

# Package ‘MeLSI’

May 15, 2026

**Type** Package

**Title** Metric Learning for Statistical Inference in Microbiome Analysis

**Version** 1.0.0

**Description** MeLSI (Metric Learning for Statistical Inference) is a novel machine learning method for microbiome data analysis that learns optimal distance metrics to improve statistical power in detecting group differences. Unlike traditional distance metrics (Bray-Curtis, Euclidean, Jaccard), MeLSI adapts to the specific characteristics of your dataset to maximize separation between groups. The method uses an ensemble of weak learners to identify which microbial features drive group differences, providing both improved statistical power and biological interpretability through feature importance weights.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Depends** R (>= 4.5.0)

**URL** <https://github.com/NathanBresette/MeLSI>

**BugReports** <https://github.com/NathanBresette/MeLSI/issues>

**Imports** vegan, ggplot2, phyloseq, stats, utils

**Suggests** testthat, knitr, rmarkdown, BiocManager, BiocStyle,  
BiocParallel, Matrix, microbiome

**VignetteBuilder** knitr

**biocViews** Software, StatisticalMethod, Microbiome

**git\_url** <https://git.bioconductor.org/packages/MeLSI>

**git\_branch** RELEASE\_3\_23

**git\_last\_commit** ce5c7ee

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.23

**Date/Publication** 2026-05-14

**Author** Nathan Bresette [aut, cre] (ORCID: <https://orcid.org/0009-0003-1554-6006>),  
 Aaron C. Ericsson [aut] (ORCID: <https://orcid.org/0000-0002-3053-7269>),  
 Carter Woods [aut] (ORCID: <https://orcid.org/0009-0007-5345-2712>),  
 Ai-Ling Lin [aut, fnd] (ORCID: <https://orcid.org/0000-0002-5197-2219>)

**Maintainer** Nathan Bresette <nathanbresette04@gmail.com>

## Contents

clr_transform . . . . .	2
generate_test_data . . . . .	3
melsi . . . . .	4
plot_feature_importance . . . . .	5
plot_pcoa . . . . .	6
plot_vip . . . . .	7
<b>Index</b>	<b>8</b>

---

clr_transform	<i>CLR Transformation for Microbiome Data</i>
---------------	---

---

## Description

Applies centered log-ratio (CLR) transformation to microbiome count data. This transformation is recommended for microbiome data before running MeLSI.

## Usage

```
clr_transform(X, pseudocount = 1)
```

## Arguments

X	Feature matrix (samples x taxa) with raw counts or relative abundances
pseudocount	Small constant to add before log transformation (default: 1)

## Value

CLR-transformed matrix with preserved column names

## Examples

```
# Generate synthetic data
test_data <- generate_test_data(n_samples = 20, n_taxa = 30, n_signal_taxa = 5)
X <- test_data$counts

# Transform microbiome data
X_clr <- clr_transform(X)
```

```
# Verify transformation
stopifnot(is.matrix(X_clr))
stopifnot(nrow(X_clr) == nrow(X))
```

---

generate\_test\_data      *Generate Synthetic Microbiome Data for Testing*

---

## Description

Creates synthetic microbiome count data with specified characteristics for testing and demonstration purposes.

## Usage

```
generate_test_data(  
  n_samples = 60,  
  n_taxa = 100,  
  n_signal_taxa = 10,  
  effect_size = 2,  
  group_balance = 0.5  
)
```

## Arguments

n_samples	Total number of samples to generate
n_taxa	Total number of taxa (features)
n_signal_taxa	Number of taxa with differential abundance between groups
effect_size	Magnitude of differential effect (fold-change)
group_balance	Proportion of samples in group A (default 0.5 for balanced)

## Value

A list containing:

- counts: Matrix of count data (samples x taxa)
- metadata: Data frame with sample metadata including Group
- signal\_taxa: Vector of indices for taxa with true differential abundance

## Examples

```
# Generate balanced dataset with 60 samples and 100 taxa
test_data <- generate_test_data(n_samples = 60, n_taxa = 100, n_signal_taxa = 10)
X <- test_data$counts
y <- test_data$metadata$Group
```

---

melsi

*Run MeLSI Analysis*


---

## Description

Performs MeLSI (Metric Learning for Statistical Inference) analysis for microbiome data. Automatically handles both pairwise comparisons (2 groups) and multi-group analysis (3+ groups).

