

# Package ‘mecoturn’

July 24, 2025

**Type** Package

**Title** Decipher Microbial Turnover along a Gradient

**Version** 0.3.1

**Author** Chi Liu [aut, cre],  
Xiangzhen Li [ctb],  
Minjie Yao [ctb]

**Maintainer** Chi Liu <liuchi0426@126.com>

**Description** Two pipelines are provided to study microbial turnover along a gradient, including the beta diversity and microbial abundance change. The 'betaturn' class consists of the steps of community dissimilarity matrix generation, matrix conversion, differential test and visualization. The workflow of 'taxaturn' class includes the taxonomic abundance calculation, abundance transformation, abundance change summary, statistical analysis and visualization. Multiple statistical approaches can contribute to the analysis of microbial turnover.

**URL** <https://github.com/ChiLiubio/mecoturn>

**Depends** R (>= 3.5.0)

**Imports** R6, ggplot2 (>= 3.4.0), microeco (>= 0.20.0), GUniFrac,  
magrittr, ggpubr, lmerTest, betareg, glmmTMB

**Suggests** ape, agricolae

**License** GPL-3

**LazyData** true

**Encoding** UTF-8

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2025-07-24 07:30:02 UTC

**RoxygenNote** 7.3.2

## Contents

|                     |   |
|---------------------|---|
| betaturn . . . . .  | 2 |
| taxaturn . . . . .  | 5 |
| wheat_16S . . . . . | 9 |

## Index

10

---

|          |                                                         |
|----------|---------------------------------------------------------|
| betaturn | <i>Analyze the 'turnover' of microbial communities.</i> |
|----------|---------------------------------------------------------|

---

## Description

Analyze the 'turnover' of microbial communities, i.e. beta-diversity along a gradient <doi:10.1111/j.1461-0248.2010.01552.x>. The workflow consists of the steps of dissimilarity matrix generation, matrix conversion, differential test and visualization.

## Methods

## Public methods:

- `betaturn$new()`
- `betaturn$cal_group_distance()`
- `betaturn$cal_group_distance_diff()`
- `betaturn$plot_group_distance()`
- `betaturn$clone()`

Method `new()`:*Usage:*

```
betaturn$new(
  dataset,
  measure = "bray",
  filter_thres = 0,
  abundance.weighted = TRUE,
  null.model = NULL,
  runs = 1000,
  iterations = 1000,
  ...
)
```

*Arguments:*

`dataset` the object of microtable class of microeco package.

`measure` default "bray"; beta diversity dissimilarity metric; must be one of `c("bray", "jaccard", "wei_unifrac", "unwei_unifrac", "betaMPD", "betaMNTD", "betaNRI", "betaNTI", "ses_UniFrac", "RCbray")` or other options in parameter method of `vegan::vegdist` function. If the distance matrix has been in the `beta_diversity` list of microtable object, the function can ignore this step. Otherwise, the function can generate the corresponding beta diversity distance matrix in the microtable object. `bray`: Bray-Curtis; `RCbray`: Raup-Crick based Bray-Curtis; `wei_unifrac`: weighted UniFrac; `ses_UniFrac`: standardized deviation of UniFrac.

`filter_thres` default 0; the relative abundance threshold used to filter features with low abundance.

`abundance.weighted` default TRUE; whether use abundance-weighted method for the phylogenetic metrics.

`null.model` default NULL; one of `c("taxa.labels", "richness", "frequency", "sample.pool", "phylogeny.pool", "independentswap", "trialswap")`, in which "taxa.labels" can only be used for phylogenetic analysis. See `null.model` parameter of `ses.mntd` function in `picante` package for the algorithm details.

`runs` default 1000; simulation number of times for null model.

`iterations` default 1000; iteration number for part null models to perform; see `iterations` parameter of `picante::randomizeMatrix` function.

... parameters passed to `cal_betadiv` function of `microtable` class when provided measure is not in the current vector; parameters passed to `cal_betamntd` function of `trans_nullmodel` class when `measure = "betaMNTD"`; parameters passed to `cal_ses_betamntd` function of `trans_nullmodel` class when `measure = "betaNTI"`.

**Returns:** dataset, stored in the object. The new dataset has a `beta_diversity` list and the calculated distance matrix in the list.

*Examples:*

```
data(wheat_16S)
```

```
b1 <- betaturn$new(wheat_16S, measure = "bray")
```

**Method** `cal_group_distance()`: Convert sample distances within groups or between groups.

*Usage:*

```
betaturn$cal_group_distance(
  group,
  within_group = TRUE,
  by_group = NULL,
  ordered_group = NULL,
  sep = " vs ",
  add_cols = NULL
)
```

*Arguments:*

`group` one colname of `sample_table` in `microtable` object used for group distance conversion.

`within_group` default TRUE; whether transform sample distance within groups? If FALSE, transform sample distances between any two groups.

