Package ‘ggpicrust2’

October 6, 2026

Type Package

Title Make 'PICRUSt2' Output Analysis and Visualization Easier

Version 2.5.19

Author Chen Yang [aut, cre],
Liangliang Zhang [aut]

Maintainer Chen Yang <cafferychen7850@gmail.com>

Description Provides a convenient way to analyze and visualize 'PICRUSt2' output with predefined plots and functions. Allows for generating statistical plots about microbiome functional predictions and offers customization options. Features a one-click option for creating publication-level plots, saving time and effort in producing professional-grade figures. Streamlines the 'PICRUSt2' analysis and visualization process. For more details, see Yang et al. (2023) <doi:10.1093/bioinformatics/btad470>.

BugReports https://github.com/cafferychen777/ggpicrust2/issues

URL https://github.com/cafferychen777/ggpicrust2,
https://cafferychen777.github.io/ggpicrust2/

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports aplot, dplyr, ggplot2, grid, ggh4x, readr, tibble, tidyr, ggprism, patchwork, ggplotify, magrittr, progress, stats, methods, grDevices, tidygraph, ggraph, utils

Depends R (>= 3.5.0)

Suggests Biobase, ggdendro, KEGGREST, ComplexHeatmap, BiocGenerics, knitr, rmarkdown, testthat (>= 3.0.0), ALDEx2, DESeq2, edgeR, GGally, limma, MicrobiomeStat, SummarizedExperiment, circlize, lefser, Maaslin2, metagenomeSeq, fgsea, clusterProfiler, enrichplot, DOSE, ggVennDiagram, UpSetR, igraph, ggridges, ggrepel, logging

Config/testthat/edition 3

biocViews Microbiome, Metagenomics, Software

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2026-10-06 18:20:02 UTC

Contents

aggregate_taxa_contributions . . . 3
calculate_smart_text_size . . . 5
color_themes . . . 5
compare_daa_results . . . 6
compare_gsea_daa . . . 7
compare_metagenome_results . . . 9
create_gradient_colors . . . 12
create_legend_theme . . . 12
create_pathway_class_theme . . . 13
daa_annotated_results_df . . . 14
daa_results_df . . . 15
ec_reference . . . 16
format_pvalue_smart . . . 17
get_available_themes . . . 17
get_color_theme . . . 18
get_significance_colors . . . 18
get_significance_stars . . . 19
ggpicrust2 . . . 19
gsea_pathway_annotation . . . 22
import_MicrobiomeAnalyst_daa_results . . . 23
kegg_abundance . . . 24
kegg_pathway_reference . . . 25
ko2kegg_abundance . . . 26
ko_abundance . . . 29
ko_reference . . . 30
ko_to_go_reference . . . 30
ko_to_kegg_reference . . . 32
legend_annotation_utils . . . 34
metacyc_abundance . . . 34
metacyc_reference . . . 35
metacyc_to_ec_reference . . . 36
metadata . . . 36
pathway_annotation . . . 37
pathway_daa . . . 39
pathway_errorbar . . . 44
pathway_errorbar_table . . . 48
pathway_gsea . . . 51
pathway_heatmap . . . 56

pathway_pca 61
pathway_ridgeplot 64
pathway_volcano 66
prepare_gene_sets 69
preview_color_theme 69
read_contrib_file 70
read_pathway_contrib_file 71
read_strat_file 72
resolve_annotation_overlaps 73
safe_extract 74
smart_color_selection 74
taxa_contribution_bar 75
taxa_contribution_heatmap 77
visualize_gsea 78

Index **82**

`aggregate_taxa_contributions`

Aggregate taxa contributions for visualization

Description

Core aggregation function that bridges PICRUSt2 contribution data with differential abundance analysis results. Optionally maps ASV/OTU IDs to taxonomic names and filters to significant pathways.

Usage

```
aggregate_taxa_contributions(
  contrib_data,
  taxonomy = NULL,
  tax_level = "Genus",
  top_n = 10,
  daa_results_df = NULL,
  pathway_ids = NULL,
  p_threshold = 0.05,
  contribution_col = "auto"
)
```

Arguments

`contrib_data`	A data.frame from `read_contrib_file` or `read_strat_file`.
`taxonomy`	Optional data frame or matrix mapping taxon IDs to taxonomy. Taxon IDs may be stored in a standard ID column or in explicit row names. Supports QIIME2 format (semicolon-delimited taxonomy strings) or DADA2 format (separate columns for each rank).

tax_level	Character. Taxonomic rank for aggregation. One of "Kingdom", "Phylum", "Class", "Order", "Family", "Genus", "Species". Default "Genus".
top_n	Integer. Number of top taxa to keep by total contribution; remaining are lumped as "Other". Default 10.
daa_results_df	Optional data.frame from pathway_daa, used to filter contributions to significant pathways.
pathway_ids	Optional character vector of pathway IDs to filter. Alternative to daa_results_df.
p_threshold	Numeric. Significance cutoff when using daa_results_df. Default 0.05. Must be in the range (0, 1].
contribution_col	Character. Column to aggregate. Use "auto" to select the first available column from norm_taxon_function_contrib, taxon_function_abun, taxon_rel_function_abun, and abundance.

Details

With daa_results_df, significant feature IDs are used as filters. KEGG pathway IDs are expanded to member KOs only when the contribution input is KO-level. This selects KO rows; output function_id values remain KO IDs and are not reconstructed pathway contributions. With pathway-level input (for example MetaCyc), IDs are matched directly. Aggregation sums the selected contribution column within each sample/function/taxonomic label. Percentage plots normalize those sums afterward; aggregation itself does not convert raw abundance to fractions.

Taxonomy can be provided in two formats:

- QIIME2: A column named Taxon or taxonomy containing semicolon-delimited strings (e.g., "k__Bacteria;p__Firmicutes;...")
- DADA2: Separate columns for each rank (Kingdom, Phylum, etc.)

Contribution table identifier columns (sample, function_id, and taxon) and requested pathway/function IDs must be non-empty and non-missing, because R's aggregation functions otherwise omit missing keys without preserving contribution totals. Contribution tables must not mix gene-family-level and pathway-level identifiers in the same aggregation.

Value

A tidy data.frame with columns: sample, function_id, taxon_label, contribution.

Examples

```
# Basic usage with synthetic data
contrib <- data.frame(
  sample = rep(c("S1", "S2"), each = 6),
  function_id = rep(c("K00001", "K00002", "K00003"), 4),
  taxon = rep(c("ASV1", "ASV2"), each = 3, times = 2),
  taxon_function_abun = seq_len(12)
)
agg <- aggregate_taxa_contributions(contrib, top_n = 2)
head(agg)
```

calculate_smart_text_size *Smart Text Size Calculator*

Description

Smart Text Size Calculator

Usage

```
calculate_smart_text_size(n_items, base_size = 10, min_size = 8, max_size = 14)
```

Arguments

`n_items`	Number of items to display
`base_size`	Base text size
`min_size`	Minimum text size
`max_size`	Maximum text size

Value

Calculated text size

color_themes *Color Theme System for ggpicrust2*

Description

This module provides a comprehensive color theme system for ggpicrust2 visualizations, including journal-specific themes, colorblind-friendly palettes, and intelligent color selection based on data characteristics.

compare_daa_results *Compare the Consistency of Statistically Significant Features*

Description

This function compares the consistency and inconsistency of statistically significant features obtained using different methods in ‘pathway_daa‘ from the ‘ggpicrust2‘ package. It creates a report showing the number of common and different features identified by each method, and the features themselves.

Usage

```
compare_daa_results(daa_results_list, method_names, p_values_threshold = 0.05)
```

Arguments

`daa_results_list`
: A list of data frames containing statistically significant features obtained using different methods.

`method_names`
: A character vector of names for each method used.

`p_values_threshold`
: A numeric value representing the threshold for the p-values. Features with p-values less than this threshold are considered statistically significant. Default is 0.05. Must be in the range (0, 1].

Details

Each list element is one discovery set. Split multi-test output such as ALDEx2 by its `method` column before comparison; otherwise the tests' discoveries are pooled within that element. This function does not rerun testing or adjust p-values again.

For multi-group DAA output, each discovery is compared as a `feature + group-pair` unit. The group pair is treated as unordered for this set-level comparison, because the function compares whether methods identified a significant difference, not the effect-size direction. This prevents the same feature from being counted as method-consistent when different methods found it in different pairwise contrasts, while still treating `A vs B` and `B vs A` as the same biological comparison. Rows whose two group labels are identical are invalid self-comparisons and are rejected. If all significant discoveries share one group pair, the printed feature lists use feature IDs only for backward-readable output; otherwise feature lists include the canonical contrast as `feature [group1 vs group2]`.

Value

A data frame with the comparison results. The data frame has the following columns:

- `method`: The name of the method.
- `num_features`: The total number of statistically significant features obtained by the method.
- `num_common_features`: The number of features that are common to other methods.

- num_diff_features: The number of features that are different from other methods.
- common_features: The names of the features that are common to all methods.
- diff_features: The names of the features that are different from other methods.

Examples

```
# Minimal DAA-like results from three methods (no external dependencies required)
deseq2_df <- data.frame(
  feature = c("ko00010", "ko00020", "ko00564"),
  group1 = c("A", "A", "A"),
  group2 = c("B", "B", "B"),
  p_adjust = c(0.01, 0.20, 0.03),
  stringsAsFactors = FALSE
)

edgeR_df <- data.frame(
  feature = c("ko00010", "ko00680", "ko00564"),
  group1 = c("A", "A", "A"),
  group2 = c("B", "B", "B"),
  p_adjust = c(0.02, 0.04, 0.01),
  stringsAsFactors = FALSE
)

maaslin2_df <- data.frame(
  feature = c("ko00010", "ko03030", "ko00564"),
  group1 = c("A", "A", "A"),
  group2 = c("B", "B", "B"),
  p_adjust = c(0.03, 0.02, 0.04),
  stringsAsFactors = FALSE
)

daa_results_list <- list(DESeq2 = deseq2_df, edgeR = edgeR_df, Maaslin2 = maaslin2_df)
comparison_results <- compare_daa_results(
  daa_results_list = daa_results_list,
  method_names = c("DESeq2", "edgeR", "Maaslin2"),
  p_values_threshold = 0.05
)
comparison_results
```

---

compare_gsea_daa *Compare GSEA and DAA results*

Description

This function compares the results from Gene Set Enrichment Analysis (GSEA) and Differential Abundance Analysis (DAA) to identify similarities and differences.

Usage

```
compare_gsea_daa(
  gsea_results,
  daa_results,
  plot_type = "venn",
  p_threshold = 0.05
)
```

Arguments

gsea_results	A data frame containing GSEA results from the pathway_gsea function
daa_results	A data frame containing DAA results from the pathway_daa function
plot_type	A single character string specifying the visualization type: "venn", "upset", or "scatter"
p_threshold	A numeric value specifying the significance threshold. Must be in the range (0, 1].

Details

Venn and UpSet plots compare significant pathway sets with unique pathway IDs. Restrict inputs to the shared tested pathway universe for a method comparison; an untested pathway is not a non-significant result. Preserve each analysis's original multiple-testing adjustment. The scatter plot compares a GSEA score with a DAA effect size and requires one row per pathway in each input. If a result table contains multiple methods, contrasts, or group pairs for the same pathway, filter it to a single score/contrast context before using `plot_type = "scatter"`. For scatter plots, GSEA and DAA directions must be explicit. Preranked GSEA positive NES values represent `gsea_results$group1` versus `gsea_results$group2`, while DAA `log2_fold_change` values represent `daa_results$group2 / daa_results$group1`. The DAA effect size is therefore aligned to the GSEA-positive direction before plotting. For Venn and UpSet plots, if both inputs include `group1` and `group2`, each input must represent one comparable group pair. This prevents pathways found in different biological contrasts from being counted as method agreement. Direction metadata must be complete: an input may provide both columns or neither, but not only one of them. When neither method has significant pathways, Venn and UpSet requests return the same explicit zero-count summary plot.

Value

A list with two elements: `plot` (a ggplot2 object, or an UpSetR object when `plot_type = "upset"` and UpSetR is installed) and `results` (a named list with the overlap, GSEA-only, and DAA-only pathway sets plus their counts). For `plot_type = "scatter"`, `results$scatter_data` contains the merged score/effect-size table, including `daa_log2_fold_change_aligned`.

Examples

```
## Not run:
# Load example data
data(ko_abundance)
data(metadata)
```

```
# Prepare abundance data
abundance_data <- as.data.frame(ko_abundance)
rownames(abundance_data) <- abundance_data[, "#NAME"]
abundance_data <- abundance_data[, -1]

# Run the competitive camera test
gsea_results <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "camera"
)

# Test pathway abundance, not individual KO abundance.
pathway_abundance <- ko2kegg_abundance(data = ko_abundance)
daa_results <- pathway_daa(
  abundance = pathway_abundance,
  metadata = metadata,
  group = "Environment",
  daa_method = "LinDA"
)

common_ids <- intersect(gsea_results$pathway_id, daa_results$feature)
gsea_results <- gsea_results[gsea_results$pathway_id %in% common_ids, ]
daa_results <- daa_results[daa_results$feature %in% common_ids, ]

# Compare results
comparison <- compare_gsea_daa(
  gsea_results = gsea_results,
  daa_results = daa_results,
  plot_type = "venn"
)

## End(Not run)
```

`compare_metagenome_results`

Compare Metagenome Results

Description

Compare Metagenome Results

Usage

```
compare_metagenome_results(
  metagenomes,
  names,
```

```
  daa_method = "ALDEx2",
  p_adjust_method = "BH",
  reference = NULL,
  p.adjust = NULL,
  correlation_permutations = 999,
  correlation_seed = 123,
  correlation_p_adjust_method = "BH"
)
```

Arguments

`metagenomes` A list of metagenome matrices with rows as KOs and columns as samples. Each matrix must have unique, non-empty feature row names and sample column names, and finite non-negative numeric abundance values. Each matrix in the list should correspond to a different metagenome.

`names` A unique, non-empty character vector of names for the metagenomes in the same order as in the ‘metagenomes‘ list.

`daa_method` Character. Paired differential abundance method. Choices are `"ALDEx2"` (paired ALDEx2 t and Wilcoxon tests) or `"paired Wilcoxon"` (paired Wilcoxon signed-rank tests on sample-wise relative abundances). Methods that model the metagenomes as independent groups are not supported because all matrices represent the same aligned biological samples.

`p_adjust_method` A character specifying the method for p-value adjustment. Possible choices are: "BH" (Benjamini-Hochberg), "holm", "bonferroni", "hochberg", "fdr", and "none". The default is "BH".

`reference` Optional metagenome name used as the first group in pairwise DAA comparisons. Other metagenome pairs are still compared.

`p.adjust` Deprecated alias for `p_adjust_method`. Do not supply both parameters with different values.

`correlation_permutations` Non-negative integer. Number of joint sample-label permutations used to test the median per-feature Spearman correlation. Use 0 to skip correlation p-values. Default 999.

`correlation_seed` Non-negative integer used to generate correlation permutations reproducibly. The caller’s random-number state is restored.

`correlation_p_adjust_method` P-value adjustment method for the unique off-diagonal metagenome correlation tests. Default `"BH"`.

Details

Metagenome matrices are aligned to the same sample identifiers. Each DAA comparison uses the feature intersection for that pair, so an unrelated third metagenome cannot remove testable features from the pair. Correlations use the feature intersection across all metagenomes so every matrix entry is based on the same feature universe. Because each matrix measures the same biological

samples, DAA must retain this pairing. Independent-group DAA methods would treat repeated measurements as independent replicates and are therefore rejected.

Correlation inference permutes the sample columns of one metagenome jointly across all features. This preserves within-metagenome feature dependence while breaking only the cross-metagenome sample correspondence. Monte Carlo p-values use the plus-one correction `(b + 1) / (B + 1)`.

Value

A list containing three elements:

- "daa": paired differential abundance results for every metagenome pair.
- "correlation": a list containing `cor_matrix`, `p_matrix`, `p_adjust_matrix`, and `n_features_matrix`. Correlations are medians of finite per-feature Spearman correlations. P-values use joint sample-label permutation and are adjusted across unique off-diagonal metagenome pairs. Diagonal p-values are `NA`.
- "heatmap": a ComplexHeatmap object visualizing the correlation matrix. Use `print()` or `draw()` to display it.

