

Package ‘phangorn’

October 30, 2009

Title Phylogenetic analysis in R

Version 0.99-3

Date 2009-30-10

Author Klaus Schliep

Description Phylogenetic analysis in R (Estimation of phylogenetic trees and networks using Maximum Likelihood, Maximum Parsimony, Distance methods & Hadamard conjugation)

Maintainer Klaus Schliep <klaus.schliep@gmail.com>

Imports ape, stats

Depends quadprog, ape (>= 2.2-3)

License GPL (>= 2)

Repository CRAN

Date/Publication 2009-10-30 17:32:37

R topics documented:

allTrees	2
bootstrap.pml	3
chloroplast	3
designTree	4
dfactorial	5
dist.hamming	5
distanceHadamard	6
hadamard	7
Laurasiatherian	9
NJ	9
nni	10
parsimony	11
phyDat	12
pml	13

pmlCluster	16
pmlMix	17
pmlPart	19
read.aa	21
SH.test	22
simSeq	23
splitsNetwork	24
treedist	25
upgma	26
yeast	27
Index	28

allTrees	<i>Compute all trees topologies.</i>
----------	--------------------------------------

Description

allTrees computes all tree topologies for rooted or unrooted trees with up to 10 tips. allTrees returns bifurcating trees.

Usage

```
allTrees(n, rooted = FALSE, tip.label = NULL)
```

Arguments

n	Number of tips (<=10).
rooted	Rooted or unrooted trees (default: rooted).
tip.label	Tip labels.

Value

an object of class multiPhylo.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

Examples

```
trees <- allTrees(5)
par(mfrow = c(3, 5))
for(i in 1:15)plot(trees[[i]])
```

bootstrap.pml *Bootstrap*

Description

bootstrap.pml performs (non-parametric) bootstrap analysis and bootstrap.phyDat produces a list of bootstrapped datasets.

Usage

```
bootstrap.pml(x, bs = 3, trees = TRUE, ...)  
bootstrap.phyDat(x, FUN, bs = 100, ...)
```

Arguments

x	an object of class pml or phyDat.
bs	number of bootstrap samples.
trees	return trees only (default) or whole pml objects.
...	further parameters used by optim.pml.
FUN	the function to estimate the trees.

Value

returns an object of class multi.phylo or a list where each element is an object of class pml.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[optim.pml](#), [pml](#)

chloroplast *Chloroplast alignment*

Description

Amino acid alignment of 15 genes of 19 different chloroplast.

Usage

```
data(yeast)
```

Examples

```
data(chloroplast)  
chloroplast
```

`designTree`*Compute a design matrix*

Description

`designTree` and `designSplits` compute design matrices for the estimation of edge length of (phylogenetic) trees using linear models. `designTree` also computes a contrast matrix if the method is "rooted".

Usage

```
designTree(tree, method = "unrooted", ...)  
designSplits(x, splits = "all", ...)
```

Arguments

<code>tree</code>	an object of class <code>phylo</code>
<code>method</code>	design matrix for an "unrooted" or "rooted" ultrametric tree
<code>x</code>	number of taxa.
<code>splits</code>	one of "all", "star"
<code>...</code>	further arguments, passed to other methods.

Value

a matrix.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[fastme](#), [distanceHadamard](#)

Examples

```
example(NJ)  
dm <- as.matrix(dm)  
y <- dm[lower.tri(dm)]  
X <- designTree(tree)  
summary(lm(y~X-1))
```

dfactorial *Arithmetic Operators*

Description

double factorial function

Usage

```
dfactorial(x)
ldfactorial(x)
```

Arguments

x a numeric skalar or vector

Value

dfactorial(x) returns the double factorial, that is $x = 1 * 3 * 5 * \dots * x$ and ldfactorial(x) is the natural logarithm of it.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[factorial](#)

Examples

```
dfactorial(1:10)
```

dist.hamming *Pairwise Distances from Sequences*

Description

dist.hamming and dist.logDet compute pairwise distances for an object of class phyDat.
dist.ml fits distances for amino acid models.

Usage

```
dist.hamming(x, ratio = TRUE)
dist.logDet(x)
dist.ml(x, model="JC69", ...)
```

Arguments

`x` an object of class `dist.logDet`
`ratio` compute uncorrected ('p') distance or character difference.
`model` One of "JC69", "WAG", "JTT", "LG" or "Dayhoff"
`...` Further arguments passed to or from other methods.