`by_group` default NULL; one colname of `sample_table` in `microtable` object. If provided, convert distances according to the provided `by_group` parameter. This is especially useful for ordering and filtering values further. When `within_group = TRUE`, the result of `by_group` parameter is the format of paired groups. When `within_group = FALSE`, the result of `by_group` parameter is the format same with the group information in `sample_table`.

`ordered_group` default NULL; a vector representing the ordered elements of group parameter; only useful when `within_group = FALSE`.

`sep` default TRUE; a character string to separate the group names after merging them into a new name.

`add_cols` default NULL; add several columns of `sample_table` to the final `res_group_distance` table according to the `by_group` column; invoked only when `within_group = FALSE`.

**Returns:** `res_group_distance` stored in object.

*Examples:*

```
b1$cal_group_distance(group = "Type", within_group = FALSE, by_group = "Plant_ID")
```

**Method** `cal_group_distance_diff()`: Differential test of distances among groups.

*Usage:*

```
betaturn$cal_group_distance_diff(...)
```

*Arguments:*

... parameters passed to `cal_group_distance_diff` function of `trans_beta` class in `microeco` package.

*Returns:* `res_group_distance_diff` stored in object.

*Examples:*

```
b1$cal_group_distance_diff(method = "wilcox")
```

**Method** `plot_group_distance()`: Plot the distance between samples within or between groups.

*Usage:*

```
betaturn$plot_group_distance(...)
```

*Arguments:*

... parameters passed to `plot_group_distance` function of `trans_beta` class in `microeco` package.

*Returns:* `ggplot`.

*Examples:*

```
b1$plot_group_distance()
```

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
betaturn$clone(deep = FALSE)
```

*Arguments:*

`deep` Whether to make a deep clone.

## Examples

```
## -----
## Method `betaturn$new`
## -----

data(wheat_16S)
b1 <- betaturn$new(wheat_16S, measure = "bray")

## -----
## Method `betaturn$cal_group_distance`
## -----

b1$cal_group_distance(group = "Type", within_group = FALSE, by_group = "Plant_ID")

## -----
## Method `betaturn$cal_group_distance_diff`
## -----
```

```

b1$cal_group_distance_diff(method = "wilcox")

## -----
## Method `betaturn$plot_group_distance`
## -----

b1$plot_group_distance()

```

---

taxaturn

---

*Analyze the 'turnover' of taxa.*


---

## Description

Analyze the 'turnover' of taxa along a defined gradient. The workflow of taxaturn class includes the taxonomic abundance calculation, abundance transformation, abundance change summary, statistical analysis and visualization.

## Methods

### Public methods:

- `taxaturn$new()`
- `taxaturn$cal_diff()`
- `taxaturn$plot()`
- `taxaturn$clone()`

### Method `new()`:

*Usage:*

```

taxaturn$new(
  dataset,
  taxa_level = "Phylum",
  group,
  ordered_group,
  by_ID = NULL,
  by_group = NULL,
  filter_thres = 0
)

```

*Arguments:*

`dataset` the object of microtable class of microeco package.

`taxa_level` default "Phylum"; taxonomic rank name, such as "Genus". An integer is also acceptable. If the provided `taxa_level` is not found in `taxa_abund` list, the function will invoke the `cal_abund` function to obtain the relative abundance automatically.

`group` sample group used for the selection; a colname of input `microtable$sample_table`.

`ordered_group` a vector representing the ordered elements of `group` parameter.

`by_ID` default NULL; a column of `sample_table` used to obtain the consistent change along provided elements. So `by_ID` can be ID (unique repetition) or even group (with repetitions). If it denotes unique ID, consistent change can be performed across each ID. It is also especially useful for the paired wilcox test (or paired t test) in the following analysis. If it does not represent unique ID, the mean of each group will be calculated, and consistent change across groups will be obtained.

`by_group` default NULL; NULL or other colname of `sample_table` of input dataset used to show the result for different groups; NULL represents the output is the default consistent change across all the elements in `by_ID`; a colname of `sample_table` of input dataset means the consistent change is obtained for each group instead of all the elements in `by_group`; Note that the `by_group` can be same with `by_ID`, in which the final change is the result of each element in `by_group`. So generally `by_group` has a larger scale than `by_ID` parameter in terms of the sample numbers in each element.

`filter_thres` default 0; the mean abundance threshold used to filter features with low abundance.

*Returns:* `res_abund`, `res_change_pair` and `res_change` in the object:

`res_abund` The Mean, SD or SE of abundances for all the samples or each group. Mean: mean of abundances; SD: standard deviation; SE: standard error.

`res_change_pair` The difference value of abundances between two niches, i.e. the latter minus the former.

`res_change` The summary of the abundance change results in `res_change_pair`.

*Examples:*

```
data(wheat_16S)
t1 <- taxaturn$new(wheat_16S, taxa_level = "Phylum", group = "Type",
  ordered_group = c("S", "RS", "R"), by_ID = "Plant_ID", filter_thres = 0.01)
```

**Method** `cal_diff()`: Differential test of taxonomic abundance across groups

*Usage:*

