Package 'CNull'

March 16, 2017

Type Package

Title Fast Algorithms for Frequency-Preserving Null Models in Ecology

Version 1.0

Date 2017-3-14

Author Constantinos Tsirogiannis [aut, cre], Adrija Kalvisa [aut]

Maintainer Constantinos Tsirogiannis <`tsirogiannis.c@gmail.com`>

Description Efficient computations for null models that require shuffling columns on big matrix data.
This package provides functions for faster computation of diversity measure statistics
when independent random shuffling is applied to the columns of a given matrix.
Given a diversity measure f and a matrix M, the provided functions can generate random samples
(shuffled matrix rows of M), the mean and variance of f, and the p-values of this measure
for two different null models that involve independent random shuffling of the columns of M.
The package supports computations of alpha and beta diversity measures.

License GPL-3

Imports Rcpp (>= 0.12.9), ape, PhyloMeasures, Matrix

LinkingTo Rcpp

LazyLoad yes

SystemRequirements C++11

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-03-16 08:43:52

R topics documented:

`CNull-package` *CNull: Fast Algorithms for Frequency-Preserving Null Models in Ecology*

Description

A package that speeds up statistical analysis requiring shuffling columns on big matrix data. The package provides functions for faster computation of diversity measure statistics when independent random shuffling is applied to the columns of a given matrix. Given a diversity measure f and a matrix M, the provided functions can generate random samples (shuffled matrix rows of M), the mean and variance of f, and the p-values of this measure for two different null models that involve independent random shuffling of the columns of M. The package supports computations of alpha and beta diversity measures.

Details

Package:	CNull
Type:	Package
Version:	1.0
Date:	2017-3-1
License:	GPL-3

The package considers two different null models for shuffling a matrix M; we call the first model the *permutation* model (mentioned as SIM2 by Gotelli, Gotelli 2000). In this model, a matrix is shuffled by permuting the entries of each column in M independently, with uniform probability among all possible permutations. In the second model, the *individual-based* model (Stegen et al. 2013), the entries of each column in M are summed up, and then the column sum is distributed among the entries of the column. Both null models have been proven to be symmetric for every row in M (Tsirogiannis et al.); for any two rows i and j in M and a given alpha-diversity measure f, the values f(i) and f(j) have exactly the same distribution when the elements in M are perturbed according to one of the models described above.

The current package provides functions that implement the two described null models in a very efficient manner, which allows for processing very large matrix data even on standard computers.

Given a matrix M, a measure f, and a number r of desired random repetitions, the package functions can compute the following for each of the described null models:

- A set of r random values for f generated by shuffling M according to one of the null models.
- The mean and variance of f.
- The p-values of f on M.
- A set of random rows generated from M according to one of the supported null models.

The package provides one function for each of the above problems, and for each of the described null models. The package provides functions both for alpha and beta diversity measures.

Author(s)

Constantinos Tsirogiannis and Adrija Kalvisa

Maintainer: Constantinos Tsirogiannis <tsirogiannis.c@gmail.com>

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22:202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Phylogenetic Diversity measure (PD)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (pd.query(args[[1]],mt))}
```

```
# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate the mean and variance of f in the permutation null
# model using 1000 Monte Carlo randomizations
permutation.moments.a(comm,f=my.f,args=arguments,reps=2000)
```

`individual.based.communities.a`

Produces a set of random communities from a given matrix, based on the individual-based null model (Stegen et al. 2013). These communities can be used for alpha diversity computations

---

Description

Given a matrix M and a number of repetitions k, the function produces k random communities based on the individual-based model. This is equivalent to shuffling M according to this model as many as k times , each time outputing only a certain row (e.g. the top one) of the shuffled matrix. An alpha diversity measure f can be applied on the output communities to determine the null distribution of f for a row in M. This distribution is the same for every row of M. This is because the examined null model produces the same distribution for all rows of M; after shuffling M, each row has the same probability to store a specific community as any other in the resulting matrix.

Usage

