Package 'ncmR'

October 21, 2026

Type Package

Title Fit Neutral Community Model to Microbiome or Ecological Data

Version 0.4.0

Date 2026-10-22

Description Provides tools for fitting the neutral community model (NCM) to assess the role of stochastic processes in community assembly. The package implements the framework of Sloan et al. (2006) <doi:10.1111/j.1462-2920.2005.00956.x>, enabling users to evaluate neutral dynamics in ecological and microbial communities.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports bslib (>= 0.9.0), colourpicker, dplyr, DT, ggplot2 (>= 4.0.0), ggtext, grid, Hmisc, minpack.lm, shiny, shinyjs, systemfonts, zip

RoxygenNote 7.3.3

URL https://github.com/h-xuanjiu/ncmR

NeedsCompilation no

Author Yuxuan He [aut, cre, cph] (ORCID: <https://orcid.org/0009-0007-1149-5554>)

Maintainer Yuxuan He <heyuxuan0525@outlook.com>

Repository CRAN

Date/Publication 2026-10-21 17:00:02 UTC

Contents

example_data 2
fit_ncm 2
pie_plot 4
print.summary_ncm 6

print.summary_ncm_group 7
run_app 8
scatter_plot 9
summary.NCM 12
summary.NCM_group 13

Index **14**

`example_data` *Example OTU table and grouping information for NCM demonstration*

Description

A simulated dataset containing an OTU table (ASV counts) and corresponding group labels. The data are intended to illustrate the usage of the `fit_ncm` function.

Usage

```
example_data
```

Format

A list with two components:

otu A data frame with 15 rows (samples) and 2000 columns (ASVs). Row names are sample identifiers, column names are ASV identifiers. Values represent counts of each ASV in each sample.

grp A data frame with 15 rows and 1 column named `group`. Row names correspond to the samples in `otu`. The `group` column contains character labels ("A", "A", ..., etc.) indicating group membership for each sample.

Source

Simulated data for package examples.

`fit_ncm` *Fit Neutral Community Model (NCM) with optional grouping and total pooling*

Description

Fit Neutral Community Model (NCM) with optional grouping and total pooling

Usage

```
fit_ncm(
  otu,
  grp = NULL,
  group_col = "group",
  groups = NULL,
  simplify = TRUE,
  return_model = FALSE,
  ...
)
```

Arguments

otu	OTU table, rows = samples, columns = species (data.frame or matrix)
grp	Optional data.frame with rownames = sample IDs, and one column specifying group membership. If NULL, all samples in otu are used as one group.
group_col	Column name in grp that contains group labels (default "group"). The group label ".ncm_total" is reserved for the combined fit.
groups	Character vector of group names to analyze. If NULL and grp is provided, the function fits models for each group AND for all samples combined (total). If non-NULL, only those groups are analyzed (no total model).
simplify	If TRUE and only one model is fitted, return the model list directly (not nested).
return_model	Whether to return the nlsLM model object (default FALSE to save space).
...	Additional arguments passed to nlsLM (e.g., start, lower, upper, control). A vector of initial values can be supplied as `start = list(m = c(0.01, 0.1))`; values are tried in order until a fit converges.

Details

Only converged fits are returned. Fitting errors and non-convergence trigger another attempt with the next initial value. If all attempts fail, the warning includes the underlying failure reasons. Predicted-frequency Wilson intervals are restricted to the probability range `[0, 1]`. When the predicted frequency is exactly 0 or 1, the corresponding interval endpoint is set exactly to that boundary. This removes floating-point boundary errors before classifying taxa.

Value

A list containing model results. If multiple models, a named list with keys: - ".ncm_total" (if total model fitted) and group names. Each value is a list with: m, N, Nm, ci, rsqr, predictions, (model optional). Numeric group labels are converted to character names. Other group labels, including "all" and "all.1", are preserved. A group named ".ncm_total" causes an error because this name is reserved.

Examples

```
# Load example data
data(example_data)
```

```
otu <- example_data$otu
grp <- example_data$grp

# 1. No grouping: fit the total model using all samples
res_total <- fit_ncm(otu)
print(paste("m =", round(res_total$m, 4)))
print(paste("R^2 =", round(res_total$rsqr, 4)))
head(res_total$predictions)

# 2. With grouping: fit models for all groups + total
res_all <- fit_ncm(otu, grp)
res_all$.ncm_total$m

# 3. Only specific groups (no total model), returns a single model object
res_sub <- fit_ncm(otu, grp, groups = "A")
res_sub$status_summary
```

`pie_plot` *Pie chart of neutral community model classifications*

Description

Show the proportions of taxa classified as Above, Below, or Neutral. Each row represents one taxon; abundance does not weight the proportions. Labels are placed outside the circle with leader lines and separated on each side to keep small neighbouring slices readable.

