Package ‘mrangr’

January 19, 2026

Title Mechanistic Metacommunity Simulator

Version 1.0.0

Description Flexible, mechanistic, and spatially explicit simulator of metacommunities. It extends our previous package - 'rangr' (see <https://github.com/ropensci/rangr>), which implemented a mechanistic virtual species simulator integrating population dynamics and dispersal. The 'mrangr' package adds the ability to simulate multiple species interacting through an asymmetric matrix of pairwise relationships, allowing users to model all types of biotic interactions — competitive, facilitative, or neutral — within spatially explicit virtual environments.

License MIT + file LICENSE

Depends R (>= 3.5.0)

Imports assertthat, FieldSimR, graphics, grDevices, gstat, methods, mgcv, parallel, rangr, RColorBrewer, stats, terra, utils

Suggests bookdown, knitr, rmarkdown, testthat (>= 3.0.0), tools

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

NeedsCompilation no

Author Katarzyna Markowska [aut, cre, cph],
Lechosław Kuczyński [aut, cph]

Maintainer Katarzyna Markowska <`katarzyna.markowska@amu.edu.pl`>

Repository CRAN

Date/Publication 2026-01-19 18:20:13 UTC

Contents

`a_eg` *Example Of Interaction Coefficients Matrix*

Description

A square numeric matrix representing interaction coefficients between species. `a_ij` is the per-capita interaction strength of species `j` on species `i`. It expresses the change in carrying capacity of species `i` by a single individual of species `j`. This data is compatible with `n1_map_eg.tif` and `K_map_eg.tif` maps.

Usage

```
a_eg
```

Format

A numeric matrix with 4 rows and 4 columns containing interaction coefficients.

Source

Data generated in-house to serve as an example

`community_eg` *Example Community Data*

Description

A pre-initialized `sim_com_data` object used to demonstrate community structure and simulation input. It contains 4 species with spatially correlated carrying capacity and initial abundance maps.

This object can be accessed via the `get_community` function.

Format

An object of class `sim_com_data` from the `mrangr` package generated using `initialise_com`.

Source

Data generated in-house to serve as an example

See Also

`get_community`, `initialise_com`

`diagonal` *Compute Maximum Possible Distance for a Raster Object*

Description

Calculates the diagonal length of a raster's extent, accounting for the coordinate reference system.

Usage

```
diagonal(x)
```

Arguments

`x`	A raster object.

Value

The diagonal distance in meteres.

Examples

```
library(terra)

# Read data from the mrangr package
K_map <- rast(system.file("input_maps/K_map_eg.tif", package = "mrangr"))

diagonal(K_map)
```

`get_community` *Load Example Community Object*

Description

Loads a pre-simulated example of a spatial community object, useful for demos and testing.

Usage

```
get_community()
```

Value

An object of class `sim_com_data` containing community structure, simulation parameters, species-specific carrying capacity and initial abundance maps.

Examples

```
community <- get_community()
summary(community)
```

`get_simulated_com` *Load Example Simulated Community Results*

Description

Loads a pre-run simulation output, based on the example community data. Useful for examples, unit tests, or visualization.

Usage

```
get_simulated_com()
```

Value

An object of class `sim_com_results` containing simulation output for a community over time.

Examples

```
sim <- get_simulated_com()
plot(sim)
```

`grf` *Generate a Gaussian Random Field*

Description

Generates a Gaussian random field (GRF) based on the Matern model of spatial autocorrelation.

Usage

```
grf(x, range, fun = "scale", ...)
```

Arguments

`x`	A template raster of class `SpatRaster` (from the **terra** package).
`range`	Numeric. The range parameter of the variogram model (in spatial units of `x` raster).
`fun`	A function to apply to the generated values (default is `scale` to standardize the GRF).
`...`	Additional arguments passed to the function specified in `fun`.

Value

A `SpatRaster` object containing the generated Gaussian random field.

Examples

```
library(terra)
r <- rast(nrows = 100, ncols = 100, xmin = 0, xmax = 100, ymin = 0, ymax = 100)
grf_field <- grf(r, range = 30)
plot(grf_field)
```

`initialise_com` *Initialise Community Simulation Data*

Description

Prepares community-level input data for a spatial simulation. This function builds on `rangr::initialise()` by organising inputs for multiple species and their interactions.

