Package ‘phybase’

October 5, 2018

Type Package

Title Basic functions for phylogenetic analysis

Version 2.0

Date 2017-9-25

Author Liang Liu

Depends R (>= 3.2.0), ape (>= 3.2), Matrix

Maintainer Liang Liu <lliu@uga.edu>

Description This package provides functions to read, write, manipulate, estimate, and summarize phylogenetic trees including species trees which contain not only the topology and branch lengths but also population sizes. The input/output functions can read tree files in which trees are presented in parenthetic format. The trees are read in as a string and then transformed to a matrix which describes the relationship of nodes and branch lengths. The nodes matrix provides an easy access for developers to further manipulate the tree, while the tree string provides interface with other phylogenetic R packages such as ``ape". The input/output functions can also be used to change the format of tree files between NEXUS and PHYLIP. Some basic functions have already been established in the package for manipulating trees such as deleting and swapping nodes, rooting and unrooting trees, changing the root of the tree. The package also includes functions such as ``consensus", ``coaltime, "popsize" for summarizing phylogenetic trees, calculating the coalescence time, population size, and tree distance. The function maxtree is built in the package to esimtate the species tree from multiple gene trees.

License GPL (>= 2)

LazyData true

NeedsCompilation yes

R topics documented:

phybase-package 3
alignment.mle.remove 4
alignment.reference.remove 4
alignment.summary 5
alignment.trim 5
bootstrap 6
bootstrap.mulgene 7
control.mpest 8
dat.coaltree 8

dat.finch 8
dat.modelcode 9
dat.unrootedtree 9
del.brlens 9
del.comments 10
del.node 10
dist.dna 11
dist.internode 12
dist.species 12
dist.taxa 13
file.beast2phylip 14
file.concatData 14
file.fasta2phylip 15
file.nexus2phylip 15
file.phylip2nexus 16
file.separateGeneData 16
is.clock 17
is.rootedtree 18
loglike.coal 18
loglike.triple 19
node.height 20
parse.modeltest 21
parse.phyml 21
parse.raxml 22
read.dna.seq 22
read.tree.nodes 23
read.tree.string 24
run.modeltest 25
run.mpest 25
run.seqgen 26
sim.coal.mpest 26
sim.coaltree 27
sim.coaltree.sp 28
sim.coaltree.sp.mu 29
sim.dna 30
sim.SeqfromSp 31
site.pattern 32
site.summary 32
species.name 33
sptree.alleletable 33
sptree.maxtree 34
sptree.njst 35
sptree.star 36
sptree.steac 37
test.2sptree 38
test.equalgenetree 38
test.hybrid 39
test.submodel.valid.phyml 39
test.submodel.valid.raxml 40
tree.brlens 40
tree.consensus 41
tree.distance 41

tree.name2node 42
tree.noclock2clock 43
tree.node2name 43
tree.partition 44
tree.plot 45
tree.probdist 46
tree.subtree 46
tree.upgma 47
write.dna.seq 48
write.seq.phylip 49
write.subtree 49
write.tree.string 50

Index **52**

`phybase-package` *Basic functions for Phylogenetic trees*

Description

This package provides functions to read, write, manipulate, simulate, estimate, and summarize phylogenetic trees including species trees which contains not only the topology and branch lengths but also population sizes. The input/output functions can read tree files in which trees are presented in parenthetic format. The trees are read in as a string and then transformed to a matrix which describes the relationship of nodes and branch lengths. The nodes matrix provides an easy access for developers to further manipulate the tree, while the tree string provides interface with other phylogenetic R packages such as "ape". The input/output functions can also be used to change the format of tree files between NEXUS and PHYLIP. Some basic functions have already been established in the package for manipulating trees such as deleting and swapping nodes, rooting and unrooting trees, changing the root of the tree. The package includes functions such as "consensus", "coaltime, "popsize", "treedist" for summarizing phylogenetic trees, calculating the coalescence time, population size, and tree distance. The function maxtree, star.sptree, and steac.sptree are built in the package to esimtate the species tree from multiple gene trees. The packages offers function to simulate DNA sequences from gene trees under substitution models.

Details

Package:	PhyBase
Type:	Package
Version:	2.0
Date:	2017-09-26
License:	GPL (>=2.0.0)

Author(s)

Liang Liu

Maintainer: Liang Liu <lliu@uga.edu>

`alignment.mle.remove` *alignment.mle.remove sequences*

Description

This function removes the sequences whose branch lengths in the ML tree is 5 times greater than the branch lengths in the reference tree, i.e., the concatenation tree. The new alignments are saved in the files .final.

Usage

```
alignment.mle.remove(path_raxml = "raxmlHPC", seqfiles, contreefile)
```

Arguments

`path_raxml`	the full path of program raxml
`seqfiles`	the input sequence files
`contreefile`	the concatenation tree file

Author(s)

Liang Liu

`alignment.reference.remove` *alignment.reference.remove removes sequences*

Description

This function removes sequences whose branch lengths in the fitted reference tree is 5 times greater than the corresponding branch lengths in the reference tree. The new alignments are saved in the files .removed.

Usage

```
alignment.reference.remove(path_raxml = "raxmlHPC", seqfiles, nconcatgene)
```

Arguments

`path_raxml`	the full path of program raxml
`seqfiles`	the input sequence files
`nconcatgene`	the number of genes used for building the concatenation tree

Author(s)

Liang Liu

`alignment.summary` *summary statistics of sequences*

Description

This function calculates the summary statistics of the alignments across genes.

Usage

```
alignment.summary(seqfile)
```

Arguments

`seqfile`	the input sequence files; one gene per file

Author(s)

Liang Liu

`alignment.trim` *alignment.trim sequences*

Description

This function trims gappy regions in the alignment.

Usage

```
alignment.trim(path_trimal, inputfolder, outputfolder)
```

Arguments

`path_trimal`	the full path of program Trimal
`inputfolder`	inputfolder contains the sequence files of original alignments in phylip format; one gene per file
`outputfolder`	outputfolder contains the sequence files of trimmed alignments

Author(s)

Liang Liu

bootstrap *Bootstrap sequences*

Description

This function can be used to bootstrap sequences.

Usage

```
bootstrap(sequence)
```

Arguments

sequence	sequence matrix.

Details

In the sequences matrix, the columns are "Taxa" and the rows are "sites". The function will bootstrap the rows.

Value

the function returns a sequence matrix with sites randomly sampled from the original matrix with replacement.

Author(s)

Liang Liu

Examples

```
#construct the DNA sequences of three taxa
seq <- matrix("A",ncol=4,nrow=3)
rownames(seq)<-c("taxa1","taxa2","taxa3")
seq[,2]<-"G"
seq[,3]<-"C"
seq[,4]<-"T"
bootstrap_sample = bootstrap(seq)

data(dat.finch)
bootstrap_sample <- bootstrap(dat.finch$seq)
```

`bootstrap.mulgene` *Bootstrap sequences from multiple loci*

Description

The function bootstraps sequence columns for each locus sampled from the original multilocus data. It consists of two step. First, it bootstraps loci. Then it bootstraps sequences for each locus.

Usage

```
bootstrap.mulgene(sequence,gene,name,boot,outfile="")
```

Arguments

`sequence`	data matrix
`gene`	location of each locus
`name`	taxa names of sequences
`boot`	the number of bootstrap samples
`outfile`	output file

Details

In the sequences matrix, the rows are "Taxa" and the columns are "sites".

