

# Package ‘FAVA’

March 27, 2025

**Title** Quantify Compositional Variability Across Relative Abundance Vectors

**Version** 1.0.9

**Description** Implements the statistic FAVA, an Fst-based Assessment of Variability across vectors of relative Abundances, as well as a suite of helper functions which enable the visualization and statistical analysis of relative abundance data. The 'FAVA' R package accompanies the paper, “Quantifying compositional variability in microbial communities with FAVA” by Morrison, Xue, and Rosenberg (2025) <[doi:10.1073/pnas.2413211122](https://doi.org/10.1073/pnas.2413211122)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**URL** <https://maikemorrison.github.io/FAVA/>,  
[https://maikemorrison.github.io/FAVA/articles/microbiome\\_tutorial.html](https://maikemorrison.github.io/FAVA/articles/microbiome_tutorial.html)

**BugReports** <https://github.com/MaikeMorrison/FAVA/issues>

**Imports** dplyr, ggplot2, rlang, tidyr, stringr

**Suggests** patchwork (>= 1.2.0), rmarkdown, viridis, kableExtra, purrr,  
ape, gridExtra, phyloseq, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Depends** R (>= 2.10)

**LazyData** true

**NeedsCompilation** no

**Author** Maike Morrison [aut, cre, cph]  
(<<https://orcid.org/0000-0003-0430-1401>>)

**Maintainer** Maike Morrison <[maike.morrison@gmail.com](mailto:maike.morrison@gmail.com)>

**Repository** CRAN

**Date/Publication** 2025-03-27 11:50:30 UTC

## Contents

bootstrap_fava . . . . .	2
fava . . . . .	4
fava_norm . . . . .	6
gini_simpson . . . . .	7
gini_simpson_mean . . . . .	8
gini_simpson_pooled . . . . .	10
plot_relabund . . . . .	11
relab_phyloseq . . . . .	14
time_weights . . . . .	15
window_fava . . . . .	16
window_list . . . . .	17
window_plot . . . . .	18
xue_microbiome_sample . . . . .	19
xue_species_info . . . . .	19
xue_species_similarity . . . . .	20
xue_species_tree . . . . .	20

<b>Index</b>	<b>21</b>
--------------	-----------

---

bootstrap_fava	<i>Statistically compare FAVA values between pairs of relative abundance matrices.</i>
----------------	--

---

## Description

bootstrap\_fava uses bootstrapping to statistically compare FAVA values between pairs of relative abundance matrices. bootstrap\_fava takes the same options as fava, so, as with fava, you can separately analyze multiple populations or groups of samples (specify group), and account for similarity among categories (specify S) or uneven weighting of rows (specify w or time). bootstrap\_fava follows the bootstrapping procedure defined by Efron and Tibshirani (1993). Details on the bootstrapping procedure are available in the Methods section of the accompanying paper.

## Usage

```
bootstrap_fava(
  relab_matrix,
  n_replicates = 1000,
  group,
  K = NULL,
  S = NULL,
  w = NULL,
  time = NULL,
  normalized = FALSE,
  seed = NULL,
  alternative = "two.sided"
)
```

**Arguments**

relab_matrix	A data frame with rows containing non-negative entries that sum to 1. Each row represents a sample, each column represents a category, and each entry represents the abundance of that category in the sample. If relab_matrix contains any metadata, it must be on the left-hand side of the matrix, the right K entries of each row must sum to 1, and K must be specified. Otherwise, all entries of each row must sum to 1. relab_matrix must have at least one metadata column describing which group each row belongs to. The name(s) of the group column(s) must be provided in the group parameter.
n_replicates	The number of bootstrap replicate matrices to generate. Default is n_replicates = 1000.
group	A string (or vector of strings) specifying the name(s) of the column(s) that describes which group(s) each row (sample) belongs to. Use if relab_matrix is a single matrix containing multiple groups of samples you wish to compare.
K	Optional; an integer specifying the number of categories in the data. Default is $K = \text{ncol}(\text{relab\_matrix})$ .
S	Optional; a $K \times K$ similarity matrix with diagonal elements equal to 1 and off-diagonal elements between 0 and 1. Entry $S[i, k]$ for $i \neq k$ is the similarity between category $i$ and category $k$ , equaling 1 if the categories are to be treated as identical and equaling 0 if they are to be treated as totally dissimilar. The default value is $S = \text{diag}(\text{ncol}(\text{relab\_matrix}))$ .
w	Optional; a vector of length $I$ with non-negative entries that sum to 1. Entry $w[i]$ represents the weight placed on row $i$ in the computation of the mean abundance of each category across rows. The default value is $w = \text{rep}(1/\text{nrow}(\text{relab\_matrix}), \text{nrow}(\text{relab\_matrix}))$ .
time	Optional; a string specifying the name of the column that describes the sampling time for each row. Include if you wish to weight FAVA by the distance between samples.
normalized	Optional; should normalized FAVA be used? Default is normalized = FALSE; use normalized = TRUE to compute normalized FAVA. FAVA can only be normalized if it is not weighted.
seed	Optional; an integer to be used as a random seed for the simulations.
alternative	Optional; do you want to do a one- or two.sided test? Default is alternative = "two.sided". If you wish to do a one-sided test, specify either alternative = "lesser" or alternative = "greater".

