. Fold change and p-values are not used in the underlying centrality scores.

Value

A ggplot object (horizontal bar chart of RBC). Returns NULL if centrality_results has zero rows.

Examples

```
# Always-runnable minimal example (no network)
cen <- data.frame(
    Metabolite = c("C00031", "C00022", "C00074"),
    RBC_Metabolite = c(0.12, 0.08, 0.05),
    stringsAsFactors = FALSE
)
create_centrality_plot(cen, top_n = 3)


PathwayVsMetabolites <- fetch_kegg_pathway_metabolites(organism = "hsa")
kegg_lookup <- fetch_kegg_compound_lookup()

# Metabolomics matrix (bundled example data)
metabolomics_path <- system.file(
    "extdata", "example_data.csv",
    package = "enrichmet"
)

if (metabolomics_path == "") {
    stop("Example file 'example_data.csv' not found in inst/extdata/")
}

metabolomics_mat <- read.csv(
    metabolomics_path,
    row.names = 1,
    check.names = FALSE
)
da_out <- run_de(
    metabolomics_mat, "TK-CMV", "K-CMV",
    fc_threshold = 1, pval_threshold = 0.05
)

query <- da_out$kegg_ready$kegg_id[
    da_out$kegg_ready$Significant != "Not significant"
]
query <- query[!is.na(query) & query != ""]

cen <- calculate_metabolite_centrality(PathwayVsMetabolites)
cen <- cen[cen$Metabolite %in% query, ]

create_centrality_plot(cen, top_n = 15, kegg_lookup = kegg_lookup)
```

`create_enrichment_plot`

Create Pathway Enrichment Plot

Description

Generates a visualization of pathway enrichment results.

Usage

```
create_enrichment_plot(enrichment_results)
```

Arguments

enrichment_results	Data frame with enrichment analysis results. Must contain columns Pathway, Log_P_value, and Count.

Value

A ggplot object showing enrichment results.

Examples

```
# Always-runnable minimal example (no network, no dependencies)
enr <- data.frame(
    Pathway = c("Glycolysis", "TCA cycle", "Pentose phosphate pathway"),
    Log_P_value = c(3.2, 2.1, 1.4),
    Count = c(8, 5, 3),
    stringsAsFactors = FALSE
)
create_enrichment_plot(enr)


PathwayVsMetabolites <- fetch_kegg_pathway_metabolites(organism = "hsa")

# Metabolomics matrix (bundled example data)
metabolomics_path <- system.file(
    "extdata", "example_data.csv",
    package = "enrichmet"
)

if (metabolomics_path == "") {
    stop("Example file 'example_data.csv' not found in inst/extdata/")
}

metabolomics_mat <- read.csv(
    metabolomics_path,
    row.names = 1,
    check.names = FALSE
)
da_out <- run_de(
    metabolomics_mat, "TK-CMV", "K-CMV",
    fc_threshold = 1, pval_threshold = 0.05
)
query <- da_out$kegg_ready$kegg_id[
    da_out$kegg_ready$Significant != "Not significant"
]
query <- query[!is.na(query) & query != ""]

enr_real <- perform_enrichment_analysis(
```

```
    inputMetabolites = query,
    PathwayVsMetabolites = PathwayVsMetabolites,
    top_n = 15,
    p_value_cutoff = 1
)

create_enrichment_plot(enr_real)
```

`create_gsea_plot` *Create GSEA Plot*

Description

Generates a visualization of GSEA results.

Usage

```
create_gsea_plot(gsea_results, top_n = 20, kegg_lookup = NULL)
```

Arguments

`gsea_results`	Data frame from perform_gsea_analysis().
`top_n`	Number of top pathways to display (default = 20).
`kegg_lookup`	Optional data frame for mapping pathway IDs to names.

Value

A ggplot object showing GSEA results.

Examples

```
# Create comprehensive example GSEA results

gsea_results <- data.frame(
  pathway = c("Glycolysis / Gluconeogenesis", "Citrate cycle (TCA cycle)",
              "Pentose phosphate pathway", "Pyruvate metabolism",
              "Amino sugar and nucleotide sugar metabolism",
              "Glycine, serine and threonine metabolism",
              "Cysteine and methionine metabolism",
              "Valine, leucine and isoleucine degradation",
              "Fatty acid degradation", "Oxidative phosphorylation",
              "Purine metabolism", "Pyrimidine metabolism",
              "Alanine, aspartate and glutamate metabolism",
              "Butanoate metabolism", "Propanoate metabolism",
              "Starch and sucrose metabolism", "Galactose metabolism",
              "Fructose and mannose metabolism", "Glycerolipid metabolism",
              "Glycerophospholipid metabolism"),
  pval = c(1.2e-8, 3.4e-6, 2.1e-5, 7.8e-5, 1.5e-4, 3.2e-4, 8.7e-4,
           0.0012, 0.0023, 0.0041, 0.0067, 0.0089, 0.012, 0.018,
           0.025, 0.032, 0.041, 0.055, 0.067, 0.078),
  NES = c(2.3, 2.1, 1.9, 1.8, 1.7, 1.6, 1.5, -1.4, -1.3, 1.2,
```

```
          -1.1, 1.0, 0.9, -0.8, 0.7, 0.6, -0.5, 0.4, 0.3, -0.2),
  input_count = c(15, 12, 10, 9, 8, 7, 7, 6, 6, 5, 8, 7, 6, 5, 4, 7, 5, 6, 4, 3)
)

# Create GSEA plot
plot <- create_gsea_plot(gsea_results, top_n = 15)
plot
```

`create_heatmap_plot` *Create Enrichment Heatmap*

Description

Generates a heatmap showing metabolite-pathway enrichment significance.

Usage

```
create_heatmap_plot(
  enrichment_results,
  PathwayVsMetabolites,
  inputMetabolites,
  kegg_lookup = NULL,
  min_pathways = 2,
  min_metabolites = 1,
  base_font_size = 12,
  top_n = NULL
)
```

Arguments

`enrichment_results`	Data frame from perform_enrichment_analysis().
`PathwayVsMetabolites`	A data frame with pathway-metabolite associations.
`inputMetabolites`	A character vector of metabolite IDs.
`kegg_lookup`	Optional data frame for KEGG ID to name mapping.
`min_pathways`	Minimum number of metabolites a pathway must have to be included.
`min_metabolites`	Minimum number of pathways a metabolite must be in to be included.
`base_font_size`	Base font size for the heatmap labels.
`top_n`	Number of top metabolites to include (by significance).

Value

A ComplexHeatmap object showing enrichment significance.

A ComplexHeatmap object showing enrichment significance.

