Package 'MicrobTiSDA'

May 7, 2026

Title Microbiome Time-Series Data Analysis

Version 0.1.0

Description Provides tools specifically designed for analyzing longitudinal microbiome data. This tool integrates seven functional modules, providing a systematic framework for microbiome time-series analysis. For more details on inferences involving interspecies interactions see Fisher (2014) <doi:10.1371/journal.pone.0102451>. Details on this package are also described in an unpublished manuscript.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/Lishijiagg/MicrobTiSDA

BugReports https://github.com/Lishijiagg/MicrobTiSDA/issues

Imports caret, cluster, dplyr, ggdendro, ggplot2, ggraph, MASS, mgcv, plyr, pracma, randomForest, reshape2, scales, splines, stats, tibble, tidygraph, tidyr, vegan, visNetwork

Depends R (>= 3.5)

LazyData true

LazyDataCompression xz

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Shijia Li [aut, cre] (ORCID: <https://orcid.org/0009-0006-6827-5344>)

Maintainer Shijia Li <s.li2@uva.nl>

Repository CRAN

Date/Publication 2025-09-22 18:20:02 UTC

Contents

`Classify.vis` *Visualize Microbial Feature Classification Results Using NMDS*

Description

This function generates a non-metric multidimensional scaling (NMDS) plot to visualize OTU/ASV classification results. It combines an OTU/ASV table with predicted group labels to produce a scatter plot in NMDS space, where each sample is colored according to its predicted group.

Usage

```
Classify.vis(
  classified_results,
  dist_method = "bray",
  fig_title = "NMDS visualization",
  legd_title = "Predicted Groups",
  points_size = 1,
  legend_title_size = 8,
  legend_text_size = 6,
  axis_title_size = 6,
  axis_text_size = 6
)
```

Arguments

classified_results	A list object of Data.rf.classifier output.
dist_method	Dissimilarity index, defaults to bray. For other options, including manhattan, euclidean, canberra etc., see vegdist.
fig_title	A character string specifying the title of the NMDS plot. Default is 'NMDS visualization'.
legd_title	A character string for the legend title representing the predicted groups. Default is 'Predicted Groups'.
points_size	A numeric value specifying the size of the points in the plot. Default is 1.
legend_title_size	A numeric value specifying the font size of the legend title. Default is 8.
legend_text_size	A numeric value specifying the font size of the legend text. Default is 6.
axis_title_size	A numeric value specifying the font size of the axis titles. Default is 6.
axis_text_size	A numeric value specifying the font size of the axis text. Default is 6.

Details

The function expects as input the results of output of the function Data.rf.classifier. It computes NMDS coordinates based on the OTU/ASV data using the specified distance method (defaulting to Bray-Curtis) and then maps the predicted group labels onto the NMDS coordinates. The visualization is produced using ggplot2, with polygons delineating the convex hull of each predicted group.

Value

A ggplot2 object displaying the NMDS plot with samples colored by their predicted group and convex hulls outlining each group.

Author(s)

Shijia Li

Examples

```
# Example OTU count data (20 OTUs x 10 samples)
set.seed(123)
otu_data <- matrix(sample(0:100, 200, replace = TRUE), nrow = 20)
colnames(otu_data) <- paste0("Sample", 1:10)
rownames(otu_data) <- paste0("OTU", 1:20)

# Example metadata with group labels
metadata <- data.frame(Group = rep(c("Control", "Treatment"), each = 5))

# Run the classifier
rf_result <- Data.rf.classifier(raw_data = otu_data,
```

```
                              metadata = metadata,
                              train_p = 0.7,
                              Group = "Group",
                              OTU_counts_filter_value = 50)
# If you wish to select the top 5 features:
result <- Rf.biomarkers(rf = rf_result, feature_select_num = 5)

nm_plot <- Classify.vis(classified_results = result,
                        dist_method = 'bray',
                        fig_title = 'NMDS Plot',
                        legd_title = 'Predicted Groups',
                        points_size = 1.5)
```

`Data.cluster` *Cluster OTU Time-Series Data Based on Regression Model prediction and Generate Dendrogram Plots*

Description

This function performs hierarchical clustering on predicted OTU time-series data for different groups and generates corresponding dendrogram plots. For each group in the input list, the function computes a correlation-based distance matrix, performs hierarchical clustering using the specified clustering method (e.g. `average`), and then converts the result into a dendrogram.

Usage

```
Data.cluster(
  predicted_data,
  clust_method = "complete",
  font_size = 0.2,
  dend_title_size = 15
)
```

Arguments

`predicted_data`	The output data frame from the `Pred.data`.
`clust_method`	A string, the agglomeration method to be used. This argument should be one of "ward.D", "ward.D2", "single", "complete", "average", "mcquitty", "median", "centroid". Detail see `hclust`.
`font_size`	A numeric value specifying the font size for text labels in the dendrogram plots (default: `0.2`).
`dend_title_size`	A numeric value specifying the font size of the dendrogram plot title (default: `15`).

Details

For each group in the input `predicted_data`, the function first extracts the predicted OTU data (excluding the last column, which is assumed to contain time information) and computes a correlation matrix, which is converted into a distance matrix via

$$d_{\text{corr}}(x, y) = 1 - \frac{\sum_{i=1}^{n}(x_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where x and y represent the two OTU time series being compared, n denotes the total number of time points, and $\bar{x}$ and $\bar{y}$ denote the means of the respective time series. Hierarchical clustering is performed on the above distance matrix using the method specified in `clust_method`.

Value

An object of `MicrobTiSDA.cluster` with three elements:

predicted_data The original input list of predicted data.

cluster_results A list of hierarchical clustering objects (one per group).

cluster_figures A list of ggplot2 objects containing the dendrogram plots for each group.

Author(s)

Shijia Li

Examples

```
# Example metadata with grouping variables
metadata <- data.frame(
  TimePoint = c(1, 2, 3, 4),
  Sample = c('S1', 'S2', 'S3', 'S4'),
  GroupA = c('A', 'A', 'B', 'B'),
  GroupB = c('X', 'Y', 'X', 'Y')
)

# Example pre-processed data (e.g., transformed abundance data)
Pre_processed_Data <- data.frame(
  Feature1 = rnorm(4),
  Feature2 = rnorm(4)
)

# Create design matrix using grouping variables
design_data <- Design(metadata, Group_var = c('GroupA', 'GroupB'), Pre_processed_Data,
                      Sample_Time = 'TimePoint', Sample_ID = 'Sample')

reg <- Reg.SPLR(design_data,
                  Pre_processed_Data,
                  z_score = 2,
                  unique_values = 5,
                  Knots = NULL,
                  max_Knots = 5)
predictions <- Pred.data(reg,
```

```
                        metadata,
                        Group = "GroupA",
                        time_step = 1,
                        Sample_Time = "TimePoint")
result <- Data.cluster(predicted_data = predictions,
                       clust_method = "average",
                       font_size = 0.2,
                       dend_title_size = 15)
```

`Data.cluster.cut` *Cut and Annotate Dendrogram Based on a Specified Cut-Off Height*

Description

This function processes clustering outputs from `Data.cluster` to update dendrogram plots. Depending on the user's preference, it either automatically determines the optimal number of clusters via silhouette analysis or uses a user-specified cut-off height.

Usage

```
Data.cluster.cut(
  cluster_outputs,
  cut_height,
  cut_height_dist = 0.2,
  font_size = 0.2,
  auto_cutree = FALSE
)
```

Arguments

`cluster_outputs`	The output object of `Data.cluster`.
`cut_height`	A numeric value specifying the cut-off height for cutting the dendrogram when `auto_cutree` is `FALSE`.
`cut_height_dist`	A numeric value used to adjust the vertical distance of the cut-off line annotation in the dendrogram plot (default: `0.2`).
`font_size`	A numeric value specifying the font size for text labels in the dendrogram plots (default: `0.2`).
`auto_cutree`	Logical; if `TRUE`, the function automatically determines the optimal number of clusters based on silhouette width (default: `FALSE`).

Details

The function takes as input a list containing predicted data and clustering results (typically generated by another function) from `Data.cluster`, and then computes a correlation-based distance matrix for each group. If `auto_cutree` is `TRUE`, the function performs a repeated k-fold cross-validation by iterating over a range of potential cluster numbers and computing the average silhouette width, thereby determining the optimal number of clusters. The dendrogram is then cut accordingly, and the resulting clusters are used to annotate the dendrogram plot with different colors for each cluster.

If `auto_cutree` is `FALSE`, the function uses the provided `cut_height` to cut the dendrogram. It then assigns cluster emberships based on this cut-off and updates the dendrogram plot by adding a horizontal dashed line at the specified cut-off and annotating the plot with the cut-off value. In both cases, the function prints the dendrogram plot for each group and returns a list containing the clustering results and the corresponding ggplot2 objects of the dendrograms.

Value

A object of `MicrobTiSDA.clusterCut` with two elements:

cluster_results A list of clustering objects for each group.

cluster_figures A list of ggplot2 objects containing the annotated dendrogram plots for each group.

Author(s)

Shijia Li

Examples

```
# Example metadata with grouping variables
metadata <- data.frame(
  TimePoint = c(1, 2, 3, 4),
  Sample = c('S1', 'S2', 'S3', 'S4'),
  GroupA = c('A', 'A', 'B', 'B'),
  GroupB = c('X', 'Y', 'X', 'Y')
)

# Example pre-processed data (e.g., transformed abundance data)
Pre_processed_Data <- data.frame(
  Feature1 = rnorm(4),
  Feature2 = rnorm(4)
)

# Create design matrix using grouping variables
design_data <- Design(metadata, Group_var = c('GroupA', 'GroupB'), Pre_processed_Data,
                      Sample_Time = 'TimePoint', Sample_ID = 'Sample')

reg <- Reg.SPLR(design_data,
                  Pre_processed_Data,
                  z_score = 2,
                  unique_values = 5,
                  Knots = NULL,
                  max_Knots = 5)
```

```
predictions <- Pred.data(reg,
                         metadata,
                         Group = "GroupA",
                         time_step = 1,
                         Sample_Time = "TimePoint")
result <- Data.cluster(predicted_data = predictions,
                       clust_method = "average",
                       font_size = 0.2,
                       dend_title_size = 15)

result <- Data.cluster.cut(cluster_outputs = result,
                           cut_height = 0.3,
                           cut_height_dist = 0.2,
                           auto_cutree = FALSE)

# To automatically determine the optimal number of clusters:
result_auto <- Data.cluster.cut(cluster_outputs = result, auto_cutree = TRUE)
```

`Data.filter` *Filtering for Microbial Features of Low Abundance and Low Prevalence.*

Description

This function filteres an OTU/ASV table based on overall counts and prevalence thresholds, and optionally applies a logarithmic transformation. When grouping variables are provided, the function performs abundance and prevalence filtering within each group separately.