**Value**

A named list containing the following entries:

- **p\_values**: The probability of observing the observed difference in variability between each pair of groups if there were no difference between groups. Computed as the fraction of bootstrap differences greater than or equal to the observed difference. Depends on what alternative is specified ("greater", "lesser", or "two.sided").

- `bootstrap_distribution_plot`: The distribution of bootstrap replicate differences in each variability value. The observed differences are shown in red. The further the red points are from 0, the more significant the statistical difference between groups.
- `observed_stats`: The observed diversity statistics for the groups.
- `bootstrap_stats`: The bootstrap replicate diversity statistics for the groups.

## Examples

```
# Statistically compare values of FAVA between
# subjects in the xue_microbiome_sample data:

boot_out = bootstrap_fava(relab_matrix = xue_microbiome_sample,
                          n_replicates = 2, # use 1000 for a real analysis
                          seed = 1,
                          group = "subject",
                          K = 524,
                          S = xue_species_similarity)

# Table of P-values comparing values of FAVA between group 1 and group 2:
boot_out$P_values

# Plots of the bootstrap distributions of differences in FAVA between each pair of matrices,
# and how the true observed differences (red dots) compare to the distribution.
boot_out$bootstrap_distribution_plot
```

---

fava

*Compute the Fst of a matrix of compositional vectors*

---

## Description

This function computes the population-genetic statistic  $F_{st}$  on any matrix with rows that sum to 1. Values of 0 are achieved when each row is a permutation of  $(1, 0, \dots, 0)$  and at least two categories have non-zero abundance across all rows. The value equals 1 when each row is identical.

## Usage

```
fava(
  relab_matrix,
  K = NULL,
  S = NULL,
  w = NULL,
  time = NULL,
  group = NULL,
  normalized = FALSE
)
```

**Arguments**

<code>relab_matrix</code>	A matrix or data frame with rows containing non-negative entries that sum to 1. Each row represents a sample, each column represents a category, and each entry represents the abundance of that category in the sample. If <code>relab_matrix</code> contains any metadata, it must be on the left-hand side of the matrix, the right K entries of each row must sum to 1, and K must be specified. Otherwise, all entries of each row must sum to 1.
<code>K</code>	Optional; an integer specifying the number of categories in the data. Default is <code>K=ncol(relab_matrix)</code> .
<code>S</code>	Optional; a K x K similarity matrix with diagonal elements equal to 1 and off-diagonal elements between 0 and 1. Entry <code>S[i,k]</code> for <code>i!=k</code> is the similarity between category <code>i</code> and category <code>k</code> , equaling 1 if the categories are to be treated as identical and equaling 0 if they are to be treated as totally dissimilar. The default value is <code>S = diag(ncol(relab_matrix))</code> .
<code>w</code>	Optional; a vector of length I with non-negative entries that sum to 1. Entry <code>w[i]</code> represents the weight placed on row <code>i</code> in the computation of the mean abundance of each category across rows. The default value is <code>w = rep(1/nrow(relab_matrix), nrow(relab_matrix))</code> .
<code>time</code>	Optional; a string specifying the name of the column that describes the sampling time for each row. Include if you wish to weight FAVA by the distance between samples.
<code>group</code>	Optional; a string (or vector of strings) specifying the name(s) of the column(s) that describes which group(s) each row (sample) belongs to. Use if <code>relab_matrix</code> is a single matrix containing multiple groups of samples you wish to compare.
<code>normalized</code>	Optional; should normalized FAVA be used? Default is <code>normalized = FALSE</code> ; use <code>normalized = TRUE</code> to compute normalized FAVA. FAVA can only be normalized if it is not weighted.