Value

The function generates a data file in phylip format.

Author(s)

Liang Liu <`lliu@uga.edu`>

See Also

`bootstrap`

Examples

```
#construct the DNA sequences of three taxa
seq <- matrix("A",ncol=4,nrow=3)
rownames(seq)<-c("taxa1","taxa2","taxa3")
seq[,2]<-"G"
seq[,3]<-"C"
seq[,4]<-"T"

name<-rownames(seq) #taxa names of the sequences

#construct two loci. The first two nucleotides represent the first locus,
#while nucleotide 3 and 4 represent the second locus.
gene<-matrix(0,ncol=2,nrow=2)
gene[1,]<-c(1,2)
gene[2,]<-c(3,4)
```

```
gene
bootstrap.mulgene(seq,gene,name,boot=2,outfile="bootdata.txt")
```

`control.mpest` *generate a control file for mpest*

Description

This function can generate a control file for mpest

Usage

```
control.mpest(genetreefile, ngene, randomseed=-1, nrun, speciesnames, outputfile)
```

Arguments

`genetreefile`	the gene tree file
`ngene`	the number of genes
`randomseed`	the default is -1; otherwise, a random seed will be generated
`nrun`	the number of runs; each run has a different starting point, and mp-est will find the tree with the maximum likelihood score across all runs
`speciesnames`	the names of species
`outputfile`	the name of the control file

Author(s)

Liang Liu

`dat.coaltree` *species tree and gene trees*

Description

An example of gene trees generated from the coalescent model given the species tree

Author(s)

Liang Liu <`lliu@uga.edu`>

`dat.finch` *multilocus sequence data*

Description

An example of multilocus sequence data. It includes DNA sequences from 30 genes for 4 species.

Author(s)

Liang Liu <`lliu@uga.edu`>

`dat.modelcode` *substitution model code*

Description

The number of free parametres and the code of 22 substitution models used in phyml

Author(s)

Liang Liu <`lliu@uga.edu`>

`dat.unrootedtree` *unrooted trees*

Description

An example of unrooted trees

Author(s)

Liang Liu <`lliu@uga.edu`>

`del.brlens` *Delete branch lengths from trees*

Description

This function deletes branch lengths from trees.

Usage

```
del.brlens(tree)
```

Arguments

`tree`	trees in the newick format

Author(s)

Liang Liu

Examples

```
treestr<-"((((H:0.00402,C:0.00402):0.00304,G:0.00707):0.00929,O:0.01635):0.1,W:0.12);"
del.brlens(treestr)
```

del.comments *Delete comments*

Description

This function deletes comments in the data file.

Usage

```
del.comments(X)
```

Arguments

X	a vector of strings as the data file is read using scan

Author(s)

Liang Liu

Examples

```
treestr<-"((((H:0.00402,C:0.00402):0.00304[#0.01],G:0.00707):0.00929,O:0.01635):0.1,W:0.12);"
del.comments(treestr)
```

del.node *Delete a node from the tree*

Description

This function deletes a node (and its descendant nodes) from the tree.

Usage

```
del.node(inode, name, nodematrix)
```

Arguments

inode	the node to be deleted
name	the species names
nodematrix	the tree node matrix

Details

The species names are those defined in the original tree before deleting the node `inode`. No need to delete the species name of `inode`! If `inode` is an internode, the whole subtree below `inode` will be deleted.

Value

nodes	the tree node matrix after deleting `inode`
treestr	the tree string of the tree after deleting `inode`.

Author(s)

Liang Liu

Examples

```
treestr<-"((((H:0.00402,C:0.00402):0.00304,G:0.00707):0.00929,O:0.01635):0.1,W:0.12);"
spname<-read.tree.nodes(treestr)$names
nodematrix<-read.tree.nodes(treestr,spname)$nodes
del.node(6,spname,nodematrix)

##unrooted tree
data(dat.unrootedtree)
name<-paste("S",1:29,sep="")
nodematrix<-read.tree.nodes(dat.unrootedtree[1])$nodes
del.node(6,name,nodematrix)
```

`dist.dna` *Calculate pairwise distances among DNA sequences*

Description

Calculate pairwise distances among DNA sequences. The sites with missing characters are excluded.

Usage

```
dist.dna(sequences, nst = 0)
```

Arguments

`sequences`	DNA sequences
`nst`	substitution model. 0:no model, 1:JC

Details

If nst=0, the distance is equal to the proportion of sites with different nucleotides.

Value

The function returns a distance matrix.

Author(s)

Liang Liu <`lliu@uga.edu`>

References

Jukes, TH and Cantor, CR. 1969. Evolution of protein molecules. Pp. 21-123 in H. N. Munro, ed. Mammalian protein metabolism. Academic Press, New York.

See Also

`tree.upgma`

Examples

```
data(dat.finch)
dist.dna(dat.finch$seq,nst=1)
```

`dist.internode` *find the distance of two taxa*

Description

This function calculates the distance of two sequences on the basis of number of ancestors between two sequences.

Usage

```
dist.internode(tree, taxaname)
```

Arguments

`tree`	a tree in the Newick format
`taxaname`	taxa names

Author(s)

Liang Liu

Examples

```
treestr<-"((((H:0.1,C:0.1):0.1,G:0.1):0.1,O:0.1):0.1,W:0.1);"
taxaname<-species.name(treestr)
dist.internode(treestr, taxaname)
```

`dist.species` *Calculate pairwise distances among species*

Description

If some species have multiple taxa, the pairwise distance between two species is equal to the average of the distances between all pairs of taxa in the two species. This functions returns the pairwise distances among species (average over all taxa in the species).

Usage

```
dist.species(dist, species.structure)
```

Arguments

`dist`	the distance matrix of taxa
`species.structure`	a matrix with rows representing species and columns representing taxa. 1: the species (row) has the taxon at the corresponding column. see the example.

Value

This functions returns the distance matrix of species.

Author(s)

Liang Liu

See Also

See Also as `dist.taxa`

Examples

```
treestr<-"((((H:0.00402,C:0.00402):0.00304,G:0.00705):0.00929,O:0.01635):0.1,W:0.11635);"
nodematrix<-read.tree.nodes(treestr)$nodes
dist<-dist.taxa(nodematrix,5)
species.structure<-matrix(0,nrow=2,ncol=5) #2 species and 5 taxa
species.structure[1,]<-c(1,1,1,0,0)        #taxa 1,2,3 belong to the first species
species.structure[2,]<-c(0,0,0,1,1)        #taxa 4,5 belong to the second species
dist.species(dist,species.structure)
```

`dist.taxa` *Calculate all pairwise distances among taxa in the tree*

Description

The function computes all pairwise distances among taxa in the tree.

Usage

```
dist.taxa(nodematrix, nspecies)
```

Arguments

`nodematrix`	the tree node matrix
`nspecies`	the number of taxa in the tree

Value

The function returns a distance matrix.