Usage

```
pie_plot(object, ...)

## S3 method for class 'NCM'
pie_plot(
  object,
  colors = c(Above = "#ED7D70", Below = "#2B889B", Neutral = "#B57FAF"),
  label_type = c("percent", "count", "both", "none"),
  label_size = 4,
  label_color = "#25312e",
  border_color = "white",
  border_width = 0.7,
  start = 0,
  direction = 1,
  legend_title_text = NULL,
  base_theme = NULL,
  theme = NULL,
  ...
)
```

```
## S3 method for class 'data.frame'
pie_plot(
  object,
  colors = c(Above = "#ED7D70", Below = "#2B889B", Neutral = "#B57FAF"),
  label_type = c("percent", "count", "both", "none"),
  label_size = 4,
  label_color = "#25312e",
  border_color = "white",
  border_width = 0.7,
  start = 0,
  direction = 1,
  legend_title_text = NULL,
  base_theme = NULL,
  theme = NULL,
  map = NULL,
  ...
)

## Default S3 method:
pie_plot(object, ...)
```

Arguments

`object`	An NCM result or a data frame with one row per taxon.
`...`	Additional arguments passed to methods. Unsupported arguments produce an error.
`colors`	Three colours for Above, Below, and Neutral. A named vector can specify these colours in any order; an unnamed vector follows that order.
`label_type`	Slice labels: `"percent"`, `"count"`, `"both"`, or `"none"`. Percentages have two decimal places.
`label_size`	Label size in mm (default: 4).
`label_color`	A single colour for labels (default: `"#25312e"`).
`border_color`	Slice border colour (default: `"white"`). Use `NA` to omit borders.
`border_width`	Slice border width in mm (default: 0.7).
`start`	Starting angle in radians, measured clockwise from 12 o'clock (default: 0).
`direction`	Draw clockwise (1, the default) or counterclockwise (-1).
`legend_title_text`	Legend title (default: `NULL`).
`base_theme`	A complete ggplot2 theme. `NULL` uses `ggplot2::theme_void(base_size = 12)`. Axes, panel grids, and the panel border are removed from the base theme for the pie chart.
`theme`	Additional ggplot2 theme settings, applied after the base theme and pie-specific axis settings. These can override any of those settings.
`map`	Optional named character vector mapping `freq`, `lower`, `upper`, or `status` to data-frame column names, for example `c(status = "classification")`.

Details

With freq, lower, and upper columns, taxa are classified exactly as in scatter_plot(): freq > upper is Above, freq < lower is Below, and values on or between the bounds are Neutral. These three columns take precedence over an existing status column. Frequencies and bounds must be finite numbers between 0 and 1, with lower <= upper. Alternatively, a status column may contain the exact values "Above", "Below", and "Neutral". Missing or invalid classifications are rejected rather than counted as Neutral. Confidence bounds may exceed 0 or 1 by up to 64 machine epsilons to allow numerical roundoff in interval calculations. Their original values are retained when classifying taxa, matching scatter_plot().

Absent categories remain in the legend at 0 a slice and an outside label. The legend order is Neutral, Below, Above, matching scatter_plot(). The returned plot's data contains one row per category with status, n, and percentage (on a 0–100 scale). The plot supports normal additions such as + ggplot2::labs(title = "Community composition") and + ggplot2::theme(legend.position = "bottom").

Value

A ggplot object.

Methods (by class)

- pie_plot(NCM): Plot classifications from an NCM result.
- pie_plot(data.frame): Plot a prediction table or a table of classifications.
- pie_plot(default): Default method for unsupported input types.

Examples

```
data(example_data)
model <- fit_ncm(example_data$otu)
pie_plot(model)
pie_plot(model, label_type = "both",
  theme = ggplot2::theme(legend.position = "bottom"))

# A prediction table can also be used without model parameters.
pie_plot(model$predictions)
pie_plot(data.frame(classification = c("Neutral", "Above", "Below")),
  map = c(status = "classification"))
```

---

print.summary_ncm *Print summary of NCM model*

Description

Print summary of NCM model

Usage

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

Arguments

x	An object of class "summary_ncm".
...	Additional arguments (not used).

