Package: mobsim (via r-universe)

August 22, 2024

Type Package

Title Spatial Simulation and Scale-Dependent Analysis of Biodiversity Changes

Version 0.3.1

Date 2024-03-11

Description Simulation, analysis and sampling of spatial biodiversity data (May, Gerstner, McGlinn, Xiao & Chase 2017) <doi:10.1111/2041-210x.12986>. In the simulation tools user define the numbers of species and individuals, the species abundance distribution and species aggregation. Functions for analysis include species rarefaction and accumulation curves, species-area relationships and the distance decay of similarity.

License GPL (>= 3)

Depends R (>= 4.0.0)

Imports Rcpp, vegan, sads (>= 0.4.1), grDevices, utils, graphics, stats, methods

LinkingTo Rcpp

Suggests rmarkdown, spatstat.geom, spatstat.random, testthat (>= 3.0.0), mockery, knitr

Config/testthat/edition 3

Config/testthat/parallel true

VignetteBuilder knitr

RoxygenNote 7.3.1

Language en-GB

Encoding UTF-8

URL `https://github.com/MoBiodiv/mobsim`

BugReports `https://github.com/MoBiodiv/mobsim/issues`

Repository https://mobiodiv.r-universe.dev

RemoteUrl https://github.com/mobiodiv/mobsim

RemoteRef HEAD

RemoteSha 7c66cdb33312ae2d58ad32f2f4364c74803513b3

Contents

`abund_rect` *Get local species abundance distribution*

Description

Get local abundance distribution in rectangle bounded by x0, y0, x0 + xsize, y0 + ysize

Usage

```
abund_rect(x0, y0, xsize, ysize, comm)
```

Arguments

`x0`	x-coordinate of lower left corner
`y0`	y-coordinate of lower left corner
`xsize`	Size of the subplot in x-direction
`ysize`	Size of the subplot in y-direction
`comm`	`community` object

Value

Integer vector with local species abundances

`community` *Create spatial community object*

Description

Creates a spatial community object with defined extent and with coordinates and species identities of all individuals in the community.

Usage

```
community(x, y, spec_id, xrange = c(0, 1), yrange = c(0, 1))
```

Arguments

`x, y`	Coordinates of individuals (numeric)
`spec_id`	Species names or IDs; can be integers, characters or factors
`xrange`	Extent of the community in x-direction (numeric vector of length 2)
`yrange`	Extent of the community in y-direction (numeric vector of length 2)

Value

Community object which includes three items:

1. census: data.frame with three columns: x, y, and species names for each individual
2. x_min_max: extent of the community in x-direction
3. y_min_max: extent of the community in y-direction

Examples

```
x <- runif(100)
y <- runif(100)
species_names <- rep(paste("species",1:10, sep = ""), each = 10)

com1 <- community(x,y, species_names)
plot(com1)
summary(com1)
```

`community_to_sad` *Get species abundance distribution from community object*

Description

Get species abundance distribution from community object

Usage

```
community_to_sad(comm)
```

Arguments

`comm`	Community object

Value

Object of class `sad`, which contains a named integer vector with species abundances

Examples

```
sim1 <- sim_poisson_community(s_pool = 200, n_sim = 20000, sad_type = "lnorm",
                              sad_coef = list("cv_abund" = 2))
sad1 <- community_to_sad(sim1)
plot(sad1, method = "rank")
plot(sad1, method = "octave")
```

`dist_decay` *Distance decay of similarity*

Description

Estimate pairwise similarities of communities in subplots as function of distance

Usage

```
dist_decay(
  comm,
  prop_area = 0.005,
  n_samples = 20,
  method = "bray",
  binary = FALSE
)
```

Arguments

`comm`	`community` object
`prop_area`	Subplot size as proportion of the total area
`n_samples`	Number of randomly located subplots
`method`	Choice of (dis)similarity index. See `vegdist`
`binary`	Perform presence/absence standardization before analysis? See `vegdist`

Value

Object of class `dist_decay`: a dataframe with distances between subplot pairs and the respective similarity indices.

Examples

```
sim_com1 <- sim_thomas_community(100, 10000, sigma = 0.1, mother_points = 2)
dd1 <- dist_decay(sim_com1, prop_area = 0.005, n_samples = 20)
plot(dd1)
```

`dist_decay_quadrats` *Distance decay of similarity with user-defined quadrats*

Description

Estimate pairwise similarities of communities in quadrats as function of distance. The function allows the user to compute distance decay between the quadrats of his/her choice.

Usage

```
dist_decay_quadrats(samples, method = "bray", binary = FALSE)
```

Arguments

`samples`	A list given by `sample_quadrats`
`method`	Choice of (dis)similarity index. See `vegdist`
`binary`	Perform presence/absence standardization before analysis? See `vegdist`

Value

Object of class `dist_decay`: a dataframe with distances between subplot pairs and the respective similarity indices.

Examples

```
sim_com1 <- sim_thomas_community(100, 10000, sigma = 0.1, mother_points = 2)
oldpar<- par(mfrow=c(1,2))
samples <- sample_quadrats(sim_com1, avoid_overlap = TRUE, quadrat_area=.005,
                           n_quadrats = 50, plot = TRUE)
dd_quadrats <- dist_decay_quadrats(samples)
plot(dd_quadrats)
par(oldpar)
```

`divar` *Diversity-area relationships*

Description

Estimate diversity indices in subplots of different sizes. This includes the well-known species-area and endemics-area relationships.