Usage

```
initialise_com(
  n1_map = NULL,
  K_map,
  r,
  a,
  dlist = NULL,
  invasion = NULL,
  use_names_K_map = TRUE,
  ...
)
```

Arguments

`n1_map`	A `SpatRaster` with one layer per species representing the initial abundance. If `NULL` (default), random initial values will be generated from a Poisson distribution using `K_map`.
`K_map`	A `SpatRaster` with one layer per species representing carrying capacities.
`r`	A numeric vector of intrinsic growth rates. It can be a single-element vector (if all species have the same intrinsic growth rate) or a vector of length equal to the number of species in the community.
`a`	A square numeric matrix representing interaction coefficients between species. Each element `a_ij` is the per-capita interaction strength of species `j` on species `i`. It expresses the change in carrying capacity of species `i` by a single individual of species `j`. The diagonal must be `NA` and the matrix must be a square matrix of order equal to the number of species. It does not have to be symmetric.
`dlist`	Optional. A list; target cells at a specified distance calculated for every cell within the study area.
`invasion`	Optional. A named list of specifying invasion configuration (can be prepared using `initialise_inv`). Must contain: **invaders** Integer vector of invading species indices. **propagule_size** Number of individuals introduced per invasion event. **invasion_times** Matrix of invasion times, with one row per invader.
`use_names_K_map`	Logical. If `TRUE`, the layer names of `K_map` are used as species names. If `FALSE`, species are numbered sequentially (`1:number_of_species`). Defaults to `TRUE`.

`...`	Additional named arguments passed to `initialise()`. Each must be either length 1 or equal to the number of species. **kernel_args** Optional. A list of lists, each containing named arguments for the corresponding species' kernel function. Must be the same length as number of species.

Value

A list of class `sim_com_data` containing:

spec_data A list of `sim_data` objects (one per species) returned by `initialise()`.

nspec The number of species.

a The interaction matrix.

r Intrinsic growth rate(s).

n1_map Initial abundance maps (wrapped `SpatRaster`).

K_map Carrying capacity maps (wrapped `SpatRaster`).

max_dist The maximum dispersal distance across all species.

dlist A list; target cells at a specified distance calculated for every cell within the study area.

invasion Invasion configuration (if any).

call The matched call.

Examples

```
library(terra)

# Read data from the mrangr package

## Input maps
K_map <- rast(system.file("input_maps/K_map_eg.tif", package = "mrangr"))
n1_map <- rast(system.file("input_maps/n1_map_eg.tif", package = "mrangr"))

## Interaction coefficients matrix
a <- a_eg

# Initialise simulation parameters
community_01 <-
  initialise_com(
  K_map = K_map,
  n1_map = n1_map,
  r = 1.1,
  a = a,
  rate = 0.002)

# With invaders
invasion <- initialise_inv(
  invaders = c(1, 3),
  invasion_times = c(2, 5))
```

```
community_02 <- initialise_com(
  K_map = K_map,
  r = 1.1,
  a = a,
  rate = 0.002,
  invasion = invasion)


# Custom kernel function
abs_rnorm <- function(n, mean, sd) {
  abs(rnorm(n, mean = mean, sd = sd))
}

community_03 <- initialise_com(
  K_map = K_map,
  n1_map = n1_map,
  r = c(1.1, 1.05, 1.2, 1),
  a = a,
  kernel_fun = c("rexp", "rexp", "abs_rnorm", "abs_rnorm"),
  kernel_args = list(
    list(rate = 0.002),
    list(rate = 0.001),
    list(mean = 0, sd = 1000),
    list(mean = 0, sd = 2000))
)
```

`initialise_inv` *Initialise Invasion Parameters*

Description

Prepares a list of invasion configuration details, including the identifiers of the invading species, the times of invasion and the number of individuals introduced at each event. Result of this helper function is designed to be passed to `initialise_com()` as `invasion` argument.

Usage

```
initialise_inv(invaders, invasion_times, propagule_size = 1)
```

Arguments

`invaders`	An integer vector of species indices indicating which species are invaders.These indices should match the species layers in the input maps (`n1_map` and `K_map`).
`invasion_times`	A matrix or vector specifying when each invader enters the system. If a vector is provided, it is assumed to apply to all invaders. If a matrix, it must have one row per invader and columns corresponding to invasion events.
`propagule_size`	A numeric scalar specifying the number of individuals introduced at each invasion time. Defaults to 1.