Value

an object of class `dist`

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Lockhart, P. J., Steel, M. A., Hendy, M. D. and Penny, D. (1994) Recovering evolutionary trees under a more realistic model of sequence evolution. *Molecular Biology and Evolution*, **11**, 605–602.

See Also

For more distance methods for nucleotide data see [dist.dna](#)

Examples

```
data(Laurasiatherian)
dm1 <- dist.hamming(Laurasiatherian)
tree1 <- NJ(dm1)
dm2 <- dist.logDet(Laurasiatherian)
tree2 <- NJ(dm2)
treedist(tree1, tree2)
```

distanceHadamard *Distance Hadamard*

Description

Distance Hadamard produces spectra of splits from a distance matrix.

Usage

```
distanceHadamard(dm)
```

Arguments

`dm` A distance matrix.

Value

`distanceHadamard` returns a matrix. The first column contains the distance spectra, the second one the edge-spectra.

Author(s)

Klaus Schliep (klaus.schliep@gmail.com), Tim White

References

Hendy, M. D. and Penny, D. (1993). Spectral Analysis of Phylogenetic Data. *Journal of Classification*, **10**, 5-24.

See Also

[hadamard](#)

Examples

```
data(yeast)
dm = dist.hamming(yeast)
dm = as.matrix(dm)
fit = distanceHadamard(dm)
```

hadamard

Hadamard Matrices and Fast Hadamard Multiplication

Description

A collection of functions to perform Hadamard conjugation.

Usage

```
hadamard(x)
fhm(v)
h2st(obj, levels=c("r", "y"))
h4st(obj, levels = c("a", "c", "g", "t"))
write.nexus.splits(obj, file="")
```

Arguments

<code>x</code>	a vector of length 2^n , where n is an integer.
<code>v</code>	a vector of length 2^n , where n is an integer.
<code>obj</code>	a data.frame or character matrix, typical a sequence alignment.
<code>levels</code>	levels of the sequences.
<code>file</code>	a file name.

Details

`h2st` and `h4st` perform Hadamard conjugation for 2-state (binary, RY-coded) or 4-state (DNA/RNA) data. `write.nexus.splits` writes splits returned from `h2st` or `distanceHadamard` to a nexus file, which can be processed by Spectronet or Splitstree.

Value

`hadamard` returns a Hadamard matrix. `fhm` returns the fast Hadamard multiplication.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

- Hendy, M.D. (1989). The relationship between simple evolutionary tree models and observable sequence data. *Systematic Zoology*, **38** 310–321.
- Hendy, M. D. and Penny, D. (1993). Spectral Analysis of Phylogenetic Data. *Journal of Classification*, **10**, 5–24.
- Hendy, M. D. (2005). Hadamard conjugation: an analytical tool for phylogenetics. In O. Gascuel, editor, *Mathematics of evolution and phylogeny*, Oxford University Press, Oxford
- Waddell P. J. (1995). Statistical methods of phylogenetic analysis: Including hadamard conjugation, LogDet transforms, and maximum likelihood. *PhD thesis*.

See Also

[distanceHadamard](#)

Examples

```
H = hadamard(3)
v = 1:8
H
fhm(v)
## Not run:
data(Laurasiatherian)
dat = as.character(Laurasiatherian)[1:10,]
# RY-coding
dat[dat=="a"] = "r"
dat[dat=="g"] = "r"
dat[dat=="c"] = "y"
dat[dat=="t"] = "y"

fit = h2st(dat)
write.nexus.splits(fit[fit$edges>1e-4, ][-1,], file = "test.nxs")
# read this file into Spectronet or Splitstree to show the network
unlink("test.nxs")
## End(Not run)
```

Laurasiatherian *Laurasiatherian data (AWCMEE)*

Description

Laurasiatherian RNA sequence data

Usage

```
data(Laurasiatherian)
```

Source

Data have been taken from <http://awcmee.massey.ac.nz> and were converted to R format by (klaus.schliep@gmail.com).

Examples

```
data(Laurasiatherian)
str(Laurasiatherian)
```

NJ *Neighbor-Joining*

Description

This function performs the neighbor-joining tree estimation of Saitou and Nei (1987). UNJ is the unweighted version from Gascuel (1997).

Usage

```
NJ(x)
UNJ(x)
```

Arguments

x A distance matrix.

Value

an object of class "phylo".