Author(s)

Liang Liu <`lliu@uga.edu`>

Examples

```
treestr<-"((((H:0.00402,C:0.00402):0.00304,G:0.00705):0.00929,O:0.01635):0.1,W:0.11635);"
nodematrix<-read.tree.nodes(treestr)$nodes
dist.taxa(nodematrix,5)
```

`file.beast2phylip` *convert beast files to phylip files*

Description

This function converts beast xml files to phylip files

Usage

```
file.beast2phylip(beastfile)
```

Arguments

`beastfile`	the beast xml file

Author(s)

Liang Liu

`file.concatData` *concatenate sequences from multiple files*

Description

This function concatenates sequences from multiple files.

Usage

```
file.concatData(inputfiles, confile)
```

Arguments

`inputfiles`	the sequence files to be concatenated
`confile`	the name of the output file to which the concatenated seuqences are saved

Author(s)

Liang Liu

`file.fasta2phylip` *convert fasta files to phylip files*

Description

This function converts fasta files to phylip files

Usage

```
file.fasta2phylip(inputfolder, outputfolder)
```

Arguments

`inputfolder`	the folder of the fasta files
`outputfolder`	the folder of the phylip files

Author(s)

Liang Liu

`file.nexus2phylip` *convert nexus files to phylip files*

Description

This function converts nexus files to phylip files

Usage

```
file.nexus2phylip(inputfolder, outputfolder)
```

Arguments

`inputfolder`	the folder of the fasta files
`outputfolder`	the folder of the phylip files

Author(s)

Liang Liu

`file.phylip2nexus` *convert phylip files to nexus files*

Description

This function converts phylip files to nexus files

Usage

```
file.phylip2nexus(inputfolder, outputfolder)
```

Arguments

`inputfolder`	the folder of the fasta files
`outputfolder`	the folder of the phylip files

Author(s)

Liang Liu

`file.separateGeneData` *separate the concatenated sequences into individual genes*

Description

This function separates the concatenated sequences into individual genes.

Usage

```
file.separateGeneData(nexusfile, missing=c("?","-","N","n"))
```

Arguments

`nexusfile`	sequence data file in nexus format with the character block for individual genes
`missing`	missing characters

Author(s)

Liang Liu

`is.clock` *Is a clock tree or not*

Description

This function checks the tree to see if the branch lengths satisfy the molecular clock assumption. For each node, the lengths of the left lineage and right lineage are compared. If they are not equal to each other and the difference is greater than `threshold`, the function will return FALSE. This function does not perform statistical test for the molecular clock assumption.

Usage

```
is.clock(nodematrix, nspecies,threshold)
```

Arguments

`nodematrix`	the tree node matrix
`nspecies`	the number of species
`threshold`	the critical value for the difference between the length of the left decendant lineage and that of the right decendant lieage of an internode. The difference below the threshold is treated as no difference.

Value

The function returns TRUE for a clock tree and FALSE for a non-clock tree.

Author(s)

Liang Liu <`lliu@uga.edu`>

See Also

`is.rootedtree`

Examples

```
treestr<-"((((H:0.00402,C:0.00402):0.00304,G:0.00705):0.00929,O:0.01635):0.1,W:0.11635);"
nodematrix<-read.tree.nodes(treestr)$nodes

##if the threshold is set to be large, the tree is a clock tree
is.clock(nodematrix,5,0.0001)
##[1] TRUE

##if the threshold is a small number, the tree is not a clock tree.
is.clock(nodematrix,5,0.00001)
##[1] FALSE
```

`is.rootedtree` *Is the tree rooted or not*

Description

This function can test if the tree is rooted.

Usage

```
is.rootedtree(tree)
```

Arguments

`tree`	tree string or tree node matrix

Value

The function returns TRUE if the tree is a rooted tree. Otherwise, it returns FALSE.

Author(s)

Liang Liu <`lliu@uga.edu`>

See Also

`is.clock`

Examples

```
data(dat.unrootedtree)
is.rootedtree(dat.unrootedtree[1])

data(dat.coaltree)
is.rootedtree(dat.coaltree$sptree)
```

`loglike.coal` *loglikelihood of the species tree, i.e., Rannala and Yang formula*

Description

This function calculates the loglikelihood of a species tree from a set of gene trees using the Rannala and Yang formula

Usage

```
loglike.coal(gtree, sptree, taxaname,spname,species.structure,strict=T)
```

Arguments

`gtree`	a collection of gene trees
`sptree`	a species tree in newick format
`taxaname`	the names of taxa
`spname`	the names of species
`species.structure`	define which sequence belong to which species
`strict`	whether or not to check the result

Value

The function returns the log likelihood score.

Author(s)

Liang Liu

References

Rannala, B. and Z. Yang. 2003. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. Genetics 164: 1645-1656.

Examples

```
gtree<-"(((A:1,B:1):3,C:4):2,D:6);"
stree<-"(((A:0.5,B:0.5):1#0.1,C:1.5):1#0.1,D:2.5)#0.1;"
taxaname<-c("A","B","C","D")
spname<-taxaname
ntax<-length(taxaname)
nspecies<-length(spname)
species.structure<-matrix(0,nrow=nspecies,ncol=ntax)
diag(species.structure)<-1
loglike.coal(gtree,stree,taxaname,spname,species.structure)
```

`loglike.triple` *Loglikehood of Triples*

Description

The function calculates the loglikelihood for DNA sequences (snip data)

Usage

```
loglike.triple(sptree,spname,dna)
```

Arguments

`sptree`	species tree
`spname`	species names
`dna`	dna sequences

Details

This function is used to calculate the loglikelihood of triples.

Value

The function returns the loglikehood of triples.

Author(s)

Liang Liu <lliu@uga.edu>

See Also

write.subtree, read.tree.string

---

node.height *Calculate node height*

Description

The function calculates the height of a node. The tree is assumed to be an ultramatric tree.

Usage

```
node.height(inode, nodematrix, nspecies)
```

Arguments

inode	the node number
nodematrix	the tree node matrix
nspecies	the number of species in the tree

Value

The function returns the height of inode.

Author(s)

Liang Liu <lliu@uga.edu>

Examples

```
tree.string<-"(((H:4.2,C:4.2):3.1,G:7.3):6.3,O:13.5);"
nodematrix<-read.tree.nodes(tree.string)$nodes
node.height(6,nodematrix,4)
```

`parse.modeltest` *parse modeltest outputs*

Description

The function finds the selected models from the modeltest outputs

Usage

```
parse.modeltest(outputfile)
```

Arguments

`outputfile`	the modeltest output file

Author(s)

Liang Liu <`lliu@uga.edu`>

`parse.phyml` *parse phyml outputs*

Description

The function gets the parameter estimates from the phyml output files.

Usage

```
parse.phyml(phyml_stats_file="phyml_stats.txt")
```

Arguments

`phyml_stats_file`	phyml_stats_file

Author(s)

Liang Liu <`lliu@uga.edu`>

`parse.raxml` *parse raxml outputs*

Description

The function gets the parameter estimates from the raxml output files.

Usage

```
parse.raxml(raxml_info_file="RAxML_info.out")
```

Arguments

`raxml_info_file`	raxml_info_file

Author(s)

Liang Liu <`lliu@uga.edu`>

`read.dna.seq` *Read sequences from files*

Description

The function reads sequences from files in the nexus or phylip format.

Usage

```
read.dna.seq(file="", format="nexus")
```

Arguments

`file`	the input file name
`format`	nexus or phylip

Value

`seq`	sequences
`gene`	partitions on the sequences. Each partition represents a gene or a locus.

