Package 'miaSim'

October 2, 2022

Type Package

Version 1.2.0

Title Microbiome Data Simulation

Description Microbiome time series simulation with generalized Lotka-Volterra model, Self-Organized Instability (SOI), and other models. Hubbell's Neutral model is used to determine the abundance matrix. The resulting abundance matrix is applied to SummarizedExperiment or TreeSummarizedExperiment objects.

License Artistic-2.0 | file LICENSE

biocViews Microbiome, Software, Sequencing, DNASeq, ATACSeq, Coverage, Network

Encoding UTF-8

LazyData false

RoxygenNote 7.1.2

Depends SummarizedExperiment, TreeSummarizedExperiment

Imports deSolve, stats, poweRlaw, gtools, S4Vectors, MatrixGenerics

Suggests rmarkdown, knitr, BiocStyle, testthat

URL https://github.com/microbiome/miaSim

BugReports https://github.com/microbiome/miaSim/issues

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/miaSim

git_branch RELEASE_3_15

git_last_commit 496a898

git_last_commit_date 2022-04-26

Date/Publication 2022-10-02

Author Karoline Faust [aut],
Yu Gao [aut],
Emma Gheysen [aut],
Daniel Rios Garza [aut],

Yagmur Simsek [cre, aut],
Leo Lahti [aut] (<https://orcid.org/0000-0001-5537-637X>)

Maintainer Yagmur Simsek <yagmur.simsek@hsrw.org>

R topics documented:

`convertToSE` *SummarizedExperiment*(`SE`) *or* `TreeSE` *construction function*

Description

Storing the data in SummarizedExperiment enables access to various tools for further analysis of data. A large number of Bioconductor packages contain extension of SummarizedExperiment class. SummarizedExperiment class offers data and metadata synchronization, while still accommodating specialized data structures for particular scientific applications.

Usage

```
convertToSE(matrix, output, ...)
```

Arguments

`matrix`	is a matrix-like or list of matrix-like object. Rows refer to taxa and columns refer to samples.
`output`	character value for storing the matrix in either TreeSummarizedExperiment (`output = TSE`) or SummarizedExperiment (default: `output = SE`)
`...`	: additional parameters that can be implemented in the `SE` object.

Details

Further examples for `SE` object manipulation and analysis can be found at https://microbiome.github.io/OMA/

The resulting abundance matrix from the simulation functions used in `miaSim` can be easily converted to SummarizedExperiment class object.

Value

SummarizedExperiment an object containing abundance matrix

Examples

```
ExampleHubbellRates <- simulateHubbellRates(
    community_initial = c(0,5,10), migration_p = 0.01,
    metacommunity_p = NULL, k_events = 1, growth_rates = NULL, norm = FALSE,
    t_end=1000)

HubbellSE <- convertToSE(matrix = ExampleHubbellRates$counts,
                         colData = ExampleHubbellRates$time,
                         metadata = ExampleHubbellRates$metadata)
```

---

powerlawA *Interaction matrix with Power-Law network adjacency matrix*

Description

Where N is the an Interspecific Interaction matrix with values drawn from a normal distribution H the interaction strength heterogeneity drawn from a power-law distribution with the parameter alpha, and G the adjacency matrix of with out-degree that reflects the heterogeneity of the powerlaw. A scaling factor s may be used to constrain the values of the interaction matrix to be within a desired range. Diagonal elements of A are defined by the parameter d.

Usage

```
powerlawA(n_species, alpha = 3, stdev = 1, s = 0.1, d = -1, symmetric = FALSE)
```

Arguments

n_species	integer number of species
alpha	numeric power-law distribution parameter. Should be > 1. (default: alpha = 3.0) Larger values will give lower interaction strength heterogeneity, whereas values closer to 1 give strong heterogeneity in interaction strengths between the species. In other words, values of alpha close to 1 will give Strongly Interacting Species (SIS).
stdev	numeric standard deviation parameter of the normal distribution with mean 0 from which the elements of the nominal interspecific interaction matrix N are drawn. (default: stdev = 1)
s	numeric scaling parameter with which the final global interaction matrix A is multiplied. (default: s = 0.1)
d	numeric diagonal values, indicating self-interactions (use negative values for stability). (default: s = 1.0)
symmetric	logical scalar returning a symmetric interaction matrix (default: symmetric=FALSE)