References

Fernandes AD, Macklaim JM, Linn TG, Reid G, Gloor GB. Unifying the analysis of high-throughput sequencing datasets: characterizing RNA-seq, 16S rRNA gene sequencing and selective growth experiments by compositional data analysis. Microbiome. 2014;2:15.

Phipson B, Smyth GK. Permutation P-values Should Never Be Zero: Calculating Exact P-values When Permutations Are Randomly Drawn. Statistical Applications in Genetics and Molecular Biology. 2010;9(1).

Examples

```
library(dplyr)
library(ComplexHeatmap)
# Generate example data
set.seed(123)
# First metagenome
metagenome1 <- abs(matrix(rnorm(1000), nrow = 100, ncol = 10))
rownames(metagenome1) <- paste0("KO", 1:100)
colnames(metagenome1) <- paste0("sample", 1:10)
# Second metagenome
metagenome2 <- abs(matrix(rnorm(1000), nrow = 100, ncol = 10))
rownames(metagenome2) <- paste0("KO", 1:100)
colnames(metagenome2) <- paste0("sample", 1:10)
# Put the metagenomes into a list
metagenomes <- list(metagenome1, metagenome2)
# Define names
names <- c("metagenome1", "metagenome2")
# Call the function
results <- compare_metagenome_results(
  metagenomes,
  names,
```

```
  daa_method = "paired Wilcoxon",
  correlation_permutations = 99
)
# Print the correlation matrix
print(results$correlation$cor_matrix)
# Display the heatmap
print(results$heatmap)
```

---

create_gradient_colors *Create Gradient Colors*

Description

Creates gradient colors for fold change visualization

Usage

```
create_gradient_colors(theme_name = "default", n_colors = 11, diverging = TRUE)
```

Arguments

theme_name	Character string specifying the theme
n_colors	Positive integer specifying the number of colors in the gradient
diverging	Logical. Whether to create a diverging gradient (for fold changes)

Value

A vector of colors

---

create_legend_theme *Create Enhanced Legend Theme*

Description

Create Enhanced Legend Theme

Usage

```
create_legend_theme(
  position = "top",
  direction = "horizontal",
  title = NULL,
  title_size = 12,
  text_size = 10,
  key_size = 0.8,
  key_width = NULL,
  key_height = NULL,
  ncol = NULL,
  nrow = NULL,
  box_just = "center",
  margin = ggplot2::margin(0, 0, 0, 0)
)
```

Arguments

`position`	Legend position ("top", "bottom", "left", "right", "none")
`direction`	Legend direction ("horizontal", "vertical")
`title`	Legend title
`title_size`	Title font size
`text_size`	Text font size
`key_size`	Key size in cm
`key_width`	Key width
`key_height`	Key height
`ncol`	Number of columns
`nrow`	Number of rows
`box_just`	Legend box justification
`margin`	Legend margin

Value

ggplot2 theme elements

`create_pathway_class_theme` *Create Pathway Class Annotation Theme*

Description

Create Pathway Class Annotation Theme

Usage

```
create_pathway_class_theme(
  text_size = "auto",
  text_color = "black",
  text_face = "bold",
  text_family = "sans",
  text_angle = 0,
  text_hjust = 0.5,
  text_vjust = 0.5,
  bg_color = NULL,
  bg_alpha = 0.2,
  position = "left"
)
```

Arguments

text_size	Text size
text_color	Text color
text_face	Text face ("plain", "bold", "italic")
text_family	Text family
text_angle	Text angle in degrees
text_hjust	Horizontal justification (0-1)
text_vjust	Vertical justification (0-1)
bg_color	Background color
bg_alpha	Background alpha
position	Annotation position ("left", "right", "none")

Value

List of annotation styling parameters

---

daa_annotated_results_df

Differentially Abundant Analysis Results with Annotation

Description

This is a result dataset after processing 'kegg_abundance' through the 'pathway_daa' with the LinDA method and further annotation with 'pathway_annotation'.

Usage

```
daa_annotated_results_df
```

Format

A data frame with 10 variables:

adj_method Method used for adjusting p-values.

feature Feature being tested.

group1 One group in the comparison.

group2 The other group in the comparison.

method Statistical test used.

p_adjust Adjusted p-value.

p_values P-values from the statistical test.

pathway_class Class of the pathway.

pathway_description Description of the pathway.

pathway_map Map of the pathway.

pathway_name Name of the pathway.

Source

From ggpicrust2 package demonstration.

References

Douglas GM, Maffei VJ, Zaneveld J, Yurgel SN, Brown JR, Taylor CM, Huttenhower C, Langille MGI. PICRUSt2 for prediction of metagenome functions. Nat Biotechnol. 2020.

---

daa_results_df *DAA Results Dataset*

Description

This dataset is the result of processing 'kegg_abundance' through the 'LinDA' method in the 'pathway_daa' function. It includes information about the feature, groups compared, p values, and method used.

Usage

```
daa_results_df
```

Format

A data frame with columns:

adj_method Method used for p-value adjustment.

feature The feature (pathway) being compared.

group1 The first group in the comparison.

group2 The second group in the comparison.

method The method used for the comparison.

p_adjust The adjusted p-value from the comparison.

p_values The raw p-value from the comparison.

Source

From ggpicrust2 package demonstration.

References

Douglas GM, Maffei VJ, Zaneveld J, Yurgel SN, Brown JR, Taylor CM, Huttenhower C, Langille MGI. PICRUSt2 for prediction of metagenome functions. Nat Biotechnol. 2020.

`ec_reference` *EC Number Reference Dataset*

Description

A reference dataset mapping Enzyme Commission (EC) numbers to their descriptions. Used internally by ggpicrust2 for annotating enzyme-level functional predictions.

Usage

```
data("ec_reference")
```

Format

A data frame with 8405 observations and the following columns:

`id` Character. EC number in the format "EC:X.X.X.X"

`description` Character. Human-readable enzyme name/description

Source

KEGG REST API (https://rest.kegg.jp)

Examples

```
data("ec_reference")
head(ec_reference)
```

format_pvalue_smart *Smart P-value Formatting*

Description

Smart P-value Formatting

Usage

```
format_pvalue_smart(
  p_values,
  format = "smart",
  stars = TRUE,
  thresholds = c(0.001, 0.01, 0.05),
  star_symbols = c("***", "**", "*")
)
```

Arguments

p_values	Numeric vector of p-values
format	Character string specifying format type
stars	Logical, whether to include star symbols
thresholds	Numeric vector of significance thresholds
star_symbols	Character vector of star symbols

Value

Character vector of formatted p-values

get_available_themes *Get Available Color Themes*

Description

Get Available Color Themes

Usage

```
get_available_themes()
```

Value

A character vector of available theme names

get_color_theme *Get Color Theme*

Description

Get Color Theme

Usage

```
get_color_theme(theme_name = "default", n_colors = 8)
```

Arguments

theme_name	Character string specifying one of the names returned by ‘get_available_themes()‘
n_colors	Positive integer specifying the number of colors needed. Requests beyond a theme’s base palette are interpolated so categorical colors are not recycled exactly.

Value

A list containing theme colors and settings

get_significance_colors *Get Significance Colors*

Description

Get Significance Colors

Usage

```
get_significance_colors(
  p_values,
  thresholds = c(0.001, 0.01, 0.05),
  colors = c("#d73027", "#fc8d59", "#fee08b"),
  default_color = "#999999"
)
```

Arguments

p_values	Numeric vector of p-values
thresholds	Numeric vector of significance thresholds
colors	Character vector of colors for each significance level
default_color	Default color for non-significant values

Value

Character vector of colors

---

get_significance_stars *Get Significance Stars*

Description

Get Significance Stars

Usage

```
get_significance_stars(
  p_values,
  thresholds = c(0.001, 0.01, 0.05),
  symbols = c("***", "**", "*")
)
```

Arguments

p_values	Numeric vector of p-values
thresholds	Numeric vector of significance thresholds
symbols	Character vector of star symbols

Value

Character vector of star symbols

---

ggpicrust2 *This function integrates pathway name/description annotations, ten of the most advanced differential abundance (DA) methods, and visualization of DA results.*

Description

This function integrates pathway name/description annotations, ten of the most advanced differential abundance (DA) methods, and visualization of DA results.

Usage

```
ggpicrust2(
  file = NULL,
  data = NULL,
  metadata,
  group,
  pathway,
  daa_method = "ALDEx2",
  ko_to_kegg = FALSE,
  filter_for_prokaryotes = TRUE,
  p_adjust_method = "BH",
  order = "group",
  p_values_bar = TRUE,
  x_lab = NULL,
  select = NULL,
  reference = NULL,
  colors = NULL,
  p_values_threshold = 0.05,
  p.adjust = NULL
)
```

Arguments

`file` A character string representing the file path of the input file containing KO abundance data in picrust2 export format. The input file should have KO identifiers in the first column and sample identifiers in the first row. The remaining cells should contain the abundance values for each KO-sample pair.

`data` An optional data.frame or matrix containing KO/pathway abundance data. Data frames may use the same format as the input file, with feature identifiers in the first non-numeric column and samples in the remaining columns. Already-normalized matrix/data.frame inputs with feature identifiers in row names and samples in columns are also accepted. If provided, the function will use this data instead of reading from the file. By default, this parameter is set to NULL.

`metadata` A tibble, consisting of sample information

`group` A character, name of the group

`pathway` A character, consisting of "EC", "KO", "MetaCyc"

`daa_method` a character specifying the method for differential abundance analysis, default is "ALDEx2", choices are: - "ALDEx2": ANOVA-Like Differential Expression tool for high throughput sequencing data - "DESeq2": Differential expression analysis based on the negative binomial distribution using DESeq2 - "edgeR": Exact test for differences between two groups of negative-binomially distributed counts using edgeR - "limma voom": Limma-voom framework for the analysis of RNA-seq data - "metagenomeSeq": Fit logistic regression models to test for differential abundance between groups using metagenomeSeq - "LinDA": Linear models for differential abundance analysis of microbiome compositional data - "Maaslin2": Multivariate Association with Linear Models (MaAsLin2)

for differential abundance analysis - "Lefser": Linear discriminant analysis effect size using the lefser package

`ko_to_kegg` Logical or logical-like string controlling conversion of KO abundance to KEGG pathway abundance.

`filter_for_prokaryotes` Logical. If TRUE (default), filters out KEGG pathways that are specific to eukaryotes (e.g., human diseases, organismal systems) when ko_to_kegg = TRUE. Set to FALSE to include all KEGG pathways.

`p_adjust_method` A character specifying the method for p-value adjustment, default is "BH".

`order` A character to control the order of the main plot rows

`p_values_bar` A character to control if the main plot has the p_values bar

`x_lab` A character to control the x-axis label name, you can choose from "feature","pathway_name" and "description"

`select` A vector consisting of pathway names to be selected

`reference` A character, a reference group level for several DA methods

`colors` A vector consisting of colors number

`p_values_threshold` A numeric value specifying the threshold for statistical significance of differential abundance. Pathways with adjusted p-values below this threshold will be displayed in the plot. Default is 0.05. Must be in the range (0, 1].

`p.adjust` Deprecated alias for `p_adjust_method`. Do not supply both parameters with different values.

Value

A list containing:

- Numbered elements (1, 2, ...): Sub-lists for each DA method, each containing:
  - `plot`: A ggplot2 error bar plot visualizing the differential abundance results
  - `results`: A data frame of differential abundance results for that method
- `abundance`: The processed abundance data (KEGG pathway or original) for downstream analysis
- `metadata`: The metadata data frame
- `group`: The group variable name used in the analysis
- `daa_results_df`: The complete annotated DAA results data frame
- `ko_to_kegg`: Logical indicating whether KO to KEGG conversion was performed

These additional fields allow seamless integration with `pathway_pca` and `pathway_heatmap` for further visualization without re-preparing data.

Examples

```
## Not run:
# Requires MicrobiomeStat and KEGGREST; KEGG annotation uses the internet.
data("ko_abundance")
data("metadata")
results <- ggpicrust2(
  data = ko_abundance, metadata = metadata, group = "Environment",
  pathway = "KO", daa_method = "LinDA", ko_to_kegg = TRUE,
  x_lab = "pathway_name"
)
results[[1]]$plot
head(results[[1]]$results)

# Reuse the aligned abundance and metadata for exploratory plots.
pathway_pca(results$abundance, results$metadata, results$group)
sig_features <- unique(results$daa_results_df$feature[
  !is.na(results$daa_results_df$p_adjust) &
    results$daa_results_df$p_adjust < 0.05
])
if (length(sig_features) > 0) {
  pathway_heatmap(results$abundance[sig_features, , drop = FALSE],
                  results$metadata, results$group)
}
# For your own files, replace data = ko_abundance with file = "your_file.tsv"
# and supply matching metadata. See vignette("using_ggpicrust2").

## End(Not run)
```

---

gsea_pathway_annotation

Annotate GSEA results with pathway information

Description

This function adds pathway annotations to GSEA results, including pathway names, descriptions, and classifications.

Usage

```
gsea_pathway_annotation(gsea_results, pathway_type = "KEGG")
```

Arguments

gsea_results	A data frame containing GSEA results from the pathway_gsea function
pathway_type	A character string specifying the pathway type: "KEGG", "MetaCyc", or "GO"

Details

The pathway_id column must contain non-empty values without NA. Unknown but non-empty pathway IDs are retained as their own display names when no reference annotation is found.

Value

A data frame with annotated GSEA results

Examples

```
## Not run:
# Load example data
data(ko_abundance)
data(metadata)

# Prepare abundance data
abundance_data <- as.data.frame(ko_abundance)
rownames(abundance_data) <- abundance_data[, "#NAME"]
abundance_data <- abundance_data[, -1]

# Run GSEA analysis (using camera method - recommended)
gsea_results <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "camera"
)

# Annotate results
annotated_results <- gsea_pathway_annotation(
  gsea_results = gsea_results,
  pathway_type = "KEGG"
)

## End(Not run)
```

---

import_MicrobiomeAnalyst_daa_results

Import Differential Abundance Analysis (DAA) results from MicrobiomeAnalyst

---

Description

This function imports DAA results from an external platform such as MicrobiomeAnalyst. It can be used to compare the results obtained from different platforms.

Usage

```
import_MicrobiomeAnalyst_daa_results(
  file_path = NULL,
  data = NULL,
  method = "MicrobiomeAnalyst",
  group_levels = c("control", "treatment")
)
```

Arguments

- `file_path` — a character string specifying the path to the CSV file containing the DAA results from MicrobiomeAnalyst. If this parameter is NULL and no data frame is provided, an error will be thrown. Default is NULL.
- `data` — a data frame containing the DAA results from MicrobiomeAnalyst. Feature identifiers can be stored in a feature/name column or in non-default row names. P-value and adjusted p-value columns are detected by semantic names such as `Pvalues` and `FDR`; `Statistics` and fold-change columns are optional. If this parameter is NULL and no file path is provided, an error will be thrown. Default is NULL.
- `method` — a single non-empty character string specifying the method used for the DAA. This will be added as a new column in the returned data frame. Default is "MicrobiomeAnalyst".
- `group_levels` — a character vector specifying at least two unique group levels for the DAA. These values will be added as new columns in the returned data frame. Default is c("control", "treatment").

Value

a data frame containing the DAA results from MicrobiomeAnalyst with validated `feature`, `p_values`, and `p_adjust` columns, optional `Statistics` and `log2_fold_change` columns when present in the imported result, plus additional columns for the method and group levels.

Examples

```
# Synthetic export with explicit feature and probability columns.
exported <- data.frame(
  feature = c("K00001", "K00002"),
  Pvalues = c(0.01, 0.20), FDR = c(0.02, 0.20)
)
daa_results <- import_MicrobiomeAnalyst_daa_results(data = exported)

# The same table can be read from a CSV file.
export_file <- tempfile(fileext = ".csv")
utils::write.csv(exported, export_file, row.names = FALSE)
import_MicrobiomeAnalyst_daa_results(file_path = export_file)
unlink(export_file)
```

---

kegg_abundance *KEGG Abundance Dataset*

Description

A dataset derived from 'ko_abundance' by the function 'ko2kegg_abundance' in the ggpicrust2 package. Each row corresponds to a KEGG pathway, and each column corresponds to a sample.