**Value**

A numeric value between 0 and 1.

**Examples**

```
# Compute the Fst of
# the following compositional vectors:
q1 = c(1, 0, 0, 0)
q2 = c(0.5, 0.5, 0, 0)
q3 = c(1/4, 1/4, 1/4, 1/4)
q4 = c(0, 0, 1, 0)
relative_abundances = matrix(c(q1, q2, q3, q4),
                             byrow = TRUE, nrow = 4)

fava(relative_abundances)

# Incorporating weights:

# Compute fava ignoring
```

```
# rows 2 and 3
row_weights = c(0.5, 0, 0, 0.5)
fava(relative_abundances, w = row_weights)

# Compute fava assuming that
# categories 1 and 2 are identical:
similarity_matrix = diag(4)
similarity_matrix[1,2] = 1
similarity_matrix[2,1] = 1
fava(relative_abundances, S = similarity_matrix)

# Assume categories 1 and 2 are identical AND
# ignore rows 2 and 4:
row_weights = c(0.5, 0, 0.5, 0)
fava(relative_abundances, w = row_weights, S = similarity_matrix)
```

---

fava\_norm

*Compute the normalized Fst of a matrix of compositional vectors*

---

## Description

This function computes the normalized Fst given the number of rows and the mean abundance of the most abundant category. We employ the normalization employed in the [FSTruct package](#) by Morrison, Alcalá, and Rosenberg (2020) [doi:10.1111/17550998.13647](https://doi.org/10.1111/17550998.13647).

## Usage

```
fava_norm(relab_matrix, K = ncol(relab_matrix))
```

## Arguments

relab_matrix	A matrix or data frame with rows containing non-negative entries that sum to 1. Each row represents a sample, each column represents a category, and each entry represents the abundance of that category in the sample. If relab_matrix contains any metadata, it must be on the left-hand side of the matrix, the right K entries of each row must sum to 1, and K must be specified. Otherwise, all entries of each row must sum to 1.
K	Optional; an integer specifying the number of categories in the data. Default is K=ncol(relab_matrix).

## Value

A numeric value between 0 and 1.

**Examples**

```
# Compute the weighted fava of
# the following compositional vectors:
q1 = c(1, 0, 0, 0)
q2 = c(0.5, 0.5, 0, 0)
q3 = c(1/4, 1/4, 1/4, 1/4)
q4 = c(0, 0, 1, 0)
relative_abundances = matrix(c(q1, q2, q3, q4),
                             byrow = TRUE, nrow = 4)

fava_norm(relative_abundances)
```

gini\_simpson

*Compute the Gini-Simpson index of a compositional vector***Description**

This function computes the Gini-Simpson index, a statistical measure of variability known in population genetics as heterozygosity, of a vector of non-negative entries which sum to 1. The function returns a number between 0 and 1 which quantifies the variability of the vector. Values of 0 are achieved when the vector is a permutation of (1,0,..., 0). The value approaches 1 as the number of categories  $K$  increases when the vector is equal to (1/ $K$ , 1/ $K$ , ..., 1/ $K$ ).

**Usage**

```
gini_simpson(q, K = length(q), S = diag(K))
```

**Arguments**

q	A vector with $K=\text{length}(q)$ non-negative entries that sum to 1.
K	Optional; an integer specifying the number of categories in the data. Default is $K=\text{length}(q)$ .
S	Optional; a $K \times K$ similarity matrix with diagonal elements equal to 1 and off-diagonal elements between 0 and 1. Entry $S[i,k]$ for $i \neq k$ is the similarity between category $i$ and category $k$ , equalling 1 if the categories are to be treated as identical and equaling 0 if they are to be treated as totally dissimilar. The default value is $S = \text{diag}(\text{ncol}(q))$ .

**Value**

A numeric value between 0 and 1.

