Package 'PhyloMeasures'

January 14, 2017

Type Package

Title Fast and Exact Algorithms for Computing Phylogenetic Biodiversity Measures

Version 2.1

Date 2017-1-14

Author Constantinos Tsirogiannis [aut, cre], Brody Sandel [aut]

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

Description Given a phylogenetic tree T and an assemblage S of species represented as a subset of tips in T, we want to compute a measure of the diversity of the species in S with respect to T. The current package offers efficient algorithms that can process large phylogenetic data for several such measures. Most importantly, the package includes algorithms for computing efficiently the standardized versions of phylogenetic measures and their p-values, which are essential for null model comparisons. Among other functions, the package provides efficient computation of richness-standardized versions for indices such as the net relatedness index (NRI), nearest taxon index (NTI), phylogenetic diversity index (PDI), and the corresponding indices of two-sample measures. The package also introduces a new single-sample measure, the Core Ancestor Cost (CAC); the package provides functions for computing the value and the standardised index of the CAC and, more than that, there is an extra function available that can compute exactly any statistical moment of the measure. The package supports computations under different null models, including abundance-weighted models.

License GPL-3

Imports ape, methods

LazyLoad yes

SystemRequirements C++11

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-01-14 01:13:19

R topics documented:

`PhyloMeasures-package` *PhyloMeasures: Fast Computations of Phylogenetic Biodiversity Measures*

Description

The package supports fast biodiversity computations on large phylogenetic data. More specifically, the package provides functions for computing the stdandardized values and p-values of several phylogenetic biodiversity measures. The measures which are supported in the current version of the package are: the Phylogenetic Diversity (PD), the Mean Pairwise Distance (MPD), the Mean Nearest Taxon Distance (MNTD), the Core Ancestor Cost (CAC), the Common Branch Length (CBL), the Community Distance (CD), the Community Distance Nearest Taxon (CDNT), the Phylogenetic Sorensen's Similarity (PhyloSor), and the Unique Fraction (UniFrac).

Details

Package:	PhyloMeasures
Type:	Package
Version:	2.1
Date:	2017-1-14
License:	GPL-3

The package contains three types of functions; functions that compute the (standardized) value of a phylogenetic biodiversity measure (query functions), functions that compute the p-values of a measure (p-value functions), and functions that compute the statistical moments of a measure (moment functions).

For the single sample measures, the computation of the standardized values, the statistical moments, and of the p-values can be done under three different null models. We refer to these models as uniform, frequency by richness, and sequential. All these models maintain species richness, and the latter two models are abundance-weighted. The strong advantage of this package is the very efficient algorithms that support the provided functions. These algorithms make it possible to process very large phylogenies in reasonable time even on a standard computer. For more details and experimental results, the reader can refer to the main paper for this package, but also the other works that introduce the efficient algorithms used in the package functions.

Author(s)

Constantinos Tsirogiannis and Brody Sandel

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

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A Package for Computing Phylogenetic Biodiversity Measures and Their Statistical Moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C. and B. Sandel. 2016. Fast Computation of Measures of Phylogenetic Beta Diversity. PLoS ONE, 11(4): e0151167, doi:10.1371/journal.pone.0151167.

Tsirogiannis, C., B. Sandel and D. Cheliotis. 2012. Efficient Computation of Popular Phylogenetic Tree Measures. Algorithms in Bioinformatics, LNCS 7534: 30-43.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New Algorithms for Computing Phylogenetic Biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

Webb, C.O. 2000. Exploring the Phylogenetic Structure of Ecological Communities: An Example for Rain Forest Trees. The American Naturalist 156: 145-155.

Examples

```
#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

#Use query function to calculate pd values for each community
pd.query(bird.families,comm)

#Use query function to calculate standardised versions under the uniform model
pd.query(bird.families,comm,TRUE)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
```

```
names(weights) = bird.families$tip.label

#Use query function to calculate standardized versions under the sequential model
pd.query(bird.families,comm,TRUE,null.model="sequential",
         abundance.weights=weights, reps=1000)
```

`cac.moments` *Computes the statistical moments of the Core Ancestor Cost measure*

Description

Calculates the mean and standard deviation of the Core Ancestor Cost (CAC) for a tree and a vector of tip set sizes. The means and deviations can be calculated under two different null models which maintain species richness. When the "uniform" null model is used, the function can calculate the first k statistical moments of this measure. The CAC is calculated by identifying the node in the tree that is the most recent common ancestor of at least chi proportion of the tips in the set, where chi is an input parameter larger than 0.5. The CAC is the distance of this node from the root of the tree.

Usage

```
cac.moments(tree, chi, sample.sizes, k=2,
            null.model="uniform", abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`chi`	A number in the interval (0.5,1]
`sample.sizes`	A vector of non-negative integers specifying the tip set sizes for which to calculate moments
`k`	A positive integer specifying the number of moments to compute (default = 2). If the "sequential" model is selected, the only values that can be used for this argument are either one or two.
`null.model`	A character vector (string) that defines which null model is used for computing the moments of the measure. There are two possible null models that can be used for computing the moments: these are "uniform" and "sequential". Both models maintain species richness. More specifically, the available models are defined as follows: • **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness. • **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other model (which is computed analytically), this model uses Monte-Carlo randomization. This argument is optional, and its default value is "uniform".

`abundance.weights`
: A vector of positive numeric values. These are the abundance weights that will be used if option "sequential" is selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

`reps`
: An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if the "uniform" model is selected.

`seed`
: A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if the "uniform" model is selected.

Value

A k-column matrix with length(sample.sizes) rows. Entry [i,j] in the matrix gives the j-th moment for the i-th sample size in sample.sizes . The first moment in each row is the mean, and for j larger than one, the j-th returned moment is the raw statistical moment of order j. Given a phylogenetic tree T, a sample size r and a value chi, the raw statistical moment of order j for the CAC is defined as:

$$E_{R \in Sub(T,r)}[\mathrm{CAC}_{\chi}(T,R)^{k}],$$

where $E_{R \in Sub(T,r)}$ denotes the expectation of a random variable for all tip sets in T that consist of r tips each.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

See Also

`cac.query`

Examples

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

# Calculate first four raw moments under the uniform model
cac.moments(bird.families,0.75,1:100,k=4)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

# Calculate mean and variance under the sequential model
cac.moments(bird.families,0.75,1:100,k=2,
```

```
null.model="sequential", abundance.weights=weights, reps=1000)
```

`cac.pvalues` *Computes the p-values of the Core Ancestor Cost measure*

Description

Calculates the p-values of the core ancestor cost (CAC) measure for sets of tips on a phylogeny. The p-values can be calculated under two different null models which maintain species richness. The CAC is calculated by identifying the node in the tree that is the most recent common ancestor of at least chi proportion of the tips in the set, where chi is an input parameter larger than 0.5. The CAC is the distance of this node from the root of the tree.