Usage

```
Data.filter(
  Data,
  metadata,
  OTU_counts_filter_value = 1000,
  OTU_filter_value = NA,
  log_base = NA,
  Group_var = NULL
)
```

Arguments

`Data`	A data frame or a list object which contains the selected biomarker count table (generated by `Data.rf.classifier`), where rows represent OTUs/ASVs and columns represent samples.
`metadata`	A data frame. Containing information about all samples, including at least the grouping of all samples as well as individual information (`Group` and `ID`), the sampling `Time` point for each sample, and other relevant information.

`OTU_counts_filter_value`
: An integer, indicating the sum of the minimum abundances of OTUs/ASVs in all samples. If the sum of the abundances that OTU is below the given positive integer threshold, the OTU is excluded, and vice versa, it is retained. The default is 1000. Note: if the input `Data` is the important OTU table that produced via sample classification, this argument should be `NA`, As the low abundance OTUs/ASVs might be filtered out during the classification progress by `Data.rf.classifier`.

`OTU_filter_value`
: Numeric between 0 and 1. This specifies the minimum prevalence rate of an OTU/ASV across all samples within each group or individual. OTUs/ASVs with a prevalence rate below the given threshold will be removed.

`log_base`
: This argument specifies the base of the logarithm. When the dataset is not very large, the default is `NA`, and no logarithmic transformation is applied. For large datasets, the logarithm base can be 2, "e", or 10.

`Group_var`
: A string or a vector. This specifies the grouping variables, which should match the column names in the `metadata` used to designate sample groups, and for pre-processing OTU data of each group or individual separately. For instance, to split the OTU table based on the `Group` variable, set `Group_var = "Group"`; to split the data based on the `Group` and `Diet` (if in `metadata`)categorical variables to study the interaction between different grouping variables, set `Group_var = c("Group","Diet")`.

Details

The function executes several key steps:

1. **Input Validation:** It first checks whether the input `Data` is a data frame or a list generated by function `Data.rf.classifier`. If `Data` is a list but not a data frame, the first element is extracted. Otherwise, if `Data` is neither a data frame nor an appropriate list, the function stops with an error.
2. **OTU Count Filtering:** If an `OTU_counts_filter_value` is provided (i.e., not `NA`), OTUs with total counts (across all samples) less than or equal to this value are removed.
3. **Logarithmic Transformation:** If a `log_base` is specified (allowed values are 10, 2, or e), a log transformation (with an offset of 1 to avoid log(0)) is applied to the data. If `log_base` is `NA`, the data remains untransformed.
4. **Prevalence Filtering without Grouping:** When `Group_var` is not provided (`NULL`), if an `OTU_filter_value` is specified, the function filters out OTUs whose prevalence (the proportion of samples with a non-zero count) is less than the threshold. If `OTU_filter_value` is not provided, a warning is issued and no prevalence filtering is applied.
5. **Group-based Prevalence Filtering:** If one or more grouping variables are specified in `Group_var`, the function first checks that these variables exist in `metadata`. For each group (or combination of groups if multiple variables are provided), the prevalence of each OTU is calculated, and OTUs are retained if they meet the prevalence threshold in at least one group. The filtered OTU table is then returned.

Value

A list of class `FilteredData` containing:

filtered_table The filtered OTU count table, optionally log-transformed.

parameters A list of the filtering parameters used.

metadata The input metadata, possibly augmented with a combined grouping variable if multiple `Groups` were provided.

Author(s)

Shijia Li

Examples

```
# Example OTU table
set.seed(123)
otu_table <- as.data.frame(matrix(sample(0:100, 100, replace = TRUE), nrow = 10))
rownames(otu_table) <- paste0("OTU", 1:10)
colnames(otu_table) <- paste0("Sample", 1:10)


# Example metadata
metadata <- data.frame(
  Group = rep(c("A", "B"), each = 5),
  row.names = paste0("Sample", 1:10)
)

# Filter OTU table without grouping
filtered_data <- Data.filter(
  Data = otu_table,
  metadata = metadata,
  OTU_counts_filter_value = 50,
  OTU_filter_value = 0.2
)

# Filter OTU table with grouping
filtered_data_grouped <- Data.filter(
  Data = otu_table,
  metadata = metadata,
  OTU_filter_value = 0.5,
  Group_var = "Group"
)
```

`Data.interpolate` *Interpolate Time-Series Data Based on Sample Time*

Description

This function performs interpolation on a data frame or matrix (e.g., OTU/ASV counts or other time-series measurements) using corresponding metadata time points. For each unique subject (as defined by a subject ID), the function constructs a full time series between the minimum and maximum time points and applies interpolation (defaulting to cubic interpolation) to generate data for missing time points. The function returns both the interpolated time-series data and the associated updated metadata.

Usage

```
Data.interpolate(
  Data,
  metadata,
  Sample_Time,
  Sample_ID,
  interp_method = "cubic",
  Group_var
)
```

Arguments

`Data`	A data frame where rows represent OTUs/ASVs and columns represent samples Or the output of the function `Data.filter`.
`metadata`	A data frame. Containing information about all samples, including at least the grouping of all samples as well as individual information (`Group` and `ID`), the sampling `Time` point for each sample, and other relevant information.
`Sample_Time`	A character string specifying the column name in `metadata` that contains time information.
`Sample_ID`	A character string specifying the column name in `metadata` that identifies unique samples of each subject.
`interp_method`	A character string specifying the interpolation method to be used by `interp1`. Default is `'cubic'`. Other methods accepted by `interp1` (e.g., `'linear'`) can also be used.
`Group_var`	A character string specifying the column name in `metadata` that indicates group membership.

Details

This function processes the input data and metadata by interating over each unique subject ID defined in `Sample_ID`. For each subject, it subsets and sorts the metadata by `Sample_Time` and constructs a complete time series from the minimum to maximum time values with a step of 1. It then extracts the corresponding data columns and performs interpolation (Using the specified

interp_method, with cubic as the default) on each feature across the full time series. Simultaneously, updated metadata is generated for the interpolated time points, preserving the subject ID and group information as indicated by Group_var. The function returns a list object containing the interpolated data matrix and the corresponding updated metadata.

Value

An object of class "MicrobTiSDA.interpolate" containing:

Interpolated_Data
: A data frame of interpolated abundance data.

Interpolated_Data_metadata
: A data frame of corresponding interpolated metadata.

Author(s)

Shijia Li

Examples

```
# Example data: 5 features across 8 samples with time points from two subjects.
set.seed(123)
Data <- matrix(sample(1:100, 40, replace = TRUE), nrow = 5)
rownames(Data) <- paste0("Feature", 1:5)
colnames(Data) <- paste0("Sample", 1:8)

# Create metadata with time points, sample IDs, and group assignments.
metadata <- data.frame(
  Time = c(1, 3, 5, 7, 2, 4, 6, 8),
  ID = c(rep("Subject1", 4), rep("Subject2", 4)),
  Group = c(rep("A", 4), rep("B", 4)),
  row.names = paste0("Sample", 1:8)
)

# Interpolate the data using cubic interpolation.
interp_results <- Data.interpolate(Data = Data,
                                   metadata = metadata,
                                   Sample_Time = "Time",
                                   Sample_ID = "ID",
                                   interp_method = "cubic",
                                   Group_var = "Group")
```

Data.opp.cor.vis *Visualize Opposing Temporal profiles among Microbial features.*

Description

This function identifies and visualizes OTUs/ASVs that exhibit opposing temporal trends based on a correlation threshold. It computes the correlation matrix of predicted OTU/ASV time-series data, selects those OTUs with correlations below a specified threshold, and generates smoothed temporal profile plots. Optionally, raw data points are overlaid and taxonomic annotations are added if provided.

Usage

```
Data.opp.cor.vis(
  predicted_data,
  pre_processed_data,
  Design_data,
  ng_cor_thres,
  Taxa = NULL,
  plot_dots = TRUE,
  figure_x_scale = 5,
  title_size = 10,
  axis_title_size = 10,
  axis_text_y_size = 8,
  axis_text_x_size = 8,
  legend_title_size = 10,
  legend_text_size = 8,
  dots_size = 0.6
)
```

Arguments

predicted_data	The output data frame from the Pred.data).
pre_processed_data	The transformed data output from the Data.trans function. A pre-processed OTU data frame with sample IDs as row names and OTU IDs as column names.
Design_data	The output data from the Design).
ng_cor_thres	A numeric value specifying the correlation threshold below which OTUs are considered to exhibit opposing trends.
Taxa	A data frame providing taxonomic annotations for microbial species.
plot_dots	Logical; if TRUE, raw data points are overlaid on the temporal curves (default: TRUE).
figure_x_scale	A numeric value specifying the interval for x-axis breaks in the figures (default: 5).
title_size	A numeric value specifying the font size for the plot title (default: 10).
axis_title_size	A numeric value specifying the font size for the axis titles (default: 8).
axis_text_y_size	A numeric value specifying the font size for the y-axis text (default: 5).

`axis_text_x_size`
: A numeric value specifying the font size for the x-axis text (default: `5`).

`legend_title_size`
: A numeric value specifying the font size for legend titles (default: `5`).

`legend_text_size`
: A numeric value specifying the font size for legend text (default: `5`).

`dots_size`
: A numeric value specifying the size of the overlaid raw data points (default: `0.7`).

Details

For each group in the `predicted_data` list, the function first removes the time column and computes a correlation matrix from the predicted OTU data. It then extracts, for each OTU, the subset of OTUs that show a correlation lower than the specified threshold (ng_cor_thres). If an OTU does not have any opposing trends (i.e., all correlations exceed the threshold), it is skipped. For those OTUs meeting the criteria, the function restructures the data into long format and plots the temporal profiles using `geom_smooth` to display the primary trend (solid line) and opposing trends (dashed lines). If `plot_dots` is `TRUE`, the raw data points extracted from the design data are also overlaid. When taxonomic annotations are provided via `Taxa`, OTU labels are augmented with species information. The x-axis is scaled according to `figure_x_scale`, and plot aesthetics (titles, axis text, legends, and dot sizes) can be customized using the respective parameters.

Value

An object of class `DataOppCorVis` which contains the list of each targeted microbial feature.