Author(s)

Klaus P. Schliep (klaus.schliep@gmail.com)

References

- Saitou, N. and Nei, M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, **4**, 406–425.
- Studier, J. A and Keppler, K. J. (1988) A Note on the Neighbor-Joining Algorithm of Saitou and Nei. *Molecular Biology and Evolution*, **6**, 729–731.
- Gascuel, O. (1997) Concerning the NJ algorithm and its unweighted version, UNJ. in Birkin et. al. *Mathematical Hierarchies and Biology*, 149–170.,

See Also

[nj](#), [dist.dna](#), [dist.hamming](#), [upgma](#), [fastme](#)

Examples

```
data(Laurasiatherian)
dm <- dist.logDet(Laurasiatherian)
tree <- NJ(dm)
plot(tree)
```

nni

Tree rearrangements.

Description

`nni` returns a list of all trees which are one nearest neighbor interchange away. `rNNI` and `rSPR` are two methods which simulate random trees which are a specified number of rearrangement appart from the input tree. Both methods assume that the input tree is bifurcating. These methods may be useful in simulation studies.

Usage

```
nni(tree)
rSPR(tree, moves=1, n=1)
rNNI(tree, moves=1, n=1)
```

Arguments

<code>tree</code>	A phylogenetic tree, object of class <code>phylo</code> .
<code>moves</code>	Number of tree rearrangements to be transromed on a tree.
<code>n</code>	Number of trees to be simulated.

Value

an object of class `multiPhylo`.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

Examples

```
tree = unroot(rtree(20))
trees1 <- nni(tree)
trees2 <- rSPR(tree, 2, 10)
```

parsimony

Parsimony tree.

Description

`parsimony` returns the parsimony score of a tree. `optim.parsimony` tries to find the maximum parsimony tree using NNI rearrangements.

Usage

```
parsimony(tree, data, method="sankoff", ...)  
optim.parsimony(tree, data, cost=NULL, ...)  
fitch(tree, data, site = FALSE)  
sankoff(tree, data, cost = NULL)
```

Arguments

<code>data</code>	A object of class <code>phyDat</code> containing (dna) sequences.
<code>tree</code>	tree to start the nni search from.
<code>method</code>	one of 'fitch' or 'sankoff'.
<code>cost</code>	A cost matrix for the transitions between two states.
<code>site</code>	logical, if TRUE site wise parsimony scores are returned.
<code>...</code>	Further arguments passed to or from other methods.

Value

`parsimony` returns the maximum parsimony score (`pscore`). `optim.parsimony` returns a tree after NNI rearrangements.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Felsenstein, J. (2004). *Inferring Phylogenies*. Sinauer Associates, Sunderland.

See Also

[nni](#), [NJ](#), [pml](#)

Examples

```
data(Laurasiatherian)
dm = dist.logDet(Laurasiatherian)
tree = NJ(dm)
parsimony(tree, Laurasiatherian)
ptree <- optim.parsimony(tree, Laurasiatherian)
```

 phyDat

Conversion among Sequence Formats

Description

These functions transform several DNA formats into the phyDat format. `allSitePattern` generates an alignment of all possible site patterns.

Usage

```
phyDat(data, type = "DNA", levels = NULL, return.index=TRUE, ...)
read.phyDat(file, format="phylip", type="DNA", ...)
write.phyDat(x, file, format="phylip",...)
## S3 method for class 'DNABin':
as.phyDat(x, ...)
## S3 method for class 'phyDat':
as.character(x, ...)
## S3 method for class 'phyDat':
as.data.frame(x, ...)
## S3 method for class 'phyDat':
as.DNABin(x, ...)
allSitePattern(n, levels=c("a","c","g","t"), names=NULL)
```

Arguments

<code>data</code>	An object containing sequences.
<code>x</code>	An object containing sequences.
<code>type</code>	Type of sequences ("DNA", "AA" or "USER").
<code>levels</code>	Level attributes.
<code>return.index</code>	If TRUE returns a index of the site patterns.
<code>file</code>	A file name.
<code>format</code>	File format of the sequence alignment (see details).
<code>n</code>	Number of sequences.
<code>names</code>	Names of sequences.
<code>...</code>	further arguments passed to or from other methods.

Details

If type "USER" a vector has to be give to levels. For example `c("a", "c", "g", "t", "-")` would create a data object that can be used in phylogenetic analysis with gaps as fifth state. `allSitePattern` returns all possible site patterns and can be usefull in simulation studies. `write.phyDat` calls the function `write.dna` or `write.nexus.data` and `read.phyDat` calls the function `read.dna`, `read.aa` or `read.nexus.data` see for more details over there.

You may import data directly with `read.dna` or `read.nexus.data` and convert the data to class `phyDat`.

