Package ‘LorMe’

May 7, 2026

Title Lightweight One-Line Resolving Microbial Ecology Program

Version 2.0.2

Maintainer Ningqi Wang <`2434066068@qq.com`>

Description Provides a robust collection of functions tailored for microbial ecology analysis, encompassing both data analysis and visualization. It introduces an encapsulation feature that streamlines the process into a summary object. With the initial configuration of this summary object, users can execute a wide range of analyses with a single line of code, requiring only two essential parameters for setup. The package delivers comprehensive outputs including analysis objects, statistical outcomes, and visualization-ready data, enhancing the efficiency of research workflows. Designed with user-friendliness in mind, it caters to both novices and seasoned researchers, offering an intuitive interface coupled with adaptable customization options to meet diverse analytical needs.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports methods, ape, ggalluvial, ggplot2, ggpubr, grDevices,
magrittr, RColorBrewer, scales, stringr, tidyr, vegan, dplyr,
igraph, Hmisc, fdrtool, agricolae, multcompView, DescTools, HH,
coin

Suggests testthat (>= 3.0.0), ggrepel, readr, patchwork, grid,
permute, indicspecies, S4Vectors, tibble, DESeq2, phyloseq,
microeco

Config/testthat/edition 3

NeedsCompilation no

Author Ningqi Wang [aut, cre, cph],
Yaozhong Zhang [aut],
Gaofei Jiang [aut]

Depends R (>= 3.5.0)

Repository CRAN

Date/Publication 2026-01-12 07:30:02 UTC

Contents

`LorMe-package` *LorMe: Lightweight One-Line Resolving Microbial Ecology Program*

Description

Package for lightweight, one-line microbial ecology analysis including diversity, differential abundance, and network analysis.

Data I/O and interoperability

- `tax_summary` - Build S4 object from biom/meta files.
- `Trans_from_phylo` - Convert *phyloseq* objects to LorMe.
- `Trans_to_phylo` - Export LorMe to *phyloseq*.
- `Trans_from_microeco` - Convert *microtable* (microeco) to LorMe.
- `Trans_to_microeco` - Export LorMe to *microtable*.
- `sub_tax_summary` - Subset LorMe object by metadata.
- `add_tree` - Attach a phylogenetic tree to LorMe object.

Global configuration

- `LorMe_options` - One-line option manager (analysis level, colours, network thresholds, etc.).
- `object_config` - Interactive helper to set treatment column and comparison list.

One-line pipeline

- `LorMe_pipeline` - Run alpha diversity, beta diversity, differential abundance, indicator species, per-treatment and combined networks in a single call; results returned as a tidy list.

Community profiling

- `Alpha_diversity_calculator` - Alpha diversity analysis including Shannon, Simpson, Chao1, etc.
- `structure_plot` - Community structure analysis including PCA/PCoA/NMDS.
- `community_plot` - Community composition analysis including Bar, area and alluvial plots.
- `Top_taxa` - Calculate most abundant taxon across samples.

Differential analysis

- `Deseq_analysis` - DESeq analysis.
- `indicator_analysis` - Indicator analysis.
- `differential_bar` - Differential bar analysis.
- `volcano_plot` / `manhattan` - Volcano and Manhattan visualizations.

Network analysis

- `network_analysis` - Co-occurrence network (Cytoscape/Gephi ready).
- `network_withdiff` - Meta-network integrating differential taxa.
- `network_visual` / `network_visual_re` - Statistics and visualizations.
- `Module_composition` / `Module_abundance` - Module summaries and per-sample abundances.
- `nc` / `NC_remove` - Natural connectivity metrics.

Utility functions

- `Filter_function` - Pre-filter OTU/ASV by prevalence or abundance.
- `theme_zg` - ggplot2 theme optimized for figures.
- `color_scheme` - Ten built-in color ramps.
- `auto_signif_test` - Automatic multiple comparison.
- `compare_plot` - Quick comparison plot generator.

Author(s)

Ningqi Wang

`.DollarNames.LorMe` *Tab completion for LorMe $ access*

Description

enable tab completion when using $ with LorMe objects, including both slot names and keys in the 'data' slot.

Usage

```
## S3 method for class 'LorMe'
.DollarNames(x, pattern = "")
```

Arguments

`x`	A LorMe object
`pattern`	Character: Prefix pattern for matching

Value

Character vector of valid keys matching the pattern

`.LorMe_opts` *Global options for LorMe pipeline analysis*

Description

Global options for LorMe pipeline analysis

Usage

```
.LorMe_opts
```

Format

An object of class `list` of length 11.

Examples

```
getOption("LorMe")
```

`.onLoad` *Accessed by* `getOption("LorMe")` *or* `LorMe_options()`

Description

Accessed by `getOption("LorMe")` or `LorMe_options()`

`add_tree` *Attach phylogenetic tree to the LorMe object*

Description

Attach phylogenetic tree to the LorMe object

Usage

```
add_tree(taxobj, tree)
```

Arguments

`taxobj`	One LorMe object. Should be generated from `tax_summary`
`tree`	The phylogenetic tree. Recommanded to load by `read.tree` or `phy_tree`

Value

A LorMe object with phylogenetic tree

Examples

```
data("Two_group")
```

`Alpha_diversity_calculator`
Calculate alpha diversity based on tax summary object

Description

Calculate alpha diversity for each sample

Usage

```
Alpha_diversity_calculator(taxobj, taxlevel, prefix = "")
```

Arguments

`taxobj`	Configured tax summary objects.See in `object_config`.
`taxlevel`	taxonomy levels used for visualization.Must be one of c("Domain","Phylum","Class","Order","Family","
`prefix`	A character string as prefix of diversity index. Default:""

Value

'Alpha_diversity_calculator' returns alpha-diversity of each sample in format of column table (dataframe) combined with group information in meta file.

Examples

```
###data preparation####
data("Two_group")
require(ggplot2)

###analysis####
Alpha_results<- Alpha_diversity_calculator(taxobj = Two_group,taxlevel = "Base")

#Check data frame contained alpha diversity
head(Alpha_results$alphaframe,5)

#Check contained statistics and plot list
names(Alpha_results$plotlist)

#Check statistics for Shannon
Alpha_results$plotlist$Plotobj_Shannon$Statistics

#Extract plot for Shannon
Alpha_results$plotlist$Plotobj_Shannon$Barplot
Alpha_results$plotlist$Plotobj_Shannon$Boxplot
Alpha_results$plotlist$Plotobj_Shannon$Violinplot
```

`Alpha_diversity_calculator2`

Calculate alpha diversity based on tax summary object or dataframe table

---

Description

Calculate alpha diversity of each sample

Usage

```
Alpha_diversity_calculator2(
  taxobj = NULL,
  taxlevel = NULL,
  prefix = "",
  input,
  inputformat,
  reads
)
```

Arguments

`taxobj`	tax summary objects computed by `tax_summary`. Default:NULL.
`taxlevel`	taxonomy levels used for visualization.Must be one of c("Domain","Phylum","Class","Order","Family","
`prefix`	A character string as prefix of diversity index. Default:""
`input`	Reads or relative abundance of OTU/Taxa/gene data frame,see details in inputformat. (Useless when taxobj is set).
`inputformat`	(Useless when taxobj is set) 1:data frame with first column of OTUID and last column of taxonomy 2:data frame with first column of OTUID/taxonomy 3:data frame of all numeric
`reads`	If the input data frame were from reads table or not(relative abundance table).(Useless when taxobj is set).

Value

when tax taxobj is set, returns column table with group information combined with for alpha-diversity of each sample,else returns data frame for alpha-diversity of each sample

Note

1.When input data frame is in relative abundance table,Chao and ACE are not available

Author(s)

Wang Ningqi 2434066068@qq.com

Examples

```
### Data preparation ####
data(testotu)
groupinformation <- data.frame(
  group = c(rep("a", 10), rep("b", 10)),
  factor1 = rnorm(10),
  factor2 = rnorm(mean = 100, 10),
  subject = factor(c(1:10, 1:10))
)
```

```
# Summary OTU table into genus table and phylum table
testtax_summary <- tax_summary(
  groupfile = groupinformation,
  inputtable = testotu[, 2:21],
  reads = TRUE,
  taxonomytable = testotu[, c(1, 22)]
)

### Use taxsummary object as input ###
Alpha <- Alpha_diversity_calculator2(
  taxobj = testtax_summary,
  taxlevel = "Base"
)
head(Alpha)

# In genus level
Alpha <- Alpha_diversity_calculator2(
  taxobj = testtax_summary,
  taxlevel = "Genus",
  prefix = "Genus"
)
head(Alpha)

### Input dataframe from reads table ###
Alpha <- Alpha_diversity_calculator2(
  input = testotu,
  prefix = "Bacterial",
  inputformat = 1,
  reads = TRUE
)

### Input dataframe from relative abundance table ###
if (!require(magrittr)) install.packages("magrittr")
library(magrittr)
Alpha <- Filter_function(
  input = testotu,
  threshold = 0,
  format = 1
) %>%
  Alpha_diversity_calculator2(
    input = .,
    prefix = "Bacterial",
    inputformat = 1,
    reads = FALSE
  )
head(Alpha)
```

`anova_report` *Print Analysis of Variance report*

Description

Print Analysis of Variance report

Usage

```
anova_report(
  data,
  treatment_col,
  value_col,
  prior = FALSE,
  comparison_method = "Auto",
  equally_rep = TRUE,
  report = TRUE
)
```

Arguments

`data`	Data frame containing the treatment, value and other information.
`treatment_col`	Numeric indicating where treatment locates (column number) in data.
`value_col`	Numeric indicating where treatment value (column number) in data.
`prior`	logical. Whether conducted prior comparisons.
`comparison_method`	Default would automaticly choose method. Method of multiple comparison,must be one of "SNK", "Tukey", "bonferroni","LSD" or "Scheffe".
`equally_rep`	Logical. Whether all treatments have same number of replication.
`report`	Logical. If print report to console. Default:TRUE

Value

anova_report returns list of:

1)basic data description

2)ANOVA model

3)summary of ANOVA model

4)model of multiple comparison

5)difference of multiple comparison

6)letters of multiple comparison, which could be use for visualization.

Examples

```
{
  #' Data loading from 'agricolae' package
  data("cotton", package = "agricolae")

  #' ANOVA report with default settings
  anova_results <- anova_report(
    data = cotton,
```

```
    treatment_col = 3,
    value_col = 5
  )
  ## Here returns NULL because no significance among groups

  ## To conduct prior comparisons
  anova_results <- anova_report(
    data = cotton,
    treatment_col = 3,
    value_col = 5,
    prior = TRUE
  )

  ## Here found no difference among groups, thus change to a more sensitive method
  ## (maybe illegal, but only as an example)
  anova_results <- anova_report(
    data = cotton,
    treatment_col = 3,
    value_col = 5,
    prior = TRUE,
    comparison_method = "LSD"
  )

  #' Data loading 'iris' dataset
  data("iris")

  #' ANOVA report for 'iris' dataset
  anova_results <- anova_report(
    data = iris,
    treatment_col = 5,
    value_col = 2
  )

  ### Extract return

  ### Basic data description
  print(anova_results$basicdata)

  ### ANOVA model
  print(anova_results$anova_model)

  ### Summary of ANOVA model
  print(anova_results$anova_summary)

  ### Model of multiple comparison
  print(anova_results$multiple_comparison_model)

  ### Difference of multiple comparison
  print(anova_results$comparison_results)

  ### Letters of multiple comparison, which could be used for visualization
  print(anova_results$comparison_letters)
}
```

`auto_signif_test` *Automatic significance test*

Description

Automatically conduct significance testing

Usage

```
auto_signif_test(
  data,
  treatment_col,
  value_col,
  paired,
  subject_col,
  prior = FALSE,
  comparison_method = NULL,
  equally_rep = TRUE,
  output = "console",
  output_dir = "./",
  filename = "auto_signif_test",
  report = TRUE
)
```

Arguments

`data`	Data frame containing the treatment, value and other information.
`treatment_col`	Numeric indicating where treatment locates (column number) in data.
`value_col`	Numeric indicating where treatment value (column number) in data.
`paired`	Logical indicating whether you want a paired t-test.
`subject_col`	Only meaningful when Pair is ture. Numeric indicating where subject of treatment (column number) in data.
`prior`	logical. Whether conducted prior comparisons.
`comparison_method`	Character string. Only use for more than 2 treatment. Default would automatically choose method. Method of multiple comparison,must be one of "SNK", "Tukey", "bonferroni","LSD" or "Scheffe".
`equally_rep`	Logical indicating Whether all treatments have same number of replication.
`output`	A character string indicating output style. Default: "console", which print the report in console. And "file" is available to output report into text-file.
`output_dir`	Default:"./". Available only when output="file". The direction of output file.
`filename`	A character string indicating file name of output file. Only work when output set as 'file'.
`report`	Logical. If print report to console. Default:TRUE

Value

auto_signif_test returns results of significant test and print report in console or file. See details in example.