Value

No return value, called for side effects. Prints a formatted table of NCM results to the console.

---

print.summary_ncm_group

Print summary of NCM group results

Description

Print summary of NCM group results

Usage

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

Arguments

x	An object of class "summary_ncm_group".
...	Additional arguments (not used).

Value

No return value, called for side effects. Prints a formatted table of NCM group results to the console.

run_app *Launch the ncmR Shiny Application*

Description

Starts the interactive Shiny web application for fitting and visualizing Neutral Community Models (NCM). The app provides a user-friendly interface for uploading abundance data, fitting NCM models, and generating publication-quality plots.

Usage

```
run_app(ncm = NULL)
```

Arguments

ncm	Optional fitted NCM or NCM_group object returned by fit_ncm. If supplied, the app loads these results and opens Figure studio. NULL starts the app without fitted results.

Details

The Shiny app includes two main modules:

- **Fit NCM**: Upload abundance data (samples as rows, species as columns) and optional group metadata to fit neutral community models. Supports grouped analyses and provides model parameters, fit statistics, and species-level predictions.
- **Plotting**: Visualize fitting results or upload pre-computed NCM results to generate customizable scatter plots with fitted curves and confidence intervals, or pie charts of Above, Below and Neutral taxon proportions. Includes live preview, common style controls, and a theme inspector limited to elements used by the selected chart. Chart types keep independent saved theme edits. Base theme parameters, individual theme elements, units, margins, and batch edits can be customized. Color pickers and searchable font menus include journal figure families and installed host fonts. Figures and reproducible R recipes can be downloaded.

The app automatically detects sample IDs from the first column and handles large datasets with column pagination. Results can be downloaded as ZIP archives containing summary statistics and predictions.

Value

This function does not return a value; it launches a Shiny application interactively in the default web browser.

See Also

fit_ncm for programmatic NCM fitting, scatter_plot and pie_plot for generating NCM plots in R scripts

Examples

```
## Not run:
# Launch the app
ncmR::run_app()

# Open an existing fitted model directly in Figure studio
data(example_data)
res <- fit_ncm(example_data$otu)
ncmR::run_app(ncm = res)

# Grouped results are also supported
results <- fit_ncm(example_data$otu, example_data$grp)
ncmR::run_app(ncm = results)

## End(Not run)
```

---

scatter_plot *Scatter plot for NCM results and data frames*

Description

Generic function for creating scatter plots. Methods available for NCM result objects and data frames. Legend symbols match the plotted points, with group percentages and default-colored text labels.

Usage

```
scatter_plot(object, ...)

## S3 method for class 'NCM'
scatter_plot(
  object,
  point_alpha = 0.8,
  point_size = 3,
  point_colors = c(Above = "#ED7D70", Below = "#2B889B", Neutral = "#B57FAF"),
  fit_line_color = "#335399",
  fit_line_type = "solid",
  fit_line_size = 1,
  ci_line_color = "#335399",
  ci_line_type = "dashed",
  ci_line_size = 1,
  axis_title_x_text = "Mean relative abundance (log10)",
  axis_title_y_text = "Frequency of occupancy",
  legend_title_text = NA,
  fit_para_size = 6,
  fit_para_position = c(0.02, 0.98),
  fit_para_hjust = 0,
```

```
  fit_para_vjust = 1,
  base_theme = NULL,
  theme = NULL,
  ...
)

## S3 method for class 'data.frame'
scatter_plot(
  object,
  rsqr,
  Nm,
  m,
  map,
  point_alpha = 0.8,
  point_size = 3,
  point_colors = c(Above = "#ED7D70", Below = "#2B889B", Neutral = "#B57FAF"),
  fit_line_color = "#335399",
  fit_line_type = "solid",
  fit_line_size = 1,
  ci_line_color = "#335399",
  ci_line_type = "dashed",
  ci_line_size = 1,
  axis_title_x_text = "Mean relative abundance (log10)",
  axis_title_y_text = "Frequency of occupancy",
  legend_title_text = NA,
  fit_para_size = 6,
  fit_para_position = c(0.02, 0.98),
  fit_para_hjust = 0,
  fit_para_vjust = 1,
  base_theme = NULL,
  theme = NULL,
  ...
)

## Default S3 method:
scatter_plot(object, ...)
```

