Package 'EcoSimR'

April 3, 2015

Type Package

Title Null Model Analysis for Ecological Data

Version 0.1.0

Date 2015-04-02

BugReports https://github.com/GotelliLab/EcoSimR/issues

LazyLoad yes

LazyData yes

Description Given a site by species interaction matrix, users can make inferences about species interactions by performance hypothesis comparing test statistics against a null distribution. The current package provides algorithms and metrics for niche-overlap, body size ratios and species co-occurrence. Users can also integrate their own algorithms and metrics within these frameworks or completely novel null models. Detailed explanations about the underlying assumptions of null model analysis in ecology can be found at http://ecosimr.org.

License MIT + file LICENSE

Depends MASS

Suggests testthat, knitr

VignetteBuilder knitr

NeedsCompilation no

Author Nick Gotelli [aut],
Edmund Hart [aut, cre],
Aaron Ellison [aut]

Maintainer Edmund Hart <edmund.m.hart@gmail.com>

Repository CRAN

Date/Publication 2015-04-03 22:09:00

R topics documented:

checker 3
cooc_null_model 4
czekanowski 5

czekanowski_skew . . . 6
czekanowski_var . . . 7
c_score . . . 8
c_score_skew . . . 9
c_score_var . . . 10
dataMacWarb . . . 11
dataRodents . . . 12
dataWiFinches . . . 12
EcoSimR . . . 13
min_diff . . . 13
min_ratio . . . 14
niche_null_model . . . 14
null_model_engine . . . 16
pianka . . . 17
pianka_skew . . . 18
pianka_var . . . 19
plot.coocnullmod . . . 20
plot.nichenullmod . . . 21
plot.nullmod . . . 21
plot.sizenullmod . . . 22
ra1 . . . 23
ra2 . . . 24
ra3 . . . 25
ra4 . . . 26
ranMatGen . . . 27
reproduce_model . . . 29
sim1 . . . 30
sim10 . . . 31
sim2 . . . 32
sim3 . . . 33
sim4 . . . 34
sim5 . . . 35
sim6 . . . 36
sim7 . . . 37
sim8 . . . 38
sim9 . . . 39
sim9_single . . . 41
size_gamma . . . 41
size_null_model . . . 42
size_source_pool . . . 43
size_uniform . . . 44
size_uniform_user . . . 45
species_combo . . . 46
summary.coocnullmod . . . 47
summary.nichenullmod . . . 47
summary.nullmod . . . 48
summary.sizenullmod . . . 48
var_diff . . . 49

var_ratio 49
vector_sample 50
v_ratio 51

Index **53**

`checker` *Checker Co-occurrence Metric*

Description

Function to calculate number of unique pairs of species that never co-occur and form a "checkerboard pair".

Usage

```
checker(m = matrix(rbinom(100, 1, 0.5), nrow = 10))
```

Arguments

`m`	A binary presence-absence matrix in which rows are species and columns are sites.

Details

In Diamond's (1975) assembly rules model, pairs of species that never co-occur in any site are interpreted as examples of interspecific competition. A set of communities structured this way should contain more checkerboard pairs than expected by chance.

Value

Returns the number of unique species pairs that never co-occur.

References

Diamond, J.M. 1975. Assembly of species communities. p. 342-444 in: Ecology and Evolution of Communities. M.L. Cody and J.M. Diamond (eds.). Harvard University Press, Cambridge.

Connor, E.F. and D. Simberloff. 1979. The assembly of species communities: chance or competition? Ecology 60: 1132-1140.

Examples

```
obsChecker <- checker(m=matrix(rbinom(100,1,0.5),nrow=10))
```

cooc_null_model *Co-Occurrence Null model*

Description

Create a Co-Occurrence null model

Usage

```
cooc_null_model(speciesData, algo = "sim9", metric = "c_score",
  nReps = 1000, saveSeed = FALSE, burn_in = 500, algoOpts = list(),
  metricOpts = list(), suppressProg = FALSE)
```

Arguments

speciesData	a dataframe in which rows are species, columns are sites, and the entries indicate the absence (0) or presence (1) of a species in a site. Empty rows and empty columns should not be included in the matrix.
algo	the algorithm to use, must be "sim1", "sim2", "sim3", "sim4", "sim5", "sim6", "sim7", "sim8", "sim9", "sim10"; default is "sim9".
metric	the metric used to calculate the null model: choices are "species_combo", "checker", "c_score", "c_score_var", "c_score_skew", "v_ratio"; default is "c_score".
nReps	the number of replicate null assemblages to create; default is 1000 replicates.
saveSeed	TRUE or FALSE. If TRUE the current seed is saved so the simulation can be repeated; default is FALSE.
burn_in	The number of burn_in iterations to use with the simFast algorithm; default is 500 burn-in replicates.
algoOpts	a list containing all the options for the specific algorithm you want to use. Must match the algorithm given in the 'algo' argument.
metricOpts	a list containing all the options for the specific metric you want to use. Must match the metric given in the 'metric' argument.
suppressProg	TRUE or FALSE. If true, display of the progress bar in the console is suppressed; default is FALSE. This setting is useful for creating markdown documents with 'knitr'.

Examples

```
## Not run:

## Run the null model
finchMod <- cooc_null_model(dataWiFinches, algo="sim9",nReps=10000,burn_in = 500)
## Summary and plot info
summary(finchMod)
plot(finchMod,type="burn_in")
plot(finchMod,type="hist")
```

```
plot(finchMod,type="cooc")

## Example that is repeatable with a saved seed
finchMod <- cooc_null_model(dataWiFinches, algo="sim1",saveSeed = TRUE)
mean(finchMod$Sim)
## Run the model with the seed saved

finchMod <- cooc_null_model(dataWiFinches, algo="sim1",saveSeed=T)
## Check model output
mean(finchMod$Sim)

reproduce_model(finchMod$Sim)

finchMod <- cooc_null_model(dataWiFinches, algo="sim1")
## Check model output is the same as before
mean(finchMod$Sim)
reproduce_model(finchMod$Sim)



## End(Not run)
```

`czekanowski` *Czekanowski Niche Overlap Metric*

Description

Takes a resource utilization matrix as input and returns the average pairwise Czekanowki niche overlap index.

Usage

```
czekanowski(m = matrix(rpois(80, 1), nrow = 10))
```

Arguments

`m`	a matrix of resource utilization values.

Details

The Czekanowski niche overlap index is averaged over each unique species pair. The index measures the area of intersection of the resource utilization histograms of each species pair. Values of Czekanowski niche overlap index close to 0.0 reflect usage of exclusive resource categories, whereas values close to 1.0 reflect similar resource utilization spectra.

$$O_{jk} = O_{kj} = 1 - 0.5 \sum_{i=1}^{n} |p_{ij} - p_{ik}|$$

,

Value

Returns the average pairwise niche overlap.

Note

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

References

Feinsinger, P., E.E. Spears, and R. Poole. 1981. A simple measure of niche breadth. Ecology 62: 27-32.

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

See Also

`pianka` niche overlap index.

Examples

```
obsOverlap <- czekanowski(m=matrix(rpois(40,0.5),nrow=8))
```

`czekanowski_skew` *CzekanowskiSkew Niche Overlap Metric*

Description

Takes a niche utilization matrix returns the skew of the Czekanowski pairwise niche overlap index.

Usage

```
czekanowski_skew(m = matrix(rpois(80, 1), nrow = 10))
```

Arguments

`m`	a matrix of resource utilization values.

Details

A large positive value for skewness implies that there are more species pairs with high than low niche overlap. A large negative value for skewness implies there are more species pairs with low than high niche overlap. The performance of this algorithm has not been thoroughly tested with real data sets.

Value

Returns the skewness of the average pairwise niche overlap.

Note

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

References

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

See Also

`czekanowski` niche overlap index.

Examples

```
obsSkew <- czekanowski_skew(m=matrix(rpois(40,0.5),nrow=8))
```

---

czekanowski_var *CzekanowskiVariance Niche Overlap Metric*

Description

Takes a niche utilization matrix returns the variance of the Czekanowski niche overlap index

Usage

```
czekanowski_var(m = matrix(rpois(80, 1), nrow = 10))
```

Arguments

`m`	a matrix of resource utilization values.

Details

A large value for variance implies that some species pairs show high niche overlap and others show low niche overlap. A low value for variance implies that niche overlap (high or low) is very similar among all species pairs.