Usage

```
divar(
  comm,
  prop_area = seq(0.1, 1, by = 0.1),
  n_samples = 100,
  exclude_zeros = TRUE
)
```

Arguments

`comm`	`community` object
`prop_area`	Subplot sizes as proportion of the total area (numeric)
`n_samples`	Number of randomly located subplots per subplot size (single integer)
`exclude_zeros`	Should subplots without individuals be excluded? (logical)

Value

Dataframe with the proportional area of the subplots and mean and standard deviation of the following diversity indices:

1. Number of species
2. Number of endemics
3. Shannon index
4. Effective number of species (ENS) based on Shannon index
5. Simpson index
6. Effective number of species (ENS) based on Simpson index

See the documentation of `div_rect` for detailed information on the definition of the diversity indices.

See Also

`div_rand_rect`, `div_rect`

Examples

```
sim1 <- sim_thomas_community(100, 1000)
divar1 <- divar(sim1, prop_area = seq(0.01, 1.0, length = 20))
plot(divar1)
```

`div_rand_rect` *Distribution of local diversity indices*

Description

Get mean and standard deviation of diversity indices in several equally sized subplots of a community

Usage

```
div_rand_rect(prop_area = 0.25, comm, n_rect = 100, exclude_zeros = FALSE)
```

Arguments

`prop_area`	Size of subplots as proportion of the total area
`comm`	`community` object
`n_rect`	Number of randomly located subplots
`exclude_zeros`	Should subplots without individuals be excluded? (logical)

Value

Vector with mean and standard deviation of the following diversity indices:

1. Number of species
2. Number of endemics
3. Shannon index
4. Effective number of species (ENS) based on Shannon index
5. Simpson index
6. Effective number of species (ENS) based on Simpson index

See the documentation of `div_rect` for detailed information on the definition of the diversity indices.

Examples

```
sim1 <- sim_poisson_community(100,1000)
div_rand_rect(prop_area = 0.1, comm = sim1)
```

`div_rect` *Get local diversity indices*

Description

Get diversity indices including species richness, no. of endemics, Shannon and Simpson diversity for one rectangle subplot in the community.

Usage

```
div_rect(x0, y0, xsize, ysize, comm)
```

Arguments

`x0`	x-coordinate of lower left corner
`y0`	y-coordinate of lower left corner
`xsize`	Size of the subplot in x-direction
`ysize`	Size of the subplot in y-direction
`comm`	`community` object

Details

The effective number of species is defined as the number of equally abundant species that produce the same value of a certain diversity index as an observed community (Jost 2006). According to Chao et al. 2014 and Chiu et al. 20 ENS_shannon can be interpreted as the number of common species and ENS_simpson as the number of dominant species in a community.

Value

Named vector with six diversity indices

1. n_species: Number of species
2. n_endemics: Number of endemics
3. shannon: Shannon index index defined as $H = -\sum p_i * log(p_i)$, where p_i is the relative abundance of species i:
4. ens_shannon: Effective number of species (ENS) based on the Shannon index exp(H)
5. simpson: Simpson index index (= probability of interspecific encounter PIE) defined as $D = 1 - \sum p_i^2$
6. ens_simpson: Effective number of species (ENS) based on the Simpson index $1/D$

References

Jost 2006. Entropy and diversity. Oikos, 113, 363-375.

Chao et al. 2014. Rarefaction and extrapolation with Hill numbers: a framework for sampling and estimation in species diversity studies. Ecological Monographs, 84, 45-67.

Hsieh et al. 2016. iNEXT: an R package for rarefaction and extrapolation of species diversity (Hill numbers). Methods Ecol Evol, 7, 1451-1456.

Examples

```
sim1 <- sim_poisson_community(100,1000)
div_rect(0, 0, 0.3, 0.3, sim1)
```

`plot.community` *Plot spatial community object*

Description

Plot positions and species identities of all individuals in a community object.

Usage

```
## S3 method for class 'community'
plot(x, ..., col = NULL, pch = NULL)
```

Arguments

`x`	Community object
`...`	Other parameters to `graphics::plot`
`col`	Colour vector to mark species identities
`pch`	Plotting character to mark species identities. pch 16 is advised for large datasets

Value

This function is called for its side effects and has no return value.

Examples

```
sim1 <- sim_thomas_community(30, 500)
plot(sim1)
```

`plot.dist_decay` *Plot distance decay of similarity*

Description

Plot distance decay of similarity

Usage

```
## S3 method for class 'dist_decay'
plot(x, ...)
```

Arguments

`x`	Dataframe generated by `dist_decay`
`...`	Additional graphical parameters used in `praphics::plot`.

Details

The function plots the similarity indices between all pairs of subplots as function of distance. To indicate the relationship a `stats::loess` smoother is added to the plot.

Value

This function is called for its side effects and has no return value.

Examples

```
sim_com1 <- sim_thomas_community(100, 10000)
dd1 <- dist_decay(sim_com1)
plot(dd1)
```

`plot.divar` —————————————————————— *Plot diversity-area relationships*

Description

—————————————————————— Plot diversity-area relationships

Usage

```
## S3 method for class 'divar'
plot(x, ...)
```

Arguments

`x`	Dataframe generated by the function `divar`.
`...`	Additional graphical parameters used in `graphics::plot`.

Value

This function is called for its side effects and has no return value.

`plot.sad` *Plot species abundance distributions*

Description

Plot species abundance distributions

Usage

```
## S3 method for class 'sad'
plot(x, ..., method = c("octave", "rank"))
```

Arguments

`x`	Vector with species abundances (integer vector)
`...`	Additional graphical parameters used in `graphics::plot` or `barplot`
`method`	Plotting method, partial match to `"octave"` or `"rank"`

Details

With `method = "octave"` a histogram showing the number species in several abundance classes is generated. The abundance class are a simplified version of the "octaves" suggested by Preston (1948), which are based on log2-binning. The first abundance class includes species with 1 individual, the second with 2, the third with 3-4, the fourth with 5-8, etc.