Author(s)

Liang Liu

read.tree.nodes *Read tree nodes*

Description

Read a tree string in parenthesic format and output tree nodes, species names and whether the tree is rooted

Usage

```
read.tree.nodes(str, name = "")
```

Arguments

`str`	a tree string in the parenthetical format
`name`	species names

Details

This function reads a tree string into a matrix that describes the relationships among nodes and corresponding branch lengths. Each row in the matrix represents a node. The first n rows contain the information of the nodes at the tips of the tree. The order of the first n nodes is identical to the alphabetic order of the species names given by name. If name is null, the names will be extracted from the tree string and the first n nodes are in the same order as the species names appear in the tree string from left to right.

The numbers after ":" are branch lengths. The numbers after pound signs are population sizes. The numbers after "

Value

`nodes`	nodes is a matrix that describes the relationships among nodes and corresponding branch lengths and population sizes if the tree is a species tree. Each row corresponds a node in the tree. The matrix has 5 columns. The first column is the father of the current node. The following columns are left son, right son, branch length, and population size. The value -9 implies that the information does not exist. The last row is the root of the tree. If the tree is unrooted, the first number of the root node is -8, while it is -9 for a rooted tree.
`names`	species names in the same order of the first n nodes.
`root`	TRUE for a rooted tree, FALSE for an unrooted tree.

Author(s)

Liang Liu <`lliu@uga.edu`>

See Also

`read.tree.string`, `species.name`

Examples

```
##read an unrooted tree
data(dat.unrootedtree)
tree<-read.tree.nodes(dat.unrootedtree[1])
tree$nodes
tree$names
tree$root

#read a rooted tree
data(dat.coaltree)
tree<-read.tree.nodes(dat.coaltree$sptree)
tree$nodes
tree$names
tree$root
```

`read.tree.string` *Read tree strings from a tree file*

Description

This function reads tree strings in Newick format from a tree file. The output of the function is a vector of tree strings that can be converted to a matrix of nodes by the function `read.tree.nodes`.

Usage

```
read.tree.string(file = "", format="nexus")
```

Arguments

`file`	the tree file that contains trees in Newick format.
`format`	format = "nexus" or format = "phylip"

Details

The function can read NEXUS and PHYLIP tree files. It works for other types of tree files as long as the trees in the tree files are in Newick format. This function combining with `write.tree.string` can change the tree file format.

Value

`tree`	a vector of tree strings.
`names`	species names.
`root`	TRUE for rooted trees, FALSE for unrooted trees

Author(s)

Liang Liu <`lliu@uga.edu`>

See Also

write.tree.string, read.tree.nodes

Examples

```
##read rooted trees in PHYLIP format
cat("(((H:4.2,C:4.2):3.1,G:7.3):6.3,O:13.5);",file = "phylip.tre", sep = "\n")
tree.string<-read.tree.string("phylip.tre",format="phylip")
```

---

run.modeltest *run modeltest*

Description

The function runs jmodeltest2

Usage

```
run.modeltest(path_jmodeltest="./jmodeltest2/dist/jModelTest.jar", seqfile, nmodel=3, outputfile)
```

Arguments

`path_jmodeltest`	the path of jmodeltest2
`seqfile`	the input sequence file
`nmodel`	the number of models selected by jmodeltest2
`outputfile`	the result is saved to outputfile

Author(s)

Liang Liu <lliu@uga.edu>

---

run.mpest *run mpest*

Description

The function runs mpest

Usage

```
run.mpest(path_mpest="mpest", genetreefile, species, sptree="", ntree)
```

Arguments

`path_mpest`	the path of mpest
`genetreefile`	the path of the input gene tree file
`species`	species names
`sptree`	the species tree. If the species tree is provided, mp-est will fit the branch lengths for the given tree
`ntree`	number of gene trees

Author(s)

Liang Liu <lliu@uga.edu>

`run.seqgen` *run seq-gen*

Description

The function runs seq-gen

Usage

```
run.seqgen(path_seqgen="./Seq-Gen-1.3.4/seq-gen",nsim=1, seed=123, basefreq=rep(0.25,4), rate=rep
```

Arguments

`path_seqgen`	the path of seqgen
`nsim`	The number of simulations. All simulated data are saved to the same outputfile
`seed`	the random seed
`basefreq`	base frequencies
`rate`	six relative rates in the rate matrix
`seqlength`	the length of simulated sequences
`gamma`	the gamma parameter
`inv`	the proportion of invariant sites
`treefile`	the file of the true tree
`saveformat`	phlip, nexus, or fasta
`outputfile`	the simulated sequences are saved to outputfile

Author(s)

Liang Liu <lliu@uga.edu>

`sim.coal.mpest` *Simulate gene trees from the mpest tree under the coalescent model*

Description

This function can simulate gene trees from the mpest tree.

Usage

```
sim.coal.mpest(mpest_tree,ngenetree)
```

Arguments

`mpest_tree`	the mpest tree
`ngenetree`	number of gene trees

Value

The function returns the simulated gene trees.

Author(s)

Liang Liu <`lliu@uga.edu`>

`sim.coaltree` *Simulate a coalescence tree*

Description

This function can simulate a coalescence tree from a single population with parameter theta. The coalescence times in the tree have exponential distributions. `theta` is equal to 4uNe where Ne is the effective population size and u is the mutation rate.

Usage

```
sim.coaltree(nspecies,theta)
```

Arguments

`nspecies`	the number of species
`theta`	the population parameter

Details

theta is the population parameter theta=4N*mu.

Value

The function returns the simulated coalescence tree.

Author(s)

Liang Liu <`lliu@uga.edu`>

References

John Wakeley, Coalescent theory: An introduction.

See Also

`sim.coaltree.sp`

Examples

```
sim.coaltree(5,theta=0.2)
##[1] "((5:0.55696,(1:0.34858,3:0.34858):0.20838):2.99874,(2:0.97896,4:0.97896):2.57674)"
```

sim.coaltree.sp *simulate a gene tree from the species tree*

Description

The function simulates a gene tree from the species tree using Rannala and Yang's formula

Usage

```
sim.coaltree.sp(rootnode, nodematrix, nspecies, seq, name)
```

Arguments

rootnode	the root node of the species tree
nodematrix	the tree node matrix of the species tree
nspecies	the number of species
seq	a vector of number of sequences in each species
name	species names used in the simulated gene tree. the order of the names must be consistent with that in "nodematrix"

Value

gt	the gene tree generated from the species tree
height	the tree height of the gene tree

Author(s)

Liang Liu <lliu@uga.edu>

References

Rannala, B. and Z. Yang. 2003. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. Genetics 164: 1645-1656.

See Also

sim.coaltree

Examples

```
tree<-"(((H:0.00402#0.01,C:0.00402#0.01):0.00304#0.01,
G:0.00707#0.01):0.00929#0.01,O:0.01635#0.01)#0.01;"
spname<-species.name(tree)
nodematrix<-read.tree.nodes(tree, spname)$nodes
rootnode<-7
##define the vector seq as [2,2,2,2] which means that there are 2 sequences in each species
seq<-rep(2,4)
str<-sim.coaltree.sp(rootnode,nodematrix,4,seq,name=spname)$gt
```

`sim.coaltree.sp.mu` *Simulate a gene tree from the non-clock species tree model*

Description

The function generates a random gene tree from the species tree under the non-clock species tree model.