**Examples**

```
# Compute unweighted Gini-Simpson index:
gini_simpson(q = c(0.4, 0.3, 0.3))

# Compute Gini-Simpson index assuming that
# categories 1 and 2 are identical:
similarity_matrix = diag(3)
similarity_matrix[1,2] = 1
similarity_matrix[2,1] = 1
gini_simpson(q = c(0.4, 0.3, 0.3), S = similarity_matrix)
```

---

gini_simpson_mean	<i>Compute the mean Gini-Simpson index of the rows in a matrix of compositional vectors</i>
-------------------	---

---

**Description**

This function computes the mean Gini-Simpson index, a statistical measure of variability known in population genetics as heterozygosity, of a set of vectors of non-negative entries which sum to 1. The function returns a number between 0 and 1 which quantifies the mean variability of the vectors. Values of 0 are achieved when each vector is a permutation of (1,0,..., 0). The value approaches 1 as the number of categories  $K$  increases when the vectors are equal to  $(1/K, 1/K, \dots, 1/K)$ .

**Usage**

```
gini_simpson_mean(
  relab_matrix,
  K = NULL,
  S = NULL,
  w = NULL,
  time = NULL,
  group = NULL
)
```

**Arguments**

relab_matrix	A matrix or data frame with rows containing non-negative entries that sum to 1. Each row represents a sample, each column represents a category, and each entry represents the abundance of that category in the sample. If relab_matrix contains any metadata, it must be on the left-hand side of the matrix, the right K entries of each row must sum to 1, and K must be specified. Otherwise, all entries of each row must sum to 1.
K	Optional; an integer specifying the number of categories in the data. Default is $K = \text{ncol}(\text{relab\_matrix})$ .
S	Optional; a $K \times K$ similarity matrix with diagonal elements equal to 1 and off-diagonal elements between 0 and 1. Entry $S[i, k]$ for $i \neq k$ is the similarity between category $i$ and category $k$ , equalling 1 if the categories are to be

	treated as identical and equaling 0 if they are to be treated as totally dissimilar. The default value is $S = \text{diag}(\text{ncol}(\text{relab\_matrix}))$ .
w	Optional; a vector of length I with non-negative entries that sum to 1. Entry $w[i]$ represents the weight placed on row i in the computation of the mean abundance of each category across rows. The default value is $w = \text{rep}(1/\text{nrow}(\text{relab\_matrix}), \text{nrow}(\text{relab\_matrix}))$ .
time	Optional; a string specifying the name of the column that describes the sampling time for each row. Include if you wish to weight FAVA by the distance between samples.
group	Optional; a string (or vector of strings) specifying the name(s) of the column(s) that describes which group(s) each row (sample) belongs to. Use if <code>relab_matrix</code> is a single matrix containing multiple groups of samples you wish to compare.

**Value**

A numeric value between 0 and 1.

**Examples**

```
# To compute the mean Gini-Simpson index of
# the following compositional vectors...
q1 = c(1, 0, 0, 0)
q2 = c(0.5, 0.5, 0, 0)
q3 = c(1/4, 1/4, 1/4, 1/4)
q4 = c(0, 0, 1, 0)

# we could compute the mean manually:
mean(sapply(list(q1, q2, q3, q4), gini_simpson))

# Or we could use gini_simpson_mean:
relative_abundances = matrix(c(q1, q2, q3, q4),
                             byrow = TRUE, nrow = 4)

gini_simpson_mean(relative_abundances)

# Incorporating weights:

# Compute mean Gini-Simpson index ignoring
# rows 2 and 3
row_weights = c(0.5, 0, 0, 0.5)
gini_simpson_mean(relative_abundances, w = row_weights)

# Compute mean Gini-Simpson index assuming that
# categories 1 and 2 are identical:
similarity_matrix = diag(4)
similarity_matrix[1,2] = 1
similarity_matrix[2,1] = 1
gini_simpson_mean(relative_abundances, S = similarity_matrix)

# Assume categories 1 and 2 are identical AND
# ignore rows 2 and 4:
```

```
row_weights = c(0.5, 0, 0.5, 0)
gini_simpson_mean(relative_abundances, w = row_weights, S = similarity_matrix)
```

---

`gini_simpson_pooled`     *Compute the pooled Gini-Simpson index of the rows in a matrix of compositional vectors*

---

## Description

This function computes the Gini-Simpson index of a "pooled" vector equal to `colMeans(relab_matrix)`. Values of 0 are achieved when this pooled vector is a permutation of (1,0,..., 0). The value approaches 1 as the number of categories  $K$  increases when this pooled vector is equal to (1/ $K$ , 1/ $K$ , ..., 1/ $K$ ).