Value

Returns the variance of the average pairwise niche overlap.

Note

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

References

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

See Also

`czekanowski` niche overlap index.

Examples

```
obsVar <- czekanowski_var(m=matrix(rpois(40,0.5),nrow=8))
```

---

c_score *CScore Co-occurrence Metric*

Description

Takes a binary presence-absence matrix and returns Stone and Roberts' (1990) C-score.

Usage

```
c_score(m = matrix(rbinom(100, 1, 0.5), nrow = 10))
```

Arguments

`m`	a binary presence-absence matrix in which rows are species and columns are sites.

Details

For each unique pair of species, the C-score is calculated as

$$C_{ij} = (R_i - S)(R_j - S)$$

where R_i and R_j are the row sums for species i and j, and S is the number of shared sites in which both species i and species j are present. For any particular species pair, the larger the C-score, the more segregated the pair, with fewer shared sites. However, the index can be difficult to interpret when calculated as a matrix-wide average, because a single matrix can contain individual pairs of species that are segregated, random, or aggregated.

Degenerate matrices result from simulations where a row or column sum may be 0. <nick can you fill in the implications as to what this means if they are included or not?>

Value

Returns the average C-score calculated across all possible species pairs in the matrix.

Note

The matrix-wide C-score is not calculated for missing species, so empty rows in the matrix do not affect the result.

References

Stone. L. and A. Roberts. 1990. The checkerboard score and species distributions. Oecologia 85: 74-79.

Gotelli, N.J. and W. Ulrich. 2010. The empirical Bayes approach as a tool to identify non-random species associations. Oecologia 162:463-477.

Examples

```
obsCScore <- c_score(m=matrix(rbinom(100,1,0.5),nrow=10))
```

---

c_score_skew *CScoreSkew Co-occurrence Metric*

Description

Takes a binary presence-absence matrix and returns the skewness of the Stone and Roberts' (1990) C-score.

Usage

```
c_score_skew(m = matrix(rbinom(100, 1, 0.5), nrow = 10))
```

Arguments

m	a binary presence-absence matrix in which rows are species and columns are sites.

Details

A large positive value of skewness implies a preponderance of species pairs with large C-score values (segregated), whereas a large negative value of skewness implies a preponderance of species pairs with small C-score values (aggregated).

Value

Returns the skewness of the C-score calculated across all possible species pairs in the matrix.

Note

The matrix-wide C-score is not calculated for missing species, so empty rows in the matrix do not affect the result. This index has not been thoroughly tested with real data sets.

References

Stone, L. and A. Roberts. 1990. The checkerboard score and species distributions. Oecologia 85: 74-79.

Stone, L. and A. Roberts. 1992. Competitive exclusion, or species aggregation? An aid in deciding. Oecologia 91: 419-424.

See Also

`c_score` co-occurrence index.

Examples

```
skewCScore <- c_score_skew(m=matrix(rbinom(100,1,0.5),nrow=10))
```

---

c_score_var *CScoreVariance Co-occurrence Metric*

Description

Takes a binary presence-absence matrix and returns the variance of the Stone and Roberts' (1990) C-score.

Usage

```
c_score_var(m = matrix(rbinom(100, 1, 0.5), nrow = 10))
```

Arguments

`m`	a binary presence-absence matrix in which rows are species and columns are sites.

Details

A large value of this variance implies that some species pairs in the matrix are strongly segregated (large C-score) and other species pairs are random or aggregated.

Value

Returns the variance of the C-score calculated across all possible species pairs in the matrix.

Note

The matrix-wide C-score is not calculated for missing species, so empty rows in the matrix do not affect the result. This index has not been thoroughly tested with real data sets.

References

Stone, L. and A. Roberts. 1990. The checkerboard score and species distributions. Oecologia 85: 74-79.

Stone, L. and A. Roberts. 1992. Competitive exclusion, or species aggregation? An aid in deciding. Oecologia 91: 419-424.

See Also

c_score co-occurrence index.

Examples

```
varCScore <- c_score_var(m=matrix(rbinom(100,1,0.5),nrow=10))
```

`dataMacWarb` *MacArthur's (1958) warbler data This data matrix is from MacArthur's classic (1958) paper on the coexistence of 5 species of New England warbler. Each row of the data matrix is a different species of warbler, and each column is one of 16 different subregions of a coniferous tree. Each entry is the percentage of time that each species was observed foraging in a different subregion of the tree (see Figures 2-4 in MacArthur 1958). Zeroes indicate subregions of the tree in which a species was not recorded foraging.*

Description

MacArthur's (1958) warbler data This data matrix is from MacArthur's classic (1958) paper on the coexistence of 5 species of New England warbler. Each row of the data matrix is a different species of warbler, and each column is one of 16 different subregions of a coniferous tree. Each entry is the percentage of time that each species was observed foraging in a different subregion of the tree (see Figures 2-4 in MacArthur 1958). Zeroes indicate subregions of the tree in which a species was not recorded foraging.

References

MacArthur, R.H. 1958. Population ecology of some warblers of northeastern coniferous forests. Ecology 39: 599-699.

`dataRodents` *Desert rodent data set This data vector is from Brown's (1975) study of the coexistence of desert rodent species. Each entry is the average adult body mass in grams of six co-occurring species of Sonoran Desert rodents.*

Description

Desert rodent data set This data vector is from Brown's (1975) study of the coexistence of desert rodent species. Each entry is the average adult body mass in grams of six co-occurring species of Sonoran Desert rodents.

References

Brown, J.H. 1975. Geographical ecology of desert rodents. p. 314-341 in: Ecology and Evolution of Communities. M.L. Cody and J.M. Diamond (eds.). Harvard University Press, Cambridge.

`dataWiFinches` *West Indian Finches data This data frame is a binary presence-absence matrix for West Indies finches (Fringillidae). Each row is a different species of finch and each column is one of the 19 major islands in the West Indies. Entries indicate the presence (1) or absence (0) of a species on an island. Data from Gotelli and Abele (1982).*

Description

West Indian Finches data This data frame is a binary presence-absence matrix for West Indies finches (Fringillidae). Each row is a different species of finch and each column is one of the 19 major islands in the West Indies. Entries indicate the presence (1) or absence (0) of a species on an island. Data from Gotelli and Abele (1982).

References

Gotelli, N.J. and L.G. Abele. 1982. Statistical distributions of West Indian land bird families. Journal of Biogeography 9: 421-435.

EcoSimR — *EcoSimR EcoSimR is a collection of functions for calculating community metrics and algorithms for randomizing community data for null model analysis. Current modules are included for the analysis of niche overlap, body size overlap, and species co-occurrence. EcoSimR also allows users to define their own functions and algorithms to develop new null models.*

Description

EcoSimR EcoSimR is a collection of functions for calculating community metrics and algorithms for randomizing community data for null model analysis. Current modules are included for the analysis of niche overlap, body size overlap, and species co-occurrence. EcoSimR also allows users to define their own functions and algorithms to develop new null models.

---

min_diff — *MinDiff Size Overlap Metric*

Description

Function to calculate the minimum absolute size difference between species within an assemblage.

Usage

```
min_diff(m = runif(20))
```

Arguments

m	a vector of non-negative trait measures, one for each species

Details

Although this index is typically used to examine body size differences in an animal assemblage, it could be used for any morphological index, such as beak size, or for a phenological "trait", such as peak flowering time in a plant assemblage.

Value

Returns the minimum difference between adjacent, ordered values.

References

Simberloff, D. and W.J. Boecklen. 1981. Santa Rosalia reconsidered: size ratios and competition. Evolution 35: 1206-1228.

Examples

```
MinSizeDif <- min_diff(rgamma(20,shape=3,scale=2))
```

min_ratio *MinRatio Size Overlap Ratio Metric*

Description

Function to calculate the minimum size ratio (larger/next larger) between species within an assemblage.

Usage

```
min_ratio(m = runif(20))
```

Arguments

m	a vector of non-negative trait measures, one for each species

Details

This index is based on the minimum size ratio (larger/next larger) difference between consecutively ordered species in an assemblage. It is appropriate for morphological traits, but not phenological ones.

Value

Returns the minimum size ratio difference between adjacent, ordered values.

References

Simberloff, D. and W.J. Boecklen. 1981. Santa Rosalia reconsidered: size ratios and competition. Evolution 35: 1206-1228.

