Package 'EnvNJ'

September 27, 2021

Type Package

Title Whole Genome Phylogenies Using Sequence Environments

Version 0.1.3

Description Contains utilities for the analysis of protein sequences in a phylogenetic context. Allows the generation of phylogenetic trees base on protein sequences in an alignment-independent way.
Two different methods have been implemented. One approach is based on the frequency analysis of n-grams,
previously described in Stuart et al. (2002) <doi:10.1093/bioinformatics/18.1.100>. The other approach is based on the species-specific neighborhood preference around amino acids. Features include the conversion of a protein set into a vector
reflecting these neighborhood preferences, pairwise distances (dissimilarity) between these vectors,
and the generation of trees based on these distance matrices.

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 4.0.0)

Imports ape, bio3d, graphics, phangorn, philentropy, seqinr, stringr

Suggests testthat

NeedsCompilation no

Author Juan Carlos Aledo [aut, cre] (<https://orcid.org/0000-0002-3497-9945>)

Maintainer Juan Carlos Aledo <caledo@uma.es>

Repository CRAN

Date/Publication 2021-09-27 10:50:02 UTC

R topics documented:

`aa.at` *Residue Found at the Requested Position*

Description

Returns the residue found at the requested position.

Usage

```
aa.at(at, target)
```

Arguments

`at`	the position in the primary structure of the protein.
`target`	a character string specifying the sequence of the protein of interest.

Value

Returns a single character representing the residue found at the indicated position in the indicated protein.

See Also

aa.comp()

Examples

```
aa.at(2, 'MFSQQQRCVE')
```

`aa.comp` *Amino Acid Composition*

Description

Returns a table with the amino acid composition of the target protein.

Usage

```
aa.comp(target, uniprot = TRUE)
```

Arguments

`target`	a character string specifying the UniProt ID of the protein of interest or, alternatively, the sequence of that protein.
`uniprot`	logical, if TRUE the argument 'target' should be an ID.

Value

Returns a dataframe with the absolute frequency of each type of residue found in the target peptide.

Examples

```
aa.comp('MPSSVSWGILLLAGLCCLVPVSLAEDPQGDAAQK', uniprot = FALSE)
```

aaf *Compute the Frequency of Each Amino Acid in Each Species*

Description

Computes the frequency of each amino acid in each species.

Usage

```
aaf(data)
```

Arguments

data	input data must be a dataframe (see details).

Details

Input data must be a dataframe where each row corresponds to an individual protein, and each column identifies a species. Therefore, the columns' names of this dataframe must be coherent with the names of the OTUs being analyzed.

Value

A dataframe providing amino acid frequencies en the set of species. Rows correspond amino acids and columns to species.

See Also

env.sp(), otu.vector(), otu.space()

Examples

```
aaf(bovids)
```

bovids *13 orthologous mtDNA-encoded proteins of 11 bovine species.*

Description

A dataset containing the sequences of mtDNA-encoded proteins from different bovids

Usage

```
bovids
```

Format

A data frame with 13 rows (one per protein) and 11 variables (one per species)

Source

https://www.ncbi.nlm.nih.gov/genome/organelle/

`cos2dis` *Convert Cosines Between Vectors into Pairwise Dissimilarities*

Description

Converts cosines values into dissimilarities values.

Usage

```
cos2dis(cos)
```

Arguments

`cos`	a square upper triangular matrix where cos(i,j) is the cosine between the vector i and j.

Details

Cosines are standard measure of vector similarity, and can be converted into distance by dij = -log((1 + cos(i,j))/2).

Value

A triangular matrix with the distances.

See Also

vcos()

Examples

```
data(bovids)
vectors = otu.space(bovids[, 7:11])
cosData = vcos(vectors)
disData = cos2dis(cosData)
```

d.phy2df *Convert a Phylip Distance Matrix into a DataFrame*

Description

Converts a phylip distance matrix into a either an R dataFrame or matrix.

