Package 'ggpicrust2'

August 26, 2025

Type Package

Title Make 'PICRUSt2' Output Analysis and Visualization Easier

Version 2.5.2

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://cafferyang.com/ggpicrust2/

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

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

Config/testthat/edition 3

biocViews Microbiome, Metagenomics, Software

NeedsCompilation no

Repository CRAN

Date/Publication 2025-08-25 23:00:02 UTC

Contents

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.

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.

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

```
library(magrittr)
library(ggpicrust2)
library(tibble)
data("metacyc_abundance")
data("metadata")

# Run pathway_daa function for multiple methods
methods <- c("DESeq2", "edgeR","Maaslin2")
daa_results_list <- lapply(methods, function(method) {
pathway_daa(abundance = metacyc_abundance %>% column_to_rownames("pathway"),
metadata = metadata, group = "Environment", daa_method = method)
})
```

```
names(daa_results_list) <- methods
# Correct Maaslin2 feature names by replacing dots with hyphens.
# Note: When using Maaslin2 as the differential abundance analysis method,
# it modifies the original feature names by replacing hyphens (-) with dots (.).
# This replacement can cause inconsistencies when trying to compare results from Maaslin2
# with those from other methods that do not modify feature names.
# Therefore, this line of code reverses that replacement, converting the dots back into
# hyphens for accurate and consistent comparisons across different methods.
daa_results_list[["Maaslin2"]]$feature <- gsub("\\.", "-", daa_results_list[["Maaslin2"]]$feature)

# Compare results across different methods
comparison_results <- compare_daa_results(daa_results_list = daa_results_list,
method_names = c("DESeq2", "edgeR", "Maaslin2"))
```

`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 character string specifying the visualization type: "venn", "upset", "scatter", or "heatmap"
`p_threshold`	A numeric value specifying the significance threshold

Value

A ggplot2 object or a list containing the plot and comparison 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
gsea_results <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "fgsea"
)

# Run DAA analysis
daa_results <- pathway_daa(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment"
)

# 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

Arguments

`metagenomes`	A list of metagenomes matrices with rows as KOs and columns as samples. Each matrix in the list should correspond to a different metagenome.

names	A vector of names for the metagenomes in the same order as in the ‘metagenomes‘ list.
daa_method	A character specifying the method for differential abundance analysis (DAA). Possible choices are: "ALDEx2", "DESeq2", "edgeR", "limma voom", "metagenomeSeq", "LinDA", "Maaslin2", and "Lefse". The default is "ALDEx2".
p.adjust	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	A character specifying the reference group level for DAA. This parameter is used when there are more than two groups. The default is NULL.

Value

A list containing two elements:

- "daa": a list of results from the ‘pathway_daa‘ function. Each result is a data frame containing the differential abundance analysis results with columns for the feature ID, the test statistic, the raw p-value, and the adjusted p-value.
- "correlation": a list with two elements: "cor_matrix" and "p_matrix", which are matrices of Spearman correlation coefficients and their corresponding p-values, respectively, between every pair of metagenomes.

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 = "LinDA")
# Print the correlation matrix
print(results$correlation$cor_matrix)
# Print the p-value matrix
print(results$correlation$p_matrix)
```

`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`	Number of colors in the gradient
`diverging`	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.

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 the theme name
n_colors	Integer specifying the number of colors needed

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,
  p.adjust = "BH",
  order = "group",
  p_values_bar = TRUE,
  x_lab = NULL,
  select = NULL,
  reference = NULL,
  colors = 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 containing KO abundance data in the same format as the input file. 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
ko_to_kegg	A character to control the conversion of KO abundance to KEGG abundance
p.adjust	a character specifying the method for p-value adjustment, default is "BH", choices are: - "BH": Benjamini-Hochberg correction - "holm": Holm's correction - "bonferroni": Bonferroni correction - "hochberg": Hochberg's correction - "fdr": False discovery rate correction - "none": No p-value adjustment.
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

Value

daa.results.df, a dataframe of DA results A list of sub-lists, each containing a ggplot2 plot ('plot') and a dataframe of differential abundance results ('results') for a specific DA method. Each plot visualizes the differential abundance results of a specific DA method, and the corresponding dataframe contains the results used to create the plot.

Examples

```
## Not run:
# Load necessary data: abundance data and metadata
abundance_file <- "path/to/your/abundance_file.tsv"
metadata <- read.csv("path/to/your/metadata.csv")

# Run ggpicrust2 with input file path
results_file_input <- ggpicrust2(file = abundance_file,
                                 metadata = metadata,
                                 group = "your_group_column",
                                 pathway = "KO",
                                 daa_method = "LinDA",
                                 ko_to_kegg = "TRUE",
                                 order = "pathway_class",
                                 p_values_bar = TRUE,
                                 x_lab = "pathway_name")

