

Package ‘PhyloMeasures’

April 8, 2015

Type Package

Title Fast and Exact Algorithms for Computing Phylogenetic Biodiversity Measures

Version 1.1

Date 2015-03-26

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

Maintainer Constantinos Tsirogiannis <analekta@gmail.com>

Description Given a phylogenetic tree T and a sample 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 the implementations of efficient algorithms for several such measures. Most importantly, the package includes algorithms for computing exactly and efficiently the expectation and the variance of these measures, which are essential for null model comparisons. Efficient computation of richness-standardized versions of these metrics, such as the net relatedness index (NRI), nearest taxon index (NTI), phylogenetic diversity index (PDI), and the corresponding indices of two-sample measures are also available in this package. 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.

License GPL-3

Imports ape

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-04-08 00:39:07

R topics documented:

PhyloMeasures-package 2

cac.moments	4
cac.query	5
cbl.moments	7
cbl.query	8
cd.moments	10
cd.query	11
cdnt.averaged.query	13
cdnt.directed.query	15
cdnt.query	18
mntd.moments	20
mntd.query	21
mpd.moments	22
mpd.query	24
pd.moments	25
pd.query	26
phylosor.query	28
unifrac.query	29

Index	32
--------------	-----------

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

Description

The package provides functions for computing the value and the statistical moments 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: 1.1
 Date: 2015-03-20
 License: GPL-3

The package contains two types of functions; functions that compute the value of a phylogenetic biodiversity measure, and functions that compute the statistical moments of a measure. We refer to the functions of the first type as query functions, and we refer to the functions of the latter type as moment functions.

The package provides one query function for each supported phylogenetic measure. The only ex-

ception is the CDNT measure, for which three query functions are provided, each handling a different version of this measure. For single-sample measures (these are the PD, the MPD, the MNTD, and the CAC), the corresponding query function receives a tree object (built using package *ape*), one matrix which represents the tip sets that we want to process, and a logical value that specifies whether for each input sample we want to compute the original value of a measure, or its standardised value. The standardised value of a measure *M* for a given tip set *S* is equal to the original value of *M* for *S* minus the mean value of *M*, and this divided by the standard deviation of *M* for these sets. The mean and the standard deviation values that are used here are computed among all subsets of tips in the input tree that have exactly the same number of elements as *S*. Query functions of two-sample measures may receive an additional matrix, and also arguments that specify which pairs of input samples should be processed. The query functions are: `pd.query`, `mpd.query`, `mntd.query`, `cac.query`, `cbl.query`, `cd.query`, `cdnt.query`, `cdnt.directed.query`, `cdnt.averaged.query`, `phylosor.query`, and `unifrac.query`.

The package also provides one moment function for each supported phylogenetic measure, except the Community Distance Nearest Taxon (CDNT), the Phylogenetic Sorensen's Similarity (PhyloSor), and the Unique Fraction (UniFrac). Each moment function receives a tree object (built using package *ape*), a vector with non-negative integers that indicates the samples sizes for which we want to compute the statistical moments of the measure, and further arguments that specify the statistical moments that we want to compute for each sample size. Especially for the CAC, the corresponding moment function can compute the *k* order moment of this measure for any positive integer *k*. For the rest of the measures, the corresponding moment functions can compute the expectation and the standard deviation for a given sample size. The moment functions provided in the package are: `pd.moments`, `mpd.moments`, `mntd.moments`, `cac.moments`, `cbl.moments`, and `cd.moments`.

Author(s)

Constantinos Tsirogiannis and Brody Sandel

Maintainer: Constantinos Tsirogiannis <analekta@gmail.com>

References

- Faith, D.P. 1992. Conservation evaluation and phylogenetic diversity. *Biological Conservation* 61: 1-10.
- 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.
- Lozupone C. and R. Knight. 2005. UniFrac: a New Phylogenetic Method for Comparing Microbial Communities. *Applied and Environmental Microbiology*, 71(12):8228-35.
- 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.
- 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
pd.query(bird.families,comm,TRUE)
```

cac.moments

Computes the statistical moments of the Core Ancestor Cost measure

Description

Calculates the first k statistical moments of the Core Ancestor Cost measure for a tree, a vector of tip set sizes, and a value of chi. Contrary to other measures, this function makes it possible to compute moments which are of higher order than the mean and the variance. 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.

Usage

```
cac.moments(tree, chi, sample.sizes, k=2)
```

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)

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 centralised statistical moment of order j. Given a phylogenetic tree T, a sample size r and a value chi, the centralised statistical moment of order j for the CAC is defined as:

$$E_{R \in Sub(T,r)}[(CAC_{\chi}(T, R) - m_{CAC}(T, r))^k],$$

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, and $m_{CAC}(T, r)$ indicates the mean value of the CAC over all these sets.

Author(s)

Constantinos Tsirogiannis (constant@madalgo.au.dk)

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")

cac.moments(bird.families,0.75,1:100,k=4)
```

cac.query

Computes the (standardised) 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 standardised value of this measure for the given tip sets.