Author(s)

Shijia Li

`Data.rf.classifier` *Random Forest classification for OTU/ASV Data*

Description

This function implements a random forest classification model tailored for OTU/ASV datasets. It performs data filtering, model training, performance evaluation, cross-validation, and biomarker (important microbial features) selection based on Mean Decrease Accuracy. #' @details The function processes the input OTU count data and corresponding metadata in several steps:

1. **Data Filtering and Preparation:** If a minimum count threshold (`OTU_counts_filter_value`) is provided, OTUs with total counts below this value are removed. The OTU table is then transposed and merged with the metadata, where a specific column (specified by `Group`) indicates the group labels.
2. **Data Partitioning:** The combined dataset is split into training and testing subsets based on the proportion specified by `train_p`.

3. **Model Training:** A random forest classifier is trained on the training data. The function computes the margin scores for the training samples, which are plotted to visualize the model's confidence.
4. **Performance Evaluation:** Predictions are made on both training and testing datasets. Confusion matrices are generated to compare the actual versus predicted classes.
5. **Feature Importance and Cross-Validation:** OTU importance is assessed using Mean Decrease Accuracy. Repeated k-fold cross-validation (default 10-fold repeated `reps` times) is performed to determine the optimal number of OTUs (biomarkers). A cross-validation error curve is plotted, and the user is prompted to input the best number of OTUs based on the plot.

Usage

```
Data.rf.classifier(
  raw_data,
  metadata,
  train_p,
  Group,
  OTU_counts_filter_value = NA,
  reps = 5,
  cv_fold = 10,
  title_size = 10,
  axis_title_size = 8,
  legend_title_size = 8,
  legend_text_size = 6,
  seed = 123
)
```

Arguments

`raw_data`	A numeric matrix or data frame of counts data with OTUs/ASVs as rows and samples as columns.
`metadata`	A data frame. Containing information about all samples, including at least the grouping of all samples as well as individual information (`Group` and `ID`), the sampling `Time` point for each sample, and other relevant information.
`train_p`	A positive decimal. Indicating the percentage of data that goes to training. For example, when `train_p = 0.7`, 70% samples were randomly selected as training dataset. More information see `rfcv`.
`Group`	A string that specifies the columns in the `metadata` for grouping the temporal series samples.
`OTU_counts_filter_value`	An integer, indicating the sum of the minimum abundances of OTUs/ASVs in all samples. If the sum of the abundances that OTU/ASV is below the given positive integer threshold, the OTU/ASV is excluded, and vice versa, it is retained. The default is `NA`.
`reps`	An integer. The number of replications for cross-validation. By default, `reps = 5`. More details see `rfcv`.

`cv_fold` An integer. Number of folds in the cross-validation. By default, `cv_fold = 10`. see `rfcv`

`title_size` Numeric value for the font size of plot titles. Defaults to 10.

`axis_title_size` Numeric value for the font size of axis titles. Defaults to 8.

`legend_title_size` Numeric value for the font size of legend titles. Defaults to 8.

`legend_text_size` Numeric value for the font size of legend text. Defaults to 6.

`seed` Random seed.

Value

An object of class `DataRFClassifier` with the following elements:

Input_data The transposed and (optionally) filtered OTU table.

Predicted_results_on_train_set A vector of predicted group labels for the training set.

Predicted_results_on_test_set A vector of predicted group labels for the test set.

Traindata_confusion_matrix A confusion matrix comparing actual vs. predicted group labels for the training set.

Testdata_confusion_matrix A confusion matrix comparing actual vs. predicted group labels for the test set.

Margin_scores_train A ggplot object displaying the margin scores of the training set samples.

OTU_importance A data frame of OTU importance metrics, sorted by Mean Decrease Accuracy.

Classifier A random forest classifier object trained on the training set.

cross_validation A ggplot object showing the cross-validation error curve as a function of the number of features.

Author(s)

Shijia Li

Examples

```
# Example OTU count data (20 OTUs x 10 samples)
set.seed(123)
otu_data <- matrix(sample(0:100, 200, replace = TRUE), nrow = 20)
colnames(otu_data) <- paste0("Sample", 1:10)
rownames(otu_data) <- paste0("OTU", 1:20)

# Example metadata with group labels
metadata <- data.frame(Group = rep(c("Control", "Treatment"), each = 5))

# Run the classifier
result <- Data.rf.classifier(raw_data = otu_data,
                             metadata = metadata,
                             train_p = 0.7,
```

```
Group = "Group",
OTU_counts_filter_value = 50)
```

`Data.trans` *Transform Microbial Composition Data.*

Description

This function applies the modified centered log-ratio (MCLR) transformation function `mclr.transform` to a data matrix (e.g., OTU/ASV counts). When a grouping variable is provided the transformation is applied separately for each group defined in the metadata.

Usage

```
Data.trans(Data, metadata, Group_var)
```

Arguments

`Data`	A data frame or matrix of microbial compositional data, with rows representing microbial features (OTUs/ASVs) and columns representing samples. If the function `Data.interpolate` is preformed, the first element of the output object of `Data.interpolate` should be extract as the input of this function.
`metadata`	A data frame. Containing information about all samples, including at least the grouping of all samples as well as individual information (`Group` and `ID`), the sampling `Time` point for each sample, and other relevant information.
`Group_var`	A string or a vector. This specifies the grouping variables, which should match the column names in the `metadata` used to designate sample groups, and for pre-processing OTU data of each group or individual separately. For instance, to split the OTU table based on the `"Group"` variable, set `Group_var = "Group"`; to split the data based on the `"Group"` and `"Diet"` (if in `metadata`)categorical variables to study the interaction between different grouping variables, set `Group_var = c("Group","Diet")`.

Details

The function transforms the input data using the MCLR method. If no grouping variable is provided (i.e. `Group_var` is `NULL`), the transformation is applied to the entire dataset. If a single grouping variable is specified, the data is partitioned into subsets corresponding to the unique groups in the metadata, and the transformation is applied to each subset separately; the results are then combined using row binding. For multiple grouping variables, a composite grouping factor is created using the interaction of the specified variables, and the transformation is applied to each composite group in a similar manner.

Value

An object of class `"TransformedData"` containing the transformed count table.

Author(s)

Shijia Li

Examples

```
# Create example data matrix (5 features x 10 samples)
set.seed(123)
Data <- matrix(sample(1:100, 50, replace = TRUE), nrow = 5)
rownames(Data) <- paste0("Feature", 1:5)
colnames(Data) <- paste0("Sample", 1:10)

# Create example metadata with a grouping variable
metadata <- data.frame(Group = rep(c("A", "B"), each = 5))
rownames(metadata) <- paste0("Sample", 1:10)

# Apply MCLR transformation to the entire dataset
transformed_data <- Data.trans(Data, metadata, Group_var = NULL)

# Apply MCLR transformation separately for each group
transformed_data_by_group <- Data.trans(Data, metadata, Group_var = "Group")
```

Data.visual *Visualize Temporal OTU Profiles from Clustered Predicted Data*

Description

The Data.visual function generates visualizations of temporal profiles for OTUs by integrating clustering results, predicted time-series data, and design information. It produces ggplot2 figures for each group and for each cluster branch, displaying smoothed curves of predicted OTU abundances over time. Optionally, the function overlays raw data points and fits linear models to assess temporal trends, annotating the plots with model statistics when certain criteria are met.

Usage

```
Data.visual(
  cluster_results,
  cutree_by = "height",
  cluster_height = NA,
  cluster_branches = NA,
  predicted_data,
  Design_data,
  pre_processed_data,
  Taxa = NULL,
  plot_dots = TRUE,
  figure_x_scale = 5,
  plot_lm = FALSE,
```

```
  lm_R2 = 0.01,
  lm_abs_slope = 0.005,
  title_size = 10,
  axis_title_size = 8,
  axis_y_size = 5,
  axis_x_size = 5,
  lm_sig_size = 5,
  legend_title_size = 5,
  legend_text_size = 5,
  dots_size = 0.7
)
```

Arguments

cluster_results	A list object output from the Data.cluster).
cutree_by	A character string specifying the method to cut the dendrogram, either by "height" or by "branches".
cluster_height	A numeric vector specifying the cut-off height for each group when cutree_by = "height".
cluster_branches	A numeric vector specifying the number of clusters for each group when cutree_by = "branches".
predicted_data	The output data frame from the Pred.data).
Design_data	The output data from the Design).
pre_processed_data	The transformed data output from the Data.trans function. A pre-processed OTU data frame with sample IDs as row names and OTU IDs as column names.
Taxa	A data frame providing taxonomic annotations for microbial species.
plot_dots	Logical; if TRUE, raw data points are overlaid on the temporal curves (default: TRUE).
figure_x_scale	A numeric value specifying the interval for x-axis breaks in the figures (default: 5).
plot_lm	Logical; if TRUE, a linear model is fitted to the predicted data to detect trends, and the regression line is added (default: FALSE).
lm_R2	A numeric threshold for the minimum R-squared value required to annotate the linear model (default: 0.01).
lm_abs_slope	A numeric threshold for the minimum absolute slope required to annotate the linear model (default: 0.005).
title_size	A numeric value specifying the font size for the plot title (default: 10).
axis_title_size	A numeric value specifying the font size for the axis titles (default: 8).
axis_y_size	A numeric value specifying the font size for the y-axis text (default: 5).
axis_x_size	A numeric value specifying the font size for the x-axis text (default: 5).

`lm_sig_size` A numeric value specifying the font size for linear model annotation text (default: 5).

`legend_title_size` A numeric value specifying the font size for legend titles (default: 5).

`legend_text_size` A numeric value specifying the font size for legend text (default: 5).

`dots_size` A numeric value specifying the size of the overlaid raw data points (default: 0.7).

Details

This function uses hierarchical clustering results (obtained from a dendrogram) to cut the tree either by a specified height or by a user specified number of branches of each dendrogram in `cluster_results`. For each group in `cluster_results`, the function extracts the corresponding predicted OTU data and raw design data. Temporal profiles are visualized by plotting smooth curves (using `stat_smooth`) for each cluster branch. When `plot_dots` is set to `TRUE`, the function overlays raw data points. Additionally, if `plot_lm` is `TRUE`, a linear model is fitted to the predicted data, and if the model meets specified thresholds for R-squared (`lm_R2`) and absolute slope (`lm_abs_slope`) (i.e., R2 > 0.1 and absolute slope > 0.05), a dashed regression line is added along with an annotation of the R-squared and slope values. The resulting list of ggplot2 objects can be used to visually inspect the temporal dynamics of OTUs across different clusters and groups.

Value

An object of class `MicrobTiSDA.visual` which contains the list of visualizations of clustered microbial features.