Examples

```
# Create Enrichment Heatmap

# Generates a heatmap showing metabolite-pathway enrichment significance.


# Create comprehensive example data
enrichment_results <- data.frame(
  Pathway = c("Glycolysis / Gluconeogenesis", "Citrate cycle (TCA cycle)",
              "Pentose phosphate pathway", "Pyruvate metabolism",
              "Amino sugar and nucleotide sugar metabolism",
              "Glycine, serine and threonine metabolism",
              "Cysteine and methionine metabolism"),
  Log_P_value = c(7.2, 5.8, 4.9, 4.3, 3.8, 3.5, 3.2),
  P_value = 10^(-c(7.2, 5.8, 4.9, 4.3, 3.8, 3.5, 3.2)),
  Adjusted_P_value = 10^(-c(6.5, 5.2, 4.4, 3.9, 3.5, 3.2, 2.9))
)

PathwayVsMetabolites <- data.frame(
  Pathway = c("Glycolysis / Gluconeogenesis", "Citrate cycle (TCA cycle)",
              "Pentose phosphate pathway", "Pyruvate metabolism",
              "Amino sugar and nucleotide sugar metabolism",
              "Glycine, serine and threonine metabolism",
              "Cysteine and methionine metabolism"),
  Metabolites = c("C00031,C00022,C00074,C00036,C00103,C00197,C00186,C00221,C01172",
                  "C00024,C00036,C00042,C00122,C00149,C00158,C00417,C05379",
                  "C00117,C00231,C00279,C00345,C01172,C01236,C01218",
                  "C00022,C00024,C00036,C00074,C00122,C00149,C00232",
                  "C00031,C00095,C00103,C00121,C00140,C00208,C00267",
                  "C00037,C00048,C00065,C00143,C00258,C00300,C00497",
                  "C00021,C00073,C00094,C00155,C00283,C00542,C00957")
)

inputMetabolites <- c("C00031", "C00022", "C00074", "C00036", "C00103", "C00197",
                      "C00186", "C00221", "C00024", "C00042", "C00122", "C00149",
                      "C00158", "C00117", "C00231", "C00279", "C00037", "C00048")

kegg_lookup <- data.frame(
  kegg_id = c("C00031", "C00022", "C00074", "C00036", "C00103", "C00197",
              "C00186", "C00221", "C00024", "C00042", "C00122", "C00149",
              "C00158", "C00117", "C00231", "C00279", "C00037", "C00048",
              "C00417", "C05379", "C00345", "C01172", "C01236", "C00065",
              "C00143", "C00258", "C00021", "C00073", "C00094"),
  name = c("D-Glucose", "L-Lactate", "Oxaloacetate", "Citrate",
           "D-Fructose 6-phosphate", "3-Phospho-D-glyceroyl phosphate",
           "L-Aspartate", "Oxoglutaric acid", "Acetyl-CoA",
           "Oxaloacetic acid", "L-Glutamate", "Succinyl-CoA",
           "L-Malate", "D-Ribose 5-phosphate", "D-Xylulose 5-phosphate",
           "Sedoheptulose 7-phosphate", "Glycine", "L-Serine",
           "Succinate", "Isocitrate", "Glyceraldehyde 3-phosphate",
           "D-Fructose 1,6-bisphosphate", "D-Erythrose 4-phosphate",
           "L-Threonine", "L-Cysteine", "O-Acetyl-L-serine",
           "L-Cystathionine", "L-Homocysteine", "S-Adenosyl-L-methionine")
)

# Create enrichment heatmap with all parameters
```

```
heatmap_plot <- create_heatmap_plot(
  enrichment_results = enrichment_results,
  PathwayVsMetabolites = PathwayVsMetabolites,
  inputMetabolites = inputMetabolites,
  kegg_lookup = kegg_lookup,
  top_n = 10,
  min_pathways = 1,
  min_metabolites = 1
)

# Note: ComplexHeatmap objects need to be drawn explicitly
if (!is.null(heatmap_plot)) {
  ComplexHeatmap::draw(heatmap_plot)
}
```

`create_impact_plot` *Create Impact vs Significance Plot*

Description

Generates a visualization of pathway impact versus statistical significance.

Usage

```
create_impact_plot(significant_results_df)
```

Arguments

`significant_results_df`
: Data frame from `perform_enrichment_analysis()` (or `enrichmet()$pathway_enrichment_results`). Must contain columns `Impact`, `Log_P_value`, and `Pathway`.

Details

Each point is a pathway. Impact is a topology-based score; the vertical axis is enrichment significance. Pathways that are both statistically supported and structurally central stand out toward the upper right.

Value

A `ggplot` object showing pathway impact (x) versus $-\log_{10} p$-value (y). Returns `NULL` if `significant_results_df` has zero rows.

Examples

```
PathwayVsMetabolites <- fetch_kegg_pathway_metabolites(organism = "hsa")

metabolomics_path <- system.file(
    "extdata", "example_data.csv",
    package = "enrichmet"
)
```

```
if (metabolomics_path == "") {
    stop("Example file 'example_data.csv' not found in inst/extdata/")
}

metabolomics_mat <- read.csv(
    metabolomics_path,
    row.names = 1,
    check.names = FALSE
)
da_out <- run_de(
    metabolomics_mat, "TK-CMV", "K-CMV",
    fc_threshold = 1, pval_threshold = 0.05
)

enr <- perform_enrichment_analysis(
    inputMetabolites = da_out$kegg_ready$kegg_id[
        da_out$kegg_ready$Significant != "Not significant"
    ],
    PathwayVsMetabolites = PathwayVsMetabolites,
    top_n = 15,
    p_value_cutoff = 1
)

create_impact_plot(enr)
```

`create_interaction_plot` *Create Metabolite Interaction Network (Reactome)*

Description

Generates a metabolite interaction network based on shared Reactome reactions. Two metabolites are connected by an edge if they co-occur in the same Reactome reaction; edge weight reflects the number of reactions they share.

Usage

```
create_interaction_plot(
  inputMetabolites,
  reactome_df,
  kegg_lookup = NULL,
  min_shared_reactions = 1,
  max_metabolites_per_reaction = 25
)
```

Arguments

`inputMetabolites`	A character vector of metabolite IDs (KEGG IDs).
`reactome_df`	A data frame with the KEGG-to-Reactome reaction mapping, e.g. the result of joining a KEGG-ChEBI conversion table to Reactome's ChEBI2ReactomeReactions.txt. Must contain columns 'KEGG' and 'Reaction'. If you want to restrict to a

species (e.g. "Homo sapiens"), filter reactome_df on 'Species' before calling this function.

`kegg_lookup` Optional data frame for KEGG ID to name mapping. Should contain columns 'kegg_id' and 'name'.

`min_shared_reactions`
Minimum number of shared Reactome reactions required for two metabolites to be connected by an edge (default 1).

`max_metabolites_per_reaction`
Reactions involving more than this many of the input metabolites are skipped when building edges. This avoids hairball edges coming from very generic/large reactions that touch many unrelated metabolites (default 25).