Examples

```
MinSizeDif <- min_ratio(rgamma(20,shape=3,scale=2))
```

niche_null_model *Niche overlap null models*

Description

Create a null model for niche overlap; choices of algorithm and metric are constrained to be valid for niche null models.

Usage

```
niche_null_model(speciesData, algo = "ra3", metric = "pianka",
  nReps = 1000, saveSeed = FALSE, algoOpts = list(),
  metricOpts = list(), suppressProg = FALSE)
```

Arguments

speciesData	a data frame in which each row is a species, each column is a resource utilization category, and the entries represent the quantity of the resource used by each species. Examples might be the amount of time a species spends foraging in different microhabitats, the biomass of different prey types, or counts of the number of times an adult female oviposits eggs on different species of a host plant.
algo	the algorithm to use, must be "ra1", "ra2", "ra3", "ra4"; default is "ra3".
metric	the metric used to calculate the null model: choices are "pianka", "czekanowski", "pianka_var", "czekanowski_var", "pianka_skew", "czekanowski_skew"; default is "pianka".
nReps	the number of replicate null assemblages to create; default is 1000 replicates.
saveSeed	TRUE or FALSE. If TRUE the current seed is saved so the simulation can be repeated; default is FALSE.
algoOpts	a list containing all the options for the specific algorithm you want to use. Must match the algorithm given in the 'algo' argument.
metricOpts	a list containing all the options for the specific metric you want to use. Must match the metric given in the 'metric' argument.
suppressProg	TRUE or FALSE. If true, display of the progress bar in the console is suppressed; default is FALSE. This setting is useful for creating markdown documents with 'knitr'.

Examples

```
## Not run:
## Load MacAruthur warbler data
data(dataMacWarb)

## Run the null model
warbMod <- niche_null_model(dataMacWarb,nReps=1000)
## Summary and plot info
summary(warbMod)
plot(warbMod)
plot(warbMod,type="niche")

## End(Not run)
```

`null_model_engine` *Run null model*

Description

This function drives all the different kinds of null models that can be run. It is the underlying engine.

Usage

```
null_model_engine(speciesData, algo, metric, nReps = 1000, saveSeed = FALSE,
  algoOpts = list(), metricOpts = list(), type = NULL,
  suppressProg = FALSE)
```

Arguments

`speciesData`	a dataframe for analysis that is compatable with the metrics and algorithms used.
`algo`	the algorithm used to randomize the data.
`metric`	the metric used to quantify pattern in the data.
`nReps`	the number of null assemblages to simulate.
`saveSeed`	Save the existing random seed to allow the user to reproduce the exact model results. The default value is FALSE, in which case the random number seed that is created is not stored in the output.
`algoOpts`	a list containing options for a supplied alogrithm.
`metricOpts`	a list containing options for a supplied metric.
`type`	The type of null model being run. If the null model is intended to be used with one of the existing modules, the type should be "size","niche", or "cooc". If the user is creating an entirely new null model, type should be set to NULL (the default value).
`suppressProg`	TRUE or FALSE. If true, display of the progress bar in the console is suppressed; default is FALSE. This setting is useful for creating markdown documents with 'knitr'.

Examples

```
## Not run:
# User defined function



## End(Not run)
```

pianka *Pianka Niche Overlap Metric*

Description

Takes a resource utilization matrix as input and returns the average pairwise Pianka's niche overlap index.

Usage

```
pianka(m = matrix(rpois(80, 1), nrow = 10))
```

Arguments

m	a matrix of resource utilization values.

Details

Pianka's niche overlap index is averaged over each unique species pair. The index is symmetric, with a normalization term in the denominator for the overlap between species 1 and 2. Values of Pianka's niche overlap index close to 0.0 reflect usage of exclusive resource categories, whereas values close to 1.0 reflect similar resource utilization spectra.

$$O_{jk} = O_{kj} = \frac{\sum_n^i p_{ij} p_{jk}}{\sqrt{\sum_n^i p_{ij}^2 \sum_n^i p_{ik}^2}}$$

Value

Returns the average pairwise niche overlap.

Note

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

References

Pianka, E. 1973. The structure of lizard communities. Annual Review of Ecology and Systematics 4:53-74.

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

See Also

czekanowski niche overlap index.

Examples

```
obsOverlap <- pianka(m=matrix(rpois(40,0.5),nrow=8))
```

---

pianka_skew *PiankaSkew Niche Overlap Metric*

Description

Takes a niche utilization matrix returns the skewness of the Pianka pairwise niche overlap index.

Usage

```
pianka_skew(m = matrix(rpois(80, 1), nrow = 10))
```

Arguments

m	a matrix of resource utilization values.

Details

A large positive value for skewness implies that there are more species pairs with high than low niche overlap. A large negative value for skewness implies there are more species pairs with low than high niche overlap. The performance of this algorithm has not been thoroughly tested with real data sets.

Value

Returns the skewness of the average pairwise niche overlap.

Note

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

References

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

See Also

pianka niche overlap index.

Examples

```
obsSkew<- pianka_skew(m=matrix(rpois(40,0.5),nrow=8))
```

pianka_var *PiankaVariance Niche Overlap Metric*

Description

Takes a niche utilization matrix as in put and returns the variance of Pianka's niche overlap index.

Usage

```
pianka_var(m = matrix(rpois(80, 1), nrow = 10))
```

Arguments

m	a matrix of resource utilization values.

Details

A large value for variance implies that some species pairs show high niche overlap and others show low niche overlap. A low value for variance implies that niche overlap (high or low) is very similar among all species pairs.

Value

Returns the variance of the average pairwise niche overlap.

Note

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

References

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

See Also

pianka niche overlap index.

Examples

```
obsVar <- pianka_var(m=matrix(rpois(40,0.5),nrow=8))
```

`plot.coocnullmod` *Co-Occurrence Model Plot Function*

Description

Plot co-occurrence null model object.

Usage

```
## S3 method for class 'coocnullmod'
plot(x, type = "hist", ...)
```

Arguments

`x`	the null model object to plot.
`type`	the type of null plot to make. See details for more information.
`...`	Other variables to be passed on to base plotting.

Details

the valid types for the Co-occurrence module are "hist" to display a histogram of the simulated metric values, "cooc" to display the observed data matrix and one simulated matrix, and (for sim9 only), "burn_in" to display a trace of the metric values during the burn-in period.

The "hist" plot type is common to all EcoSimR modules. The blue histogram represents the NRep values of the metric for the simulated assemblages. The red vertical line represents the metric value for the real assemblage. The two pairs of vertical dashed black lines represent the one-tailed (long dash) and two-tailed (short dash) 95

The "cooc" plot type illustrates the binary presence-absence data (observed = red, simulated = blue). Each row in the grid is a species, each column is a site, and the entries represent the presence (color-filled) or absence (empty) of a species in a site. The rows and columns are illustrated with the same ordering as the original data matrix.

The "burn_in" plot type illustrates the trace values of the metric generated for sim9 during the burn-in period. The x axis is the replicate number and the y axis is the value of the metric. The metric for the original data matrix is illustrated as a horizontal red line. Consecutive simulated metric values are illustrated with a blue line, and the gray line is a simple loess fit to the simulated values. If the burn_in period is sufficiently long, the trace should be stable, indicating that a stationary distribution has probably been reached.

plot.nichenullmod *Niche Null Model Plot function*

Description

Plot niche overlap null model object.

Usage

```
## S3 method for class 'nichenullmod'
plot(x, type = "hist", ...)
```

Arguments

x	the null model object to plot.
type	the type of null model plot to display. See details for more information.
...	Other variables to be passed on to base plotting.

Details

the valid types for the Niche Overlap module are "hist" to display a histogram of the simulated metric values, and "niche" to display the observed data matrix and one simulated matrix.

The "hist" plot type is common to all EcoSimR modules. The blue histogram represents the NRep values of the metric for the simulated assemblages. The red vertical line represents the metric value for the real assemblage. The two pairs of vertical dashed black lines represent the one-tailed (long dash) and two-tailed (short dash) 95

The "niche" plot type illustrates the utilization data (observed = red, simulated = blue). Each row in the figure is a species, and each column is a resource utilization category. The area of each circle depicted is proportional to the utilization of a resoruce category by a species. Empty positions indicate a resource utilization of 0.0. The rows and columns are illustrated with the same ordering as the original data matrix.

plot.nullmod *plot a histogram null model*

Description

Plot a null model object.