Usage

```
cac.query(tree, matrix, chi, is.standardised = FALSE)
```

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]
is.standardised	Specifies whether the function should standardise the CAC value according to the variation in species richness. For each input tip set R, the CAC value for R is standardised 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)

Value

A vector which stores the computed (standardised) 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 (constant@madalgo.au.dk)

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.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 standardised versions
cac.query(bird.families,comm,0.8,TRUE)
```

cbl.moments	<i>Computes the moments of the Common Branch Length measure</i>
-------------	---

Description

Calculates the mean and standard deviation of the Common Branch Length (CBL) on a given tree for several pairs of tip set sizes.

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 <code>sample.sizes</code> . 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 (constant@madalgo.au.dk)

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	<i>Computes the (standardised) value of the Common Branch Length measure</i>
-----------	--

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 standardised value of this measure for the given tip sets.

Usage

```
cbl.query(tree, matrix.a, matrix.b = NULL,
          query.matrix = NULL, is.standardised = 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)

`is.standardised`

Specifies whether the function should return the standardised value of the CBL for each sample pair. The value is standardised 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 (constant@madalgo.au.dk)

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 standardised versions
cbl.query(bird.families,comm,is.standardised = TRUE)

```

cd.moments

Computes the moments of the Community Distance measure

Description

Calculates the mean and standard deviation of the Community Distance (CD) on a given tree for several pairs of tip set sizes.

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 (constant@madalgo.au.dk)

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 (standardised) 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 standardised value of this measure for the given tip sets.

Usage

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

Arguments

<code>tree</code>	A phylo tree object
<code>matrix.a</code>	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
<code>matrix.b</code>	Optional, a second matrix with a similar format as <code>matrix.a</code>
<code>query.matrix</code>	Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the CD values. Each row in <code>query.matrix</code> 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 <code>query.matrix</code> , where k is the value stored in the first column and r is the value stored in the second column. If <code>matrix.b</code> is given, the function computes the CD value between the k -th row of <code>matrix.a</code> and the r -th row of <code>matrix.b</code> . If <code>matrix.b</code> is not given, the function computes the CD value between the k -th and r -th row of <code>matrix.a</code> (default = NULL)
<code>is.standardised</code>	Specifies whether the function should return the standardised value of the CD for each sample pair. The value is standardised 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 (constant@madalgo.au.dk)

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 standardised versions
cd.query(bird.families,comm,is.standardised = TRUE)
```

cdnt.averaged.query	<i>Computes the value of the averaged Community Distance Nearest Taxon measure</i>
---------------------	--

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:

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

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 (constant@madalgo.au.dk)

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 <i>k</i> and <i>r</i> be the values that are stored in the <i>i</i> -th row of query.matrix, where <i>k</i> is the value stored in the first column and <i>r</i> 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 <i>k</i> -th row of matrix.a to the tip set in the <i>r</i> -th row of matrix.b, and the dCDNT value from the tip set in the <i>r</i> -th row of matrix.b to the tip set in the <i>k</i> -th row of matrix.a. If matrix.b is not given, the two values computed are the dCDNT from the <i>k</i> -th to the <i>r</i> -th set of matrix.a, and the dCDNT from the <i>r</i> -th to the <i>k</i> -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 first 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 (constant@madalgo.au.dk)

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	<i>Computes the value of the maximised Community Distance Nearest Taxon measure</i>
------------	---

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 (constant@madalgo.au.dk)

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. Note: this function returns a result only if the input tree is ultrametric.

Usage

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

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 returns the mean of the MNTD (default = TRUE)
comp.deviation	Specifies whether the function returns the standard deviation of the MNTD (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`, 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 (constant@madalgo.au.dk)

References

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")

mntd.moments(bird.families,1:100)
```

mntd.query	<i>Computes the (standardised) value of the Mean Nearest Taxon Distance measure</i>
------------	---

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 for the given tip sets (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`, then this function returns a result only if the input tree is ultrametric.

Usage

```
mntd.query(tree, matrix, is.standardised = FALSE)
```

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.
is.standardised	Specifies whether the function returns the standardized value of the MNTD, which reflects the variation in species richness. For each sample R, the MNTD value is standardised by subtracting the mean MNTD and dividing by the standard deviation. The mean and standard deviation are calculated among all tip sets that have the same number of elements as set R, the tip set whose value we want to standardise (default = <code>FALSE</code>)

Value

A vector with the (standardised) MNTD values for each row in the matrix

Author(s)

Constantinos Tsirogiannis (constant@madalgo.au.dk)

References

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 standardised versions
mntd.query(bird.families,comm,TRUE)
```

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.