Usage

```
d.phy2df(phyfile, as = 'matrix')
```

Arguments

phyfile	path to the file containing the distances in phylip format.
as	class of the output R data. It should be either 'dataframe' or 'matrix'.

Value

Either a dataframe or a matrix containing the distances read from the phy file.

See Also

d.df2pny()

Examples

```
## Not run: d.phy2df(phyfile = "./data_t/d_dummy.txt")
```

df2fasta *Convert Dataframe into Fasta File*

Description

Converts a dataframe into a fasta file.

Usage

```
df2fasta(df, out.file)
```

Arguments

df	a named (both rows and cols) dataframe (see details).
out.file	path and name of output file.

Details

The format of the df should be as follows. Each row represents a protein sequence and each column a species.

Value

A fasta file that is saved in the specified path.

See Also

fastaconc()

Examples

```
## Not run: df2fasta(df = bovids, out.file = "./example.fasta")
```

---

env.extract *Sequence Environment Around a Given Position*

Description

Extracts the sequence environment around a given position.

Usage

```
env.extract(seq, c, r)
```

Arguments

seq	a string protein sequence.
c	center of the environment.
r	radius of the environment.

Value

Returns a a strings with the extracted environment sequence.

References

Aledo et al. Sci Rep. 2015; 5: 16955. (PMID: 26597773)

See Also

env.matrices(), env.sp()

Examples

```
env.extract('ARGGQQQCATSYV', c = 8,  r = 2)
```

`env.fasta` *Build Trees Based on the Environment Around the Indicated Amino Acid(s)*

Description

Builds trees based on the environment around the indicated amino acid(s).

Usage

```
env.fasta(file, r = 10, aa = 'all', out.file = 'any')
```

Arguments

`file`	path to the single multispecies fasta file to be used as input.
`r`	a positive integer indicating the radius of the sequence segment considered as environment.
`aa`	the amino acid(s) to be used to encoded the species.
`out.file`	path and name of output file. Only if intermediate results data want to be saved (see details).

Details

This function builds alignment-independent phylogenetic trees. The input data is a fasta file. When an out.file path is provided, the environment sequences of each species and the vector representing each species are saved in the path provided.

Value

A list with two objects, the first one is an inter-species distance matrix. The second one is an object of class 'phylo'.

See Also

envnj(), fastaconc()

Examples

```
## Not run: env.fasta(file = "./data_t/sample5.fasta")
```

env.matrices *Environment Matrices*

Description

Provides the frequencies of each amino acid within the environment.

Usage

```
env.matrices(env)
```

Arguments

env	a character string vector containing the environments.

Value

Returns a list of two dataframes. The first, shown the environment in matrix form. The second provides the frequencies of each amino acid within the environments.

References

Aledo et al. Sci Rep. 2015; 5: 16955. (PMID: 26597773)

See Also

env.extract(), otu.vector()

Examples

```
env.matrices(c('ANQRmCTPQ', 'LYPPmQTPC', 'XXGSmSGXX'))
```

env.sp *Extract the Sequence Environments*

Description

Extracts the sequence environments around the selected amino acid(s) in the chosen species.

Usage

```
env.sp(data, sp, r = 10, aa = 'all', silent = TRUE)
```

Arguments

data	input data must be a dataframe (see details).
sp	the species of interest (it should be named as in the input dataframe).
r	a positive integer indicating the radius of the sequence segment considered as environment.
aa	the amino acid(s) which environments are going to be extracted.
silent	logical. When FALSE the program progress is reported to alleviate loneliness.

Details

Input data must be a dataframe where each row corresponds to an individual protein. The columns contain the sequence of the protein corresponding to the row in each species. Therefore, the columns' names of this dataframe must be coherent with the names of the OTUs being analyzed.

Value

A list of environment sequences. Each element from the list is a vector with the environment sequences around an amino acid. So, the length list is the same as the length of aa.