See results return in `t_test_report`, `wilcox_test_report`, `anova_report`, `kruskal_report`.

Note

1.when choose output="file", once caused error that terminate the program, use 'sink()' to end the written of exist files.

2.Please confirm your data is in format of dataframe, else may cause bug! (e.g. Do not use 'read.xlsx' to load data into tibble format)

Examples

```
### Here shows different types of experimental design ###
data("cotton", package = "agricolae")

### Two randomly designed groups ###
sig_results <- auto_signif_test(
  data = cotton,
  treatment_col = 1,
  value_col = 5
)

### Two paired design groups ###
sig_results <- auto_signif_test(
  data = cotton,
  treatment_col = 1,
  value_col = 5,
  paired = TRUE,
  subject_col = 2
)

### More than two randomly designed groups ###
sig_results <- auto_signif_test(
  data = cotton,
  treatment_col = 2,
  value_col = 5
)
head(sig_results)  # Check outputs

### Conduct prior comparisons ###
sig_results <- auto_signif_test(
  data = cotton,
  treatment_col = 2,
  value_col = 5,
  prior = TRUE
)
head(sig_results)  # Check outputs
print(sig_results$basicdata)  # Check statistical summary
print(sig_results$anova_model)  # Extract ANOVA model
```

```
print(sig_results$anova_summary)  # Check ANOVA summary
print(sig_results$multiple_comparison_model)  # Extract multiple comparison model
print(sig_results$comparison_results)  # Check between-group comparison
print(sig_results$comparison_letters)  # Check letters (can be used in visualization)

## Change multiple comparison method (maybe not illegal!!)
sig_results <- auto_signif_test(
  data = cotton,
  treatment_col = 2,
  value_col = 5,
  prior = TRUE,
  comparison_method = "LSD"
)
head(sig_results)  # Check outputs
print(sig_results$comparison_letters)  # Note that letters become different
```

`circulation_lm` *Circulation of fitting Linear Models*

Description

Using circulation to fit linear models between one dependent variable and series of independent variable

Usage

```
circulation_lm(y, xframe, margin)
```

Arguments

`y`	Dependent variable
`xframe`	Matrix or data frame of independent variable
`margin`	A vector of 1 or 2 indicates arrangement of xframe. 1:by rows 2:by columns

Details

if row names(for margin 1) and column names(for margin 2) are not given, ID column of return data frame will be row/column numbers.

Value

Data frame contains lm statistics of all Independent Variable

Note

Other arguments used in function lm were set as default. See in `lm`.

Author(s)

Wang Ningqi2434066068@qq.com

Examples

```
data(testotu)

###using margin 1, arrange by rows##
dep=testotu[1,2:21]
in_dep=testotu[-1,2:21]
lm_stat<-circulation_lm(y = dep,xframe = in_dep,margin = 1)
lm_stat

###using margin 2, arrange by column##
dep=testotu[,2]
in_dep=testotu[,3:21]
lm_stat<-circulation_lm(y = dep,xframe = in_dep,margin = 2)
lm_stat
```

`color_scheme` *Get color scheme*

Description

color_scheme() can generate color scheme from nine color scheme database and expand into colorRamp

Usage

```
color_scheme(Plan, expand = NULL, names = NULL, show = TRUE)
```

Arguments

`Plan`	Character, 'Plan1' to 'Plan10' are optional.
`expand`	Numeric, default:NULL. Numeric indicating numbers to expand color scheme into colorRamp
`names`	Character string. Names to assign for color scheme.
`show`	Logical. If show assigned color in plot panel. Default:TRUE.

Value

If parameter 'names' is not given, 'color_scheme' returns character string including color scheme.When 'names' is set,'color_scheme' returns named vector of color scheme.

Note

1.Parameter 'names' is strongly recommended to assign for fixed color scheme, see details in ggplot::scale_color_manual

Examples

```
### Commonly used example ###
my_color <- color_scheme(
  Plan = "Plan1",
  names = c("Treatment1", "Treatment2")
)

### Generate colorRamp still based on 'Plan1'
my_color <- color_scheme(
  Plan = "Plan1",
  expand = 4,
  names = c("Treatment1", "Treatment2", "Treatment3", "Treatment4")
)

### View color scheme from plan1 to plan10 in 'Plots' interface ###
color_scheme(Plan = "Plan1")
color_scheme(Plan = "Plan2")
color_scheme(Plan = "Plan3")
color_scheme(Plan = "Plan4")
color_scheme(Plan = "Plan5")
color_scheme(Plan = "Plan6")
color_scheme(Plan = "Plan7")
color_scheme(Plan = "Plan8")
color_scheme(Plan = "Plan9")
color_scheme(Plan = "Plan10")
```

`combine_and_translate` *Combine data for visualization*

Description

Combine group information and index into data frame for visualization(scatter, bar plot, alluvial,box plot etc.).

Usage

```
combine_and_translate(inputframe, groupframe, itemname, indexname, inputtype)
```

Arguments

`inputframe`	Data frame of index ,sample ID in column,requires all numeric(e.g. result from Alpha_diversity_calculator or Top_taxa function)
`groupframe`	Data frame of group information(and other abiotic/geographic factors)
`itemname`	A character string of your inputframe itemname
`indexname`	A character string of your inputframe indexname
`inputtype`	If sample ID were in row and index in column in inputframe.

Value

key-value pairs data frame

Author(s)

Wang Ningqi2434066068@qq.com

Examples

```
{
  require(magrittr)
  data(testotu)

  ## Data preparation ##
  Alpha <- Alpha_diversity_calculator2(
    input = testotu,
    prefix = "Bacterial",
    inputformat = 1,
    reads = TRUE
  )

  topotu <- data.frame(
    Top_taxa(
      input = testotu,
      n = 10,
      inputformat = 1,
      outformat = 1
    )[, -1],
    row.names = paste0(rep("otu", 11), 1:11)
  )

  groupinformation1 <- data.frame(
    group = c(rep("a", 10), rep("b", 10)),
    factor1 = rnorm(10),
    factor2 = rnorm(mean = 100, 10)
  )

  ### Use inputtype = FALSE ###
  head(Alpha)
  combine_and_translate(
    Alpha, groupinformation1,
    itemname = "Alpha", indexname = "index",
    inputtype = FALSE
  )

  ### Use inputtype = TRUE ###
  head(topotu)
  combine_and_translate(
    topotu, groupinformation1,
    itemname = "OTU", indexname = "reads",
    inputtype = TRUE
  )
```

```
}
```

`community_plot` *Generate Community Composition Plot Based on Tax_summary Object*

Description

Microbial community composition visualization in format of barplot, areaplot and alluvialplot

Usage

```
community_plot(
  taxobj,
  taxlevel,
  n = 10,
  palette = "Spectral",
  nrow = NULL,
  rmprefix = NULL
)
```

Arguments

`taxobj`	Configured tax summary objects.See in `object_config`.
`taxlevel`	Character. taxonomy levels used for visualization.Must be one of c("Domain","Phylum","Class","Order","
`n`	Numeric. Top n taxa remained according to relative abundance. Default:10
`palette`	Character. Palette for visualization,default:"Spectral",recommended to use "Paired" for more than 15 tax.
`nrow`	Numeric. Number of rows when wrap panels,default:NULL.
`rmprefix`	Numeric. Removed prefix character in taxonomy annotation.Default:NULL. See details in example.

Value

community_plot2 returns three ggplot objects, two data frame used in visualization and one character of filled mapping colors

Author(s)

Wang Ningqi2434066068@qq.com

Examples

```
{
  require(magrittr)
  ### Data preparation ###
  data("Two_group")

  ## Use taxonomy summary objects
  phylum10 <- community_plot(
    taxobj = Two_group,
    taxlevel = "Phylum",
    n = 10,
    rmprefix = "p__"
  )

  phylum10$barplot  # Check bar plot
  phylum10$areaplot  # Check area plot
  phylum10$alluvialplot  # Check alluvial plot

  phylum10$Top10Phylum %>% head(10)  # Check top taxa data frame
  phylum10$Grouped_Top10Phylum %>% head(10)  # Check grouped top taxa data frame
  print(phylum10$filled_color)  # Check mapping colors

  # Double facet
  data("Facet_group")

  # Using palette by default
  phylum10 <- community_plot(
    taxobj = Facet_group,
    taxlevel = "Phylum",
    n = 10,
    rmprefix = " p__"
  )
  phylum10$barplot
  phylum10$areaplot
  phylum10$alluvialplot

  # Another example
  genus20 <- community_plot(
    taxobj = Facet_group,
    taxlevel = "Genus",
    n = 20,
    palette = "Paired",
    rmprefix = " g__"
  )
  genus20$alluvialplot
}
```

`compare_plot` *Comparison plot generator This function help generate comparsion plot including bar plot, box plot, and violin plot*

Description

Comparison plot generator This function help generate comparsion plot including bar plot, box plot, and violin plot

Usage

```
compare_plot(
  inputframe,
  treat_location,
  value_location,
  aes_col = NULL,
  point = TRUE,
  facet_location = NULL,
  ylab_text = NULL
)
```

Arguments

`inputframe`	A data frame contain information for visualization.
`treat_location`	Numeric. Treatment column number in inputframe.
`value_location`	Numeric. Value column number in inputframe.
`aes_col`	Named character string, default:NULL. A set of aesthetic character to map treatment to.
`point`	Logical. If draw point on bar, box and violin plot. Default:TRUE.
`facet_location`	Numeric, default:NULL. Facet column number in inputframe.
`ylab_text`	Character. Text for y axis.

Value

A list contained plot and statistics

Examples

```
data("iris")
results=compare_plot(inputframe=iris,treat_location=5,
                     value_location=1,ylab_text = "Sepal Length")

#Check statistics
results$Statistics
#Extract plot
results$Barplot
results$Boxplot
results$Violinplot


iris$Treat2=rep(c(rep("A",25),rep("B",25)),3)

results=compare_plot(inputframe=iris,treat_location=5,
```

```
                     value_location=1,facet_location = 6,
                     ylab_text = "Sepal Length")

#Check statistics
results$Statistics
#Extract plot
results$Barplot
results$Boxplot
results$Violinplot
#Extract combined plot
results$All_Barplot
results$All_Boxplot
results$All_Violinplot
```

`Deseq_analysis` *Deseq Analysis Function*

Description

This function performs a differential expression analysis using the DESeq2 package. It is designed to work with microbiome data and can handle paired or non-paired samples.