Usage

```
kegg_abundance
```

Format

A data frame where rownames are KEGG pathways and column names are individual sample names, including: "SRR11393730", "SRR11393731", "SRR11393732", "SRR11393733", "SRR11393734", "SRR11393735", "SRR11393736", "SRR11393737", "SRR11393738", "SRR11393739", "SRR11393740", "SRR11393741", "SRR11393742", "SRR11393743", "SRR11393744", "SRR11393745", "SRR11393746", "SRR11393747", "SRR11393748", "SRR11393749", "SRR11393750", "SRR11393751", "SRR11393752", "SRR11393753", "SRR11393754", "SRR11393755", "SRR11393756", "SRR11393757", "SRR11393758", "SRR11393759", "SRR11393760", "SRR11393761", "SRR11393762", "SRR11393763", "SRR11393764", "SRR11393765", "SRR11393766", "SRR11393767", "SRR11393768", "SRR11393769", "SRR11393770", "SRR11393771", "SRR11393772", "SRR11393773", "SRR11393774", "SRR11393775", "SRR11393776", "SRR11393777", "SRR11393778", "SRR11393779"

Source

From ggpicrust2 package demonstration.

References

Douglas GM, Maffei VJ, Zaneveld J, Yurgel SN, Brown JR, Taylor CM, Huttenhower C, Langille MGI. PICRUSt2 for prediction of metagenome functions. Nat Biotechnol. 2020.

`kegg_pathway_reference` *KEGG Pathway Name Reference Dataset*

Description

A reference dataset mapping KEGG pathway IDs to their human-readable names. Used internally by ggpicrust2 for pathway annotation in DAA and GSEA results.

Usage

```
data("kegg_pathway_reference")
```

Format

A data frame with 505 observations and the following columns:

`pathway` Character. KEGG pathway ID in the format "koXXXXX" (e.g., "ko00010")

`pathway_name` Character. Human-readable pathway name (e.g., "Glycolysis / Gluconeogenesis")

Source

KEGG REST API (https://rest.kegg.jp)

Examples

```
data("kegg_pathway_reference")
head(kegg_pathway_reference)
```

---

ko2kegg_abundance *Convert KO abundance in picrust2 export files to KEGG pathway abundance*

Description

This function takes a file containing KO (KEGG Orthology) abundance data in picrust2 export format and converts it to KEGG pathway abundance data. The input file should be in .tsv, .txt, or .csv format.

Usage

```
ko2kegg_abundance(
  file = NULL,
  data = NULL,
  method = c("abundance", "sum"),
  filter_for_prokaryotes = TRUE,
  progress = interactive()
)
```

Arguments

`file` A character string representing the file path of the input file containing KO abundance data in picrust2 export format. The input file should have unique KO identifiers in the first column and sample identifiers in the first row. The remaining cells should contain the abundance values for each KO-sample pair.

`data` An optional data frame or numeric matrix containing KO abundance data with one row per unique KO identifier. Data frames may use the same format as the input file, with KO identifiers in the first column, or data frames and matrices may store KO identifiers in row names. Sample columns must be numeric, finite, non-missing, and non-negative. If provided, the function will use this data instead of reading from the file. By default, this parameter is set to NULL.

`method` Method for calculating pathway abundance. One of:

- `"abundance"`: (Default) Upper-half mean aggregation matching the unstructured pathway abundance rule used by the PICRUSt2 pathway pipeline. This is a KO-to-KEGG pathway aggregation approximation, not a replacement for the full PICRUSt2 pathway pipeline with MinPath and structured MetaCyc pathway inference.
- `"sum"`: Simple summation of all KO abundances. This is the legacy method and may double-count KOs belonging to multiple pathways.

`filter_for_prokaryotes`

Logical. If TRUE (default), filters out KEGG pathways that are not relevant to prokaryotic (bacterial/archaeal) analysis. The function always removes non-pathway KEGG buckets before this filter is applied. The prokaryote filter removes pathways in categories such as:

- Human diseases (cancer, neurodegenerative diseases, addiction, etc.)
- Organismal systems (immune system, nervous system, endocrine system, etc.)

Bacterial infection pathways and antimicrobial resistance pathways are retained. Set to FALSE to include all KEGG pathways (for eukaryotic analysis or custom filtering).

`progress`

Logical. Whether to show a progress bar while aggregating pathways. Defaults to `interactive()` so non-interactive scripts and tests stay quiet.

Details

The default `"abundance"` method follows the unstructured pathway abundance rule in PICRUSt2's pathway pipeline:

1. For each pathway, collect abundances of all associated KOs present in the data
2. Sort the abundances in ascending order
3. Take the upper half of the sorted values using the PICRUSt2 indexing rule `sorted[int(n / 2):]` (equivalent to `floor(n / 2) + 1` through `n` in R's one-based indexing)
4. Calculate the mean as the pathway abundance

This approach has several advantages over simple summation:

- Does not inflate abundances for pathways containing more KOs
- More robust to missing or low-abundance KOs
- Provides a more accurate representation of pathway activity

The `"sum"` method is provided for backward compatibility and simply sums all KO abundances for each pathway.

Input sample columns must contain finite non-missing non-negative values. Missing values are rejected rather than ignored because the upper-half mean aggregation depends on the number and ordering of KO abundances available for each sample.

Input KO identifiers must be unique after cleaning optional `ko:` prefixes. Duplicate KO rows are rejected because the pathway aggregation rule treats each KO as one feature/reaction entry; repeated rows would duplicate evidence and distort upper-half mean or sum aggregation.

This function does not run MinPath, does not perform PICRUSt2's structured MetaCyc pathway inference, and does not estimate pathway coverage. It is intended as a practical offline KO-to-KEGG pathway aggregation step for downstream comparison and visualization.

Value

A data frame with KEGG pathway abundance values. Rows represent KEGG pathways, identified by their KEGG pathway IDs. Columns represent samples, identified by their sample IDs from the input file.

Pathway Filtering

Before abundance calculation, KEGG BRITE hierarchies and "Not Included in Pathway or Brite" pseudo-pathways are removed because they are not KEGG pathway maps and cannot be consistently annotated as pathways (for example, `ko99980`).

When `filter_for_prokaryotes = TRUE`, the function excludes KEGG pathways that are biologically irrelevant to prokaryotic organisms. KEGG reference pathways include pathways from all domains of life, and many human/animal-specific pathways would appear in bacterial analysis simply because some KOs are shared across organisms.

The following KEGG Level 2 categories are excluded:

- Cancer pathways (overview and specific types)
- Neurodegenerative diseases (Alzheimer's, Parkinson's, etc.)
- Substance dependence (addiction pathways)
- Cardiovascular diseases
- Endocrine and metabolic diseases
- Immune diseases
- Organismal systems (immune, nervous, endocrine, digestive, etc.)

The following are RETAINED even with filtering:

- Infectious disease: bacterial (Salmonella, E. coli, Tuberculosis, etc.)
- Drug resistance: antimicrobial (antibiotic resistance)
- All Metabolism pathways
- Genetic/Environmental Information Processing
- Cellular Processes

Examples

```
## Not run:
library(ggpicrust2)
library(readr)

# Example 1: Default - filtered for prokaryotic analysis
data(ko_abundance)
kegg_abundance <- ko2kegg_abundance(data = ko_abundance)

# Example 2: Include all pathways (for eukaryotic analysis)
kegg_abundance_all <- ko2kegg_abundance(data = ko_abundance, filter_for_prokaryotes = FALSE)

# Example 3: Using legacy sum method with filtering
kegg_abundance_sum <- ko2kegg_abundance(data = ko_abundance, method = "sum")

# Example 4: From file
input_file <- tempfile(fileext = ".tsv")
utils::write.table(ko_abundance, input_file, sep = "\t",
                   quote = FALSE, row.names = FALSE)
kegg_abundance <- ko2kegg_abundance(file = input_file)
```

```
unlink(input_file)

## End(Not run)
```

---

ko_abundance *KO Abundance Dataset*

Description

This is a demonstration dataset from the ggpicrust2 package, representing the output of PICRUSt2. Each row represents a KO (KEGG Orthology) group, and each column corresponds to a sample.

Usage

```
ko_abundance
```

Format

A data frame where rownames are KO groups and column names include #NAME and individual sample names, such as: "#NAME", "SRR11393730", "SRR11393731", "SRR11393732", "SRR11393733", "SRR11393734", "SRR11393735", "SRR11393736", "SRR11393737", "SRR11393738", "SRR11393739", "SRR11393740", "SRR11393741", "SRR11393742", "SRR11393743", "SRR11393744", "SRR11393745", "SRR11393746", "SRR11393747", "SRR11393748", "SRR11393749", "SRR11393750", "SRR11393751", "SRR11393752", "SRR11393753", "SRR11393754", "SRR11393755", "SRR11393756", "SRR11393757", "SRR11393758", "SRR11393759", "SRR11393760", "SRR11393761", "SRR11393762", "SRR11393763", "SRR11393764", "SRR11393765", "SRR11393766", "SRR11393767", "SRR11393768", "SRR11393769", "SRR11393770", "SRR11393771", "SRR11393772", "SRR11393773", "SRR11393774", "SRR11393775", "SRR11393776", "SRR11393777", "SRR11393778", "SRR11393779"

Source

From ggpicrust2 package demonstration.

References

Douglas GM, Maffei VJ, Zaneveld J, Yurgel SN, Brown JR, Taylor CM, Huttenhower C, Langille MGI. PICRUSt2 for prediction of metagenome functions. Nat Biotechnol. 2020.

ko_reference — *KEGG Orthology (KO) Reference Dataset*

Description

A comprehensive reference dataset mapping KEGG Orthology (KO) identifiers to their pathway classifications and descriptions. Each KO entry can appear in multiple rows if it belongs to multiple pathways.

Usage

```
data("ko_reference")
```

Format

A data frame with 58693 observations and the following columns:

`id` Character. KO identifier (e.g., "K00001")

`PathwayL1` Character. Top-level KEGG pathway category (e.g., "Metabolism")

`PathwayL2` Character. Second-level pathway category (e.g., "Carbohydrate metabolism")

`Pathway` Character. Specific pathway name with ID (e.g., "Glycolysis / Gluconeogenesis [PATH:ko00010]")

`description` Character. KO entry description with gene name and EC number

Source

KEGG REST API (https://rest.kegg.jp)

Examples

```
data("ko_reference")
head(ko_reference)

# Check pathway hierarchy
table(ko_reference$PathwayL1)
```

ko_to_go_reference — *KO to GO Reference Mapping Dataset*

Description

A comprehensive reference dataset that maps KEGG Orthology (KO) identifiers to Gene Ontology (GO) terms. This dataset enables GO pathway analysis in ggpicrust2 by providing the necessary mappings between functional predictions and GO biological processes, molecular functions, and cellular components.

Usage

```
data("ko_to_go_reference")
```

Format

A data frame with the following columns:

`go_id` Character. GO term identifier in the format "GO:XXXXXXX"

`go_name` Character. Human-readable name of the GO term

`category` Character. GO category code. Use `table(ko_to_go_reference$category)` to see available categories.

`ko_members` Character. Semicolon-separated list of KO identifiers associated with this GO term

Details

This dataset maps KEGG Orthology (KO) identifiers to Gene Ontology (GO) terms, enabling GO-level functional analysis of PICRUSt2 predictions.

The dataset is built from authoritative biological databases:

- KEGG REST API DBLINKS field for KO to GO cross-references
- EBI QuickGO API for GO term metadata (names and categories)

KEGG DBLINKS primarily cross-references Molecular Function (MF) GO terms, because KO entries describe individual gene functions that naturally correspond to molecular activities (enzyme activities, binding functions, etc.). The current dataset contains predominantly MF terms with a small number of CC (Cellular Component) terms.

Each GO term includes at least 3 associated KO identifiers in the reference. The observed overlap and the analysis size filter determine which terms can be tested; reference membership alone does not guarantee statistical power.

Source

- KEGG REST API (https://rest.kegg.jp) - KO entry DBLINKS section
- EBI QuickGO (https://www.ebi.ac.uk/QuickGO/) - GO term metadata

References

- Kanehisa, M., & Goto, S. (2000). KEGG: kyoto encyclopedia of genes and genomes. Nucleic acids research, 28(1), 27-30.
- Ashburner, M., et al. (2000). Gene ontology: tool for the unification of biology. Nature genetics, 25(1), 25-29.
- Chen Yang, et al. (2023). ggpicrust2: an R package for PICRUSt2 predicted functional profile analysis and visualization. Bioinformatics, 39(8), btad470.

See Also

`pathway_gsea`, `ko_abundance`, `metadata`

Examples

```
# Load the dataset
data("ko_to_go_reference")

# Explore the dataset structure
head(ko_to_go_reference)
str(ko_to_go_reference)

# Check the distribution of GO categories
table(ko_to_go_reference$category)

# Find GO terms related to polymerase activity
polymerase_terms <- ko_to_go_reference[
  grepl("polymerase", ko_to_go_reference$go_name, ignore.case = TRUE), ]
head(polymerase_terms)

# Get KO members for a specific GO term (RNA polymerase activity)
rna_pol <- ko_to_go_reference[ko_to_go_reference$go_id == "GO:0003899", ]
if (nrow(rna_pol) > 0) {
  ko_list <- strsplit(rna_pol$ko_members, ";")[[1]]
  cat("KO identifiers for RNA polymerase activity:", paste(ko_list, collapse = ", "))
}

# Use in pathway analysis
## Not run:
library(ggpicrust2)
library(tibble)

# Load example data
data("ko_abundance")
data("metadata")

# Perform GO pathway GSEA analysis
gsea_results <- pathway_gsea(
  abundance = ko_abundance %>% column_to_rownames("#NAME"),
  metadata = metadata,
  group = "Environment",
  method = "fgsea",
  pathway_type = "GO",
  go_category = "MF",
  rank_method = "signal2noise"
)

# View results
head(gsea_results)

## End(Not run)
```

`ko_to_kegg_reference` *KO to KEGG Pathway Reference Data*

Description

A comprehensive mapping between KEGG Orthology (KO) identifiers and KEGG pathways. This dataset contains mappings covering 532 pathways and 23,466 unique KO IDs, filtered to include only real KEGG pathway maps (5-digit IDs).

Usage

```
ko_to_kegg_reference
```

Format

A data frame with 9 variables:

pathway_id KEGG pathway identifier (e.g., "ko00010")

pathway_number KEGG pathway number

pathway_name Full name of the pathway

ko_id KEGG Orthology identifier (e.g., "K00001")

ko_description Description of the KO

ec_number EC number associated with the KO (if applicable)

level1 KEGG pathway hierarchy Level 1 classification

level2 KEGG pathway hierarchy Level 2 classification

level3 KEGG pathway hierarchy Level 3 classification

Details

This reference data is used by the `ko2kegg_abundance` function to convert KO abundance data to KEGG pathway abundance. The data is stored internally and does not require internet connectivity to use.

The dataset covers major KEGG pathway categories including:

- Metabolism
- Genetic Information Processing
- Environmental Information Processing
- Cellular Processes
- Organismal Systems
- Human Diseases

Source

KEGG database (https://www.kegg.jp/)

See Also

`ko2kegg_abundance` for converting KO abundance to pathway abundance

Examples

```
# Load the reference data
data(ko_to_kegg_reference)

# View structure
str(ko_to_kegg_reference)

# Get unique pathways
unique_pathways <- unique(ko_to_kegg_reference$pathway_id)
length(unique_pathways)

# Find KOs for a specific pathway
glycolysis_kos <- ko_to_kegg_reference[ko_to_kegg_reference$pathway_id == "ko00010", ]
head(glycolysis_kos)
```

`legend_annotation_utils` *Legend and Annotation Utilities for ggpicrust2*

Description

This module provides enhanced legend and annotation functionality for ggpicrust2 visualizations, including intelligent p-value formatting, significance marking, and customizable legend styling.