With `method = "rank"` rank-abundance curve is generated with species abundance rank on the x-axis (descending) and species abundance on the y-axis (Hubbell 2001).

Value

This function is called for its side effects and has no return value.

References

Preston 1948. The Commonness, and rarity, of species. Ecology 29(3):254-283.

Hubbell 2001. The unified neutral theory of biodiversity and biogeography. Princeton University Press.

Examples

```
abund1 <- sim_sad(s_pool = 100, n_sim = 10000, sad_type = "lnorm",
                  sad_coef = list("cv_abund" = 1))
plot(abund1, method = "octave")
plot(abund1, method = "rank")
```

`plot.spec_sample_curve` *Plot species sampling curves*

Description

Plot species sampling curves

Usage

```
## S3 method for class 'spec_sample_curve'
plot(x, ...)
```

Arguments

`x`	Species sampling curve generated by `spec_sample_curve`
`...`	Additional graphical parameters used in `graphics::plot`.

Value

This function is called for its side effects and has no return value.

Examples

```
sim_com1 <- sim_thomas_community(s_pool = 100, n_sim = 1000)
sac1 <- spec_sample_curve(sim_com1, method = c("rare","acc"))
plot(sac1)
```

`rare_curve` *Species rarefaction curve*

Description

Expected species richness as a function of sample size

Usage

```
rare_curve(abund_vec)
```

Arguments

`abund_vec`	Species abundance distribution of the community (integer vector)

Details

This function essentially evaluates `spec_sample` for sample sizes from 1 to `sum(abund_vec)`. It is similar to the function `vegan:rarecurve` in the R package `vegan`.

Value

Numeric Vector with expected species richness in samples of 1, 2, 3 ... n individuals

References

Gotelli & Colwell 2001. Quantifying biodiversity: procedures and pitfalls in the measurement and comparison of species richness. Ecology Letters 4, 379–391.

Examples

```
sad1 <- sim_sad(100, 2000, sad_type = "lnorm", sad_coef = list("meanlog" = 2,
                                                            "sdlog" = 1))
rc1 <- rare_curve(sad1)
plot(rc1, type = "l", xlab = "Sample size", ylab = "Expected species richness")
```

`rThomas_rcpp` *Thomas process individual distribution simulation for one species*

Description

Usually used internally inside `sim_thomas_coords` This function randomly draws points (individuals) around one or several mother points using Rcpp. The function is an efficient re-implementation of the rThomas function from the spatstat package.

Arguments

`n_points`	The total number of points (individuals).
`n_mother_points`	Number of mother points (= cluster centres).
`xmother`	Vector of `n_mother_points` x coordinates for the mother points.
`ymother`	Vector of `n_mother_points` y coordinates for the mother points.
`sigma`	Mean displacement (along each coordinate axes) of a point from its mother point (= cluster centre).
`xmin`	Left limit, `default`=0.
`xmax`	Right limit, `default`=1.
`ymin`	Bottom limit, `default`=0.
`ymax`	Top limit, `default`=1.

Value

A dataframe with x and y coordinates.

Author(s)

Felix May, Alban Sagouis

`sample_quadrats` *Plot-based samples from a spatially-explicit census*

Description

This function allows to sample quadratic subplots from a spatially-explicit community. The output format are a sites x species abundance table and a sites x xy-coordinates table. The sites x species abundance is a classical data format used in community ecology. The table generated can be for instance be further analysed with the package `vegan`.

Usage

```
sample_quadrats(
  comm,
  n_quadrats = 20,
  quadrat_area = 0.01,
  plot = TRUE,
  method = "random",
  avoid_overlap = TRUE,
  x0 = 0,
  y0 = 0,
  delta_x = 0.1,
  delta_y = 0.1,
  seed = NULL
)
```

Arguments

`comm`	Community object from which the samples are generated
`n_quadrats`	(integer) Number of sampling quadrats
`quadrat_area`	(numeric) Area of the sampling quadrats
`plot`	(logical) Should the sampling design be plotted? default to TRUE.
`method`	(character) Available methods are `"random"`, `"transect"`, `"grid"`
`avoid_overlap`	(logical) For the random sampling try to generate a design without overlap of quadrats . Default is TRUE.
`x0, y0`	(numeric value) Lower left corner of the first quadrat in transect and grid sampling

`delta_x`	(numeric value) Distance between consecutive quadrats in transect and grid sampling in x-direction (the distance between the left sides is measured)
`delta_y`	(numeric value) Distance between consecutive quadrats in transect and grid sampling in y-direction (the distance between the lower sides is measured)
`seed`	(integer) Any integer passed to `set.seed` for reproducibility.

Value

A list with two items, `spec_dat` and `xy_dat`. `spec_dat` is a data.frame with sampling quadrats in rows and species abundances in columns, and `xy_dat` is a data.frame with sampling quadrats in rows and the xy-coordinates of the quadrats (lower left corner) in columns.

Examples

```
library(vegan)
sim_com1 <- sim_poisson_community(100, 10000)
comm_mat1 <- sample_quadrats(sim_com1, n_quadrats = 100,
quadrat_area = 0.002, method = "grid")
specnumber(comm_mat1$spec_dat)
diversity(comm_mat1$spec_dat, index = "shannon")
```

`sampling_grids` *Creates square quadrats aligned on a regular grid*

Description

Creates square quadrats aligned on a regular grid

Usage

```
sampling_grids(
  n_quadrats,
  xmin,
  xmax,
  ymin,
  ymax,
  x0,
  y0,
  delta_x,
  delta_y,
  quadrat_size
)
```

Arguments

`n_quadrats`	(integer) Number of sampling quadrats
`xmin`	(numeric) minimum possible value on the x axis a quadrat can cover.
`xmax`	(numeric) maximum possible value on the x axis a quadrat can cover.
`ymin`	(numeric) minimum possible value on the y axis a quadrat can cover.
`ymax`	(numeric) maximum possible value on the y axis a quadrat can cover.
`x0`, `y0`	(numeric value) Lower left corner of the first quadrat in transect and grid sampling
`delta_x`	(numeric value) Distance between consecutive quadrats in transect and grid sampling in x-direction (the distance between the left sides is measured)
`delta_y`	(numeric value) Distance between consecutive quadrats in transect and grid sampling in y-direction (the distance between the lower sides is measured)
`quadrat_size`	(numeric) width of the quadrats.