See Also

otu.vector(), otu.space()

Examples

```
data(bovids)
env.sp(data = bovids, sp = "Bos_taurus", r = 2)
```

---

envfascpp *Convert Fasta Files into Environment Vectors*

Description

Converts fasta files into environment vectors

Usage

```
envfascpp(path, r = 10, exefile, outfile)
```

Arguments

path	path to the folder that contain a the file 'list.txt', which contains the names of all the fasta files to be analyzed (one per line). The fasta files must be in the same path.
r	a positive integer indicating the radius of the sequence segment considered as environment.
exefile	path to the directory containing the envfas executable.
outfile	path to, and name of, output file.

Details

This function builds and write vectors representing the species' proteome. To use this function, you must first download the source code envfas.cpp (https://bitbucket.org/jcaledo/envnj/src/master/AncillaryCode/envfas.cpp) and compile it.

Value

A txt file per fasta file analyzed. Each txt file can be read as a vector. Thus the number of lines gives the dimension of the vector.

See Also

envnj(), fastaconc()

Examples

```
## Not run: envfastacpp("./data_t/list.txt", 10, ".", "./MyResults")
```

`envnj` *Build Trees Based on the Environment Around the Indicated Amino Acid(s)*

Description

Builds trees based on the environment around the indicated amino acid(s).

Usage

```
envnj(data, r = 10, aa = 'all', metric = "cosine", clustering = "nj", outgroup = 'any')
```

Arguments

`data`	input data must be a dataframe where each row corresponds to a protein sequence and each column to a species.
`r`	a positive integer indicating the radius of the sequence segment considered as environment.
`aa`	the amino acid(s) to be used to encoded the species.
`metric`	character string indicating the metric (see metrics() to see the methods allowed).
`clustering`	string indicating the clustering method, either "nj" or "upgma".
`outgroup`	when a rooted tree is desired, it indicates the species to be used as outgroup.

Details

This function builds alignment-independent phylogenetic trees.

Value

A list with two objects, the first one is an inter-species distance matrix. The second one is an object of class 'phylo'.

See Also

otu.space(), metrics()

Examples

```
data(bovids)
envnj(bovids[, 7:11], aa = "all", outgroup = "Pseudoryx_nghetinhensis")
```

---

fastaconc *Concatenate Fasta Files in a Single Multispecies Fasta File*

Description

Concatenate fasta files from different species in a single multispecies fasta file.

Usage

```
fastaconc(otus, inputdir = ".", out.file = "./concatenated_multispecies.fasta")
```

Arguments

otus	a character vector giving the otus' names.
inputdir	path to the directory containing the individual fasta files.
out.file	path and name of output file.

Details

When we have fasta files (extension should be '.fasta'), each one for a species containing different sequences of the given species, this function concatenate the different sequences of the same species and writes it as a single sequence in a single multispecies fasta file. If the individual fasta files are found in the working directory, the inputdir argument don't need to be passed. The names of the individual fasta files must match the otus' names.

Value

A single multispecies fasta file with the sequences of each species spliced in a single sequence.

See Also

df2fasta()

Examples

```
## Not run: fastaconc(otus = c('Glis_glis', 'Ovis_aries', 'Sus_scrofa'))
```

`metrics` *Pairwise Vector Dissimilarities*

Description

Computes the dissimilarity between n-dimensional vectors.

Usage

```
metrics(vset, method = 'euclidean', p = 2)
```

Arguments

`vset`	matrix (n x m) where each column is a n-dimensional vector.
`method`	a character string indicating the distance/dissimilarity method to be used (see details).
`p`	power of the Minkowski distance. This parameter is only relevant if the method 'minkowski' has been selected.

Details

Although many of the offered methods compute a proper distance, that is not always the case. For instance, for a non null vector, v, the 'cosine' method gives d(v, 2v) = 0, violating the coincidence axiom. For that reason we prefer to use the term dissimilarity instead of distance. The methods offered can be grouped into families.