Examples

```
metadata <- data.frame(
  TimePoint = c(1, 2, 3, 4),
  Sample = c('S1', 'S2', 'S3', 'S4'),
  GroupA = c('A', 'A', 'B', 'B'),
  GroupB = c('X', 'Y', 'X', 'Y')
)

# Example pre-processed data (e.g., transformed abundance data)
Pre_processed_Data <- data.frame(
  Feature1 = rnorm(4),
  Feature2 = rnorm(4)
)

# Create design matrix using grouping variables
design_data <- Design(metadata, Group_var = c('GroupA', 'GroupB'), Pre_processed_Data,
                      Sample_Time = 'TimePoint', Sample_ID = 'Sample')

reg <- Reg.SPLR(design_data,
                 Pre_processed_Data,
                 z_score = 2,
                 unique_values = 5,
```

```
                  Knots = NULL,
                  max_Knots = 5)
predictions <- Pred.data(reg,
                         metadata,
                         Group = "GroupA",
                         time_step = 1,
                         Sample_Time = "TimePoint")
result <- Data.cluster(predicted_data = predictions,
                       clust_method = "average",
                       font_size = 0.2,
                       dend_title_size = 15)

result <- Data.cluster.cut(cluster_outputs = result,
                           cut_height = 0.3,
                           cut_height_dist = 0.2,
                           auto_cutree = FALSE)

curves <- Data.visual(cluster_results = result,
                      cutree_by = "height",
                      cluster_height = c(0.2,0.2),
                      cluster_branches = NA,
                      predicted_data = predictions,
                      Design_data = design_data,
                      pre_processed_data = Pre_processed_Data,
                      Taxa = NULL,
                      plot_dots = TRUE)
```

`Data.visual.MESR` *Visualize Group-Level OTU Temporal Profiles from Clustered Predicted Data*

Description

This function visualizes the temporal patterns of microbial features at the group level, specifically tailored for data derived from mixed-effects spline regression (MESR) analyses. It leverages clustering results to segregate features into clusters based on their temporal trends, and then generates smoothed time-series plots for each cluster.

Usage

```
Data.visual.MESR(
  cluster_results,
  cutree_by = "height",
  cluster_height = NA,
  cluster_branches = NA,
  predicted_data,
  Design_data,
  pre_processed_data,
```

```
  Taxa = NULL,
  plot_dots = TRUE,
  figure_x_scale = 10,
  plot_lm = TRUE,
  lm_R2 = 0.01,
  lm_abs_slope = 0.005,
  title_size = 10,
  axis_title_size = 8,
  axis_y_size = 5,
  axis_x_size = 5,
  lm_sig_size = 5,
  legend_title_size = 5,
  legend_text_size = 5,
  dots_size = 0.7
)
```

Arguments

cluster_results	A list object output from the Data.cluster).
cutree_by	A character string specifying the method to cut the dendrogram, either by "height" or by "branches".
cluster_height	A numeric vector specifying the cut-off height for each group when cutree_by = "height".
cluster_branches	A numeric vector specifying the number of clusters for each group when cutree_by = "branches".
predicted_data	The output data frame from the Pred.data.MESR).
Design_data	The output data from the Design).
pre_processed_data	The transformed data output from the Data.trans function. A pre-processed OTU data frame with sample IDs as row names and OTU IDs as column names.
Taxa	A data frame providing taxonomic annotations for microbial species.
plot_dots	Logical; if TRUE, raw data points are overlaid on the temporal curves (default: TRUE).
figure_x_scale	A numeric value specifying the interval for x-axis breaks in the figures (default: 5).
plot_lm	Logical; if TRUE, a linear model is fitted to the predicted data to detect trends, and the regression line is added (default: FALSE).
lm_R2	A numeric threshold for the minimum R-squared value required to annotate the linear model (default: 0.01).
lm_abs_slope	A numeric threshold for the minimum absolute slope required to annotate the linear model (default: 0.005).
title_size	A numeric value specifying the font size for the plot title (default: 10).

`axis_title_size`
: A numeric value specifying the font size for the axis titles (default: `8`).

`axis_y_size`
: A numeric value specifying the font size for the y-axis text (default: `5`).

`axis_x_size`
: A numeric value specifying the font size for the x-axis text (default: `5`).

`lm_sig_size`
: A numeric value specifying the font size for linear model annotation text (default: `5`).

`legend_title_size`
: A numeric value specifying the font size for legend titles (default: `5`).

`legend_text_size`
: A numeric value specifying the font size for legend text (default: `5`).

`dots_size`
: A numeric value specifying the size of the overlaid raw data points (default: `0.7`).

Details

The function begins by selecting branches from hierarchical clustering objects (provided in `cluster_results`) using either a specified cut-off height or a predefined number of clusters, as determined by the `cutree_by` parameter. For each group, it extracts the corresponding raw data from `Design_data` and determines the y-axis limits based on both the pre-processed data and the predicted data. Then, for each cluster within a group, the function subsets the predicted data to include only those features belonging to that cluster. If taxonomic annotation data (`Taxa`) is provided, feature names are augmented with species-level labels. The data is then reshaped into a long format and plotted using `ggplot2`, where smoothed curves (via `stat_smooth`) depict the predicted temporal profiles. Optionally, raw data points can be overlaid (if `plot_dots` is `TRUE`), and a linear model is fitted to each cluster's data to test for significant trends. When the linear model meets criteria based on p-value (< 0.05), R^2 (greater than `lm_R2`), and a minimum absolute slope (greater than `lm_abs_slope`), a dashed regression line is added with an annotation indicating the trend direction (upward or downward) along with the R^2 and slope values. Various parameters allow customization of plot appearance including axis scales, font sizes, and legend properties.

Value

An object of class `MicrobTiSDA.MSERvisual` which contains lists of ggplot2 objects, where each top-level element corresponds to a group and each sub-element corresponds to a cluster within that group. Each plot visualizes the temporal profiles of microbial features in that cluster.

`Design` *Create Design Matrix for Regression Analysis*

Description

`Design` creates the design matrix of dummies for fitting regression models of microbiota in time.

Usage

```
Design(metadata, Group_var = NULL, Pre_processed_Data, Sample_Time, Sample_ID)
```

Arguments

metadata	A data frame containing information for all samples, which should be identical to the metadata received by other functions in MicrobTiSDA.
Group_var	A string or a vector. Same as the Group_var in Data.trans.
Pre_processed_Data	The transformed data output from the Data.trans function. A pre-processed OTU data frame with sample IDs as row names and OTU IDs as column names.
Sample_Time	A character string indicating the column name in metadata that contains sample time information.
Sample_ID	A character string indicating the column name in metadata that contains sample identifiers.

Details

The main functionality of Design is to add user-selected sample information to the pre-processed OTU/ASV table as independent variables for fitting the OTU time series regression models. One necessary independent variable for fitting is Time, so the default output of this function is the transformed OTU/ASV table with added sample Time information. If the user also inputs other qualitative variables such grouping, gender, etc., the function will define dummy variables to distinguish each group based on the number of qualitative variables entered by the user and the grouping situation of samples based on qualitative variables. Moreover, the subject ID of each sample will be added as a column to the generated design matrix.

Value

An object of class Design, which contains:

data	A data frame ready for regression modeling.
params	A list of parameters used to construct the design.

Author(s)

Shijia Li

Examples

```
# Example metadata with grouping variables
metadata <- data.frame(
  TimePoint = c(1, 2, 3, 4),
  Sample = c('S1', 'S2', 'S3', 'S4'),
  GroupA = c('A', 'A', 'B', 'B'),
  GroupB = c('X', 'Y', 'X', 'Y')
)

# Example pre-processed data (e.g., transformed abundance data)
Pre_processed_Data <- data.frame(
  Feature1 = rnorm(4),
  Feature2 = rnorm(4)
)
```

```
# Create design matrix using grouping variables
design_data <- Design(metadata, Group_var = c('GroupA', 'GroupB'), Pre_processed_Data,
                      Sample_Time = 'TimePoint', Sample_ID = 'Sample')

# Create design data without grouping variables
design_data_no_group <- Design(metadata, Group_var = NULL, Pre_processed_Data,
                               Sample_Time = 'TimePoint', Sample_ID = 'Sample')
```

`fujita.data` *Example dataset 2 - The in vitro aquatic microbiota dataset.*

Description

A count table used as an example for MicrobTiSDA functions.

Usage

```
fujita.data
```

Format

A data frame with 28 rows (ASVs) and 880 columns (samples).

Examples

```
data("fujita.data")
head("fujita.data")
```

`fujita.meta` *Example dataset 2 - sample metadata.*

Description

Metadata corresponding to the samples in counts of the in vitro aquatic microbiota dataset.

Usage

```
fujita.meta
```

Format

A data frame with 880 rows (samples) and 8 columns (variables).

Examples

```
data("fujita.meta")
head("fujita.meta")
```

fujita.taxa *Example dataset 2 - taxonomic information.*

Description

Taxonomic classification for the ASVs in the in vitro aquatic microbiota dataset.

Usage

```
fujita.taxa
```

Format

A data frame with 565 rows (ASVs) and 9 columns (taxonomy levels).

Examples

```
data("fujita.taxa")
head("fujita.taxa")
```

Interact.dyvis *Dynamic Visualization of Microbial Interaction Networks*

Description

This function visualizes microbial species interactions based on the given inter-species interaction results by Spec.interact.

Usage

```
Interact.dyvis(
  Interact_data,
  threshold,
  core_arrow_num,
  Taxa = NULL,
  fontsize = 15
)
```

Arguments

Interact_data	A list containing the inferred species interaction results generated by Spec.interact.
threshold	Indicates the minimum interaction coefficient used to visualize species interactions. The default value is 1e-6.

core_arrow_num	Indicates the number of species pairs involved in species interactions that identify keystone species. The default value is 4, meaning that the screened keystone species must interact with three other species populations in addition to its own population.
Taxa	A data frame providing taxonomic annotations for microbial species.
fontsize	Indicates the size of the text font. The default is 15.

Details

The function processes microbial interaction matrices from Spec.interact to construct species interaction networks. It selects edges with absolute interaction strengths above threshold, assigns interaction types (positive or negative). The function supports onpional taxonomic annotations for improved interpretability.

The nodes represent microbial species,. Edge colors indicate interaction types: blue for negative and orange for positive interactions. Edge widths are scaled by interaction strength. If taxonomic annotations are provided, node labels reflect species-level taxonomy; otherwise, OTU/ASV IDs are used.

Value

An S3 objects of species interact information or plots.

Author(s)

Shijia Li

---

mclr.transform *Modified Centered Log-Ratio (MCLR) Transformation*

Description

Applies a modified centered log-ratio (MCLR) transformation to compositional data. This transformation is particularly useful in microbiome and compositional data analysis, as it normalizes the data by comparing each value to the geometric mean of the positive values in its row.