`metacyc_abundance` *MetaCyc Abundance Dataset*

Description

This is a demonstration dataset from the ggpicrust2 package, representing the output of PICRUSt2. Each row represents a MetaCyc pathway, and each column corresponds to a sample.

Usage

```
metacyc_abundance
```

Format

A data frame where rownames are MetaCyc pathways and column names include "pathway" and individual sample names, such as: "pathway", "SRR11393730", "SRR11393731", "SRR11393732", "SRR11393733", "SRR11393734", "SRR11393735", "SRR11393736", "SRR11393737", "SRR11393738", "SRR11393739", "SRR11393740", "SRR11393741", "SRR11393742", "SRR11393743", "SRR11393744", "SRR11393745", "SRR11393746", "SRR11393747", "SRR11393748", "SRR11393749", "SRR11393750", "SRR11393751", "SRR11393752", "SRR11393753", "SRR11393754", "SRR11393755", "SRR11393756", "SRR11393757", "SRR11393758", "SRR11393759", "SRR11393760", "SRR11393761", "SRR11393762", "SRR11393763", "SRR11393764", "SRR11393765", "SRR11393766", "SRR11393767", "SRR11393768", "SRR11393769", "SRR11393770", "SRR11393771", "SRR11393772", "SRR11393773", "SRR11393774", "SRR11393775", "SRR11393776", "SRR11393777", "SRR11393778", "SRR11393779"

Source

From ggpicrust2 package demonstration.

References

Douglas GM, Maffei VJ, Zaneveld J, Yurgel SN, Brown JR, Taylor CM, Huttenhower C, Langille MGI. PICRUSt2 for prediction of metagenome functions. Nat Biotechnol. 2020.

---

metacyc_reference *MetaCyc Pathway Reference Dataset*

Description

A reference dataset mapping MetaCyc pathway identifiers to their descriptions. Used internally by ggpicrust2 for annotating MetaCyc pathway analysis results.

Usage

```
data("metacyc_reference")
```

Format

A data frame with 2714 observations and the following columns:

`id` Character. MetaCyc pathway identifier (e.g., "GLYCOLYSIS", "TCA")

`description` Character. Human-readable pathway description

Source

MetaCyc database (https://metacyc.org)

Examples

```
data("metacyc_reference")
head(metacyc_reference)
```

metacyc_to_ec_reference

MetaCyc Pathway to EC Number Mapping Dataset

Description

A reference dataset mapping MetaCyc pathway identifiers to their associated Enzyme Commission (EC) numbers. Used internally by ggpicrust2 for MetaCyc pathway analysis, enabling the mapping between EC-level functional predictions and MetaCyc pathways.

Usage

```
data("metacyc_to_ec_reference")
```

Format

A data frame with 575 observations and the following columns:

`pathway` Character. MetaCyc pathway identifier (e.g., "1CMET2-PWY")

`ec_numbers` Character. Semicolon-separated list of EC numbers associated with the pathway

Source

MetaCyc database (https://metacyc.org)

Examples

```
data("metacyc_to_ec_reference")
head(metacyc_to_ec_reference)

# Count EC numbers per pathway
ec_counts <- sapply(strsplit(metacyc_to_ec_reference$ec_numbers, ";"), length)
summary(ec_counts)
```

metadata

Metadata for ggpicrust2 Demonstration

Description

This is a demonstration dataset from the ggpicrust2 package. It provides the metadata required for the demonstration functions in the package. The dataset includes environmental information for each sample.

Usage

```
metadata
```

Format

A tibble with each row representing metadata for a sample.

Sample1 Metadata for Sample1, including Environment

Sample2 Metadata for Sample2, including Environment

... ...

Source

ggpicrust2 package demonstration.

References

Douglas GM, Maffei VJ, Zaneveld J, Yurgel SN, Brown JR, Taylor CM, Huttenhower C, Langille MGI. PICRUSt2 for prediction of metagenome functions. Nat Biotechnol. 2020.

---

pathway_annotation *Pathway information annotation*

Description

This function serves two main purposes: 1. Annotating pathway information from PICRUSt2 output files or data frames. 2. Adding annotations to existing ‘pathway_daa‘ result rows. This function does not aggregate KO abundance, change feature identifiers, or rerun testing. Use ‘ko2kegg_abundance()‘ before DAA when KEGG pathways are the analysis unit.

Important: When ‘ko_to_kegg = TRUE‘, this function automatically filters pathways by ‘p_adjust < p_adjust_threshold‘. If no pathways meet this criterion, the function returns the original data with NA annotation columns and issues a detailed warning message with diagnostic information and recommendations.

Usage

```
pathway_annotation(
  file = NULL,
  data = NULL,
  pathway = NULL,
  daa_results_df = NULL,
  ko_to_kegg = FALSE,
  organism = NULL,
  p_adjust_threshold = 0.05
)
```

Arguments

`file` A character string, the path to the PICRUSt2 output file.

`data` A data frame containing pathway or function abundance data. This is useful for annotating objects returned by functions such as `ko2kegg_abundance`, where pathway IDs may be stored as row names.

`pathway` A character string, the type of pathway to annotate. Options are "KO", "EC", "MetaCyc", or "KEGG".

`daa_results_df` A data frame, the output from 'pathway_daa' function. When 'ko_to_kegg = TRUE', must contain columns: feature, p_values, p_adjust, and method.

`ko_to_kegg` Logical or logical-like string selecting online KEGG annotation for DAA results. Use TRUE for KEGG pathway rows already obtained by KO-to-pathway aggregation and testing. This flag does not convert abundance or turn KO-level p-values into pathway-level p-values. Default FALSE uses local reference annotations. TRUE requires internet access and annotates only rows below the significance threshold.

`organism` A character string specifying the KEGG organism code (e.g., 'hsa' for human, 'eco' for E. coli). Default is NULL, which retrieves generic KO information not specific to any organism. Only used when ko_to_kegg is TRUE.

`p_adjust_threshold` A numeric value specifying the significance threshold for filtering pathways when 'ko_to_kegg = TRUE'. Only pathways with 'p_adjust < p_adjust_threshold' will be annotated via KEGG API. Default is 0.05. Must be in the range (0, 1]. Ignored when 'ko_to_kegg = FALSE'.

Value

A data frame with annotated pathway information.

If using the function for the first use case (file input), the output data frame will include:

- The input feature-ID column (for example `#NAME` or `function_id`); existing identifiers are preserved.
- `description`: The description of the pathway.
- `sample1, sample2, ...`: Abundance values for each sample.

If `ko_to_kegg` is set to TRUE, the output data frame will also include:

- `pathway_name`: The name of the KEGG pathway.
- `pathway_description`: The description of the KEGG pathway.
- `pathway_class`: The class of the KEGG pathway.
- `pathway_map`: The KEGG pathway map ID.

Note: When `ko_to_kegg = TRUE`, only pathways with `p_adjust < p_adjust_threshold` are processed. If no pathways meet this criterion, all annotation columns will be NA, and a detailed warning message will be issued with diagnostic information.

When `ko_to_kegg` is TRUE, the function queries the KEGG database for pathway information. By default (organism = NULL), it retrieves generic KO information that is not specific to any organism. If you are interested in organism-specific pathway information, you can specify the KEGG organism code using the `organism` parameter.

Examples

```
# Annotate KO identifiers using the bundled reference, without changing abundance.
data("ko_abundance")
annotated_abundance <- pathway_annotation(data = ko_abundance, pathway = "KO")
head(annotated_abundance[, c("#NAME", "description")])

## Not run:
# Annotate DAA results at their existing feature level.
data("metadata")
daa_results <- pathway_daa(
  abundance = ko_abundance, metadata = metadata,
  group = "Environment", daa_method = "LinDA"
)
annotated_results <- pathway_annotation(
  pathway = "KO", daa_results_df = daa_results, ko_to_kegg = FALSE
)
# For KEGG pathway results, first aggregate with ko2kegg_abundance(),
# test those pathway rows, then use ko_to_kegg = TRUE for online annotation.

## End(Not run)
```

---

pathway_daa *Differential Abundance Analysis for Predicted Functional Pathways*

Description

Performs differential abundance analysis on predicted functional pathway data using various statistical methods. This function supports multiple methods for analyzing differences in pathway abundance between groups, including popular approaches like ALDEx2, DESeq2, edgeR, and others.

Usage

```
pathway_daa(
  abundance,
  metadata,
  group,
  daa_method = "ALDEx2",
  select = NULL,
  p_adjust_method = "BH",
  reference = NULL,
  include_abundance_stats = FALSE,
  include_effect_size = TRUE,
  p.adjust = NULL,
  .pre_aligned = FALSE,
  .sample_col = NULL,
  ...,
  linda_winsor = TRUE,
```

```
  linda_adaptive = TRUE,
  linda_pseudocount = 0.5
)
```

Arguments

`abundance` A data frame or matrix containing predicted functional pathway abundance, with pathways/features as rows and samples as columns. Data frames may also provide a leading non-numeric feature ID column (for example `#NAME`, `feature`, or `pathway`); it is converted to row names before sample alignment. Feature identifiers must be explicit, non-empty, and unique. The column names should match the sample names in metadata. Values should be finite, non-missing, non-negative counts or abundance measurements. Count-based backends that require or assume integer counts (ALDEx2, DESeq2, edgeR, and metagenomeSeq) round non-integer values with a warning before fitting.

`metadata` A data frame or tibble containing sample information. Must include a 'sample' column with sample identifiers matching the column names in abundance data.

`group` Character string specifying the column name in metadata that contains group information for differential abundance analysis. Group values must be non-missing and non-empty for all aligned/selected samples.

`daa_method` Character string specifying the method for differential abundance analysis. Available choices are:

- `"ALDEx2"`: ANOVA-Like Differential Expression tool
- `"DESeq2"`: Differential expression analysis based on negative binomial distribution
- `"edgeR"`: Exact test for differences between groups using negative binomial model
- `"limma voom"`: Limma-voom framework for RNA-seq analysis
- `"metagenomeSeq"`: Zero-inflated Gaussian mixture model
- `"LinDA"`: Linear models for differential abundance analysis
- `"Maaslin2"`: Multivariate Association with Linear Models
- `"Lefser"`: Linear discriminant analysis effect size

Default is "ALDEx2".

`select` Character vector of unique sample names to include in the analysis. If NULL (default), all samples are included. The selected dataset must still contain at least four samples, at least two groups, and at least two samples per group.

`p_adjust_method` Character string specifying the method for p-value adjustment. Choices are:

- `"BH"`: Benjamini-Hochberg procedure (default)
- `"holm"`: Holm's step-down method
- `"bonferroni"`: Bonferroni correction
- `"hochberg"`: Hochberg's step-up method
- `"fdr"`: False Discovery Rate
- `"none"`: No adjustment

`reference` Character string specifying the reference level for the group comparison. If NULL (default), the first level is used as reference. When supplied, it must exactly match one observed group level after sample alignment and any `select` filtering.

`include_abundance_stats` Logical value indicating whether to include abundance statistics (mean relative abundance and standard deviation per group) in the output. Default is FALSE. When the selected `daa_method` already provides a `log2_fold_change` column (ALDEx2 with effect size, DESeq2, edgeR, limma voom, LinDA, Maaslin2, metagenomeSeq), the method-native log2 fold change is preserved and the relative-abundance ratio is not recomputed.

`include_effect_size` Logical value indicating whether to compute ALDEx2 effect size information via `ALDEx2::aldex.effect()`. When TRUE, adds `effect_size`, `diff_btw`, `log2_fold_change`, `rab_all`, and `overlap` columns, aligning ALDEx2 output with the other DAA methods that return log2 fold changes by default. Only applicable for two-group comparisons with the ALDEx2 method; ignored otherwise. Default is TRUE; set to FALSE to skip the extra `aldex.effect()` computation. For a two-group analysis, failure to compute or validate the requested effect-size output stops the analysis.

`p.adjust` Deprecated alias for `p_adjust_method`. Do not supply both parameters with different values.

`.pre_aligned` Internal logical. Set to TRUE only when the caller has already aligned abundance columns and metadata rows in identical sample order.

`.sample_col` Internal character. Sample identifier column used when `.pre_aligned = TRUE`.

`...` Reserved for future backend-specific parameters. Additional arguments are currently rejected rather than silently ignored, because ignored model/covariate arguments can make the fitted analysis differ from the analysis the user intended.

`linda_winsor` Logical. Whether the LinDA backend should winsorize feature abundances before its log-ratio model. MicrobiomeStat's count winsorization converts to relative abundance, truncates each feature at its 97th percentile, rescales by the original sample totals and rounds the result. Default TRUE preserves the historical wrapper behavior; set FALSE when fractional predicted abundances must remain unrounded.

`linda_adaptive` Logical flag forwarded to MicrobiomeStat's `adaptive` argument. Default TRUE preserves the historical behavior, which depends on the installed backend version. Set FALSE to request fixed pseudo-count handling using `linda_pseudocount`.

`linda_pseudocount` Positive finite number passed to LinDA as `pseudo.cnt`. It is added to every cell when fixed pseudo-count handling is used and at least one zero is present. Default 0.5.

Value

A data frame containing the differential abundance analysis results. The structure of the results depends on the chosen DAA method. For methods that support multi-group comparisons (like

LinDA), when there are more than two groups, the results will contain separate rows for each feature in each pairwise comparison between the reference group and each non-reference group. The data frame includes the following columns:

- `feature`: Feature/pathway identifier
- `method`: The DAA method used
- `group1`: Reference group
- `group2`: Comparison group
- `p_values`: P-values for the comparison
- `p_adjust`: Adjusted p-values
- `adj_method`: Method used for p-value adjustment

Method-native adjusted p-values are preserved when the backend provides them directly (ALDEx2 `eBH`, DESeq2 `padj`, LinDA `padj`, and Maaslin2 `qval`). Other methods are adjusted by `pathway_daa()` using `stats::p.adjust()` and `p_adjust_method`. For wrapper-computed adjustments, p-values are adjusted within each method and pairwise comparison when `method`, `group1`, and `group2` columns are available.

Methods that fit a model on the abundance data (DESeq2, edgeR, limma voom, LinDA, Maaslin2, metagenomeSeq) return a `log2_fold_change` column computed in the method's own model space. ALDEx2 returns `log2_fold_change` (plus `effect_size`, `diff_btw`, `rab_all`, `overlap`) when `include_effect_size = TRUE` (the default), derived from `ALDEx2::aldex.effect()` in CLR space. Lefser returns an `lda_score` column instead, which is its native effect-size metric. LinDA additionally returns its method-native `standard_error`, `statistic`, `degrees_of_freedom`, `ci_lower_95`, and `ci_upper_95` columns. The interval is a nominal, pointwise two-sided 95 its native standard error and residual degrees of freedom. It is not multiplicity-adjusted and does not separately propagate uncertainty in the estimated bias correction or the upstream functional predictions.

When `include_abundance_stats = TRUE`, the following additional columns are included:

- `mean_rel_abundance_group1`: Mean relative abundance for group1
- `sd_rel_abundance_group1`: Standard deviation of relative abundance for group1
- `mean_rel_abundance_group2`: Mean relative abundance for group2
- `sd_rel_abundance_group2`: Standard deviation of relative abundance for group2

A `log2_fold_change` column from relative abundance is only added when the DAA method does not already provide one, to avoid conflating model-based and ratio-based effect sizes. If the requested abundance statistics cannot be calculated for every returned feature/group pair, the function fails instead of returning a partially annotated result table.

References

- ALDEx2: Fernandes et al. (2014) Unifying the analysis of high-throughput sequencing datasets: characterizing RNA-seq, 16S rRNA gene sequencing and selective growth experiments by compositional data analysis. Microbiome.
- DESeq2: Love et al. (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology.
- edgeR: Robinson et al. (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics.

- limma-voom: Law et al. (2014) voom: precision weights unlock linear model analysis tools for RNA-seq read counts. Genome Biology.
- metagenomeSeq: Paulson et al. (2013) Differential abundance analysis for microbial marker-gene surveys. Nature Methods.
- Maaslin2: Mallick et al. (2021) Multivariable Association Discovery in Population-scale Meta-omics Studies.