Usage

```
cac.pvalues(tree, matrix, chi, null.model="uniform",
            abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`matrix`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error.
`chi`	A number in the interval (0.5,1]
`null.model`	A character vector (string) that defines which null model is used for computing the p-values of the measure. There are two possible null models that can be used for computing the p-values: these are "uniform" and "sequential". Both models maintain species richness. More specifically, the available models are defined as follows: • **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness. • **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other model (which is computed analytically), this model uses Monte-Carlo randomization. This argument is optional, and its default value is set to "uniform".
`abundance.weights`	A vector of positive numeric values. These are the abundance weights that will be used if option "sequential" is selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

reps	An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if the other null model is selected.
seed	A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if the other null model is selected.

Value

A vector which stores the computed CAC p-values. The i-th entry in this vector stores the CAC p-value of the i-th row in the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

See Also

cac.moments

Examples

```
#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

chi=0.6

#Calculate p-values under the uniform model
cac.pvalues(bird.families,comm,chi, reps=1000)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

# Calculate p-values under the sequential model
cac.pvalues(bird.families,comm,chi,null.model="sequential",
            abundance.weights=weights, reps=1000)
```

`cac.query` *Computes the (standardized) value of the Core Ancestor Cost measure*

Description

Calculates the Core Ancestor Cost (CAC) for sets of tips on a phylogeny. The CAC is calculated by identifying the node in the tree that is the most recent common ancestor of at least chi proportion of the tips in the set. The CAC is the distance of this node from the root of the tree. The same function can also calculate the standardized value of this measure under three different null models which maintain species richness.

Usage

```
cac.query(tree, matrix, chi, standardize = FALSE,
          null.model="uniform", abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`matrix`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error.
`chi`	A number in the interval (0.5,1]
`standardize`	Specifies whether the function should standardize the CAC value according to the variation in species richness. For each input tip set R, the CAC value for R is standardized by subtracting the mean CAC and dividing by the standard deviation of this measure. The mean and standard deviation are calculated among all tip sets that have the same number of elements as set R (default = FALSE)
`null.model`	A character vector (string) that defines which null model is used for computing the standardized values of the measure. There are two possible null models that can be used for computing the standardized values: these are "uniform", and "sequential". Both models maintain species richness. More specifically, the available models are defined as follows: • **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness. • **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other model (which is computed analytically), this model uses Monte-Carlo randomization. This argument is optional, and its default value is set to "uniform".

abundance.weights
: A vector of positive numeric values. These are the abundance weights that will be used if option "sequential" is selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

reps
: An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if the "uniform" model selected.

seed
: A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if this model is not selected.

Value

A vector which stores the computed (standardized) CAC values. The i-th entry in this vector stores the CAC value of the i-th row in the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

See Also

cac.moments

Examples

```
#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

#Calculate cac values for each community
cac.query(bird.families,comm,0.8)

#Calculate standardized versions under the uniform model
cac.query(bird.families,comm,0.8,TRUE)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
```

```
names(weights) = bird.families$tip.label

#Use query function to calculate standardized versions under the sequential model
cac.query(bird.families,comm,0.8,TRUE,null.model="sequential",
          abundance.weights=weights, reps=1000)
```

`cbl.moments` *Computes the moments of the Common Branch Length measure under the uniform null model*

Description

Calculates the mean and standard deviation of the Common Branch Length (CBL) on a given tree for several pairs of tip set sizes. These calculations consider equal (uniform) probability among all possible tip samples of the same richness.

Usage

```
cbl.moments(tree, sample.sizes, comp.expectation = TRUE,
            comp.deviation = TRUE)
```

Arguments

`tree`	A phylo tree object
`sample.sizes`	A two-column matrix of non-negative integers indicating the tip set sizes for which the moments should be calculated. The moments are calculated for each row of the matrix. Let x and y be the values that are stored in the i-th row of sample.sizes. For this row, the CBL moments are calculated considering all pairs of tip sets that can be extracted from the input tree such that one set has x elements and the other set has y elements.
`comp.expectation`	Specifies whether the function computes the mean of the CBL (default = TRUE)
`comp.deviation`	Specifies whether the function computes the standard deviation of the CBL (default = TRUE)

Value

If both comp.expectation and comp.deviation are TRUE, the function returns a two-column matrix with one row per element in sample.sizes; the first column of the output matrix stores the mean CBL and the second column stores the standard deviation of this measure. If only one of comp.expectation or comp.deviation is TRUE, the function returns a vector with the corresponding values instead.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Graham, C.H. and P.V.A. Fine. 2008. Phylogenetic beta diversity: linking ecological and evolutionary processes across space and time. Ecology Letters 11: 1265:1277.

Swenson, N.G. 2011. Phylogenetic beta diversity metrics, trait evolution and inferring functional beta diversity of communities. PLoS ONE: 6: e21264.

Tsirogiannis, C. and B. Sandel. In prep. Fast computation of measures of phylogenetic beta diversity.

See Also

cbl.query

Examples

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

cbl.moments(bird.families,sample.sizes = expand.grid(1:10,1:10))
```

---

cbl.query — *Computes the (standardized) value of the Common Branch Length measure*

Description

Calculates the Common Branch Length (CBL) given paired sets of tips on a phylogeny. The Common Branch Length is the beta diversity version of Phylogenetic Diversity (PD), giving the total branch length shared between two communities. The same function can also calculate the standardized value of this measure for the given tip sets. The standardized calculations consider equal (uniform) probability among all possible tip samples of the same richness.

Usage

```
cbl.query(tree, matrix.a, matrix.b = NULL,
          query.matrix = NULL, standardize = FALSE)
```

Arguments

tree	A phylo tree object
matrix.a	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error
matrix.b	Optional, a second matrix with a similar format as matrix.a

`query.matrix` Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the CBL values. Each row in query.matrix indicates a pair of tip sets for which we want to compute the CBL value. Let k and r be the values that are stored in the i-th row of query.matrix, where k is the value stored in the first column and r is the value stored in the second column. If matrix.b is given, the function computes the CBL value between the k-th row of matrix.a and the r-th row of matrix.b. If matrix.b is not given, the function computes the CBL value between the k-th and r-th row of matrix.a (default = NULL)

`standardize` Specifies whether the function should return the standardized value of the CBL for each sample pair. The value is standardized by subtracting the mean and dividing by the standard deviation of CBL. The mean and standard deviation are calculated among all tip sets that have the same number of elements as the two samples (default = FALSE)

Details

Queries can be given in four ways. If neither matrix.b nor query.matrix are given, the function computes the CBL values for all pairs of rows (tip sets) in matrix.a . If matrix.b is given but not query.matrix, the function computes the CBL values for all combinations of a row in matrix.a with rows in matrix.b. If query.matrix is given and matrix.b is not, the function returns the CBL values for the pairs of rows in matrix.a specified by query.matrix. If query.matrix and matrix.b are both given, CBL values are computed for the rows in matrix.a specified by the first column of query.matrix against the rows in matrix.b specified in the second column of query.matrix.

Value

The CBL values for the requested pairs of tip sets. If query.matrix is provided, then the values are returned in an one-dimensional vector. The i-th element of this vector is the CBL value for the pair of tip sets indicated in the i-th row of query.matrix. If query.matrix is not provided, the CBL values are returned in a matrix object; entry [i,j] in the output matrix stores the CBL value between the tip sets specified on the i-th and j-th row of matrix.a (if matrix.b is not specified), or the CBL value between the i-th row of matrix.a and the j-th row of matrix.b (if matrix.b is specified)

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Graham, C.H. and P.V.A. Fine. 2008. Phylogenetic beta diversity: linking ecological and evolutionary processes across space and time. Ecology Letters 11: 1265:1277.

Swenson, N.G. 2011. Phylogenetic beta diversity metrics, trait evolution and inferring functional beta diversity of communities. PLoS ONE: 6: e21264.

Tsirogiannis, C. and B. Sandel. In prep. Fast computation of measures of phylogenetic beta diversity.

See Also

`cbl.moments`

Examples

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

#Create 10 random communities with 50 families each
comm = matrix(0,nrow = 10,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

#Calculate all pairwise CBL values for communities in comm
cbl.query(bird.families,comm)

#Calculate pairwise CBL values from
#the first two rows of comm to all rows
cbl.query(bird.families, comm[1:2,],comm)

#Calculate the CBL from the first two rows
#to all rows using the query matrix
qm = expand.grid(1:2,1:10)
cbl.query(bird.families,comm,query.matrix = qm)

#Calculate standardized versions
cbl.query(bird.families,comm, standardize = TRUE)
```

---

cd.moments *Computes the moments of the Community Distance measure under the uniform null model*

Description

Calculates the mean and standard deviation of the Community Distance (CD) on a given tree for several pairs of tip set sizes. These calculations consider equal (uniform) probability among all possible tip samples of the same richness.