Value

A ggraph object showing the Reactome-based metabolite interaction network, or NULL if insufficient data is available.

Examples

```
reactome_df <- data.frame(
  KEGG = c("C00031", "C00031", "C00022", "C00022", "C00074", "C00074"),
  ChEBI = c("4167", "4167", "16651", "16651", "16452", "16452"),
  Reaction = c("R-HSA-1", "R-HSA-2", "R-HSA-1", "R-HSA-3",
               "R-HSA-2", "R-HSA-3"),
  Species = "Homo sapiens"
)
plot <- create_interaction_plot(
  inputMetabolites = c("C00031", "C00022", "C00074"),
  reactome_df = reactome_df
)
plot
```

---

create_membership_plot

Create Pathway Membership Plot

Description

Generates a heatmap showing pathway membership for metabolites.

Usage

```
create_membership_plot(
  PathwayVsMetabolites,
  inputMetabolites,
  kegg_lookup = NULL,
  top_n = NULL,
  min_pathway_occurrence = 1,
  min_metabolite_occurrence = 1
)
```

Arguments

PathwayVsMetabolites	A data frame with pathway-metabolite associations.
inputMetabolites	A character vector of metabolite IDs.
kegg_lookup	Optional data frame for KEGG ID to name mapping.
top_n	Number of top metabolites to display (by centrality).
min_pathway_occurrence	Minimum number of metabolites a pathway must have to be included.
min_metabolite_occurrence	Minimum number of pathways a metabolite must be in to be included.

Value

A ComplexHeatmap object showing pathway membership.

Examples

```
# Create comprehensive example data with matching metabolite IDs
inputMetabolites <- c("C00031", "C00022", "C00074", "C00036", "C00103",
                      "C00197", "C00186", "C00221", "C00024", "C00042",
                      "C00122", "C00149", "C00158", "C00117", "C00231",
                      "C00279", "C00037", "C00048", "C00065", "C00143")

PathwayVsMetabolites <- data.frame(
  Pathway = c("Glycolysis/Gluconeogenesis", "Citrate cycle (TCA cycle)",
              "Pentose phosphate pathway", "Pyruvate metabolism",
              "Amino sugar and nucleotide sugar metabolism",
              "Glycine, serine and threonine metabolism",
              "Cysteine and methionine metabolism",
              "Valine, leucine and isoleucine degradation"),
  Metabolites = c("C00031,C00022,C00074,C00036,C00103,C00197,C00186,C00221",
                  "C00024,C00036,C00042,C00122,C00149,C00158,C00417",
                  "C00117,C00231,C00279,C00345,C01172",
                  "C00022,C00024,C00036,C00074,C00122",
                  "C00031,C00095,C00103,C00121,C00140",
                  "C00037,C00048,C00065,C00143,C00258",
                  "C00021,C00073,C00094,C00155,C00283",
                  "C00123,C00183,C00233,C00407,C00979")
)

# Create KEGG lookup with matching IDs
kegg_lookup <- data.frame(
  kegg_id = c("C00031", "C00022", "C00074", "C00036", "C00103",
              "C00197", "C00186", "C00221", "C00024", "C00042",
              "C00122", "C00149", "C00158", "C00117", "C00231",
              "C00279", "C00037", "C00048", "C00065", "C00143",
              "C00258", "C00021", "C00073", "C00094", "C00155"),
  name = c("D-Glucose", "L-Lactate", "Oxaloacetate", "Citrate",
           "D-Fructose 6-phosphate", "3-Phospho-D-glyceroyl phosphate",
           "L-Aspartate", "Oxoglutaric acid", "Acetyl-CoA",
           "Oxaloacetic acid", "L-Glutamate", "Succinyl-CoA",
           "L-Malate", "D-Ribose 5-phosphate", "D-Xylulose 5-phosphate",
           "Sedoheptulose 7-phosphate", "Glycine", "L-Serine",
```

```
            "L-Threonine", "L-Cysteine", "O-Acetyl-L-serine",
            "L-Cystathionine", "L-Homocysteine", "S-Adenosyl-L-methionine", "Coenzyme A")
)

# Create membership plot
plot <- create_membership_plot(PathwayVsMetabolites, inputMetabolites, kegg_lookup, top_n = 10)
# Note: ComplexHeatmap objects need to be drawn explicitly
if (!is.null(plot)) {
  ComplexHeatmap::draw(plot)
}
```

`create_network_plot` *Create Metabolite Pathway Network*

Description

Generates a metabolite pathway interaction network visualization.

Usage

```
create_network_plot(
  inputMetabolites,
  PathwayVsMetabolites,
  kegg_lookup = NULL,
  top_n = 20,
  font_family = "sans"
)
```

Arguments

`inputMetabolites`	Character vector of metabolite IDs
`PathwayVsMetabolites`	Data frame with pathway metabolite associations
`kegg_lookup`	Optional data frame for KEGG ID to name mapping
`top_n`	Number of top metabolites to display based on centrality
`font_family`	Font family for text elements

Value

A ggraph object showing the metabolite pathway network

Examples

```
# Always-runnable minimal example (no network)
pw <- data.frame(
    Pathway = c("Glycolysis", "TCA cycle"),
    Metabolites = c("C00031,C00022,C00074", "C00036,C00042,C00026"),
    stringsAsFactors = FALSE
)
mets <- c("C00031", "C00022", "C00036")
```

```
create_network_plot(mets, pw, kegg_lookup = NULL, top_n = 2)


PathwayVsMetabolites <- fetch_kegg_pathway_metabolites(organism = "hsa")
kegg_lookup <- fetch_kegg_compound_lookup()

# Metabolomics matrix (bundled example data)
metabolomics_path <- system.file(
    "extdata", "example_data.csv",
    package = "enrichmet"
)

if (metabolomics_path == "") {
    stop("Example file 'example_data.csv' not found in inst/extdata/")
}

metabolomics_mat <- read.csv(
    metabolomics_path,
    row.names = 1,
    check.names = FALSE
)
da_out <- run_de(
    metabolomics_mat, "TK-CMV", "K-CMV",
    fc_threshold = 1, pval_threshold = 0.05
)
query <- da_out$kegg_ready$kegg_id[
    da_out$kegg_ready$Significant != "Not significant"
]
query <- query[!is.na(query) & query != ""]

create_network_plot(
    query,
    PathwayVsMetabolites,
    kegg_lookup = kegg_lookup,
    top_n = 10
)
```

`da_results_from_se` *Convert a SummarizedExperiment to a da_results Object*

Description

Reconstructs the `da_results` structure expected by `enrichmet()` from a `SummarizedExperiment`. Metabolite identifiers, fold changes, and significance statistics are read from `rowData()`, so no external `da_out` object is required.