Value

The functions return an object of class `phyDat`.

Author(s)

Klaus Schliep (klaus.schliep@gmail.com)

See Also

[DNABin](#), [as.DNABin](#), [read.dna](#), [read.aa](#) and [read.nexus.data](#) and the example of [pmlMix](#) for the use of `allSitePattern`

Examples

```
data(Laurasiatherian)
class(Laurasiatherian)
Laurasiatherian
# transform into old ape format
LauraChar <- as.character(Laurasiatherian)
# and back
Laura <- phyDat(LauraChar, return.index=TRUE)
all.equal(Laurasiatherian, Laura)
allSitePattern(5)
```

pml

Likelihood of a tree.

Description

`pml` computes the likelihood of a phylogenetic tree given a sequence alignment and a model. `optim.pml` optimizes the different model parameters.

Usage

```
pml(tree, data, bf=NULL, Q=NULL, inv=0, k=1, shape=1, rate=1, model="", ...)
optim.pml(object, optNni=FALSE, optBf=FALSE, optQ=FALSE,
  optInv=FALSE, optGamma=FALSE, optEdge=TRUE, optRate=FALSE,
  control = pml.control(maxit=10, eps=0.001, trace=TRUE), ...)
```

Arguments

<code>tree</code>	A phylogenetic <code>tree</code> , object of class <code>phylo</code> .
<code>data</code>	The (DNA) alignment.
<code>bf</code>	Base frequencies.
<code>Q</code>	A vector containing the lower triangular part of the rate matrix.
<code>inv</code>	Proportion of invariable sites.
<code>k</code>	Number of intervals of the discrete gamma distribution.
<code>shape</code>	Shape parameter of the gamma distribution.
<code>rate</code>	Rate.
<code>model</code>	Amino acid models: one of "WAG", "JTT", "Dayhoff" or "LG"
<code>object</code>	An object of class <code>pml</code> .
<code>optNni</code>	Logical value indicating whether topology gets optimized (NNI).
<code>optBf</code>	Logical value indicating whether base frequencies gets optimized.
<code>optQ</code>	Logical value indicating whether rate matrix gets optimized.
<code>optInv</code>	Logical value indicating whether proportion of variable size gets optimized.
<code>optGamma</code>	Logical value indicating whether gamma rate parameter gets optimized.
<code>optEdge</code>	Logical value indicating the edge lengths gets optimized.
<code>optRate</code>	Logical value indicating the overall rate gets optimized.
<code>control</code>	A list of parameters for controlling the fitting process.
<code>...</code>	Further arguments passed to or from other methods.

Details

The topology search uses a nearest neighbour interchange (NNI) and the implementation is similar to phyML. The option `model` is only used for amino acid models. Here is an overview how to estimate different phylogenetic models with `pml`:

<code>model</code>	<code>optBf</code>	<code>optQ</code>
Jukes-Cantor	FALSE	FALSE
F81	TRUE	FALSE
symmetric	FALSE	TRUE
GTR	TRUE	TRUE

So far 4 amino acid models are supported ("WAG", "JTT", "Dayhoff" and "LG").

Value

Returns a list of class `ll.phylo`

<code>logLik</code>	Log likelihood of the tree.
<code>siteLik</code>	Site log likelihoods.
<code>root</code>	likelihood in the root node.

weight Weight of the site patterns.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Felsenstein, J. (1981) Evolutionary trees from DNA sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, **17**, 368–376.

Felsenstein, J. (2004). *Inferring Phylogenies*. Sinauer Associates, Sunderland.

Yang, Z. (2006). *Computational Molecular evolution*. Oxford University Press, Oxford.

Whelan, S. and Goldman, N. (2001) A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. *Molecular Biology and Evolution*, **18**, 691–699

Le, S.Q. and Gascuel, O. (2008) LG: An Improved, General Amino-Acid Replacement Matrix *Molecular Biology and Evolution* **25(7)**, 1307–1320