Usage

```
Deseq_analysis(
  taxobj,
  taxlevel,
  comparison = NULL,
  cutoff,
  control_name,
  paired = FALSE,
  subject = NULL
)
```

Arguments

`taxobj`	Configured tax summary objects.See in `object_config`.
`taxlevel`	The taxonomic level for the analysis.Must be one of c("Domain","Phylum","Class","Order","Family","Ge
`comparison`	A vector of conditions to compare. Default: NULL, all unique conditions are compared (only for Two groups).
`cutoff`	The log2 fold change cutoff for considering as differential taxon.
`control_name`	Character. The name of the control group for the comparison.
`paired`	Logical. Should the samples be treated as paired? Default: False
`subject`	Optional. The subject identifier for paired samples. Default: Null

Value

A data frame with the results of the differential expression analysis.

Note

1. Regulation is judged by cutoff of q-value(adjust p value).Detail see in `DESeq`
2. For more than two groups in taxobj, the 'comparison' must be assigned.
3. The function requires the 'DESeq2', 'S4Vectors', and 'tibble' packages.

Author(s)

Wang Ningqi 2434066068@qq.com

See Also

`DESeqDataSetFromMatrix`, `DESeq`, `DataFrame`, `as_tibble`

Examples

```
if (requireNamespace("DESeq2", quietly = TRUE) &&
    requireNamespace("S4Vectors", quietly = TRUE) &&
    requireNamespace("tibble", quietly = TRUE)) {

    ### Data preparation ###
    data("Two_group")

    ### Deseq analysis ###
    deseq_results <- Deseq_analysis(
      taxobj = Two_group,
      taxlevel = "Genus",
      cutoff = 1,
      control_name = "Control"
    )

    # Visualization of volcano plot ##
    volcano_plot <- volcano_plot(
      inputframe = deseq_results,
      cutoff = 1,
      aes_col = Two_group$configuration$treat_col
    )
    volcano_plot$FC_FDR
    volcano_plot$Mean_FC

    # Visualization of Manhattan plot ##
    manhattan_object <- manhattan(
      inputframe = deseq_results,
      taxlevel = "Phylum",
      control_name = "Control",
      mode = "most",
      top_n = 10,
      rmprefix = "p__"
```

```
    )
    manhattan_object$manhattan  # Tradition manhattan plot
    manhattan_object$manhattan_circle  # Circular manhattan plot

    # For object with more than two groups
    ### Data preparation ###
    data("Three_group")

    # Specific comparison
    deseq_results_BFCF <- Deseq_analysis(
      taxobj = Three_group,
      taxlevel = "Genus",
      comparison = c("BF", "CF"),
      cutoff = 1,
      control_name = "CF"
    )
    volcano_plot <- volcano_plot(
      inputframe = deseq_results_BFCF,
      cutoff = 1,
      aes_col = Three_group$configuration$treat_col
    )
    volcano_plot$FC_FDR
  } else {
    message(
      "The 'DESeq2', 'S4Vectors', and/or 'tibble' package(s) are not installed. ",
      "Please install them to use all features of Deseq_analysis."
    )
  }
```

`Deseq_analysis2` *Deseq analysis*

Description

Deseq analysis

Usage

```
Deseq_analysis2(inputframe, condition, cutoff, control_name, paired, subject)
```

Arguments

`inputframe`	Otu/gene/taxa table with all integer numeric variables.Rownames must be Otu/gene/taxa names,colnames must be sample names with control in front and treatment behind. Reads table is recommended.
`condition`	A character string which indicates group of samples
`cutoff`	threshold of log2(Foldchange).Detail see in `DESeq`
`control_name`	A character indicating the control group name

`paired`	Logical to determine if paired comparision would be used. TRUE or FALSE.
`subject`	A character string which indicates paired design of samples

Value

Statistics dataframe of all otu/gene/taxa

Note

1. Inputframe must be all integer numeric variables without NA/NAN/inf! In case your data is not an integer one,a practical method is to multiply them in equal proportion(eg. x 1e6) then round them into integer
2. Regulation is judged by cutoff of qvalue(adjust p value).Detail see in `DESeq`
3. Set cutoff as 1 is recommened.In case of too few taxa(eg. Phylum level deseq),cutoff can be set to 0.
4. if control_name is not given, the control group will be set according to ASCII
5. The function requires the 'DESeq2', 'S4Vectors', and 'tibble' packages.

Author(s)

Wang Ningqi 2434066068@qq.com

See Also

`DESeqDataSetFromMatrix`, `DESeq`, `DataFrame`, `as_tibble`

Examples

```
{
  ### Data preparation ###
  data(testotu)
  rownames(testotu) <- testotu[, 1]
  inputotu <- testotu[, -c(1, ncol(testotu))]
  head(inputotu)
  group <- c(rep("a", 10), rep("b", 10))

  ### DESeq analysis ###
  if (requireNamespace("DESeq2", quietly = TRUE) &&
    requireNamespace("S4Vectors", quietly = TRUE) &&
    requireNamespace("tibble", quietly = TRUE)) {
    Deseqresult <- Deseq_analysis2(
      inputframe = inputotu,
      condition = group,
      cutoff = 1,
      control_name = "b"
    )

    ### Paired DESeq analysis ###
    subject <- factor(c(1:10, 1:10))
    Deseqresult <- Deseq_analysis2(
```

```
      inputframe = inputotu,
      condition = group,
      cutoff = 1,
      control_name = "b",
      paired = TRUE,
      subject = subject
    )
  }
}
```

`differential_bar` *Generate Differential Bar Plot and Error bar Plot*

Description

Generate Differential Bar Plot and Error bar Plot

Usage

```
differential_bar(
  taxobj,
  taxlevel,
  comparison = NULL,
  rel_threshold = 0.005,
  anno_row = "taxonomy",
  aes_col = NULL,
  limit_num = NULL
)
```

Arguments

`taxobj`	Configured tax summary objects.See in `object_config`.
`taxlevel`	Taxonomy levels used for visualization. Must be one of c("Domain","Phylum","Class","Order","Family",
`comparison`	A vector of conditions to compare. Default: NULL, all unique conditions are compared (only for Two groups).
`rel_threshold`	Threshold filtering taxa for differential analysis. Default:0.005
`anno_row`	Default: ’taxonomy’. Rownames for visualization. Options are ’taxonomy’ for showing taxonomic information and ’ID’ for showing taxonomic ID.
`aes_col`	A named vector of colors to be used in the plots.
`limit_num`	Numeric. The maximum number of significant results to display. Default: NULL, showing all differential taxa.

Value

A list containing the bar plot, source data for the bar plot, difference plot, and source data for the difference plot.

Note

The differential analysis is performed using two-sided Welch's t-test. The p-values are adjusted using the 'BH' (i.e., FDR) method.

Examples

```
{
  # Data preparation
  data("Two_group")

  # Simple mode
  diff_results <- differential_bar(
    taxobj = Two_group,
    taxlevel = "Genus"
  )
  print(diff_results$Barplot)  # Print Barplot
  head(diff_results$Barplot_sourcedata)  # Show source data of barplot
  print(diff_results$Differenceplot)  # Print Differential errorbar plot
 head(diff_results$Differenceplot_sourcedata)  # Show source data of Differential errorbar plot

  require(patchwork)
  diff_results$Barplot|diff_results$Differenceplot
  # Displaying ID
  diff_results <- differential_bar(
    taxobj = Two_group,
    taxlevel = "Base",
    anno_row = "ID"
  )
  print(diff_results$Barplot)

  # Threshold adjustment
  diff_results <- differential_bar(
    taxobj = Two_group,
    taxlevel = "Base",
    rel_threshold = 0.001
  )
  print(diff_results$Barplot)

  # Limit the displaying number
  diff_results <- differential_bar(
    taxobj = Two_group,
    taxlevel = "Base",
    rel_threshold = 0.001,
    limit_num = 10
  )
  print(diff_results$Barplot)

  # For object with more than two groups
  # Data preparation
  data("Three_group")

  # Specific comparison
```

```
    Three_group_col <- Three_group$configuration$treat_col
    diff_results <- differential_bar(
      taxobj = Three_group,
      taxlevel = "Genus",
      comparison = c("BF", "CF"),
      aes_col = Three_group_col
    )
    print(diff_results$Barplot)
  }
```

`Dimension_reduction` *Dimension_reduction: PCA, PCOA, and NMDS Analysis*

Description

Performs dimension reduction analysis using PCA, PCOA, or NMDS.

Usage

```
Dimension_reduction(inputframe, group, format)
```

Arguments

`inputframe`	An OTU/gene/taxa table with all numeric variables and no NA/NAN/inf values.
`group`	Group information with the sample order the same as in inputframe.
`format`	The format of analysis: 1 for PCA, 2 for PCOA, 3 for NMDS.

Value

A list containing data frames and other statistics for dimension reduction analysis.

Note

Inputframe should be a numeric matrix without NA/NAN/inf values.

The row names of inputframe should be set as OTU/gene/taxa annotations for further analysis.

The results are combined into a list for output. Use `as.data.frame(result[[1]])` to extract the data frame, and `$result$` to extract other statistics. See examples for details.

Author(s)

Wang Ningqi 2434066068@qq.com

Examples

```
### Data preparation ###
data(testotu)
rownames(testotu) <- testotu[, 1]
inputotu <- testotu[, -c(1, ncol(testotu))]
head(inputotu)

groupinformation1 <- data.frame(
  group = c(rep("a", 10), rep("b", 10)),
  factor1 = rnorm(10),
  factor2 = rnorm(mean = 100, 10)
)

### PCA ###
PCAresult <- Dimension_reduction(inputotu, groupinformation1, 1)
PCAframe <- PCAresult$outframe   # Extract data for visualization
head(PCAresult$data.pca$rotation,5)  # OTU coordinates

### PCOA ###
PCOAresult <- Dimension_reduction(inputotu, groupinformation1, 2)
PCOAframe <- PCOAresult$outframe  # Extract data for visualization
head(PCOAresult$PCOA$values,2)  # Explanation of first two axis

### NMDS ###
NMDSresult <- Dimension_reduction(inputotu, groupinformation1, 3)
NMDSframe <- NMDSresult$outframe  # Extract data for visualization
# Here we got a warning of `stress is (nearly) zero: you may have insufficient data`,
# so make sure you have sufficient data for NMDS
print(NMDSresult$NMDSstat$stress)  # Extract stress of NMDS
```

`Facet_group` *Tax summary object with Facet 2x2 Groups*

Description

Enraptured summary object with facet 2x2 Groups.Configuration has been assigned.

Usage

```
Facet_group
```

Format

Tax summary object with configuration

`Filter_function` *Filter OTU/ASV/metagenomic profile/gene profile by threshold*

Description

Sequenced data of taxonomy&gene still remains some sequencing error which we needed to be wiped off before analyzing. Here we provide function including four formats to wipe them clean.