Value

A named list with the following components:

invaders Integer vector of invading species indices.

propagule_size Number of individuals introduced per invasion event.

invasion_times Matrix of invasion times, with one row per invader.

Examples

```
# Define invaders and invasion times
initialise_inv(
  invaders = c(1, 3),
  invasion_times = matrix(c(5, 10, 5, 20), nrow = 2, byrow = TRUE),
  propagule_size = 10
)

# Uniform invasion times across all invaders
initialise_inv(
  invaders = c(2, 4),
  invasion_times = c(5, 10, 15)
)
```

`K_map_eg.tif` *Example Of Carrying Capacity Map*

Description

`SpatRaster` object with 4 layer that can be passed to `initialise_com` a as simulation (`sim_com`) starting point.

This map is compatible with `n1_map_eg.tif`.

Format

`SpatRaster` object with 4 layers, each with 15 rows and 15 columns. Contains numeric values representing carrying capacity and NA's indicating unsuitable areas.

Source

Data generated in-house to serve as an example (using spatial autocorrelation).

Examples

```
terra::rast(system.file("input_maps/K_map_eg.tif", package = "mrangr"))
```

`K_sim` *Carrying Capacity Map Simulator*

Description

Generates multiple carrying capacity maps based on spatially autocorrelated Gaussian Random Fields (GRFs), with optional correlation between layers.

Usage

```
K_sim(n, id, range, cor_mat = NULL, qfun = qnorm, ...)
```

Arguments

`n`	Integer. Number of maps to generate.
`id`	A `SpatRaster` object used as a geographic template.
`range`	Numeric. Spatial autocorrelation parameter passed to the `grf` function.
`cor_mat`	Optional correlation matrix. If `NULL`, maps are generated independently.
`qfun`	Quantile function to apply to the generated GRFs (default: `qnorm`).
`...`	Additional arguments passed to the quantile function `qfun`.

Value

A `SpatRaster` object with `n` layers, each representing a carrying capacity map.

Examples

```
library(terra)
library(FieldSimR)

# Community parameters
nspec <- 3
nrows <- ncols <- 10
xmin <- 250000; xmax <- xmin + nrows * 1000
ymin <- 600000; ymax <- ymin + ncols * 1000
id <- rast(nrows = nrows, ncols = ncols, xmin = xmin, xmax = xmax, ymin = ymin, ymax = ymax)
crs(id) <- "epsg:2180"
plot(id)

# Correlation matrix of carrying capacities
cor_mat <- matrix(c(1, 0.29, 0.32, 0.29, 1, 0.32, 0.32, 0.32, 1), nrow = nspec, ncol = nspec)
cor_mat

# Generate and define the distributions and parameters of correlated carrying capacity maps
K_map <- K_sim(nspec, id, range = 20000, cor_mat = cor_mat, qfun = qlnorm, meanlog = 2, sdlog = 0.5)
K_map
hist(K_map)
plot(K_map)
```

`n1_map_eg.tif` *Example Of Abundance Map At First Time Step Of The Simulation*

Description

`SpatRaster` object with 4 layer that can be passed to `initialise_com` a as simulation (`sim_com`) starting point.

This map is compatible with `K_map_eg.tif`.

Format

`SpatRaster` object with 4 layers, each with 15 rows and 15 columns. Contains integer values representing abundance and NA's indicating unsuitable areas.

Source

Data generated in-house to serve as an example.

Examples

```
terra::rast(system.file("input_maps/n1_map_eg.tif", package = "mrangr"))
```

`plot.sim_com_results` *Plot* `sim_com_results` *Object*

Description

Draws simulated abundance maps for any species at any time

Usage

```
## S3 method for class 'sim_com_results'
plot(
  x,
  species = seq_len(dim(x$N_map)[4]),
  time_points = x$sim_time,
  type = "continuous",
  main,
  range,
  ...
)
```

Arguments

`x`	An object of class `sim_com_results`, returned by `sim_com()`.
`species`	Integer vector. Species ID(s) to plot.
`time_points`	Integer vector. Time step(s) to plot (excluding burn-in).
`type`	Character vector of length 1. Type of map: "continuous" (default), "classes" or "interval" (case-sensitive)
`main`	Character vector. Plot titles (one for each layer)
`range`	Numeric vector of length 2. Range of values to be used for the legend (if `type = "continuous"`), which by default is calculated from the N_map slot of `sim_com_results` object
`...`	Further arguments passed to `terra::plot`

Value

#' * If `length(time_points) == 1`, returns a `SpatRaster` with species as layers.