Usage

```
mclr.transform(Z, base = exp(1), eps = 0.1)
```

Arguments

Z	A numeric matrix or data frame containing the compositional data to be transformed.
base	A numeric value specifying the logarithmic base to use (default is exp(1), i.e., the natural logarithm).
eps	A small positive constant added to the transformed data to ensure positivity and avoid zeros (default is 0.1).

Details

The MCLR method calculates the geometric mean of each sample from positive proportions only, normalized and log-transformation all non-zero components in the dataset. Specifically, let $x_{nt} \in \Omega^I$ denotes the compositional vector for the sample from subject *n* at timepoint *t*, where Ω^I represents the collection of *I* microbial features.For simplicity of illustration, assume that the first *q* elements of x_{nt} are zero while the remaining elements are non-zero. Then itcan be expressed as:

$$mclr_{\epsilon}(x_{nt}) = [0, \ldots, 0, \ln\left(\frac{x_{nt(q+1)}}{\tilde{g}(x_{nt})}\right) + \epsilon, \ldots, \ln\left(\frac{x_{ntI}}{\tilde{g}(x_{nt})}\right) + \epsilon]$$

where $\tilde{g}(x_{nt}) = \left(\prod_{i=q+1}^{p} x_{nti}\right)^{\frac{1}{I-q}}$ is the geometric mean of the non-zero elements of x_{nt}. When $\varepsilon = 0$, mclr_0 corresponds to the centered log-ratio transform applied to non-zero proportions only. When $\varepsilon > 0$, $\text{mclr}_{\varepsilon}$ applies a positive shift to all non-zero compositions. To make all non-zero values strictly positive, by default $\varepsilon = 0.1$. The MCLR transformation is invariant to the addition of extra zero components, preserves the original zero measurements, and is overal rank preserving. For more details, see Yoon et al. (2019).

Value

A data matrix of the same size as `Z` after the modified centered log-ratio transformation.

References

Yoon, Grace, Irina Gaynanova, and Christian L. Müller. "Microbial networks in SPRING-Semi-parametric rank-based correlation and partial correlation estimation for quantitative microbiome data." Frontiers in Genetics 10 (2019).

Examples

```
# Example compositional data matrix
Z <- matrix(c(1, 2, 0, 4, 5, 6, 0, 8, 9), nrow = 3, byrow = TRUE)
transformed_Z <- mclr.transform(Z, base = 10, eps = 0.1)
```

`OSLO.infant.data` *Example dataset 1 - The OSLO term infant gut microbiota dataset.*

Description

A count table used as an example for MicrobTiSDA functions.

Usage

```
OSLO.infant.data
```

Format

A data frame with 476921 rows (OTUs) and 77 columns (samples).

Examples

```
data("OSLO.infant.data")
head("OSLO.infant.data")
```

`OSLO.infant.meta` *Example dataset 1 - sample metadata.*

Description

Metadata corresponding to the samples in counts of the OSLO Term infant gut microbiota dataset.

Usage

```
OSLO.infant.meta
```

Format

A data frame with 77 rows (samples) and 4 columns (variables).

Examples

```
data("OSLO.infant.meta")
head("OSLO.infant.meta")
```

`OSLO.infant.taxa` *Example dataset 1 - taxonomic information.*

Description

Taxonomic classification for the OTUs in the OSLO Term infant gut microbiota dataset.

Usage

```
OSLO.infant.taxa
```

Format

A data frame with 476921 rows (OTUs) and 8 columns (taxonomy levels).

Examples

```
data("OSLO.infant.taxa")
head("OSLO.infant.taxa")
```

plot.DataOppCorVis *Plot Method for DataOppCorVis Objects*

Description

This function prints correlation curve plots stored in a `DataOppCorVis` object. Users can specify a group and optionally a specific feature within that group to plot.

Usage

```
## S3 method for class 'DataOppCorVis'
plot(x, group, feature, ...)
```

Arguments

`x`	A `DataOppCorVis` object containing correlation curve plots in the `curves_plot` list element.
`group`	A character string specifying the group to plot. Must be provided.
`feature`	A character string specifying the feature within the group to plot. If omitted, all features within the specified group will be plotted.
`...`	Additional arguments

Value

The input `DataOppCorVis` visualization object.

plot.DataRFClassifier *Plot Method for DataRFClassifier Objects*

Description

This function plots the margin scores and cross-validation curve of a `DataRFClassifier` object.

Usage

```
## S3 method for class 'DataRFClassifier'
plot(x, ...)
```

Arguments

`x`	A `DataRFClassifier` object containing margin scores and cross-validation curve plots.
`...`	Additional arguments.

Value

The input `DataRFClassifier` object, return invisibly.

plot.MicrobTiSDA.cluster

Plot Method for MicrobTiSDA Cluster Objects

Description

This function prints cluster plots stored in a `MicrobTiSDA.cluster` object. Users can specify which groups to plot; if `groups` is `NULL`, all available cluster plots will be displayed.

Usage

```
## S3 method for class 'MicrobTiSDA.cluster'
plot(x, groups = NULL, ...)
```

Arguments

`x`	A `MicrobTiSDA.cluster` object containing cluster plots in the `cluster_figures` list element.
`groups`	A character vector specifying the names of groups to plot. If `NULL`, all groups in `x$cluster_figures` will be plotted.
`...`	Additional arguments (currently not used) for compatibility with generic `plot` methods.

Value

The input `MicrobTiSDA.cluster` object, return invisibly.

plot.MicrobTiSDA.clusterCut

Plot Method for MicrobTiSDA Cluster Cut Objects

Description

This function prints cluster plots stored in a `MicrobTiSDA.clusterCut` object. Users can specify which groups to plot; if `groups` is `NULL`, all available cluster plots will be displayed.

Usage

```
## S3 method for class 'MicrobTiSDA.clusterCut'
plot(x, groups = NULL, ...)
```

Arguments

x	x A MicrobTiSDA.clusterCut object containing cluster plots in the cluster_figures list element.
groups	A character vector specifying the names of groups to plot. If NULL, all groups in x$cluster_figures will be plotted.
...	Additional arguments (currently not used) for compatibility with generic plot methods.

Value

The input MicrobTiSDA.clusterCut object, return invisibly.

---

plot.MicrobTiSDA.MSERvisual

Plot Method for MicrobTiSDA MSER Visualization Objects

Description

This function prints feature cluster plots stored in a MicrobTiSDA.MSERvisual object. Users can specify which groups and which clusters within each group to plot. If groups or clusters is NULL, all groups or clusters are plotted.

Usage

```
## S3 method for class 'MicrobTiSDA.MSERvisual'
plot(x, groups = NULL, clusters = NULL, ...)
```

Arguments

x	A MicrobTiSDA.MSERvisual object containing feature cluster plots in the plots list element.
groups	A character vector specifying the names of groups to plot. If NULL, all groups in x$plots will be plotted.
clusters	An integer vector specifying which clusters to plot within each selected group. If NULL, all clusters are plotted.
...	Additional arguments

Value

The input MicrobTiSDA.MSERvisual object, return invisibly.

plot.MicrobTiSDA.visual

Plot Method for MicrobTiSDA Visual Objects

Description

This function prints feature cluster plots stored in a `MicrobTiSDA.visual` object. Users can specify which groups and which clusters within each group to plot. If `groups` is `NULL`, all groups will be displayed. If `clusters` is `NULL`, all clusters within the selected groups will be plotted.

Usage

```
## S3 method for class 'MicrobTiSDA.visual'
plot(x, groups = NULL, clusters = NULL, ...)
```

Arguments

`x`	A `MicrobTiSDA.visual` object containing feature cluster plots in the `plots` list element.
`groups`	A character vector specifying the names of groups to plot. If `NULL`, all groups in `x$plots` will be plotted.
`clusters`	An integer vector specifying which clusters to plot within each selected group. If `NULL`, all clusters are plotted.
`...`	Additional arguments (currently not used) for compatibility with generic `plot` methods.

Value

The input `MicrobTiSDA.visual` object, return invisibly.

plot.microbTiSDA_dynamic_vis

Plot Method for microbTiSDA Dynamic Visualization Objects

Description

This function iterates over all visualization elements stored in a `microbTiSDA_dynamic_vis` object and prints each plot with its name.

Usage

```
## S3 method for class 'microbTiSDA_dynamic_vis'
plot(x, groups = NULL, ...)
```

Arguments

x	A microbTiSDA_dynamic_vis object containing plots in the visualization list element.
groups	A character vector specifying the names of groups to plot. if NULL, all groups in x$visualization will be plotted.
...	Additional arguments

Value

The input microbTiSDA_dynamic_vis object, returned invisibly.

---

plot.RfBiomarker *Plot Method for RfBiomarker Objects*

Description

This function prints the cross-validation figure of a RfBiomarker object and displays basic information about the object contents.

Usage

```
## S3 method for class 'RfBiomarker'
plot(x, ...)
```

Arguments

x	A RfBiomarker object containing selected important OTUs, predictions on training and test sets, and a cross-validation figure.
...	Additional arguments

Value

The input RfBiomarker object, return invisibly.

`Pred.data` *Predict Time-Series Data from Fitted Spline Models*

Description

This function generates predicted OTU abundances at new time points using previously fitted spline regression models. For each subject defined in the fitted models, it extracts the time range from the metadata, creates a new sequence of time points with a specified interval, and uses the corresponding fitted models to predict values at these new time points.

Usage

```
Pred.data(Fitted_models, metadata, Group, time_step, Sample_Time)
```

Arguments

`Fitted_models`	A list object generated by `Reg.SPLR`.
`metadata`	A data frame. Containing information about all samples, including at least the grouping of all samples as well as individual information (`Group` and `ID`), the sampling `Time` point for each sample, and other relevant information.
`Group`	A character string indicating the column name in `metadata` that defines subject information for prediction.
`time_step`	A numeric value specifying the interval between new time points in the prediction sequence.
`Sample_Time`	A character string indicating the column name in `metadata` that contains sample time information.

Details

The function accepts a list of fitted models (typically the output from `Reg.SPLR`) and sample metadata. For each subject (as defined by the `Group` parameter), it extracts the subset of metadata corresponding to that subject. The minimum and maximum time values are determined, and a new sequence of time points is generated using the provided `time_step`. For each OTU model of the subject, predictions are obtained via the `predict` function applied to the new time points, and the results are compiled into a data frame with an additional column `Predicted_Time` indicating the prediction time. The output is a list of data frames, each corresponding to a subject, containing the predicted OTU abundances.

Value

An object of class `PredictedData` which contains the predict microbial feature abundances at the new time points, with rows labeled by the group and time (formatted as `Group_T_Time`) and an additional column `Predicted_Time`.