Usage

```
Filter_function(input, threshold, format, report = TRUE)
```

Arguments

`input`	Data frame of absolute abundance of standard OTU table,with the first column of OTUID and the final column of taxonomy annotation. If your data frame is gene table or not a standard OTU table, please manually transformed into a standard input data frame.
`threshold`	threshold of filter.Relative abundance for format 1 and 4, reads number for format 2, sample size for format 3
`format`	1:filter OTU/gene below overall-sample relative abundance threshold(<) 2:filter OTU/gene below overall-sample reads threshold(<) 3:filter OTU/gene reads 0 over threshold sample size(>) 4:filter OTU/gene below relative abundance threshold in each sample(<)
`report`	Logical. If print report to console. Default:TRUE

Value

Dataframe of OTU/gene in format of absolute abundacne(reads) or relative abundance(%)

Author(s)

Wang Ningqi

Examples

```
### Data frame with absolute abundance (reads)###
### And first column of OTUID and last column of taxonomy ###
data(testotu)

#### If your data frame does not contain the OTUID column or taxonomy column,
#### you can add a simulated column to fit the input format like testotu ##

### 1. Filter OTU with total relative abundance below 0.0001###
filtered_otu <- Filter_function(
  input = testotu,
  threshold = 0.0001,
```

```
  format = 1
)

### 2. Filter OTU with total reads below 20 ###
filtered_otu <- Filter_function(
  input = testotu,
  threshold = 20,
  format = 2
)

### 3. Filter OTU reads 0 over (>=) 11 samples ###
filtered_otu <- Filter_function(
  input = testotu,
  threshold = 11,
  format = 3
)

### 4. Filter OTU with relative abundance below 0.0001 in each sample ###
filtered_otu <- Filter_function(
  input = testotu,
  threshold = 0.0001,
  format = 4
)
```

`indicator_analysis` *Indicator Analysis*

Description

Performs the indicator analysis based on taxonomic summary object

Usage

```
indicator_analysis(taxobj, taxlevel, func = "r.g", reads = FALSE)
```

Arguments

`taxobj`	Configured tax summary objects.See in `object_config`.
`taxlevel`	taxonomy levels used for visualization.Must be one of c("Domain","Phylum","Class","Order","Family","
`func`	Default: "r.g".The function to use for the indicator analysis, see in `multipatt`
`reads`	A logical value indicating whether the input data is in terms of raw reads (TRUE) or relative abundance (FALSE)

Value

A data frame with the results of the indicator analysis, including adjusted p-values, tags and taxonomic information.

Note

This function depends on the following packages: indicspecies, permute. These packages are not automatically loaded and should be installed before using this function.

See Also

`multipatt`, `how`

Examples

```
data("Two_group")
if (requireNamespace("indicspecies", quietly = TRUE) &&
    requireNamespace("permute", quietly = TRUE)) {
  set.seed(999)
  indicator_results <- indicator_analysis(
    taxobj = Two_group,
    taxlevel = "Genus"
  )
  head(indicator_results)
}
```

`kruskal_report` *Print Kruskal-Wallis Rank Sum Test report*

Description

Print Kruskal-Wallis Rank Sum Test report

Usage

```
kruskal_report(
  data,
  treatment_col,
  value_col,
  prior = FALSE,
  comparison_method = "Auto",
  equally_rep = TRUE,
  report = TRUE
)
```

Arguments

`data`	Data frame containing the treatment, value and other information.
`treatment_col`	Numeric indicating where treatment locates (column number) in data.
`value_col`	Numeric indicating where treatment value (column number) in data.
`prior`	logical. Whether conducted prior comparisons.

`comparison_method`	Default would automaticly choose method. Method of multiple comparison,must be one of "SNK" or "Tukey".
`equally_rep`	Logical. Whether all treatments have same number of replication.
`report`	Logical. If print report to console. Default:TRUE

Value

kruskal_report returns list with

1)basic data description

2)summary of Kruskal-Wallis Rank Sum Test

3)model of multiple comparison

4)difference of multiple comparison

5)letters of multiple comparison, which could be use for visualization.

Examples

```
data("cotton",package ="agricolae" )
kruskal_results=kruskal_report(data = cotton,treatment_col =3,value_col = 5)
##here returns NULL because no significance among groups

##to conduct prior comparisons.
kruskal_results=kruskal_report(data = cotton,treatment_col =3,value_col = 5,prior = TRUE)


data("iris")
kruskal_results=kruskal_report(data = iris,treatment_col = 5,value_col = 2)

###extract return##

###basic data description
kruskal_results$basicdata

###summry of Kruskal-Wallis Rank Sum Test
kruskal_results$Kruskal_Wallis_summary

###model of multiple comparision
kruskal_results$muiltiple_comparision_model

###difference of multiple comparision
kruskal_results$comparision_results

###letters of multiple comparision, which could be use for visualization.
kruskal_results$comparison_letters
```

`LorMe-class` *LorMe S4 class*

Description

A container class for microbial ecology data integration, including metadata, feature tables, configuration parameters, and phylogenetic tree.

Slots

`groupfile` data.frame: Sample metadata

`data` list: Named list containing feature and taxonomy tables

`configuration` list: Analysis configuration parameters

`tree` phyloOrNULL: Phylogenetic tree (NULL if not provided, requires ape package)

Methods

`show(object)` Print summary of the LorMe object

`x$name` Access slots or data elements via $

`.DollarNames(x, pattern)` Tab completion for $ access

`LorMe_defaults` *Restore LorMe global options to factory defaults*

Description

Equivalent to `options(LorMe = .LorMe_opts)` but provides a unified entry.

Usage

```
LorMe_defaults()
```

Value

Invisible NULL

Examples

```
LorMe_defaults()   # one-click factory reset
```

`LorMe_options` *Change LorMe options*

Description

Change LorMe options

Usage

```
LorMe_options(..., level = NULL)
```

Arguments

`...`	Named parameters or "level = list" pairs.
`level`	Optional; specifies a whole-layer replacement.

Value

Modified global options (or invisible(NULL) after factory reset).

Examples

```
LorMe_options()                                        # view current
LorMe_options(beta = list(diagram = "circle"))     # whole layer
LorMe_options(level = "beta", list(diagram = "circle"))  # within layer
```

`LorMe_pipeline` *One-stop microbial analysis pipeline for LorMe objects*

Description

`LorMe_pipeline()` performs a complete microbial ecology analysis workflow for a configured **LorMe** object, including: community profiling, differential analysis, indicator species analysis, treatment-specific sub-network construction, and global network construction.

The function is modular and allows users to execute only selected steps through the `step` argument.

Usage

```
LorMe_pipeline(taxobj, step = "all")
```

Arguments

`taxobj`	A configured **LorMe** object created by `object_config`. The object must contain preprocessed taxonomy tables, metadata, and analysis configurations.
`step`	Character vector specifying which analysis modules to run. Must be one or more of: • `"all"` – run the entire pipeline (default) • `"profile"` – alpha/beta diversity and composition analysis • `"diff"` – differential abundance (DESeq2, differential barplot) • `"sub_net"` – treatment-specific subnetworks • `"all_net"` – combined/co-occurrence network across all samples

Details

The function automatically respects global options set via `LorMe_options`. These options control:

- analysis taxonomic level
- palettes and plotting parameters
- DESeq2 parameters
- network analysis thresholds

Internal failures in any analysis sub-module do **not** stop the pipeline: the corresponding output is returned as `NULL`, and the function prints a summary of failed steps on completion.

Value

A named list containing the results of all requested modules. Depending on `step`, the list may include:

`alpha_results`	Alpha diversity tables and plots
`beta_results`	Beta diversity ordinations (e.g., PCoA)
`composition_results`	Community composition barplots
`diffbar_result`	Differential barplot results
`Deseq_results`	DESeq2 results for each comparison
`Deseq_volcano`	Volcano plots for differential analysis
`Deseq_manhattan`	Manhattan plots for differential features
`indicator_results`	Indicator species results
`indic_volcano`	Volcano plot of indicator species analysis
`indic_manhattan`	Manhattan plot of indicator analysis
`sub_network_results`	Treatment-specific subnetworks
`combine_network_results`	Global co-occurrence network

Failed modules return as `NULL`.

Examples

```
## View current global analysis options
getOption("LorMe")

## Set analysis options
LorMe_options(
  global = list(Analysis_level = "Genus"),
  sub_net = list(threshold = 0.7),
  all_net = list(threshold = 0.7)
)

## Run pipeline (time-consuming)
Two_group_analysis <- LorMe_pipeline(Two_group)

## Access results:
# Alpha diversity
Two_group_analysis$alpha_results$plotlist$Plotobj_Shannon$Boxplot

# Beta diversity
Two_group_analysis$beta_results$PCoA_Plot

# Community composition
Two_group_analysis$composition_results$barplot

# Differential analysis
Two_group_analysis$Deseq_volcano$FC_FDR
Two_group_analysis$Deseq_manhattan$manhattan

# Differential barplot
library(patchwork)
Two_group_analysis$diffbar_result$Barplot |
  Two_group_analysis$diffbar_result$Differenceplot

# Subnetworks
require(magrittr)
Two_group_analysis$sub_network_results$Treatment_sub_network %>%
  network_visual()

# Combined network
Two_group_analysis$combine_network_results %>% network_visual()

## Reset to default options
LorMe_defaults()

## Get options for re-analysis
options(LorMe = Two_group_analysis$Analysis_option) # Then try the previous analysis command

## Example: three-group comparison with custom options
LorMe_options(
  global = list(
    Analysis_level = "Species",
    compare_list = c("CF_OF", "CF_BF")
```

```
  ),
  all_net = list(threshold = 0.95, method = "pearson")
)

Three_group_analysis <- LorMe_pipeline(Three_group)
```

`manhattan` *Manhattan Plot Generator*

Description

Generate Manhattan Plot base on Deseq_analysis or indicator_analysis results

Usage

```
manhattan(
  inputframe,
  taxlevel = "Phylum",
  control_name,
  mode = "all",
  top_n = NULL,
  palette = "Set1",
  select_tax = NULL,
  rmprefix = NULL
)
```

Arguments

`inputframe`	A data frame generated from `Deseq_analysis` or `indicator_analysis`
`taxlevel`	Taxonomy levels used for visualization.Must be one of c("Domain","Phylum","Class","Order","Family","
`control_name`	Character. The name of the control group for the comparison.
`mode`	The mode for selecting which taxa to plot: "all" for all taxa, "most" for the top N taxa, and "select" for specific taxa selection
`top_n`	The number of top taxa to plot when mode is set to "most"
`palette`	Character. Palette for visualization,default:"Set1".Optional palette same as 'RColorBrewer'. "Plan1" to "Plan10" were also optional,see in `color_scheme`
`select_tax`	A vector of taxa to be selected for plotting when mode is "select".
`rmprefix`	A string prefix to be removed from the taxonomic annotation.Default:NULL.

Value

a list containing the Manhattan plot, circular Manhattan plot, source data, and color assignments

Examples

```
{
  # Data preparation
  data("Two_group")

  # DESeq analysis
  deseq_results <- Deseq_analysis(
    taxobj = Two_group,
    taxlevel = "Base",
    cutoff = 1,
    control_name = "Control"
  )

  # Indicator analysis
  indicator_results <- indicator_analysis(
    taxobj = Two_group,
    taxlevel = "Genus"
  )

  # Show all with Manhattan plot
    manhattan_object <- manhattan(
      inputframe = deseq_results,
      taxlevel = "Phylum",
      control_name = "Control"
    )
    print(manhattan_object$manhattan)  # Tradition Manhattan plot
    print(manhattan_object$manhattan_circle)  # Circular Manhattan plot
    print(manhattan_object$sourcedata)  # Source data for plot
    print(manhattan_object$aes_color)  # Aesthetic color for plot

  # Top 8 Phyla with most taxon
    manhattan_object <- manhattan(
      inputframe = indicator_results,
      taxlevel = "Phylum",
      control_name = "Control",
      mode = "most",
      top_n = 8,
      palette = "Set1"
    )
    print(manhattan_object$manhattan)

  # Specific phyla
  # Top nine dominant phyla
    community <- community_plot(
      taxobj = Two_group,
      taxlevel = "Phylum",
      n = 9,
      palette = "Paired",
      rmprefix = "p__"
    )

    manhattan_object <- manhattan(
```

```
        inputframe = indicator_results,
        taxlevel = "Phylum",
        control_name = "Control",
        mode = "select",
        palette = community$filled_color,
        select_tax = names(community$filled_color),
        rmprefix = "p__"
      )
      print(manhattan_object$manhattan)
      print(manhattan_object$manhattan_circle)
  }
```

`Module_abundance` *Calculate network module abundance for each sample*

Description

Calculate network module abundance for each sample

Usage

```
Module_abundance(network_obj, No.module)
```

Arguments

`network_obj`	Network analysis results generated from `network_analysis`
`No.module`	Numeric or numeric vector of No.module

Value

A list containing module abundance in metafile and column table of corresponding data frame

Examples

```
# data preparation
data("Two_group")
## network analysis
network_results <- network_analysis(taxobj = Two_group,
                                    taxlevel = "Genus",
                                    n = 10,
                                    threshold = 0.8)
require(ggplot2)
# one module
moduleframe <- Module_abundance(network_obj = network_results, No.module = 3)
moduleframe$rowframe        # combine into metafile
moduleframe$columnframe     # column table
# statistics
moduleframe$plotlist$Plotobj_Module3$Statistics
```

```
# extract plot
moduleframe$plotlist$Plotobj_Module3$Barplot
moduleframe$plotlist$Plotobj_Module3$Boxplot
moduleframe$plotlist$Plotobj_Module3$Violinplot

# multiple modules
moduleframe <- Module_abundance(network_results, c(1, 3, 6))
moduleframe$rowframe
moduleframe$columnframe # column table can be used in ggplot visualization
# same as above to extract plots and statistics
moduleframe$plotlist$Plotobj_Module6$Barplot
```

`Module_composition` *Pie chart for network module composition*

Description

This function analyzes the composition of modules within a network object, providing a visual and data summary based on taxonomic levels.