Usage

```
da_results_from_se(
  se,
  fc_threshold = 1,
  pval_threshold = 0.05,
```

```
  split_complex_ids = TRUE
)
```

Arguments

se	A SummarizedExperiment (or an object that inherits from it) whose rowData() contains at least the columns met_id, log2fc, pval, and padj.
fc_threshold	Numeric. Absolute log2 fold change threshold used to label metabolites as "Up" or "Down" when the Significant column is not already present in rowData(). Default is 1.
pval_threshold	Numeric. Adjusted p value threshold used together with fc_threshold to assign significance labels when Significant is absent. Default is 0.05.
split_complex_ids	Logical. If TRUE, metabolite identifiers that embed multiple KEGG IDs (for example neg_00096_C00025|C00979) are split into one row per KEGG ID in the reconstructed kegg_ready table. If FALSE, only the first KEGG ID per metabolite is retained. Default is TRUE.

Details

da_results_from_se() performs the following steps:

1. Extracts rowData(se) as a data frame and verifies that met_id, log2fc, pval, and padj are present.
2. Recomputes the Significant column if it is absent, using the supplied fc_threshold and pval_threshold, matching the conventions used by run_de().
3. Rebuilds the kegg_ready table by extracting KEGG IDs from met_id.
4. Rebuilds the summary_stats list from the assay dimensions and the significance labels.

The helper does not require the original da_out object. Once the differential statistics live in rowData(), the SummarizedExperiment is self contained for the purposes of enrichmet().

Value

A named list with three components:

full_results Data frame with columns met_id, log2fc, pval, padj, optionally ave_expr, and Significant.

kegg_ready Data frame with the same columns as full_results plus a kegg_id column, one row per KEGG ID when split_complex_ids = TRUE.

summary_stats List of summary counts and thresholds, mirroring the output of run_de().

See Also

run_de, enrichmet

Examples

```
library(SummarizedExperiment)

# Example abundance matrix
set.seed(123)
mat <- matrix(
```

```
    rnorm(100, mean = 10, sd = 2),
    nrow = 10,
    dimnames = list(
        paste0("Met_", 1:10),
        c(paste0("Control_", 1:5), paste0("Treat_", 1:5))
    )
)

# Differential statistics stored in rowData()
rd <- DataFrame(
    met_id   = rownames(mat),
    log2fc   = rnorm(10),
    pval     = runif(10, 0, 0.1),
    padj     = runif(10, 0, 0.2),
    ave_expr = rowMeans(mat)
)

se <- SummarizedExperiment(
    assays  = list(counts = mat),
    rowData = rd
)

# Rebuild da_results from the SummarizedExperiment alone
da_out_from_se <- da_results_from_se(se)
str(da_out_from_se, max.level = 1)
```

`enrichmet` *ENRICHMET: Comprehensive Pathway Analysis for Metabolomics Data*

Description

A modular tool for metabolite pathway enrichment analysis that can perform multiple analytical tasks including pathway enrichment, metabolite set enrichment analysis (GSEA), network centrality analysis, and interaction network visualization. The function can be run as a complete workflow or using individual analysis steps.

Usage

```
enrichmet(
  inputMetabolites = NULL,
  PathwayVsMetabolites,
  example_data = NULL,
  da_results = NULL,
  top_n = 100,
  p_value_cutoff = 1,
  kegg_lookup = NULL,
  reactome_df = NULL,
  output_dir = NULL,
  save_excel = FALSE,
  analysis_type = c("enrichment", "gsea", "centrality", "network", "heatmap",
    "membership", "interaction"),
```

```
  run_plots = TRUE,
  network_top_n = 20,
  heatmap_top_n = 20,
  membership_top_n = 20,
  min_pathway_occurrence = 1,
  min_metabolite_occurrence = 1,
  use_significant_only = TRUE,
  split_complex_ids = TRUE,
  significance_threshold = "both",
  include_volcano = TRUE,
  fc_cutoff_up = 1,
  fc_cutoff_down = -1,
  fdr_cutoff_da = 0.05,
  force_custom_filters = FALSE,
  backgroundMetabolites = NULL,
  input_type = c("kegg", "name", "mixed")
)
```

Arguments

inputMetabolites	A character vector of metabolite identifiers and/or names, or a data frame from `run_de()$kegg_ready`. For `input_type = "kegg"` (default), values should be KEGG IDs (e.g. `"C00031"`); complex forms like `"C00042|C02170"` are supported when `split_complex_ids = TRUE`. For `input_type = "name"` or `"mixed"`, common names (e.g. `"glucose"`) are mapped to KEGG IDs via `kegg_lookup`. Required if `da_results` is not provided.
PathwayVsMetabolites	A data frame containing pathways and their associated metabolites. Must include columns `Pathway` and `Metabolites`.
example_data	A data frame containing metabolite-level data for GSEA analysis. Should include columns `met_id`, `pval`, and `log2fc`. Required for GSEA.
da_results	Optional output from `run_de()`. If provided, metabolites are taken from `da_results$kegg_ready`. Mutually exclusive with prioritising `inputMetabolites` when both are supplied (da_results wins).
top_n	An integer specifying the number of top pathways to include in the pathway enrichment results (default is 100). Use `NULL` to return all pathways.
p_value_cutoff	A numeric value for the adjusted p-value threshold for filtering significant pathways (default is 1, no filtering).
kegg_lookup	Optional data frame for KEGG ID name mapping. Must contain columns `kegg_id` and `name`. **Required** when `input_type` is `"name"` or `"mixed"` (e.g. `fetch_kegg_compound_look`
reactome_df	Optional data frame containing the KEGG-to-Reactome reaction mapping. Must contain columns `KEGG` and `Reaction`.
output_dir	Optional directory path for saving output files. If `NULL` (default), no files are written.
save_excel	Logical; whether to save results as Excel files (default = FALSE).
analysis_type	Character vector specifying which analyses to run. Options: `"enrichment"`, `"gsea"`, `"centrality"`, `"network"`, `"interaction"`, `"heatmap"`, `"membership"`. Default: all.
run_plots	Logical; whether to generate visualization plots (default = TRUE).

`network_top_n` Number of top pathways in the network plot (default = 20).

`heatmap_top_n` Number of top pathways in the heatmap (default = 20).

`membership_top_n`
: Number of top pathways in the membership matrix (default = 20).

`min_pathway_occurrence`
: Minimum pathway occurrence for heatmap/membership (default = 1).

`min_metabolite_occurrence`
: Minimum metabolite occurrence for heatmap/membership (default = 1).

`use_significant_only`
: Logical; use only significant metabolites from `da_results` (default = TRUE). Only applies when `da_results` is provided.

`split_complex_ids`
: Logical; split complex KEGG IDs like `"C00042|C02170"` into individual IDs (default = TRUE).

`significance_threshold`
: When using `da_results`: `"up"`, `"down"`, or `"both"` (default: `"both"`).

`include_volcano`
: Logical; include volcano plot from `da_results` (default = TRUE).