Examples

```
# Load example data
data(ko_abundance)
data(metadata)

# Prepare abundance data
abundance_data <- as.data.frame(ko_abundance)
rownames(abundance_data) <- abundance_data[, "#NAME"]
abundance_data <- abundance_data[, -1]

# Run differential abundance analysis using ALDEx2
results <- pathway_daa(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment"
)

# Using a different method (DESeq2)
deseq_results <- pathway_daa(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  daa_method = "DESeq2"
)

# Create example data with more samples
abundance <- data.frame(
  sample1 = c(10, 20, 30),
  sample2 = c(20, 30, 40),
  sample3 = c(30, 40, 50),
  sample4 = c(40, 50, 60),
  sample5 = c(50, 60, 70),
  row.names = c("pathway1", "pathway2", "pathway3")
)

metadata <- data.frame(
  sample = c("sample1", "sample2", "sample3", "sample4", "sample5"),
  group = c("control", "control", "treatment", "treatment", "treatment")
)

# Run differential abundance analysis using ALDEx2
results <- pathway_daa(abundance, metadata, "group")
```

```
# Using a different method (limma voom instead of DESeq2 for this small example)
limma_results <- pathway_daa(abundance, metadata, "group",
                             daa_method = "limma voom")

# Analyze specific samples only
subset_results <- pathway_daa(abundance, metadata, "group",
                              select = c("sample1", "sample2", "sample3", "sample4"))

# ALDEx2 returns effect size columns by default
# (effect_size, diff_btw, log2_fold_change, rab_all, overlap).
# Ranking by |log2_fold_change| is generally more biologically informative
# than ranking by p-value, especially for large datasets where small effects
# can reach statistical significance without being biologically meaningful.
aldex2_res <- pathway_daa(abundance, metadata, "group", daa_method = "ALDEx2")
head(aldex2_res)

# Opt out of the extra aldex.effect() computation if only p-values are needed
aldex2_pvals_only <- pathway_daa(abundance, metadata, "group",
                                 daa_method = "ALDEx2",
                                 include_effect_size = FALSE)
```

---

pathway_errorbar *The function pathway_errorbar() is used to visualize the results of functional pathway differential abundance analysis as error bar plots.*

Description

The function pathway_errorbar() is used to visualize the results of functional pathway differential abundance analysis as error bar plots.

Usage

```
pathway_errorbar(
  abundance,
  daa_results_df,
  Group,
  ko_to_kegg = FALSE,
  p_values_threshold = 0.05,
  order = "group",
  select = NULL,
  p_value_bar = TRUE,
  colors = NULL,
  x_lab = NULL,
  log2_fold_change_color = "#87ceeb",
  max_features = 30,
  color_theme = "default",
  pathway_class_colors = NULL,
```

```
  smart_colors = FALSE,
  accessibility_mode = FALSE,
  legend_position = "top",
  legend_direction = "horizontal",
  legend_title = NULL,
  legend_title_size = 12,
  legend_text_size = 10,
  legend_key_size = 0.8,
  legend_ncol = NULL,
  legend_nrow = NULL,
  pvalue_format = "numeric",
  pvalue_stars = TRUE,
  pvalue_colors = FALSE,
  pvalue_size = "auto",
  pvalue_angle = 0,
  pvalue_thresholds = c(0.001, 0.01, 0.05),
  pvalue_star_symbols = c("***", "**", "*"),
  pathway_class_text_size = "auto",
  pathway_class_text_color = "black",
  pathway_class_text_face = "bold",
  pathway_class_text_angle = 0,
  pathway_class_position = "right",
  pathway_names_text_size = "auto"
)
```

Arguments

`abundance` A data frame with row names representing pathways and column names representing samples. Values must be non-negative abundances. The function normalizes each sample by its total over all supplied features before computing group means and standard deviations. Supply the full feature matrix and use `select` to limit displayed rows; pre-filtering the matrix changes the denominator. Data frames may also provide a leading non-numeric feature ID column (for example `#NAME`, `feature`, or `pathway`); it is converted to row names before sample-count validation.

`daa_results_df` A data frame containing the results of the differential abundance analysis of the pathways, generated by the pathway_daa function. x_lab should be a column name of daa_results_df. Within the selected method and group pair, feature identifiers must be unique.

`Group` A vector assigning each sample to a group, not a metadata column name. Prefer `setNames(metadata$Environment, metadata$sample_name)` with your grouping and sample-ID columns. The groups are used to color the samples in the figure. Values must be non-missing and non-empty for all abundance columns, and must include the DAA result's `group1` and `group2` labels. A named vector is aligned to abundance column names; otherwise the vector is interpreted in abundance-column order.

`ko_to_kegg` A logical parameter indicating whether there was a convertion that convert ko abundance to kegg abundance.

`p_values_threshold`
: A numeric parameter specifying the threshold for statistical significance of differential abundance. Pathways with adjusted p-values below this threshold will be considered significant. Must be in the range (0, 1].

`order`
: A single character string controlling the ordering of the rows in the figure. The options are: "p_values" (order by p-values), "name" (order by pathway name), "group" (order by the group with the highest mean relative abundance), or "pathway_class" (order by the pathway category).

`select`
: A vector of pathway names to be included in the figure. This can be used to limit the number of pathways displayed. If NULL, all pathways will be displayed.

`p_value_bar`
: A logical parameter indicating whether to display a bar showing the p-value threshold for significance. If TRUE, the bar will be displayed.

`colors`
: A vector of colors to be used to represent the groups in the figure. Each color corresponds to a group. If NULL, colors will be selected based on the color_theme.

`x_lab`
: A character string to be used as the x-axis label in the figure. The default value is "description" for KOs'descriptions and "pathway_name" for KEGG pathway names.

`log2_fold_change_color`
: A character string specifying the color for log2 fold change bars. Default is "#87ceeb" (light blue). Can also be "auto" to use theme-based colors.

`max_features`
: A numeric parameter specifying the maximum number of features to display before issuing a warning. Default is 30. Set to a higher value to display more features, or Inf to disable the limit entirely.

`color_theme`
: A character string specifying the color theme to use. Options include: "default", "nature", "science", "cell", "nejm", "lancet", "colorblind_friendly", "viridis", "plasma", "minimal", "high_contrast", "pastel", "bold". Default is "default".

`pathway_class_colors`
: A vector of colors for pathway class annotations. If NULL, colors will be selected from the theme.

`smart_colors`
: A logical parameter indicating whether to use intelligent color selection based on data characteristics. Default is FALSE.

`accessibility_mode`
: A logical parameter indicating whether to use accessibility-friendly colors. Default is FALSE.

`legend_position`
: A character string specifying legend position. Options: "top", "bottom", "left", "right", "none". Default is "top".

`legend_direction`
: A character string specifying legend direction. Options: "horizontal", "vertical". Default is "horizontal".

`legend_title`
: A character string for legend title. If NULL, no title is displayed.

`legend_title_size`
: A numeric value specifying legend title font size. Default is 12.

`legend_text_size`
: A numeric value specifying legend text font size. Default is 10.

`legend_key_size`
: A numeric value specifying legend key size in cm. Default is 0.8.

`legend_ncol`
: A numeric value specifying number of columns in legend. If NULL, automatic layout is used.

`legend_nrow`
: A numeric value specifying number of rows in legend. If NULL, automatic layout is used.

`pvalue_format`
: A character string specifying p-value format. Options: "numeric", "scientific", "smart", "stars_only", "combined". Default is "numeric".

`pvalue_stars`
: A logical parameter indicating whether to display significance stars. Default is TRUE.

`pvalue_colors`
: A logical parameter indicating whether to use color coding for significance levels. Default is FALSE.

`pvalue_size`
: A numeric value or "auto" for p-value text size. Default is "auto".

`pvalue_angle`
: A numeric value specifying p-value text angle in degrees. Default is 0.

`pvalue_thresholds`
: A numeric vector of significance thresholds. Default is c(0.001, 0.01, 0.05).

`pvalue_star_symbols`
: A character vector of star symbols for significance levels. Default is c("***", "**", "*").

`pathway_class_text_size`
: A numeric value or "auto" for pathway class text size. Default is "auto".

`pathway_class_text_color`
: A character string for pathway class text color. Use "auto" for theme-based color. Default is "black".

`pathway_class_text_face`
: A character string for pathway class text face. Options: "plain", "bold", "italic". Default is "bold".

`pathway_class_text_angle`
: A numeric value specifying pathway class text angle in degrees. Default is 0.

`pathway_class_position`
: A character string specifying pathway class position. Options: "left", "right", "none". Default is "right".

`pathway_names_text_size`
: A numeric value or "auto" for pathway names (y-axis labels) text size. Default is "auto".

Details

The abundance panel contains descriptive means and standard deviations of per-sample relative abundance. If DAA supplies `log2_fold_change`, the effect panel preserves that model estimate. Otherwise it uses a pseudocount-stabilized log2 ratio of group mean relative abundances. A model coefficient need not equal the descriptive abundance ratio.

Value

A ggplot2 (patchwork) plot showing the differential abundance results, or NULL with a warning when all pathway annotations are missing. Check for significant results before plotting; an empty significant set is a valid analysis outcome.

Examples

```
## Not run:
# Requires the optional MicrobiomeStat and KEGGREST packages.
data("ko_abundance")
data("metadata")
kegg_abundance <- ko2kegg_abundance(data = ko_abundance)
sample_groups <- setNames(metadata$Environment, metadata$sample_name)

daa_results_df <- pathway_daa(
  abundance = kegg_abundance,
  metadata = metadata,
  group = "Environment",
  daa_method = "LinDA"
)
# For ALDEx2, select one test (e.g. ALDEx2_Welch's t test) first.
daa_annotated_results_df <- pathway_annotation(
  pathway = "KO",
  daa_results_df = daa_results_df,
  ko_to_kegg = TRUE
)

if (any(daa_annotated_results_df$p_adjust < 0.05, na.rm = TRUE)) {
  p <- pathway_errorbar(
    abundance = kegg_abundance,
    daa_results_df = daa_annotated_results_df,
    Group = sample_groups,
    ko_to_kegg = TRUE,
    order = "pathway_class",
    x_lab = "pathway_name"
  )
}
# See vignette("using_ggpicrust2") for the complete stepwise workflow.

## End(Not run)
```

---

pathway_errorbar_table

Generate Abundance Statistics Table for Pathway Analysis

Description

This function generates a table containing mean relative abundance, standard deviation, and log2 fold change statistics for pathways, similar to the data used in pathway_errorbar plots but returned as a data frame instead of a plot.

Usage

```
pathway_errorbar_table(
  abundance,
```

```
  daa_results_df,
  Group,
  ko_to_kegg = FALSE,
  p_values_threshold = 0.05,
  select = NULL,
  max_features = 30,
  metadata = NULL,
  sample_col = NULL
)
```

Arguments

`abundance` A data frame or matrix containing predicted functional pathway abundance, with pathways/features as rows and samples as columns. Data frames may also provide a leading non-numeric feature ID column (for example `#NAME`, `feature`, or `pathway`); it is converted to row names before sample alignment or Group length validation. The column names should match the sample names in metadata.

`daa_results_df` A data frame containing differential abundance analysis results from pathway_daa function. Must contain columns: feature, group1, group2, p_adjust. Within the selected method and group pair, feature identifiers must be unique.

`Group` A vector containing group assignments for each sample in the same order as the columns in abundance matrix. Alternatively, if metadata is provided, this should match the order of samples in metadata. Values must be non-missing and non-empty after alignment, and must include the selected DAA result's `group1` and `group2` labels. Without metadata, a vector named with sample IDs is aligned to abundance columns. Prefer this to an unchecked unnamed vector.

`ko_to_kegg` Logical value indicating whether to use KO to KEGG conversion. Default is FALSE.

`p_values_threshold` Numeric value for p-value threshold to filter significant features. Default is 0.05. Must be in the range (0, 1].

`select` Character vector of specific features to include. If NULL, all significant features are included.

`max_features` Maximum number of features to include in the table. Default is 30.

`metadata` Optional data frame containing sample metadata. If provided, the Group vector will be reordered to match the abundance column order.

`sample_col` Character string specifying the column name in metadata that contains sample identifiers. Default is NULL, which triggers auto-detection via the same logic used by `pathway_daa()` (common names like "sample", "Sample", "sample_id", "sample_name", or metadata rownames).

Details

Relative abundances use all supplied feature rows as each sample's denominator. Filtering the input matrix first changes the summaries. The table's log2 fold change is a descriptive group-mean ratio with a pseudocount, not a covariate-adjusted DAA model coefficient.

Value

A data frame containing the following columns:

- `feature`: Feature/pathway identifier
- `group1`: Reference group name
- `group2`: Comparison group name
- `mean_rel_abundance_group1`: Mean relative abundance for group1
- `sd_rel_abundance_group1`: Standard deviation of relative abundance for group1
- `mean_rel_abundance_group2`: Mean relative abundance for group2
- `sd_rel_abundance_group2`: Standard deviation of relative abundance for group2
- `log2_fold_change`: Log2 fold change (group2/group1)
- `p_adjust`: Adjusted p-value from differential analysis
- Additional annotation columns (e.g., `description`, `pathway_name`, `pathway_class`) if present in the input daa_results_df

Examples

```
## Not run:
# Load example data
data("ko_abundance")
data("metadata")

# Convert KO abundance to KEGG pathways
kegg_abundance <- ko2kegg_abundance(data = ko_abundance)

# Perform differential abundance analysis
daa_results_df <- pathway_daa(
  abundance = kegg_abundance,
  metadata = metadata,
  group = "Environment",
  daa_method = "LinDA"
)

# Annotate results
daa_annotated_results_df <- pathway_annotation(
  pathway = "KO",
  daa_results_df = daa_results_df,
  ko_to_kegg = TRUE
)

# Generate abundance statistics table
abundance_stats_table <- pathway_errorbar_table(
  abundance = kegg_abundance,
  daa_results_df = daa_annotated_results_df,
  Group = setNames(metadata$Environment, metadata$sample_name),
  ko_to_kegg = TRUE,
  p_values_threshold = 0.05
)
```

```
# View the results
head(abundance_stats_table)

## End(Not run)
```

---

pathway_gsea *Gene Set Enrichment Analysis for PICRUSt2 output*

Description

This function performs Gene Set Enrichment Analysis (GSEA) on PICRUSt2 predicted functional data to identify enriched pathways between different conditions.

Usage

```
pathway_gsea(
  abundance,
  metadata,
  group,
  pathway_type = "KEGG",
  method = "camera",
  covariates = NULL,
  contrast = NULL,
  inter.gene.cor = 0.01,
  rank_method = "signal2noise",
  nperm = 1000,
  min_size = 5,
  max_size = 500,
  p_adjust_method = "BH",
  seed = 42,
  go_category = "all",
  organism = "ko",
  p.adjust = NULL,
  comparison = NULL,
  transformation = c("voom", "logCPM"),
  gene_sets = NULL
)
```

Arguments

abundance — A data frame containing gene/enzyme abundance data, with feature IDs in row names and samples as columns. Data frames may also provide a leading non-numeric feature ID column (for example `#NAME`, `feature`, or `pathway`); it is converted to row names automatically. Values must be finite, non-missing, and non-negative count-like abundances; negative or non-finite values are rejected rather than being coerced to zero. For KEGG analysis: features should be KO

IDs (e.g., K00001). For MetaCyc analysis: features should be EC numbers (e.g., EC:1.1.1.1 or 1.1.1.1), NOT pathway IDs. MetaCyc pathway-like identifiers are rejected because GSEA requires gene/enzyme-level input; use `pathway_daa` for pathway-level MetaCyc abundance tables. For GO analysis: features should be KO IDs that will be mapped to GO terms. NOTE: This function requires gene-level data, not pathway-level abundances. For pathway abundance analysis, use `pathway_daa` instead

`metadata` A data frame containing sample metadata. After sample alignment, all retained samples must have non-missing, non-empty values in the grouping column.