Usage

```
sim.coaltree.sp.mu(sptree, spname, seq, numgenetree,method="dirichlet",alpha=5.0)
```

Arguments

`sptree`	species tree
`spname`	species names
`seq`	the species-sequences struction, i.e., which sequence belongs to which species
`numgenetree`	the number of gene trees to be generated
`alpha`	the parameter in the gamma distribution. see also `mutation_exp`
`method`	either gamma or dirichlet

Value

`gt`	the simulated gene tree
`st`	the node matrix of the species tree
`seqname`	the names of sequences

Author(s)

Liang Liu

Examples

```
sptree<-"(((A:0.5,B:0.5):1#0.1,C:1.5):1#0.1,D:2.5)#0.1;"
spname<-c("A","B","C","D")
seq<-c(1,1,1,1) #each species has only one sequence.
sim.coaltree.sp.mu(sptree, spname, seq, numgenetree=1,method="dirichlet",alpha=5.0)
```

sim.dna *Simulate DNA sequences from substitution models*

Description

Simulate DNA sequences from a tree using substitution model

Usage

```
sim.dna(nodematrix,seqlength,model,kappa=2,rate=c(1,1,1,1,1,1),
frequency=c(1/4,1/4,1/4,1/4))
```

Arguments

`nodematrix`	the tree node matrix
`seqlength`	sequence length
`model`	1 JC, 2 H2P, 3 HKY, 4 GTR
`kappa`	the transition/transversion ratio
`rate`	the six rates used in GTR model
`frequency`	frequencies of four types of nucleotides

Value

The function returns DNA sequences simulated from the gene tree `nodematrix`. The sequences are coded as 1:A, 2:C, 3:G, 4:T.

Author(s)

Liang Liu <`lliu@uga.edu`>

References

Jukes, TH and Cantor, CR. 1969. Evolution of protein molecules. Pp. 21-123 in H. N. Munro, ed. Mammalian protein metabolism. Academic Press, New York.

See Also

`sim.coaltree`

Examples

```
tree<-"(((H:0.00402,C:0.00402):0.00304,G:0.00707):0.00929,O:0.01635);"
nodematrix<-read.tree.nodes(tree)$nodes
sim.dna(nodematrix,100, model=2, kappa=4)
```

`sim.SeqfromSp` *simulate DNA sequences from a species tree*

Description

The function simulates sequences from a species tree. The function assumes that seq-gen has been installed

Usage

```
sim.SeqfromSp(sptree, spname, ntaxasp, ngene, theta=0, noclock=0,
simsequence=1, murate="Dirichlet",alpha=5, seqlength=100, rate=c(1,1,1,1,1,1), frequency=c(1/4,1/
outfile, format="phylip", concat=TRUE)
```

Arguments

`sptree`	A species tree which must be a rooted tree.
`spname`	species names
`ntaxasp`	a vector of the number of individuals in each species
`ngene`	number of genes
`theta`	population size
`noclock`	0: clocklike species tree 1: nonclocklike species tree
`simsequence`	1: simulate sequences and gene trees, 0: simulate gene trees
`murate`	distribution of mutation rates
`alpha`	the shape parameter of dirichlet distribution
`seqlength`	the number of nucleotides along the sequences
`rate`	rates
`frequency`	nucleotide frequency
`outfile`	the full path of the output file
`format`	either "phylip" or "nexus"
`concat`	save the concatenated sequences or save single-gene sequences as different data in the same file

Value

The function writes sequences into a file.

Author(s)

Liang Liu <`lliu@uga.edu`>

`site.pattern` *Site patterns*

Description

The function returns site patterns.

Usage

```
site.pattern(seq)
```

Arguments

`seq`	DNA sequences with rows representing taxa and columns representing sites

Value

The function returns a matrix. Each row in the matrix represents a site pattern and the last number at each row is the frequency of the site pattern appeared in the DNA sequences.

Author(s)

Liang Liu <lliu@uga.edu>

Examples

```
seq<- matrix("A",nrow=4,ncol=5)
seq[1,]<-c("A","A","G","C","C")
seq[2,]<-c("A","G","G","C","C")
seq[3,]<-c("T","A","G","C","C")
seq[4,]<-c("A","A","G","T","T")
site.pattern(seq)
```

`site.summary` *summarize alignments*

Description

This function summarizes alignments.

Usage

```
site.summary(sequence)
```

Arguments

`sequence`	DNA alignment

Author(s)

Liang Liu

`species.name` *Species names in a tree string*

Description

The function can be used to obtain species names from a tree string.

Usage

```
species.name(str)
```

Arguments

`str`	a tree string in the parenthetical format

Details

The function returns the species names. If the tree string contains only the node number instead of species names, the function will return the node numbers.

Value

The function returns the species names.

Author(s)

Liang Liu <`lliu@uga.edu`>

See Also

`read.tree.string`

Examples

```
tree.string<-"(((H:4.2,C:4.2):3.1,G:7.3):6.3,O:13.5);"
species.name(tree.string)
```

`sptree.alleletable` *Create a sequence-species relationship*

Description

This function can create a matrix to present the sequence-species relationship.

Usage

```
sptree.alleletable(numsgenenodes)
```

Arguments

`numsgenenodes`	number of sequences for each species

Details

The matrix created by this function can be used as species.structure.

Author(s)

Liang Liu

Examples

```
numsgenenodes<-c(1,1,1,1,1,2,2,1,1,1,1,2,3,2,2,2,1,1,1,2,1,8,2,2,2,1,1,1)
species.structure<-sptree.alleletable(numsgenenodes)
```

`sptree.maxtree` *Maximum Tree*

Description

The function computes the Maximum Tree from multiple gene trees.

Usage

```
sptree.maxtree(genetreevector,spname,taxaname,species.structure)
```

Arguments

`genetreevector`	a vector of gene trees
`spname`	the species names
`taxaname`	the names of taxa
`species.structure`	the correspondence between species and taxa

Value

The function returns the node matrix and tree string of the maximum tree. It also returns the species names.

Author(s)

Liang Liu <`lliu@uga.edu`>

References

Liu, L. and D.K. Pearl. Species trees from gene trees: reconstructing Bayesian posterior distributions of a species phylogeny using estimated gene tree distributions. Systematic Biology, 2007, 56:504-514.

Edwards, S.V., L. Liu., and D.K. Pearl. High resolution species trees without concatenation. PNAS, 2007, 104:5936-5941.