```
individual.based.communities.a(matrix, reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A matrix which stores communities generated based on the individual-based model. Each row of the matrix corresponds to a different randomized community. The number of columns, and the names of the columns in the output matrix are the same as in the input matrix. The output matrix is *not* a shuffled version of the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22:202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`individual.based.random.values.a`

Examples

```
require(CNull)

#Create a random integer matrix
comm = matrix(sample(1:300),nrow=15,ncol=20)

#Use individual-based model to produce 2000 random communities
individual.based.communities.a(comm,reps=2000)
```

`individual.based.communities.b`
Produces random pairs of communities from a given matrix, based on the individual-based null model (Stegen et al. 2013). These communities can be used for beta diversity computations

---

Description

Given a matrix M and a number of repetitions k, the function produces k pairs of random communities based on the individual-based null model. This is equivalent to shuffling M according to this model as many as k times, each time outputing only two certain rows (e.g. the two top ones) of the shuffled matrix. A beta diversity measure f can be applied on the output communities to determine the null distribution of f for two rows in M. This distribution is the same for every pairs of rows in M. This is because the examined null model produces the same distribution for all pairs of rows in M; after shuffling M, each pair of rows has the same probability to store two specific communities as any other pair in the resulting matrix.

Usage

```
individual.based.communities.b(matrix, reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A matrix which stores pairs of communities generated based on the individual-based model. The output matrix has two times reps rows, corresponding to as many as reps pairs of randomized communities. For every i in 1:reps, rows in the output matrix with indices 2i-1 and 2i designate a valid random pair of communities. The number of columns, and the names of the columns in the output matrix are the same as in the input matrix. The output matrix is *not* a shuffled version of the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22:202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`individual.based.random.values.b`

Examples

```
require(CNull)

#Create a random integer matrix
comm = matrix(sample(1:300),nrow=15,ncol=20)

#Use individual-based model to produce 2000 random pairs of communities
individual.based.communities.b(comm,reps=2000)
```

`individual.based.moments.a` *Given a matrix and an alpha diversity measure f, calculates the mean and variance of f based on the individual-based null model (Stegen et al. 2013)*

Description

Given a matrix M and an alpha diversity measure f, the function computes the mean and variance of f for a row in M, when M is shuffled according to the individual-based model. The returned mean and variance is the same for every row in M. This is because the examined null model produces the same distribution for all rows in M; after shuffling M, each row has the same probability to store a specific community as any other row in the resulting matrix.

Usage

```
individual.based.moments.a(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	An alpha diversity function f. The interface of f should be such that f(matrix,args) returns a numeric vector V where the i-th element of V is equal to the value of f when applied at the i-th row of the given matrix. To fit to this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A list with two real numbers; the mean and the variance of f on the given matrix for the individual-based model.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22: 202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

individual.based.pvalues.a

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Phylogenetic Diversity measure (PD)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (pd.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate the mean and variance of f in the individual-based null
# model using 2000 Monte Carlo randomizations
individual.based.moments.a(comm,f=my.f,args=arguments,reps=2000)
```

individual.based.moments.b

Given a matrix and a beta diversity measure f, calculates the mean and variance of f based on the individual-based null model (Stegen et al. 2013)

Description

Given a matrix M and a beta diversity measure f, the function computes the mean and variance of f between a pair of rows in M, when M is shuffled according to the individual-based model. The returned mean and variance is the same for every pair of rows in M. This is because the examined null model produces the same distribution for all pairs of rows in M; after shuffling M, each pair of rows has the same probability to store two specific communities as any other pair in the resulting matrix.

Usage

```
individual.based.moments.b(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	A beta diversity function f. The interface of f should be such that f(matrix,args) returns a matrix V where the entry stored at the i-th row and j-th column of V is equal to the value of f when applied at the i-th and j-th row of the input matrix. To fit this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A list with two real numbers; the mean and the variance of f on the given matrix for the individual-based model.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22: 202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`individual.based.pvalues.b`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")
```