Value

The interaction matrix A with dimensions (n_species x n_species)

References

Gibson TE, Bashan A, Cao HT, Weiss ST, Liu YY (2016) On the Origins and Control of Community Types in the Human Microbiome. PLOS Computational Biology 12(2): e1004688. https://doi.org/10.1371/journal.pcbi.1004

Examples

```
# Low interaction heterogeneity
A_low <- powerlawA(n_species = 10, alpha = 3)
# Strong interaction heterogeneity
A_strong <- powerlawA(n_species = 10, alpha = 1.01)
```

`randomA` *Generate random uniform interaction matrix*

Description

Generate random simplified interaction matrix from a uniform distribution.

Usage

```
randomA(
  n_species,
  d = -0.5,
  min_strength = -0.5,
  max_strength = 0.5,
  connectance = 0.02,
  symmetric = FALSE
)
```

Arguments

`n_species`	integer number of species
`d`	numeric diagonal values (should be negative) (default: `d = -0.5`)
`min_strength`	numeric value of minimal off-diagonal interaction strength (default: `min_strength = -0.5`)
`max_strength`	numeric value of maximal off-diagonal interaction strength (default: `max_strength = 0.5`)
`connectance`	numeric interaction probability (default: `connectance = 0.02`)
`symmetric`	logical scalar returning a symmetric interaction matrix (default: `symmetric=FALSE`)

Value

randomA returns a matrix A with dimensions (n_species x n_species)

Examples

```
high_inter_A <- randomA(n_species = 10, d = -0.4, min_strength = -0.8,
                                    max_strength = 0.8, connectance = 0.5)

low_inter_A <- randomA(n_species = 10, connectance = 0.01)
```

---

randomE *Generate random efficiency matrix*

Description

Generate random efficiency matrix for consumer resource model from Dirichlet distribution. Positive efficiencies indicate the consumption of resources, whilst negatives indicate that the species would produce the resource.

Usage

```
randomE(
  n_species,
  n_resources,
  min_con = round(n_resources/4),
  max_con = round(n_resources/3),
  min_prod = round(n_resources/6),
  max_prod = round(n_resources/4),
  maintenance = 0.5
)
```

Arguments

n_species	integer number of species
n_resources	integer number of resources
min_con	integer minimum number of resources consumed by each species
max_con	integer maximum number of resources consumed by each species
min_prod	integer minimum number of resources produced by each species
max_prod	integer maximum number of resources produced by each species
maintenance	numeric value between 0~1 the proportion of resources used to maintain the living of microorganisms. 0 means all the resources will be used for the reproduction of microorganisms, and 1 means all the resources would be used to maintain the living of organisms and no resources would be left for their growth(reproduction).

Value

randomE returns a matrix E with dimensions (n_species x n_resources), and each row represents a species.

Examples

```
# example with minimum parameters
ExampleEfficiencyMatrix2 <- randomE(n_species = 5, n_resources = 12)
```

---

simulateConsumerResource

Consumer-resource model simulation

Description

Simulates a community time series using the consumer-resource model. The change rate of each species was defined as dx/dt = mumax*sum(monod)*X, where mumax is the vector of maximum growth rates for the species, monod is the monod growth rate, S/(Ks+S), where S is the concentration of the limiting resource, and Ks is the half-velocity constant for species X and S. X is the vector of abundances of species. The concentrations of resource will be set to 0 if they were calculated less than 0 during the iteration.

Usage

```
simulateConsumerResource(
  n_species,
  n_resources,
  eff = randomE(n_species, n_resources),
  consumers = runif(n = n_species, min = 0.1, max = 10),
  resources = runif(n = n_resources, min = 1, max = 100),
  mumax = rep(1, n_species),
 k_table = matrix(rgamma(n = n_species * n_resources, shape = 50, rate = 0.25), nrow =
    n_species),
  t_end = 1000,
  ...
)
```