Examples

```
genetreevector<-c("((((H:0.00302,C:0.00302):0.00304,G:0.00605):0.01029,O:0.01635):0.1,W:0.11635);",
"((((H:0.00402,G:0.00402):0.00304,C:0.00705):0.00929,O:0.01635):0.1,W:0.11635);");
species.structure<-matrix(0,5,5)
diag(species.structure)<-1
name<-species.name(genetreevector[1])
sptree.maxtree(genetreevector,name,name,species.structure)
```

`sptree.njst` *calculate the sptree.njst tree*

Description

This function can estimate species trees from a set of unrooted gene trees

Usage

```
sptree.njst(genetrees, taxaname, spname, species.structure)
```

Arguments

`genetrees`	a set of unrooted gene trees
`taxaname`	names of taxa
`spname`	names of species
`species.structure`	the taxaname-spname table

Author(s)

Liang Liu

Examples

```
sptree<-"(A:0.4,(B:0.3,(C:0.2,(D:0.1,E:0.1):0.1):0.1):0.1);"

spname<-species.name(sptree)
nspecies<-length(spname)
rootnode<-9
nodematrix<-read.tree.nodes(sptree,spname)$node
seq<-rep(1,nspecies)
species.structure<-matrix(0,nspecies,nspecies)
diag(species.structure)<-1

##population size, theta
nodematrix[,5]<-0.1
ngene<-5
genetree<-rep("",ngene)

##generate gene trees
for(i in 1:ngene)
{
genetree[i]<-sim.coaltree.sp(rootnode,nodematrix,nspecies,seq,spname)$gt
}

##construct the sptree.njst tree
sptree.njst(genetree,spname, spname, species.structure)
```

sptree.star *Build a STAR tree*

Description

The function can build a STAR tree from a set of gene trees. Although STAR can handle missing sequences, it requires that all possible pairs of species (n choose 2) should appear in at least one gene tree. Otherwise, STAR cannot calculate the pairwise distances among species.

Usage

```
sptree.star(trees, speciesname, taxaname, species.structure,outgroup,method="nj")
```

Arguments

`trees`	the gene tree vector
`speciesname`	species names
`taxaname`	taxa names
`species.structure`	a matrix defining the species-taxa relationship
`outgroup`	outgroup
`method`	UPGMA or NJ

Value

The function returns a STAR tree.

Author(s)

Liang Liu <`lliu@uga.edu`>

Examples

```
#create three gene trees
treestr<-rep("",4)
treestr[1]<-"((((H:0.00402,C:0.00402):0.00304,G:0.00706):0.00929,O:0.01635):0.1,W:0.11635);"
treestr[2]<-"((((H:0.00402,G:0.00402):0.00304,C:0.00706):0.00929,O:0.01635):0.1,W:0.11635);"
treestr[3]<-"((((O:0.00402,C:0.00402):0.00304,G:0.00706):0.00929,H:0.01635):0.1,W:0.11635);"
treestr[4]<-"((((H:0.00402,C:0.00402):0.00304,G:0.00706):0.00929,O:0.01635):0.1,W:0.11635);"

speciesname<-species.name(treestr[1])
taxaname<-speciesname
species.structure<-matrix(0,ncol=5,nrow=5)
diag(species.structure)<-1

sptree.star(treestr, speciesname, taxaname, species.structure,outgroup="W",method="nj")
```

sptree.steac *Build a STEAC tree*

Description

The function can build a STEAC tree from a set of gene trees.

Usage

```
sptree.steac(trees, speciesname, taxaname, species.structure,outgroup,method="nj")
```

Arguments

`trees`	the gene tree vector
`speciesname`	species names
`taxaname`	taxa names
`species.structure`	a matrix defining the species-taxa relationship
`outgroup`	outgroup
`method`	UPGMA or NJ

Value

The function returns a STEAC tree.

Author(s)

Liang Liu <`lliu@uga.edu`>

Examples

```
#create three gene trees
treestr<-rep("",4)
treestr[1]<-"((((H:0.00402,C:0.00402):0.00304,G:0.00706):0.00929,O:0.01635):0.1,W:0.11635);"
treestr[2]<-"((((H:0.00402,G:0.00402):0.00304,C:0.00706):0.00929,O:0.01635):0.1,W:0.11635);"
treestr[3]<-"((((O:0.00402,C:0.00402):0.00304,G:0.00706):0.00929,H:0.01635):0.1,W:0.11635);"
treestr[4]<-"((((H:0.00402,C:0.00402):0.00304,G:0.00706):0.00929,O:0.01635):0.1,W:0.11635);"

speciesname<-species.name(treestr[1])
taxaname<-speciesname
species.structure<-matrix(0,ncol=5,nrow=5)
diag(species.structure)<-1

sptree.steac(treestr, speciesname, taxaname, species.structure, outgroup="W", method="nj")
```

test.2sptree *testing if two species trees are significantly different*

Description

This function is testing if two species trees are significantly different from each other.

Usage

```
test.2sptree(path_mpest="mpest", sptree1, sptree2, genetreefile, ngenetree, nbootstrap)
```

Arguments

`path_mpest`	the full path of the mp-est binary
`sptree1`	the species tree 1
`sptree2`	the species tree 2
`genetreefile`	the gene tree file
`ngenetree`	the number of gene trees
`nbootstrap`	the number of bootstrap replicates

Author(s)

Liang Liu <`lliu@uga.edu`>

test.equalgenetree *testing if gene trees are identical*

Description

The function is the likelihood ratio test for the concatenation assumption that all gene trees have the same topology.

Usage

```
test.equalgenetree(path_raxml, inputfolder, nbootstrap)
```

Arguments

`path_raxml`	the full path of the raxml binary
`inputfolder`	the folder that contains all gene data. Each gene is a separate file
`nbootstrap`	The number of bootstrap replicates

Author(s)

Liang Liu <`lliu@uga.edu`>

`test.hybrid` *testing if there are two species trees*

Description

The test is based on the proportion of gene trees supporting the alternative species tree.

Usage

```
test.hybrid(path_mpest, genetreefile, tree1, tree2, nbootstrap=100)
```

Arguments

`path_mpest`	the full path of the mpest binary
`genetreefile`	the input gene tree file
`tree1`	the null species tree
`tree2`	the alternative species tree
`nbootstrap`	the number of bootstrap replicates

Author(s)

Liang Liu <`lliu@uga.edu`>

`test.submodel.valid.phyml` *model validation of the substitution models*

Description

The function includes chi-squres goodness-of-fit test for validating the substitution model. There are three tests, (1) the chi-square test for base frequencies, (2) the chi-square test for double-nucleotide frequencies, (3) the test for site patterns and the test statistics is sum(|exp_freq-obs_freq|).

Usage

```
test.submodel.valid.phyml(inputfile, path_phyml, path_seqgen, model, gamma=TRUE, pinv=FALSE, nboot
```

Arguments

`inputfile`	the sequence file
`path_phyml`	the full path of the phyml binary
`path_seqgen`	the path of the seq-gen binary
`model`	the substitution model
`gamma`	gamma parameter
`pinv`	proportion of invariant sites
`nbootstrap`	The number of bootstrap replicates in the site pattern test.

Author(s)

Liang Liu <`lliu@uga.edu`>

`test.submodel.valid.raxml` *model validation of the substitution models*

Description

The function includes chi-squres goodness-of-fit test for validating the GTRGAMMA model. There are three tests, (1) the chi-square test for base frequencies, (2) the chi-square test for double-nucleotide frequencies, (3) the test for site patterns and the test statistics is sum(|exp_freq-obs_freq|).

Usage

```
test.submodel.valid.raxml(inputfile, path_raxml, path_seqgen, nbootstrap=100)
```

Arguments

`inputfile`	the sequence file
`path_raxml`	the full path of the raxml binary
`path_seqgen`	the path of the seq-gen binary
`nbootstrap`	The number of bootstrap replicates in the site pattern test.

Author(s)

Liang Liu <`lliu@uga.edu`>

`tree.brlens` *summarizing branch lengths*

Description

This function summarizes the branch lengths of a tree.