Author(s)

Shijia Li

Examples

```
# Example metadata with grouping variables
metadata <- data.frame(
  TimePoint = c(1, 2, 3, 4),
  Sample = c('S1', 'S2', 'S3', 'S4'),
  GroupA = c('A', 'A', 'B', 'B'),
  GroupB = c('X', 'Y', 'X', 'Y')
)

# Example pre-processed data (e.g., transformed abundance data)
Pre_processed_Data <- data.frame(
  Feature1 = rnorm(4),
  Feature2 = rnorm(4)
)

# Create design matrix using grouping variables
design_data <- Design(metadata, Group_var = c('GroupA', 'GroupB'), Pre_processed_Data,
                      Sample_Time = 'TimePoint', Sample_ID = 'Sample')

reg <- Reg.SPLR(design_data,
                  Pre_processed_Data,
                  z_score = 2,
                  unique_values = 5,
                  Knots = NULL,
                  max_Knots = 5)
predictions <- Pred.data(reg,
                         metadata,
                         Group = "GroupA",
                         time_step = 1,
                         Sample_Time = "TimePoint")
```

`Pred.data.MESR` *Predict Time-Series Data from Fitted Mixed-Effect Spline regression Models*

Description

This function generates predicted OTU abundances at new time points using previously fitted spline regression models. For each group defined in the fitted models, it extracts the time range from the metadata, creates a new sequence of time points with a specified interval, and uses the corresponding fitted models to predict values at these new time points.

Usage

```
Pred.data.MESR(Fitted_models, metadata, Group, time_step, Sample_Time)
```

Arguments

Fitted_models	A list object generated by Reg.MESR.
metadata	A data frame. Containing information about all samples, including at least the grouping of all samples as well as individual information (Group and ID), the sampling Time point for each sample, and other relevant information.
Group	A character string indicating the column name in metadata that defines group information for prediction.
time_step	A numeric value specifying the interval between new time points in the prediction sequence.
Sample_Time	A character string indicating the column name in metadata that contains sample time information.

Details

The function accepts a list of fitted models (typically the output from Reg.MESR) and sample metadata. For each group (as defined by the Group parameter), it extracts the subset of metadata corresponding to that group. The minimum and maximum time values are determined, and a new sequence of time points is generated using the provided time_step. For each OTU model of the group, predictions are obtained via the predict function applied to the new time points, and the results are compiled into a data frame with an additional column Predicted_Time indicating the prediction time. The output is a list of data frames, each corresponding to a group, containing the predicted OTU abundances.

Value

An object of PredictedDataMESR. Each element of the list corresponds to a group and contains the predicted OTU abundances at the new time points, with rows labeled by the group and time (formatted as Group_T_Time) and an additional column Predicted_Time.

Author(s)

Shijia Li

Examples

```
metadata <- data.frame(
  TimePoint = c(1, 2, 3, 4),
  Sample = c('S1', 'S2', 'S3', 'S4'),
  GroupA = c('A', 'A', 'B', 'B'),
  GroupB = c('X', 'Y', 'X', 'Y')
)

# Example pre-processed data (e.g., transformed abundance data)
Pre_processed_Data <- data.frame(
  Feature1 = rnorm(4),
  Feature2 = rnorm(4)
)

# Create design matrix using grouping variables
```

```
design_data <- Design(metadata, Group_var = c('GroupA', 'GroupB'), Pre_processed_Data,
                      Sample_Time = 'TimePoint', Sample_ID = 'Sample')

fit_result <- Reg.MESR(Data_for_Reg = design_data,
                       pre_processed_data = Pre_processed_Data,
                       unique_values = 5,
                       z_score = 2,
                       Knots = NULL,
                       max_Knots = 5)

predictions <- Pred.data.MESR(fit_result,
                              metadata,
                              Group = "Group",
                              time_step = 1,
                              Sample_Time = "TimePoint")
```

`preterm.data` *Example dataset 3 - The preterm infant gut microbiota dataset.*

Description

A count table used as an example for MicrobTiSDA functions.

Usage

```
preterm.data
```

Format

A data frame with 202 rows (ASVs) and 51 columns (samples).

Examples

```
data("preterm.data")
head("preterm.data")
```

`preterm.meta` *Example dataset 3 - sample metadata.*

Description

Metadata corresponding to the samples in counts of the in vitro aquatic microbiota dataset.

Usage

```
preterm.meta
```

Format

A data frame with 51 rows (samples) and 36 columns (variables).

Examples

```
data("preterm.meta")
head("preterm.meta")
```

`preterm.taxa` *Example dataset 3 - taxonomic information.*

Description

Taxonomic classification for the ASVs in the in vitro aquatic microbiota dataset.

Usage

```
preterm.taxa
```

Format

A data frame with 923 rows (ASVs) and 7 columns (taxonomy levels).

Examples

```
data("preterm.taxa")
head("preterm.taxa")
```

`print.DataOppCorVis` *Information of Microbial features with opposite temporal shapes to users selected feature*

Description

Information of Microbial features with opposite temporal shapes to users selected feature

Usage

```
## S3 method for class 'DataOppCorVis'
print(x, ...)
```

Arguments

`x`	An object of class DataOppCorVis
`...`	Additional arguments

Value

The input `DataOppCorVis` object, returned invisibly.

`print.DataRFClassifier` *Information of the constructed Random Forest classification model*

Description

Information of the constructed Random Forest classification model

Usage

```
## S3 method for class 'DataRFClassifier'
print(x, ...)
```

Arguments

`x`	An object of class DataRFClassifier
`...`	Additional arguments

Value

The input `DataRFClassifier` object, returned invisibly.

`print.Design` *Print information of the Design matrix*

Description

Print information of the Design matrix

Usage

```
## S3 method for class 'Design'
print(x, ...)
```

Arguments

`x`	An object of class Design.
`...`	Additional arguments

Value

The input `Design` object, returned invisibly.

print.FilteredData *Print Method for FilteredData Object*

Description

Print Method for FilteredData Object

Usage

```
## S3 method for class 'FilteredData'
print(x, ...)
```

Arguments

x	A FilteredData object
...	Additional arguments

Value

The input FilteredData object, returned invisibly.

print.microbTiSDA *Print MicrobTiSDA objects*

Description

Print MicrobTiSDA objects

Usage

```
## S3 method for class 'microbTiSDA'
print(x, ...)
```

Arguments

x	An object of class microbTiSDA
...	Additional arguments

Value

The input microbTiSDA object, returned invisibly.

print.MicrobTiSDA.cluster

Print the information of fitted temporal profile clusters

Description

Print the information of fitted temporal profile clusters

Usage

```
## S3 method for class 'MicrobTiSDA.cluster'
print(x, ...)
```

Arguments

x	An object of class MicrobTiSDA.cluster
...	Additional arguments

Value

The input `MicrobTiSDA.cluster` object, returned invisibly.

print.MicrobTiSDA.clusterCut

Print the plot information of user selected feature clustering figures

Description

Print the plot information of user selected feature clustering figures

Usage

```
## S3 method for class 'MicrobTiSDA.clusterCut'
print(x, ...)
```

Arguments

x	An object of class MicrobTiSDA.clusterCut
...	Additional arguments

Value

The input `MicrobTiSDA.clusterCut` object, returned invisibly.

print.MicrobTiSDA.interpolate

Print MicrobTiSDA.interpolate object

Description

Print MicrobTiSDA.interpolate object

Usage

```
## S3 method for class 'MicrobTiSDA.interpolate'
print(x, ...)
```

Arguments

x	An object of class MicrobTiSDA.interpolate
...	Additional arguments

Value

The input MicrobTiSDA.interpolate object, returned invisibly.

print.MicrobTiSDA.MSERvisual

Information of visualizations of feature temporal patterns fitted with mixed-effect spline regression models

Description

Information of visualizations of feature temporal patterns fitted with mixed-effect spline regression models

Usage

```
## S3 method for class 'MicrobTiSDA.MSERvisual'
print(x, ...)
```

Arguments

x	An object of class MicrobTiSDA.MESRvisual
...	Additional arguments

Value

The input MicrobTiSDA.MSERvisual object, returned invisibly.

print.MicrobTiSDA.visual

Print the information of those clustered microbial features' temporal patterns

Description

Print the information of those clustered microbial features' temporal patterns

Usage

```
## S3 method for class 'MicrobTiSDA.visual'
print(x, ...)
```

Arguments

x	An object of class MicrobTiSDA.visual
...	Additional arguments

Value

The input `MicrobTiSDA.visual` object, returned invisibly.

print.microbTiSDA_dynamic_vis

Print information of species interaction dynamic visualizations

Description

Print information of species interaction dynamic visualizations

Usage

```
## S3 method for class 'microbTiSDA_dynamic_vis'
print(x, ...)
```

Arguments

x	An object of class `microbTiSDA_dynamic_vis`
...	Additional arguments

Value

The input `microbTiSDA_dynamic_vis` object, returned invisibly.

print.MicrobTiSDA_spline_regression *Print information of fitted natural spline regression models*

Description

Print information of fitted natural spline regression models

Usage

```
## S3 method for class 'MicrobTiSDA_spline_regression'
print(x, ...)
```

Arguments

x	An object of class MicrobTiSDA_spline_regression
...	Additional arguments

Value

The input `MicrobTiSDA_spline_regression` object, returned invisibly.

print.PredictedData *Print the information of fitted regression model predicted data*

Description

Print the information of fitted regression model predicted data

Usage

```
## S3 method for class 'PredictedData'
print(x, ...)
```

Arguments

x	An object of class PredictedData
...	Additional arguments

Value

The input `PredictedData` object, returned invisibly.

print.PredictedDataMESR *Print method for PredictedDataMESR objects*

Description

Print method for PredictedDataMESR objects

Usage

```
## S3 method for class 'PredictedDataMESR'
print(x, ...)
```

Arguments

x	An object of class PredictedDataMESR.
...	Additional arguments.

Value

The input PredictedDataMESR object, returned invisibly.

print.RegMESR *Information of the fitted mixed-effect spline regression models*

Description

Information of the fitted mixed-effect spline regression models

Usage

```
## S3 method for class 'RegMESR'
print(x, ...)
```

Arguments

x	An object of class RegMESR
...	Additional arguments

Value

The input RegMESR object, returned invisibly.

print.TransformedData *Print Method for TransformedData Object*

Description

Print Method for TransformedData Object

Usage

```
## S3 method for class 'TransformedData'
print(x, ...)
```

Arguments

x	A TransformedData object
...	Additional arguments

Value

No return value, called for its side effect of printing a summary to the console.