Usage

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

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)

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 of the MPD 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 (constant@madalgo.au.dk)

References

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.query](#)

Examples

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

mpd.moments(bird.families,1:100)
```

mpd.query	<i>Computes the (standardised) value of the Mean Pairwise Distance measure</i>
-----------	--

Description

Calculates the value of the Mean Pairwise Distance (MPD) measure for sets of tips on a phylogeny. The same function can also calculate the standardised value of this measure for the given tip sets (this is equal to minus one times the so-called Net Relatedness Index, NRI).

Usage

```
mpd.query(tree, matrix, is.standardised = FALSE)
```

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.
is.standardised	Specifies whether the function should standardize the MPD for variation in species richness. For each tip set R, the observed MPD is standardised by subtracting the mean of the 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 R, the tip set whose value we want to standardise (default = FALSE)

Value

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

Author(s)

Constantinos Tsirogiannis (constant@madalgo.au.dk)

References

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 standardised versions
mpd.query(bird.families,comm,TRUE)
```

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.

Usage

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

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)

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 (constant@madalgo.au.dk)

References

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

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.query](#)

Examples

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

pd.moments(bird.families,1:100)
```

pd.query	<i>Computes the (standardised) value of the unrooted Phylogenetic Diversity measure</i>
----------	---

Description

Calculates the unrooted phylogenetic diversity (PD) measure for sets of tips on a phylogeny. The same function can also calculate the standardised value of this measure for the given tip sets (this is equal to the Phylogenetic Diversity Index, PDI).

Usage

```
pd.query(tree, matrix, is.standardised = FALSE)
```

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.

is.standardised

Specifies whether the function should standardize the PD for variation in species richness. For each tip set R, the observed PD is standardised 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 R, the tip set whose value we want to standardise (default = FALSE)

Value

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

Author(s)

Constantinos Tsirogiannis (constant@madalgo.au.dk)

References

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

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 pd values for each community
pd.query(bird.families,comm)

#Calculate standardised versions
pd.query(bird.families,comm,TRUE)
```

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)

Brody Sandel (brody.sandel@bios.au.dk) and Constantinos Tsirogiannis (constant@madalgo.au.dk)

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

<code>tree</code>	A phylo tree object
<code>matrix.a</code>	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
<code>matrix.b</code>	Optional, a second matrix with a similar format as <code>matrix.a</code>
<code>query.matrix</code>	Optional, a two-column matrix specifying the pairs of rows (tip sets) for which the function computes the UniFrac values. Each row in <code>query.matrix</code> indicates a pair of tip sets for which we want to compute the UniFrac value. Let <code>k</code> and <code>r</code> be the values that are stored in the <code>i</code> -th row of <code>query.matrix</code> , where <code>k</code> is the value stored in the first column and <code>r</code> is the value stored in the second column. If <code>matrix.b</code> is given, the function computes the UniFrac value between the <code>k</code> -th row of <code>matrix.a</code> and the <code>r</code> -th row of <code>matrix.b</code> . If <code>matrix.b</code> is not given, the function computes the UniFrac value between the <code>k</code> -th and <code>r</code> -th row of <code>matrix.a</code> (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)

Brody Sandel (brody.sandel@bios.au.dk) and Constantinos Tsirogiannis (constant@madalgo.au.dk)

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

*Topic **biodiversity**

PhyloMeasures-package, 2

*Topic **distance**

PhyloMeasures-package, 2

*Topic **phylogeny**

PhyloMeasures-package, 2

cac.moments, 4, 6

cac.query, 5, 5

cbl.moments, 7, 9

cbl.query, 8, 8

cd.moments, 10, 13

cd.query, 11, 11

cdnt.averaged.query, 13, 17, 19

cdnt.directed.query, 15, 15, 19

cdnt.query, 15, 17, 18

mntd.moments, 20, 22

mntd.query, 21, 21

mpd.moments, 22, 25

mpd.query, 23, 24

pd.moments, 25, 27

pd.query, 26, 26

PhyloMeasures (PhyloMeasures-package), 2

PhyloMeasures-package, 2

phylosor.query, 28

unifrac.query, 29