Examples

```
example(NJ)
# Jukes-Cantor (starting tree from NJ)
fitJC <- pml(tree, Laurasiatherian)
# optimise edge length parameter
fitJC <- optim.pml(fitJC)
fitJC

## Not run:
# search for a better tree using NNI rearrangements
fitJC <- optim.pml(fitJC, optNni=TRUE)
fitJC
plot(fitJC$tree)

# JC + Gamma + I - model
fitJC_GI <- update(fitJC, k=4, inv=.2)
# optimise shape parameter + proportion of invariant sites
fitJC_GI <- optim.pml(fitJC_GI, optGamma=TRUE, optInv=TRUE)
# GTR + Gamma + I - model
fitGTR <- optim.pml(fitJC_GI, optNni=TRUE, optGamma=TRUE, optInv=TRUE, optBf=TRUE, optQ=TRUE)
## End(Not run)

# 2-state data (RY-coded)

dat <- as.character(Laurasiatherian)
# RY-coding
dat[dat=="a"] <- "r"
dat[dat=="g"] <- "r"
dat[dat=="c"] <- "y"
dat[dat=="t"] <- "y"
dat <- phyDat(dat, levels=c("r","y"))
```

```

fit2ST <- pml(tree, dat, k=4, inv=.25)
fit2ST <- optim.pml(fit2ST, optNni=TRUE, optGamma=TRUE, optInv=TRUE)
fit2ST
# show some of the methods available for class pml
methods(class="pml")

```

pmlCluster

Stochastic Partitioning

Description

Stochastic Partitioning of genes into p cluster.

Usage

```
pmlCluster(formula, fit, weight, p = 4, part = NULL, ...)
```

Arguments

formula	a formula object (see details).
fit	an object of class pml.
weight	weight is matrix of frequency of site patterns for all genes.
p	number of clusters.
part	starting partition, otherwise a random partiton is generated.
...	Further arguments passed to or from other methods.

Details

The formula object allows to specify which parameter get optimised. The formula is generally of the form $edge + bf + Q \sim rate + shape + \dots$, on the left side are the parameters which get optimised over all cluster, on the right the parameter which are optimised specific to each cluster. The parameters available are "nni", "bf", "Q", "inv", "shape", "edge", "rate". Each parameter can be used only once in the formula. There are also some restriction on the combinations how parameters can get used. "rate" is only available for the right side. When "rate" is speciefied on the left hand side "edge" has to be specified (on either side), if "rate" is specified on the right hand side it follows directly that edge is too.

Value

pmlCluster returns a list with elements

logLik	log-likelihood of the fit
trees	a list of all trees during the optimisation.
fits	fits for the final partitions

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[pml](#), [pmlPart](#), [pmlMix](#), [SH.test](#)

Examples

```
## Not run:
data(yeast)
dm <- dist.logDet(yeast)
tree <- NJ(dm)
fit=pml(tree,yeast)
fit = optim.pml(fit)

weight=xtabs(~ index+genes,attr(yeast, "index"))
set.seed(1)

sp <- pmlCluster(edge~rate, fit, weight, p=4)
sp
SH.test(sp)
## End(Not run)
```

pmlMix

Phylogenetic mixture model

Description

Phylogenetic mixture model.

Usage

```
pmlMix(formula, fit, m = 2, omega = rep(1/m, m), ...)
```

Arguments

formula	a formula object (see details).
fit	an object of class pml.
m	number of mixtures.
omega	mixing weights.
...	Further arguments passed to or from other methods.

Details

The formula object allows to specify which parameter get optimised. The formula is generally of the form $\text{edge} + \text{bf} + \text{Q} \sim \text{rate} + \text{shape} + \dots$, on the left side are the parameters which get optimised over all mixtures, on the right the parameter which are optimised specific to each mixture. The parameters available are "nni", "bf", "Q", "inv", "shape", "edge", "rate". Each parameters can be used only once in the formula. "rate" and "nni" are only available for the right side of the formula. On the other hand parameters for invariable sites are only allowed on the lefthand side. The convergence of the algorithm is very slow and is likely that the algorithm can get stuck in local optima.

Value

pmlMix returns a list with elements

logLik	log-likelihood of the fit
omega	mixing weights.
fits	fits for the final mixtures.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[pml](#), [pmlPart](#), [pmlCluster](#)