Usage

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

Arguments

x	the null model object to plot.
...	Other variables to be passed on to base plotting.

Details

The "hist" plot type is common to all EcoSimR modules. The blue histogram represents the NRep values of the metric for the simulated assemblages. The red vertical line represents the metric value for the real assemblage. The two pairs of vertical dashed black lines represent the one-tailed (long dash) and two-tailed (short dash) 95

---

plot.sizenullmod *Size Ratio Plot Function*

Description

Plot Size Ratio null model object.

Usage

```
## S3 method for class 'sizenullmod'
plot(x, type = "hist", ...)
```

Arguments

x	the null model object to plot.
type	the type of null model plot to display. See details for more information.
...	Other variables to be passed on to base plotting.

Details

the valid types for the Size Overlap module are "hist" to display a histogram of the simulated metric values, and "size" to display the observed data matrix and one simulated matrix.

The "hist" plot type is common to all EcoSimR modules. The blue histogram represents the NRep values of the metric for the simulated assemblages. The red vertical line represents the metric value for the real assemblage. The two pairs of vertical dashed black lines represent the one-tailed (long dash) and two-tailed (short dash) 95

The "size" plot type illustrates the trait data (observed = red, simulated = blue). Each circle in the number line represents the trait value of a different species. For the observed and simulated data, a histogram of the ordered size differences is also illustrated.

ra1 *RA1 Niche Overlap Randomization Algorithm*

Description

Randomizes a numeric utilization matrix speciesData by replacing all elements with a random uniform [0,1] value.

Usage

```
ra1(speciesData = matrix(rpois(80, 1), nrow = 10))
```

Arguments

speciesData	a resource utilization matrix (rows = species, columns = discrete resource states) filled with non-negative real numbers.

Details

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

Value

Returns a random utilization matrix with the same dimensions as the input matrix.

Note

Because all matrix elements, including zeroes, are replaced with a random uniform distribution, the null expectation is based on an assemblage of generalist species with maximum niche breadth. This algorithm retains neither the niche breadth of the individuals species nor the placement of 0 values (= unutilized resource states) in the matrix. These assumptions are unrealistic, and a random matrix with zeroes will generate significantly low niche overlap values with this metric. It is not recommended for niche overlap analysis.

References

Kobayashi, S. 1991. Interspecific relations in forest floor coleopteran assemblages: niche overlap and guild structure. Researches in Population Ecology 33: 345-360.

Examples

```
ranUtil <- ra1(speciesData=matrix(rpois(40,0.5),nrow=8))
```

ra2 *RA2 Niche Overlap Randomization Algorithm*

Description

Randomizes a numeric utilization matrix speciesData by replacing all non-zero elements with a random uniform [0,1] value.

Usage

```
ra2(speciesData = matrix(rpois(80, 1), nrow = 10))
```

Arguments

speciesData	a resource utilization matrix (rows = species, columns = discrete resource states) filled with non-negative real numbers.

Details

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

Value

Returns a random utilization matrix with the same dimensions as the input matrix.

Note

This algorithm retains the number and position of zero states in the original matrix. However, all non-zero values are again replaced by a random [0,1] value, which tends to inflate niche breadths of the simulated assemblage. Although the results are not as severe as for RA1, this algorithm is still prone to Type I errors, and is not recommended for niche overlap analysis.

References

Kobayashi, S. 1991. Interspecific relations in forest floor coleopteran assemblages: niche overlap and guild structure. Researches in Population Ecology 33: 345-360.

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

Examples

```
ranUtil <- ra2(speciesData=matrix(rpois(40,0.5),nrow=8))
```

ra3 *RA3 Niche Overlap Randomization Algorithm*

Description

Randomizes a numeric utilization matrix speciesData by reshuffling the elements within each row.

Usage

```
ra3(speciesData = matrix(rpois(80, 1), nrow = 10))
```

Arguments

speciesData	a resource utilization matrix (rows = species, columns = discrete resource states) filled with non-negative real numbers.

Details

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

Value

Returns a random utilization matrix with the same dimensions as the input matrix.

Note

This algorithm retains the niche breadth and zero states for each species, but randomizes the assignment of each utilization value to a different niche category. It performs effectively in simulation studies and is recommended for analysis of niche overlap patterns.

References

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

Examples

```
ranUtil <- ra3(speciesData=matrix(rpois(40,0.5),nrow=8))
```

ra4 *RA4 Niche Overlap Randomization Algorithm*

Description

Randomizes a numeric utilization matrix speciesData by reshuffling the non-zero elements within each row.

Usage

```
ra4(speciesData = matrix(rpois(80, 1), nrow = 10))
```

Arguments

speciesData	a resource utilization matrix (rows = species, columns = discrete resource states) filled with non-negative real numbers.

Details

The resource utilization matrix (rows = species, columns = discrete resource categories) may include zeroes, but no negative numbers or missing values. Relative resource within a species is first calculated, so the rows need not sum to 1.0.

Value

Returns a random utilization matrix with the same dimensions as the input matrix.

Note

This algorithm is similar to RA3, but adds the additional constraint of retaining the positions of all of the zero elements of the matrix, and reshuffling only the non-zero elements of the matrix within each row. It is more conservative than RA3, but has a low Type I error rate, and, along with RA3, is recommended for null model analysis of niche overlap.

References

Winemiller, K.O. and E.R. Pianka. 1990. Organization in natural assemblages of desert lizards and tropical fishes. Ecological Monographs 60: 27-55.

Examples

```
ranUtil <- ra4(speciesData=matrix(rpois(40,0.5),nrow=8))
```

ranMatGen *Random Matrix Generator*

Description

Create a random matrix with specified dimensions, percentage fill, marginal distributions, and matrix type (abundance or binary) to test the behavior of null model randomization algorithms.

Usage

```
ranMatGen(aBetaRow = 1, bBetaRow = 1, aBetaCol = 1, bBetaCol = 1,
  numRows = 20, numCols = 5, mFill = 0.5, abun = 0, emptyRow = FALSE,
  emptyCol = FALSE)
```

Arguments

aBetaRow	parameter shape1 for beta of row marginals
bBetaRow	parameter shape2 for beta of row marginals
aBetaCol	parameter shape1 for beta of column marginals
bBetaCol	parameter shape2 for beta of column marginals
numRows	number of rows in random matrix
numCols	number of columns in random matrix
mFill	proportion of matrix cells to be filled with non-zero elements
abun	expected total abundance (from Poisson distribution) of individuals in the matrix. If set to the default value of 0, abundances are not shown but are replaced by occurrences (1s)
emptyRow	specifies whether empty rows in random matrix will be retained (TRUE) or stripped out (FALSE)
emptyCol	specified wither empty columns will be retained (TRUE) or stripped out (FALSE)

Details

Understanding the behavior of null models with artificial data is an essential step before applying them to real data. This function creates stochastic community matrices in which each row is a species, each column is a site or island, and each entry is the occurrence (presence-absence) or abundance of a species in a site. For the analysis of niche overlap, the sites can be treated as unordered niche categories, and the abundances are the utilization values for each species.

Row and column marginal distributions are described from a beta distribution, with user supplied coefficients. Marginal distributions are rescaled to one, and the conjoint probability of a species occurring in a site is determined with the outer product of the species (=row) and site(= column) marginals. This simple calculation assumes sites and species are independent, and excludes site x species interactions as well as species x species interactions.

The user specifies the percent fill of the matrix, and this number of cells are randomly selected without replacement using the cojoint probabilities calculated from each marginal distribution. If

the user has requested a presence-absence matrix ('abun = 0'), these cells are assigned a value of 1. If the user has reqested an abundance matrix ('abun > 0'), then the value of abundance specifies the summed abundance of all individuals in the matrix. The value of 'abun' is used to set the lambda parameter for each occupied cell, and then a single draw from a Poisson distribution is used for the abundance in that cell. Small conjoint marginal probabilities can lead to empty rows or columns and the user can specify whether or not to retain empty rows and columns. The matrix rows and columns are sorted in descending order according to the marginal frequencies, and these are returned (matrix$rowMarg and matrix$colMarg) along with the matrix (matrix$m) in list form.