Reg.MESR *Fit Mixed-Effects Spline Regression Models with GAM for Microbial Features on Group-Level*

Description

This function fits mixed-effect spline regression models using GAM to capture nonlinear temporal trends in microbial community data at the group level. It incorporates random effects for individual IDs within each group and uses natural splines to model the relationship between time and microbial feature abundances.

Usage

```
Reg.MESR(
  Data_for_Reg,
  pre_processed_data,
  unique_values = 5,
  z_score = NA,
  Knots = NULL,
  max_Knots = 3,
  seed = 123
)
```

Arguments

`Data_for_Reg`	The data frame output from the `Design`.
`pre_processed_data`	The transformed data output from the `Data.trans` function. A pre-processed OTU data frame with sample IDs as row names and OTU IDs as column names.
`unique_values`	An integer specifying the minimum number of unique values required in an OTU for spline fitting (default: 5).
`z_score`	A numeric value specifying the z-score threshold for outlier filtering; if `NA`, no outlier filtering is performed (default: `NA`).
`Knots`	An optional numeric vector specifying the knots to use in the spline regression. If `NULL`, the optimal number of knots is determined by minimizing the GCV criterion (default: `NULL`).
`max_Knots`	An integer indicating the maximum number of knots to consider when selecting the optimal spline model (default: 3).
`seed`	Random seed.

Details

This function fits mixed-effects spline regression models. Unlike `Reg.SPLR`, this function captures the overall temporal dynamics of microbial features across different groups. Assuming the dataset contains *m* groups, each with *n* subjects, the model is formulated as:

$$x_{(mi)}(t) = \beta_{m0} + \sum_{k=1}^{K} \beta_{mk} \cdot N_{mk}(t) + b_{n(m)} + \epsilon$$

where $x_{(mi)}(t)$ represents the abundance of microbial feature *i* at time point *t* in group *m*. The random effect $b_{n(m)}$ reflects the departure of individual *n* in group *m* from overall population average effects. The parameter K refers to the number of basis functions, which is equal to the number of knots plus one. (i.e., $K = number of knots + 1$).

The `Reg.MESR` function first extracts independent variables from a design matrix (produced by the `Design`) by removing columns corresponding to the pre-processed OTU data. For each design dummy variable (excluding the first and last columns), the function subsets the data to include only the observations where the dummy variable equals 1. Then, for each OTU in the pre-processed data, it optionally filters out outliers based on a specified z-score threshold. If the number of unique transformed OTU values exceeds a given threshold (`unique_values`), a mixed-effects spline regression model is fitted using a natural spline on the time variable along with a random effect for the sample ID. When the `Knots` parameter is not provided, the function iterates over a range of knot numbers (from 1 to `max_Knots`), selects the optimal model by minimizing the Generalized Cross-Validation (GCV) criterion, and extracts the corresponding knot locations. Alternatively, if `Knots` is provided, it is directly used in model fitting. The resulting fitted models and associated knots information are organized into nested lists and returned.

Value

An object of class `RegMESR` with two elements: `fitted_model` is a nested list containing the fitted mixed-effects GAM models for each design dummy variable and OTU; `knots_info_each_model` is a corresponding nested list with the knots used for each model.

Author(s)

Shijia Li

Examples

```
metadata <- data.frame(
  TimePoint = c(1, 2, 3, 4),
  Sample = c('S1', 'S2', 'S3', 'S4'),
  GroupA = c('A', 'A', 'B', 'B'),
  GroupB = c('X', 'Y', 'X', 'Y')
)

# Example pre-processed data (e.g., transformed abundance data)
Pre_processed_Data <- data.frame(
  Feature1 = rnorm(4),
  Feature2 = rnorm(4)
)

# Create design matrix using grouping variables
design_data <- Design(metadata, Group_var = c('GroupA', 'GroupB'), Pre_processed_Data,
                      Sample_Time = 'TimePoint', Sample_ID = 'Sample')

fit_result <- Reg.MESR(Data_for_Reg = design_data,
                       pre_processed_data = Pre_processed_Data,
                       unique_values = 5,
                       z_score = 2,
                       Knots = NULL,
                       max_Knots = 5)
```

Reg.SPLR *Fit Spline Regression Models with Knot Selection*

Description

The Reg.SPLR function fits natural spline regression models to time-series OTU data for each subgroup defined by design dummy variables. It uses gam to model the relationship between OTU abundance and time, incorporating a z-score based outlier filtering step (optional) and selecting the optimal number of knots via the Generalized Cross-Validation (GCV) criterion when not provided by the user.

Usage

```
Reg.SPLR(
  Data_for_Reg,
  pre_processed_data,
  z_score = NA,
  unique_values = 5,
  Knots = NULL,
```

```
  max_Knots = 5,
  seed = 123
)
```

Arguments

`Data_for_Reg`	The data frame output from the `Design`.
`pre_processed_data`	The transformed data output from the `Data.trans` function. A pre-processed OTU data frame with sample IDs as row names and OTU IDs as column names.
`z_score`	A numeric value specifying the z-score threshold for outlier filtering; if `NA`, no outlier filtering is performed (default: `NA`).
`unique_values`	An integer specifying the minimum number of unique values required in an OTU for spline fitting (default: 5).
`Knots`	An optional numeric vector specifying the knots to use in the spline regression. If `NULL`, the optimal number of knots is determined by minimizing the GCV criterion (default: `NULL`).
`max_Knots`	An integer indicating the maximum number of knots to consider when selecting the optimal spline model (default: 5).
`seed`	Random seed.

Details

`Reg_SPLR` utilize Generalised Additive Model (GAM,see `gam`) to fit natural splines. Here, let $x_{ni}(t)$ denote the transformed abundance of microbial feature i at time point t. To explain the evolution of microbial abundance along the time (t), the following generalized additive model is considered

$$x_{ni}(t) = \beta_{n0} + \sum_{k=1}^{K} \beta_{nk} \cdot N_{nk}(t) + \epsilon_{ni}$$

where β_{n0} is the intercept term, β_{nk} denotes the regression coefficients for the k-th natural spline basis function $N_{nk}(t)$, and ϵ_{ni} is the error term. The parameter K refers to the number of basis functions, which is equal to the number of knots plus one. (i.e., $K = number of knots + 1$).

If a z-score threshold is specified via `z_score`, observations with absolute z-scores exceeding this threshold are removed prior to model fitting. When the OTU has more unique values than specified by `unique_values`, the function searches over a range of knot numbers (from 1 up to `max_Knots`) to select the optimal model based on the GCV criterion; if `Knots` is provided, it is used directly in constructing the natural spline basis. The fitted models and their corresponding knots information are stored in a nested list structure and returned.

Value

An object of class `"MicrobTiSDA_spline_regression"` containing:

`fitted_model`	A nested list of the fitted spline regression models for each design dummy variable and OTU.
`knots_info`	Contains the corresponding knots information for each model.

Examples

```
# Example metadata with grouping variables
metadata <- data.frame(
  TimePoint = c(1, 2, 3, 4),
  Sample = c('S1', 'S2', 'S3', 'S4'),
  GroupA = c('A', 'A', 'B', 'B'),
  GroupB = c('X', 'Y', 'X', 'Y')
)

# Example pre-processed data (e.g., transformed abundance data)
Pre_processed_Data <- data.frame(
  Feature1 = rnorm(4),
  Feature2 = rnorm(4)
)

# Create design matrix using grouping variables
design_data <- Design(metadata, Group_var = c('GroupA', 'GroupB'), Pre_processed_Data,
                      Sample_Time = 'TimePoint', Sample_ID = 'Sample')

result <- Reg.SPLR(design_data,
                   Pre_processed_Data,
                   z_score = 2,
                   unique_values = 5,
                   Knots = NULL,
                   max_Knots = 5)
# Access the fitted model for a particular design dummy variable and OTU:
fitted_model_example <- result$fitted_model[['Group1']][['OTU_name']]
```

`Rf.biomarkers` *Select Biomarkers Based on Random Forest Cross-Validation Results*

Description

This function extracts the top biomarkers from a random forest classification result based on cross-validation and user specified number of microbial features. It updates the cross-validation plot by adding a vertical dashed line at the specified number of features, and then selects the top features (biomarkers) based on their importance ranking.

Usage

```
Rf.biomarkers(rf = rf_results, feature_select_num)
```

Arguments

`rf`	A list containing the results of the random forest classification. Default to `Data.rf.classifier`.

`feature_select_num`
: A numeric value specifying the number of top features (biomarkers) to select. Typically, the numer of specified biomarkers needs to be determined by the user based on the cross-validation result plot output by `Data.rf.classifier`.

Details

The function takes an object (usually the output from `Data.rf.classifier`, which includes a cross-validation plot, an OTU importance table, and the original input data) and a user-specified number of features to select. It then updates the cross-validation plot by adding a vertical dashed line at the position corresponding to the number of selected features. Next, it extracts the top features from the OTU importance table (ordered by Mean Decrease Accuracy) and creates q table of these features from the original microbial feature table. The function returns a list that includes both the transposed biomarker table and the modified cross-validation plot.

Value

An object of class `RfBiomarker` with two elements:

OTU_importance A data frame of the selected biomarkers (transposed feature table).

cross_validation_fig A ggplot object of the cross-validation plot with a vertical dashed line indicating the feature selection cutoff.

Author(s)

Shijia Li

Examples

```
# Example OTU count data (20 OTUs x 10 samples)
set.seed(123)
otu_data <- matrix(sample(0:100, 200, replace = TRUE), nrow = 20)
colnames(otu_data) <- paste0("Sample", 1:10)
rownames(otu_data) <- paste0("OTU", 1:20)

# Example metadata with group labels
metadata <- data.frame(Group = rep(c("Control", "Treatment"), each = 5))

# Run the classifier
rf_result <- Data.rf.classifier(raw_data = otu_data,
                                metadata = metadata,
                                train_p = 0.7,
                                Group = "Group",
                                OTU_counts_filter_value = 50)
# If you wish to select the top 5 features:
result <- Rf.biomarkers(rf = rf_result, feature_select_num = 5)
# View the biomarker table
print(result$OTU_importance)
# View the updated cross-validation plot
print(result$cross_validation_fig)
```

Spec.interact *Species Interaction Inferrences*

Description

This function describes interspecies interactions based on the discrete-time Lotka-Volterral model.