## Usage

```
gini_simpson_pooled(
  relab_matrix,
  K = NULL,
  S = NULL,
  w = NULL,
  time = NULL,
  group = NULL
)
```

## Arguments

<code>relab_matrix</code>	A matrix or data frame with rows containing non-negative entries that sum to 1. Each row represents a sample, each column represents a category, and each entry represents the abundance of that category in the sample. If <code>relab_matrix</code> contains any metadata, it must be on the left-hand side of the matrix, the right $K$ entries of each row must sum to 1, and $K$ must be specified. Otherwise, all entries of each row must sum to 1.
<code>K</code>	Optional; an integer specifying the number of categories in the data. Default is <code>K=ncol(relab_matrix)</code> .
<code>S</code>	Optional; a $K \times K$ similarity matrix with diagonal elements equal to 1 and off-diagonal elements between 0 and 1. Entry <code>S[i,k]</code> for $i \neq k$ is the similarity between category $i$ and category $k$ , equalling 1 if the categories are to be treated as identical and equaling 0 if they are to be treated as totally dissimilar. The default value is <code>S = diag(ncol(relab_matrix))</code> .
<code>w</code>	Optional; a vector of length $I$ with non-negative entries that sum to 1. Entry <code>w[i]</code> represents the weight placed on row $i$ in the computation of the mean abundance of each category across rows. The default value is <code>w = rep(1/nrow(relab_matrix), nrow(relab_matrix))</code> .
<code>time</code>	Optional; a string specifying the name of the column that describes the sampling time for each row. Include if you wish to weight FAVA by the distance between samples.

**group** Optional; a string (or vector of strings) specifying the name(s) of the column(s) that describes which group(s) each row (sample) belongs to. Use if `relab_matrix` is a single matrix containing multiple groups of samples you wish to compare.

### Value

A numeric value between 0 and 1.

### Examples

```
# To compute the pooled Gini-Simpson index of
# the following compositional vectors...
q1 = c(1, 0, 0, 0)
q2 = c(0.5, 0.5, 0, 0)
q3 = c(1/4, 1/4, 1/4, 1/4)
q4 = c(0, 0, 1, 0)

# we could compute the mean manually:
qPooled = (q1 + q2 + q3 + q4)/4
gini_simpson(qPooled)

# Or we could use gini_simpson_pooled:
relative_abundances = matrix(c(q1, q2, q3, q4),
                             byrow = TRUE, nrow = 4)

gini_simpson_pooled(relative_abundances)

# Incorporating weights:

# Compute pooled Gini-Simpson index ignoring
# rows 2 and 3
row_weights = c(0.5, 0, 0, 0.5)
gini_simpson_pooled(relative_abundances, w = row_weights)

# Compute pooled Gini-Simpson index assuming that
# categories 1 and 2 are identical:
similarity_matrix = diag(4)
similarity_matrix[1,2] = 1
similarity_matrix[2,1] = 1
gini_simpson_pooled(relative_abundances, S = similarity_matrix)

# Assume categories 1 and 2 are identical AND
# ignore rows 2 and 4:
row_weights = c(0.5, 0, 0.5, 0)
gini_simpson_pooled(relative_abundances, w = row_weights, S = similarity_matrix)
```

**Description**

This function enables graphical visualization of a matrix of compositional data. In the output plot, each vertical bar represents a single vector; the height of each color in the bar corresponds to the abundance of each category in that vector. Because this function produces a ggplot object, its output can be modified using standard ggplot2 syntax.

**Usage**

```
plot_relabund(
  relab_matrix,
  group = NULL,
  time = NULL,
  w = NULL,
  K = NULL,
  arrange = FALSE
)
```