Arguments

object	An object to plot: NCM result or data.frame
...	Additional arguments passed to methods. Unsupported arguments produce an error.
point_alpha	Alpha transparency for points (default: 0.8)
point_size	Point size (default: 3)
point_colors	Named vector of colors for Above/Below/Neutral status
fit_line_color	Fitted line color (default: "#335399")
fit_line_type	Line type for fitted curve (default: "solid")

`fit_line_size` Line width for fitted curve (default: 1)

`ci_line_color` Confidence interval line color (default: "#335399")

`ci_line_type` Confidence interval line type (default: "dashed")

`ci_line_size` Confidence interval line width (default: 1)

`axis_title_x_text`
X-axis title text

`axis_title_y_text`
Y-axis title text

`legend_title_text`
Legend title text (default: NA)

`fit_para_size` Fitting parameter text size in mm (default: 6)

`fit_para_position`
Parameter annotation position as NPC coordinates c(x, y) (default: c(0.02, 0.98))

`fit_para_hjust` Parameter horizontal justification (default: 0)

`fit_para_vjust` Parameter vertical justification (default: 1)

`base_theme` A complete ggplot2 theme, such as `ggplot2::theme_bw(base_size = 12)` or `ggplot2::theme_classic()`. NULL uses the package's default NCM theme based on theme_bw().

`theme` A ggplot2 theme object containing additional settings, typically created with `ggplot2::theme()`. NULL adds no overrides.

`rsqr` R-squared value from NCM fit

`Nm` Nm parameter from NCM fit

`m` m parameter from NCM fit

`map` Named vector for column name mapping (e.g., c(p = "abundance"))

Details

All theme settings are supplied through `base_theme` and `theme`. The base theme is applied first, followed by `theme`. Settings merge using ggplot2's normal inheritance rules: shared settings such as `axis.title`, `axis.text`, and `text` apply to children that do not have an explicit override. The default NCM theme uses relative text sizes so that global text size changes propagate. Use `axis.title` and `axis.text` for shared axis styles, or their `.x`/`.y` variants for individual axes. Legend size, position, and alignment are controlled by `legend.text`, `legend.title`, `legend.position`, `legend.position.inside`, and `legend.justification.inside`. Set the global font family with `theme(text = element_text(family = ...))` or a complete theme's `base_family` argument. Theme text sizes are in points. Model annotations inherit the final global text family and colour; their size and position remain controlled by the `fit_para_*` arguments.

Value

A ggplot object

Methods (by class)

- `scatter_plot(NCM)`: Plot NCM result with fitted curve, confidence intervals, and parameter annotations. Supports full customization of visual elements.
- `scatter_plot(data.frame)`: Plot data.frame by converting to NCM object. Requires manual specification of model parameters (rsqr, Nm, m) and optional column name mapping.
- `scatter_plot(default)`: Default method for unsupported types.

Examples

```
# Load example data
data(example_data)
otu <- example_data$otu

# fit the total model using all samples
res_total <- fit_ncm(otu)

# plot
scatter_plot(res_total)

# Use a complete base theme and shared axis text settings
scatter_plot(res_total,
  fit_para_size = 3.5,
  base_theme = ggplot2::theme_classic(base_size = 12),
  theme = ggplot2::theme(
    axis.title = ggplot2::element_text(size = 14),
    axis.text = ggplot2::element_text(size = 10),
    legend.position = "bottom"
  )
)

# Load example data
data(example_data)
otu <- example_data$otu

# fit the total model using all samples
res_total <- fit_ncm(otu)

df <- res_total$predictions

# Plot with manual parameters
scatter_plot(df, rsqr = 0.95, Nm = 500, m = 0.5678)
```

---

summary.NCM *Summary method for NCM objects*

Description

Summary method for NCM objects

Usage

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

Arguments

object	An object of class "NCM" (single model).
...	Additional arguments (not used).

Value

An object of class "summary_ncm" containing key model statistics.

---

summary.NCM_group *Summary method for NCM group results*

Description

Summary method for NCM group results

Usage

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

Arguments

object	An object of class "NCM_group" (list of NCM objects).
...	Additional arguments (not used).

Value

An object of class "summary_ncm_group" containing summary for each group.

Index

∗ **datasets**
example_data, 2

example_data, 2

fit_ncm, 2, 2, *8*

pie_plot, 4, *8*
print.summary_ncm, 6
print.summary_ncm_group, 7

run_app, 8

scatter_plot, *8*, 9
summary.NCM, 12
summary.NCM_group, 13