- If only one species is selected with multiple time points, returns a single `SpatRaster`.

Examples

```
# Read simulation data from the mrangr package
simulated_com <- get_simulated_com()

# Plot
plot(simulated_com)
```

`plot_series` *Community Time-Series Plot*

Description

This function plots a community time-series for a given location and time.

Usage

```
plot_series(
  obj,
  x = seq(dim(obj$N_map)[1]),
  y = seq(dim(obj$N_map)[2]),
  time = seq(obj$sim_time),
  species = seq(dim(obj$N_map)[4]),
  trans = NULL,
  ...
)
```

Arguments

`obj`	An object of class `sim_com_results`.
`x`	Indices for the x-dimension - first dimension of the `obj$N_map` (default: full range).
`y`	Indices for the y-dimension - second dimension of the `obj$N_map` (default: full range).
`time`	Indices for the time-dimension - third dimension of the `obj$N_map` (default: full range).
`species`	Indices for the species - fourth dimension of the `obj$N_map` (default: full range).
`trans`	An optional function to apply to the calculated mean series before plotting (e.g., `log`, `log1p`). Defaults to `NULL` (no transformation).
`...`	Additional graphical parameters passed to `plot`.

Value

Invisibly returns a matrix of the mean (and possibly transformed) abundance values for each species.

Examples

```
# Read simulation data from the mrangr package
simulated_com <- get_simulated_com()

# Plot
plot_series(simulated_com)
plot_series(simulated_com, x = 5:12, y = 1:5)
plot_series(simulated_com, time = 1:5)
plot_series(simulated_com, trans = log1p)
```

`print.sim_com_data` *Print* `sim_com_data` *Object*

Description

Print `sim_com_data` Object

Usage

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

Arguments

`x`	`sim_com_data` object; returned by the `initialise_com` function
`...`	further arguments passed to or from other methods; currently none specified

Value

`sim_com_data` object is invisibly returned (the `x` param)

Examples

```
# Read community data from the mrangr package
community <- get_community()

# Print
print(community)
```

`print.sim_com_results` *Print* `sim_com_results` *Object*

Description

Print `sim_com_results` Object

Usage

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

Arguments

`x`	`sim_com_results` object; returned by the `sim_com` function
`...`	further arguments passed to or from other methods; none specified

Value

`sim_com_results` object is invisibly returned (the `x` param)

Examples

```
# Read simulation data from the mrangr package
simulated_com <- get_simulated_com()

# Print
print(simulated_com)
```

`print.summary.sim_com_data`

Print `summary.sim_com_data` *Object*

Description

Print `summary.sim_com_data` Object

Usage

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

Arguments

`x`	An object of class `summary.sim_com_data`
`...`	Additional arguments (not used)

Value

Invisibly returns `x`

Examples

```
# Read community data from the mrangr package
community <- get_community()

# Print summary
sim_com_data_summary <- summary(community)
print(sim_com_data_summary)
```

`print.summary.sim_com_results`

Print `summary.sim_results` *Object*

Description

Print `summary.sim_results` Object

Usage

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

Arguments

`x`	`summary.sim_com_results` object; returned by `summary.sim_com_results` function
`...`	further arguments passed to or from other methods; currently none specified

Value

None

Examples

```
# Read simulation data from the mrangr package
simulated_com <- get_simulated_com()

# Print summary
sim_com_summary <- summary(simulated_com)
print(sim_com_summary)
```

`set_zero` *Set Non-Missing Values to Zero*

Description

This function takes an object and sets all non-missing values to zero, while leaving missing values unchanged.

Usage

```
set_zero(x)
```

Arguments

`x`	A vector or other object for which `is.na()` and subsetting with `[]` are defined (e.g., vector, data frame, SpatRaster).

Value

An object of the same type as `x` with all originally non-missing elements replaced by zero.

Examples

```
# Example with a numeric vector
vec <- c(1, 2, NA, 4, NA, 5)
set_zero(vec)
```

`simulated_com_eg` *Example Simulated Community Output*

Description

A `sim_com_results` object containing results of a 20-step simulation of a 4-species community.

The simulation was generated using the `community_eg` object.

This object can be accessed via the `get_simulated_com` function.