Usage

```
Module_composition(
  network_obj,
  No.module,
  taxlevel = "Phylum",
  mode = "all",
  top_n = NULL,
  palette = "Set1",
  select_tax = NULL,
  rmprefix = NULL
)
```

Arguments

`network_obj`	Network analysis results generated from `network_analysis`
`No.module`	Numeric or numeric vector of No.module
`taxlevel`	Taxonomy levels used for visualization.Must be one of c("Domain","Phylum","Class","Order","Family","
`mode`	The mode for selecting which taxa to plot: "all" for all taxa, "most" for the top N taxa, and "select" for specific taxa selection
`top_n`	The number of top taxa to plot when mode is set to "most"
`palette`	Character. Palette for visualization,default:"Set1".See optional palette in same as 'RColorBrewer'. And "Plan1" to "Plan10" were also optional,see in `color_scheme`
`select_tax`	A vector of taxa to be selected for plotting when mode is "select".
`rmprefix`	A string prefix to be removed from the taxonomic annotation

Value

The function returns a list containing pie chart of specific module,corresponding source data and color assignments

Examples

```
#Data loading
data("Two_group")

# Network analysis
network_Two_group <- network_analysis(
   taxobj = Two_group,
   taxlevel = "Genus",
   reads = TRUE,
   n = 8,
   threshold = 0.7
 )

 # Show all taxa
 module_results <- Module_composition(
   network_obj = network_Two_group,
   No.module = c(2, 5),
   taxlevel = "Phylum"
 )
 print(module_results$Module5$Pie)
 print(module_results$Module2$Pie)  # View pie chart
 head(module_results$Module2$source_data_Module2)  # View source data for pie chart
 print(module_results$aes_color)  # Check aesthetic color

 # Show taxa with top five frequency
 module_results <- Module_composition(
   network_obj = network_Two_group,
   No.module = c(2, 5),
   taxlevel = "Phylum",
   mode = "most",
   top_n = 5
 )
 print(module_results$Module2$Pie_plot_Module2)

 # Show specific taxa
 community <- community_plot(
   taxobj = Two_group,
   taxlevel = "Phylum",
   n = 5,
   palette = "Paired"
 )  # Get top 5 dominant phyla
 top5_phyla <- names(community$filled_color)

 module_results <- Module_composition(
   network_obj = network_Two_group,
   No.module = c(2, 5),
   taxlevel = "Phylum",
```

```
    mode = "select",
    palette = community$filled_color,
    select_tax = top5_phyla
  )
  print(module_results$Module2$Pie_plot_Module2)

  # Specific taxa with no prefix 'p__'
  module_results <- Module_composition(
    network_obj = network_Two_group,
    No.module = 2,
    taxlevel = "Phylum",
    mode = "select",
    select_tax = c("Proteobacteria", "Actinobacteria")
  )
  print(module_results$Module2$Pie_plot_Module2)

  # Remove 'p__' prefix
  module_results <- Module_composition(
    network_obj = network_Two_group,
    No.module = 2,
    taxlevel = "Phylum",
    mode = "most",
    top_n = 5,
    palette = "Set2",
    rmprefix = "p__"
  )
  print(module_results$Module2$Pie_plot_Module2)
```

`nc` *Calculate Network Natural Connectivity*

Description

Calculate Network Natural Connectivity

Usage

```
nc(adj_matrix)
```

Arguments

`adj_matrix`	Adjacency data frame or matrix. Can be calculated from `network_analysis`

Value

Numeric value of natural connectivity

Examples

```
{
  ### Data preparation ###
  data("Two_group")

  ### One input network analysis ###
  network_results <- network_analysis(
    taxobj = Two_group,
    taxlevel = "Base",
    reads = FALSE,
    n = 10,
    threshold = 0.6
  )

  # Convert network results to a data frame for the adjacency matrix
  network_matrix <- as.data.frame(network_results[[3]])  # Complete adjacency matrix

  # Check initial natural connectivity
  nc_initial <- nc(network_matrix)
  print(nc_initial)  # Print the initial natural connectivity
}
```

`NC_remove` *Natural connectivity analysis*

Description

Natural connectivity analysis

Usage

```
NC_remove(input, num, seed = 1)
```

Arguments

`input`	Network adjacency matrix. Can be generated by `network_analysis`
`num`	Max number of removed nodes. Default: Automatically match max number that can be removed.
`seed`	Random seed Number to be set. Default: 1. See in `set.seed`

Value

NC_remove returns data frame with removed nodes and corresponding natural connectivity

Author(s)

Wang Ningqi2434066068@qq.com

Examples

```
{
  ### Data preparation ###
  data("Two_group")

  ### One input network analysis ###
  network_results <- network_analysis(
    taxobj = Two_group,
    taxlevel = "Base",
    reads = FALSE,
    n = 10,
    threshold = 0.6
  )

  network_matrix <- as.data.frame(network_results[[3]])  # Complete adjacency matrix

  # Check initial natural connectivity
  nc <- nc(network_matrix)

  # Conduct natural connectivity analysis
  nc_remove <- NC_remove(input = network_matrix)
  head(nc_remove)
  tail(nc_remove)

  # Set target number for natural connectivity analysis
  nc_remove <- NC_remove(input = network_matrix, num = 400)
}
```

`network_analysis` *Conduct Network analysis based on tax summary object*

Description

Conduct Network analysis based on tax summary object

Usage

```
network_analysis(
  taxobj,
  taxlevel,
  reads = FALSE,
  n,
  threshold,
  rel_threshold = 0,
  method = "spearman",
  display = TRUE
)
```

Arguments

`taxobj`	tax summary objects computed by `tax_summary`.
`taxlevel`	taxonomy levels used for analysis. Must be one of c("Domain","Phylum","Class","Order","Family","Genu
`reads`	Logical,default:FALSE. Taxonomy abundance type used in analysis.FALSE for relative abundance, TRUE for absolute abundance.
`n`	Numeric. Number of sample size indicating kept asv/otu/gene/taxa appearing. Recommended to set more than half of total sample size.
`threshold`	Numeric.Threshold of absolute correlation value (r value for pearson method and rho value for spearman method).
`rel_threshold`	Numeric.Threshold of relative abundance included in the network analysis.Default:0
`method`	Character, default: "spearman". A character indicating which correlation coefficient method to be computed. One of "pearson" or "spearman"
`display`	Logical, default:TRUE. If display a preview plot of network based on igraph. FALSE for the first attempt is recommended in case of too many vertices and edges.

Details

1. We had optimized the correlation algorithm to achieve a faster running speed. It takes less than 2 minute to calculate dataframe correlation and p value which more than 400 samples and 10000 OTUs for computer with dual Core i5 processor. However, too many vertices(>2000) or links(>10000) may slow the statistical process and visualization,so we recommend that in your first attempt,set `display` paramter as `F` to have a preview. Then you can adjust your n/threshold/method paramter to generate a suitable visualization network
2. We display a preview plot so as to adjusting your network. Generally a global figure (like we show in examples) with less than 1000 vertices and 5000 edges/links is recommended. Further more,we recommend you to output the statistics and adjacency table and use software like cytoscape or gephi for better visualization.

Value

One list contains nodes information table, adjacency column table, adjacency matrix and 'igraph' object.

Note

1. Replicates should be at least 5,more than 8 is recommend.
2. In case of too many edges/links or not a global network plot, you can stop the process immediately to provent wasting too much time.

Examples

```
{
  ### Data preparation ###
  data("Two_group")
  set.seed(999)
```

```
  ## Analysis
  network_results <- network_analysis(
    taxobj = Two_group,
    taxlevel = "Genus",
    n = 10,
    threshold = 0.8
  )

  # Nodes information table
  network_nodes <- network_results$Nodes_info
  head(network_nodes)

  # Adjacency table
  network_adjacency <- network_results$Adjacency_column_table
  head(network_adjacency)

  # Complete adjacency matrix
  network_matrix <- network_results$Adjacency_matrix
  print(network_matrix[1:10, 1:10])

  # igraph object
  igraph_object <- network_results$Igraph_object
  network_stat(igraph_object)  # In case you want to see statistics again
  # or do other analysis based on igraph.
}
```

`network_analysis2` *Conduct Network analysis*

Description

A convenient and fast network analysis function, with output results suitable for cytoscape and gephi

Usage

```
network_analysis2(
  input,
  inputtype,
  n,
  threshold,
  method = "spearman",
  display = TRUE,
  input2,
  input2type
)
```

Arguments

`input`	Input dataframe with otu/gene/taxa in row and sample ID in column,at least 5 replicates(more than 8 replicates are recommened).
`inputtype`	Input dataframe type 1:dataframe with first column of OTUID and last column of taxonomy 2:dataframe with first column of OTUID/taxonomy 3:dataframe of all numeric
`n`	Only keep otu/gene/taxa appearing in n sample size
`threshold`	Threshold of correlation r value
`method`	A character string indicating which correlation coefficient is to be computed. One of "pearson" or "spearman"
`display`	If display a preview plot of network based on igraph. FALSE for the first attempt is recommended in case of too many vertices and edges.
`input2`	A second input data frame with otu/gene/taxa in row and sample ID in column. Default:NULL
`input2type`	The second input data frame type. Details the same as above. Default:NULL

Details

1. We had optimized the correlation algorithm to achieve a faster running speed. It takes less than 2 minute to calculate dataframe correlation and p value which more than 400 samples and 10000 OTUs for computer with dual Core i5 processor. However, too many vertices(>2000) or links(>10000) may slow the statistical process and visualization,so we recommend that in your first attempt,set `display` paramter as `F` to have a preview. Then you can adjust your n/threshold/method paramter to generate a suitable visualization network
2. We display a preview plot so as to adjusting your network. Generally a global figure (like we show in examples) with less than 1000 vertices and 5000 edges/links is recommended. Further more,we recommend you to output the statistics and adjacency table and use software like cytoscape or gephi for better visualization.

Value

One list contains a statistics table of network vertices/nodes and an adjacency table. One preview plot of network in the plot interface and an igraph object(named `igraph1`) in global environment.

Note

1. Replicates should be at least 5,more than 8 is recommend.
2. In case of too many edges/links or not a global network plot, you can stop the process immediately to provent wasting too much time.