`fc_cutoff_up` Fold-change cutoff for upregulated metabolites (default = 1).

`fc_cutoff_down` Fold-change cutoff for downregulated metabolites (default = -1).

`fdr_cutoff_da` FDR cutoff for DA significance (default = 0.05).

`force_custom_filters`
: Logical; force custom FC/FDR even if a `Significant` column exists (default = FALSE).

`backgroundMetabolites`
: Optional character vector of KEGG IDs defining the statistical background (universe) for background-corrected enrichment. If `NULL`, the full pathway database is used.

`input_type` Character; how to interpret `inputMetabolites`:

- `"kegg"` (default) treat entries as KEGG IDs (`C#####`).
- `"name"` treat entries as metabolite names and map them to KEGG IDs via `kegg_lookup`.
- `"mixed"` keep valid KEGG IDs and map the rest as names.

Details

Input can be KEGG compound IDs, metabolite names, or a mix of both (controlled by `input_type`). Name mapping uses `kegg_lookup` (e.g. from `fetch_kegg_compound_lookup`)

This function performs comprehensive pathway analysis for metabolomics data. Users can run the complete workflow or call individual modular functions for specific analyses. See the individual function documentation for more details.

When `input_type` is `"name"` or `"mixed"`, names are matched case-insensitively against `kegg_lookup$name` (exact, then starts-with, then contains). Unmapped entries are dropped with a warning. All downstream analyses (including the Reactome interaction network) receive the resulting clean KEGG ID vector.

Value

A list containing results from the specified analyses. Possible components:

- `input_metabolites_used` Character vector of KEGG IDs used for analysis
- `pathway_enrichment_all` Data frame of all pathway enrichment results
- `pathway_enrichment_results` Data frame of filtered pathway enrichment results
- `gsea_results` Data frame of GSEA analysis results
- `metabolite_centrality` Data frame of centrality analysis results
- `volcano_plot` ggplot object (only when `da_results` provided)
- `pathway_plot` ggplot object for pathway enrichment
- `impact_plot` ggplot object for impact vs significance
- `gsea_plot` ggplot object for GSEA results
- `rbc_plot` ggplot object for relative betweenness centrality
- `network_plot` ggraph object for metabolite pathway network
- `heatmap_plot` ComplexHeatmap object for enrichment significance
- `membership_plot` ComplexHeatmap object for pathway membership
- `interaction_plot` ggraph object for Reactome interaction network

See Also

Individual analysis functions: `prepare_gmt_data` for GMT format conversion, `calculate_metabolite_centrality` for network centrality, `perform_enrichment_analysis` for pathway enrichment, `perform_gsea_analysis` for metabolite set enrichment, `create_enrichment_plot` for pathway visualization, `create_impact_plot` for impact vs significance, `create_gsea_plot` for GSEA results visualization, `create_centrality_plot` for centrality visualization, `fetch_kegg_compound_lookup` for building `kegg_lookup`

Examples

```
# Pathway map and compound names (KEGG REST API)
PathwayVsMetabolites <- fetch_kegg_pathway_metabolites(organism = "hsa")
kegg_lookup <- fetch_kegg_compound_lookup()

# Metabolomics matrix (bundled example data)
metabolomics_path <- system.file(
    "extdata", "example_data.csv",
    package = "enrichmet"
)

if (metabolomics_path == "") {
    stop("Example file 'example_data.csv' not found in inst/extdata/")
}

metabolomics_mat <- read.csv(
    metabolomics_path,
    row.names = 1,
    check.names = FALSE
)

da_out <- run_de(
    metabolomics_mat, "TK-CMV", "K-CMV",
```

```
    fc_threshold = 1, pval_threshold = 0.05
)

example_path <- system.file(
    "extdata", "summary_stat.csv",
    package = "enrichmet"
)

if (example_path == "") {
    stop("Example file 'summary_stat.csv' not found in inst/extdata/")
}

example_data <- read.csv(example_path, stringsAsFactors = FALSE)


# Enrichment from DA results (query set taken from da_out)
results <- enrichmet(
    inputMetabolites = NULL,
    PathwayVsMetabolites = PathwayVsMetabolites,
    da_results = da_out,
    example_data = example_data,
    kegg_lookup = kegg_lookup,
    analysis_type = c("enrichment", "gsea", "centrality"),
    top_n = 10,
    p_value_cutoff = 1
)

head(results$pathway_enrichment_results, 3)
```

`extract_kegg_ids` *Extract KEGG compound IDs from pathway data*

Description

Extracts unique KEGG compound IDs from the `Metabolites` column of a KEGG pathway-metabolite table.

Usage

```
extract_kegg_ids(PathwayVsMetabolites)
```

Arguments

`PathwayVsMetabolites`
: Data frame containing a `Metabolites` column with comma-separated KEGG compound IDs.

Value

A character vector containing unique KEGG compound IDs.

Examples

```
PathwayVsMetabolites <- fetch_kegg_pathway_metabolites(organism = "hsa")
kegg_ids <- extract_kegg_ids(PathwayVsMetabolites)
head(kegg_ids)
```

`extract_measured_kegg_ids` *Extract measured KEGG metabolite IDs from a summary statistics table*

Description

Turns a per-feature summary-statistics table (typical untargeted LC-MS differential analysis output) into a vector of KEGG compound IDs. This is the set of metabolites that were measurable/testable on the platform and is the appropriate universe for background-corrected enrichment, as opposed to every compound annotated in the pathway database.

Usage

```
extract_measured_kegg_ids(summary_stats)
```

Arguments

`summary_stats`	A data frame with a `met_id` column whose values embed KEGG IDs (pattern `"C"` + 5 digits), e.g. `"neg_00021_C00245"`, or ambiguous annotations such as `"neg_00096_C00025|C00979"` (both candidate IDs are kept).

Value

A character vector of unique KEGG IDs found in `summary_stats`.

See Also

`perform_enrichment_analysis_bg`

Examples

```
# Summary statistics (local example file)
example_path <- system.file(
    "extdata", "summary_stat.csv",
    package = "enrichmet"
)

if (example_path == "") {
    stop("Example file 'summary_stat.csv' not found in inst/extdata/")
}

example_data <- read.csv(example_path, stringsAsFactors = FALSE)
measured <- extract_measured_kegg_ids(example_data)
length(measured)
head(measured)
```

fetch_kegg_chebi *Fetch KEGG-ChEBI mappings*

Description

Downloads the KEGG-to-ChEBI conversion table and filters it to the supplied KEGG compound IDs.

Usage

```
fetch_kegg_chebi(kegg_ids)
```

Arguments

kegg_ids	Character vector of KEGG compound IDs.

Value

A data frame with columns KEGG and ChEBI.