L_p family::

('euclidean', 'manhattan', 'minkowski', 'chebyshev')

Euclidean = sqrt(sum | P_i - Q_i |^2)

Manhattan = sum | P_i - Q_i |

Minkowski = (sum| P_i - Q_i |^p)^1/p

Chebyshev = max | P_i - Q_i |

L_1 family::

('sorensen', 'soergel', 'lorentzian', 'kulczynski', 'canberra')

Sorensen = sum | P_i - Q_i | / sum (P_i + Q_i)

Soergel = sum | P_i - Q_i | / sum max(P_i , Q_i)

Lorentzian = sum ln(1 + | P_i - Q_i |)

Kulczynski = sum | P_i - Q_i | / sum min(P_i , Q_i)

Canberra = sum | P_i - Q_i | / (P_i + Q_i)

Intersection family::
('non-intersection', 'wavehedges', 'czekanowski', 'motyka')
Non-intersection = 1 - sum min(P_i , Q_i)
Wave-Hedges = sum | P_i - Q_i | / max(P_i , Q_i)
Czekanowski = sum | P_i - Q_i | / sum | P_i + Q_i |
Motyka = sum max(P_i , Q_i) / sum (P_i , Q_i)

Inner product family::
('cosine', 'jaccard')
Cosine = - ln(0.5 (1 + (P_i Q_i) / sqrt(sum P_i^2) sqrt(sum Q_i^2)))
Jaccard = 1 - sum (P_i Q_i) / (sum P_i^2 + sum Q_i^2 - sum (P_i Q_i))

Squared-chord family::
('bhattacharyya', 'squared_chord')
Bhattacharyya = - ln sum sqrt(P_i Q_i)
Squared-chord = sum (sqrt(P_i) - sqrt(Q_i))^2

Squared Chi family::
('squared_chi')
Squared-Chi = sum ((P_i - Q_i)^2 / (P_i + Q_i))

Shannon's entropy family::
('kullback-leibler', 'jeffreys', 'jensen-shannon', 'jensen_difference')
Kullback-Leibler = sum P_i * log(P_i / Q_i)
Jeffreys = sum (P_i - Q_i) * log(P_i / Q_i)
Jensen-Shannon = 0.5(sum P_i ln(2P_i / (P_i + Q_i)) + sum Q_i ln(2Q_i / (P_i + Q_i)))
Jensen difference = sum (0.5(P_i log(P_i) + Q_i log(Q_i)) - 0.5(P_i + Q_i) ln(0.5(P_i + Q_i))

Mismatch family::
('hamming', 'mismatch', 'mismatchZero', 'binary')
Hamming = (# coordinates where P_i != Q_i) / n
Mismatch = # coordinates where that P_i != Q_i
MismatchZero = Same as mismatch but after removing the coordinates where both vectors have zero.
Binary = (# coordinates where a vector has 0 and the other has a non-zero value) / n.

Combinations family::
('taneja', 'kumar-johnson', 'avg')
Taneja = sum (P_i + Q_i / 2) log(P_i + Q_i / (2 sqrt(P_i * Q_i)))
Kumar-Johnson = sum (P_i^2 - Q_i^2)^2 / 2 (P_i Q_i)^1.5
Avg = 0.5 (sum | P_i - Q_i| + max | P_i - Q_i |)

Value

A matrix with the computed dissimilarity values.

References

Sung-Hyuk Cha (2007). International Journal of Mathematical Models and Methods in Applied Sciences. Issue 4, vol. 1

Luczac et al. (2019). Briefings in Bioinformatics 20: 1222-1237.

https://r-snippets.readthedocs.io/en/latest/real_analysis/metrics.html

See Also

vcos(), vdis()

Examples

```
metrics(matrix(1:9, ncol =3), 'cosine')
```

`msa.merge` *Carry Out a MSA of a Set of Different Orthologous Proteins*

Description

Carries out a MSA of a set of different orthologous proteins in different species.

Usage

```
msa.merge(data, outfile = 'any')
```

Arguments

`data`	input data must be a dataframe where each row corresponds to a protein and each column to a species.
`outfile`	path to the place where a fasta file is going to be saved. If 'any', no file is saved.