Usage

```
Spec.interact(
  Data,
  metadata,
  Group_var,
  abund_centered_method = "median",
  num_iterations = 10,
  error_threshold = 0.001,
  pre_error = 10000,
  seed = NULL
)
```

Arguments

Data	A matrix or data frame of the transformed species abundance data.
metadata	A data frame. Containing information about all samples, including at least the grouping of all samples as well as individual information (Group and ID), the sampling Time point for each sample, and other relevant information.
Group_var	A character string specifying the column name in metadata that defines the groups for analysis.
abund_centered_method	A character string indicating the method to compute species equilibrium abundance. Accepted values are median (default) and mean.
num_iterations	An integer specifying the number of bagging iterations for the iterative variable selection process. Default is 10.
error_threshold	A numeric value representing the relative error improvement threshold for adding new predictors during bagging iteration. Default is 1e-3.
pre_error	A numeric value specifying the initial (large) error used for comparison in the iterative procedure. Default is 10000.
seed	Random seed, default by NULL.

Details

This function implements the discrete-time Lotka-Volterra model to characterize species interactions in microbiome time-series data. The model describes the abundance (MCLR transformed)

x_{ni} of species i for subject n at time $t + \Delta t$ as:

$$x_{ni}(t + \Delta t) = \eta_{ni}(t)x_{ni}(t) \exp\left(\Delta t \sum_j c_{nij}(x_{nj}(t) - < x_{nj} >)\right)$$

where $< x_{nj} >$ represents the equilibrium abundance of species j, typically defined as the median abundance across samples from the same subject; c_{nij} denotes the interaction coefficient of species j on species i; and $\eta_{ni}(t)$ accounts for log-normally distributed stochastic effects. For computational simplicity, stochastic effects are ignored, Δt is set to 1. Taking the natural logarithm yealds:

$$\ln x_{ni}(t + 1) - \ln x_{ni}(t) = \sum_j c_{nij}(x_{nj}(t) - < x_{nj} >)$$

To improve sparsity and interpretability, the LIMITS algorithm is applied, incorporating stepwise regression and bagging. First, 50% of the samples are randomly selected as the training set while the rest serve as the test set. An initial regression model includes only the self-interaction term:

$$\ln x_{ni}(t + 1) - \ln x_{ni}(t) = c_{nii}(x_{ni}(t) - < x_{ni} >)$$

Stepwise regression then iteratively adds species interaction terms from a candidate set S, forming:

$$\ln x_{ni}(t + 1) - \ln x_{ni}(t) = c_{nii}(x_{ni}(t) - < x_{ni} >) + \sum_{j \in S} c_{nij}(x_{nj}(t) - < x_{nj} >)$$

The inclusion of a new term is determined based on the improvement in mean squared error (MSE) on the test set:

$$\theta = \frac{\text{MSE}_{\text{before}} - \text{MSE}_{\text{after}}}{\text{MSE}_{\text{before}}}$$

If θ exceeds a predefined threshold (default 10^{-3}), the species is included. Bagging is performed over B iterations by repeating the random splitting and stepwise regression, enhancing robustness. The final interaction coefficient matrix is computed as:

$$c_{nij} = \text{median}(c_{nij}^{(1)}, c_{nij}^{(2)}, ..., c_{nij}^{(B)})$$

This approach refines the inferred species interactions while ensuring sparsity.

Value

A S3 object with an element for each group defined by `Group_var`. Each element is a list containing:

interaction_matrices A three-dimensional array of estimated interaction coefficients with dimensions corresponding to features $\times$ features $\times$ iterations.

final_interaction_matrix A two-dimensional matrix of interaction coefficients obtained by taking the median over the iterations.

Author(s)

Shijia Li

Examples

```
# Example usage:
set.seed(123)
Data <- matrix(sample(1:100, 50, replace = TRUE), nrow = 5)
rownames(Data) <- paste0("Feature", 1:5)
colnames(Data) <- paste0("Sample", 1:10)

# Create example metadata with a grouping variable
metadata <- data.frame(Group = rep(c("A", "B"), each = 5))
rownames(metadata) <- paste0("Sample", 1:10)
metadata$Time = rep(c(1,2,3,4,5),2)
metadata$ID = paste("ID",seq(1:10),"")

results <- Spec.interact(Data = as.data.frame(t(Data)),
                         metadata = metadata,
                         Group_var = "Group",
                         abund_centered_method = "median",
                         num_iterations = 5,
                         error_threshold = 1e-3,
                         pre_error = 10000)
```

`summary.Design` *Summary Method for Design Objects*

Description

Summary Method for Design Objects

Usage

```
## S3 method for class 'Design'
summary(object, ...)
```

Arguments

`object`	An object of class `Design`.
`...`	Additional arguments

Value

A summary list of the designed matrix.

summary.FilteredData *Summary Method for FilteredData Object*

Description

Summary Method for FilteredData Object

Usage

```
## S3 method for class 'FilteredData'
summary(object, ...)
```

Arguments

object	A FilteredData object.
...	Additional arguments

Value

A summary list containing dimensions and filtering parameters

summary.microbTiSDA *Summary method for MicrobTiSDA objects Provides a structure summary of a* microbTiSDA *object, including its parameters and results. It prints object class, parameters settings, number of result groups/items, and previews of tabular of list-based result elements.*

Description

Summary method for MicrobTiSDA objects Provides a structure summary of a microbTiSDA object, including its parameters and results. It prints object class, parameters settings, number of result groups/items, and previews of tabular of list-based result elements.

Usage

```
## S3 method for class 'microbTiSDA'
summary(object, max_preview = 5, ...)
```

Arguments

object	A microbTiSDA object created by the package functions
max_preview	Integer. Maximum number of rows and columns to preview for tabular results. Default is 5
...	Additional arguments

Value

A summary list.

`summary.MicrobTiSDA.cluster`
Summary Method for MicrobTiSDA Cluster Objects

Description

Provides a summary of clustering results for objecs of class `MicrobTiSDA.cluster`. It reports the number of OTUs for each group, or indicates if group was excluded.

Usage

```
## S3 method for class 'MicrobTiSDA.cluster'
summary(object, ...)
```

Arguments

`object`	An object of class `MicrobTiSDA.cluster`, typically produced by clustering functions in the **MicrobTiSDA** pacjage.
`...`	Additional arguments.

Value

A summary list of the clustered objects.

`summary.MicrobTiSDA.clusterCut`
Summary Method for MicrobTiSDA ClusterCut OBjects

Description

Provides a summary of cluster cutting results for objects of class `MicrobTiSDA.clusterCut`. It reports the number of clusters for each group, or indicates if no valid clustering was performend.

Usage

```
## S3 method for class 'MicrobTiSDA.clusterCut'
summary(object, ...)
```

Arguments

`object`	An object of class `MicrobTiSDA.clusterCut`, typically produced by cluster cutting function in the **MicrobTiSDA** package.
`...`	Additional arguments

Value

A summary list of the clustered objects.

`summary.MicrobTiSDA.interpolate`
Summary method for MicrobTiSDA.interpolate objects

Description

Provides a structure summary of a `MicrobTiSDA.interpolate` object, including its parameters and results. It prints object class, the information of interpolated microbial count data and updated metadata.

Usage

```
## S3 method for class 'MicrobTiSDA.interpolate'
summary(object, ...)
```

Arguments

`object`	A `MicrobTiSDA.interpolate` object created by the package functions.
`...`	Additional arguments

Value

A summary list of the interpolated data.

`summary.MicrobTiSDA.visual`
Summary Method for MicrobTiSDA Visual Objects

Description

Provides a summary of visualization results for objects of class `MicrobTiSDA.visual`. It reports, for each group, the number of cluster-speficif visualization plot available.

Usage

```
## S3 method for class 'MicrobTiSDA.visual'
summary(object, ...)
```

Arguments

`object`	An object of class `MicrobTiSDA.visual`, typically produced by visualization functions in the **MicrobTiSDA** package.
`...`	Additional arguments

Value

A list of ggplot2 objects.

`summary.microbTiSDA_dynamic_vis`
Summary Method for MicrobTiSDA Dynamic Interaction Visualization Objects

---

Description

Provides a concise summary of the dynamic interaction visualization results stored in a 'microbTiSDA_dynamic_vis' object. The summary includes the number of nodes and edges in each group.

Usage

```
## S3 method for class 'microbTiSDA_dynamic_vis'
summary(object, ...)
```

Arguments

`object`	An object of class `microbTiSDA_dynamic_vis`, typically created by visualization functions in the **MicrobTiSDA** package.
`...`	Additional arguments.

Value

A summary list of the dynamic visualization for interspecies interactions.

`summary.MicrobTiSDA_spline_regression`
Summary Method for MicrobTiSDA Spline Regression Objects

Description

Provides a concise summary for objects of class `MicrobTiSDA_spline_regression`. Prints the names of independent variables and shows an example of the first fitted microbial feature.

Usage

```
## S3 method for class 'MicrobTiSDA_spline_regression'
summary(object, ...)
```

Arguments

object	An object of class MicrobTiSDA_spline_regression, usually created by spline regression functions in the **MicrobTiSDA** package.
...	Additional arguments.

Value

A summary list of the fitted natural spline regression models.

---

summary.PredictedData *Summary Method for PredictedData Objects*

Description

Provides a concise summary of a PredictedData object, including number of groups, number of time points, number of features, and the time range for each group.

Usage

```
## S3 method for class 'PredictedData'
summary(object, ...)
```

Arguments

object	A PredictedData object returned by Pred.data.
...	Additional arguments.

Value

A summary list of the model predicted data.

---

summary.PredictedDataMESR *Summary method for PredictedDataMESR objects*

Description

Summary method for PredictedDataMESR objects

Usage

```
## S3 method for class 'PredictedDataMESR'
summary(object, ...)
```

Arguments

object	An object of class PredictedDataMESR.
...	Additional arguments.

Value

A summary list of the model predicted data.

---

summary.RegMESR *Summary method for RegMESR objects*

Description

Provides a concise summary of the results from a Regularized Multivariate Exponential Spline Regression (RegMESR) analysis, including the number of groups fitted, the total number of fitted models, and the model parameters.

Usage

```
## S3 method for class 'RegMESR'
summary(object, ...)
```

Arguments

object	An object of class "RegMESR".
...	Additional arguments

Value

A summary list of the fitted mixed-effect natural spline regression models.

---

summary.RfBiomarker *Summary method for RfBiomarker objects*

Description

Provides a concise summary of the results from a Random Forest biomarker analysis, including the number of selected OTUs, their names, prediction results on training and test sets, and information about cross-validation.

Usage

```
## S3 method for class 'RfBiomarker'
summary(object, ...)
```

Arguments

object	An object of class "RfBiomarker".
...	Additional arguments

Value

Summary information of the constructed random forest classification model.

---

summary.TransformedData

Summary Method for TransformedData Objects

Description

Summary Method for TransformedData Objects

Usage

```
## S3 method for class 'TransformedData'
summary(object, ...)
```

Arguments

object	A TransformedData object returned by Data.trans().
...	Additional arguments.

Value

A summary list of the transformed data.

Index