Usage

```
cd.moments(tree, sample.sizes, comp.expectation = TRUE,
           comp.deviation = TRUE)
```

Arguments

tree	A phylo tree object
sample.sizes	A two-column matrix of non-negative integers indicating the tip set sizes for which the moments should be calculated. The moments are calculated for each row of the matrix. Let x and y be the values that are stored in the i-th row of sample.sizes. For this row, the CD moments are calculated considering all pairs of tip sets that can be extracted from the input tree such that one set has x elements and the other set has y elements.

`comp.expectation`
: Specifies whether the function returns the mean of the CD (default = TRUE)

`comp.deviation`
: Specifies whether the function returns the standard deviation of the CD (default = TRUE)

Value

If both comp.expectation and comp.deviation are TRUE, the function returns a two-column matrix with one row per element in sample.sizes, such that the first column stores the mean CD and the second column stores the standard deviation of this measure. If only one of comp.expectation or comp.deviation is TRUE, the function returns a vector with the corresponding values instead.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Graham, C.H. and P.V.A. Fine. 2008. Phylogenetic beta diversity: linking ecological and evolutionary processes across space and time. Ecology Letters 11: 1265:1277.

Swenson, N.G. 2011. Phylogenetic beta diversity metrics, trait evolution and inferring functional beta diversity of communities. PLoS ONE: 6: e21264.

Tsirogiannis, C. and B. Sandel. In prep. Fast computation of measures of phylogenetic beta diversity.

See Also

`cd.query`

Examples

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

cd.moments(bird.families,sample.sizes = expand.grid(1:10,1:10))
```

---

cd.query *Computes the (standardized) value of the Community Distance measure*

Description

Calculates the Community Distance (CD) given paired sets of tips on a phylogeny. The Community Distance is the beta diversity version of Mean Pairwise Distance (MPD), giving the average phylogenetic distance between two communities. The same function can also calculate the standardized value of this measure for the given tip sets. The standardized calculations consider equal (uniform) probability among all possible tip samples of the same richness.

Usage

```
cd.query(tree, matrix.a, matrix.b = NULL,
         query.matrix = NULL, standardize = FALSE)
```

Arguments

tree	A phylo tree object
matrix.a	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error
matrix.b	Optional, a second matrix with a similar format as matrix.a
query.matrix	Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the CD values. Each row in query.matrix indicates a pair of tip sets for which we want to compute the CD value. Let k and r be the values that are stored in the i-th row of query.matrix, where k is the value stored in the first column and r is the value stored in the second column. If matrix.b is given, the function computes the CD value between the k-th row of matrix.a and the r-th row of matrix.b. If matrix.b is not given, the function computes the CD value between the k-th and r-th row of matrix.a (default = NULL)
standardize	Specifies whether the function should return the standardized value of the CD for each sample pair. The value is standardized by subtracting the mean and dividing by the standard deviation of the CD. The mean and standard deviation are calculated among all tip sets that have the same number of elements as the two samples (default = FALSE)

Details

Queries can be given in four ways. If neither matrix.b nor query.matrix are given, the function computes the CD values for all pairs of rows (tip sets) in matrix.a . If matrix.b is given but not query.matrix, the function computes the CD values for all combinations of a row in matrix.a with rows in matrix.b. If query.matrix is given and matrix.b is not, the function returns the CD values for the pairs of rows in matrix.a specified by query.matrix. If query.matrix and matrix.b are both given, CD values are computed for the rows in matrix.a specified by the first column of query.matrix against the rows in matrix.b specified in the second column of query.matrix.

Value

The CD values for the requested pairs of tip sets. If query.matrix is provided, then the values are returned in an one-dimensional vector. The i-th element of this vector is the CD value for the pair of tip sets indicated in the i-th row of query.matrix. If query.matrix is not provided, the CD values are returned in a matrix object; entry [i,j] in the output matrix stores the CD value between the tip sets specified on the i-th and j-th row of matrix.a (if matrix.b is not specified), or the CD value between the i-th row of matrix.a and the j-th row of matrix.b (if matrix.b is specified)

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Graham, C.H. and P.V.A. Fine. 2008. Phylogenetic beta diversity: linking ecological and evolutionary processes across space and time. Ecology Letters 11: 1265:1277.

Swenson, N.G. 2011. Phylogenetic beta diversity metrics, trait evolution and inferring functional beta diversity of communities. PLoS ONE: 6: e21264.

Tsirogiannis, C. and B. Sandel. In prep. Fast computation of measures of phylogenetic beta diversity.

See Also

cd.moments

Examples

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

#Create 10 random communities with 50 families each
comm = matrix(0,nrow = 10,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

#Calculate all pairwise CD values for communities in comm
cd.query(bird.families,comm)

#Calculate pairwise distances from
#the first two rows of comm to all rows
cd.query(bird.families, comm[1:2,],comm)

#Calculate the distances from the first two rows
#to all rows using the query matrix
qm = expand.grid(1:2,1:10)
cd.query(bird.families,comm,query.matrix = qm)

#Calculate standardized versions
cd.query(bird.families,comm,standardize = TRUE)
```

---

cdnt.averaged.query *Computes the value of the averaged Community Distance Nearest Taxon measure*

Description

Calculates the averaged Community Distance Nearest Taxon (aCDNT) given paired sets of tips on a phylogeny. The aCDNT is a beta diversity version of MNTD. It is computed based on the values of the directed CDNT (dCDNT); let A and B be two tip sets in the input tree. Let dCDNT(A,B) be the directed CDNT from A to B, and let dCDNT(B,A) be the directed CDNT from B to A. Let also |A| denote the number of elements in A, and |B| the number of elements in B. The aCDNT between these two tip sets is equal to:

$$\frac{|A| \cdot \mathrm{dCDNT}(A, B) + |B| \cdot \mathrm{dCDNT}(B, A)}{|A| + |B|}$$

The version of this function that computes the standardised value of aCDNT is not yet available.