Arguments

n_species	integer number of species
n_resources	interger number of resources
eff	a matrix of efficiency. How efficient are resources converted into biomass, negative values represent excreted resources (default: eff = randomE(n_species, n_resources))
consumers	numeric vector of species (default: consumers = runif(n = n_species, min = 0.1, max = 10))

resources	numeric vector of resources (default: resources = runif(n = n_resources, min = 1, max = 100))
mumax	numeric vector of maximum mu of species (default: mumax = rep(1, n_species))
k_table	a matrix of K values in monod model (default: k_table = matrix(rgamma(n=n_species*n_resources, = 50, rate = 0.25), nrow = n_species))
t_end	numeric scalar indicating the final time of the simulation (default: t_end = 1000)
...	additional parameters including 't_start', 't_step', and 't_store'

Value

an abundance matrix with species and resources abundance as rows and time points as columns

See Also

convertToSE

Examples

```
# example1 users provide least parameters.
ExampleConsumerResource <- simulateConsumerResource(n_species = 2,
n_resources = 4)
```

---

simulateGLV *Generalized Lotka-Volterra (gLV) simulation*

Description

Simulates time series with the generalized Lotka-Volterra model.

Usage

```
simulateGLV(
  n_species,
  A,
  x = runif(n_species),
  b = runif(n_species),
  sigma_drift = 0.01,
  sigma_epoch = 0.3,
  sigma_external = 0.3,
  p_epoch = 0.01,
  t_external_events = c(12, 36, 48),
  t_external_durations = c(3, 10, 99),
  stochastic = FALSE,
  norm = FALSE,
  t_end = 1000,
  ...
)
```

Arguments

n_species	integer number of species
A	interaction matrix
x	numeric initial abundances
b	numeric growth rates
sigma_drift	numeric degree of drift (turnover of species) in each time step. (default: sigma_drift = 0.01)
sigma_epoch	numeric degree of epoch change of community (default: sigma_epoch = 0.3)
sigma_external	numeric degree of the external events/disturbances (default: sigma_external = 0.3)
p_epoch	numeric value of the probability/frequency of inherit periodic changes of community (default: p_epoch = 0.01)
t_external_events	numeric value of starting times of external events (default: t_external_events = c(12, 36, 48))
t_external_durations	numeric durations of external events (default: t_external_durations = c(3, 10, 99))
stochastic	logical scalar choosing whether the gLV model should be stochastic (default: stochastic = FALSE)
norm	logical scalar returning normalised abundances (proportions in each generation) (default: norm = FALSE)
t_end	numeric value of simulation end time (default: t_end = 1000)
...	additional parameters including 't_start', 't_step', and 't_store'

Details

Simulates a community time series using the generalized Lotka-Volterra model, defined as dx/dt = x(b+Ax), where x is the vector of species abundances, diag(x) is a diagonal matrix with the diagonal values set to x. A is the interaction matrix and b is the vector of growth rates.

Value

simulateGLV returns an abundance matrix

See Also

convertToSE

Examples

```
A <- miaSim::powerlawA(4, alpha = 1.01)

ExampleGLV <- simulateGLV(n_species = 4, A, t_end = 1000)
```

`simulateHubbell` *Hubbell's neutral model simulation*

Description

Neutral species abundances simulation according to the Hubbell model.

Usage

```
simulateHubbell(
  n_species,
  M,
  I = 1000,
  d = 10,
  m = 0.02,
  tskip = 0,
  tend,
  norm = FALSE
)
```

Arguments

`n_species`	integer amount of different species initially in the local community
`M`	integer amount of different species in the metacommunity, including those of the local community
`I`	integer value of fixed amount of individuals in the local community (default: `I = 1000`)
`d`	integer value of fixed amount of deaths of local community individuals in each generation (default: `d = 10`)
`m`	numeric immigration rate: the probability that a death in the local community is replaced by a migrant of the metacommunity rather than by the birth of a local community member (default: `m = 0.02`)
`tskip`	integer number of generations that should not be included in the outputted species abundance matrix. (default: `tskip = 0`)
`tend`	integer number of simulations to be simulated
`norm`	logical scalar choosing whether the time series should be returned with the abundances as proportions (`norm = TRUE`) or the raw counts (default: `norm = FALSE`)

Value

`simulateHubbell` returns an abundance matrix with species abundance as rows and time points as columns

References

Rosindell, James et al. "The unified neutral theory of biodiversity and biogeography at age ten." Trends in ecology & evolution vol. 26,7 (2011).