```
#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Common Branch Length measure (CBL)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (cbl.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate the mean and variance of f in the individual-based null
# model using 2000 Monte Carlo randomizations
individual.based.moments.b(comm,f=my.f,args=arguments,reps=2000)
```

individual.based.pvalues.a

Produces the p-values of an alpha diversity measure f on a given matrix, based on the individual-based null model (Stegen et al. 2013)

Description

Given a matrix M and an alpha diversity measure f, the function calculates the p-values of f based on the individual-based null model.

Usage

```
individual.based.pvalues.a(matrix,f,args,reps=1000)
```

Arguments

matrix	A matrix with integer values. The matrix should not contain any NA values.
f	An alpha diversity function f. The interface of f should be such that f(matrix,args) returns a numeric vector V where the i-th element of V is equal to the value of f when applied at the i-th row of the given matrix. To fit to this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
args	A list with extra arguments needed by f.
reps	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A numeric vector that stores the p-values of f, calculated based on the individual-based null model. The i-th element of the vector stores the p-value for the i-th row of the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22: 202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`individual.based.communities.a`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Phylogenetic Diversity measure (PD)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (pd.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate the p-values of f for the communities in comm
# based on the individual-based model, using 2000 Monte Carlo randomizations.
individual.based.pvalues.a(comm,f=my.f,args=arguments,reps=2000)
```

individual.based.pvalues.b

Produces the p-values of a beta diversity measure f on a given matrix, based on the individual-based null model (Stegen et al. 2013)

Description

Given a matrix M and a beta diversity measure f, the function calculates the p-values of f based on the individual-based null model.

Usage

```
individual.based.pvalues.b(matrix, f, args, observed.vals, reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	A beta diversity function f. The interface of f should be such that f(matrix,args) returns a matrix V where the entry stored at the i-th row and j-th column of V is equal to the value of f when applied at the i-th and j-th row of the input matrix. To fit this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`observed.vals`	A set of pre-calculated values for which we want to compute their p-values.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A numeric vector that stores the p-values of f, calculated based on the individual-based null model. The i-th element of the vector stores the p-value for the i-th element in vector observed.vals.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22: 202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

individual.based.communities.b

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Common Branch Length measure (CBL)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (cbl.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

#Compute the values of f for all pairs of observed communities in M.
#Turn the resulting matrix with the observed diversity values into a vector
obs.v=my.f(comm,arguments)
vals = as.vector(t(obs.v))

# Calculate the p-values of f for the communities in comm
# based on the individual-based model, using 2000 Monte Carlo randomizations.
individual.based.pvalues.b(comm,f=my.f,args=arguments,observed.vals=vals,reps=2000)
```

---

individual.based.random.values.a

Produces a set of random values for an alpha diversity measure f on a given matrix, based on the individual-based null model (Stegen et al. 2013)

Description

Given a matrix M, an alpha diversity measure f and a number of repetitions k, the function produces k random values of f based on the individual-based model. This is equivalent to shuffling M according to this model as many as k times , each time outputing the value of f only for a certain row (e.g. the top one) of the shuffled matrix. The output values can be used to determine the null distribution of f for a row of M. This distribution is the same for every row of M. This is because the examined null model produces the same distribution for all rows of M; after shuffling M, each row has the same probability to store a specific community C as any other in the resulting matrix.

Usage

```
individual.based.random.values.a(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	An alpha diversity function f. The interface of f should be such that f(matrix,args) returns a numeric vector V where the i-th element of V is equal to the value of f when applied at the i-th row of the given matrix. To fit to this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A vector of as many as reps elements. Stores the randomized values of f calculated based on the individual-based null model.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22: 202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`individual.based.moments.a`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Phylogenetic Diversity measure (PD)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (pd.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate 2000 randomized values of f on comm
# based on the individual-based null model.
individual.based.random.values.a(comm,f=my.f,args=arguments,reps=2000)
```

`individual.based.random.values.b`

Produces a set of random values for a beta diversity measure f on a given matrix, based on the individual-based null model (Stegen et al. 2013)

Description

Given a matrix M, a beta diversity measure f and a number of repetitions k, the function produces k random values of f based on the individual-based model. This is equivalent to shuffling M according to this model as many as k times , each time outputing the value of f only for a certain pair of rows (e.g. the two top ones) of the shuffled matrix. The output values can be used to determine the null distribution of f for a pair of rows in M. This distribution is the same for every pairs of rows in M. This is because the examined null model produces the same distribution for all row pairs of M; after shuffling M, each row has the same probability to store a given community as any other in the resulting matrix.