**Arguments**

relab_matrix	A matrix or data frame with rows containing non-negative entries that sum to 1. Each row represents a sample, each column represents a category, and each entry represents the abundance of that category in the sample. If relab_matrix contains any metadata, it must be on the left-hand side of the matrix, the right K entries of each row must sum to 1, and K must be specified. Otherwise, all entries of each row must sum to 1.
group	Optional; a string specifying the name of the column that describes which group each row (sample) belongs to. Use if matrices is a single matrix containing multiple groups of samples you wish to compare.
time	Optional; a string specifying the name of the column that describes the sampling time for each row. Include if you wish to weight FAVA by the distance between samples.
w	Optional; a vector of length I with non-negative entries that sum to 1. Entry $w[i]$ represents the weight placed on row $i$ in the computation of the mean abundance of each category across rows. The default value is $w = \text{rep}(1/\text{nrow}(\text{relab\_matrix}), \text{nrow}(\text{relab\_matrix}))$ .
K	Optional; an integer specifying the number of categories in the data. Default is $K = \text{ncol}(\text{relab\_matrix})$ .
arrange	Optional; controls horizontal ordering of samples and vertical ordering of categories. If <code>arrange = TRUE</code> or <code>arrange = "both"</code> , samples are ordered by the categories of greatest abundance and categories are ordered in decreasing abundance. If <code>arrange = "vertical"</code> , sample order is unchanged but categories are ordered in decreasing abundance. If <code>arrange = "horizontal"</code> , samples are ordered by the most abundant categories, but category order is unchanged. If <code>arrange</code> is missing or <code>arrange = FALSE</code> , neither order is changed.

**Value**

A ggplot object containing a bar plot visualization of the relative abundance matrix.

**Examples**

```

# Make an example matrix of compositional data
# Each row is an individual. Rows sum to 1.
population_A = matrix(c(
  .5, .3, .2,
  .4, .2, .4,
  .5, .4, .1,
  .6, .1, .3,
  .2, 0, .8
),
nrow = 5,
byrow = TRUE
)

plot_relabund(relab_matrix = population_A,
              K = 3, # How many categories per vector?
              arrange = FALSE
              )
plot_relabund(relab_matrix = population_A,
              K = 3, # How many categories per vector?
              arrange = "horizontal"
              )
plot_relabund(relab_matrix = population_A,
              K = 3, # How many categories per vector?
              arrange = "vertical"
              )
plot_relabund(relab_matrix = population_A,
              K = 3, # How many categories per vector?
              arrange = TRUE # could also be "both"
              )

# You can modify the plot as you would any ggplot2 object
plot_relabund(relab_matrix = population_A,
              K = 3, # How many categories per vector?
              arrange = TRUE
              ) +
# Below are example, optional modifications to the default plot
ggplot2::ggtitle("Population A") +
ggplot2::scale_fill_brewer("Blues") +
ggplot2::scale_color_brewer("Blues") +
ggplot2::xlab("Individuals")
# Note that both scale_fill and scale_color are needed to change the color of the bars.

# Plot a dataset which has 2 populations

population_B = matrix(c(
  .9, 0, .1,
  .6, .4, 0,
  .7, 0, .3,
  .3, .4, .3,

```

```

      .5, .3, .2
    ),
    nrow = 5,
    byrow = TRUE
  )

populations_AB = cbind(data.frame(c("A", "A", "A", "A", "A",
                                   "B", "B", "B", "B", "B")),
                      rbind(population_A, population_B))
colnames(populations_AB) = c("population", "category_1", "category_2", "category_3")

plot_relabund(relab_matrix = populations_AB, group = "population")
plot_relabund(relab_matrix = populations_AB, group = "population", arrange = "vertical")
plot_relabund(relab_matrix = populations_AB, group = "population", arrange = "horizontal")
plot_relabund(relab_matrix = populations_AB, group = "population", arrange = "both")

```

---

relab_phyloseq	<i>Generate a relative abundance matrix with sample metadata and OTU abundances from a phyloseq object.</i>
----------------	---

---

### Description

The R package phyloseq streamlines the storage and analysis of microbiome sequence data. This function takes a phyloseq object and extracts the OTU table and the sample metadata and combines them into one relative abundance matrix with rows corresponding to samples, metadata on the left-hand side, and OTU relative abundances on the right-hand side.

### Usage

```
relab_phyloseq(phyloseq_object)
```

### Arguments

phyloseq\_object

A phyloseq object containing both an OTU table (otu\_table) and sample metadata (sample\_data).

### Value

A data frame with rows representing samples and columns representing sample data categories or OTU relative abundances. OTU abundances are automatically normalized so that they sum to 1 for each sample, though a warning will be provided if a renormalization was necessary.