See Also

convertToSE

Examples

```
ExampleHubbell <- simulateHubbell(n_species = 8, M = 10, I = 1000, d = 50,
                                                          m = 0.02, tend = 100)
```

---

simulateHubbellRates *Hubbell's neutral model simulation applied to time series*

Description

Neutral species abundances simulation according to the Hubbell model. This model shows that losses in society can be replaced either by the birth of individuals or by immigration depending on their probabilities. The specific time between the events of birth or migration is calculated and time effect is considered to determine the next event.

Usage

```
simulateHubbellRates(
  community_initial,
  migration_p = 0.1,
  metacommunity_p = NULL,
  k_events = 1,
  growth_rates = NULL,
  norm = FALSE,
  t_end = 1000,
  list = TRUE,
  ...
)
```

Arguments

community_initial	numeric value a vector of integers, containing species counts greater or equal to zero.
migration_p	numeric immigration possibility. It defines the probability of migration when replacement is needed in the community. The value can be between 0 and 1. The sum of the probability of migration and the probability birth must be 1.
metacommunity_p	numeric value the probability of a species being found in the metacommunity.
k_events	integer number of steps performed at a time point. It can be equal or more than 1. Bigger k_events increases speed while decreasing precision.
growth_rates	numeric rate of change in community size.

norm	logical scalar choosing whether the time series should be returned with the abundances as proportions (norm = TRUE) or the raw counts (default: norm = FALSE)
t_end	numeric value of simulation end time (default: t_end = 1000)
list	logical scalar deciding whether output is a list object or not (default: norm = TRUE)
...	additional parameters including 't_start', 't_step', and 't_store'

Value

a community abundance matrix or a list object that contains growth rates, time points and metacommunity probabilities

References

Rosindell J, Hubbell SP, Etienne RS. The unified neutral theory of biodiversity and biogeography at age ten. Trends Ecol Evol. 2011 Jul;26(7):340-8. doi: 10.1016/j.tree.2011.03.024. Epub 2011 May 10. PMID: 21561679.

See Also

convertToSE

Examples

```
ExampleHubbellRates <- simulateHubbellRates(community_initial = c(0,5,10),
             migration_p = 0.01, metacommunity_p = NULL, k_events = 1,
             growth_rates = NULL, norm = FALSE, t_end=1000)
```

---

simulateRicker *Generate time series with the Ricker model*

Description

The Ricker model is a discrete version of the generalized Lotka-Volterra model and is implemented here as proposed by Fisher and Mehta in PLoS ONE 2014.

Usage

```
simulateRicker(
  n_species,
  A,
  x = runif(n_species),
  K = runif(n_species),
  sigma = 0.05,
  explosion_bound = 10^8,
  tskip = 0,
```

```
  tend,
  norm = FALSE
)
```

Arguments

n_species	integer number of species
A	interaction matrix
x	numeric initial abundances
K	numeric carrying capacities
sigma	numeric value of noise level, if set to a non-positive value, no noise is added (default: sigma = 0.05)
explosion_bound	numeric value of boundary for explosion (default: explosion_bound = 10^8)
tskip	integer number of generations that should not be included in the outputted species abundance matrix (default: tskip = 0)
tend	integer number of simulations to be simulated
norm	logical scalar returning normalised abundances (proportions in each generation) (default: norm = FALSE)

Value

simulateRicker returns an abundance matrix with species abundance as rows and time points as columns

References

Fisher & Mehta (2014). Identifying Keystone Species in the Human Gut Microbiome from Metagenomic Timeseries using Sparse Linear Regression. PLoS One 9:e102451

See Also

convertToSE

Examples

```
A <- powerlawA(10, alpha = 1.01)
ExampleRicker <- simulateRicker(n_species=10,A,tend=100)
```

simulateSOI *Self-Organised Instability model (SOI) simulation*

Description

Generate time-series with The Self-Organised Instability (SOI) model. Implements a K-leap method for accelerating stochastic simulation.