Examples

```
kegg_chebi <- fetch_kegg_chebi(c("C00022", "C00031"))
head(kegg_chebi)
```

fetch_kegg_compound_lookup *Fetch KEGG compound names*

Description

Downloads the KEGG compound list and returns KEGG compound IDs with their corresponding names.

Usage

```
fetch_kegg_compound_lookup(clean_names = TRUE)
```

Arguments

clean_names	Logical. If TRUE, only the first compound name is retained when multiple names are provided.

Value

A data frame with columns kegg_id and name.

kegg_lookup <- fetch_kegg_compound_lookup() head(kegg_lookup)

fetch_kegg_pathway_metabolites *Fetch KEGG pathway-metabolite relationships*

Description

Downloads KEGG pathway and compound relationship data and returns pathway-level metabolite mappings via the public KEGG REST API.

Usage

```
fetch_kegg_pathway_metabolites(
  organism = "hsa",
  clean_pathway_names = TRUE,
  metaboanalyst_style = TRUE
)
```

Arguments

organism	Character. KEGG organism code. Default is "hsa".
clean_pathway_names	Logical. If TRUE, KEGG organism information is removed from pathway names.
metaboanalyst_style	Logical. If TRUE (default), apply a MetaboAnalyst-style filter that keeps metabolic pathway maps (numeric IDs < 2000) and excludes Global (011xx) and Overview (012xx) maps. If FALSE, all pathways returned for the organism are retained.

Value

A data frame with columns PathwayID, Pathway, and Metabolites.

Examples

```
PathwayVsMetabolites <- fetch_kegg_pathway_metabolites(organism = "hsa")
head(PathwayVsMetabolites)
```

fetch_kegg_reactome *Build KEGG-Reactome reaction mappings*

Description

Links KEGG compounds to Reactome reactions through ChEBI identifiers.

Usage

```
fetch_kegg_reactome(kegg_ids, species = "Homo sapiens")
```

Arguments

kegg_ids	Character vector of KEGG compound IDs.
species	Character. Reactome species name.

Value

A data frame containing KEGG-to-ChEBI-to-Reactome mappings.

Examples

```
reactome_df <- fetch_kegg_reactome(
    kegg_ids = c("C00022", "C00031"),
    species = "Homo sapiens"
)
head(reactome_df)
```

---

fetch_lion_lipid_ontology

Fetch LION lipid ontology

Description

Downloads the LION ontology from BioPortal and converts the OBO file into a pathway-metabolite table suitable for enrichment analysis. Requires a BioPortal API key; no LION data are redistributed with the package.

Usage

```
fetch_lion_lipid_ontology(api_key = NULL, submission = 1)
```

Arguments

api_key	BioPortal API key. If NULL, the function uses the BIOPORTAL_API_KEY environment variable.
submission	LION submission number. Default is 1.

Value

A data.frame with columns Pathway, Metabolites, and Count.

Examples

```
# Expected output structure (no API call)
pathway_lipids <- data.frame(
    Pathway = c("Glycerophospholipids", "Sphingolipids"),
    Metabolites = c("PC(16:0/18:1),PE(18:0/18:1)", "SM(d18:1/16:0)"),
    Count = c(2L, 1L),
    stringsAsFactors = FALSE
)
head(pathway_lipids)
```

```
# Live download when a BioPortal key is available
if (nzchar(Sys.getenv("BIOPORTAL_API_KEY"))) {
    pathway_lipids <- fetch_lion_lipid_ontology()
    head(pathway_lipids)
} else {
    message(
        "Set Sys.setenv(BIOPORTAL_API_KEY = \"your_key\") ",
        "to run the live LION download example."
    )
}
```

`fetch_reactome_reactions` *Fetch Reactome reactions*

Description

Downloads the Reactome ChEBI-to-reaction mapping and filters it to the requested species. Data are retrieved at runtime and cached; no Reactome dump is redistributed with the package.

Usage

```
fetch_reactome_reactions(species = "Homo sapiens")
```

Arguments

`species`	Character. Species name used to filter Reactome data.

Value

A data frame containing ChEBI, Reactome, reaction, evidence, and species information.

Examples

```
reactome <- fetch_reactome_reactions(species = "Homo sapiens")
head(reactome)
nrow(reactome)
```

get_cached_file *Download and cache a remote file*

Description

Downloads a remote resource and stores it in the EnrichMet cache. Existing cached resources are reused automatically. Failed downloads are retried, and a plain download is used if the cache fails.

Usage

```
get_cached_file(url, retries = 3L)
```

Arguments

url	Character. URL of the remote file.
retries	Integer. Number of download attempts. Default is 3.

Value

Character path to the cached file.

Examples

```
if (curl::has_internet()) {
    f <- get_cached_file("https://rest.kegg.jp/list/pathway/hsa")
    head(readLines(f))
}
```

perform_enrichment_analysis *Perform Pathway Enrichment Analysis*

Description

Uses Fisher's exact test to identify enriched pathways for a set of metabolites.

Usage

```
perform_enrichment_analysis(
  inputMetabolites,
  PathwayVsMetabolites,
  top_n = 100,
  p_value_cutoff = 1
)
```

Arguments

inputMetabolites	A character vector of metabolite IDs for which pathway enrichment analysis is to be performed.
PathwayVsMetabolites	A data frame containing pathways and their associated metabolites. Must include columns 'Pathway' and 'Metabolites'.
top_n	An integer specifying the number of top significant pathways to return (default is 100).
p_value_cutoff	A numeric value for adjusting the p-value threshold for filtering significant pathways (default is 1, no filtering).

Value

A data frame of pathway enrichment results including p-values, impact scores, and coverage statistics.

Examples

```
# Create meaningful pathway data with real KEGG IDs
PathwayVsMetabolites <- data.frame(
  Pathway = c("Glycolysis / Gluconeogenesis",
              "Citrate cycle (TCA cycle)",
              "Pentose phosphate pathway",
              "Amino sugar and nucleotide sugar metabolism",
              "Pyruvate metabolism"),
  Metabolites = c("C00031,C00022,C00197,C00221,C00631,C01172",
                  "C00022,C00036,C00024,C00158,C00149,C00311",
                  "C00117,C00257,C00121,C00085,C00118",
                  "C00031,C00095,C00103,C00185",
                  "C00022,C00024,C00036,C00158")
)

# Example: Input metabolites enriched in Glycolysis
inputMetabolites <- c("C00031", "C00022", "C00197", "C00221", "C00631", "C01172")

# Perform enrichment analysis
results <- perform_enrichment_analysis(
  inputMetabolites = inputMetabolites,
  PathwayVsMetabolites = PathwayVsMetabolites,
  p_value_cutoff = 0.05
)
head(results)
```

---

perform_enrichment_analysis_bg

Perform pathway enrichment with optional background correction

---

Description

Fisher's exact test for pathway over-representation, optionally restricted to a measured-metabolite universe (background correction).