'aBetaRow', 'bBetaRow', 'aBetaCol', and 'bBetaCol' specify the two shape parameters for the row and column marginals. The marginal values are created by a single random draw from these beta distributions, and then are rescaled so they sum to 1.0. Thus, the mean parameter value specified by the beta distribution does not matter in the calculation. Instead, it is the size of the variance that determines the amount of heterogeneity among row or column margins. Small values for the two shape parameters generate greater heterogeneity among the rows or columns marginals of the matrix.

Thus, a distribution with 'aBetaRow=1000' and 'bBetaRow=1000' will generate marginal probabilities that are virtually identical for the different species (=rows), whereas 'aBetaRow=1' and 'bBetaRow=1' will generate uniform probabilities. These default values applied to both rows and columns will generate a typical presence-absence matrix, with some common and some sparse species, and some species-rich and species-poor sites.

Setting 'numRows', 'numCols', and 'mFill' allow the test matrix to be tailored to match the observed matrix. However, it may be necessary to increase 'numRows' and 'nCols' if the parameters often generate empty rows or columns.

If low values of 'abun' are specified, some occupied cells may be set to 0 because of a random draw from the Poisson distribution for that matrix cell.

Once the test matrix is created, it can be used to explore any of the combinations of algorithm and matrix that are available in EcoSimR.

Examples

```
## Not run:
## Create a null matrix similar to MacArthur's warblers
testMatrix <- ranMatGen(aBetaCol=1000,bBetaCol=1000,
                        aBetaRow=1,bBetaRow=1,
                        numRows=5,numCols=16,
                        mFill=0.75, abun=1000,
                        emptyRow=FALSE,emptyCol=TRUE)$m

## Run the null model
testMod <- niche_null_model(testMatrix)

## Summary and niche utilization plot
summary(testAnalysis)
plot(testMod,type="niche")


## Create a null matrix similar to West Indies Finches
testMatrix <- ranMatGen(aBetaCol=0.5,bBetaCol=0.5,
                        aBetaRow=0.5,bBetaRow=0.5,
```

```
                        numRows=30,numCols=30,
                        mFill=0.25,abun=0,emptyRow=FALSE,
                        emptyCol=FALSE)$m

## Run the null model
testMod <- cooc_null_model(testMatrix$m,
                           algo="simFast",
                           burnin=10000,n.reps=1000)

## Summary and matrix, burn-in plots
summary(testMod)
plot(testMod,type="cooc")
plot(testMod,type="burnin")

## End(Not run)
```

`reproduce_model` *Reproduce a result*

Description

Helps reproduce the result of a simulation by restoring the RNG to the state of a supplied null model object.

Usage

```
reproduce_model(model)
```

Arguments

`model`	the model object containing the result to be reproduced.

Details

Works by resetting the RNG state to what it was for a given EcoSimR simulation. This only works if the user saved the seed with the saveSeed parameter.

Examples

```
## Not run:
finchMod <- cooc_null_model(dataWiFinches, algo="sim1",saveSeed=T)
## Check model output
mean(finchMod$Sim)

reproduce_model(finchMod)

finchMod <- cooc_null_model(dataWiFinches, algo="sim1")
## Check model output is the same as before
mean(finchMod$Sim)
reproduce_model(finchMod)
```

```
## End(Not run)
```

---

sim1 *Sim1 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling all of its elements equiprobably.

Usage

```
sim1(speciesData)
```

Arguments

speciesData	binary presence-absence matrix (rows = species, columns = sites).

Details

This algorithm assumes species and sites are equiprobable. It preserves the total matrix fill, but places no other constraints on row or column totals.

Value

Returns a binary presence-absence matrix with the same dimensions and percent fill as the input matrix.

Note

This is the simplest of all randomization algorithms for a presence- absence matrix. However, it assumes that both species and sites are equiprobable, and has poor Type I error frequencies when tested with purely random matrices. If the input matrix is sparse, it will often generate null matrices with empty rows or columns. Not recommended for co-occurrence analysis.

References

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. Ecology 81: 2606-2621.

Examples

```
randomMatrix <- sim1(speciesData=matrix(rbinom(40,1,0.5),nrow=8))
```

sim10 *Sim10 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling all elements. Rows and column probabilities are proportional to user-supplied row and column weights, which define relative suitability probabilities for species and sites. Makes a call to the vector_sample function.

Usage

```
sim10(speciesData, rowWeights = runif(dim(speciesData)[1]),
  colWeights = runif(dim(speciesData)[2]))
```

Arguments

speciesData	binary presence-absence matrix (rows = species, columns = sites).
rowWeights	vector of positive values representing species weights.
colWeights	vector of positive values representing site weights.

Details

This function incorporates vectors of weights for species and/or sites to condition the simulation. These two vectors are used as outer products to set cell probabilities for the entire matrix. Thus:

$$p(cell_{ij}) = p(row_i)p(col_j)$$

Weights must be positive real numbers. The algorithm will scale them so they sum to 1.0, so they can be used in their natural units (e.g. island area, species abudance), and will be scaled properly. If all species (or sites) are assumed to be equally likely, the weight vector should be set to the same constant for all elements.

Value

Returns a binary presence-absence matrix with the same dimensions and fill as the input matrix.

Note

sim10 allows users to incorporate independent data on species occurrence probabilities and site suitabilities. This represents an important conceptual advance over standard co-occurrence analyses, which must infer these probabilities from the matrix itself. sim10 may generate empty rows or columns, especially if weights are very small for some species or sites. Also, the results may be sensitive to algebraic transformations of the weights (x, x^2, log(x), etc.), and these transformations may be hard to justify biologically. Nevertheless, sim10 is worth exploring for rich data sets with site and species attributes.

References

Jenkins, D.G. 2006. In search of quorum effects in metacommunity structure: species co-occurrence analyses. Ecology 87:1523-1531

Gotelli, N.J., G.R. Graves, and C. Rahbek. 2010. Macroecological signals of species interactions in the Danish avifauna. Proceedings of the National Academy of Sciences, U.S.A. 107: 530-535.

See Also

vector_sample for weighted vector sampling.

Examples

```
randomMatrix <- sim10(speciesData=matrix(rbinom(40,1,0.5),nrow=8))
```

---

sim2 *Sim2 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling elements within each row equiprobably.

Usage

```
sim2(speciesData)
```

Arguments

speciesData	binary presence-absence matrix (rows = species, columns = sites).

Details

This algorithm assumes sites are equiprobable, but preserves differences among species (= row sums).

Value

Returns a binary presence-absence matrix with the same dimensions and rowsums as the input matrix.

Note

This algorithm preserves differences in the commonness and rarity of species (= rowsums), but assumes that all sites are equiprobable. It would not be appropriate for islands that vary greatly in area, but it would be appropriate for quadrat censuses in a relatively homogeneous environment. sim2 can sometimes generate matrices with empty columns, but this is unlikely unless the matrix is very sparse. sim2 has good Type I error frequenceis when tested against random matrices. However, if sites do vary in their suitability or habitat quality, it will often identify aggregated patterns of

species co-occurrence. sim2 and sim9 have the best overall performance for species co-occurrence analyses. However, because they differ in their assumptions about site quality, they often differ in their results, with sim9 often detecting random or segregated patterns for matrices in which sim2 detects aggregated patterns.

References

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. ecology 81: 2606-2621.

See Also

sim9 co-occurrence algorithm.

Examples

```
randomMatrix <- sim2(speciesData=matrix(rbinom(40,1,0.5),nrow=8))
```

---

sim3 *Sim3 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling elements within each column equiprobably.

Usage

```
sim3(speciesData)
```

Arguments

speciesData	binary presence-absence matrix (rows = species, columns = sites).

Details

This algorithm assumes species are equiprobable, but preserves differences among sites (= column sums).

Value

Returns a binary presence-absence matrix with the same dimensions and colsums as the input matrix.

Note

This algorithm preserves differences in species richness among sites (= colsums), but assumes that all species are equiprobable. This assumption is usually unrealistic, and sim3 has a high frequency of Type I errors with random matrices, so it is not recommended for co-occurrence analysis.

References

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. Ecology 81: 2606-2621.

Examples

```
randomMatrix <- sim3(speciesData=matrix(rbinom(40,1,0.5),nrow=8))
```

`sim4` *Sim4 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling elements within each row. Sampling weights for each column are proportional to column sums. Makes a call to the vector_sample function.