Value

a data.frame with 2 columns x and y giving the coordinates of the lower left corner of the square quadrats.

`sampling_one_quadrat` *Creates one square quadrat randomly located in the landscape*

Description

Creates one square quadrat randomly located in the landscape

Usage

```
sampling_one_quadrat(xmin, xmax, ymin, ymax, seed = NULL)
```

Arguments

`xmin`	(numeric) minimum possible value on the x axis a quadrat can cover.
`xmax`	(numeric) maximum possible value on the x axis a quadrat can cover.
`ymin`	(numeric) minimum possible value on the y axis a quadrat can cover.
`ymax`	(numeric) maximum possible value on the y axis a quadrat can cover.
`seed`	(integer) Any integer passed to `set.seed` for reproducibility.

Value

a data.frame with 2 columns x and y giving the coordinates of the lower left corner of the square quadrat.

`sampling_random_bruteforce` *Creates coordinates (lower left corner of a quadrat) randomly distributed but without overlapping each other*

Description

This function works without having the `spatstat.random` package install.

Usage

```
sampling_random_bruteforce(
  n_quadrats,
  min_dist,
  xmin,
  xmax,
  ymin,
  ymax,
  seed = NULL
)
```

Arguments

`n_quadrats`	Number of sampling quadrats
`min_dist`	(numeric) minimal distance between two points to avoid overlap. Equal to the length of a quadrat diagonal
`xmin`	(numeric) minimum possible value on the x axis a quadrat can cover.
`xmax`	(numeric) maximum possible value on the x axis a quadrat can cover.
`ymin`	(numeric) minimum possible value on the y axis a quadrat can cover.
`ymax`	(numeric) maximum possible value on the y axis a quadrat can cover.
`seed`	(integer) Any integer passed to `set.seed` for reproducibility.

Value

a data.frame with 2 columns x and y giving the coordinates of the lower left corner of the square quadrats.

`sampling_random_overlap` *Creates coordinates (lower left corner of a quadrat) randomly distributed that may overlap each other*

Description

Creates coordinates (lower left corner of a quadrat) randomly distributed that may overlap each other

Usage

```
sampling_random_overlap(
  n_quadrats,
  min_dist,
  xmin,
  xmax,
  ymin,
  ymax,
  seed = NULL
)
```

Arguments

`n_quadrats`	Number of sampling quadrats
`min_dist`	(numeric) minimal distance between two points to avoid overlap. Equal to the length of a quadrat diagonal
`xmin`	(numeric) minimum possible value on the x axis a quadrat can cover.
`xmax`	(numeric) maximum possible value on the x axis a quadrat can cover.
`ymin`	(numeric) minimum possible value on the y axis a quadrat can cover.
`ymax`	(numeric) maximum possible value on the y axis a quadrat can cover.
`seed`	(integer) Any integer passed to `set.seed` for reproducibility.

Value

a data.frame with 2 columns x and y giving the coordinates of the lower left corner of the square quadrats.

`sampling_random_spatstat`

Creates coordinates (lower left corner of a quadrat) randomly distributed but without overlapping each other

---

Description

Efficient algorithm from package `spatstat.random` is used. Produces similar results as `sampling_random_bruteforce`.

Usage

```
sampling_random_spatstat(
  n_quadrats,
  min_dist,
  xmin,
  xmax,
  ymin,
  ymax,
  seed = NULL
)
```

Arguments

`n_quadrats`	Number of sampling quadrats
`min_dist`	(numeric) minimal distance between two points to avoid overlap. Equal to the length of a quadrat diagonal
`xmin`	(numeric) minimum possible value on the x axis a quadrat can cover.
`xmax`	(numeric) maximum possible value on the x axis a quadrat can cover.
`ymin`	(numeric) minimum possible value on the y axis a quadrat can cover.
`ymax`	(numeric) maximum possible value on the y axis a quadrat can cover.
`seed`	(integer) Any integer passed to `set.seed` for reproducibility.

Value

a data.frame with 2 columns x and y giving the coordinates of the lower left corner of the square quadrats.

`sampling_transects` *Creates square quadrats aligned along a transect*

Description

Creates square quadrats aligned along a transect

Usage

```
sampling_transects(
  n_quadrats,
  xmin,
  xmax,
  ymin,
  ymax,
  x0,
  y0,
  delta_x,
  delta_y,
  quadrat_size
)
```

Arguments

`n_quadrats`	(integer) Number of sampling quadrats
`xmin`	(numeric) minimum possible value on the x axis a quadrat can cover.
`xmax`	(numeric) maximum possible value on the x axis a quadrat can cover.
`ymin`	(numeric) minimum possible value on the y axis a quadrat can cover.
`ymax`	(numeric) maximum possible value on the y axis a quadrat can cover.
`x0`, `y0`	(numeric value) Lower left corner of the first quadrat in transect and grid sampling
`delta_x`	(numeric value) Distance between consecutive quadrats in transect and grid sampling in x-direction (the distance between the left sides is measured)
`delta_y`	(numeric value) Distance between consecutive quadrats in transect and grid sampling in y-direction (the distance between the lower sides is measured)
`quadrat_size`	(numeric) width of the quadrats.