Examples

```
X <- allSitePattern(5)
tree <- read.tree(text = "((t1:0.3,t2:0.3):0.1,(t3:0.3,t4:0.3):0.1,t5:0.5);")
fit <- pml(tree,X, k=4)
weights <- 1000*exp(fit$site)
attr(X, "weight") <- weights
fit1 <- update(fit, data=X, k=1)
fit2 <- update(fit, data=X)

(fitMixture <- pmlMix(edge~rate, fit1 , m=4))
(fit2 <- optim.pml(fit2, optGamma=TRUE))

## Not run:
data(Laurasiatherian)
dm <- dist.logDet(Laurasiatherian)
tree <- NJ(dm)
fit=pml(tree, Laurasiatherian)
fit = optim.pml(fit)

fit2 <- update(fit, k=4)
fit2 <- optim.pml(fit2, optGamma=TRUE)

fitMix = pmlMix(edge ~ rate, fit, m=4)
```

```

fitMix

#
# simulation of mixture models
#

X <- allSitePattern(5)
tree1 <- read.tree(text = "(t1:0.1,t2:0.5):0.1, (t3:0.1,t4:0.5):0.1,t5:0.5);")
tree2 <- read.tree(text = "(t1:0.5,t2:0.1):0.1, (t3:0.5,t4:0.1):0.1,t5:0.5);")
tree1 <- unroot(tree1)
tree2 <- unroot(tree2)
fit1 <- pml(tree1,X)
fit2 <- pml(tree2,X)

weights <- 2000*exp(fit1$site) + 1000*exp(fit2$site)
attr(X, "weight") <- weights

fit1 <- pml(tree1, X)
fit2 <- optim.pml(fit1)
logLik(fit2)
AIC(fit2, k=log(3000))

fitMixEdge = pmlMix( ~ edge, fit1, m=2)
logLik(fitMixEdge)
AIC(fitMixEdge, k=log(3000))

fit.p <- pmlPen(fitMixEdge, .25)
logLik(fit.p)
AIC(fit.p, k=log(3000))

## End(Not run)

```

pmlPart

Partition model.

Description

Model to estimate phylogenies for partitioned data.

Usage

```
pmlPart(formula, object, ...)
```

Arguments

formula	a formula object (see details).
object	an object of class pml or a list of objects of class pml .
...	Further arguments passed to or from other methods.

Details

The formula object allows to specify which parameter get optimised. The formula is generally of the form $\text{edge} + \text{bf} + \text{Q} \sim \text{rate} + \text{shape} + \dots$, on the left side are the parameters which get optimised over all partitions, on the right the parameter which are optimised specific to each partition. The parameters available are "nni", "bf", "Q", "inv", "shape", "edge", "rate". Each parameters can be used only once in the formula. "rate" and "nni" are only available for the right side of the formula.

For partitions with different edge weights, but same topology, pmlPen can try to find more parsimonious models (see example).

Value

kcluster returns a list with elements

logLik	log-likelihood of the fit
trees	a list of all trees during the optimisation.
object	an object of class "pml" or "pmlPart"

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[pml](#), [pmlCluster](#), [pmlMix](#)

Examples

```
data(yeast)
dm <- dist.logDet(yeast)
tree <- NJ(dm)
fit <- pml(tree, yeast)
fits <- optim.pml(fit)

weight=xtabs(~ index+genes, attr(yeast, "index"))[,1:10]

set.seed(1)

sp <- pmlPart(edge ~ rate + inv, fits, weight=weight)
sp

sp2 <- pmlPart(~ edge + inv, fits, weight=weight)
sp2
AIC(sp2)

sp3 <- pmlPen(sp2, lambda = 2)
AIC(sp3)
```

read.aa *Read Amino Acid Sequences in a File*

Description

This function reads amino acid sequences in a file, and returns a matrix list of DNA sequences with the names of the taxa read in the file as rownames.

Usage

```
read.aa(file, format = "interleaved", skip = 0,  
        nlines = 0, comment.char = "#", seq.names = NULL)
```

Arguments

file	a file name specified by either a variable of mode character, or a double-quoted string.
format	a character string specifying the format of the DNA sequences. Three choices are possible: "interleaved", "sequential", or "fasta", or any unambiguous abbreviation of these.
skip	the number of lines of the input file to skip before beginning to read data.
nlines	the number of lines to be read (by default the file is read until its end).
comment.char	a single character, the remaining of the line after this character is ignored.
seq.names	the names to give to each sequence; by default the names read in the file are used.

Value

a matrix of amino acid sequences.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Anonymous. FASTA format description. <http://www.ncbi.nlm.nih.gov/BLAST/fasta.html>

Felsenstein, J. (1993) Phylip (Phylogeny Inference Package) version 3.5c. Department of Genetics, University of Washington. <http://evolution.genetics.washington.edu/phylip/phylip.html>

See Also

[read.dna](#), [read.GenBank](#), [phyDat](#), [read.alignment](#)

`SH.test`*Shimodaira-Hasegawa Test*

Description

This function computes the Shimodaira–Hasegawa test for a set of trees.