Usage

```
individual.based.random.values.b(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	A beta diversity function f. The interface of f should be such that f(matrix,args) returns a matrix V where the entry stored at the i-th row and j-th column of V is equal to the value of f when applied at the i-th and j-th row of the input matrix. To fit this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A vector of as many as reps elements. Stores the randomized values of f calculated based on the individual-based null model.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Stegen, J. C., Freestone, A. L., Crist, T. O., Anderson, M. J., Chase, J. M., Comita, L. S., Cornell, H. V., Davies, K. F., Harrison, S. P., Hurlbert, A. H., Inouye, B. D., Kraft, N. J. B., Myers, J. A., Sanders, N. J., Swenson, N. G., Vellend, M. (2013), Stochastic and Deterministic Drivers of Spatial and Temporal Turnover in Breeding Bird Communities. Global Ecology and Biogeography, 22: 202-212.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`individual.based.moments.b`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
```

```
colnames(comm) = bird.families$tip.label

#Set function f to be the Common Branch Length measure (CBL)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (cbl.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate 2000 randomized values of f on comm
# based on the individual-based null model.
individual.based.random.values.b(comm,f=my.f,args=arguments,reps=2000)
```

`permutation.communities.a`

Produces a set of random communities from a given matrix, based on the permutation (SIM2) null model. These communities can be used for alpha diversity computations

---

Description

Given a matrix M and a number of repetitions k, the function produces k random communities based on the permutation model. This is equivalent to shuffling M according to this model as many as k times , each time outputing only a certain row (e.g. the top one) of the shuffled matrix. An alpha diversity measure f can be applied on the output communities to determine the null distribution of f for a row in M. This distribution is the same for every row of M. This is because the examined null model produces the same distribution for all rows of M; after shuffling M, each row has the same probability to store a specific community as any other in the resulting matrix.

Usage

```
permutation.communities.a(matrix, reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A matrix which stores communities generated based on the permutation model. Each row of the matrix corresponds to a different randomized community. The number of columns, and the names of the columns in the output matrix are the same as in the input matrix. The output matrix is *not* a shuffled version of the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`permutation.random.values.a`

Examples

```
require(CNull)

#Create a random integer matrix
comm = matrix(sample(1:300),nrow=15,ncol=20)

#Use permutation model to produce 2000 random communities
permutation.communities.a(comm,reps=2000)
```

`permutation.communities.b`
Produces random pairs of communities from a given matrix, based on the permutation (SIM2) null model. These communities can be used for beta diversity computations

---

Description

Given a matrix M and a number of repetitions k, the function produces k pairs of random communities based on the permutation null model. This is equivalent to shuffling M according to this model as many as k times, each time outputing only two certain rows (e.g. the two top ones) of the shuffled matrix. A beta diversity measure f can be applied on the output communities to determine the null distribution of f for two rows in M. This distribution is the same for every pairs of rows in M. This is because the examined null model produces the same distribution for all pairs of rows in M; after shuffling M, each pair of rows has the same probability to store two specific communities as any other pair in the resulting matrix.

Usage

```
permutation.communities.b(matrix, reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A matrix which stores pairs of communities generated based on the permutation model. The output matrix has two times reps rows, corresponding to as many as reps pairs of randomized communities. For every i in 1:reps, rows in the output matrix with indices 2i-1 and 2i designate a valid random pair of communities. The number of columns, and the names of the columns in the output matrix are the same as in the input matrix. The output matrix is *not* a shuffled version of the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`permutation.random.values.b`

Examples

```
require(CNull)

#Create a random integer matrix
comm = matrix(sample(1:300),nrow=15,ncol=20)

#Use permutation model to produce 2000 random pairs of communities
permutation.communities.b(comm,reps=2000)
```

`permutation.moments.a` *Given a matrix and an alpha diversity measure f, calculates the mean and variance of f based on the permutation (SIM2) null model*

Description

Given a matrix M and an alpha diversity measure f, the function computes the mean and variance of f for a row in M, when M is shuffled according to the permutation model. The returned mean and variance is the same for every row in M. This is because the examined null model produces the same distribution for all rows in M; after shuffling M, each row has the same probability to store a specific community as any other row in the resulting matrix.