```
taxaturn$cal_diff(
  method = c("wilcox", "t.test", "anova", "betareg", "lme", "glmm")[1],
  group2num = FALSE,
  ...
)
```

*Arguments:*

`method` default "wilcox"; see the following available options:

**'wilcox'** Wilcoxon Rank Sum and Signed Rank Tests for all paired groups

**'t.test'** Student's t-Test for all paired groups

**'anova'** one-way or multi-way anova

**'betareg'** Beta Regression based on the `betareg` package

**'lme'** lme: Linear Mixed Effect Model based on the `lmerTest` package

**'glmm'** Generalized linear mixed model (GLMM) based on the `glmmTMB` package with the beta family function, i.e. `family = glmmTMB::beta_family(link = "logit")`. For more parameters, please see `glmmTMB::glmmTMB` function. In the return table, `Conditional_R2` and `Marginal_R2` represent total variance (explained by both fixed and random effects) and the variance explained by fixed effects, respectively. The significance

of fixed factors are tested by Chi-square test from function `car::Anova`. The significance of 'Estimate' in each term of fixed factors comes from the model.

`group2num` default FALSE; whether convert ordered groups to integer numbers when method is "lme" or "glmm".

... parameters passed to `trans_diff$new`.

*Returns:* `res_change` or `res_diff` in the object.

*Examples:*

```
t1$cal_diff(method = "wilcox")
```

**Method** `plot()`: Plot the line chart.

*Usage:*

```
taxaturn$plot(
  select_taxon = NULL,
  color_values = RColorBrewer::brewer.pal(8, "Dark2"),
  delete_prefix = TRUE,
  plot_type = c("point", "line", "errorbar", "smooth")[1:3],
  errorbar_SE = TRUE,
  rect_fill = TRUE,
  rect_color = c("grey70", "grey90"),
  rect_alpha = 0.2,
  position = position_dodge(0.1),
  errorbar_size = 1,
  errorbar_width = 0.1,
  point_size = 3,
  point_alpha = 0.8,
  line_size = 0.8,
  line_alpha = 0.8,
  line_type = 1,
  ...
)
```

*Arguments:*

`select_taxon` default NULL; a taxon name. Note that if `delete_prefix` is TRUE, the provided `select_taxon` should be taxa names without long prefix (those before `l`); if `delete_prefix` is FALSE, the `select_taxon` should be full names same with those in the `res_abund` of the object.

`color_values` default `RColorBrewer::brewer.pal(8, "Dark2")`; colors palette for the plotting.

`delete_prefix` default TRUE; whether delete the prefix in the taxa names.

`plot_type` default `c("point", "line", "errorbar", "smooth")[1:3]`; a vector of visualization types. Multiple elements are available. 'smooth' denotes the fitting with `geom_smooth` function of `ggplot2` package.

`errorbar_SE` default TRUE; TRUE: plot the errorbar with mean  $\pm$  se; FALSE: plot the errorbar with mean  $\pm$  sd.

`rect_fill` default TRUE; Whether fill color in each rectangular area.

`rect_color` default `c("grey70", "grey90")`; the colors used to fill different rectangular area.

rect\_alpha default 0.2; the fill color transparency in rectangular area.  
 position default position\_dodge(0.1); Position adjustment for the points and lines, either as a string (such as "identity"), or the result of a call to a position adjustment function.  
 errorbar\_size default 1; errorbar size.  
 errorbar\_width default 0.1; errorbar width.  
 point\_size default 3; point size for taxa.  
 point\_alpha default 0.8; point transparency.  
 line\_size default 0.8; line size.  
 line\_alpha default 0.8; line transparency.  
 line\_type default 1; an integer; line type.  
 ... parameters passed to geom\_smooth when 'smooth' is in plot\_type parameter.

*Returns:* ggplot2 plot.

*Examples:*

```
t1$plot()
```

**Method** clone(): The objects of this class are cloneable with this method.

*Usage:*

```
taxaturn$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
## -----
## Method `taxaturn$new`
## -----

data(wheat_16S)
t1 <- taxaturn$new(wheat_16S, taxa_level = "Phylum", group = "Type",
  ordered_group = c("S", "RS", "R"), by_ID = "Plant_ID", filter_thres = 0.01)

## -----
## Method `taxaturn$cal_diff`
## -----

t1$cal_diff(method = "wilcox")

## -----
## Method `taxaturn$plot`
## -----

t1$plot()
```

---

wheat\_16S*The example dataset in the mecoturn package*

---

**Description**

The dataset wheat\_16S is structured with microtable class for the demonstration of examples.

**Usage**

```
data(wheat_16S)
```

**Format**

An R6 class object

**Details**

- sample\_table: sample information table
- otu\_table: species-community abundance table
- tax\_table: taxonomic table
- phylo\_tree: phylogenetic tree

# Index

- \* **R6**
  - wheat\_16S, [9](#)
- \* **microtable**
  - wheat\_16S, [9](#)
- \* **object**
  - wheat\_16S, [9](#)
- betaturn, [2](#)
- taxaturn, [5](#)
- wheat\_16S, [9](#)