Usage

```
sim4(speciesData)
```

Arguments

`speciesData`	binary presence-absence matrix (rows = species, columns = sites).

Details

This algorithm preserves differences among species in occurrence frequencies, but assumes differences among sites in suitability are proportional to observed species richness (= column sums).

Value

Returns a binary presence-absence matrix with the same dimensions and rowsums as the input matrix.

Note

This algorithm preserves differences in the commonness and rarity of species (= rowsums), but assumes differences among sites in suitability are proportional to observed species richness (= column sums). sim4 has a somewhat high frequency of Type I errors with random matrices, so it is not recommended for co-occurrence analysis.

References

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. Ecology 81: 2606-2621.

Examples

```
randomMatrix <- sim4(speciesData = matrix(rbinom(40,1,0.5),nrow=8))
```

sim5 *Sim5 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling elements within each column. Sampling weights for each row are proportional to row sums. Makes a call to the vector_sample function.

Usage

```
sim5(speciesData)
```

Arguments

speciesData	binary presence-absence matrix (rows = species, columns = sites).

Details

This algorithm preserves differences among sites in species richness, but assumes differences among species in commonness and rarity are proportional to observed species occurrences (= row sums).

Value

Returns a binary presence-absence matrix with the same dimensions and colsums as the input matrix.

Note

This algorithm preserves differences among sites in species richness (= colsums), but assumes differences among species in commonness and rarity are proportional to observed species occurrences (= rowsums). sim5 has a high frequency of Type I errors with random matrices, so it is not recommended for co-occurrence analysis.

References

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. Ecology 81: 2606-2621.

Examples

```
randomMatrix <- sim5(speciesData = matrix(rbinom(40,1,0.5),nrow=8))
```

`sim6` *Sim6 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling all elements. Rows are equiprobable, and columns are proportional to column sums. Makes a call to the vector_sample function.

Usage

```
sim6(speciesData)
```

Arguments

`speciesData`	binary presence-absence matrix (rows = species, columns = sites).

Details

This algorithm assumes that species are equiprobable, but that differences in suitability among sites are proportional to observed species richness (=colsums).

Value

Returns a binary presence-absence matrix with the same dimensions and fill as the input matrix.

Note

This algorithm assumes that species are equiprobable, and that differences among sites are proportional to observed species richness (=colsums). sim6 has a high frequency of Type I errors with random matrices, so it is not recommended for co-occurrence analysis.

References

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. Ecology 81: 2606-2621.

Examples

```
randomMatrix <- sim6(speciesData = matrix(rbinom(40,1,0.5),nrow=8))
```

sim7 *Sim7 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling all elements. Columns are equiprobable, and rows are proportional to row sums. Makes a call to the vector_sample function.

Usage

```
sim7(speciesData)
```

Arguments

speciesData	binary presence-absence matrix (rows = species, columns = sites).

Details

This algorithm assumes that sites are equiprobable, but that differences in frequency of occurrence among species are proportional to observed species richness (=colsums).

Value

Returns a binary presence-absence matrix with the same dimensions and fill as the input matrix.

Note

This algorithm assumes that species are equiprobable, and that differences among sites are proportional to observed species richness (=colsums). sim7 has a high frequency of Type I errors with random matrices, so it is not recommended for co-occurrence analysis.

References

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. Ecology 81: 2606-2621.

Examples

```
randomMatrix <- sim7(speciesData = matrix(rbinom(40,1,0.5),nrow=8))
```

sim8 *Sim8 Co-occurrence Randomization Algorithm*

Description

Randomizes a binary matrix speciesData by reshuffling all elements. Columns are proportional to column sums, and rows are proportional to row sums. Makes a call to the vector_sample function.

Usage

```
sim8(speciesData)
```

Arguments

speciesData	binary presence-absence matrix (rows = species, columns = sites).

Details

This algorithm assumes that the probability that a species occurs in a site is depends on the joint independent probability of randomly selecting the species and randomly selecting the site, with these probabilities set proportional to row and column sums of the matrix.

Value

Returns a binary presence-absence matrix with the same dimensions and fill as the input matrix.

Note

This algorithm is theoretically attractive because it incorporates heterogeneity in species occurrences and species richness per site in a probabilistic way that does not fix row and column frquencies. However, in spite of its appeal, sim8 does not generate average row and column sums that match the original matrix, and it is susceptible to Type I errors when tested with random matrices. It is not recommended for co-occurrence analysis. See Ulrich and Gotelli (2012) for a more complicated algorithm for probabilistic row and column totals that has better statistical behavior.

References

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. Ecology 81: 2606-2621.

Ulrich, W. and N.J. Gotelli. 2012. A null model algorithm for presence- absence matrices based on proportional resampling. Ecological Modelling 244:20-27.

Examples

```
randomMatrix <- sim8(speciesData = matrix(rbinom(40,1,0.5),nrow=8))
```

sim9 *Sim9 Co-occurrence Randomization Algorithm*

Description

An improved implementation of the sequential swap algorithm.

Usage

```
sim9(speciesData, algo, metric, nReps = 1000, saveSeed = FALSE,
  burn_in = 0, algoOpts = list(), metricOpts = list(),
  suppressProg = TRUE)
```

Arguments

speciesData	a dataframe in which rows are species, columns are sites, and the entries indicate the absence (0) or presence (1) of a species in a site. Empty rows and empty columns should not be included in the matrix.
algo	the algorithm to use, must be "sim1", "sim2", "sim3", "sim4", "sim5", "sim6", "sim7", "sim8", "sim9", "sim10"; default is "sim9".
metric	the metric used to calculate the null model: choices are "species_combo", "checker", "c_score", "c_score_var", "c_score_skew", "v_ratio"; default is "c_score".
nReps	the number of replicate null assemblages to create; default is 1000 replicates.
saveSeed	TRUE or FALSE. If TRUE the current seed is saved so the simulation can be repeated; default is FALSE.
burn_in	The number of burn_in iterations to use with the simFast algorithm; default is 500 burn-in replicates.
algoOpts	a list containing all the options for the specific algorithm you want to use. Must match the algorithm given in the ‘algo‘ argument.
metricOpts	a list containing all the options for the specific metric you want to use. Must match the metric given in the ‘metric‘ argument.
suppressProg	TRUE or FALSE. If true, display of the progress bar in the console is suppressed; default is FALSE. This setting is useful for creating markdown documents with ‘knitr‘.

Details

Generating a set of random matrices with fixed row and column sums is a challenging computational problem. In the ecological literature, these matrices have been created by an MCMC "sequential swap" algorithm (Gotelli 2000). Two rows are two columns are chosen randomly ,and if the 4 cells form a 01/10 pattern, the cell values can be swapped to 10/01 and then replaced in the matrix. This generates a slightly different matrix with the same row and column totals. If the cells cannot be swapped, the trial is discarded. Because only 4 cells are reshuffled, it takes many successive swaps to eliminate transient effects as the matrix moves away from the original configuration and

approaches a stationary distribution. A second disadvantage of the sequential swap is that all matrices are not sampled equiprobably because the failed swaps are discarded. This bias seems small for binary matrices that are typically generated by ecological studies (< 100 x 100), but could be important for "big data" applications.

EcoSimR uses an unbiased and more efficient algorithm, which Strona et al. (2014) have recently dubbed the "curveball algorithm". In this algorithm, two rows from the matrix are randomly chosen to create a submatrix. Within the submatrix, columns in which the column sums are equal to zero are randomly swapped. The resulting submatrix is then returned to the full matrix, with modified values in two of the rows. If no swapping is possible (which is an improbable event for most ecological matrices), the unswapped matrix is still retained. The curveball algorithm is much more efficient than the sequential swap because most iterations reshuffle many elements in the matrix simultaneously. Strona et al. (2014) show empirically that this algorithm gives unbiased results. However, the resulting MCMC chains will still exhibit autocorrelation for consecutive matrices, especially if the matrix is very large. Future versions of EcoSimR will allow for a thinning parameter to avoid using every sequential matrix from the MCMC chain. The current version of EcoSimR allows for control over the burn-in period and generates a burn-in plot so the user can see whether stationarity has been achieved.

References

Chen, Y., P. Diaconis, S.P. Holmes, and J.S. Liu. 2005. Sequential Monte Carlo methods for statistical analysis of tables. JASA 100: 109-120.