Usage

```
tree.brlens(tree)
```

Arguments

`tree`	a tree string

Value

`summary`	summary of the branch lengths
`sd`	standard deviation of the branch lengths
`molClock`	standard deviation of tip-root distances

Author(s)

Liang Liu

Examples

```
data(dat.unrootedtree)
tree.brlens(dat.unrootedtree[1])
```

`tree.consensus` *consensus tree*

Description

The function builds consensus tree.

Usage

```
tree.consensus(treefile, outfile, rooted=FALSE, sumtreepath="sumtrees.py")
```

Arguments

`treefile`	tree file
`outfile`	the consensus tree file
`rooted`	rooted or unrooted trees
`sumtreepath`	the full path of sumtrees.py

Value

`contree`	the consensus tree with bootstrap support values
`bsvalue`	bootstrap values

Author(s)

Liang Liu <`lliu@uga.edu`>

`tree.distance` *tree distance*

Description

This function calculates the distance between two trees. Two trees are pruned to have the same set of species.

Usage

```
tree.distance(tree1,tree2, method="RF", normalize=TRUE)
```

Arguments

`tree1`	a tree string
`tree2`	a tree string
`method`	RF: RF distance, SC: branch score distance
`normalize`	normalized by (2 * the number of internal branches)

Value

It calculates the distance of two trees.

Author(s)

Liang Liu

Examples

```
data(dat.unrootedtree)
tree.distance(dat.unrootedtree[1], dat.unrootedtree[2])
```

---

tree.name2node *Replace species names by their node numbers*

Description

This function replaces the species names in the tree string with their node numbers.

Usage

```
tree.name2node(treestr,name="")
```

Arguments

treestr	the tree string
name	the species names

Details

If species names are not given, the function will use the sorted species names in the tree string.

Value

The function returns the tree string with the species names replaced by the node numbers.

Author(s)

Liang Liu <lliu@uga.edu>

See Also

tree.node2name

Examples

```
treestr<-"(((H:4.2,C:4.2):3.1,G:7.3):6.3,O:13.5);"
name<-c("H","G", "C", "O")
tree.name2node(treestr,name)
```

`tree.noclock2clock` *Convert a non-clocklike tree to a clocklike tree*

Description

This function converts a non-clocklike tree to a clocklike tree using an ad-hoc approach described in the paper Liu et al 2007.

Usage

```
tree.noclock2clock(inode, treematrix, nspecies)
```

Arguments

`inode`	root of the tree
`treematrix`	tree node matrix
`nspecies`	the number of species in the tree

Value

The function returns the tree node matrix of the clocklike tree.

Author(s)

Liang Liu

References

~put references to the literature/web site here ~

Examples

```
treestr<-"(((H:1,C:3):2,G:6):2,O:10);"
name<-species.name(treestr)
treenode<-read.tree.nodes(treestr,name)$nodes
tree.noclock2clock(7,treenode,4)
```

`tree.node2name` *Replace node numbers by species names in a tree string*

Description

This function replaces node numbers in a tree string by species names.

Usage

```
tree.node2name(treestr,name="")
```

Arguments

`treestr`	a tree string
`name`	species names

Value

The function returns the tree string with the node numbers replaced by the species names.

Author(s)

Liang Liu

See Also

`tree.name2node`

Examples

```
treestr<-"(((1:4.2,2:4.2):3.1,3:7.3):6.3,4:13.5);"
name<-c("H","C", "G", "O")
tree.node2name(treestr,name)
```

`tree.partition` *partition a tree*

Description

partition a tree.

Usage

```
tree.partition(tree,nspecies)
```

Arguments

`tree`	the tree node matrix
`nspecies`	the number of species

Value

The function returns a matrix. Each row represents a particular partition of the tree. The position of "1" in the matrix indicates the presence of the corresponding species in the partition. The last number at each row is the frequency of that partition. This function returns the partition matrix for only one tree.

Author(s)

Liang Liu

Examples

```
treestr<-"((((H:0.00402,C:0.00402):0.00304,G:0.00707):0.00929,O:0.01635):0.1,W:0.12);"
nodematrix<-read.tree.nodes(treestr)$nodes
tree.partition(nodematrix,5)
#
#     [,1] [,2] [,3] [,4] [,5] [,6]
#[1,]    1    0    1    0    0    1
#[2,]    1    1    1    0    0    1
#[3,]    1    1    1    1    0    1
#
#The last number of each row is the frequency of the corresponding partition.
#For example, the frequency of the first partition (1 0 1 0 0) is 1.
#The first partition includes species 1 and 3
#as indicated by the position of 1 in the partition.
#Each row represens a partition and its frequency.
```

`tree.plot` *tree plot*

Description

The function plots phylogenetic trees.

Usage

```
tree.plot(tree)
```

Arguments

`tree`	a phylogenetic tree in newrick format

Author(s)

use the function "plot.phylo" in package ape to plot phylogenetic trees.

See Also

`write.subtree`, `read.tree.string`

Examples

```
treestr<-"(((H:4.2,C:4.2):3.1,G:7.3):6.3,O:13.5);"
tree.plot(treestr)
```

`tree.probdist` *tree.probdist*

Description

This function can be used to find the probability distribution of trees.

Usage

```
tree.probdist(treefile, sumtreepath="sumtrees.py")
```

Arguments

`treefile`	the tree file
`sumtreepath`	the full path of sumtrees.py

Value

the function returns the probability distribution of trees.

Author(s)

Liang Liu

`tree.subtree` *Subtree*

Description

The function returns the subtree under the node `inode`

Usage

```
tree.subtree(inode, name, nodematrix)
```

Arguments

`inode`	the root node of the subtree
`name`	the species names
`nodematrix`	the tree node matrix

Value

The function returns the tree string of the subtree.

Author(s)

Liang Liu <`lliu@harvard.edu`>

See Also

`del.node`

Examples

```
treestr<-"((((H:0.00402,C:0.00402):0.00304,G:0.00707):0.00929,O:0.01635):0.1,W:0.12);"
nodematrix<-read.tree.nodes(treestr)$nodes
spname<-read.tree.nodes(treestr)$names
tree.subtree(7,spname,nodematrix)
```

`tree.upgma` *tree.upgma tree*

Description

The function computes the upgma tree from multiple gene trees.

Usage

```
tree.upgma(dist, name, method="average")
```

Arguments

`dist`	a distance matrix
`name`	the species names
`method`	the method for recalculate pairwise distances. two options: averge or min.

Value

The function returns a tree node matrix, a tree string and species names.

Author(s)

Liang Liu <`lliu@uga.edu`>

See Also

`sptree.maxtree`

Examples

```
dist<-matrix(runif(25),5,5)
dist<-(dist+t(dist))/2
diag(dist)<-0
tree.upgma(dist,name=c("H","G","C","O","W"))
```

write.dna.seq *Write sequences to a Nexus file*

Description

write sequences to a Nexus file.