Usage

```
perform_enrichment_analysis_bg(
  inputMetabolites,
  PathwayVsMetabolites,
  top_n = 100,
  p_value_cutoff = 1,
  backgroundMetabolites = NULL
)
```

Arguments

`inputMetabolites`	Character vector of query metabolite IDs (e.g. KEGG).
`PathwayVsMetabolites`	Data frame with columns `Pathway` and `Metabolites` (comma-separated IDs).
`top_n`	Integer; number of top pathways to return (default 100).
`p_value_cutoff`	Numeric; filter on raw p-value (default 1).
`backgroundMetabolites`	Optional character vector defining the statistical universe. If `NULL`, all metabolites in `PathwayVsMetabolites` are used. If supplied (e.g. from `extract_measured_kegg_ids`), the universe and each pathway's membership are intersected with this set before testing.

Details

When `backgroundMetabolites` is provided, enrichment is tested only among metabolites that were measured, which avoids inflating the background with database compounds that were never observed.

Value

A data frame of enrichment results (p-values, impact, coverage, etc.).

See Also

`extract_measured_kegg_ids`, `fetch_kegg_pathway_metabolites`

Examples

```
# Always-runnable minimal example (no network)
pw <- data.frame(
    Pathway = c("Glycolysis", "TCA cycle"),
    Metabolites = c("C00031,C00022,C00074", "C00036,C00042,C00026"),
    stringsAsFactors = FALSE
)
query <- c("C00031", "C00022", "C00036")
enr <- perform_enrichment_analysis_bg(
    inputMetabolites = query,
    PathwayVsMetabolites = pw,
    top_n = 5,
    p_value_cutoff = 1
)
head(enr)
```

```
PathwayVsMetabolites <- fetch_kegg_pathway_metabolites(organism = "hsa")

example_path <- system.file(
    "extdata", "summary_stat.csv",
    package = "enrichmet"
)

if (example_path == "") {
    stop("Example file 'summary_stat.csv' not found in inst/extdata/")
}

example_data <- read.csv(example_path, stringsAsFactors = FALSE)
measured <- extract_measured_kegg_ids(example_data)

metabolomics_path <- system.file(
    "extdata", "example_data.csv",
    package = "enrichmet"
)

if (metabolomics_path == "") {
    stop("Example file 'example_data.csv' not found in inst/extdata/")
}

metabolomics_mat <- read.csv(
    metabolomics_path,
    row.names = 1,
    check.names = FALSE
)

da_out <- run_de(
    metabolomics_mat, "TK-CMV", "K-CMV",
    fc_threshold = 1, pval_threshold = 0.05
)
query <- da_out$kegg_ready$kegg_id[
    da_out$kegg_ready$Significant != "Not significant"
]
query <- query[!is.na(query) & query != ""]

enr <- perform_enrichment_analysis_bg(
    inputMetabolites = query,
    PathwayVsMetabolites = PathwayVsMetabolites,
    top_n = 10,
    p_value_cutoff = 1
)
head(enr[, c("Pathway", "P_value", "Adjusted_P_value")], 3)

enr_bg <- perform_enrichment_analysis_bg(
    inputMetabolites = query,
    PathwayVsMetabolites = PathwayVsMetabolites,
    top_n = 10,
    p_value_cutoff = 1,
    backgroundMetabolites = measured
)
head(enr_bg[, c("Pathway", "P_value", "Adjusted_P_value")], 3)
```

`perform_gsea_analysis` *Perform GSEA-style Pathway Enrichment Analysis*

Description

This function performs Gene Set Enrichment Analysis (GSEA) adapted for metabolomics data. It tests whether predefined metabolite sets (pathways) show statistically significant enrichment at the top or bottom of a ranked list of metabolites. Handles complex metabolite IDs by extracting clean KEGG IDs for pathway matching.

Usage

```
perform_gsea_analysis(
  example_data,
  PathwayVsMetabolites,
  minSize = 5,
  maxSize = 500,
  ranking_method = "signed_pval",
  id_col = NULL,
  pval_col = "pval",
  padj_col = "padj",
  log2fc_col = "log2fc",
  nperm = 1000,
  seed = NULL,
  kegg_lookup = NULL
)
```

Arguments

`example_data`	A data frame containing metabolite statistics. Should contain: - Either 'kegg_id' or 'met_id' column for metabolite identifiers - 'log2fc' and 'pval' columns for ranking
`PathwayVsMetabolites`	A data frame with pathway information containing 'Pathway' and 'Metabolites' columns.
`minSize`	Minimum number of metabolites in a pathway to test (default = 5).
`maxSize`	Maximum number of metabolites in a pathway to test (default = 500).
`ranking_method`	Method for ranking metabolites. Options: "signed_pval" (default), "absolute_fc", or "pval_only".
`id_col`	Column name in example_data containing metabolite IDs. Default is NULL (auto-detects 'kegg_id' or 'met_id').
`pval_col`	Column name for p-values. Default is "pval".
`padj_col`	Column name for adjusted p-values. Default is "padj".
`log2fc_col`	Column name for log2 fold changes. Default is "log2fc".
`nperm`	Number of permutations for GSEA. Default is 1000.
`seed`	Random seed for reproducibility. Default is NULL.
`kegg_lookup`	Optional data frame for KEGG ID to name mapping.

Value

A data frame containing the GSEA enrichment results with columns:

- pathway - Pathway name
- pval - Nominal p-value
- padj - Adjusted p-value (FDR)
- ES - Enrichment score
- NES - Normalized enrichment score
- nMoreExtreme - Number of permutations more extreme
- size - Size of the pathway after filtering
- leadingEdge - Metabolites driving the enrichment
- input_count - Number of metabolites from input data in pathway
- significance - Significance stars (*** p<0.001, ** p<0.01, * p<0.05, NS)

Examples

```
# Example 1: Basic GSEA with simulated data
example_data <- data.frame(
  kegg_id = paste0("C", sprintf("%05d", 1:100)),
  log2fc = rnorm(100, mean = 0, sd = 1),
  pval = runif(100, 0.001, 0.1)
)

# Create pathway data
PathwayVsMetabolites <- data.frame(
  Pathway = c("Pathway_A", "Pathway_B", "Pathway_C"),
  Metabolites = c("C00001,C00002,C00003,C00004,C00005,C00006,C00007",
                  "C00008,C00009,C00010,C00011,C00012,C00013,C00014",
                  "C00015,C00016,C00017,C00018,C00019,C00020,C00021")
)

# Run GSEA
gsea_results <- perform_gsea_analysis(example_data, PathwayVsMetabolites)
head(gsea_results)
```

---

prepare_gmt_data *Prepare GMT Data for Pathway Analysis*

Description

Converts pathway-metabolite data to GMT format required for GSEA analysis. Handles various metabolite ID formats and provides detailed filtering information.