`group` A character string specifying the column name in metadata that contains the grouping variable

`pathway_type` A single character string specifying the pathway type: "KEGG", "MetaCyc", or "GO"

`method` A single character string specifying the GSEA method:

- `"camera"`: Competitive gene set test using limma's camera function (default). Uses an inter-gene correlation adjustment under a competitive null.
- `"fry"`: Fast approximation to rotation gene set testing using limma's fry function. Self-contained test that is computationally efficient.
- `"fgsea"`: Fast preranked GSEA implementation. Note: preranked methods may produce unreliable p-values due to not accounting for inter-gene correlations (Wu et al., 2012).
- `"GSEA"` or `"clusterProfiler"`: clusterProfiler's GSEA implementation.

`covariates` A character vector specifying column names in metadata to use as covariates for adjustment. Only supported when method is "camera" or "fry"; supplying covariates with preranked methods is an error because those methods use a precomputed rank vector rather than a design matrix. Default is NULL (no covariates). Covariate values must be complete for all aligned samples; rows with missing model variables are rejected rather than being silently dropped by `model.matrix()`. The resulting design matrix must also be finite and full rank; constant or fully confounded covariates are rejected. Example: `covariates = c("age", "sex", "BMI")`

`contrast` For "camera" or "fry" methods, specify the coefficient or contrast to test. Supplying `contrast` with preranked methods is an error; use `comparison` instead. Default `NULL` automatically tests the single non-reference group coefficient in two-group designs. Multi-group designs must specify this explicitly. A character value must exactly match a design column name or a non-reference group level; substring matching is not used. A numeric scalar is treated as a design-column index, and a numeric vector must have length equal to the number of design columns. Named numeric vectors are matched and reordered by design column names; unnamed numeric vectors are interpreted in design column order.

`inter.gene.cor` Numeric value specifying the inter-gene correlation for camera method. Default is 0.01. Use NA to estimate correlation from data for each gene set.

`rank_method` A single character string specifying the ranking statistic for preranked methods (fgsea, GSEA, clusterProfiler): "signal2noise", "t_test", "log2_ratio", or "diff_abundance"

`nperm` An integer specifying the number of permutations (for clusterProfiler method only). The fgsea method uses adaptive multilevel splitting and does not require a fixed permutation count.

`min_size` An integer specifying the minimum gene set size

`max_size` An integer specifying the maximum gene set size

`p_adjust_method` A character string specifying the p-value adjustment method

`seed` An integer specifying the random seed for reproducibility

`go_category` A single character string specifying GO category to use. "all" (default) uses all categories present in the reference data. Valid categories are determined by the reference data (currently MF and CC). See `table(ko_to_go_reference$category)` for available categories.

`organism` Deprecated and has no effect. The KEGG and GO reference data bundled with ggpicrust2 are KO-based (organism-independent), so gene sets are returned in KO space regardless of this argument. Retained only for signature compatibility; passing any value other than the default `"ko"` emits a deprecation warning. Will be removed in a future release.

`p.adjust` Deprecated alias for `p_adjust_method`. Do not supply both parameters with different values.

`comparison` For preranked methods only (`"fgsea"`, `"GSEA"`, or `"clusterProfiler"`), an optional length-2 character vector `c(group1, group2)` defining the ranking direction. Ranking statistics are calculated as group1 versus group2: positive `signal2noise`, `t_test`, `diff_abundance`, and `log2_ratio` values indicate higher abundance in `group1` (for `log2_ratio`, `log2(group1 / group2)`). If `NULL`, exactly two aligned group levels must be present and their factor-level order is used. Multi-group preranked analyses must specify `comparison` explicitly.

`transformation` Transformation for camera/fry: `"voom"` (the historical default) uses count-dependent observation weights; `"logCPM"` uses `log2(1e6 * abundance / column_total + 0.5)` and abundance-trend variance moderation without voom weights. The latter is invariant to positive per-sample rescaling and can be used for predicted or relative abundances whose units are not observed read counts. It is a different statistical model, not a fallback after a voom failure or a guarantee of calibration. Requires positive sample totals. This parameter does not apply to preranked methods. Neither transformation accounts for uncertainty from the upstream functional prediction. Results also depend on the measured feature universe and, for camera, the correlation model.

`gene_sets` Optional named list of feature identifiers for explicitly defined gene sets. When supplied, replaces the bundled sets for all methods; size filtering and multiple-testing adjustment still apply. Identifiers must match the normalized abundance row names. `pathway_type` still controls input identifier handling. Do not combine custom sets with `organism` or `go_category`; these are bundled-reference selection arguments.

Details

Method Selection:

The `camera` method (default):

- Uses an inter-gene correlation adjustment (default 0.01)
- It supports covariate adjustment through the design matrix
- It performs a competitive test (genes in set vs. genes not in set)

The `fry` method is a fast alternative that:

- Performs a self-contained test (are genes in the set differentially expressed?)
- Is computationally very efficient for large numbers of gene sets
- Also supports covariate adjustment

Preranked methods use a ranked-feature null rather than camera's competitive model with a correlation adjustment. The hypotheses and input assumptions differ; neither method is universally calibrated for predicted abundances. For these methods, positive ES/NES values indicate gene-set enrichment near the top of the ranked list. With `comparison = c(group1, group2)`, the top of the list corresponds to features higher in `group1`; reverse `comparison` to reverse the biological direction.

Covariate Adjustment:

When using `method = "camera"` or `method = "fry"`, you can adjust for confounding variables by specifying them in the `covariates` parameter. This is particularly important in microbiome studies where factors like age, sex, BMI, and batch effects can influence results.

Value

A data frame containing GSEA results with columns:

- `pathway_id`: Pathway identifier
- `pathway_name`: Pathway name/description
- `size`: Number of genes in the pathway
- `direction`: Direction of enrichment ("Up" or "Down", for camera/fry)
- `pvalue`: Raw p-value
- `p.adjust`: Adjusted p-value (FDR)
- `method`: The method used for analysis

For fgsea/clusterProfiler methods, additional columns include ES, NES, leading_edge, group1, and group2. Positive ES/NES values are in the `group1` versus `group2` direction. For camera/fry methods, limma does not return NES; ggpicrust2 retains a legacy `NES` column containing a signed `-log10(pvalue)` score for visualization compatibility and labels it with `score_type` and `score_label`.

References

Wu, D., & Smyth, G. K. (2012). Camera: a competitive gene set test accounting for inter-gene correlation. Nucleic Acids Research, 40(17), e133.

Wu, D., Lim, E., Vaillant, F., Asselin-Labat, M. L., Visvader, J. E., & Smyth, G. K. (2010). ROAST: rotation gene set tests for complex microarray experiments. Bioinformatics, 26(17), 2176-2182.

Examples

```
## Not run:
# Load example data
data(ko_abundance)
data(metadata)

# Prepare abundance data
abundance_data <- as.data.frame(ko_abundance)
rownames(abundance_data) <- abundance_data[, "#NAME"]
abundance_data <- abundance_data[, -1]

# Method 1: Using camera (recommended) - accounts for inter-gene correlations
gsea_results <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "camera"
)

# Method 2: Using camera with covariate adjustment
gsea_results_adj <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  covariates = "Mouse_Sex",
  pathway_type = "KEGG",
  method = "camera"
)

# Method 3: Using fry for fast self-contained testing
gsea_results_fry <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "fry"
)

# Method 4: A different null, using a prespecified preranked comparison
gsea_results_fgsea <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "fgsea",
  comparison = c("Pro-survival", "Pro-inflammatory"),
  seed = 42
)

# Visualize results
visualize_gsea(gsea_results, plot_type = "enrichment_plot", n_pathways = 10)
```

```
## End(Not run)
```

---

pathway_heatmap *Create pathway heatmap with support for multiple grouping variables*

Description

This function creates a heatmap of the predicted functional pathway abundance data with support for single or multiple grouping variables. The function first performs z-score normalization on the abundance data, then converts it to a long format and orders the samples based on the grouping information. The heatmap supports nested faceting for multiple grouping variables and is created using the ‘ggplot2‘ library.

Usage

```
pathway_heatmap(
  abundance,
  metadata,
  group,
  secondary_groups = NULL,
  colors = NULL,
  font_size = 12,
  show_row_names = TRUE,
  show_legend = TRUE,
  custom_theme = NULL,
  low_color = "#0571b0",
  mid_color = "white",
  high_color = "#ca0020",
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  clustering_method = "complete",
  clustering_distance = "euclidean",
  dendro_line_size = 0.5,
  dendro_labels = FALSE,
  facet_by = NULL,
  colorbar_title = "Z Score",
  colorbar_position = "right",
  colorbar_width = 0.6,
  colorbar_height = 9,
  colorbar_breaks = NULL
)
```

Arguments

abundance A matrix or data frame of pathway abundance data, where columns correspond to samples and rows correspond to pathways. Data frames may also provide

a leading non-numeric feature ID column (for example `#NAME`, `feature`, or `pathway`); it is converted to row names before sample alignment. Values must be finite numeric values and may contain negative or zero-sum transformed sample profiles. Must contain at least two samples.

`metadata` A data frame of metadata, where each row corresponds to a sample and each column corresponds to a metadata variable. Grouping variables must be complete and contain at least two levels among the samples that align to the abundance columns.

`group` A character string specifying the column name in the metadata data frame that contains the primary group variable. Must contain at least two groups.

`secondary_groups` A character vector specifying additional grouping variables for creating nested faceted heatmaps. If NULL, only the primary group will be used. These variables will be used as secondary levels in the faceting hierarchy and must also remain complete with at least two levels after sample alignment.

`colors` A vector of colors used for the background of the facet labels in the heatmap. If NULL or not provided, a default color set is used for the facet strips.

`font_size` A numeric value specifying the font size for the heatmap.

`show_row_names` A logical value indicating whether to show row names in the heatmap.

`show_legend` A logical value indicating whether to show the legend in the heatmap.

`custom_theme` A custom theme for the heatmap.

`low_color` A character string specifying the color for low values in the heatmap gradient. Default is "#0571b0" (blue).

`mid_color` A character string specifying the color for middle values in the heatmap gradient. Default is "white".

`high_color` A character string specifying the color for high values in the heatmap gradient. Default is "#ca0020" (red).

`cluster_rows` A logical value indicating whether to cluster rows (pathways). Default is FALSE.

`cluster_cols` A logical value indicating whether to cluster columns (samples). Default is FALSE.

`clustering_method` A character string specifying the clustering method. Options: "complete", "average", "single", "ward.D", "ward.D2", "mcquitty", "median", "centroid". Default is "complete". Ward methods require `clustering_distance = "euclidean"`.

`clustering_distance` A character string specifying the distance metric. Options: "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "correlation", "spearman". Default is "euclidean". Use "average" or "complete" linkage for correlation or rank-based distances.

`dendro_line_size` A numeric value specifying the line width of dendrogram branches. Default is 0.5.

`dendro_labels` A logical value indicating whether to show dendrogram labels. Default is FALSE.

facet_by **[Deprecated]** A character string specifying an additional grouping variable for creating faceted heatmaps. This parameter is deprecated and will be removed in future versions. Use `secondary_groups` instead; supplying different values to both parameters is an error.

colorbar_title A character string specifying the title for the color bar. Default is "Z Score".

colorbar_position A character string specifying the position of the color bar. Options: "right", "left", "top", "bottom". Default is "right".

colorbar_width A numeric value specifying the width of the color bar. Default is 0.6.

colorbar_height A numeric value specifying the height of the color bar. Default is 9.

colorbar_breaks An optional numeric vector specifying custom breaks for the color bar.

Value

A ggplot heatmap object representing the heatmap of the predicted functional pathway abundance data.

Examples

```
library(ggpicrust2)
library(ggh4x)
library(dplyr)
library(tidyr)
library(tibble)
library(magrittr)

# Create example functional pathway abundance data
kegg_abundance_example <- matrix(rexp(30), nrow = 3, ncol = 10)
colnames(kegg_abundance_example) <- paste0("Sample", 1:10)
rownames(kegg_abundance_example) <- c("PathwayA", "PathwayB", "PathwayC")

# Create example metadata
metadata_example <- data.frame(
  sample_name = colnames(kegg_abundance_example),
  group = factor(rep(c("Control", "Treatment"), each = 5)),
  batch = factor(rep(c("Batch1", "Batch2"), times = 5))
)

# Custom colors for facet strips
custom_colors <- c("skyblue", "salmon")

# Example 1: Basic heatmap
pathway_heatmap(kegg_abundance_example, metadata_example, "group", colors = custom_colors)

# Example 2: Heatmap with row clustering
pathway_heatmap(
  abundance = kegg_abundance_example,
```

```
  metadata = metadata_example,
  group = "group",
  cluster_rows = TRUE,
  clustering_method = "complete",
  clustering_distance = "euclidean",
  dendro_line_size = 0.8
)

# Example 3: Heatmap with column clustering using correlation distance
pathway_heatmap(
  abundance = kegg_abundance_example,
  metadata = metadata_example,
  group = "group",
  cluster_cols = TRUE,
  clustering_method = "average",
  clustering_distance = "correlation"
)

# Example 4: Multi-level grouping with secondary_groups (NEW FEATURE)
pathway_heatmap(
  abundance = kegg_abundance_example,
  metadata = metadata_example,
  group = "group",
  secondary_groups = "batch",
  colors = c("lightblue", "lightcoral", "lightgreen", "lightyellow")
)

# Example 5: Custom colorbar settings
pathway_heatmap(
  abundance = kegg_abundance_example,
  metadata = metadata_example,
  group = "group",
  colorbar_title = "Expression Level",
  colorbar_position = "bottom",
  colorbar_width = 8,
  colorbar_height = 0.8,
  colorbar_breaks = c(-2, -1, 0, 1, 2)
)

# Example 6: Advanced heatmap with clustering and custom aesthetics
pathway_heatmap(
  abundance = kegg_abundance_example,
  metadata = metadata_example,
  group = "group",
  cluster_rows = TRUE,
  cluster_cols = FALSE,  # Don't cluster columns to preserve group order
  clustering_method = "average",
  clustering_distance = "manhattan",
  dendro_line_size = 1.0,
  low_color = "#053061",     # Dark blue
  mid_color = "#f7f7f7",     # Light gray
  high_color = "#67001f",    # Dark red
  colorbar_title = "Z-Score",
```

```
  colorbar_position = "left"
)

# Use real dataset
data("metacyc_abundance")
data("metadata")
metacyc_daa_results_df <- pathway_daa(
  abundance = metacyc_abundance %>% column_to_rownames("pathway"),
  metadata = metadata,
  group = "Environment",
  daa_method = "LinDA"
)
annotated_metacyc_daa_results_df <- pathway_annotation(
  pathway = "MetaCyc",
  daa_results_df = metacyc_daa_results_df,
  ko_to_kegg = FALSE
)
feature_with_p_0.05 <- metacyc_daa_results_df %>% filter(p_adjust < 0.05)

# Example 7: Real data with hierarchical clustering
pathway_heatmap(
  abundance = metacyc_abundance %>%
    right_join(
      annotated_metacyc_daa_results_df %>%
      select(all_of(c("feature","description"))),
      by = c("pathway" = "feature")
    ) %>%
    filter(pathway %in% feature_with_p_0.05$feature) %>%
    select(-"pathway") %>%
    filter(!is.na(description)) %>%
    distinct(description, .keep_all = TRUE) %>%
    column_to_rownames("description"),
  metadata = metadata,
  group = "Environment",
  cluster_rows = TRUE,
  clustering_method = "average",
  clustering_distance = "correlation",
  colors = custom_colors,
  low_color = "#2166ac",  # Custom blue for low values
  mid_color = "#f7f7f7",  # Light gray for mid values
  high_color = "#b2182b", # Custom red for high values
  colorbar_title = "Standardized Abundance"
)

# Example 8: Multiple grouping variables (NEW FEATURE)
# Create extended metadata with additional grouping variables
metadata_extended <- metadata_example %>%
  mutate(
    sex = factor(rep(c("Male", "Female"), times = 5)),
    age_group = factor(rep(c("Young", "Old"), each = 5))
  )

# Multi-level grouping with three variables
```

```
pathway_heatmap(
  abundance = kegg_abundance_example,
  metadata = metadata_extended,
  group = "group",                    # Primary grouping
  secondary_groups = c("batch", "sex"), # Secondary groupings
  colors = c("lightblue", "lightcoral")
)

# Example 9: Migration from facet_by to secondary_groups
# OLD WAY (deprecated, will show warning):
# pathway_heatmap(abundance, metadata, group = "Environment", facet_by = "Group")

# NEW WAY (recommended):
# pathway_heatmap(abundance, metadata, group = "Environment", secondary_groups = "Group")

# Example 10: Real data with multiple grouping variables
pathway_heatmap(
  abundance = metacyc_abundance %>%
    right_join(
      annotated_metacyc_daa_results_df %>%
      select(all_of(c("feature","description"))),
      by = c("pathway" = "feature")
    ) %>%
    filter(pathway %in% feature_with_p_0.05$feature) %>%
    select(-"pathway") %>%
    filter(!is.na(description)) %>%
    distinct(description, .keep_all = TRUE) %>%
    column_to_rownames("description"),
  metadata = metadata,
  group = "Environment",              # Primary: Pro-survival vs others
  secondary_groups = "Group",         # Secondary: Broad Institute vs Jackson Labs
  cluster_rows = TRUE,
  clustering_method = "average",
  clustering_distance = "correlation"
)
```

`pathway_pca` *Perform Principal Component Analysis (PCA) on functional pathway abundance data*

Description

This function performs PCA analysis on pathway abundance data and creates an informative visualization that includes a scatter plot of the first two principal components (PC1 vs PC2) with density plots for both PCs. The plot helps to visualize the clustering patterns and distribution of samples across different groups.