Usage

```
permutation.moments.a(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	An alpha diversity function f. The interface of f should be such that f(matrix,args) returns a numeric vector V where the i-th element of V is equal to the value of f when applied at the i-th row of the given matrix. To fit to this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A list with two real numbers; the mean and the variance of f on the given matrix for the permutation model.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`permutation.pvalues.a`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
```

```
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Phylogenetic Diversity measure (PD)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (pd.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate the mean and variance of f in the permutation null
# model using 2000 Monte Carlo randomizations
permutation.moments.a(comm,f=my.f,args=arguments,reps=2000)
```

permutation.moments.b *Given a matrix and a beta diversity measure f, calculates the mean and variance of f based on the permutation (SIM2) null model*

Description

Given a matrix M and a beta diversity measure f, the function computes the mean and variance of f between a pair of rows in M, when M is shuffled according to the permutation model. The returned mean and variance is the same for every pair of rows in M. This is because the examined null model produces the same distribution for all pairs of rows in M; after shuffling M, each pair of rows has the same probability to store two specific communities as any other pair in the resulting matrix.

Usage

```
permutation.moments.b(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	A beta diversity function f. The interface of f should be such that f(matrix,args) returns a matrix V where the entry stored at the i-th row and j-th column of V is equal to the value of f when applied at the i-th and j-th row of the input matrix. To fit this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A list with two real numbers; the mean and the variance of f on the given matrix for the permutation model.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

permutation.pvalues.b

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Common Branch Length measure (CBL)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (cbl.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate the mean and variance of f in the permutation null
# model using 2000 Monte Carlo randomizations
permutation.moments.b(comm,f=my.f,args=arguments,reps=2000)
```

---

permutation.pvalues.a *Produces the p-values of an alpha diversity measure f on a given matrix, based on the permutation (SIM2) null model*

Description

Given a matrix M and an alpha diversity measure f, the function calculates the p-values of f based on the permutation null model.

Usage

```
permutation.pvalues.a(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	An alpha diversity function f. The interface of f should be such that f(matrix,args) returns a numeric vector V where the i-th element of V is equal to the value of f when applied at the i-th row of the given matrix. To fit to this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A numeric vector that stores the p-values of f, calculated based on the permutation null model. The i-th element of the vector stores the p-value for the i-th row of the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`permutation.communities.a`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)
```

```
#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Phylogenetic Diversity measure (PD)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (pd.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate the p-values of f for the communities in comm
# based on the permutation model, using 2000 Monte Carlo randomizations.
permutation.pvalues.a(comm,f=my.f,args=arguments,reps=2000)
```

permutation.pvalues.b *Produces the p-values of a beta diversity measure f on a given matrix, based on the permutation (SIM2) null model*

Description

Given a matrix M and a beta diversity measure f, the function calculates the p-values of f based on the permutation null model.

Usage

```
permutation.pvalues.b(matrix, f, args, observed.vals, reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	A beta diversity function f. The interface of f should be such that f(matrix,args) returns a matrix V where the entry stored at the i-th row and j-th column of V is equal to the value of f when applied at the i-th and j-th row of the input matrix. To fit this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`observed.vals`	A set of pre-calculated values for which we want to compute their p-values.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A numeric vector that stores the p-values of f, calculated based on the permutation null model. The i-th element of the vector stores the p-value for the i-th element in vector observed.vals.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`permutation.communities.b`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Common Branch Length measure (CBL)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (cbl.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

#Compute the values of f for all pairs of observed communities in M.
#Turn the resulting matrix with the observed diversity values into a vector
obs.v=my.f(comm,arguments)
vals = as.vector(t(obs.v))

# Calculate the p-values of f for the communities in comm
# based on the permutation model, using 2000 Monte Carlo randomizations.
permutation.pvalues.b(comm,f=my.f,args=arguments,observed.vals=vals,reps=2000)
```

permutation.random.values.a

Produces a set of random values for an alpha diversity measure f on a given matrix, based on the permutation (SIM2) null model