**Examples**

```

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

  # Make a small phyloseq object for demonstration
  phyloseq_subset = phyloseq::subset_taxa(phyloseq::subset_samples(GlobalPatterns,
                                                                    X.SampleID %in%
                                                                    c("CL3", "CC1")),
                                          Order == "Cenarchaeales")
  otu_table = relab_phyloseq(phyloseq_subset)
  otu_table[, 1:10]
}

```

---

time_weights	<i>Compute a normalized weighting vector based on a vector of sampling times.</i>
--------------	---

---

**Description**

This function takes a vector of sampling times,  $t = (t_1, t_2, \dots, t_I)$  and computes a normalized vector which can be used to weight each sample based on the time between the subsequent and the preceding samples. The weighting vector  $w$  is defined such that each entry,  $w_i = d_i/2T$ , where  $T = t_I - t_1$  and  $d_i = t_{i+1} - t_{i-1}$  for  $i$  not equal to 1 or  $I$ .  $d_1 = t_2 - t_1$  and  $d_I = t_I - t_{I-1}$ .

**Usage**

```
time_weights(times, group = NULL)
```

**Arguments**

times	A numeric vector of sampling times. Each entry must be greater than the previous entry.
group	Optional; a character vector specifying the group identity of each sampling time. Use if there are samples from multiple replicates or subjects in one dataset.

**Value**

A numeric vector. Each entry provides a weight for each entry in the provided times vector. If group is not specified, the vector sums to 1. If group is specified, the vector sums to the number of distinct groups.

**Examples**

```

time_vector = c(1, 8, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
               32, 33, 34, 35, 36, 37, 38, 39, 44, 50, 57, 64)

time_weights(times = time_vector)

```

---

 window\_fava

 Compute FAVA in sliding windows.
 

---

### Description

This function computes FAVA in sliding window slices of a dataset.

### Usage

```

window_fava(
  relab_matrix,
  window_size,
  window_step = 1,
  group = NULL,
  index = NULL,
  time = NULL,
  w = NULL,
  S = NULL,
  K = NULL,
  normalized = FALSE,
  alpha = 0.5
)

```

### Arguments

relab_matrix	A matrix or data frame with rows containing non-negative entries that sum to 1. Each row represents a sample, each column represents a category, and each entry represents the abundance of that category in the sample. If relab_matrix contains any metadata, it must be on the left-hand side of the matrix, the right K entries of each row must sum to 1, and K must be specified. Otherwise, all entries of each row must sum to 1.
window_size	An integer number specifying the number of samples per window.
window_step	Optional; an integer specifying the distance between the first entry of adjacent windows. Default is window_step=1.
group	Optional; a string specifying the name of the column that describes which group each row (sample) belongs to. Use if relab_matrix is a single matrix containing multiple groups of samples you wish to compare.
index	Optional; a string specifying the name of the column in relab_matrix containing an index for each sample. For example, if relab_matrix contains time series data, index would be the column containing the time of each sample. If index is not specified but time is, time is by default used as the index.
time	Optional; a string specifying the name of the column that describes the sampling time for each row. Include if you wish to weight FAVA by the distance between samples.

w	Optional; a vector of length I with non-negative entries that sum to 1. Entry $w[i]$ represents the weight placed on row $i$ in the computation of the mean abundance of each category across rows. The default value is $w = \text{rep}(1/\text{nrow}(\text{relab\_matrix}), \text{nrow}(\text{relab\_matrix}))$ .
S	Optional; a $K \times K$ similarity matrix with diagonal elements equal to 1 and off-diagonal elements between 0 and 1. Entry $S[i, k]$ for $i \neq k$ is the similarity between category $i$ and category $k$ , equaling 1 if the categories are to be treated as identical and equaling 0 if they are to be treated as totally dissimilar. The default value is $S = \text{diag}(\text{ncol}(\text{relab\_matrix}))$ .
K	Optional; an integer specifying the number of categories in the data. Default is $K = \text{ncol}(\text{relab\_matrix})$ .
normalized	Optional; should normalized FAVA be used? Default is <code>normalized = FALSE</code> ; use <code>normalized = TRUE</code> to compute normalized FAVA. FAVA can only be normalized if it is not weighted.
alpha	Optional; number between 0 and 1 specifying the opacity of the horizontal lines plotted. Default is <code>alpha = 0.5</code> .

**Value**

A list of values of FAVA for each window.