Details

The input data has the same format that the input data used for EnvNJ or SVD-n-Gram methods. Thus, the name of columns must correspond to that of species.

Value

A dataframe containing the MSA (species x position).

See Also

msa.tree

Examples

```
## Not run:
data(bovids)
msa.merge(bovids)
## End(Not run)
```

`msa.tree` *Infer a tree based on a MSA*

Description

Infers a tree base on a MSA.

Usage

```
msa.tree(data, outgroup = 'any')
```

Arguments

`data`	input data must be a dataframe where each row corresponds to a protein and each column to a species.
`outgroup`	when a rooted tree is desired, indicate the species to be used as outgroup.

Details

The input data has the same format that the input data used for EnvNJ or SVD-n-Gram methods. Thus, the name of columns must correspond to that of species.

Value

A list containing the (i) MSA, (ii) the distance matrix and (iii) the tree.

See Also

msa.merge

Examples

```
## Not run:
data(bovids)
msa.tree(bovids)
## End(Not run)
```

ncd *Compute Normalized Compression Distances*

Description

Computes normalized compression distances.

Usage

```
ncd(seq1, seq2)
```

Arguments

seq1	character string indicating the path to the first fasta file to be analyzed.
seq2	character string indicating the path to the second fasta file to be analyzed.

Details

The two fasta files must be in the working directory. This function use zpaq to compress files. Thus, the zpaq software must be installed on your system and in the search path for executables if you wish to use this function. NCD = (Z(xy) - min(Z(x), Z(y))) / max(Z(x), Z(y)) Where Z(x), Z(y) and Z(xy) are the lengths of the compressed versions of seq1, seq2 and the concatenated sequences 1 and 2, respectively.

Value

A non-negative real value reflecting the dissimilarity between seq1 and seq2.

See Also

ncdnj()

Examples

```
try(ncd(seq1 = "./A.fasta", seq2 = "./B.fasta"))
```

ncdnj *Compute a Distance Matrix Using Normalized Compression Distance*

Description

Computes a distance matrix using normalized compression distance.

Usage

```
ncdnj(wd)
```

Arguments

wd	character string indicating the path to the directory where the input files can be found (see details).

Details

The input files, which must be found at the wd provided, consist of a file named 'list.txt' containing the names of the fasta files to be analyzed (one per line). The referred fasta files also must be found at the provided wd. This function use zpaq to compress files. Thus, the zpaq software must be installed on your system and in the search path for executables if you wish to use this function.

Value

A list where the first element is a symmetric distance matrix and the second one is a phylogenetic tree build using NJ.

See Also

ncd()

Examples

```
try(ncdnj("./data_t"))
```

ngram *Compute n-Gram Frequencies Vector*

Description

Computes the n-gram frequencies vector for a given protein.

Usage

```
ngram(prot, k = 4)
```

Arguments

prot	a character string corresponding to the primary structure of the protein.
k	a positive integer, between 1 and 5, indicating the k-mer of the words to be counted.

Details

The one letter code for amino acids is used (capital).

Value

A dataframe with two columns, the first one given the peptides and the second one the corresponding absolute frequency.

References

Stuart et al. Bioinformatics 2002; 18:100-108.

See Also

ngraMatrix(), ffp(), svdgram()

Examples

```
ngram(bovids$Bos_taurus[1], k = 3)
```

---

ngraMatrix *Compute n-Gram Frequencies Dataframe*

Description

Computes the n-gram frequencies dataframe for the protein and species provides.

Usage

```
ngraMatrix(data, k = 4, silent = FALSE)
```

Arguments

data	a dataframe with as many columns as species and one row per orthologous protein. The rows and columns must be named accordingly.
k	a positive integer, between 1 and 5, indicating the k-mer of the words to be counted.
silent	logical, set to FALSE to avoid loneliness.

Details

The argument prot can be obtained using orth() and orth.seq().