Usage

```
cdnt.averaged.query(tree, matrix.a, matrix.b = NULL,
                    query.matrix = NULL)
```

Arguments

`tree`	A phylo tree object
`matrix.a`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error
`matrix.b`	Optional, a second matrix with a similar format as matrix.a
`query.matrix`	Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the aCDNT values. Each row in query.matrix indicates a pair of tip sets for which we want to compute the aCDNT value. Let k and r be the values that are stored in the i-th row of query.matrix, where k is the value stored in the first column and r is the value stored in the second column. If matrix.b is given, the aCDNT value is computed between the k-th row of matrix.a and the r-th row of matrix.b. If matrix.b is not given, the aCDNT value is computed between the k-th and r-th row of matrix.a (default = NULL)

Details

Queries can be given in four ways. If neither matrix.b nor query.matrix are given, the function computes the aCDNT values for all pairs of rows (tip sets) in matrix.a . If matrix.b is given but not query.matrix, the function computes the aCDNT values for all combinations of a row in matrix.a with rows in matrix.b. If query.matrix is given and matrix.b is not, the function returns the aCDNT values for the pairs of rows in matrix.a specified by query.matrix. If query.matrix and matrix.b are both given, aCDNT values are computed for the rows in matrix.a specified by the first column of query.matrix against the rows in matrix.b specified in the second column of query.matrix

Value

The aCDNT values for the requested pairs of tip sets. If query.matrix is provided, then the values are returned in an one-dimensional vector. The i-th element of this vector is the aCDNT value for the pair of tip sets indicated in the i-th row of query.matrix. If query.matrix is not provided, the aCDNT values are returned in a matrix object; entry [i,j] in the output matrix stores the aCDNT value between the tip sets specified on the i-th and j-th row of matrix.a (if matrix.b is not specified), or the aCDNT value between the i-th row of matrix.a and the j-th row of matrix.b (if matrix.b is specified)

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Graham, C.H. and P.V.A. Fine. 2008. Phylogenetic beta diversity: linking ecological and evolutionary processes across space and time. Ecology Letters 11: 1265:1277.

Swenson, N.G. 2011. Phylogenetic beta diversity metrics, trait evolution and inferring functional beta diversity of communities. PLoS ONE: 6: e21264.

See Also

cdnt.directed.query, cdnt.query

Examples

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

#Create 10 random communities with 50 families each
comm = matrix(0,nrow = 10,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

#Calculate all pairwise aCDNT values for communities in comm
cdnt.averaged.query(bird.families,comm)

#Calculate pairwise distances from
#the first two rows of comm to all rows
cdnt.averaged.query(bird.families, comm[1:2,],comm)

#Calculate the distances from the first two rows
#to all rows using the query matrix
qm = expand.grid(1:2,1:10)
cdnt.averaged.query(bird.families,comm,query.matrix = qm)
```

`cdnt.directed.query` *Computes the value of the directed Community Distance Nearest Taxon measure*

Description

Calculates the directed Community Distance Nearest Taxon (dCDNT) given paired sets of tips on a phylogeny. The dCDNT is a beta diversity version of MNTD, giving the mean distance on the tree of tips in a set A to their nearest neighbors in a set B. Note that the measure is asymmetrical, as for taxon t in A the nearest neighbor in B may be taxon s, but for taxon s the nearest neighbor in A might be a third taxon u. Therefore, for every input pair of tip sets A and B, the function returns two values; the dCDNT value from A to B, and the dCDNT value from B to A. The version of this function that computes the standardised value of the measure is not yet available.

Usage

```
cdnt.directed.query(tree, matrix.a, matrix.b = NULL,
                    query.matrix = NULL)
```

Arguments

`tree`	A phylo tree object
`matrix.a`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error
`matrix.b`	Optional, a second matrix with a similar format as matrix.a
`query.matrix`	Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the dCDNT values. Each row in query.matrix indicates a pair of tip sets for which we want to compute the two dCDNT values. Let k and r be the values that are stored in the i-th row of query.matrix, where k is the value stored in the first column and r is the value stored in the second column. If matrix.b is given, the function computes both the dCDNT value from the tip set in the k-th row of matrix.a to the tip set in the r-th row of matrix.b, and the dCDNT value from the tip set in the r-th row of matrix.b to the tip set in the k-th row of matrix.a . If matrix.b is not given, the two values computed are the dCDNT from the k-th to the r-th set of matrix.a, and the dCDNT from the r-th to the k-th set of matrix.a (default = NULL)

Details

Queries can be given in four ways. If neither matrix.b nor query.matrix are given, the function computes the two dCDNT values for each pair of rows in matrix.a . If matrix.b is given but not query.matrix, the function computes the two dCDNT values for each combination of a row in matrix.a with a row in matrix.b . If query.matrix is given and matrix.b is not, the function returns

two dCDNT values for each pair of rows in matrix.a specified by query.matrix. If query.matrix and matrix.b are both given, two dCDNT values are computed for each pair of a row in matrix.a specified by the fist column of query.matrix with a row in matrix.b specified in the second column of query.matrix.

Value

The dCDNT values for the requested pairs of tip sets. If query.matrix is provided, then the values are returned in a list that contains two elements; each element is a one-dimensional vector storing dCDNT values. Let k and r be the values stored in the i-th row of query.matrix. For the first vector in the returned list, the i-th element of this vector is the dCDNT value from the tip set of the k-th row in matrix.a to the tip set of the r-th row in matrix.a (if matrix.b is not specified), or to the tip set of the r-th row of matrix.b (if this matrix is specified). For the second vector of the output list, the i-th element of this vector is the dCDNT value from the tip set of the r-th row in matrix.a (if matrix.b is not specified), or from the tip set of the r-th row of matrix.b (if this matrix is specified) to the tip set of the k-th row in matrix.a.

If query.matrix is not provided and matrix.b is provided, the function returns a list which consists of two matrix objects; for the first matrix in this list, entry [i,j] stores the dCDNT value from the tip set specified on the i-th row of matrix.a to the tip set in the j-th row of matrix.b. For the second matrix in the output list, entry [i,j] stores the dCDNT value from the tip set specified on the j-th row of matrix.b to the tip set in the i-th row of matrix.a.

If neither query.matrix nor matrix.b are provided, the function returns a matrix object such that matrix entry [i,j] stores the dCDNT value from the tip set specified on the i-th row of matrix.a to the tip set in the j-th row of this matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Graham, C.H. and P.V.A. Fine. 2008. Phylogenetic beta diversity: linking ecological and evolutionary processes across space and time. Ecology Letters 11: 1265:1277.

Swenson, N.G. 2011. Phylogenetic beta diversity metrics, trait evolution and inferring functional beta diversity of communities. PLoS ONE: 6: e21264.

See Also

cdnt.query, cdnt.averaged.query

Examples

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

#Create 10 random communities with 50 families each
comm = matrix(0,nrow = 10,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
```

```
#Calculate all pairwise dCDNT values for communities in comm
cdnt.directed.query(bird.families,comm)
#Note that this matrix is asymmetrical, compare to:
cdnt.query(bird.families,comm)
cdnt.averaged.query(bird.families,comm)

#Calculate pairwise distances from
#the first two rows of comm to all rows
cdnt.directed.query(bird.families, comm[1:2,],comm)

#Calculate the distances from the first two rows
#to all rows using the query matrix
qm = expand.grid(1:2,1:10)
cdnt.directed.query(bird.families,comm,query.matrix = qm)
```

`cdnt.query` *Computes the value of the maximised Community Distance Nearest Taxon measure*

Description

Calculates the maximised Community Distance Nearest Taxon (mCDNT) given paired sets of tips on a phylogeny. The mCDNT is a beta diversity version of MNTD, and provides the distance on the tree between nearest neighbors in sets A and B. It is derived by calculating the directed CDNT (dCDNT) from set A to set B, and from set B to set A, and taking the maximum of these two values. A version of this function that computes the standardised value of the measure is not yet available.