Usage

```
pathway_pca(abundance, metadata, group, colors = NULL, show_marginal = TRUE)
```

Arguments

`abundance`	A numeric matrix or data frame containing pathway abundance data. Rows represent pathways, columns represent samples. Data frames may also provide a leading non-numeric feature ID column (for example `#NAME`, `feature`, or `pathway`); it is converted to row names before sample alignment. Column names must match the sample names in metadata. Values must be numeric and cannot contain missing values (NA).
`metadata`	A data frame containing sample information. Must include a column for grouping samples (specified by the 'group' parameter). Sample identifiers are auto-detected from columns named sample_name, Sample_ID, SampleID, etc., or from rownames.
`group`	A character string specifying the column name in metadata that contains group information for samples (e.g., "treatment", "condition", "group"). Every sample retained after abundance/metadata alignment must have a non-missing, non-empty group label, and at least two groups must remain.
`colors`	Optional. A character vector of colors for different groups. Length must match the number of unique groups. If NULL, default colors will be used.
`show_marginal`	Logical. Whether to show marginal density plots for PC1 and PC2. Default is TRUE. Set to FALSE to show only the PCA scatter plot.

Details

PCA centers and scales each feature across samples before decomposing the transposed input matrix. It does not perform library-size or compositional normalization. Apply any study-specific preprocessing before calling the function. Visual separation is descriptive, not a hypothesis test or evidence of causation.

The function automatically aligns samples between abundance data and metadata, supporting various sample identifier formats. Pathways with zero variance across samples are filtered before PCA because they are variables in `prcomp(t(abundance))` and cannot be scaled. Sample profiles are kept as observations, even if a sample has zero variance across pathways. PCA confidence ellipses are drawn only for groups with at least four samples; smaller groups remain in the scatter plot but are skipped for ellipse estimation. Marginal densities require at least two samples per group; singleton groups are omitted from those panels, and the scatter plot is returned alone when no group has enough observations for density estimation.

Value

A ggplot object showing:

- Center: PCA scatter plot with confidence ellipses (95
- Top: Density plot for PC1
- Right: Density plot for PC2

Examples

```
# Create example abundance data
abundance_data <- matrix(rexp(30), nrow = 3, ncol = 10)
colnames(abundance_data) <- paste0("Sample", 1:10)
rownames(abundance_data) <- c("PathwayA", "PathwayB", "PathwayC")

# Create example metadata
metadata <- data.frame(
  sample_name = paste0("Sample", 1:10),
  group = factor(rep(c("Control", "Treatment"), each = 5))
)

# Basic PCA plot with default colors
pca_plot <- pathway_pca(abundance_data, metadata, "group")

# PCA plot with custom colors
pca_plot <- pathway_pca(
  abundance_data,
  metadata,
  "group",
  colors = c("blue", "red")  # One color per group
)

# PCA plot without marginal density plots
pca_plot <- pathway_pca(
  abundance_data,
  metadata,
  "group",
  show_marginal = FALSE
)


# Example with real data
data("metacyc_abundance")  # Load example pathway abundance data
data("metadata")          # Load example metadata

# Generate PCA plot
# Prepare abundance data
abundance_data <- as.data.frame(metacyc_abundance)
rownames(abundance_data) <- abundance_data$pathway
abundance_data <- abundance_data[, -which(names(abundance_data) == "pathway")]

# Create PCA plot
pathway_pca(
  abundance_data,
  metadata,
  "Environment",
  colors = c("green", "purple")
)
```

pathway_ridgeplot *Ridge Plot for GSEA Results*

Description

Creates a ridge plot (joy plot) to visualize the distribution of gene/KO abundances or fold changes for enriched pathways from GSEA analysis. The distribution is descriptive: it uses group means of the supplied abundance, not the covariate-adjusted model coefficients or voom weights from a camera/fry analysis. Direction colors describe the GSEA test and need not agree with every member's abundance ratio.

Usage

```
pathway_ridgeplot(
  gsea_results,
  abundance,
  metadata,
  group,
  comparison = NULL,
  pathway_reference = NULL,
  pathway_type = "KEGG",
  n_pathways = 10,
  sort_by = "p.adjust",
  show_direction = TRUE,
  colors = c(Down = "#3182bd", Up = "#de2d26"),
  title = "Ridge Plot: Gene Distribution in Enriched Pathways",
  x_lab = "log2 Fold Change",
  scale_height = 0.9,
  alpha = 0.7
)
```

Arguments

`gsea_results`	A data frame containing GSEA results from `pathway_gsea`. Must contain pathway_id column and either NES or direction column.
`abundance`	A data frame or matrix containing the original abundance data (genes/KOs as rows, samples as columns) used in the GSEA analysis. Data frames may also provide a leading non-numeric feature ID column (for example `#NAME`, `feature`, or `pathway`); it is converted to row names before sample alignment.
`metadata`	A data frame containing sample metadata with group information.
`group`	Character string specifying the column name in metadata for grouping.
`comparison`	Optional character vector of length 2 specifying `c(group1, group2)` for fold-change calculation. The ridge plot displays `log2(group2 / group1)`. If NULL, the data must contain exactly two group levels and their factor-level order is used. For multi-group metadata, specify `comparison` explicitly so the plot matches the GSEA contrast being interpreted.

`pathway_reference`
: A data frame containing pathway-to-gene mappings. Must have columns: pathway_id (or go_id for GO) and a column containing gene/KO members (semicolon-separated). If NULL, attempts to use built-in KEGG or GO reference data.

`pathway_type`
: Character string specifying the pathway type: "KEGG", "GO", or "MetaCyc". Default is "KEGG".

`n_pathways`
: Integer specifying the number of top pathways to display. Default is 10.

`sort_by`
: Character string specifying how to sort pathways: "NES" (Normalized Enrichment Score), "pvalue", or "p.adjust". Default is "p.adjust".

`show_direction`
: Logical. If TRUE, colors ridges by enrichment direction. Default is TRUE.

`colors`
: Named character vector with colors for "Up" and "Down" directions. Default is blue for down-regulated and red for up-regulated.

`title`
: Character string for plot title.

`x_lab`
: Character string for x-axis label.

`scale_height`
: Numeric value controlling the overlap of ridges. Default is 0.9. Higher values create more overlap.

`alpha`
: Numeric value for ridge transparency (0-1). Default is 0.7.

Details

The ridge plot displays the distribution of gene abundances (or fold changes) for genes within each enriched pathway. This visualization helps to:

- Understand the overall direction of change for each pathway
- Identify pathways with consistent vs. heterogeneous gene expression
- Compare the magnitude of changes across pathways

Fold changes are calculated as `log2(group2 / group1)`, where `group1` and `group2` come from `comparison` or, for a two-group dataset, from the factor-level order of `group`. GSEA `pathway_id` values must be non-empty and unique. Missing or empty `pathway_name` values are displayed as their corresponding `pathway_id`.

The plot requires the `ggridges` package to be installed.

Value

A ggplot2 object that can be further customized or saved.

See Also

`pathway_gsea`, `visualize_gsea`, `pathway_volcano`

Examples

```
## Not run:
library(ggpicrust2)
library(tibble)

# Load example data
data("ko_abundance")
data("metadata")

# Run GSEA (using camera method - recommended)
gsea_results <- pathway_gsea(
  abundance = ko_abundance %>% column_to_rownames("#NAME"),
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "camera"
)

# Create ridge plot
ridge_plot <- pathway_ridgeplot(
  gsea_results = gsea_results,
  abundance = ko_abundance %>% column_to_rownames("#NAME"),
  metadata = metadata,
  group = "Environment",
  n_pathways = 10
)
print(ridge_plot)

## End(Not run)
```

`pathway_volcano` *Volcano Plot for Pathway Differential Abundance Analysis*

Description

Creates a volcano plot to visualize the results of differential abundance analysis, showing both statistical significance (-log10 p-value) and effect size (log2 fold change).

Usage

```
pathway_volcano(
  daa_results,
  fc_col = "log2_fold_change",
  p_col = "p_adjust",
  label_col = "pathway_name",
  fc_threshold = 1,
  p_threshold = 0.05,
  label_top_n = 10,
```

```
  point_size = 2,
  point_alpha = 0.6,
  colors = c(Down = "#3182bd", `Not Significant` = "grey60", Up = "#de2d26"),
  show_threshold_lines = TRUE,
  title = "Volcano Plot: Pathway Differential Abundance",
  x_lab = "log2 Fold Change",
  y_lab = "-log10(Adjusted P-value)"
)
```

Arguments

- `daa_results` — A data frame containing differential abundance analysis results, typically from `pathway_daa`. Must contain columns for fold change, p-values, and optionally pathway names.
- `fc_col` — Character string specifying the column name for log2 fold change values. Default is "log2_fold_change". Legacy name "log2FoldChange" is also accepted.
- `p_col` — Character string specifying the column name for adjusted p-values. Default is "p_adjust".
- `label_col` — Character string specifying the column name for pathway labels. Default is "pathway_name". If NULL, no labels will be shown. A non-NULL column must exist when ‘label_top_n > 0‘.
- `fc_threshold` — Numeric. Absolute fold change threshold for significance. Default is 1 (2-fold change). Pathways with |log2FC| > fc_threshold are considered biologically significant.
- `p_threshold` — Numeric. P-value threshold for statistical significance. Default is 0.05. Must be in the range (0, 1].
- `label_top_n` — Integer. Number of top significant pathways to label. Default is 10. Set to 0 to disable labels.
- `point_size` — Numeric. Size of points in the plot. Default is 2.
- `point_alpha` — Numeric. Transparency of points (0-1). Default is 0.6.
- `colors` — Named character vector with colors for "Down", "Not Significant", and "Up". Names, when supplied, must match those three categories exactly. Default uses blue for down-regulated, grey for non-significant, and red for up-regulated.
- `show_threshold_lines` — Logical. Whether to show dashed lines for fold change and p-value thresholds. Default is TRUE.
- `title` — Character string for plot title. Default is "Volcano Plot: Pathway Differential Abundance".
- `x_lab` — Character string for x-axis label. Default is "log2 Fold Change".
- `y_lab` — Character string for y-axis label. Default is "-log10(Adjusted P-value)".

Details

The volcano plot is a scatter plot that shows statistical significance (y-axis) versus fold change (x-axis). Points are colored by significance:

- **Up (red)**: Pathways with p < p_threshold AND log2FC > fc_threshold
- **Down (blue)**: Pathways with p < p_threshold AND log2FC < -fc_threshold
- **Not Significant (grey)**: All other pathways

The function automatically labels the top N most significant pathways using `ggrepel::geom_text_repel()` if the ggrepel package is installed. Exact zero p-values are plotted at '.Machine$double.xmin'; positive subnormal p-values retain their original magnitude.

Value

A ggplot2 object that can be further customized or saved.

See Also

`pathway_daa`, `pathway_annotation`, `pathway_errorbar`

Examples

```
## Not run:
library(ggpicrust2)
library(tibble)

# Load example data
data("ko_abundance")
data("metadata")

# Convert KO to KEGG abundance
kegg_abundance <- ko2kegg_abundance(data = ko_abundance)

# Run differential abundance analysis
daa_results <- pathway_daa(
  abundance = kegg_abundance,
  metadata = metadata,
  group = "Environment",
  daa_method = "LinDA"
)

# Annotate results
daa_annotated <- pathway_annotation(
  pathway = "KO",
  ko_to_kegg = TRUE,
  daa_results_df = daa_results
)

# Create volcano plot
volcano_plot <- pathway_volcano(daa_annotated)
print(volcano_plot)

# Customize the plot
volcano_plot <- pathway_volcano(
  daa_annotated,
  fc_threshold = 0.5,
```

```
  p_threshold = 0.01,
  label_top_n = 15,
  colors = c("Down" = "darkblue", "Not Significant" = "lightgrey", "Up" = "darkred")
)

## End(Not run)
```

`prepare_gene_sets` *Prepare gene sets for GSEA*

Description

Prepare gene sets for GSEA

Usage

```
prepare_gene_sets(pathway_type = "KEGG", organism = "ko", go_category = "all")
```

Arguments

`pathway_type`	A single character string specifying the pathway type: "KEGG", "MetaCyc", or "GO"
`organism`	Deprecated and has no effect; gene sets are KO-based for both KEGG and GO. See `pathway_gsea` for details. Retained for signature compatibility only.
`go_category`	A single character string specifying the GO category to use. "all" (default) uses all categories. Valid values are determined by the reference data; see `table(ko_to_go_reference$category)`.

Value

A list of pathway gene sets

`preview_color_theme` *Preview Color Theme*

Description

Creates a visual preview of a color theme

Usage

```
preview_color_theme(theme_name = "default", save_plot = FALSE, filename = NULL)
```

Arguments

theme_name	Character string specifying the theme name
save_plot	Whether to save the preview plot
filename	Filename for saved plot

Value

A ggplot object showing the color preview

---

read_contrib_file *Read PICRUSt2 contribution file*

Description

Parses PICRUSt2 contribution files such as pred_metagenome_contrib.tsv. It also accepts the long contribution schema used by path_abun_contrib.tsv; for clarity, use read_pathway_contrib_file when reading pathway-level contribution output.

Usage

```
read_contrib_file(
  file = NULL,
  data = NULL,
  type = c("auto", "gene_family", "pathway")
)
```

Arguments

file	Path to the contribution file (.tsv, .txt, .csv, or gzipped variants).
data	A data.frame already loaded from the contribution file. If both file and data are provided, data is used.
type	Character. Contribution file level. One of "auto", "gene_family", or "pathway". Default "auto".

Details

The contribution file records how much each ASV/OTU contributes to the predicted abundance of each gene family or pathway in each sample. PICRUSt2 versions differ in whether they include norm_taxon_function_contrib; when that column is absent downstream aggregation can use taxon_function_abun or taxon_rel_function_abun. Contribution tables must describe one feature level at a time. Mixed gene-family identifiers such as KOs/ECs and pathway identifiers are rejected, because direct pathway matching and KEGG pathway-to-KO expansion have different biological meanings. Identifier columns (sample, function_id/function, and taxon) must contain non-empty values without NA; otherwise downstream aggregation would silently drop or mislabel contributions.

Value

A data.frame with columns: `sample`, `function_id`, `taxon`, contribution abundance columns from the original file, and `feature_level`.

Examples

```
# From a data.frame
contrib_df <- data.frame(
  sample = rep(c("S1", "S2"), each = 4),
  `function` = rep(c("K00001", "K00002"), 4),
  taxon = rep(c("ASV1", "ASV2"), each = 2, times = 2),
  taxon_function_abun = seq_len(8),
  check.names = FALSE
)
result <- read_contrib_file(data = contrib_df)
head(result)
```

---

read_pathway_contrib_file

Read PICRUSt2 pathway-level contribution file

Description

Parses `path_abun_contrib.tsv` output from PICRUSt2's pathway pipeline and returns the same normalized contribution schema used by `aggregate_taxa_contributions`.