Usage

```
SH.test(..., B = 10000, data=NULL)
```

Arguments

`...` either a series of objects of class "pml" separated by commas, a list containing such objects or an object of class "pmlPart".

`B` the number of bootstrap replicates.

`data` an object of class "phyDat".

Value

a numeric vector with the P-value associated with each tree given in

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Shimodaira, H. and Hasegawa, M. (1999) Multiple comparisons of log-likelihoods with applications to phylogenetic inference. *Molecular Biology and Evolution*, **16**, 1114–1116.

See Also

[pml](#), [pmlPart](#), [pmlCluster](#)

Examples

```
data(Laurasiatherian)
dm <- dist.logDet(Laurasiatherian)
tree1 <- NJ(dm)
tree2 <- unroot(upgma(dm))
fit1 <- pml(tree1, Laurasiatherian)
fit2 <- pml(tree2, Laurasiatherian)
fit1 <- optim.pml(fit1) # optimise edge weights
fit2 <- optim.pml(fit2)
SH.test(fit1, fit2)
## Not run:
example(pmlPart)
```

```
SH.test(sp, B=1000)
## End(Not run)
```

simSeq *Simulate sequences.*

Description

Simulate sequences for a given evolutionary tree.

Usage

```
simSeq(tree, l=1000, Q=NULL, bf=NULL, rootseq=NULL, type="DNA",
       model="", levels=NULL, rate=1, ancestral=FALSE)
```

Arguments

tree	a phylogenetic tree tree, an object of class phylo.
l	length of the sequence to simulate.
Q	the rate matrix.
bf	base frequencies.
rootseq	a vector of length l containing the root sequence, other root sequence is randomly generated.
type	Type of sequences ("DNA", "AA" or "USER").
model	Amino acid models: one of "WAG", "JTT", "Dayhoff" or "LG"
levels	levels takes a character vector of the different bases, default is for nucleotide sequences, only used when type = "USER".
rate	rate.
ancestral	Return ancestral sequences?

Details

simSeq simulates sequence alignments. So far rate variation is not yet implemented, but one can combine different alignments having their own rate. In fact it is possible to generate DNA, RNA, amino acid, or 0/1.

Value

simSeq returns an object of class phyDat.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also[phyDat](#)**Examples**

```
tree = rtree(5)
plot(tree)
nodelabels()

# Example for simple DNA alignment
data = simSeq(tree, l = 10, type="DNA", bf=c(.1,.2,.3,.4), Q=1:6)
as.character(data)

# Example to simulate discrete Gamma rate variation
rates = phangorn::discrete.gamma(1,4)
data1 = simSeq(tree, l = 100, type="AA", model="WAG", rates[1])
data2 = simSeq(tree, l = 100, type="AA", model="WAG", rates[2])
data3 = simSeq(tree, l = 100, type="AA", model="WAG", rates[3])
data4 = simSeq(tree, l = 100, type="AA", model="WAG", rates[4])
data <- c(data1,data2, data3, data4)

write.phyDat(data, file="temp.dat", format="sequential",nbc0l = -1, colsep = "")
unlink("temp.dat")
```

splitsNetwork

Phylogenetic Network

Description

splitsNetwork estimates a splits graph from a distance matrix.

Usage

```
splitsNetwork(dm, gamma=.1, lambda=1e-6, weight=NULL)
```

Arguments

dm	A distance matrix.
gamma	penalty value for the L1 constraint.
lambda	penalty value for the L2 constraint.
weight	a vector of weights.

Details

`splitsNetwork` fits phylogenetic networks using L1, L2 and non-negativity constraints. The function minimises the penalised least squares

$$\beta = \min \sum (dm - X\beta)^2 + \lambda \|\beta\|_2^2$$

with respect to

$$\|\beta\|_1 \leq \gamma, \beta \geq 0$$

where X is a design matrix constructed with `designSplits`. External edges are fitted without constraints.

Value

`splitsNetwork` returns a matrix. The first column contains the indices of the splits, the second column an unconstrained fit without penalty terms and the third column the constraint fit.

Author(s)

Klaus Schliep (klaus.schliep@gmail.com)

References

K. P. Schliep (2009). Some Applications of statistical phylogenetics (PhD Thesis)

See Also

[distanceHadamard](#), [designTree](#)

Examples

```
data(yeast)
dm = dist.ml(yeast)
fit = splitsNetwork(dm)
write.nexus.splits(fit)
```

treedist

Distances between trees

Description

`treedist` computes different tree distance methods and `RF.dist` the Robinson-Foulds distance.