Author(s)

Wang Ningqi 2434066068@qq.com

Examples

```
{
  ### Data preparation ###
  data(testotu)
  rownames(testotu) <- testotu[, 1]
  inputotu <- testotu[, -c(1, ncol(testotu))]
  head(inputotu)
  set.seed(999)
  ### One input network analysis ###
  network_result <- network_analysis2(
    inputotu,
    3,
    10,
    0.9,
    "spearman",
    TRUE
  )

  # Nodes information table
  network_nodes <- network_result$Nodes_info
  head(network_nodes)

  # Adjacency table
  network_adjacency <- network_result$Adjacency_column_table
  head(network_adjacency)

  # Complete adjacency matrix
  network_matrix <- network_result$Adjacency_matrix
  print(network_matrix[1:10, 1:10])

  # igraph object
  igraph_object <- network_result$Igraph_object
  network_stat(igraph_object)  # In case you want to see statistics again
  # or do other analysis based on igraph.

  ### Two inputs network analysis ###
  inputotu1 <- inputotu[1:456, ]
  inputotu2 <- inputotu[524:975, ]
  network_result <- network_analysis2(
    input = inputotu1,
    inputtype = 3,
    input2 = inputotu2,
    input2type = 3,
    n = 10,
    threshold = 0.85,
    method = "spearman",
    display = TRUE
  )

  #### Incorrect demonstration !! ###
  {
    network_result <- safe_run(network_analysis2(inputotu, 3, 3, 0.8, "spearman", TRUE))
```

```
  }
  # Total edges/links: 10199
  # Total vertices: 826
  # Too many edges and not a global network

}
```

`network_stat` *Igraph network statistics*

Description

Igraph network statistics

Usage

```
network_stat(input, report = TRUE)
```

Arguments

`input`	An igraph object.
`report`	Logical. If print report to console. Default:TRUE

Value

network statistics

Author(s)

Wang Ningqi 2434066068@qq.com

`network_visual` *Network Visualization*

Description

Visualizes a network based on a network object from `network_analysis`.

Usage

```
network_visual(
  network_obj,
  mode = "major_module",
  major_num = 5,
  taxlevel = NULL,
  select_tax = NULL,
  palette = "Set1",
  vertex.size = 6
)
```

Arguments

`network_obj`	A network analysis results object generated from `network_analysis`.
`mode`	The visualization mode, optionally "major_module" or "major_tax".
`major_num`	The number of major modules to display in the network.
`taxlevel`	Taxonomy levels used for visualization when mode is "major_tax".
`select_tax`	A vector of taxa to be selected for displaying in "major_tax" mode.
`palette`	Character. Palette for visualization.
`vertex.size`	Numeric. The size of the vertices.

Value

A list containing the configured igraph object and the coordinates of the vertices, with network visualization displayed in the plots panel.

Examples

```
{
  # Data preparation
  data("Two_group")
  set.seed(999)
  # Analysis
  network_results <- network_analysis(
    taxobj = Two_group,
    taxlevel = "Species",
    n = 10,
    threshold = 0.6
  )

  # Default mode
  network_visual_obj <- network_visual(network_obj = network_results)

  # View again
  network_visual_re(network_visual_obj)

  # More modules
  network_visual_obj <- network_visual(
```

```
    network_obj = network_results,
    major_num = 10
  )

  # Specific tax
  # Generate top 5 phyla for displaying
  community <- community_plot(
    taxobj = Two_group,
    taxlevel = "Phylum",
    n = 5,
    palette = "Paired"
  )
  display_phyla <- names(community$filled_color)

  network_visual_obj <- network_visual(
    network_obj = network_results,
    mode = "major_tax",
    taxlevel = "Phylum",
    select_tax = display_phyla,
    palette = community$filled_color
  )

  # Another sample for specific tax
  network_visual_obj <- network_visual(
    network_obj = network_results,
    mode = "major_tax",
    taxlevel = "Phylum",
    select_tax = "p__Proteobacteria"
  )
}
```

`network_visual_re` *Re-visualize network plot from* `network_visual` *or* `network_withdiff`

Description

Re-visualize network plot from `network_visual` or `network_withdiff`

Usage

```
network_visual_re(
  network_visual_obj,
  module_paint = FALSE,
  module_num = NULL,
  module_palette = c("aquamarine3", "antiquewhite2", "goldenrod2"),
  vertex.size = 6,
  vertex.shape = "circle"
)
```

Arguments

`network_visual_obj`	Network object from `network_visual` or `network_withdiff`
`module_paint`	Logical. If network module should be painted. Only work for network object from `network_withdiff`.
`module_num`	Numeric indicating which module to be painted.
`module_palette`	Character string with at least two elements. Palette for painting modules.
`vertex.size`	Numeric. The size of the vertices, default:6. Only for network object from `network_visual`
`vertex.shape`	Character. The shape of vertices, default: "circle"

Value

NULL but visualization in plot panel.

`network_withdiff` *Network Analysis with Differential Species*

Description

Meta network analysis integrating differential taxon into a network analysis

Usage

```
network_withdiff(network_obj, diff_frame, aes_col = NULL, tag_threshold = 5)
```

Arguments

`network_obj`	Network analysis results generated from `network_analysis`
`diff_frame`	Differential analysis results generated from `indicator_analysis` or `Deseq_analysis`.
`aes_col`	A named vector of colors to be used to highlight differential taxon vertices
`tag_threshold`	Numeric. A threshold for the minimum number of differential taxon to display.

Value

A list containing the configured igraph object, vertices coordinates, parameters, and tag statistics.

Examples

```
{
  # Data preparation
  data("Two_group")
  set.seed(999)
  # Analysis
  network_results <- network_analysis(
    taxobj = Two_group,
    taxlevel = "Genus",
    n = 10,
    threshold = 0.8
  )
  indicator_results <- indicator_analysis(
    taxobj = Two_group,
    taxlevel = "Genus"
  )
  deseq_results <- Deseq_analysis(
    taxobj = Two_group,
    taxlevel = "Genus",
    cutoff = 1,
    control_name = "Control"
  )

  # Visualize
  network_diff_obj <- network_withdiff(
    network_obj = network_results,
    diff_frame = indicator_results
  )
  # Check contained tags for each model
  print(network_diff_obj$tag_statistics$sum_of_tags)
  # Check contained different tags for each model
  print(network_diff_obj$tag_statistics$detailed_tags)

  # Re-visualize
  network_visual_re(
    network_visual_obj = network_diff_obj,
    module_paint = TRUE,
    module_num = c(1, 4)
  )  # Show module with most Treatment indicators

  my_module_palette <- color_scheme(
    c("#83BA9E", "#F49128"),
    5
  )
  network_visual_re(
    network_visual_obj = network_diff_obj,
    module_paint = TRUE,
    module_num = c(1, 4, 6, 3, 8),
    module_palette = my_module_palette
  )  # Show module with most Treatment indicators

  # Available also for DESeq analysis results
```

```
network_diff_obj <- network_withdiff(
  network_obj = network_results,
  diff_frame = deseq_results
)

# Parameter adjustment
network_diff_obj <- network_withdiff(
  network_obj = network_results,
  diff_frame = indicator_results,
  tag_threshold = 20
)  # The 'tag_threshold' set too high

network_diff_obj <- network_withdiff(
  network_obj = network_results,
  diff_frame = indicator_results,
  tag_threshold = 10
) # Set lower
# Check contained tags for each model
print(network_diff_obj$tag_statistics$sum_of_tags)
# Check contained different tags for each model
print(network_diff_obj$tag_statistics$detailed_tags)

network_diff_obj <- network_withdiff(
  network_obj = network_results,
  diff_frame = indicator_results,
  tag_threshold = 1
)  # Set too low

# Another example
data("Three_group")
network_results <- network_analysis(
  taxobj = Three_group,
  taxlevel = "Genus",
  n = 15,
  threshold = 0.9
)
indicator_results <- indicator_analysis(
  taxobj = Three_group,
  taxlevel = "Genus"
)

tag_color <- c(
  "CF" = "#F8766D",
  "CF_OF" = "#FFFF00",
  "OF" = "#00BA38",
  "OF_BF" = "#800080",
  "BF" = "#619CFF",
  "CF_BF" = "#00FFFF"
)
network_diff_obj <- network_withdiff(
  network_obj = network_results,
  diff_frame = indicator_results,
  aes_col = tag_color,
```

```
    tag_threshold = 10
  )

  # Re-visualize
  print(network_diff_obj$tag_statistics$detailed_tags)
  network_visual_re(
    network_visual_obj = network_diff_obj,
    module_paint = TRUE,
    module_num = c(8, 10, 11)
  )  # Show module with most BF indicators
  network_visual_re(
    network_visual_obj = network_diff_obj,
    module_paint = TRUE,
    module_num = c(1, 6, 8)
  )  # Show module with most BF and OF_BF indicators
}
```

`object_config` *Set taxonomy summary configuration*

Description

This function set taxonomy summary configuration by assigning treatment column number, facet column number, replication column number, treatment mapping color, treatment order and facet order.

Usage

```
object_config(
  taxobj,
  treat_location,
  facet_location = NULL,
  rep_location = "auto",
  subject_location = NULL,
  treat_col = NULL,
  treat_order = NULL,
  facet_order = NULL
)
```

Arguments

`taxobj`	Taxonomic summary object generated by `tax_summary`
`treat_location`	Numeric. Treatment column number in meta file/group information.
`facet_location`	Numeric, default:NULL. Facet column number in meta file/group information.
`rep_location`	Numeric. Replication column number in meta file/group information. By default:"auto",will automaticlly assigned the rep_location.

`subject_location`	Numeric, default:NULL.Subject column number in meta file/group information (used for pairwise experiment).
`treat_col`	Named character string, default:NULL. A set of aesthetic character to map treatment to.
`treat_order`	Character string, default:NULL. The character string indicating treatment displaying order.
`facet_order`	Character string, default:NULL. The character string indicating facet displaying order.

Value

object_config returns taxonomy summary object with configuration.

Author(s)

Wang Ningqi2434066068@qq.com

Examples

```
{
  ### Data preparation ###
  data(testotu)
  groupinformation <- data.frame(
    group = c(rep("a", 10), rep("b", 10)),
    factor1 = rnorm(10),
    factor2 = rnorm(mean = 100, 10),
    subject = factor(c(1:10, 1:10)),
    group2 = c(rep("e", 5), rep("f", 5), rep("e", 5), rep("f", 5))
  )

  ### Packaging metafile, community data, and taxonomy table ###
  test_object <- tax_summary(
    groupfile = groupinformation,
    inputtable = testotu[, 2:21],
    reads = TRUE,
    taxonomytable = testotu[, c(1, 22)]
  )

  ### Object configuration ###
  test_object_plan1 <- object_config(
    taxobj = test_object,
    treat_location = 1,
    rep_location = 4
  )

  ### Facet configuration ###
  test_object_plan2 <- object_config(
    taxobj = test_object,
    treat_location = 1,
    rep_location = 4,
```

```
    facet_location = 5
  )
}
```

`safe_run` *Running function with safe mode*

Description

Running function with safe mode

Usage

```
safe_run(func)
```

Arguments

`func`	Functions to run

Value

Same as of functions

Examples

```
safe_run(print("Hello"))
```

`show,LorMe-method` *Show method for LorMe objects*

Description

Custom summary display when LorMe object is printed.

Usage

```
## S4 method for signature 'LorMe'
show(object)
```

Arguments

`object`	A LorMe object

`structure_plot` *Visualize microbial community composition structure based on tax summary object*

Description

Function for visualization of microbial structure with PCAplot, PCoAplot and NMDSplot

Usage

```
structure_plot(
  taxobj,
  taxlevel,
  ptsize = 2,
  diagram = NULL,
  ellipse.level = 0.85,
  facet_row = NULL
)
```

Arguments

`taxobj`	Configured tax summary objects.See in `object_config`.
`taxlevel`	taxonomy levels used for visualization.Must be one of c("Domain","Phylum","Class","Order","Family","
`ptsize`	Numeric, default: 2. Size of point in plot. See `geom_point` for details.
`diagram`	Character, default: NULL. A character indicating group diagram, should be in c("ellipse", "stick", "polygon").
`ellipse.level`	Numeric, default: 0.85. The level at which to draw an ellipse, or, if type = "euclid", the radius of the circle to be drawn. See `stat_ellipse` for details.
`facet_row`	Numeric, default: NULL. Number of rows when wrap panels. See `facet_wrap` for details.