Usage

```
cdnt.query(tree, matrix.a, matrix.b = NULL,
           query.matrix = NULL)
```

Arguments

`tree`	A phylo tree object
`matrix.a`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error
`matrix.b`	Optional, a second matrix with a similar format as matrix.a
`query.matrix`	Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the mCDNT values. Each row in query.matrix indicates a pair of tip sets for which we want to compute the mCDNT value. Let k and r be the values that are stored in the i-th row of query.matrix, where k is the

value stored in the first column and r is the value stored in the second column. If matrix.b is given, the mCDNT value is computed between the k-th row of matrix.a and the r-th row of matrix.b. If matrix.b is not given, the mCDNT value is computed between the k-th and r-th row of matrix.a (default = NULL)

Details

Queries can be given in four ways. If neither matrix.b nor query.matrix are given, the function computes the mCDNT values for all pairs of rows (tip sets) in matrix.a . If matrix.b is given but not query.matrix, the function computes the mCDNT values for all combinations of a row in matrix.a with rows in matrix.b. If query.matrix is given and matrix.b is not, the function returns the mCDNT values for the pairs of rows in matrix.a specified by query.matrix. If query.matrix and matrix.b are both given, mCDNT values are computed for the rows in matrix.a specified by the first column of query.matrix against the rows in matrix.b specified in the second column of query.matrix

Value

The mCDNT values for the requested pairs of tip sets. If query.matrix is provided, then the values are returned in an one-dimensional vector. The i-th element of this vector is the mCDNT value for the pair of tip sets indicated in the i-th row of query.matrix. If query.matrix is not provided, the mCDNT values are returned in a matrix object; entry [i,j] in the output matrix stores the mCDNT value between the tip sets specified on the i-th and j-th row of matrix.a (if matrix.b is not specified), or the mCDNT value between the i-th row of matrix.a and the j-th row of matrix.b (if matrix.b is specified)

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Graham, C.H. and P.V.A. Fine. 2008. Phylogenetic beta diversity: linking ecological and evolutionary processes across space and time. Ecology Letters 11: 1265:1277.

Swenson, N.G. 2011. Phylogenetic beta diversity metrics, trait evolution and inferring functional beta diversity of communities. PLoS ONE: 6: e21264.

See Also

cdnt.directed.query, cdnt.averaged.query

Examples

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

#Create 10 random communities with 50 families each
comm = matrix(0,nrow = 10,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
```

```
#Calculate all pairwise mCDNT values for communities in comm
cdnt.query(bird.families,comm)

#Calculate pairwise distances from
#the first two rows of comm to all rows
cdnt.query(bird.families, comm[1:2,],comm)

#Calculate the distances from the first two rows
#to all rows using the query matrix
qm = expand.grid(1:2,1:10)
cdnt.query(bird.families,comm,query.matrix = qm)
```

`mntd.moments` *Computes the moments of the Mean Nearest Taxon Distance measure*

Description

Calculates the mean and standard deviation of the Mean Nearest Taxon Distance (MNTD) for a tree and a vector of tip set sizes. The means and deviations can be calculated under three different null models which maintain species richness. Note: this function returns a result for the "uniform" and "frequency.by.richness" models only if the input tree is ultrametric.

Usage

```
mntd.moments(tree, sample.sizes, comp.expectation = TRUE, comp.deviation = TRUE,
            null.model="uniform", abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`sample.sizes`	A vector of non-negative integers specifying the tip set sizes for which to calculate moments
`comp.expectation`	Specifies whether the function should compute the mean (default = TRUE)
`comp.deviation`	Specifies whether the function should compute the standard deviation (default = TRUE)
`null.model`	A character vector (string) that defines which null model is used for computing the moments of the measure. There are three possible null models that can be used for computing the moments: these are "uniform", "frequency.by.richness", and "sequential". All these models maintain species richness. More specifically, the available models are defined as follows: • **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness. • **"frequency.by.richness"** is an abundance-weighted model where species samples are chosen in a manner similar to the following process; first, each

species is selected independently with probability proportional to its abundance. If the resulting sample consists of exactly the same number of elements as the input assemblage then it is used by the null model, otherwise it is tossed and the whole process is repeated.

- **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other two models (which are computed analytically), this model uses Monte-Carlo randomization.

This argument is optional, and its default value is set to "uniform".

`abundance.weights`
: A vector of positive numeric values. These are the abundance weights that will be used if either of the options "frequency.by.richness" or "sequential" are selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

`reps`
: An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if any of the other two null models is selected.

`seed`
: A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if any of the other two null models is selected.

Value

If both comp.expectation and comp.deviation are TRUE, the function returns a two-column matrix with one row per element in sample.sizes, where the first column stores the mean MNTD and the second column stores the standard deviation for this sample size. If only one of comp.expectation or comp.deviation are TRUE, the function returns a vector with the corresponding values instead.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

Webb, C.O. 2000. Exploring the phylogenetic structure of ecological communities: An example for rain forest trees. The American Naturalist 156: 145-155.

See Also

`mntd.query`

Examples

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

# Calculate mean and variance under the uniform model
mntd.moments(bird.families,1:100)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

# Calculate mean and variance under the sequential model
mntd.moments(bird.families,1:100,
             null.model="sequential", abundance.weights=weights, reps=1000)
```

`mntd.pvalues` *Computes the p-values of the Mean Nearest Taxon Distance measure*

Description

Calculates the p-values of the mean nearest taxon distance (MNTD) measure for sets of tips on a phylogeny. The p-values can be calculated under two different null models which maintain species richness.

Usage

```
mntd.pvalues(tree, matrix, null.model="uniform",
             abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`matrix`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error.
`null.model`	A character vector (string) that defines which null model is used for computing the p-values of the measure. There are two possible null models that can be used for computing the p-values: these are "uniform" and "sequential". Both models maintain species richness. More specifically, the available models are defined as follows: • **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness.

- **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other model (which is computed analytically), this model uses Monte-Carlo randomization.

This argument is optional, and its default value is set to "uniform".

`abundance.weights`
: A vector of positive numeric values. These are the abundance weights that will be used if option "sequential" is selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

`reps`
: An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if the other null model is selected.

`seed`
: A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if the other null model is selected.

Value

A vector which stores the computed MNTD p-values. The i-th entry in this vector stores the MNTD p-value of the i-th row in the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and D. Cheliotis. 2012. Efficient computation of popular phylogenetic tree measures. Algorithms in Bioinformatics, LNCS 7534: 30-43.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

See Also

`mntd.moments`

Examples

```
#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
```

```
#Calculate p-values under the uniform model
mntd.pvalues(bird.families,comm, reps=1000)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

# Calculate p-values under the sequential model
mntd.pvalues(bird.families,comm,null.model="sequential",
             abundance.weights=weights, reps=1000)
```

`mntd.query` *Computes the (standardized) value of the Mean Nearest Taxon Distance measure*

Description

Calculates the Mean Nearest Taxon Distance (MNTD) for sets of tips on a phylogeny. The same function can also calculate the standardised value of this measure under three different null models which maintain species richness (the standardised value is equal to minus one times the so-called Nearest Taxon Index, NTI). Note: if the input argument is.standardised is set to TRUE and the null model used is either "uniform" or "frequency.by.richness", then this function returns a result only if the input tree is ultrametric.