# Run ggpicrust2 with imported data.frame
abundance_data <- read_delim(abundance_file, delim="\t", col_names=TRUE, trim_ws=TRUE)

# Run ggpicrust2 with input data
results_data_input <- ggpicrust2(data = abundance_data,
                                 metadata = metadata,
                                 group = "your_group_column",
                                 pathway = "KO",
                                 daa_method = "LinDA",
                                 ko_to_kegg = "TRUE",
                                 order = "pathway_class",
                                 p_values_bar = TRUE,
                                 x_lab = "pathway_name")

# Access the plot and results dataframe for the first DA method
example_plot <- results_file_input[[1]]$plot
example_results <- results_file_input[[1]]$results

# Use the example data in ggpicrust2 package
data(ko_abundance)
data(metadata)
results_file_input <- ggpicrust2(data = ko_abundance,
                                 metadata = metadata,
                                 group = "Environment",
                                 pathway = "KO",
                                 daa_method = "LinDA",
                                 ko_to_kegg = TRUE,
                                 order = "pathway_class",
```

```
                                  p_values_bar = TRUE,
                                  x_lab = "pathway_name")
# Analyze the EC or MetaCyc pathway
data(metacyc_abundance)
results_file_input <- ggpicrust2(data = metacyc_abundance,
                                  metadata = metadata,
                                  group = "Environment",
                                  pathway = "MetaCyc",
                                  daa_method = "LinDA",
                                  ko_to_kegg = FALSE,
                                  order = "group",
                                  p_values_bar = TRUE,
                                  x_lab = "description")

## End(Not run)
```

`ggpicrust2_extended` *Integrated analysis with ggpicrust2 including GSEA*

Description

This function extends the ggpicrust2 functionality to include Gene Set Enrichment Analysis (GSEA).

Usage

```
ggpicrust2_extended(..., run_gsea = FALSE, gsea_params = list())
```

Arguments

`...`	Parameters passed to ggpicrust2()
`run_gsea`	Logical value indicating whether to perform GSEA analysis
`gsea_params`	List of parameters to pass to pathway_gsea()

Value

A list containing ggpicrust2 and GSEA results

Examples

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

# Run integrated analysis
integrated_results <- ggpicrust2_extended(
  data = ko_abundance,
  metadata = metadata,
  group = "Environment",
```

```
  pathway = "KO",
  daa_method = "LinDA",
  ko_to_kegg = TRUE,
  run_gsea = TRUE,
  gsea_params = list(
    method = "fgsea",
    rank_method = "signal2noise",
    nperm = 1000
  )
)

# Access DAA results
daa_results <- integrated_results$daa_results

# Access GSEA results
gsea_results <- integrated_results$gsea_results

# Access plots
daa_plot <- integrated_results$daa_plot
gsea_plot <- integrated_results$gsea_plot

## 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"

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
gsea_results <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "fgsea"
)

# 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.

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. If this parameter is NULL and no file path is provided, an error will be thrown. Default is NULL.
`method`	a 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 the group levels for the DAA. This 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 additional columns for the method and group levels.

Examples

```
## Not run:
# Assuming you have a CSV file named "DAA_results.csv" in your current directory
daa_results <- import_MicrobiomeAnalyst_daa_results(file_path = "DAA_results.csv")

## End(Not run)
```

---

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.

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)
```

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 containing KO abundance data in the same format as the input file. If provided, the function will use this data instead of reading from the file. By default, this parameter is set to NULL.

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. Each cell contains the abundance of a specific KEGG pathway in a given sample, calculated by summing the abundances of the corresponding KOs in the input file.

Examples

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

# Example 1: Demonstration with a hypothetical input file

# Prepare an input file path
input_file <- "path/to/your/picrust2/results/pred_metagenome_unstrat.tsv"

# Run ko2kegg_abundance function
kegg_abundance <- ko2kegg_abundance(file = input_file)

# Alternatively, read the data from a file and use the data argument
file_path <- "path/to/your/picrust2/results/pred_metagenome_unstrat.tsv"
ko_abundance <- read_delim(file_path, delim = "\t")
```

```
kegg_abundance <- ko2kegg_abundance(data = ko_abundance)

# Example 2: Working with real data
# In this case, we're using an existing dataset from the ggpicrust2 package.

# Load the data
data(ko_abundance)

# Apply the ko2kegg_abundance function to our real dataset
kegg_abundance <- ko2kegg_abundance(data = ko_abundance)

## 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.

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.