Value

Microbial structure analysis object.

Note

1. Do not use NMDS when warning: In metaMDS(t(inputframe)) :stress is (nearly) zero: you may have insufficient data
2. Ellipse not available when replicates less than 3, please use 'stick' or 'polygon' instead

Examples

```
###data preparation####
data("Two_group")

###analysis####
set.seed(999)
community_structure<- structure_plot(taxobj = Two_group,taxlevel = "Base")
 #check output list in console (not run)
 ######Output list##
 #####Plot#
 ####PCAplot:named as('PCA_Plot')(1/3)
 ####PCoAplot:named as('PCoA_Plot')(2/3)
 ####NMDSplot:named as('NMDS_Plot')(3/3)
 #####Analysis object#
 ####PCA object:named as('PCA_object')
 ####PCoA object:named as('PCoA_object')
 ####NMDS object:named as ('NMDS_object')
 #####Coordinates dataframe#
 ####PCA Coordinates dataframe:named as('PCA_coordinates')
 ####PCoA Coordinates dataframe:named as('PCoA_coordinates')
 ####NMDS Coordinates dataframe:named as('NMDS_coordinates')
 ######Done##
 #check PERMANOVA results
 community_structure$PERMANOVA_statistics

 #extract plot
 community_structure$PCA_Plot
 community_structure$PCoA_Plot
 community_structure$NMDS_Plot

 #extract object
 PCA_obj<- community_structure$PCA_object
 print(PCA_obj)

 #extract coordinates frame
 PCA_coord<- community_structure$PCA_coordinates
 head(PCA_coord)

 #stick plot
 set.seed(999)
 community_structure<- structure_plot(taxobj = Two_group,taxlevel = "Base",diagram = "stick")
 community_structure$PCoA_Plot

 #faced form
 data("Facet_group")
 set.seed(999)
 community_structure<- structure_plot(taxobj = Facet_group,taxlevel = "Genus",diagram = "stick")
 community_structure$PERMANOVA_statistics
 community_structure$PCA_Plot
 community_structure$PCoA_Plot
 community_structure$NMDS_Plot
```

`sub_tax_summary` *Subsetting tax summary objects*

Description

Subsetting tax summary objects

Usage

```
sub_tax_summary(taxobj, ..., specificnum = NULL, taxnum = NULL)
```

Arguments

`taxobj`	tax summary objects computed by `tax_summary`.
`...`	logical expression that are are defined in terms of the variables in Groupfile of tax summary objects. See details in `subset`.
`specificnum`	specific numbers indicating samples to keep based on Groupfile of tax summary objects.
`taxnum`	specific numbers indicating taxonomy to keep based on Base file

Value

Subset of tax summary objects.Same as `tax_summary`.

Author(s)

Wang Ningqi 2434066068@qq.com

Examples

```
data("Three_group")

# Check meta file
print(Three_group$groupfile)

# Subsetting tax summary objects

# Select BF and OF groups
sub_testtax_summary <- sub_tax_summary(Three_group, Group %in% c("BF", "OF"))
print(sub_testtax_summary$groupfile)
```

`tax_summary` *Encapsulate meta file, feature tables and taxonomy annotation into tax summary object*

Description

The function packages meta file, feature tables and taxonomy annotation into tax summary object

Usage

```
tax_summary(
  groupfile,
  inputtable,
  reads = TRUE,
  taxonomytable,
  into = "standard",
  sep = ";",
  outputtax = c("Phylum", "Genus")
)
```

Arguments

`groupfile`	A data frame containing treatment information
`inputtable`	OTU/ASV/species data frame with all numeric. Samples ID should be in column names.
`reads`	Logical.True for reads table and FALSE for percentage table. Default: TRUE
`taxonomytable`	Taxonomy annotation data frame,with first column OTU/ASV/TAX number ID and second column taxonomy annotation. See details in example.
`into`	Names of separated taxonomy to create as character vector. Must select from c("Domain","Phylum","Class","Order","Family","Genus","Species"). Shortcut input:1)By default."standard":c("Domain","Phylum","Class","Order","Family","Genus","Species"). Used for standard taxonomy annotation to OTU/ASV table. 2)"complete":c("Domain","Kingdom","Phylu Used for complete taxonomy annotation to meta genomic table.
`sep`	Separator of taxonomy table.Default: ";".
`outputtax`	Names of output taxonomy level table. Default:c("Phylum","Genus"). Shortcut input is available with 'standard' and 'complete' same as above.

Value

LorMe object containing taxonomy table data frame,containing reads and percentage table for each specified output.

Note

For taxonomy annotation with both 'Domain' and 'Kingdom' level, please set 'into' parameter as 'complete'!!!

Author(s)

Wang Ningqi 2434066068@qq.com

Examples

```
{
  # Load data
  data(testotu)

  # Create group information data frame
  groupinformation <- data.frame(
    group = c(rep("a", 10), rep("b", 10)),
    factor1 = rnorm(10),
    factor2 = rnorm(mean = 100, 10),
    subject = factor(c(1:10, 1:10))
  )

  # Packaging data into a taxonomy summary object
  test_object <- tax_summary(
    groupfile = groupinformation,
    inputtable = testotu[, 2:21],
    reads = TRUE,
    taxonomytable = testotu[, c(1, 22)]
  )

  # Check integrated object
  print(test_object)

  # Extract genus relative abundance table
  test_Genus <- test_object$Genus_percent
  head(test_Genus)

  # Check corresponding taxonomy information of genus table
  test_Genus_tax <- test_object$Genus_taxonomy
  head(test_Genus_tax)

  # Summary base table into all taxonomy levels with standard output
  test_object <- tax_summary(
    groupfile = groupinformation,
    inputtable = testotu[, 2:21],
    reads = TRUE,
    taxonomytable = testotu[, c(1, 22)],
    outputtax = "standard"
  )
  head(test_object$Species_percent)  # View first 10 rows of species percentage
  head(test_object$Genus)  # View first 10 rows of genus table
}
```

`tbRDA_analysis` *tbRDA analysis*

Description

RDA analysis including co-linearity diagnostics and necessary statistics.

Usage

```
tbRDA_analysis(otudata, envdata, collinearity, perm.test = TRUE)
```

Arguments

`otudata`	Feature table of all numeric variable, with annotation in row names
`envdata`	Environmental factor of all numeric variable,with sample-ID in row names and environmental factor in column names
`collinearity`	If done collinearity diagnostics. Default,TRUE.
`perm.test`	Logical. If conduct permutation test. Default:TRUE.

Value

Three permutation test result print ,one preview plot ,a RDA object(default name:otu.tab.1) and a summary of RDA object

Note

1. When Axis length in first axis more than 4, you should choose CCA instead of RDA.

Author(s)

Wang Ningqi 2434066068@qq.com

Examples

```
### Data preparation ###
library(vegan)
data(varechem)
head(varechem)
data(testotu)
require(tidyr); require(magrittr)  ## Or use pipe command in "dplyr"

sep_testotu <- Filter_function(
  input = testotu,
  threshold = 0.0001,
  format = 1
) %>%
separate(
  ., col = taxonomy,
  into = c("Domain", "Phylum", "Order", "Family", "Class", "Genus", "Species"),
  sep = ";"
)

top10phylum <- aggregate(
  sep_testotu[, 2:21],
```

```
  by = list(sep_testotu$Phylum),
  FUN = sum
) %>%
Top_taxa(
  input = .,
  n = 10,
  inputformat = 2,
  outformat = 1
)
rownames(top10phylum) <- top10phylum[, 1]
top10phylum <- top10phylum[, -1]

group <- data.frame(
  group = c(rep("a", 10), rep("b", 10)),
  factor1 = rnorm(10),
  factor2 = rnorm(mean = 100, 10)
)

### RDA analysis ###
set.seed(999)
RDAresult <- tbRDA_analysis(
  top10phylum,
  varechem[1:20, ],
  TRUE
)

# Environmental statistics
print(RDAresult$factor_statistics)

# Visualization using ggplot
rda_object <- RDAresult$rda_object
rda_summary <- RDAresult$rdasummary
rda_scores <- RDAresult$rdascores
rda_env <- as.data.frame(rda_scores$biplot)
rda_sample <- as.data.frame(rda_scores$sites)
rda_otu <- as.data.frame(rda_scores$species)

xlab <- paste0("RDA1:", round(RDAresult$rdasummary$concont$importance[2, 1], 4) * 100, "%")
ylab <- paste0("RDA2:", round(RDAresult$rdasummary$concont$importance[2, 2], 4) * 100, "%")

library(ggrepel)
# Create a sample RDA plot
RDAplot <- ggplot(data = rda_sample, aes(RDA1, RDA2)) +
  geom_point(aes(color = group$group), size = 2) +
  geom_point(data = rda_otu, pch = "+", color = "orange", size = 4) +
  geom_hline(yintercept = 0) +
  geom_vline(xintercept = 0) +
  geom_segment(data = rda_env, aes(x = 0, y = 0, xend = RDA1 * 0.8, yend = RDA2 * 0.8),
               arrow = arrow(angle = 22.5, length = unit(0.35, "cm")),
               linetype = 1, size = 0.6, colour = "red") +
  geom_text_repel(color = "red", data = rda_env,
                  aes(RDA1, RDA2, label = row.names(rda_env))) +
  labs(x = xlab, y = ylab, color = "Treatment",
```

```
        title = paste0("p = ", anova.cca(rda_object)["Model", "Pr(>F)"])) +
    stat_ellipse(aes(color = group$group), level = 0.95) +
    geom_text_repel(size = 3, color = "orange",
                    data = subset(rda_otu, RDA1 > 0.1 | RDA1 < (-0.1)),
              aes(RDA1, RDA2, label = rownames(subset(rda_otu, RDA1>0.1|RDA1<(-0.1))))) +
    theme_zg()

  # Print the RDA plot
  print(RDAplot)
```

`testotu` *test otudata*

Description

A dataset containing 20 samples and 1000 OTUs from soil to test,taxonomy information is covered randomly(not actual)

Usage

```
testotu
```

Format

A data frame with 1000 rows and 22 variables.

`theme_zg` *A classic theme for ggplot*

Description

A classic theme for ggplot

Usage

```
theme_zg()
```

Value

ggplot theme

Note

Build inside the LorMe package, Please use theme_zg() as a theme directly

---

`Three_group` *Tax summary object with three groups*

Description

Enraptured summary object with three groups.Configuration has been assigned.

Usage

```
Three_group
```

Format

Tax summary object with configuration

`Top_taxa` *Calculate top taxa and others*

Description

Top taxa is widely used in data analysis,here we provide a simple function to calculate which simplify your R script.

Usage

```
Top_taxa(input, n, inputformat, outformat)
```

Arguments

`input`	Reads or relative abundance(recommended) of OTU/Taxa/gene data frame,see details in inputformat
`n`	Top n taxa remained according to relative abundance
`inputformat`	1:data frame with first column of OTUID and last column of taxonomy 2:data frame with first column of OTUID/taxonomy (recommended!!!) 3:data frame of all numeric,with row names of OTUID/taxonomy
`outformat`	1. return outformat the same as inputformat 2. return data frame of all numeric with OTU/gene/taxa ID in row names(not available for inputformat 1).