Usage

```
read_pathway_contrib_file(file = NULL, data = NULL)
```

Arguments

`file`	Path to the contribution file (.tsv, .txt, .csv, or gzipped variants).
`data`	A data.frame already loaded from the contribution file. If both `file` and `data` are provided, `data` is used.

Value

A normalized data.frame with pathway IDs in `function_id`.

Examples

```
# From a data.frame
path_contrib_df <- data.frame(
  sample = rep(c("S1", "S2"), each = 2),
  `function` = rep(c("PWY-1234", "PWY-5678"), times = 2),
  taxon = c("ASV1", "ASV2", "ASV1", "ASV2"),
  taxon_function_abun = c(10, 5, 12, 4),
  check.names = FALSE
)
path_contrib <- read_pathway_contrib_file(data = path_contrib_df)
head(path_contrib)
```

---

read_strat_file *Read PICRUSt2 stratified abundance file*

Description

Parses the `pred_metagenome_strat.tsv` file produced by PICRUSt2's `--strat_out` flag and converts the wide format to tidy long format.

Usage

```
read_strat_file(file = NULL, data = NULL)
```

Arguments

`file`	Path to the stratified file (.tsv, .txt, .csv, or gzipped variants).
`data`	A data.frame already loaded from the stratified file. If both `file` and `data` are provided, `data` is used.

Details

The stratified file has function IDs in the first column, sequence/taxon IDs in the second column, and sample abundances in the remaining columns (wide format). This function pivots to long format for downstream analysis. Function IDs, taxon IDs, and sample column names must be non-empty; sample column names must also be unique.

Value

A tidy data.frame with columns: `function_id`, `taxon`, `sample`, `abundance`.

Examples

```
# From a data.frame
strat_df <- data.frame(
  `function` = c("K00001", "K00001", "K00002"),
  sequence = c("ASV1", "ASV2", "ASV1"),
  S1 = c(10, 5, 8),
  S2 = c(12, 3, 7),
  check.names = FALSE
)
result <- read_strat_file(data = strat_df)
head(result)
```

---

resolve_annotation_overlaps

Detect and Resolve Annotation Overlaps

Description

Detect and Resolve Annotation Overlaps

Usage

```
resolve_annotation_overlaps(labels, positions, min_distance = 1)
```

Arguments

`labels`	Character vector of labels
`positions`	Numeric vector of positions
`min_distance`	Minimum distance between labels

Value

Adjusted positions

safe_extract *Safely Extract Elements from a List*

Description

Safely extracts elements from a list, returning NA if the extraction fails

Usage

```
safe_extract(list, field, index = 1)
```

Arguments

list	A list object from which to extract elements
field	The name of the field to extract from the list
index	The index position to extract from the field. Default is 1

Value

A single character value if successful, otherwise NA_character_ when the field is absent, empty, or shorter than index.

Examples

```
# Create a sample list
my_list <- list(
  a = list(x = 1:3),
  b = list(y = 4:6)
)

# Extract existing element
safe_extract(my_list, "a", 1)

# Extract non-existing element (returns NA)
safe_extract(my_list, "c", 1)
```

smart_color_selection *Smart Color Selection*

Description

Intelligently selects colors based on data characteristics

Usage

```
smart_color_selection(
  n_groups,
  has_pathway_class = FALSE,
  data_type = "abundance",
  accessibility_mode = FALSE
)
```

Arguments

`n_groups` Number of groups in the data

`has_pathway_class` Whether pathway class information is available

`data_type` Type of data ("abundance", "pvalue", "foldchange")

`accessibility_mode` Whether to require accessibility-friendly colors. When TRUE, this constraint takes precedence over data-type heuristics.

Value

A list with suggested theme and colors

`taxa_contribution_bar` *Stacked bar plot of taxa contributions*

Description

Creates a stacked bar plot showing taxa contributions to predicted functional abundances, faceted by function or sample group.

Usage

```
taxa_contribution_bar(
  contrib_agg,
  metadata,
  group,
  function_ids = NULL,
  n_functions = 6,
  facet_by = "function",
  show_percentage = TRUE,
  color_theme = "default",
  font_size = 12,
  legend_position = "right",
  custom_title = NULL
)
```

Arguments

contrib_agg	A data.frame from aggregate_taxa_contributions.
metadata	A data.frame containing sample metadata.
group	Character. Column name in metadata for grouping samples.
function_ids	Optional character vector of function IDs to plot. If NULL (default), the top n_functions by between-sample variance in total contribution are shown. Single-sample inputs are ranked by total contribution because variance is undefined. Facets preserve this ranking or the order of explicitly supplied IDs.
n_functions	Integer. Number of functions to show when function_ids is NULL. Default 6.
facet_by	Character. Facet by "function" (default) or "group".
show_percentage	Logical. Normalize bars to 100%? Default TRUE.
color_theme	Character. Color theme name, passed to get_color_theme. Default "default".
font_size	Numeric. Base font size. Default 12.
legend_position	Character. Legend position. Default "right".
custom_title	Optional character string for the plot title.

Details

The sample, function_id, and taxon_label columns must contain non-empty values without NA. These columns define plotting and aggregation groups, so missing identifiers would otherwise be dropped by R aggregation or shown as unlabeled categories. When show_percentage = TRUE, every plotted sample/function combination must have a positive total contribution. Relative percentages are undefined for zero-total combinations; use show_percentage = FALSE to display absolute zero contributions.

Value

A ggplot2 object.

Examples

```
# Synthetic example
agg <- expand.grid(
  sample = c("S1", "S2", "S3", "S4"),
  function_id = c("K00001", "K00002"),
  taxon_label = c("Genus_A", "Genus_B", "Other"),
  stringsAsFactors = FALSE
)
agg$contribution <- runif(nrow(agg))
metadata <- data.frame(
  sample = c("S1", "S2", "S3", "S4"),
  group = c("Control", "Control", "Treatment", "Treatment")
)
p <- taxa_contribution_bar(agg, metadata, group = "group")
```

taxa_contribution_heatmap

Heatmap of taxa contributions across functions

Description

Creates a heatmap showing mean taxa contributions across pathways/functions, with optional clustering and pathway annotations.

Usage

```
taxa_contribution_heatmap(
  contrib_agg,
  annotation_data = NULL,
  n_functions = 20,
  cluster_rows = TRUE,
  cluster_cols = TRUE,
  clustering_method = "complete",
  clustering_distance = "euclidean",
  low_color = "#f7f7f7",
  high_color = "#ca0020",
  font_size = 12,
  dendro_line_size = 0.5,
  custom_title = NULL
)
```

Arguments

`contrib_agg` A data.frame from `aggregate_taxa_contributions`.

`annotation_data` Optional data.frame from `pathway_annotation` for replacing function IDs with readable descriptions. It must contain either `feature`/`description` or `pathway`/`pathway_name` columns.

`n_functions` Integer. Number of functions to include. Default 20.

`cluster_rows` Logical. Cluster rows (taxa)? Default TRUE.

`cluster_cols` Logical. Cluster columns (functions)? Default TRUE.

`clustering_method` Character. Method for `hclust`. Default `"complete"`. Ward methods require `clustering_distance = "euclidean"`.

`clustering_distance` Character. Distance metric. Default `"euclidean"`. Supported values are `"euclidean"`, `"maximum"`, `"manhattan"`, `"canberra"`, `"binary"`, and `"minkowski"`.

`low_color` Character. Color for low values. Default `"#f7f7f7"`.

`high_color` Character. Color for high values. Default `"#ca0020"`.

`font_size` Numeric. Base font size. Default 12.

`dendro_line_size`
: Numeric. Dendrogram line width. Default 0.5.

`custom_title`
: Optional plot title.

Details

The `sample`, `function_id`, and `taxon_label` columns must contain non-empty values without `NA`. These columns define plotting and aggregation groups, so missing identifiers would otherwise be dropped by R aggregation or shown as unlabeled categories. PICRUSt2 contribution outputs are sparse; combinations absent from `contrib_agg` are treated as zero when computing mean contribution across samples. If `annotation_data` contains multiple non-empty labels for the same plotted function ID, the function errors instead of silently choosing one label. Repeated rows with the same ID and same label are allowed, and label whitespace is normalized before comparison and display.

Value

A `ggplot2` or `patchwork` object.

Examples

```
agg <- data.frame(
  sample = rep(c("S1", "S2"), each = 6),
  function_id = rep(rep(c("K00001", "K00002", "K00003"), each = 2), 2),
  taxon_label = rep(c("Genus_A", "Genus_B"), 6),
  contribution = runif(12)
)
p <- taxa_contribution_heatmap(agg)
```

`visualize_gsea` *Visualize GSEA results*

Description

This function creates various visualizations for Gene Set Enrichment Analysis (GSEA) results. It automatically detects whether pathway names are available (from gsea_pathway_annotation()) and uses them for better readability, falling back to pathway IDs if names are not available.

Usage

```
visualize_gsea(
  gsea_results,
  plot_type = "enrichment_plot",
  n_pathways = 20,
  sort_by = "p.adjust",
  colors = NULL,
  abundance = NULL,
```

```
  metadata = NULL,
  group = NULL,
  network_params = list(),
  heatmap_params = list(),
  pathway_label_column = NULL,
  scale = NULL
)
```

Arguments

`gsea_results` A data frame containing GSEA results from the pathway_gsea function

`plot_type` Visualization type: "enrichment_plot" (a score-summary bar chart, not a running enrichment curve), "dotplot", "barplot", "network", or "heatmap". Network edges use leading-edge overlap; the heatmap shows per-pathway mean leading-edge abundance, standardized across samples. These two modes need leading-edge genes from preranked methods, which camera/fry do not provide.

`n_pathways` An integer specifying the number of pathways to display

`sort_by` A character string specifying the sorting criterion: "NES", "pvalue", or "p.adjust"

`colors` A vector of valid R colors used for the default heatmap group annotation palette. Colors are cycled when the number of observed groups exceeds the palette length.

`abundance` A data frame containing the original abundance data (required for heatmap visualization). Data frames may also provide a leading non-numeric feature ID column (for example `#NAME`, `feature`, or `pathway`); it is converted to row names before sample alignment.

`metadata` A data frame containing sample metadata (required for heatmap visualization)

`group` A character string specifying the column name in metadata that contains the grouping variable (required for heatmap visualization)

`network_params` A named list of network overrides. Supported names are ‘similarity_measure‘, ‘similarity_cutoff‘, ‘layout‘, ‘node_color_by‘, and ‘edge_width_by‘.

`heatmap_params` A named list of heatmap overrides. Supported names are ‘cluster_rows‘, ‘cluster_columns‘, ‘show_rownames‘, ‘annotation_colors‘, and ‘col_fun‘. Custom ‘annotation_colors‘ must be a list containing a named ‘Group‘ color vector whose names exactly match the observed group labels.

`pathway_label_column` A character string specifying which column to use for pathway labels. If NULL (default), the function will automatically use ’pathway_name’ if available, otherwise ’pathway_id’. This allows for custom labeling when using annotated GSEA results.

`scale` Optional palette/scale for customizing colors. Accepts: (1) a character vector of colors, (2) a function that returns colors given an integer (e.g., viridisLite::viridis), or (3) for non-heatmap plots, a ggplot2 scale object for the mapped aesthetic (e.g., ggplot2::scale_fill_gradientn(...)). When NULL, defaults keep current behavior. Applies to: enrichment_plot (fill, continuous), dotplot (color, continuous), barplot (fill, discrete Positive/Negative), network (color, diverging

around 0), heatmap (main heatmap col; row annotation stays default unless overridden in heatmap_params).

For every `plot_type`, the selected rows after sorting and `n_pathways` filtering must contain non-empty, unique `pathway_id` values so that each displayed pathway maps to exactly one GSEA result row.

Results from `pathway_gsea(method = "camera")` and `pathway_gsea(method = "fry")` include a legacy `NES` column for visualization compatibility, but limma does not estimate a true normalized enrichment score for these methods. When `score_label` is present, axis and legend labels use it instead of labeling the value as NES.

The function selects the top rows after sorting; it does not automatically filter by significance. Inspect adjusted p-values before interpreting a displayed pathway as significant.

Value

A ggplot2 object or ComplexHeatmap object

Examples

```
## Not run:
# Load example data
data(ko_abundance)
data(metadata)

# Prepare abundance data
abundance_data <- as.data.frame(ko_abundance)
rownames(abundance_data) <- abundance_data[, "#NAME"]
abundance_data <- abundance_data[, -1]

# Run GSEA analysis (using camera method - recommended)
gsea_results <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "camera"
)

# Create enrichment plot with pathway IDs (default)
visualize_gsea(gsea_results, plot_type = "enrichment_plot", n_pathways = 10)

# Annotate results for better pathway names
annotated_results <- gsea_pathway_annotation(
  gsea_results = gsea_results,
  pathway_type = "KEGG"
)

# Create plots with readable pathway names
visualize_gsea(annotated_results, plot_type = "dotplot", n_pathways = 20)
visualize_gsea(annotated_results, plot_type = "barplot", n_pathways = 15)
```

```
# Only preranked methods provide leading-edge genes for network/heatmap.
fgsea_results <- pathway_gsea(
  abundance_data, metadata, "Environment", method = "fgsea",
  comparison = c("Pro-survival", "Pro-inflammatory"), seed = 42
)
visualize_gsea(fgsea_results, plot_type = "network", n_pathways = 15)

# Use custom column for labels (if available)
visualize_gsea(annotated_results, plot_type = "barplot",
               pathway_label_column = "pathway_name", n_pathways = 10)

# Create heatmap
visualize_gsea(
  fgsea_results,
  plot_type = "heatmap",
  n_pathways = 15,
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment"
)

## End(Not run)
```

Index

∗ **datasets**
daa_annotated_results_df, 14
daa_results_df, 15
ec_reference, 16
kegg_abundance, 24
kegg_pathway_reference, 25
ko_abundance, 29
ko_reference, 30
ko_to_go_reference, 30
ko_to_kegg_reference, 32
metacyc_abundance, 34
metacyc_reference, 35
metacyc_to_ec_reference, 36
metadata, 36
∗ **functional-analysis**
ko_to_go_reference, 30
∗ **gene-ontology**
ko_to_go_reference, 30
∗ **microbiome**
ko_to_go_reference, 30

aggregate_taxa_contributions, 3, *71*, *76*, *77*

calculate_smart_text_size, 5
color_themes, 5
compare_daa_results, 6
compare_gsea_daa, 7
compare_metagenome_results, 9
create_gradient_colors, 12
create_legend_theme, 12
create_pathway_class_theme, 13

daa_annotated_results_df, 14
daa_results_df, 15

ec_reference, 16

format_pvalue_smart, 17

get_available_themes, 17
get_color_theme, 18, *76*
get_significance_colors, 18
get_significance_stars, 19
ggpicrust2, 19
gsea_pathway_annotation, 22

import_MicrobiomeAnalyst_daa_results, 23

kegg_abundance, 24
kegg_pathway_reference, 25
ko2kegg_abundance, 26, *33*, *38*
ko_abundance, 29, *31*
ko_reference, 30
ko_to_go_reference, 30
ko_to_kegg_reference, 32

legend_annotation_utils, 34

metacyc_abundance, 34
metacyc_reference, 35
metacyc_to_ec_reference, 36
metadata, *31*, 36

pathway_annotation, 37, *68*, *77*
pathway_daa, *4*, 39, *52*, *67*, *68*
pathway_errorbar, 44, *68*
pathway_errorbar_table, 48
pathway_gsea, *31*, 51, *64*, *65*, *69*
pathway_heatmap, *21*, 56
pathway_pca, *21*, 61
pathway_ridgeplot, 64
pathway_volcano, *65*, 66
prepare_gene_sets, 69
preview_color_theme, 69

read_contrib_file, *3*, 70
read_pathway_contrib_file, *70*, 71
read_strat_file, *3*, 72
resolve_annotation_overlaps, 73

safe_extract, 74
smart_color_selection, 74

taxa_contribution_bar, 75
taxa_contribution_heatmap, 77

visualize_gsea, *65*, 78