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. 2. Annotating pathway information from the output of ‘pathway_daa‘ function, and converting KO abundance to KEGG pathway abundance when ‘ko_to_kegg‘ is set to TRUE.

Usage

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

Arguments

file	A character string, the path to the PICRUSt2 output file.
pathway	A character string, the type of pathway to annotate. Options are "KO", "EC", or "MetaCyc".
daa_results_df	A data frame, the output from 'pathway_daa' function.
ko_to_kegg	A logical, decide if convert KO abundance to KEGG pathway abundance. Default is FALSE. Set to TRUE when using the function for the second use case.
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.

Value

A data frame with annotated pathway information. If using the function for the first use case, the output data frame will include the following columns:

- id: The pathway ID.
- 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 the following columns:

- 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.

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

```
## Not run:
# Example 1: Annotate pathways from PICRUSt2 output file
pathway_annotation(file = "path/to/picrust2_output.tsv",
                              pathway = "KO")

## End(Not run)

## Not run:
# Example 2: Annotate pathways from pathway_daa output
# Assuming you have daa_results from pathway_daa function
daa_results <- pathway_daa(abundance, metadata, group = "Group")
annotated_results <- pathway_annotation(pathway = "KO",
                              daa_results_df = daa_results,
                              ko_to_kegg = FALSE)

## End(Not run)
```

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.

Arguments

abundance	A data frame with row names representing pathways and column names representing samples. Each element represents the relative abundance of the corresponding pathway in the corresponding sample.
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.
Group	A data frame or a vector that assigns each sample to a group. The groups are used to color the samples in the figure.
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 p-values below this threshold will be considered significant.
order	A parameter 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 "smart".

`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 "left".

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

Value

A ggplot2 plot showing the error bar plot of the differential abundance analysis results for the functional pathways. The plot visualizes the differential abundance results of a specific differential abundance analysis method. The corresponding dataframe contains the results used to create the plot.

Examples

```
## Not run:
# Example 1: Analyzing KEGG pathway abundance
metadata <- read_delim(
  "path/to/your/metadata.txt",
  delim = "\t",
  escape_double = FALSE,
  trim_ws = TRUE
)

# data(metadata)

kegg_abundance <- ko2kegg_abundance(
  "path/to/your/pred_metagenome_unstrat.tsv"
)

# data(kegg_abundance)

# Please change group to "your_group_column" if you are not using example dataset
group <- "Environment"

daa_results_df <- pathway_daa(
  abundance = kegg_abundance,
  metadata = metadata,
  group = group,
  daa_method = "ALDEx2",
  select = NULL,
  reference = NULL
```

```
)

# Please check the unique(daa_results_df$method) and choose one
daa_sub_method_results_df <- daa_results_df[daa_results_df$method
== "ALDEx2_Welch's t test", ]

daa_annotated_sub_method_results_df <- pathway_annotation(
  pathway = "KO",
  daa_results_df = daa_sub_method_results_df,
  ko_to_kegg = TRUE
)

# Please change Group to metadata$your_group_column if you are not using example dataset
Group <- metadata$Environment

p <- pathway_errorbar(
  abundance = kegg_abundance,
  daa_results_df = daa_annotated_sub_method_results_df,
  Group = Group,
  p_values_threshold = 0.05,
  order = "pathway_class",
  select = daa_annotated_sub_method_results_df %>%
  arrange(p_adjust) %>%
  slice(1:20) %>%
  select("feature") %>% pull(),
  ko_to_kegg = TRUE,
  p_value_bar = TRUE,
  colors = NULL,
  x_lab = "pathway_name",
  log2_fold_change_color = "#FF5733" # Custom color for log2 fold change bars
)

# Example 2: Analyzing EC, MetaCyc, KO without conversions
metadata <- read_delim(
  "path/to/your/metadata.txt",
  delim = "\t",
  escape_double = FALSE,
  trim_ws = TRUE
)
# data(metadata)

metacyc_abundance <- read.delim("path/to/your/metacyc_abundance.tsv")

# data(metacyc_abundance)

group <- "Environment"

daa_results_df <- pathway_daa(
  abundance = metacyc_abundance %>% column_to_rownames("pathway"),
  metadata = metadata,
  group = group,
  daa_method = "LinDA",
  select = NULL,
```

```
  reference = NULL
)


daa_annotated_results_df <- pathway_annotation(
  pathway = "MetaCyc",
  daa_results_df = daa_results_df,
  ko_to_kegg = FALSE
)

Group <- metadata$Environment

p <- pathway_errorbar(
  abundance = metacyc_abundance %>% column_to_rownames("pathway"),
  daa_results_df = daa_annotated_results_df,
  Group = Group,
  p_values_threshold = 0.05,
  order = "group",
  select = NULL,
  ko_to_kegg = FALSE,
  p_value_bar = TRUE,
  colors = NULL,
  x_lab = "description",
  log2_fold_change_color = "#006400" # Dark green for log2 fold change bars
)

## 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 = "sample_name"
)
```

Arguments

abundance	A data frame or matrix containing predicted functional pathway abundance, with pathways/features as rows and samples as columns. 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.
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.
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.
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 "sample_name".