Value

a data.frame with 2 columns x and y giving the coordinates of the lower left corner of the square quadrats.

`sim_poisson_community` *Simulate community with random spatial positions.*

Description

This function simulates a community with a certain abundance distribution and and random spatial coordinates. This function consecutively calls `sim_sad` and `sim_poisson_coords`

Usage

```
sim_poisson_community(
  s_pool,
  n_sim,
  sad_type = "lnorm",
  sad_coef = list(cv_abund = 1),
  fix_s_sim = FALSE,
  xrange = c(0, 1),
  yrange = c(0, 1),
  seed = NULL
)
```

Arguments

`s_pool`	Number of species in the pool (integer)
`n_sim`	Number of individuals in the simulated community (integer)
`sad_type`	Root name of the species abundance distribution model of the species pool (character) - e.g., "lnorm" for the lognormal distribution (`rlnorm`); "geom" for the geometric distribution (`rgeom`), or "ls" for Fisher's log-series distribution (`rls`). See the table in **Details** below, or `rsad` for all SAD model options.
`sad_coef`	List with named arguments to be passed to the distribution function defined by the argument `sad_type`. An overview of parameter names is given in the table below. In `mobsim` the log-normal and the Poisson log-normal distributions can alternatively be parameterized by the coefficient of variation (cv) of the relative abundances in the species pool. Accordingly, `cv_abund` is the standard deviation of abundances divided by the mean abundance (no. of individuals / no. of species). `cv_abund` is thus negatively correlated with the evenness of the species abundance distribution. Please note that the parameters *mu* and *sigma* are not equal to the mean and standard deviation of the log-normal distribution.
`fix_s_sim`	Should the simulation constrain the number of species in the simulated local community? (logical)
`xrange`	Extent of the community in x-direction (numeric vector of length 2)
`yrange`	Extent of the community in y-direction (numeric vector of length 2)
`seed`	Integer. Any integer passed to `set.seed` for reproducibility.

Value

A community object as defined by `community`.

Author(s)

Felix May

Examples

```
com1 <- sim_poisson_community(s_pool = 20, n_sim = 500, sad_type = "lnorm",
sad_coef = list("meanlog" = 2, "sdlog" = 1))
plot(com1)
```

`sim_poisson_coords` *Simulate random spatial coordinates*

Description

Add random spatial positions to a species abundance distribution.

Usage

```
sim_poisson_coords(abund_vec, xrange = c(0, 1), yrange = c(0, 1), seed = NULL)
```

Arguments

`abund_vec`	Species abundance vector (integer)
`xrange`	Extent of the community in x-direction (numeric vector of length 2)
`yrange`	Extent of the community in y-direction (numeric vector of length 2)
`seed`	Integer. Any integer passed to `set.seed` for reproducibility.

Value

A community object as defined by `community`.

Author(s)

Felix May

Examples

```
abund <- sim_sad(s_pool = 100, n_sim = 1000)
sim_com1 <- sim_poisson_coords(abund)
plot(sim_com1)
summary(sim_com1)
```

`sim_sad` *Simulate species abundance distributions*

Description

Simulate species abundance distribution (SAD) of a local community with user-defined number of species and relative abundance distribution in the pool, and user-defined number of individuals in the simulated local community.

Usage

```
sim_sad(
  s_pool = NULL,
  n_sim = NULL,
  sad_type = c("lnorm", "bs", "gamma", "geom", "ls", "mzsm", "nbinom", "pareto",
    "poilog", "power", "powbend", "weibull"),
  sad_coef = list(cv_abund = 1),
  fix_s_sim = FALSE,
  drop_zeros = TRUE,
  seed = NULL
)
```

Arguments

`s_pool`	Number of species in the pool (integer)
`n_sim`	Number of individuals in the simulated community (integer)
`sad_type`	Root name of the species abundance distribution model of the species pool (character) - e.g., "lnorm" for the lognormal distribution (`rlnorm`); "geom" for the geometric distribution (`rgeom`), or "ls" for Fisher's log-series distribution (`rls`). See the table in **Details** below, or `rsad` for all SAD model options.
`sad_coef`	List with named arguments to be passed to the distribution function defined by the argument `sad_type`. An overview of parameter names is given in the table below. In `mobsim` the log-normal and the Poisson log-normal distributions can alternatively be parameterized by the coefficient of variation (cv) of the relative abundances in the species pool. Accordingly, `cv_abund` is the standard deviation of abundances divided by the mean abundance (no. of individuals / no. of species). `cv_abund` is thus negatively correlated with the evenness of the species abundance distribution. Please note that the parameters *mu* and *sigma* are not equal to the mean and standard deviation of the log-normal distribution.
`fix_s_sim`	Should the simulation constrain the number of species in the simulated local community? (logical)
`drop_zeros`	Should the function remove species with abundance zero from the output? (logical)
`seed`	Integer. Any integer passed to `set.seed` for reproducibility.

Details

The function `sim_sad` was built using code of the function `sads::rsad` from the R package `sads`. However, in contrast to `sads::rsad`, the function `sim_sad` allows to define the number of individuals in the simulated local community. This is implemented by converting the abundance distribution simulated based on `sads::rsad` into a relative abundance distribution. This relative abundance distribution is considered as the species pool for the local community. In a second step the required no. of individuals (`n_sim`) is sampled (with replacement) from this relative abundance distribution.

Please note that this might effect the interpretation of the parameters of the underlying statistical distribution, e.g. the mean abundance will always be `n_sim/s_pool` irrespective of the settings of `sad_coef`.