## Usage

```
melsi(
  X,
  y,
  analysis_type = "auto",
  n_perms = 200,
  B = 30,
  m_frac = 0.8,
  show_progress = TRUE,
  plot_vip = TRUE,
  correction_method = "BH",
  BPPARAM = NULL
)
```

## Arguments

X	A matrix of feature abundances with samples as rows and features as columns
y	A vector of group labels for each sample
analysis_type	Type of analysis to perform: - "auto" (default): Automatically choose based on number of groups - "pairwise": For 2 groups or all pairwise comparisons for 3+ groups - "omnibus": Global analysis for 3+ groups (requires at least 3 groups) - "both": Both omnibus and pairwise for 3+ groups
n_perms	Number of permutations for p-value calculation (default: 200)
B	Number of weak learners in the ensemble (default: 30)
m_frac	Fraction of features to use in each weak learner (default: 0.8)
show_progress	Whether to display progress information (default: TRUE)
plot_vip	Whether to display Variable Importance Plot (default: TRUE)
correction_method	Multiple testing correction method for pairwise comparisons (default: "BH")
BPPARAM	A <a href="#">BiocParallelParam</a> object specifying the parallel backend to use for permutation testing. If NULL (default), permutations run sequentially. Requires the <b>BiocParallel</b> package.

## Value

For 2 groups or pairwise analysis: List with F-statistic, p-value, feature weights, etc. For 3+ groups: List containing omnibus results, pairwise results, or both.

### Examples

```
# Generate test data
test_data <- generate_test_data(n_samples = 40, n_taxa = 50, n_signal_taxa = 5)
X <- test_data$counts
y <- test_data$metadata$Group

# CLR transformation
X_clr <- clr_transform(X)

# Run MeLSI analysis
results <- melsi(X_clr, y, n_perms = 19, B = 10, show_progress = FALSE)

# Check results
stopifnot(is.list(results))
stopifnot("F_observed" %in% names(results))
stopifnot("p_value" %in% names(results))
```

---

plot\_feature\_importance

*Plot Feature Importance from MeLSI Analysis*

---

### Description

Creates a barplot showing the top features ranked by their learned weights

### Usage

```
plot_feature_importance(
  feature_weights,
  top_n = 8,
  main_title = NULL,
  directionality = NULL
)
```

### Arguments

feature_weights	Named vector of feature weights
top_n	Number of top features to display (default: 8)
main_title	Optional title for the plot
directionality	Optional named vector indicating which group has higher abundance for each feature

### Value

A ggplot2 object (invisibly)

## Examples

```
# Generate test data and run MeLSI
test_data <- generate_test_data(n_samples = 30, n_taxa = 20, n_signal_taxa = 5)
X <- test_data$counts
y <- test_data$metadata$Group
X_clr <- clr_transform(X)
results <- melsi(X_clr, y, n_perms = 19, B = 10, show_progress = FALSE)

# Plot feature importance
plot_feature_importance(results$feature_weights, top_n = 10)
```

---

plot\_pcoa

*Plot PCoA from MeLSI Results*

---

## Description

Creates a Principal Coordinates Analysis (PCoA) plot using the learned MeLSI distance matrix.

## Usage

```
plot_pcoa(melsi_results, X, y, title = "PCoA using MeLSI Distance")
```

## Arguments

melsi_results	Results object from melsi() function
X	Original feature matrix (samples x taxa)
y	Group labels vector
title	Optional custom title for the plot (default: "PCoA using MeLSI Distance")

## Value

A ggplot2 object (invisibly)

## Examples

```
# Generate test data and run MeLSI
test_data <- generate_test_data(n_samples = 30, n_taxa = 20, n_signal_taxa = 5)
X <- test_data$counts
y <- test_data$metadata$Group
X_clr <- clr_transform(X)
results <- melsi(X_clr, y, n_perms = 19, B = 10, show_progress = FALSE)

# Plot PCoA
plot_pcoa(results, X_clr, y)
```

---

plot_vip	<i>Plot VIP from MeLSI Results (User-Friendly Wrapper)</i>
----------	--

---

**Description**

Simplified function to plot Variable Importance (VIP) directly from MeLSI results. Automatically extracts feature weights and optionally includes directionality information.

**Usage**

```
plot_vip(melsi_results, top_n = 15, title = NULL, directionality = TRUE)
```

**Arguments**

melsi_results	Results object from melsi() function
top_n	Number of top features to display (default: 15)
title	Optional custom title for the plot
directionality	Whether to include directionality coloring (default: TRUE)

**Value**

A ggplot2 object (invisibly)

**Examples**

```
# Generate test data and run MeLSI
test_data <- generate_test_data(n_samples = 30, n_taxa = 20, n_signal_taxa = 5)
X <- test_data$counts
y <- test_data$metadata$Group
X_clr <- clr_transform(X)
results <- melsi(X_clr, y, n_perms = 19, B = 10, show_progress = FALSE)

# Plot VIP with directionality (default)
plot_vip(results, top_n = 10)
```

# Index

`BiocParallelParam`, [4](#)

`clr_transform`, [2](#)

`generate_test_data`, [3](#)

`melsi`, [4](#)

`plot_feature_importance`, [5](#)

`plot_pcoa`, [6](#)

`plot_vip`, [7](#)