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 = "ALDEx2"
)

# Filter for specific method
daa_sub_method_results_df <- daa_results_df[
  daa_results_df$method == "ALDEx2_Welch's t test",
]

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

# Generate abundance statistics table
abundance_stats_table <- pathway_errorbar_table(
  abundance = kegg_abundance,
  daa_results_df = daa_annotated_sub_method_results_df,
  Group = metadata$Environment,
  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 = "fgsea",
  rank_method = "signal2noise",
  nperm = 1000,
  min_size = 10,
  max_size = 500,
  p.adjust = "BH",
  seed = 42,
  go_category = "BP"
)
```

Arguments

abundance	A data frame containing KO/EC/MetaCyc abundance data, with features as rows and samples as columns
metadata	A data frame containing sample metadata
group	A character string specifying the column name in metadata that contains the grouping variable
pathway_type	A character string specifying the pathway type: "KEGG", "MetaCyc", or "GO"
method	A character string specifying the GSEA method: "fgsea", "GSEA", or "clusterProfiler"
rank_method	A character string specifying the ranking statistic: "signal2noise", "t_test", "log2_ratio", or "diff_abundance"
nperm	An integer specifying the number of permutations
min_size	An integer specifying the minimum gene set size
max_size	An integer specifying the maximum gene set size
p.adjust	A character string specifying the p-value adjustment method
seed	An integer specifying the random seed for reproducibility
go_category	A character string specifying the GO category: "BP" (Biological Process), "MF" (Molecular Function), or "CC" (Cellular Component). Only used when pathway_type = "GO". Default is "BP".

Value

A data frame containing 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
gsea_results <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "fgsea"
)

# 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.

Arguments

`abundance`	A matrix or data frame of pathway abundance data, where columns correspond to samples and rows correspond to pathways. 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.
`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.
`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".

clustering_distance A character string specifying the distance metric. Options: "euclidean", "maximum", "manhattan", "canberra", "binary", "minkowski", "correlation", "spearman". Default is "euclidean".

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.

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(rnorm(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 = "ward.D2",
  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") %>%
    column_to_rownames("description"),
  metadata = metadata,
  group = "Environment",
  cluster_rows = TRUE,
  clustering_method = "ward.D2",
  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") %>%
```

```
    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 = "ward.D2",
  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)
```

Arguments

abundance	A numeric matrix or data frame containing pathway abundance data. Rows represent pathways, columns represent samples. 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 named "sample_name" matching the column names in abundance • A column for grouping samples (specified by the 'group' parameter)
group	A character string specifying the column name in metadata that contains group information for samples (e.g., "treatment", "condition", "group").
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.

Details

The function performs several validations on input data:

- Abundance matrix must have at least 2 pathways and 3 samples
- All values in abundance matrix must be numeric
- Sample names must match between abundance and metadata
- Group column must exist in metadata
- If custom colors are provided, they must be valid color names or codes

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(rnorm(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
)


# 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")
)
```

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 = "BP")
```

Arguments

pathway_type	A character string specifying the pathway type: "KEGG", "MetaCyc", or "GO"
organism	A character string specifying the organism (only relevant for KEGG and GO)
go_category	A character string specifying the GO category: "BP" (Biological Process), "MF" (Molecular Function), "CC" (Cellular Component), or "all"

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

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

The extracted element if successful, NA if extraction fails

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 use accessibility-friendly colors

Value

A list with suggested theme and colors

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
)
```

Arguments

gsea_results	A data frame containing GSEA results from the pathway_gsea function
plot_type	A character string specifying the visualization type: "enrichment_plot", "dotplot", "barplot", "network", or "heatmap"
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 colors for the visualization
abundance	A data frame containing the original abundance data (required for heatmap visualization)
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 list of parameters for network visualization
heatmap_params	A list of parameters for heatmap visualization
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.

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
gsea_results <- pathway_gsea(
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment",
  pathway_type = "KEGG",
  method = "fgsea"
)

# 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)

# Create network plot with custom labels
visualize_gsea(annotated_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(
  annotated_results,
  plot_type = "heatmap",
  n_pathways = 15,
  abundance = abundance_data,
  metadata = metadata,
  group = "Environment"
```

```
)

## End(Not run)
```

Index