Value

Data frame with top n taxa

Author(s)

Wang Ningqi2434066068@qq.com

Examples

```
### Data preparation ####
data(testotu)
require(tidyr); require(magrittr)  ## Or use pipe command in "dplyr"

testotu.pct <- data.frame(
  OTU.ID = testotu[, 1],
  sweep(testotu[, -c(1, 22)], 2, colSums(testotu[, -c(1, 22)]), "/"),
  taxonomy = testotu[, 22]
)

sep_testotu <- Filter_function(
  input = testotu,
  threshold = 0.0001,
  format = 1
) %>%
  separate(
    ., col = taxonomy,
    into = c("Domain", "Phylum", "Order", "Family", "Class", "Genus", "Species"),
    sep = ";"
  )

phylum <- aggregate(
  sep_testotu[, 2:21], by = list(sep_testotu$Phylum), FUN = sum
)

phylum1 <- data.frame(row.names = phylum[, 1], phylum[, -1])

##### Input format 1, top 100 OTU #####
top100otu <- Top_taxa(
  input = testotu.pct,
  n = 100,
  inputformat = 1,
  outformat = 1
)

##### Input format 2, top 15 phylum #####
head(phylum)
top15phylum <- Top_taxa(
  input = phylum,
  n = 15,
  inputformat = 2,
  outformat = 1
)

##### Input format 3, top 15 phylum #####
head(phylum1)
top15phylum <- Top_taxa(
```

```
  input = phylum1,
  n = 15,
  inputformat = 3,
  outformat = 1
)
```

`Trans_from_microeco` *Convert a microeco object to a LorMe object*

Description

Convert a microeco object to a LorMe object

Usage

```
Trans_from_microeco(
  microtab,
  into = "standard",
  outputtax = c("Phylum", "Genus"),
  reads = TRUE
)
```

Arguments

`microtab`	A `microtable` object
`into`	Names of separated taxonomy to create as character vector. Must select from c("Domain","Phylum","Class","Order","Family","Genus","Species"). Shortcut input:1)By default."standard":c("Domain","Phylum","Class","Order","Family","Genus","Species"). Used for standard taxonomy annotation to OTU/ASV table. 2)"complete":c("Domain","Kingdom","Phylu Used for complete taxonomy annotation to meta genomic table.
`outputtax`	Default:c("Phylum","Genus").Names of output taxonomy level table. Shortcut input is available with 'standard' and 'complete' same as above.
`reads`	Logical.True for reads table and FALSE for percentage table. Default: TRUE

Value

LorMe object containing taxonomy table data frame,containing reads and percentage table for each specified output.

Note

For taxonomy annotation with both 'Domain' and 'Kingdom' level, please set 'into' parameter as 'complete'!!!

Examples

```
## Not run:
if (requireNamespace("microeco", quietly = TRUE)) {
 data("dataset", package = "microeco")

 ## convert microeco object to LorMe format
 dataset_obj <- Trans_from_microeco(dataset, outputtax = "standard")

 ## specify experimental design
 dataset_obj_plan <- object_config(dataset_obj,
                         treat_location = 4)

 ## generate community-structure PCoA plot (Genus level)
 community_structure <- structure_plot(dataset_obj_plan, taxlevel = "Genus")
 community_structure$PCoA_Plot
}
```

`Trans_from_phylo` *Convert phyloseq object to LorMe object*

Description

Convert phyloseq object to LorMe object

Usage

```
Trans_from_phylo(
  physeq,
  into = "standard",
  outputtax = c("Phylum", "Genus"),
  reads = TRUE
)
```

Arguments

`physeq`	The phyloseq object to convert. See in `phyloseq` object
`into`	Names of separated taxonomy to create as character vector. Must select from c("Domain","Phylum","Class","Order","Family","Genus","Species"). Shortcut input:1)By default."standard":c("Domain","Phylum","Class","Order","Family","Genus","Species"). Used for standard taxonomy annotation to OTU/ASV table. 2)"complete":c("Domain","Kingdom","Phylu Used for complete taxonomy annotation to meta genomic table.
`outputtax`	Default:c("Phylum","Genus").Names of output taxonomy level table. Shortcut input is available with 'standard' and 'complete' same as above.
`reads`	Logical.True for reads table and FALSE for percentage table. Default: TRUE

Value

LorMe object containing taxonomy table data frame,containing reads and percentage table for each specified output.

Note

For taxonomy annotation with both 'Domain' and 'Kingdom' level, please set 'into' parameter as 'complete'!!!

The taxonomyTable of phyloseq object should contained as least standard annotations for convert!

Examples

```
## Not run:

if (requireNamespace("phyloseq", quietly = TRUE)) {
 data("GlobalPatterns", package = "phyloseq")

 ## convert phyloseq object to LorMe format
 gp_obj <- Trans_from_phylo(GlobalPatterns, outputtax = "standard")

 ## specify experimental design
 gp_plan <- object_config(gp_obj,
                          treat_location = 6,
                          rep_location   = 7)

 ## generate community-structure PCoA plot (Genus level)
 community_structure <- structure_plot(gp_plan, taxlevel = "Genus")
 community_structure$PCoA_Plot
}
```

`Trans_to_microeco` *Convert a LorMe object to a microeco object*

Description

Convert a LorMe object to a microeco object

Usage

```
Trans_to_microeco(taxobj, use_reads = TRUE)
```

Arguments

`taxobj`	A `LorMe` object (usually generated by `tax_summary`).
`use_reads`	Logical. `FALSE` (default) uses **relative abundance** (`Base_percent`); `TRUE` uses **raw counts** (`Base`).

Value

A `microtable` object.

Examples

```
## Not run:
data("Two_group")

Two_group_microeco=Trans_to_microeco(Two_group)

t1 <- trans_abund$new(dataset = Two_group_microeco, taxrank = "Phylum", ntaxa = 10)
t1$plot_bar(others_color = "grey70",
            legend_text_italic = FALSE) +
        theme(axis.title.y = element_text(size = 18)) #Just to show it can be used for analysis

## End(Not run)
```

`Trans_to_phylo` *Convert a LorMe object to a phyloseq object*

Description

Convert a LorMe object to a phyloseq object

Usage

```
Trans_to_phylo(taxobj, use_reads = TRUE)
```

Arguments

`taxobj`	A `LorMe` object (usually generated by `tax_summary`).
`use_reads`	Logical. `FALSE` (default) uses **relative abundance** (`Base_percent`); `TRUE` uses **raw counts** (`Base`).

Value

A `phyloseq` object containing `otu_table`, `sample_data`, `tax_table` and optionally `phy_tree` (only if `taxobj` has a tree).

Examples

```
## Not run:
data("Two_group")

Two_group_phylo=Trans_to_phylo(taxobj = Two_group,use_reads = F)

plot_bar(Two_group_phylo,fill="Phylum") #Just to show it can be used for analysis

## End(Not run)
```

`Two_group` *Tax summary object with two groups*

Description

Enraptured summary object with two groups.Configuration has been assigned.

Usage

```
Two_group
```

Format

Tax summary object with configuration

`t_test_report` *Print Student's t-Test report*

Description

Print Student's t-Test report

Usage

```
t_test_report(
  data,
  treatment_col,
  value_col,
  paired,
  subject_col,
  report = TRUE
)
```

Arguments

`data`	Data frame containing the treatment, value and other information.
`treatment_col`	Numeric indicating where treatment locates (column number) in data.
`value_col`	Numeric indicating where treatment value (column number) in data.
`paired`	Logical indicating whether you want a paired t-test.
`subject_col`	Only meaningful when Pair is ture. Numeric indicating where subject of treatment (column number) in data.
`report`	Logical. If print report to console. Default:TRUE

Value

t_test_report returns list containing:

1. data frame of basic data descrption
2. results of student's t-Test

Examples

```
{
  ### Data preparation ###
  testdata <- data.frame(
    treatment = c(rep("A", 6), rep("B", 6)),
    subject = rep(c(1:6), 2),
    value = c(rnorm(6, 2), rnorm(6, 1))
  )

  # Perform t-test (unpaired)
  t_test_result <- t_test_report(
    data = testdata,
    treatment_col = 1,
    value_col = 3
  )

  # Perform paired t-test
  t_test_result <- t_test_report(
    data = testdata,
    treatment_col = 1,
    value_col = 3,
    paired = TRUE,
    subject_col = 2
  )

  ### Basic data description ###
  print(t_test_result[[1]])
  print(t_test_result$basicdata)

  ### T-test results ###
  print(t_test_result[[2]])
  print(t_test_result$t.test_results)
}
```

`volcano_plot` *Generate Volcano plot base on Deseq_analysis or indicator_analysis results*

Description

Generate Volcano plot base on Deseq_analysis or indicator_analysis results

Usage

```
volcano_plot(inputframe, cutoff = NULL, aes_col = c("#FE5C5C", "#75ABDE"))
```

Arguments

`inputframe`	A data frame containing the results based on `Deseq_analysis` or `indicator_analysis` (only two group indicators)
`cutoff`	A numeric value specifying the fold change cutoff,should be the same as in `Deseq_analysis`
`aes_col`	A named vector of colors to be used in the plots

Value

A list of two ggplot objects, one for the fold change versus adjusted p-value plot and another for the mean abundance versus fold change or enrichment factor plot.

Author(s)

Wang Ningqi 2434066068@qq.com

Examples

```
###data prepration###

{
  # Load data
  data("Two_group")

  # Define color based on treatment column
  mycolor <- Two_group$configuration$treat_col

  ### DESeq analysis ###
  deseq_results <- Deseq_analysis(
    taxobj = Two_group,
    taxlevel = "Genus",
    cutoff = 1,
    control_name = "Control"
  )

  ### Or indicator analysis ###
  indicator_results <- indicator_analysis(
    taxobj = Two_group,
    taxlevel = "Genus"
  )

  # Create volcano plot for DESeq results
  volcano_plot <- volcano_plot(
    inputframe = deseq_results,
    cutoff = 1,
    aes_col = mycolor
  )
```

```
  print(volcano_plot$FC_FDR)  # Fold Change and FDR values
  print(volcano_plot$Mean_FC)  # Mean Fold Change values

  # Create volcano plot for indicator results
  volcano_plot <- volcano_plot(
    inputframe = indicator_results,
    cutoff = 1,
    aes_col = mycolor
  )
  print(volcano_plot$FC_FDR)  # Fold Change and FDR values
  print(volcano_plot$Mean_FC)  # Mean Fold Change values
}
```

`wilcox_test_report` *Print Wilcoxon Rank Sum and Signed Rank Tests reprot*

Description

Print Wilcoxon Rank Sum and Signed Rank Tests reprot

Usage

```
wilcox_test_report(
  data,
  treatment_col,
  value_col,
  paired = FALSE,
  subject_col = NULL,
  report = TRUE
)
```

Arguments

`data`	Data frame containing the treatment, value and other information.
`treatment_col`	Numeric indicating where treatment locates (column number) in data.
`value_col`	Numeric indicating where treatment value (column number) in data.
`paired`	Logical indicating whether you want a paired test.
`subject_col`	Only meaningful when Pair is ture. Numeric indicating where subject of treatment (column number) in data.
`report`	Logical. If print report to console. Default:TRUE

Value

wilcox_test_report returns data frame of basic data description.

Examples

```
{
  # Data preparation
  testdata <- data.frame(
    treatment = c(rep("A", 6), rep("B", 6)),
    subject = rep(1:6, 2),
    value = c(rnorm(6, 2), rnorm(6, 1))
  )

  # Wilcoxon test (unpaired)
  wilcox_result <- wilcox_test_report(
    data = testdata,
    treatment_col = 1,
    value_col = 3
  )

  # Wilcoxon signed rank test (paired)
  wilcox_result <- wilcox_test_report(
    data = testdata,
    treatment_col = 1,
    value_col = 3,
    paired = TRUE,
    subject_col = 2
  )

  ### Basic data description ###
  print(wilcox_result)
}
```

`$,LorMe-method` *Access LorMe object elements via $*

Description

S4 method to enable $ operator for LorMe objects, supporting both slots and elements in the 'data' slot.

Usage

```
## S4 method for signature 'LorMe'
x$name
```

Arguments

`x`	A LorMe object
`name`	Character: Slot name or key in the 'data' slot

Value

Value of the specified slot or data element

Index