Value

A list with two dataframes. The first one with nsp * npr columns (nsp: number of species, npr: number of proteins per species) and npe rows (npe: number of peptides, 20 for n = 1, 400 for n = 2, 8000 for n = 3 and 160000 for n = 4). The entries of the dataframe are the number of times that the indicated peptide has been counted in the given protein. Orthologous proteins are in consecutive columns, thus the first nsp columns are the orthologous of protein 1 and so on. The second dataframe contains the Species Vector Sums (each vector describes one species).

References

Stuart et al. Bioinformatics 2002; 18:100-108.

See Also

ngram(), svdgram()

Examples

```
ngraMatrix(bovids[,1:3], k = 2)
```

`otu.space` *Compute the Matrix Representing the Species Vector Subspace*

Description

Computes the matrix representing the species vector subspace.

Usage

```
otu.space(data, r = 10, aa = "all", silent = TRUE)
```

Arguments

`data`	input data must be a dataframe (see details).
`r`	a positive integer indicating the radius of the sequence segment considered as environment.
`aa`	the amino acid(s) to be used to encoded the species.
`silent`	logical. If FALSE, the running progress is reported.

Details

Input data must be a dataframe where each row corresponds to an individual protein, and each column identifies a species. Therefore, the columns' names of this dataframe must be coherent with the names of the OTUs being analyzed. The dimension of the vector representing each species will depend on the settings. For instance, if we choose a single amino acid and a radius of 10 for the sequence environment, then we will get a vector of dimension 400 (20 amino acids x 20 positions). If we opt for the 20 amino acids and r = 10, then the vector will be of dimension 8000 (400 for each amino acid * 20 amino acids). Please, note that r is selected in the function env.sp() that will provide the input dataframe for the current function.

Value

A matrix representing the species vector subspace.

See Also

env.sp(), otu.vector()

Examples

```
data(bovids)
otu.space(bovids[, 1:5], r = 2)
```

---

otu.vector *Convert a Set of Sequence Environments into a Vector*

Description

Converts a set of sequence environments into a vector.

Usage

```
otu.vector(envl, sp = "", aa = "all", silent = TRUE)
```

Arguments

envl	a list containing the sequence environment of a species (as the one returned by the function env.sp()).
sp	character string indicating the species being analyzed.
aa	the amino acid(s) to be used to encoded the species.
silent	logical. When FALSE the program progress is reported to alleviate loneliness.

Details

The dimension of the vector representing the species will depend on the settings. For instance, if we choose a single amino acid and a radius of 10 for the sequence environment, then we will get a vector of dimension 400 (20 amino acids x 20 positions). If we opt for the 20 amino acids and r = 10, then the vector will be of dimension 8000 (400 for each amino acid * 20 amino acids). Please, note that r is selected in the function env.sp() that will provide the input dataframe for the current function.

Value

A matrix representing the species. This matrix can be converted into a vector representing the target species just typing as.vector(matrix). Each coordinate is the frequency of a given amino acid at a certain position from the environment (see details).

See Also

env.sp(), otu.space()

Examples

```
data(bovids)
cow = env.sp(bovids, "Bos_taurus")
otu.vector(cow)
```

`reyes` *13 orthologous mtDNA-encoded proteins of 34 mammalian species.*

Description

A dataset containing the sequences of mtDNA-encoded proteins from different mammalian species (Reyes et al. 1996, Mol.Biol. Evol. 17:979)

Usage

```
reyes
```

Format

A data frame with 13 rows (one per protein) and 34 variables (one per species)

Source

https://www.ncbi.nlm.nih.gov/genome/organelle/

`svdgram` *Compute Phylogenetic Trees Using an n-Gram and SVD Approach*

Description

Computes phylogenetic trees using an n-gram and SVD approach.