Format

An object of class `sim_com_results` from the `mrangr` package generated using `sim_com`.

Source

Data generated in-house to serve as an example

See Also

`get_simulated_com`, `plot.sim_com_results`, `sim_com`

`sim_com` *Simulate Community Dynamics Over Time*

Description

This function simulates species interactions and population dynamics over a given period. It accounts for species invasions and updates population abundances at each time step.

Usage

```
sim_com(obj, time, burn = 0, progress_bar = TRUE)
```

Arguments

`obj`	An object of class `sim_com_data`, as returned by `initialise_com()`.
`time`	Integer. Total number of simulation steps. Must be >= 2.
`burn`	Integer. Number of initial burn-in steps to exclude from the output. Must be >= 0 and < `time`.
`progress_bar`	Logical. Whether to display a progress bar during the simulation.

Value

An object of class `sim_com_results`, a list containing:

extinction Named logical vector indicating species that went extinct.

sim_time Integer. Duration of the output simulation (excluding burn-in).

id A `SpatRaster` object used as a geographic template.

N_map 4D array [rows, cols, time, species] of population abundances.

Examples

```
# Read community data from the mrangr package
community <- get_community()

# Simulation
simulated_com_01 <- sim_com(obj = community, time = 10)

# Simulation with burned time steps
simulated_com_02 <- sim_com(obj = community, time = 10, burn = 3)
```

`summary.sim_com_data` *Summary Of* `sim_com_data` *Object*

Description

Summary Of `sim_com_data` Object

Usage

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

Arguments

`object`	`sim_com_data` object; returned by `initialise_com` function
`...`	further arguments passed to or from other methods; currently none used

Value

A list of class `summary.sim_com_data`

Examples

```
# Read community data from the mrangr package
community <- get_community()

# Summary
summary(community)
```

`summary.sim_com_results` *Summary Of* `sim_com_results` *Object*

Description

Summary Of `sim_com_results` Object

Usage

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

Arguments

`object`	`sim_com_results` object; returned by `sim_com` function
`...`	further arguments passed to or from other methods; none specified

Value

`summary.sim_com_results` object

Examples

```
# Read simulation data from the mrangr package
simulated_com <- get_simulated_com()

# Summary
summary(simulated_com)
```

`to_rast.sim_com_results` *Convert* `sim_com_results` *to SpatRaster(s)*

Description

Converts simulated population abundance data from a `sim_com_results` object (produced by `sim_com()`) into `SpatRaster` objects.

Usage

```
## S3 method for class 'sim_com_results'
to_rast(
  obj,
  species = seq_len(dim(obj$N_map)[4]),
  time_points = obj$sim_time,
  ...
)
```

Arguments

`obj`	An object of class `sim_com_results`, returned by `sim_com()`.
`species`	Integer vector. Species ID(s) to extract.
`time_points`	Integer vector. Time step(s) to extract (excluding burn-in).
`...`	Currently unused.

Value

- If `length(time_points) == 1`, returns a `SpatRaster` with species as layers.
- If `length(time_points) > 1`, returns a named list of `SpatRaster` objects, one per species.
- If only one species is selected with multiple time points, returns a single `SpatRaster`.

Examples

```
# Read simulation data from the mrangr package
simulated_com <- get_simulated_com()

# Extract one timestep, all species
r1 <- to_rast(simulated_com, time_points = 10)

# Extract multiple timesteps, one species
r2 <- to_rast(simulated_com, species = 2, time_points = c(1, 5, 10))

# Extract multiple timesteps, multiple species
r3 <- to_rast(simulated_com, species = c(1, 2), time_points = c(1, 5, 10))
```

`update.sim_com_data` *Update* `sim_com_data` *Object*

Description

Updates the parameters used to create a `sim_com_data` object, returned by `initialise_com()`.

Usage

```
## S3 method for class 'sim_com_data'
update(object, ..., evaluate = TRUE)
```

Arguments

`object`	A `sim_com_data` object, as returned by `initialise_com()`.
`...`	Named arguments to update. These should be valid arguments to `initialise_com()`. If `kernel_fun` is updated, any associated `kernel_args` (if present in previous call) will also be replaced.
`evaluate`	Logical (default `TRUE`). If `TRUE`, the function returns the re-evaluated updated object; if `FALSE`, it returns the updated function call without evaluating it.