Cobb, G. W., and Chen, Y.-P. 2003. An Application of Markov Chain Monte Carlo to Community Ecology. American Mathematical Monthly 110: 265-288.

Gotelli, N.J. 2000. Null model analysis of species co-occurrence patterns. Ecology 81: 2606-2621.

Strona. G., D. Nappo, F. Boccacci, S. Fattorini, and J. San-Miguel-Ayanz. 2014. A fast and unbiased procedure to randomize ecological binary matrices with fixed row and column totals. Nature Communications 5:4114 | DOI: 10.1038/ncomms5114.

Examples

```
## Not run:

## Run the null model
finchMod <- cooc_null_model(dataWiFinches, algo="sim1",nReps=1000000,burn_in = 500)
## Summary and plot info
summary(finchMod)
plot(finchMod,type="burn_in")
plot(finchMod,type="hist")
plot(finchMod,type="cooc")

## End(Not run)
```

sim9_single *sim9_single*

Description

Function for a single iteration of the sequential swap.

Usage

```
sim9_single(speciesData = matrix(rbinom(100, 1, 0.5), nrow = 10))
```

Arguments

speciesData	binary presence-absence matrix.

Details

See details for sim9.

size_gamma *SizeGamma Size Overlap Randomization Algorithm*

Description

Function to generate a random distribution of body sizes by drawing from a gamma distribution. Shape and rate parameters of the gamma are estimated from the empirical data.

Usage

```
size_gamma(speciesData = rnorm(50, mean = 100, sd = 1))
```

Arguments

speciesData	a vector of body sizes or other trait measurements of species. All values must be positive real numbers.

Value

Returns a vector of simulated body sizes as the same length as speciesData.

Note

The shape and rate parameters are estimated from the real data using the maximum likelihood estimators generated from the fitdr function in the MASS library. The flexible gamma distribution can be fit to a variety of normal, log-normal, and exponential distributions that are typical for trait data measured on a continuous non-negative scale.

See Also

fitdistr in the MASS library.

Examples

```
obsOverlap <- size_gamma(speciesData=rnorm(50,mean=100,sd=1))
```

`size_null_model` *Size Ratio*

Description

Create a Size Ratio null model

Usage

```
size_null_model(speciesData, algo = "size_uniform", metric = "var_ratio",
  nReps = 1000, saveSeed = FALSE, algoOpts = list(),
  metricOpts = list(), suppressProg = FALSE)
```

Arguments

`speciesData`	a data vector in which each entry is the measured trait (such as body size or flowering date) for each species in the assemblage.
`algo`	the algorithm to use, must be "size_uniform", "size_uniform_user", "size_source_pool", "size_gamma".
`metric`	the metric used to caluclate the null model: choices are "min_diff", "min_ratio", "var_diff", "var_ratio"; default is "var_ratio".
`nReps`	the number of replicate null assemblages to create; default is 1000 replicates.
`saveSeed`	TRUE or FALSE. If TRUE the current seed is saved so the simulation can be repeated; default is FALSE.
`algoOpts`	a list containing all the options for the specific algorithm you want to use. Must match the algorithm given in the ‘algo‘ argument.
`metricOpts`	a list containing all the options for the specific metric you want to use. Must match the metric given in the ‘metric‘ argument.
`suppressProg`	TRUE or FALSE. If true, display of the progress bar in the console is suppressed; default is FALSE. This setting is useful for creating markdown documents with ‘knitr‘.

Examples

```
## Not run:
## Run the null model
rodentMod <- size_null_model(dataRodents)
## Summary and plot info
summary(rodentMod)
plot(rodentMod,type="hist")
plot(rodentMod,type="size")

##  Uniform Size model with user inputs
rodentMod2 <- size_null_model(dataRodents,algo="size_uniform_user",
algoOpts = list(userLow = 3,userHigh=15))
summary(rodentMod2)
plot(rodentMod2,type="hist")
plot(rodentMod2,type="size")

### Source pool model

rodentMod_sp <- size_null_model(dataRodents,algo="size_source_pool",
algoOpts = list(sourcePool = runif(dim(dataRodents)[1],1,15)))

summary(rodentMod_sp)
plot(rodentMod_sp,type="hist")
plot(rodentMod_sp,type="size")


## End(Not run)
```

`size_source_pool` *SizeSourcePoolDraw Size Overlap Randomization Algorithm*

Description

Function to randomize body sizes by drawing species from a user-defined source pool. Species are drawn without replacement, and there is a specified probability vector for the source pool species

Usage

```
size_source_pool(speciesData = 21:30, sourcePool = runif(n = 2 *
  length(speciesData), min = 10, max = 50), speciesProbs = rep(1,
  length(sourcePool)))
```

Arguments

`speciesData`	a vector of observed body sizes.
`sourcePool`	a vector of body sizes of species in the user-defined pool of potential colonists.
`speciesProbs`	a vector of relative colonization weights of length 'sourcePool'.

Value

Returns a vector of body sizes of an assemblage randomly drawn from a user-defined source pool.

Note

Although delineating a source pool of species and estimating their relative colonization probabilities is difficult, this is the most realistic approach to constructing a null distribution.

References

Strong, D.R. Jr., L.A. Szyska, and D. Simberloff. 1979. Tests of community-wide character displacement against null hypotheses. Evolution 33: 897-913. Schluter, D. and P.R. Grant. 1984. Determinants of morphological patterns in communities of Darwin's finches. American Naturalist 123: 175-196.

Examples

```
obsOverlap <- size_source_pool(dataRodents$Sonoran)
```

---

size_uniform *SizeUniform Size Overlap Randomization Algorithm*

Description

Function to randomize body sizes within a uniform distribution with boundaries set by the largest and smallest species in the assemblage.

Usage

```
size_uniform(speciesData = runif(20))
```

Arguments

speciesData	a vector of positive real values representing the body sizes or trait values for each species.

Details

If the assemblage contains n species, only the body sizes of the inner n - 2 species are randomized.

Value

Returns a vector of body sizes that have been randomly assigned. The largest and smallest body sizes in the randomized assemblage match those in the empirical data.

Note

Although the distribution of body sizes may not be truly uniform, it may be approximately uniform within the range of observed values, particularly for small assemblages.

References

Simberloff, D. and W. Boecklen. 1981. Santa Rosalia reconsidered: size ratios and competition. Evolution 35: 1206-1228.

Tonkyn, D.W. and B.J. Cole. 1986. The statistical analysis of size ratios. American Naturalist 128: 66-81.

See Also

size_gamma size distribution function.

Examples

```
nullSizes <-size_uniform(speciesData=runif(20))
```

---

size_uniform_user *SizeUser Size Overlap Randomization Algorithm*

Description

Observed body sizes are randomized with a uniform distribution for which the user has defined the minimum and maximum possible body size.

Usage

```
size_uniform_user(speciesData = runif(n = 20), userLow = 0.9 *
  min(speciesData), userHigh = 1.1 * max(speciesData))
```

Arguments

speciesData	a vector of observed body sizes.
userLow	a user-defined lower limit.
userHigh	a user-defined upper limit.

Details

Within the user-defined limits, body sizes of all n species are randomized, whereas uniform_size randomizes only n - 2 of the body sizes and uses the extreme values to set the endpoints.

Value

Returns a vector of randomized body sizes.

Note

As the difference between the lower and upper boundaries is increased the test will yield results that are random or aggregated, even though the same data might yield a segregated pattern when the uniform_size algorithm is used. For this reason, this algorithm is not recommended for size ratio analyses.

See Also

size_uniform size distribution algorithm.

Examples

```
nullSizes <- size_uniform_user(speciesData=runif(20,min=10,max=20),userLow=8,userHigh=24)
```

`species_combo` *SpeciesCombo Co-occurrence Metric*

Description

Function to calculate number of unique species combinations in a matrix

Usage

```
species_combo(m = matrix(rbinom(100, 1, 0.5), nrow = 10))
```

Arguments

`m`	a binary presence-absence matrix in which rows are species and columns are sites.

Details

In Diamond's (1975) assembly rules model, species interactions lead to certain "forbidden combinations" of species. A set of communities structured this way should contain fewer species combinations than expected by chance.

Value

Returns the number of unique species combinations represented by the different columns (= sites) in the matrix.

Note

This metric is most useful when the number of sites (= columns) is relatively large compared to the number of species (= rows). Empty sites are excluded from the matrix and are not counted as a unique species combination.