Usage

```
mntd.query(tree, matrix, standardize = FALSE,
           null.model="uniform", abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`matrix`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error.
`standardize`	Specifies whether the function should standardize the MNTD for variation in species richness. For each tip set S, the observed MNTD is standardized by subtracting the mean MNTD and dividing by the standard deviation of this measure. The mean and standard deviation are calculated among all tip sets that have the same number of elements as set S, the tip set whose value we want to standardize (default = FALSE).

`null.model` A character vector (string) that defines which null model is used for computing the standardized values of the measure. There are three possible null models that can be used for computing the standardized values: these are "uniform", "frequency.by.richness", and "sequential". All these models maintain species richness. More specifically, the available models are defined as follows:

- **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness.
- **"frequency.by.richness"** is an abundance-weighted model where species samples are chosen in a manner similar to the following process; first, each species is selected independently with probability proportional to its abundance. If the resulting sample consists of exactly the same number of elements as the input assemblage then it is used by the null model, otherwise it is tossed and the whole process is repeated.
- **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other two models (which are computed analytically), this model uses Monte-Carlo randomization.

This argument is optional, and its default value is set to "uniform".

`abundance.weights` A vector of positive numeric values. These are the abundance weights that will be used if either of the options "frequency.by.richness" or "sequential" are selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

`reps` An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if any of the other two null models is selected.

`seed` A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if any of the other two null models is selected.

Value

A vector which stores the computed (standardized) MNTD values. The i-th entry in this vector stores the (standardized) MNTD value of the i-th row in the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

Webb, C.O. 2000. Exploring the phylogenetic structure of ecological communities: An example for rain forest trees. The American Naturalist 156: 145-155.

See Also

mntd.moments

Examples

```
#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

#Calculate mntd values for each community
mntd.query(bird.families,comm)

#Calculate standardized versions under the uniform model
mntd.query(bird.families,comm,TRUE)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

#Use query function to calculate standardized versions under the sequential model
mntd.query(bird.families,comm,TRUE,null.model="sequential",
           abundance.weights=weights, reps=1000)
```

---

mpd.moments *Computes the moments of the Mean Pairwise Distance measure*

Description

Calculates the mean and standard deviation of the Mean Pairwise Distance (MPD) for a tree and a vector of tip set sizes. The means and deviations can be calculated under three different null models which maintain species richness.

Usage

```
mpd.moments(tree, sample.sizes, comp.expectation = TRUE, comp.deviation = TRUE,
            null.model="uniform", abundance.weights, reps=1000, seed)
```

Arguments

tree	A phylo tree object
sample.sizes	A vector of non-negative integers specifying the tip set sizes for which to calculate moments
comp.expectation	Specifies whether the function should compute the mean (default = TRUE)

`comp.deviation` Specifies whether the function should compute the standard deviation (default = TRUE)

`null.model` A character vector (string) that defines which null model is used for computing the moments of the measure. There are three possible null models that can be used for computing the moments: these are "uniform", "frequency.by.richness", and "sequential". All these models maintain species richness. More specifically, the available models are defined as follows:

- **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness.
- **"frequency.by.richness"** is an abundance-weighted model where species samples are chosen in a manner similar to the following process; first, each species is selected independently with probability proportional to its abundance. If the resulting sample consists of exactly the same number of elements as the input assemblage then it is used by the null model, otherwise it is tossed and the whole process is repeated.
- **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other two models (which are computed analytically), this model uses Monte-Carlo randomization.

This argument is optional, and its default value is set to "uniform".

`abundance.weights` A vector of positive numeric values. These are the abundance weights that will be used if either of the options "frequency.by.richness" or "sequential" are selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

`reps` An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if any of the other two null models is selected.

`seed` A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if any of the other two null models is selected.

Value

If both comp.expectation and comp.deviation are TRUE, the function returns a two-column matrix with one row per element in sample.sizes, where the first column stores the mean MPD and the second column stores the standard deviation for this sample size. If only one of comp.expectation or comp.deviation are TRUE, the function returns a vector with the corresponding values instead.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Webb, C.O. 2000. Exploring the phylogenetic structure of ecological communities: An example for rain forest trees. The American Naturalist 156: 145-155.

See Also

mpd.query

Examples

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

# Calculate mean and variance under the uniform model
mpd.moments(bird.families,1:100)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

# Calculate mean and variance under the sequential model
mpd.moments(bird.families,1:100,
            null.model="sequential", abundance.weights=weights, reps=1000)
```

---

mpd.pvalues *Computes the p-values of the Mean Pairwise Distance measure*

Description

Calculates the p-values of the mean pairwise distance (MPD) measure for sets of tips on a phylogeny. The p-values can be calculated under two different null models which maintain species richness.

Usage

```
mpd.pvalues(tree, matrix, null.model="uniform",
            abundance.weights, reps=1000, seed)
```

Arguments

tree	A phylo tree object
matrix	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error.

`null.model` A character vector (string) that defines which null model is used for computing the p-values of the measure. There are two possible null models that can be used for computing the p-values: these are "uniform" and "sequential". Both models maintain species richness. More specifically, the available models are defined as follows:

- **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness.
- **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other model (which is computed analytically), this model uses Monte-Carlo randomization.

This argument is optional, and its default value is set to "uniform".

`abundance.weights` A vector of positive numeric values. These are the abundance weights that will be used if option "sequential" is selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

`reps` An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if the other null model is selected.

`seed` A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if the other null model is selected.

Value

A vector which stores the computed MPD p-values. The i-th entry in this vector stores the MPD p-value of the i-th row in the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and D. Cheliotis. 2012. Efficient computation of popular phylogenetic tree measures. Algorithms in Bioinformatics, LNCS 7534: 30-43.

See Also

`mpd.moments`

Examples

```
#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

#Calculate p-values under the uniform model
mpd.pvalues(bird.families,comm, reps=1000)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

# Calculate p-values under the sequential model
mpd.pvalues(bird.families,comm,null.model="sequential",
            abundance.weights=weights, reps=1000)
```

`mpd.query` *Computes the (standardized) value of the Mean Pairwise Distance measure*

Description

Calculates the Mean Pairwise Distance (MPD) measure for sets of tips on a phylogeny. The same function can also calculate the standardized value of this measure under three different null models which maintain species richness (this is equal to the Net Relatedness Index, NRI).

Usage

```
mpd.query(tree, matrix, standardize = FALSE,
          null.model="uniform", abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`matrix`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error.
`standardize`	Specifies whether the function should standardize the MPD for variation in species richness. For each tip set S, the observed MPD is standardized by subtracting the mean MPD and dividing by the standard deviation of this measure. The mean and standard deviation are calculated among all tip sets that have the

same number of elements as set S, the tip set whose value we want to standardize (default = FALSE).

`null.model` A character vector (string) that defines which null model is used for computing the standardized values of the measure. There are three possible null models that can be used for computing the standardized values: these are "uniform", "frequency.by.richness", and "sequential". All these models maintain species richness. More specifically, the available models are defined as follows:

- **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness.
- **"frequency.by.richness"** is an abundance-weighted model where species samples are chosen in a manner similar to the following process; first, each species is selected independently with probability proportional to its abundance. If the resulting sample consists of exactly the same number of elements as the input assemblage then it is used by the null model, otherwise it is tossed and the whole process is repeated.
- **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other two models (which are computed analytically), this model uses Monte-Carlo randomization.

This argument is optional, and its default value is set to "uniform".

`abundance.weights` A vector of positive numeric values. These are the abundance weights that will be used if either of the options "frequency.by.richness" or "sequential" are selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.

`reps` An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if any of the other two null models is selected.

`seed` A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if any of the other two null models is selected.

Value

A vector which stores the computed (standardized) MPD values. The i-th entry in this vector stores the (standardized) MPD value of the i-th row in the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and D. Cheliotis. 2012. Efficient computation of popular phylogenetic tree measures. Algorithms in Bioinformatics, LNCS 7534: 30-43.

Webb, C.O. 2000. Exploring the phylogenetic structure of ecological communities: An example for rain forest trees. The American Naturalist 156: 145-155.

See Also

mpd.moments

Examples