Usage

```
svdgram(matrix, rank, species, SVS = TRUE)
```

Arguments

matrix	either a dataframe or a matrix where each row represents a property of a protein (for instance, the frequencies of tetrapeptides) and each column represents a different protein (or species).
rank	a numeric array providing the ranks that want to be used to approach the data matrix using SVD.
species	character array providing the species' names.
SVS	logical. When the matrix passed as argument correspond to the peptide-protein matrix and SVS is set to TRUE, then the function will compute a matrix where the columns are the Species Vector Sums. Alternatively, if the matrix passed as argument is already a matrix where the columns encode for species, SVS should be set to FALSE.

Details

When the matrix passed as argument is a matrix of peptide-protein, the function implement the method described by Stuart et al. 2002 (see references).

Value

An object of class multiPhylo containing a tree for each rank value required.

References

Stuart et al. Bioinformatics 2002; 18:100-108.

See Also

ngraMatrix()

Examples

```
a <- ngraMatrix(bovids[, 1:4], k = 2)[[2]][, -1]
species <- names(a)
svdgram(matrix = a, rank = 4, species = species, SVS = FALSE)
```

---

vcos *Compute Pairwise Cosines of the Angles Between Vectors*

Description

Computes pairwise cosines of the angles between vectors.

Usage

```
vcos(vectors, silent = TRUE, digits = 3)
```

Arguments

vectors	a named list (or dataframe) containing n-dimensional vectors.
silent	logical, set to FALSE to avoid loneliness.
digits	integer indicating the number of decimal places.

Details

Cosines are standard measure of vector similarity. If the angle between two vectors in n-dimensional space is small, then the individual elements of their vectors must be very similar to each other in value, and the calculated cosine derived from these values is near one. If the vectors point in opposite directions, then the individual elements of their vectors must be very dissimilar in value, an the calculated cosine is near minus one.

Value

A triangular matrix with the cosines of the angles formed between the given vectors.

See Also

vdis()

Examples

```
vcos(otu.space(bovids[, 1:4]))
```

---

vdis *Compute Pairwise Distances Between Vectors*

Description

Computes pairwise distances between vectors.

Usage

```
vdis(cos)
```

Arguments

cos	a square upper triangular matrix where cos(i,j) is the cosine between the vector i and j.

Details

Cosines are standard measure of vector similarity, and can be converted into distance by dij = -log((1 + cos(i,j))/2).

Value

A triangular matrix with the distances.

See Also

vcos()

Examples

```
data(bovids)
vectors = otu.space(bovids[, 7:11])
cosData = vcos(vectors)
disData = suppressWarnings(vdis(cosData))
```

`vect2tree` *Convert a Set of Vectors into a Tree*

Description

Converts a set of vectors into a tree.

Usage

```
vect2tree(path, metric = "cosine", clustering = "nj")
```

Arguments

`path`	path to the working directory. This directory must contain a txt file per vector and an additional txt file named vlist.txt that provides the names (one per line) of the vector txt files.
`metric`	character string indicating the metric (see metrics() to see the methods allowed).
`clustering`	string indicating the clustering method, either "nj" or "upgma".

Details

This function computes the distance matrix and builds the corresponding tree.

Value

a list with two elements: a distance matrix and a tree.

See Also

envnj(), fastaconc(), envfascpp()

Examples

```
## Not run: vec2tree("./data_t")
```

vtree *Build a Tree When Species Are Encoded by n-Dim Vectors*

Description

Builds a tree when species are encoded by n-dim vectors.

Usage

```
vtree(matrix, outgroup = 'any')
```

Arguments

matrix	either a dataframe or matrix where each column represents an OTU.
outgroup	when a rooted tree is desired, it indicates the species to be used as outgroup.

Details

The method is based on a distance matrix obtained after converting the cos between vector (similarity measurement) in a dissimilarity measurement.

Value

A list with two objects, the first one is an inter-species distance matrix. The second one is an object of class 'phylo'.

See Also

svdgram

Examples

```
data(bovids)
mymatrix <- ngraMatrix(bovids[, 6:11], k = 2)[[2]][, 2:7]
vtree(mymatrix, outgroup = "Pseudoryx_nghetinhensis")
```

Index