References

Diamond, J.M. 1975. Assembly of species communities. p. 342-444 in: Ecology and Evolutoin of Communities. M.L. Cody and J.M. Diamond (eds.). Harvard University Press, Cambridge.

Pielou, D.P. and E.C. Pielou. 1968. Association among species of infrequent occurrence: the insect and spider fauna of Polyporus betulinus (Bulliard) Fries. Journal of Theoretical Biology 21: 202-216.

Examples

```
obsCombo <- species_combo(m=matrix(rbinom(100,1,0.5),nrow=10))
```

`summary.coocnullmod` *Generic function for calculating null model summary statistics.*

Description

Takes as input a list of Null.Model.Out, with Obs, Sim, Elapsed Time, and Time Stamp values.

Usage

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

Arguments

`object`	the null model object to print a summary.
`...`	extra parameters

Details

The summary output includes a timestamp and complete statistics on the simulated values of the metric, including the mean, variance, and one and two-tailed 95

`summary.nichenullmod` *Generic function for calculating null model summary statistics*

Description

Takes as input a list of Null.Model.Out, with Obs, Sim, Elapsed Time, and Time Stamp values

Usage

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

Arguments

`object`	the null model object to print a summary of.
`...`	Extra parameters for summary .

Details

The summary output includes a timestamp and complete statistics on the simulated values of the metric, including the mean, variance, and one and two-tailed 95

summary.nullmod *Generic function for calculating null model summary statistics*

Description

Takes as input a list of Null.Model.Out, with Obs, Sim, Elapsed Time, and Time Stamp values

Usage

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

Arguments

object	the null model object to print a summary of.
...	Extra parameters for summary.

Details

The summary output includes a timestamp and complete statistics on the simulated values of the metric, including the mean, variance, and one and two-tailed 95

summary.sizenullmod *Generic function for calculating null model summary statistics*

Description

Takes as input a list of Null.Model.Out, with Obs, Sim, Elapsed Time, and Time Stamp values

Usage

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

Arguments

object	the null model object to print a summary of.
...	Extra parameters for summary.

Details

The summary output includes a timestamp and complete statistics on the simulated values of the metric, including the mean, variance, and one and two-tailed 95

var_diff *VarDiff Size Overlap Ratio Metric*

Description

Function to calculate the variance in size differences between adjacent, ordered species. If there is a tendency towards a constant absolute size difference between adjacent species, this variance will be relatively small. Alternatively, if some adjacent species are close in size, but others are very distant, this variance will be large. Small variances might be indicative of assemblages in which there is a competitively-based limit to similarity.

Usage

```
var_diff(m = runif(20))
```

Arguments

m	a vector of non-negative trait measures, one for each species

Value

Returns the variance of the absolute difference between adjacent, ordered values.

References

Poole, R.W. and B.J. Rathcke. 1979. Regularity, randomness, and aggregation in flowering phenologies. Science 203:470-471.

Simberloff, D. and W.J. Boecklen. 1981. Santa Rosalia reconsidered: size ratios and competition. Evolution 35: 1206-1228.

Examples

```
SizeDifVar <- var_diff(rgamma(20,shape=3,scale=2))
```

var_ratio *VarRatio Size Overlap Ratio Metric*

Description

Function to calculate the variance in size ratios between adjacent, ordered species. If there is a tendency towards a constant size ratio between adjacent species, this variance will be relatively small. Alternatively, if some adjacent species are close in size, but others are very distant, this variance will be large. Small variances might be indicative of assemblages in which there is a competitively-based limit to similarity.

Usage

```
var_ratio(m = runif(20))
```

Arguments

m	a vector of non-negative trait measures, one for each species

Value

Returns the variance of the size ratios between adjacent, ordered values.

References

Poole, R.W. and B.J. Rathcke. 1979. Regularity, randomness, and aggregation in flowering phenologies. Science 203:470-471.

Simberloff, D. and W.J. Boecklen. 1981. Santa Rosalia reconsidered: size ratios and competition. Evolution 35: 1206-1228.

Examples

```
SizeRatioVar <- var_ratio(rgamma(20,shape=3,scale=2))
```

---

vector_sample *Vector Sample Function*

Description

Takes an input binary vector and a weight vector. Reassigns 1s randomly in proportion to vector weights.

Usage

```
vector_sample(speciesData, weights)
```

Arguments

speciesData	binary vector representing species presences and absences.
weights	a vector of non-negative read numbers representing probabilistic weights for species occurrencs of the same length as speciesData.

Details

This function takes an input vector of binary presence-absence values and a vector of non-negative probability weights. Both vectors must be of identical length.

Value

Returns a re-ordered binary vector in which the occurrences are placed in cells with probabilities proportional to values given in weights.

Note

Several of the randomization algorithms use this function to assign species occurrences with probabilities that reflect species or site differences. It is an effective method for conditioning the marginal probabilities of a null matrix on independent measurements of site or species characteristics.

References

Gotelli, N.J., G.R. Graves, and C. Rahbek. 2010. Macroecological signals of species interactions in the Danish avifauna. Proceedings of the National Academy of Sciences, U.S.A. 107: 530-535.

See Also

`sim10` randomization algorithm.

Examples

```
myColonizer <- vector_sample(speciesData=rbinom(10,1,0.5),weights=runif(10))
```

---

v_ratio *SchlutersVRatio Co-occurrence Metric*

Description

Takes a binary presence-absence matrix or a matrix of abundances and returns Schluter's (1984) variance ratio.

Usage

```
v_ratio(m = matrix(rbinom(100, 1, 0.5), nrow = 10))
```

Arguments

`m`	a binary presence-absence matrix in which rows are species and columns are sites. The entries may be either abundances or occurrences of indivdual species.

Details

The variance ratio is the ratio of the variance in species number among sites to the sum of the variance of the species occurrences. If the average covariation in abundance (or occurrence) of each species pair is close to zero, the expected value for this ratio is approximately 1.0. V-ratios larger than 1.0 imply positive average covariation in the abundance of species pairs, whereas V-ratios significantly smaller than 1.0 imply negative average covariation.

Value

Returns the variance ratio of the matrix.

Note

This index is determined exclusively by the row and column sums of the matrix, so it cannot be used with null model algorithms that hold both of those elements fixed. A simple randomization of the rows of the matrix (see sim2) assumes that all sites are equiprobable, so it may generate large values (= positive covariance) that reflect heterogeneity among sites.

References

Schluter, D. 1984. A variance test for detecting species associations, with some example applications. Ecology 65: 998-1005.

McCulloch, C.E. 1985. Variance tests for species association. Ecology 66: 1676-1681.

Examples

```
varCScore <- v_ratio(m=matrix(rbinom(100,1,0.5),nrow=10))
```

Index

∗Topic **datasets**
dataMacWarb, 11
dataRodents, 12
dataWiFinches, 12
∗Topic **data**
dataMacWarb, 11
dataRodents, 12
dataWiFinches, 12

c_score, 8, *10, 11*
c_score_skew, 9
c_score_var, 10
checker, 3
cooc_null_model, 4
czekanowski, 5, *7, 8*, *17*
czekanowski_skew, 6
czekanowski_var, 7

dataMacWarb, 11
dataRodents, 12
dataWiFinches, 12

EcoSimR, 13
EcoSimR-package *(*EcoSimR*)*, 13

fitdistr, *42*

min_diff, 13
min_ratio, 14

niche_null_model, 14
null_model_engine, 16

pianka, *6*, 17, *18, 19*
pianka_skew, 18
pianka_var, 19
plot.coocnullmod, 20
plot.nichenullmod, 21
plot.nullmod, 21
plot.sizenullmod, 22

ra1, 23
ra2, 24
ra3, 25
ra4, 26
ranMatGen, 27
reproduce_model, 29

sim1, 30
sim10, 31, *51*
sim2, 32
sim3, 33
sim4, 34
sim5, 35
sim6, 36
sim7, 37
sim8, 38
sim9, *33*, 39
sim9_single, 41
size_gamma, 41, *45*
size_null_model, 42
size_source_pool, 43
size_uniform, 44, *46*
size_uniform_user, 45
species_combo, 46
summary.coocnullmod, 47
summary.nichenullmod, 47
summary.nullmod, 48
summary.sizenullmod, 48

v_ratio, 51
var_diff, 49
var_ratio, 49
vector_sample, *32*, 50