```
#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

#Calculate mpd values for each community
mpd.query(bird.families,comm)

#Calculate standardized versions under the uniform model
mpd.query(bird.families,comm,TRUE)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

#Use query function to calculate standardized versions under the sequential model
mpd.query(bird.families,comm,TRUE,null.model="sequential",
          abundance.weights=weights, reps=1000)
```

---

pd.moments *Computes the moments of the Phylogenetic Diversity measure*

Description

Calculates the mean and standard deviation of the unrooted Phylogenetic Diversity (PD) for a tree and a vector of tip set sizes. The means and deviations can be calculated under three different null models which maintain species richness.

Usage

```
pd.moments(tree, sample.sizes, comp.expectation = TRUE, comp.deviation = TRUE,
           null.model="uniform", abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`sample.sizes`	A vector of non-negative integers specifying the tip set sizes for which to calculate moments
`comp.expectation`	Specifies whether the function should compute the mean (default = TRUE)
`comp.deviation`	Specifies whether the function should compute the standard deviation (default = TRUE)
`null.model`	A character vector (string) that defines which null model is used for computing the moments of the measure. There are three possible null models that can be used for computing the moments: these are "uniform", "frequency.by.richness", and "sequential". All these models maintain species richness. More specifically, the available models are defined as follows: • **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness. • **"frequency.by.richness"** is an abundance-weighted model where species samples are chosen in a manner similar to the following process; first, each species is selected independently with probability proportional to its abundance. If the resulting sample consists of exactly the same number of elements as the input assemblage then it is used by the null model, otherwise it is tossed and the whole process is repeated. • **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other two models (which are computed analytically), this model uses Monte-Carlo randomization. This argument is optional, and its default value is set to "uniform".
`abundance.weights`	A vector of positive numeric values. These are the abundance weights that will be used if either of the options "frequency.by.richness" or "sequential" are selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.
`reps`	An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if any of the other two null models is selected.
`seed`	A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if any of the other two null models is selected.

Value

If both comp.expectation and comp.deviation are TRUE, the function returns a two-column matrix with one row per element in sample.sizes, where the first column stores the mean PD and the second column stores the standard deviation for this sample size. If only one of comp.expectation or comp.deviation are TRUE, the function returns a vector with the corresponding values instead.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Faith, D.P. 1992. Conservation evaluation and phylogenetic diversity. Biological Conservation 61: 1-10.

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

See Also

pd.query

Examples

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

# Calculate mean and variance under the uniform model
pd.moments(bird.families,1:100)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

# Calculate mean and variance under the sequential model
pd.moments(bird.families,1:100,
           null.model="sequential", abundance.weights=weights, reps=1000)
```

---

pd.pvalues *Computes the p-values of the unrooted Phylogenetic Diversity measure*

Description

Calculates the p-values of the unrooted phylogenetic diversity (PD) measure for sets of tips on a phylogeny. The p-values can be calculated under two different null models which maintain species richness.

Usage

```
pd.pvalues(tree, matrix, null.model="uniform",
           abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`matrix`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error.
`null.model`	A character vector (string) that defines which null model is used for computing the p-values of the measure. There are two possible null models that can be used for computing the p-values: these are "uniform" and "sequential". Both models maintain species richness. More specifically, the available models are defined as follows: • **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness. • **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other model (which is computed analytically), this model uses Monte-Carlo randomization. This argument is optional, and its default value is set to "uniform".
`abundance.weights`	A vector of positive numeric values. These are the abundance weights that will be used if option "sequential" is selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.
`reps`	An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if the other null model is selected.
`seed`	A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if the other null model is selected.

Value

A vector which stores the computed PD p-values. The i-th entry in this vector stores the PD p-value of the i-th row in the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Faith, D.P. 1992. Conservation evaluation and phylogenetic diversity. Biological Conservation 61: 1-10.

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and D. Cheliotis. 2012. Efficient computation of popular phylogenetic tree measures. Algorithms in Bioinformatics, LNCS 7534: 30-43.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

See Also

pd.moments

Examples

```
#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

#Calculate p-values under the uniform model
pd.pvalues(bird.families,comm, reps=1000)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

#Use query function to calculate standardized versions under the sequential model
pd.pvalues(bird.families,comm,null.model="sequential",
           abundance.weights=weights, reps=1000)
```

---

pd.query *Computes the (standardized) value of the unrooted Phylogenetic Diversity measure*

Description

Calculates the unrooted phylogenetic diversity (PD) measure for sets of tips on a phylogeny. The same function can also calculate the standardized value of this measure under three different null models which maintain species richness (this is equal to the Phylogenetic Diversity Index, PDI).

Usage

```
pd.query(tree, matrix, standardize = FALSE,
         null.model="uniform", abundance.weights, reps=1000, seed)
```

Arguments

`tree`	A phylo tree object
`matrix`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error.
`standardize`	Specifies whether the function should standardize the PD for variation in species richness. For each tip set S, the observed PD is standardized by subtracting the mean PD and dividing by the standard deviation of this measure. The mean and standard deviation are calculated among all tip sets that have the same number of elements as set S, the tip set whose value we want to standardize (default = FALSE).
`null.model`	A character vector (string) that defines which null model is used for computing the standardized values of the measure. There are three possible null models that can be used for computing the standardized values: these are "uniform", "frequency.by.richness", and "sequential". All these models maintain species richness. More specifically, the available models are defined as follows: • **"uniform"** considers samples with equal (uniform) probability among all possible tip samples of the same richness. • **"frequency.by.richness"** is an abundance-weighted model where species samples are chosen in a manner similar to the following process; first, each species is selected independently with probability proportional to its abundance. If the resulting sample consists of exactly the same number of elements as the input assemblage then it is used by the null model, otherwise it is tossed and the whole process is repeated. • **"sequential"** is an abundance-weighted null model where species samples are chosen based on the same method as R's *sample* function. Unlike the other two models (which are computed analytically), this model uses Monte-Carlo randomization. This argument is optional, and its default value is set to "uniform".
`abundance.weights`	A vector of positive numeric values. These are the abundance weights that will be used if either of the options "frequency.by.richness" or "sequential" are selected. The names stored at the vector must match the names of the tips in the tree. This argument is redundant if the "uniform" model is selected.
`reps`	An integer that defines the number of Monte-Carlo random repetitions that will be performed when using the "sequential" model. This argument is redundant if any of the other two null models is selected.
`seed`	A positive integer that defines the random seed used in the Monte-Carlo randomizations of the "sequential" model. This argument is optional, and becomes redundant if any of the other two null models is selected.

Value

A vector which stores the computed (standardized) PD values. The i-th entry in this vector stores the (standardized) PD value of the i-th row in the input matrix.

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com)

References

Faith, D.P. 1992. Conservation evaluation and phylogenetic diversity. Biological Conservation 61: 1-10.