Description

Given a matrix M, an alpha diversity measure f and a number of repetitions k, the function produces k random values of f based on the permutation model. This is equivalent to shuffling M according to this model as many as k times , each time outputing the value of f only for a certain row (e.g. the top one) of the shuffled matrix. The output values can be used to determine the null distribution of f for a row of M. This distribution is the same for every row of M. This is because the examined null model produces the same distribution for all rows of M; after shuffling M, each row has the same probability to store a specific community C as any other in the resulting matrix.

Usage

```
permutation.random.values.a(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	An alpha diversity function f. The interface of f should be such that f(matrix,args) returns a numeric vector V where the i-th element of V is equal to the value of f when applied at the i-th row of the given matrix. To fit to this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A vector of as many as reps elements. Stores the randomized values of f calculated based on the permutation null model.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`permutation.moments.a`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Phylogenetic Diversity measure (PD)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (pd.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate 2000 randomized values of f on comm
# based on the permutation null model.
permutation.random.values.a(comm,f=my.f,args=arguments,reps=2000)
```

`permutation.random.values.b`

Produces a set of random values for a beta diversity measure f on a given matrix, based on the permutation (SIM2) null model

---

Description

Given a matrix M, a beta diversity measure f and a number of repetitions k, the function produces k random values of f based on the permutation model. This is equivalent to shuffling M according to this model as many as k times , each time outputing the value of f only for a certain pair of rows (e.g. the two top ones) of the shuffled matrix. The output values can be used to determine the null distribution of f for a pair of rows in M. This distribution is the same for every pairs of rows in M. This is because the examined null model produces the same distribution for all row pairs of M; after shuffling M, each row has the same probability to store a given community as any other in the resulting matrix.

Usage

```
permutation.random.values.b(matrix,f,args,reps=1000)
```

Arguments

`matrix`	A matrix with integer values. The matrix should not contain any NA values.
`f`	A beta diversity function f. The interface of f should be such that f(matrix,args) returns a matrix V where the entry stored at the i-th row and j-th column of V is equal to the value of f when applied at the i-th and j-th row of the input matrix. To fit this interface, the user might have to develop f as a wrapper around an existing R function (see **Examples**).
`args`	A list with extra arguments needed by f.
`reps`	The number of randomizations. This argument is optional and its default value is set to one thousand.

Value

A vector of as many as reps elements. Stores the randomized values of f calculated based on the permutation null model.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Gotelli, N. J., 2000. Null Model Analysis of Species Co-Occurrence Patterns. Ecology, 81(9), pp.2606-2621.

Tsirogiannis, C., A. Kalvisa, B. Sandel and T. Conradi. Column-Shuffling Null Models Are Simpler Than You Thought. To appear.

See Also

`permutation.moments.b`

Examples

```
#In the next example null-model calculations are
#performed using a function of phylogenetic diversity.
#Hence, we first load the required packages.
require(CNull)
require(ape)
require(PhyloMeasures)

#Load phylogenetic tree of bird families from package "ape"
data(bird.families, package = "ape")

#Create 100 random communities with 50 families each
comm = matrix(0,nrow = 100,ncol = length(bird.families$tip.label))
for(i in 1:nrow(comm)) {comm[i,sample(1:ncol(comm),50)] = 1}
colnames(comm) = bird.families$tip.label

#Set function f to be the Common Branch Length measure (CBL)
#as defined in the R package PhyloMeasures.
my.f = function(mt,args){ return (cbl.query(args[[1]],mt))}

# This function takes one extra argument, which is a phylogenetic tree.
# Hence, create a list whose only element is the desired tree.
arguments = list()
arguments[[1]] = bird.families

# Calculate 2000 randomized values of f on comm
# based on the permutation null model.
permutation.random.values.b(comm,f=my.f,args=arguments,reps=2000)
```

Index