When `fix_s_sim = FALSE` the species number in the local community might deviate from `s_pool` due to stochastic sampling. When `fix_s_sim = TRUE` the local number of species will equal `s_pool`, but this constraint can result in systematic biases from the theoretical distribution parameters. Generally, with `fix_s_sim = TRUE` additional very rare species will be added to the community, while the abundance of the most common ones is reduced to keep the defined number of individuals.

Here is an overview of all available models (`sad_type`) and their respective coefficients (`sad_coef`). Further information is provided by the documentation of the specific functions that can be accesses by the links. Please note that the coefficient `cv_abund` for the log-normal and Poisson log-normal model are only available within `mobsim`.

SAD function	**Distribution name**	**coef #1**	**coef #2**	**coef #3**
`sads::rbs`	Mac-Arthur's brokenstick	N	S	
`stats:rgamma`	Gamma distribution	shape	rate	scale
`rgeom`	Geometric distribution	prob		
`rlnorm`	Log-normal distributions	meanlog	sdlog	cv_abund
`rls`	Fisher's log-series distribution	N	alpha	
`sads::rmzsm`	Metacommunity zero-sum multinomial	J	theta	
`stats::rnbinom`	Negative binomial distribution	size	prob	mu
`sads::rpareto`	Pareto distribution	shape	scale	
`sads::rpoilog`	Poisson-lognormal distribution	mu	sigma	cv_abund
`sads::rpower`	Power discrete distributions	s		
`sads::rpowbend`	Puyeo's Power-bend discrete distribution	s	omega	
`stats::rweibull`	Weibull distribution	shape	scale	

Value

Object of class `sad`, which contains a named integer vector with species abundances

Author(s)

Felix May

Examples

```
#Simulate log-normal species abundance distribution
sad_lnorm1 <- sim_sad(s_pool = 100, n_sim = 10000, sad_type = "lnorm",
                      sad_coef = list("meanlog" = 5, "sdlog" = 0.5))
```

```
plot(sad_lnorm1, method = "octave")
plot(sad_lnorm1, method = "rank")

# Alternative parameterization of the log-normal distribution
sad_lnorm2 <- sim_sad(s_pool = 100, n_sim = 10000, sad_type = "lnorm",
                      sad_coef = list("cv_abund" = 0.5))
plot(sad_lnorm2, method = "octave")

# Fix species richness in the simulation by adding rare species
sad_lnorm3a <- sim_sad(s_pool = 500, n_sim = 10000, sad_type = "lnorm",
                       sad_coef = list("cv_abund" = 5), fix_s_sim = TRUE)
sad_lnorm3b <- sim_sad(s_pool = 500, n_sim = 10000, sad_type = "lnorm",
                       sad_coef = list("cv_abund" = 5))

plot(sad_lnorm3a, method = "rank")
points(1:length(sad_lnorm3b), sad_lnorm3b, type = "b", col = 2)
legend("topright", c("fix_s_sim = TRUE","fix_s_sim = FALSE"),
       col = 1:2, pch = 1)

# Different important SAD models

# Fisher's log-series
sad_logseries <- sim_sad(s_pool = NULL, n_sim = NULL, sad_type = "ls",
                         sad_coef = list("N" = 1e5, "alpha" = 20))

# Poisson log-normal
sad_poilog <- sim_sad(s_pool = 100, n_sim = 10000, sad_type = "poilog",
                      sad_coef = list("mu" = 5, "sig" = 0.5))

# Mac-Arthur's broken stick
sad_broken_stick <- sim_sad(s_pool = NULL, n_sim = NULL, sad_type = "bs",
                            sad_coef = list("N" = 1e5, "S" = 100))

# Plot all SADs together as rank-abundance curves
plot(sad_logseries, method = "rank")
lines(1:length(sad_lnorm2), sad_lnorm2, type = "b", col = 2)
lines(1:length(sad_poilog), sad_poilog, type = "b", col = 3)
lines(1:length(sad_broken_stick), sad_broken_stick, type = "b", col = 4)
legend("topright", c("Log-series","Log-normal","Poisson log-normal","Broken stick"),
       col = 1:4, pch = 1)
```

`sim_thomas_community` *Simulate community with clumped spatial positions.*

Description

This function simulates a community with a certain abundance distribution and with intraspecific aggregation, i.e. individuals of the same species are distributed in clusters.

Usage

```
sim_thomas_community(
  s_pool,
  n_sim,
  sad_type = "lnorm",
  sad_coef = list(cv_abund = 1),
  fix_s_sim = FALSE,
  sigma = 0.02,
  cluster_points = NA,
  mother_points = NA,
  xmother = NA,
  ymother = NA,
  xrange = c(0, 1),
  yrange = c(0, 1),
  seed = NULL
)
```

Arguments

`s_pool`	Number of species in the pool (integer)
`n_sim`	Number of individuals in the simulated community (integer)
`sad_type`	Root name of the species abundance distribution model of the species pool (character) - e.g., "lnorm" for the lognormal distribution (`rlnorm`); "geom" for the geometric distribution (`rgeom`), or "ls" for Fisher's log-series distribution (`rls`). See the table in **Details** below, or `rsad` for all SAD model options.
`sad_coef`	List with named arguments to be passed to the distribution function defined by the argument `sad_type`. An overview of parameter names is given in the table below. In `mobsim` the log-normal and the Poisson log-normal distributions can alternatively be parameterized by the coefficient of variation (cv) of the relative abundances in the species pool. Accordingly, `cv_abund` is the standard deviation of abundances divided by the mean abundance (no. of individuals / no. of species). `cv_abund` is thus negatively correlated with the evenness of the species abundance distribution. Please note that the parameters *mu* and *sigma* are not equal to the mean and standard deviation of the log-normal distribution.
`fix_s_sim`	Should the simulation constrain the number of species in the simulated local community? (logical)
`sigma`	Mean displacement (along each coordinate axes) of a point from its mother point (= cluster centre). `Sigma` correlates with cluster extent. When `length(sigma) == length(abund_vec)`, each species receives a specific cluster extent. Otherwise, the first value of `sigma` is recycled and all species share the same cluster extent. When `sigma` of any species is more than twice as large as the largest plot dimension, a random Poisson distribution is simulated, which is more efficient than a Thomas cluster process. The parameter `sigma` corresponds to the `scale` parameter of the function `rThomas` in the package spatstat.random.