Tsirogiannis, C. and B. Sandel. 2015. PhyloMeasures: A package for computing phylogenetic biodiversity measures and their statistical moments. Ecography, doi: 10.1111/ecog.01814, 2015.

Tsirogiannis, C., B. Sandel and D. Cheliotis. 2012. Efficient computation of popular phylogenetic tree measures. Algorithms in Bioinformatics, LNCS 7534: 30-43.

Tsirogiannis, C., B. Sandel and A. Kalvisa. 2014. New algorithms for computing phylogenetic biodiversity. Algorithms in Bioinformatics, LNCS 8701: 187-203.

See Also

pd.moments

Examples

```
#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

#Use query function to calculate pd values for each community
pd.query(bird.families,comm)

#Use query function to calculate standardized versions under the uniform model
pd.query(bird.families,comm,TRUE)

# Create random abundance weights
weights = runif(length(bird.families$tip.label))
names(weights) = bird.families$tip.label

#Use query function to calculate standardized versions under the sequential model
pd.query(bird.families,comm,TRUE,null.model="sequential",
         abundance.weights=weights, reps=1000)
```

`phylosor.query` *Computes the value of the Phylogenetic Sorensen's Similarity measure*

Description

Calculates the Phylogenetic Sorensen's Similarity (PhyloSor) given paired sets of tips on a phylogeny. A version of this function that computes the standardised value of the measure is not yet available.

Usage

```
phylosor.query(tree, matrix.a, matrix.b = NULL,
               query.matrix = NULL)
```

Arguments

`tree`	A phylo tree object
`matrix.a`	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error
`matrix.b`	Optional, a second matrix with a similar format as matrix.a
`query.matrix`	Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the PhyloSor values. Each row in query.matrix indicates a pair of tip sets for which we want to compute the PhyloSor value. Let k and r be the values that are stored in the i-th row of query.matrix, where k is the value stored in the first column and r is the value stored in the second column. If matrix.b is given, the function computes the PhyloSor value between the k-th row of matrix.a and the r-th row of matrix.b. If matrix.b is not given, the function computes the PhyloSor value between the k-th and r-th row of matrix.a (default = NULL)

Details

Queries can be given in four ways. If neither matrix.b nor query.matrix are given, the function computes the PhyloSor values for all pairs of rows (tip sets) in matrix.a . If matrix.b is given but not query.matrix, the function computes the PhyloSor values for all combinations of a row in matrix.a with rows in matrix.b. If query.matrix is given and matrix.b is not, the function returns the PhyloSor values for the pairs of rows in matrix.a specified by query.matrix. If query.matrix and matrix.b are both given, PhyloSor values are computed for the rows in matrix.a specified by the first column of query.matrix against the rows in matrix.b specified in the second column of query.matrix.

Value

The PhyloSor values for the requested pairs of tip sets. If query.matrix is provided, then the values are returned in an one-dimensional vector. The i-th element of this vector is the PhyloSor value for the pair of tip sets indicated in the i-th row of query.matrix. If query.matrix is not provided, the PhyloSor values are returned in a matrix object; entry [i,j] in the output matrix stores the PhyloSor value between the tip sets specified on the i-th and j-th row of matrix.a (if matrix.b is not specified), or the PhyloSor value between the i-th row of matrix.a and the j-th row of matrix.b (if matrix.b is specified)

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com) and Brody Sandel (bsandel@scu.edu)

References

Graham, C.H. and P.V.A. Fine. 2008. Phylogenetic beta diversity: linking ecological and evolutionary processes across space and time. Ecology Letters 11: 1265:1277.

Swenson, N.G. 2011. Phylogenetic beta diversity metrics, trait evolution and inferring functional beta diversity of communities. PLoS ONE: 6: e21264.

Examples

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

#Create 10 random communities with 50 families each
comm = matrix(0,nrow = 10,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

#Calculate all pairwise PhyloSor values for communities in comm
phylosor.query(bird.families,comm)

#Calculate pairwise distances from
#the first two rows of comm to all rows
phylosor.query(bird.families, comm[1:2,],comm)

#Calculate the distances from the first two rows
#to all rows using the query matrix
qm = expand.grid(1:2,1:10)
phylosor.query(bird.families,comm,query.matrix = qm)
```

---

unifrac.query *Computes the value of the Unique Fraction measure*

Description

Calculates the Unique Fraction (UniFrac) given paired sets of tips on a phylogeny. A version of this function that computes the standardised value of the measure is not yet available.

Usage

```
unifrac.query(tree, matrix.a, matrix.b = NULL,
              query.matrix = NULL)
```

Arguments

tree	A phylo tree object
matrix.a	A matrix with binary (0/1) values, where each row represents a tip set. Each column name in the matrix must match a tip label on the input tree. If not all values in the matrix are binary, we consider two cases; if the matrix contains only non-negative values, all values are coerced to binary ones and a warning message is printed. If the matrix contains at least one negative value, the function throws an error
matrix.b	Optional, a second matrix with a similar format as matrix.a
query.matrix	Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the UniFrac values. Each row in query.matrix indicates a pair of tip sets for which we want to compute the UniFrac value. Let k and r be the values that are stored in the i-th row of query.matrix, where k is the value stored in the first column and r is the value stored in the second column. If matrix.b is given, the function computes the UniFrac value between the k-th row of matrix.a and the r-th row of matrix.b. If matrix.b is not given, the function computes the UniFrac value between the k-th and r-th row of matrix.a (default = NULL)

Details

Queries can be given in four ways. If neither matrix.b nor query.matrix are given, the function computes the UniFrac values for all pairs of rows (tip sets) in matrix.a . If matrix.b is given but not query.matrix, the function computes the UniFrac values for all combinations of a row in matrix.a with rows in matrix.b. If query.matrix is given and matrix.b is not, the function returns the UniFrac values for the pairs of rows in matrix.a specified by query.matrix. If query.matrix and matrix.b are both given, UniFrac values are computed for the rows in matrix.a specified by the first column of query.matrix against the rows in matrix.b specified in the second column of query.matrix.

Value

The UniFrac values for the requested pairs of tip sets. If query.matrix is provided, then the values are returned in an one-dimensional vector. The i-th element of this vector is the UniFrac value for the pair of tip sets indicated in the i-th row of query.matrix. If query.matrix is not provided, the UniFrac values are returned in a matrix object; entry [i,j] in the output matrix stores the UniFrac value between the tip sets specified on the i-th and j-th row of matrix.a (if matrix.b is not specified), or the UniFrac value between the i-th row of matrix.a and the j-th row of matrix.b (if matrix.b is specified)

Author(s)

Constantinos Tsirogiannis (tsirogiannis.c@gmail.com) and Brody Sandel (bsandel@scu.edu)

References

Lozupone C. and R. Knight. 2005. UniFrac: a New Phylogenetic Method for Comparing Microbial Communities. Applied and Environmental Microbiology,71(12):8228-35.

Examples

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

#Create 10 random communities with 50 families each
comm = matrix(0,nrow = 10,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

#Calculate all pairwise UniFrac values for communities in comm
unifrac.query(bird.families,comm)

#Calculate pairwise distances from
#the first two rows of comm to all rows
unifrac.query(bird.families, comm[1:2,],comm)

#Calculate the distances from the first two rows
#to all rows using the query matrix
qm = expand.grid(1:2,1:10)
unifrac.query(bird.families,comm,query.matrix = qm)
```

Index