Usage

```
write.dna.seq(sequence, name, file = "", format="nexus",
program="mrbayes",partition=matrix(0,ncol=2,nrow=1),
clock=0, popmupr=0, ngen=1000000,nrun=1,nchain=1,samplefreq=100,
taxa=as.vector,burnin=1000,gamma="(3,0.02)",
outgroup=1,outfile="",append = FALSE)
```

Arguments

`sequence`	DNA sequences
`name`	taxa names
`file`	output file
`program`	either mrbayes or best.
`format`	nexus or phylip
`partition`	each partition corresponds a gene or a locus.
`clock`	1:clock, 0:no clock
`popmupr`	for non-clock species tree model
`ngen`	number of generations
`nrun`	number of runs
`nchain`	number of chains
`samplefreq`	sampling frequency
`taxa`	species names if best is defined
`burnin`	burn in
`outgroup`	the node number of the outgroup
`outfile`	output file
`append`	append or not
`gamma`	parameters in the inverse gamma distribution as the prior of theta.

Author(s)

Liang Liu

`write.seq.phylip` *write concatenated sequences to a file*

Description

This function writes concatenated sequences to a file.

Usage

```
write.seq.phylip(sequence, name, length, outfile = "",append=FALSE)
```

Arguments

`sequence`	concatenated sequences as strings
`name`	species names
`length`	the length of sequences per line in the output file
`outfile`	output file
`append`	FALSE or TRUE

Author(s)

Liang Liu

`write.subtree` *Write a sub-tree into a string*

Description

write a tree or a sub-tree into a string in parenthetical format

Usage

```
write.subtree(inode, nodematrix,taxaname,root,print.support=FALSE)
```

Arguments

`inode`	the root node of a sub-tree
`nodematrix`	a tree node matrix
`taxaname`	taxa names
`root`	the root node of a sub-tree
`print.support`	print out support values if print.support=TRUE

Details

If inode is the root of the tree, the function will write the whole tree into a string in parenthetical format. If inode is not the root node, the function will write the sub-tree into a string. The function works for both rooted trees and unrooted trees.

Value

The function returns a tree string in parenthetical format

Author(s)

Liang Liu <lliu@uga.edu>

See Also

write.tree.string, read.tree.nodes

Examples

```
data(dat.coaltree)
tree<-read.tree.nodes(dat.coaltree$sptree)
tree$nodes
tree$names
write.subtree(7,tree$nodes,tree$names,7)
```

---

write.tree.string *Write a tree file*

Description

The function writes tree strings to a file in NEXUS or PHYLIP format.

Usage

```
write.tree.string(X, format = "Nexus", file = "", name = "")
```

Arguments

X	a vector of tree strings
format	tree file format
file	the file name
name	the species names

Details

If name is provided, the function will use name as the species names in the translation block in the NEXUS tree file. Otherwise, the species names will be extracted from the tree strings.

Value

The function returns a tree file in the format of NEXUS or PHYLIP.

Author(s)

Liang Liu <lliu@uga.edu>

References

Felsenstein, J. The Newick tree format. `http://evolution.genetics.washington.edu/phylip/newicktree.html`

See Also

`write.subtree`, `read.tree.string`

Index

∗Topic **IO**
- loglike.triple, 19
- read.tree.nodes, 23
- read.tree.string, 24
- tree.plot, 45
- write.subtree, 49
- write.tree.string, 50

∗Topic **datasets**
- dat.coaltree, 8
- dat.finch, 8
- dat.modelcode, 9
- dat.unrootedtree, 9

∗Topic **programming**
- alignment.mle.remove, 4
- alignment.reference.remove, 4
- alignment.summary, 5
- alignment.trim, 5
- bootstrap, 6
- bootstrap.mulgene, 7
- control.mpest, 8
- del.brlens, 9
- del.comments, 10
- del.node, 10
- dist.dna, 11
- dist.internode, 12
- dist.species, 12
- dist.taxa, 13
- file.beast2phylip, 14
- file.concatData, 14
- file.fasta2phylip, 15
- file.nexus2phylip, 15
- file.phylip2nexus, 16
- file.separateGeneData, 16
- is.clock, 17
- is.rootedtree, 18
- loglike.coal, 18
- parse.modeltest, 21
- parse.phyml, 21
- parse.raxml, 22
- phybase-package, 3
- read.dna.seq, 22
- run.modeltest, 25
- run.mpest, 25
- run.seqgen, 26
- sim.coal.mpest, 26
- sim.coaltree, 27
- sim.coaltree.sp, 28
- sim.coaltree.sp.mu, 29
- sim.dna, 30
- sim.SeqfromSp, 31
- site.pattern, 32
- site.summary, 32
- species.name, 33
- sptree.alleletable, 33
- sptree.maxtree, 34
- sptree.njst, 35
- sptree.star, 36
- sptree.steac, 37
- test.2sptree, 38
- test.equalgenetree, 38
- test.hybrid, 39
- test.submodel.valid.phyml, 39
- test.submodel.valid.raxml, 40
- tree.brlens, 40
- tree.consensus, 41
- tree.distance, 41
- tree.noclock2clock, 43
- tree.partition, 44
- tree.probdist, 46
- tree.subtree, 46
- tree.upgma, 47
- write.dna.seq, 48
- write.seq.phylip, 49

∗Topic **univar**
- node.height, 20
- tree.name2node, 42
- tree.node2name, 43

alignment.mle.remove, 4
alignment.reference.remove, 4
alignment.summary, 5
alignment.trim, 5

bootstrap, 6, *7*
bootstrap.mulgene, 7

control.mpest, 8

dat.coaltree, 8
dat.finch, 8
dat.modelcode, 9
dat.unrootedtree, 9
del.brlens, 9
del.comments, 10
del.node, 10, *47*
dist.dna, 11
dist.internode, 12
dist.species, 12
dist.taxa, *13*, 13

file.beast2phylip, 14
file.concatData, 14
file.fasta2phylip, 15
file.nexus2phylip, 15
file.phylip2nexus, 16
file.separateGeneData, 16

is.clock, 17, *18*
is.rootedtree, *17*, 18

loglike.coal, 18
loglike.triple, 19

node.height, 20

parse.modeltest, 21
parse.phyml, 21
parse.raxml, 22
PhyBase (phybase-package), 3
phybase-package, 3

read.dna.seq, 22
read.tree.nodes, 23, *24*, *25*, *50*
read.tree.string, *20*, *23*, 24, *33*, *45*, *51*
run.modeltest, 25
run.mpest, 25
run.seqgen, 26

sim.coal.mpest, 26
sim.coaltree, 27, *28*, *30*
sim.coaltree.sp, *27*, 28
sim.coaltree.sp.mu, 29
sim.dna, 30
sim.SeqfromSp, 31
site.pattern, 32
site.summary, 32
species.name, *23*, 33
sptree.alleletable, 33
sptree.maxtree, 34, *47*
sptree.njst, 35
sptree.star, 36
sptree.steac, 37

test.2sptree, 38
test.equalgenetree, 38
test.hybrid, 39
test.submodel.valid.phyml, 39
test.submodel.valid.raxml, 40
tree.brlens, 40
tree.consensus, 41
tree.distance, 41
tree.name2node, 42, *44*
tree.noclock2clock, 43
tree.node2name, *42*, 43
tree.partition, 44
tree.plot, 45
tree.probdist, 46
tree.subtree, 46
tree.upgma, *11*, 47

write.dna.seq, 48
write.seq.phylip, 49
write.subtree, *20*, *45*, 49, *51*
write.tree.string, *24*, *25*, *50*, 50