**Examples**

```
A = matrix(c(.3,.7,0,.1,0,.9,.2,.5,.3,.1,.8,.1,.3,.4,.3,.6,.4,0,0,.5,.5),
           ncol = 3, byrow = TRUE)
window_out = window_fava(relab_matrix = A, window_size = 4, normalized = TRUE)
```

---

window_list	<i>Generate sliding windows of specified length given the maximum number of samples</i>
-------------	---

---

**Description**

This function generates a list of sliding windows conditional on two parameters: the length of each window (number of samples) and the total number of samples present in the data.

**Usage**

```
window_list(window_size, length, window_step = 1)
```

**Arguments**

window_size	An integer number specifying the number of samples per window.
length	An integer number specifying the total number of samples.
window_step	Optional; an integer number specifying the distance between the first entry of adjacent windows. Default is <code>window_step=1</code> .

**Value**

A list of samples of sample indices. Each list entry represents one window.

**Examples**

```
window_list(window_size = 6, length = 40)
window_list(window_size = 6, length = 40, window_step = 2)
```

---

window\_plot

*Generate a plot of FAVA in sliding windows.*

---

**Description**

This function generates a plot of normalized or unnormalized, weighted or unweighted FAVA computed in sliding windows across samples for one or many groups of samples.

**Usage**

```
window_plot(window_fava, alpha = 0.5)
```

**Arguments**

window\_fava     The output of window\_fava.

alpha            Optional; number between 0 and 1 specifying the opacity of the horizontal lines plotted. Default is alpha = 0.5.

**Value**

A ggplot2 object.

**Examples**

```
A = matrix(c(.3,.7,0,.1,0,.9,.2,.5,.3,.1,.8,.1,.3,.4,.3,.6,.4,0,0,.5,.5),
           ncol = 3, byrow = TRUE)
window_out = window_fava(relab_matrix = A, window_size = 4, normalized = TRUE)
window_out$window_data
window_out$window_plot
```

---

xue\_microbiome\_sample *Temporal microbiome composition data*

---

### Description

A subset of the data generated by Xue et al. (2024) detailing longitudinal composition of the human gut microbiome for three subjects who experience an antibiotic perturbation between days 29 and 34. We include only the subjects XAA, XBA, and XCA. Data from [doi:10.1101/2023.09.26.559480](https://doi.org/10.1101/2023.09.26.559480).

### Usage

xue\_microbiome\_sample

### Format

xue\_microbiome\_sample:

A data frame with 75 rows and 1,348 columns:

**subject** Subject ID: XBA, XDA, or XMA

**timepoint** Time (days) of sample collection

... Species names ...

---

xue\_species\_info *Table of species information*

---

### Description

A data frame providing taxonomic information for the species included in xue\_species\_tree.

### Usage

xue\_species\_info

### Format

xue\_species\_info:

A data frame with 1346 rows and 9 columns:

**species\_id** The species\_id given in xue\_microbiome\_sample

**kingdom, phylum, class, order, family, genus, species** The corresponding taxonomic category for each species

**species\_id\_number** The numeric code associated with each species, as used in xue\_species\_tree

---

xue\_species\_similarity

*Species similarity matrix for the species included in xue\_microbiome\_sample*

---

**Description**

A similarity matrix, with entry (i,j) corresponding to the pairwise similarity between species i and species j. This similarity matrix was derived from a phylogenetic distance matrix, inferred from the tree xue\_species\_tree, using the expression  $s(i,j) = \exp(-d(i,j))$ , where  $d(i,j)$  is the phylogenetic distance between species i and j.

**Usage**

xue\_species\_similarity

**Format**

xue\_species\_similarity:

A data frame with 524 rows and 524 columns, each corresponding to one species.

---

xue\_species\_tree

*Phylogenetic tree for the species included in xue\_microbiome\_sample*

---

**Description**

A phylogenetic tree in the Newick format.

**Usage**

xue\_species\_tree

**Format**

xue\_species\_tree:

A Newick tree.

# Index

## \* datasets

- xue\_microbiome\_sample, 19
- xue\_species\_info, 19
- xue\_species\_similarity, 20
- xue\_species\_tree, 20

bootstrap\_fava, 2

fava, 4

fava\_norm, 6

gini\_simpson, 7

gini\_simpson\_mean, 8

gini\_simpson\_pooled, 10

plot\_relabund, 11

relab\_phyloseq, 14

time\_weights, 15

window\_fava, 16

window\_list, 17

window\_plot, 18

xue\_microbiome\_sample, 19

xue\_species\_info, 19

xue\_species\_similarity, 20

xue\_species\_tree, 20