Usage

```
prepare_gmt_data(
  PathwayVsMetabolites,
  metabolites_in_data = NULL,
  min_pathway_size = 5
)
```

Arguments

PathwayVsMetabolites
: A data frame containing pathways and their associated metabolites. Must include columns 'Pathway' and 'Metabolites'.

metabolites_in_data
: Optional character vector of metabolite IDs to filter pathways.

min_pathway_size
: Minimum number of metabolites required in a pathway (default = 5).

Value

A list of pathways with their associated metabolites in GMT format.

Examples

```
# Example 1: Basic usage with KEGG metabolites
PathwayVsMetabolites <- data.frame(
  Pathway = c("Glycolysis", "TCA Cycle", "Pentose Phosphate"),
  Metabolites = c(
    "C00031,C00022,C00036,C00103,C00111,C00118,C00186,C00197,C00221,C00236",
    "C00024,C00026,C00042,C00122,C00149,C00158,C00311,C00417,C05125",
    "C00117,C00118,C00121,C00231,C00279,C00345,C00620,C01151,C01236"
  )
)

gmt_data <- prepare_gmt_data(PathwayVsMetabolites)
str(gmt_data)

# Example 2: Filtering with specific KEGG IDs
my_metabolites <- c(
  "C00031", "C00022", "C00036", "C00103", "C00024", "C00026"
)
filtered_gmt <- prepare_gmt_data(PathwayVsMetabolites, my_metabolites)
str(filtered_gmt)

# Example 3: Complex IDs with multiple separators
complex_pathways <- data.frame(
  Pathway = c("PathwayA", "PathwayB"),
  Metabolites = c(
    "C00031|C00022,C00036; C00103",
    "C00024|C00026; C00042,C00149"
  )
)
complex_gmt <- prepare_gmt_data(complex_pathways)
str(complex_gmt)


# Example 4: Realistic human metabolic pathways
human_pathways <- data.frame(
  Pathway = c(
    "Glycolysis / Gluconeogenesis",
    "Citrate cycle (TCA cycle)",
    "Pentose phosphate pathway",
    "Fatty acid metabolism"
  ),
  Metabolites = c(
```

```
    paste0(
      "C00031,C00022,C00036,C00103,C00111,C00118,C00186,C00197,",
      "C00221,C00236,C00267,C00354,C00631,C00668,C01172"
    ),
    "C00024,C00026,C00036,C00042,C00122,C00149,C00158,C00311,C00417,C05125",
    paste0(
      "C00117,C00118,C00121,C00231,C00279,C00345,C00620,C01151,",
      "C01236,C01801,C05378,C05382"
    ),
    "C00083,C00162,C00233,C00487,C00558,C01205,C02571,C05263,C06104,C16255"
  )
)
human_gmt <- prepare_gmt_data(human_pathways)
str(human_gmt)

# Example 5: Adjusting minimum pathway size
# For smaller datasets, you might want to reduce the minimum size
small_gmt <- prepare_gmt_data(
  PathwayVsMetabolites,
  min_pathway_size = 3
)
str(small_gmt)
```

---

run_de *Run Differential Expression Analysis using LIMMA*

Description

Performs differential expression analysis between two sample groups using LIMMA, generates volcano plots, and prepares results for downstream enrichment analysis.

Usage

```
run_de(
  df,
  group1_pattern,
  group2_pattern,
  plot_volcano = TRUE,
  top_n_labels = 20,
  fc_threshold = 1,
  pval_threshold = 0.05,
  min_samples = 3,
  split_complex_ids = FALSE
)
```

Arguments

df	A data frame or matrix with metabolites as rows and samples as columns. Rownames should be metabolite identifiers.
group1_pattern	Character string pattern to identify samples in group 1
group2_pattern	Character string pattern to identify samples in group 2

plot_volcano	Logical indicating whether to generate volcano plot (default: TRUE)
top_n_labels	Number of top features to label in volcano plot (default: 20)
fc_threshold	Fold change threshold for significance (default: 1)
pval_threshold	P-value threshold for significance (default = 0.05)
min_samples	Minimum number of samples required per group (default = 3)
split_complex_ids	Logical indicating whether to split complex KEGG IDs like "C00025|C00979" into individual metabolites (default = TRUE)

Value

A list containing:

full_results	Full results with all metabolites
kegg_ready	Results filtered for KEGG metabolites only
volcano_plot	ggplot object of volcano plot (if plot_volcano = TRUE)
summary_stats	Summary statistics of the analysis

Examples

```
# Example showing clear differential expression patterns
set.seed(123)

# Create data with 30 metabolites and 20 samples
n_metabolites <- 30
n_samples <- 20
simulated_data <- matrix(0, nrow = n_metabolites, ncol = n_samples)

# Control samples (columns 1-10)
simulated_data[, 1:10] <- rnorm(n_metabolites * 10, mean = 10, sd = 2)

# Treatment samples (columns 11-20) with clear patterns
simulated_data[1:5, 11:20] <- rnorm(5 * 10, mean = 15, sd = 2)   # Upregulated
simulated_data[6:10, 11:20] <- rnorm(5 * 10, mean = 5, sd = 2)    # Downregulated
simulated_data[11:30, 11:20] <- rnorm(20 * 10, mean = 10, sd = 2) # Not significant

# Set row and column names
rownames(simulated_data) <- paste0("Met_", 1:30)
colnames(simulated_data) <- c(paste0("Control_", 1:10), paste0("Treatment_", 1:10))

# Run differential expression analysis
results <- run_de(
  df = simulated_data,
  group1_pattern = "Control",
  group2_pattern = "Treatment",
  plot_volcano = TRUE,
  fc_threshold = 1,
  pval_threshold = 0.05
)

# View significant metabolites
sig_metabolites <- results$full_results[results$full_results$Significant != "Not significant", ]
head(sig_metabolites)
```

```
# View summary statistics
results$summary_stats
```

Index

calculate_metabolite_centrality, 2, *19*
create_centrality_plot, 3, *19*
create_enrichment_plot, 4, *19*
create_gsea_plot, 6, *19*
create_heatmap_plot, 7
create_impact_plot, 9, *19*
create_interaction_plot, 10
create_membership_plot, 11
create_network_plot, 13

da_results_from_se, 14

enrichmet, *15*, 16
extract_kegg_ids, 20
extract_measured_kegg_ids, 21, *28*

fetch_kegg_chebi, 22
fetch_kegg_compound_lookup, *18, 19*, 22
fetch_kegg_pathway_metabolites, 23, *28*
fetch_kegg_reactome, 23
fetch_lion_lipid_ontology, 24
fetch_reactome_reactions, 25

get_cached_file, 26

perform_enrichment_analysis, *19*, 26
perform_enrichment_analysis_bg, *21*, 27
perform_gsea_analysis, *19*, 30
prepare_gmt_data, *19*, 31

run_de, *15*, 33