Usage

```
simulateSOI(n_species, I, A, k = 5, com = NULL, tend, norm = FALSE)
```

Arguments

n_species	integer number of species
I	integer community size, number of available sites (individuals)
A	interaction matrix
k	integer number of transition events that are allowed to take place during one leap. (default: k = 5). Higher values reduce runtime, but also accuracy of the simulation.
com	a vector of initial community abundances If (default: com = NULL), based on migration rates
tend	integer timepoints to be returned in the time series (number of generations)
norm	logical scalar indicating whether the time series should be returned with the abundances as proportions (norm = TRUE) or the raw counts (default: norm = FALSE)

Value

abundance matrix consisting of species abundance as rows and time points as columns

See Also

convertToSE

Examples

```
A <- miaSim::powerlawA(10, alpha = 1.2)

ExampleSOI <- simulateSOI(n_species = 10, I = 1000, A, k=5, com = NULL,
                                          tend = 150, norm = TRUE)
```

simulateStochasticLogistic

Stochastic Logistic simulation

Description

Simulates a community time series using the logistic model. The change rate of the species was defined as `dx/dt = b*x*(1-(x/k))*rN - dr*x`, where b is the vector of growth rates, x is the vector of initial species abundances, k is the vector of maximum carrying capacities, rN is a random number ranged from 0 to 1 which changes in each time step, dr is the vector of constant death rates. Also, the vectors of initial dead species abundances can be set. The number of species will be set to 0 if the dead species abundances surpass the alive species abundances.

Usage

```
simulateStochasticLogistic(
  n_species,
  b = runif(n = n_species, min = 0.1, max = 0.2),
  k = runif(n = n_species, min = 1000, max = 2000),
  dr = runif(n = n_species, min = 5e-04, max = 0.0025),
  x = runif(n = n_species, min = 0.1, max = 10),
  sigma_drift = 0.001,
  sigma_epoch = 0.1,
  sigma_external = 0.3,
  p_epoch = 0.001,
  t_external_events = c(0, 240, 480),
  t_external_durations = c(0, 1, 1),
  stochastic = TRUE,
  t_end = 1000,
  ...
)
```

Arguments

`n_species`	integer number of species
`b`	numeric growth rates (default: `b = runif(n = n_species, min = 0.1, max = 0.2)`)
`k`	numeric value of carrying capacities (default: `k = runif(n = n_species, min = 1000, max = 2000)`)
`dr`	numeric value of death rates (default: `dr = runif(n = n_species, min = 0.0005, max = 0.0025)`)
`x`	numeric initial abundances (default: `x = runif(n = n_species, min = 0.1, max = 10)`)
`sigma_drift`	numeric degree of drift (turnover of species) in each time step. (default: `sigma_drift = 0.001`)
`sigma_epoch`	numeric degree of epoch change of community (default: `sigma_epoch = 0.1`)

sigma_external	numeric degree of external events/disturbances (default: sigma_external = 0.3)
p_epoch	numeric value of probability/frequency of inherit periodic changes of community (default: p_epoch = 0.001)
t_external_events	numeric value of starting times of external events (default: t_external_events = c(0, 240, 480))
t_external_durations	numeric value of durations of external events (default: t_external_durations = c(0, 1, 1))
stochastic	logical scalar choosing whether the logistic model should be stochastic (controlled by multiplying the growth rate by a random number) (default: stochastic = TRUE)
t_end	numeric final time of the simulation (default: t_end = 1000)
...	additional parameters including 't_start', 't_step', and 't_store'

Value

simulateStochasticLogistic returns an abundance matrix with species abundance as rows and time points as columns

See Also

convertToSE

Examples

```
## ATTENTION: Don't set a large value to t.step, otherwise the computer won't
#give a correct solution to the logistic ODE(ordinary differential equation).
#Keeping t_step under 0.05 or 0.01 is a good practice.

#while (!exists("ExampleLogistic"))
ExampleLogistic <- simulateStochasticLogistic(n_species = 5)
#plot the calculated points
matplot(ExampleLogistic, type = "l")

#calculation by setting initial parameters explicitly
ExampleLogistic2 <- simulateStochasticLogistic(n_species = 2,
b = c(0.2, 0.1), k = c(1000, 2000), dr = c(0.001, 0.0015), x = c(3, 0.1),
sigma_drift = 0.001, sigma_epoch = 0.3, sigma_external = 0.5,p_epoch = 0.001,
t_external_events = c(100, 200, 300), t_external_durations = c(1, 2, 3),
t_start = 0, t_end = 1500, t_step = 0.01,
t_store = 1500, stochastic = TRUE)
```

Index