Details

- If dispersal-related arguments such as `max_dist`, `kernel_fun`, or `kernel_args` are changed, the existing `dlist` is removed and recalculated unless a new `dlist` is explicitly provided.
- If `n1_map` or `K_map` is updated, the `dlist` will also be removed to ensure consistency.

Value

If `evaluate = TRUE`, a new `sim_com_data` object with updated parameters. If `evaluate = FALSE`, a `call` object representing the updated function call.

See Also

`initialise_com()`

Examples

```
library(terra)

# Read data from the mrangr package

## Input maps
K_map <- rast(system.file("input_maps/K_map_eg.tif", package = "mrangr"))
n1_map <- rast(system.file("input_maps/n1_map_eg.tif", package = "mrangr"))

## Competition coefficients matrix
a <- a_eg

# Initialise simulation parameters
community_01 <-
  initialise_com(
  K_map = K_map,
  r = 1.1,
  a = a,
  rate = 0.002)

# Update simulation parameters
```

```
# Custom kernel function
abs_rnorm <- function(n, mean, sd) {
  abs(rnorm(n, mean = mean, sd = sd))
}

community_02 <- update(community_01,
  kernel_fun = c("rexp", "rexp", "abs_rnorm", "abs_rnorm"),
  kernel_args = list(
   list(rate = 0.002),
   list(rate = 0.001),
   list(mean = 0, sd = 1000),
   list(mean = 0, sd = 2000)))
```

`virtual_ecologist` *Virtual Ecologist*

Description

Organizes and extracts community data from a simulated community object based on one of three sampling methods: random proportion, constant random sites, or user-provided sites.

Usage

```
virtual_ecologist(
  obj,
  type = c("random_one_layer", "random_all_layers", "from_data"),
  sites = NULL,
  prop = 0.01,
  obs_error = c("rlnorm", "rbinom"),
  obs_error_param = NULL
)
```

Arguments

`obj` An object created by the `sim_com()` function, containing simulation data.

`type` character vector of length 1; describes the sampling type (case-sensitive):

- `"random_one_layer"` - random selection of cells for which abundances are sampled; the same set of selected cells is used across all time steps.
- `"random_all_layers"` - random selection of cells for which abundances are sampled; a new set of cells is selected for each time step.
- `"from_data"` - user-defined selection of cells for which abundances are sampled; the user is required to provide a `data.frame` containing three columns: `"x"`, `"y"` and `"time"`.

`sites` An optional data frame specifying the sites for data extraction. This data frame must contain three columns: `x`, `y` and `time`.

`prop`	A numeric value between 0 and 1. The proportion of cells to randomly sample from the raster.
`obs_error`	character vector of length 1; type of the distribution that defines the observation process: "`rlnorm`" (log-normal distribution) or "`rbinom`" (binomial distribution).
`obs_error_param`	numeric vector of length 1; standard deviation (on a log scale) of the random noise in the observation process when `"rlnorm"` is used, or probability of detection (success) when `"rbinom"` is used.

Value

A data frame with 6 columns:

- `id`: unique cell identifier (factor)
- `x`, `y`: sampled cell coordinates
- `species`: species number or name
- `time`: sampled time step
- `n`: sampled abundance

Examples

```
# Read simulated community data from the mrangr package
simulated_com <- get_simulated_com()

# Option 1: Randomly sample sites (the same for each year)
sampled_data_01 <- virtual_ecologist(simulated_com)
head(sampled_data_01)

# Option 2: Randomly sample sites (different for each year)
sampled_data_02 <- virtual_ecologist(simulated_com, type = "random_all_layers")
head(sampled_data_02)

# Option 3: Sample sites based on user-provided data frame
custom_sites <- data.frame(
  x = c(250500, 252500, 254500),
  y = c(600500, 602500, 604500),
  time = c(1, 10, 20)
)
sampled_data_03 <- virtual_ecologist(simulated_com, sites = custom_sites)
head(sampled_data_03)

# Option 4. Add noise - "rlnorm"
sampled_data_04 <- virtual_ecologist(
  simulated_com,
  sites = custom_sites,
  obs_error = "rlnorm",
  obs_error_param = log(1.2)
)
head(sampled_data_04)
```

```
# Option 5. Add noise - "rbinom"
sampled_data_05 <- virtual_ecologist(
  simulated_com,
  sites = custom_sites,
  obs_error = "rbinom",
  obs_error_param = 0.8
)
head(sampled_data_05)
```

Index