`cluster_points`	Mean number of points per cluster. If this is a single value, species have the same average number of points per cluster. If this is a vector of the same length as `abund_vec`, each species has a specific mean number of points per cluster. If no value is provided, the number of points per cluster is determined from the abundance and from `mother_points`. If `mother_points` and `cluster_points` are given OR `xmother` and `ymother`, and cluster points are given, `cluster_points` is overridden. If `mother_points`=0, there will be no clustering even if `cluster_points`=400 (high clustering) because `cluster_points` is overridden. The parameter `cluster_points` corresponds to the `mu` parameter of `spatstat.random::rThomas`.
`mother_points`	Number of mother points (= cluster centres). If this is a single value, all species have the same number of clusters. For example `mother_points = 1` can be used to simulate only one cluster per species, which then represents the complete species range. If `mother_points` is a vector of the same length as `abund_vec`, each species has a specific number of clusters. If `mother_points` equals 0 there is no clustering and the distribution is homogeneous. If no value is provided, the number of clusters is determined from the abundance and the number of points per cluster (`cluster_points`).
`xmother`	List of length equal to the number of species. Each list element is a vector of x coordinates for every mother points. If one element is NA, the the corresponding species is not clustered.
`ymother`	List of length equal to the number of species. Each list element is a vector of y coordinates for every mother points. If one element is NA, the the corresponding species is not clustered.
`xrange`	Extent of the community in x-direction. If this a numeric vector of length 2, all species share the same range. To specify different x ranges for all species, `xrange` should be a data.frame with 2 columns, min and max.
`yrange`	Extent of the community in y-direction. If this a numeric vector of length 2, all species share the same range. To specify different y ranges for all species, `xrange` should be a data.frame with 2 columns, min and max.
`seed`	Integer. Any integer passed to `set.seed` for reproducibility.

Details

This function consecutively calls `sim_sad` and `sim_thomas_coords`

See the documentations of `sim_sad` and `sim_thomas_coords` for details.

Value

A community object as defined by `community`

Author(s)

Felix May

Examples

```
com1 <- sim_thomas_community(s_pool = 20, n_sim = 500, sad_type = "lnorm",
                             sad_coef = list("meanlog" = 2, "sdlog" = 1),
                             sigma = 0.01)
plot(com1)
```

`sim_thomas_coords` *Simulate clumped spatial coordinates*

Description

Add clumped (aggregated) positions to a species abundance distribution. Clumping is simulated using a Thomas cluster process, also known as Poisson cluster process (Morlon et al. 2008, Wiegand & Moloney 2014)

Usage

```
sim_thomas_coords(
  abund_vec,
  sigma = 0.02,
  mother_points = NA,
  xmother = NA,
  ymother = NA,
  cluster_points = NA,
  xrange = c(0, 1),
  yrange = c(0, 1),
  seed = NULL
)
```

Arguments

`abund_vec`	Species abundance vector (integer)
`sigma`	Mean displacement (along each coordinate axes) of a point from its mother point (= cluster centre). `Sigma` correlates with cluster extent. When `length(sigma) == length(abund_vec)`, each species receives a specific cluster extent. Otherwise, the first value of `sigma` is recycled and all species share the same cluster extent. When `sigma` of any species is more than twice as large as the largest plot dimension, a random Poisson distribution is simulated, which is more efficient than a Thomas cluster process. The parameter `sigma` corresponds to the `scale` parameter of the function `rThomas` in the package spatstat.random.
`mother_points`	Number of mother points (= cluster centres). If this is a single value, all species have the same number of clusters. For example `mother_points` = 1 can be used to simulate only one cluster per species, which then represents the complete species range. If `mother_points` is a vector of the same length as `abund_vec`, each species has a specific number of clusters. If `mother_points` equals 0 there

	is no clustering and the distribution is homogeneous. If no value is provided, the number of clusters is determined from the abundance and the number of points per cluster (`cluster_points`).
`xmother`	List of length equal to the number of species. Each list element is a vector of x coordinates for every mother points. If one element is NA, the the corresponding species is not clustered.
`ymother`	List of length equal to the number of species. Each list element is a vector of y coordinates for every mother points. If one element is NA, the the corresponding species is not clustered.
`cluster_points`	Mean number of points per cluster. If this is a single value, species have the same average number of points per cluster. If this is a vector of the same length as `abund_vec`, each species has a specific mean number of points per cluster. If no value is provided, the number of points per cluster is determined from the abundance and from `mother_points`. If `mother_points` and `cluster_points` are given OR `xmother` and `ymother`, and cluster points are given, `cluster_points` is overridden. If `mother_points`=0, there will be no clustering even if `cluster_points`=400 (high clustering) because `cluster_points` is overridden. The parameter `cluster_points` corresponds to the `mu` parameter of `spatstat.random::rThomas`.
`xrange`	Extent of the community in x-direction. If this a numeric vector of length 2, all species share the same range. To specify different x ranges for all species, `xrange` should be a data.frame with 2 columns, min and max.
`yrange`	Extent of the community in y-direction. If this a numeric vector of length 2, all species share the same range. To specify different y ranges for all species, `xrange` should be a data.frame with 2 columns, min and max.
`seed`	Integer. Any integer passed to `set.seed` for reproducibility.