Usage

```
treedist(tree1, tree2)
RF.dist(tree1, tree2, check.labels=TRUE)
```

Arguments

tree1 A phylogenetic tree.
 tree2 A phylogenetic tree.
 check.labels compares labels of the trees.

Value

treedist returns a vector containing the following tree distance methods

symmetric.difference
 symmetric.difference or Robinson-Foulds distance

branch.score.difference
 branch.score.difference

path.difference
 path.difference

weighted.path.difference
 weighted.path.difference

Author(s)

Klaus P. Schliep (klaus.schliep@gmail.com)

References

Steel M. A. and Penny P. (1993) *Distributions of tree comparison metrics - some new results*, Syst. Biol.,42(2), 126-141

 upgma

 UPGMA

Description

UPGMA clustering. Just a wrapper function around [hclust](#).

Usage

```
upgma(D, method = "average", ...)
```

Arguments

D A distance matrix.

method The agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid". The default is "average".

... Further arguments passed to or from other methods.

Value

A phylogenetic tree of class `phylo`.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[hclust](#), [NJ](#), [as.phylo](#)

Examples

```
data(Laurasiatherian)
dm = dist.logDet(Laurasiatherian)
tree = upgma(dm)
plot(tree)
```

yeast

Yeast alignment (Rokas et al.)

Description

Alignment of 106 genes of 8 different species of yeast.

Usage

```
data(yeast)
```

References

Rokas, A., Williams, B. L., King, N., and Carroll, S. B. (2003) Genome-scale approaches to resolving incongruence in molecular phylogenies. *Nature*, **425**(6960): 798–804

Examples

```
data(yeast)
str(yeast)
```

Index

*Topic **IO**

read.aa, 20

*Topic **classif**

dfactorial, 4

treedist, 25

*Topic **cluster**

allTrees, 1

bootstrap.pml, 2

designTree, 3

dist.hamming, 5

distanceHadamard, 6

hadamard, 7

NJ, 9

nni, 10

parsimony, 10

phyDat, 12

pml, 13

pmlCluster, 16

pmlMix, 17

pmlPart, 19

simSeq, 23

splitsNetwork, 24

upgma, 26

*Topic **datasets**

chloroplast, 3

Laurasiatherian, 8

yeast, 27

*Topic **models**

SH.test, 22

allSitePattern (*phyDat*), 12

allTrees, 1

as.character.phyDat (*phyDat*), 12

as.data.frame.phyDat (*phyDat*), 12

as.DNAbin, 13

as.DNAbin.phyDat (*phyDat*), 12

as.phyDat (*phyDat*), 12

as.phylo, 27

bootstrap.phyDat (*bootstrap.pml*),

2

bootstrap.pml, 2

chloroplast, 3

designSplits (*designTree*), 3

designTree, 3, 25

dfactorial, 4

dist.dna, 5, 9

dist.hamming, 5, 9

dist.logDet (*dist.hamming*), 5

dist.ml (*dist.hamming*), 5

distanceHadamard, 4, 6, 7, 8, 25

DNAbin, 13

factorial, 4

fastme, 4, 9

fhm (*hadamard*), 7

fitch (*parsimony*), 10

h2st (*hadamard*), 7

h4st (*hadamard*), 7

hadamard, 6, 7

hclust, 26, 27

Laurasiatherian, 8

ldfactorial (*dfactorial*), 4

NJ, 9, 11, 27

nj, 9

nni, 10, 11

optim.parsimony (*parsimony*), 10

optim.pml, 3

optim.pml (*pml*), 13

parsimony, 10

phyDat, 12, 21, 24

pml, 3, 11, 13, 16, 18, 20, 22

pmlCluster, 16, 18, 20, 22

pmlCluster2 (*pmlCluster*), 16

pmlMix, 13, 16, 17, 20
pmlPart, 16, 18, 19, 22
pmlPen (*pmlMix*), 17
PNJ (*parsimony*), 10

read.aa, 13, 20
read.alignment, 21
read.dna, 12, 13, 21
read.GenBank, 21
read.nexus.data, 12, 13
read.phyDat (*phyDat*), 12
RF.dist (*treedist*), 25
rNNI (*nni*), 10
rSPR (*nni*), 10

sankoff (*parsimony*), 10
SH.test, 16, 22
simSeq, 23
splitsNetwork, 24

treedist, 25

UNJ (*NJ*), 9
upgma, 9, 26

write.nexus.splits (*hadamard*), 7
write.phyDat (*phyDat*), 12

yeast, 27