Details

To generate a Thomas cluster process of a single species this function uses a C++ re-implementation of the function `rThomas` in the package spatstat.random.

There is an inherent link between the parameters `abund_vec`, `mother_points`, and `cluster_points`. For every species the abundance has to be equal to the number of clusters (`mother_points`) times the number of points per cluster (`cluster_points`).

$$abundance = mother_{p}oints * cluster_{p}oints$$

Accordingly, if one of the parameters is provided, the other one is directly calculated from the abundance. Values for `mother_points` override values for `cluster_points`. If none of the parameters is specified, it is assumed that for every species there is a similar number of clusters and of points per cluster.

$$mother_{p}oints = cluster_{p}oints = \sqrt(abundance),$$

In this case rare species have few clusters with few points per cluster, while abundant species have many clusters with many points per cluster.

Value

A community object as defined by `community`.

Author(s)

Felix May, Alban Sagouis

References

Morlon et al. 2008. A general framework for the distance-decay of similarity in ecological communities. Ecology Letters 11, 904-917.

Wiegand and Moloney 2014. Handbook of Spatial Point-Pattern Analysis in Ecology. CRC Press

See Also

`rThomas`

Examples

```
abund <- c(10,20,50,100)
sim1 <- sim_thomas_coords(abund, sigma = 0.02)
plot(sim1)

# Simulate species "ranges"
sim2 <- sim_thomas_coords(abund, sigma = 0.02, mother_points = 1)
plot(sim2)

# Equal numbers of points per cluster
sim3 <- sim_thomas_coords(abund, sigma = 0.02, cluster_points = 5)
plot(sim3)

# With large sigma the distribution will be essentially random (see Details)
sim4 <- sim_thomas_coords(abund, sigma = 10)
plot(sim4)

# Some random and some clustered species with different numbers of mother points.
mother_points <- sample(0:3, length(abund), replace = TRUE)
sim5 <- sim_thomas_coords(abund, mother_points = mother_points, sigma=0.01)
plot(sim5)

# Specifying mother point coordinates or no-clustering (\code{NA}).
mother_points <- sample(1:3, length(abund), replace = TRUE)
xmother <- lapply(1:length(abund), function(i) runif(mother_points[i], 0, 1))
ymother <- lapply(1:length(abund), function(i) runif(mother_points[i], 0, 1))
xmother[[1]] <- NA
ymother[[1]] <- NA
sim6 <- sim_thomas_coords(abund, xmother=xmother, ymother=ymother, sigma=0.01)
plot(sim6)

# Species having different ranges.
xrange <- data.frame(t(sapply(1:length(abund), function(i) sort(runif(2, 0, 1)))))
```

```
yrange <- data.frame(t(sapply(1:length(abund), function(i) sort(runif(2, 0, 1)))))
sim7 <- sim_thomas_coords(abund, mother_points=1, sigma=1, xrange=xrange, yrange=yrange)
plot(sim7)
```

`spec_sample` *Sample species richness*

Description

Expected species richness in a random sample of fixed size.

Usage

```
spec_sample(abund_vec, n)
```

Arguments

`abund_vec`	Species abundance distribution of the community (integer vector)
`n`	Sample size in terms of number of individuals (integer)

Details

The expected number of species is calculated after Hurlbert 1971, Equation 3.

`spec_sample` is similar to the function `rarefy` in the R package `vegan`.

Value

Expected number of species in a sample of n individuals

References

Hurlbert, S.H. 1971. The nonconcept of species diversity: a critique and + alternative parameters. Ecology 52, 577-586.

Examples

```
sad1 <- sim_sad(100, 1000)
spec_sample(abund_vec = sad1, n = 20)
```

`spec_sample_curve` *Non-spatial and spatially-explicit species sampling curves*

Description

Expected species richness as function of sample size (no. of individuals), when individuals are sampled randomly (rarefaction) or when nearest-neighbours are samples (accumulation).

Usage

```
spec_sample_curve(comm, method = c("accumulation", "rarefaction"))
```

Arguments

`comm`	Community object
`method`	Partial match to `accumulation` or `rarefaction`. Also both methods can be included at the same time.

Details

Non-spatial sampling corresponds to the species rarefaction curve, which only depends on the species abundance distribution and can thus be also calculated from abundance data (see `rare_curve`).

In contrast the species-accumulation curve starts from a focal individual and only samples the nearest neighbours of the focal individual. The final species accumulation curves is calculated as the mean over the accumulation curves starting from all individuals.

In contrast to the rarefaction curve the accumulation curve is not only influenced by the species abundance distribution, but also by the spatial distribution of individuals.

Value

A dataframe with 2-3 columns. The first column indicates the sample size (numbers of individuals), and the second and third column indicate the expected species richness for spatial sampling (column: "spec_accum") and/or random sampling (column "spec_rarefied")

Examples

```
sim_com1 <- sim_thomas_community(s_pool = 100, n_sim = 1000)
sac1 <- spec_sample_curve(sim_com1, method = c("rare","acc"))

head(sac1)
plot(sac1)
```

`summary.community` *Print summary of spatial community object*

Description

Print summary of spatial community object

Usage

```
## S3 method for class 'community'
summary(object, digits = 2, ...)
```

Arguments

`object`	Community object of class `community`
`digits`	Integer. Number of digits to print
`...`	Additional arguments passed to `print`.

Value

This function is called for its side effects and has no return value.

`summary.sad` *Print summary of species abundance distribution object*

Description

Print summary of species abundance distribution object

Usage

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

Arguments

`object`	Community object of class sad
`...`	Additional arguments passed to `print`.

Value

This function is called for its side effects and has no return value.

See Also

`sim_sad`

Index