

Package ‘phangorn’

January 2, 2012

Title Phylogenetic analysis in R

Version 1.5-1

Date 2011-12-12

Author Klaus Schliep, Emmanuel Paradis

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>

Depends R (>= 2.14.0), ape (>= 2.8), igraph, Matrix

Imports quadprog

Suggests parallel, seqLogo, rgl, seqinr, xtable, flashClust

License GPL (>= 2)

Repository CRAN

Date/Publication 2011-12-13 08:36:37

R topics documented:

phangorn-package	2
allTrees	3
Ancestors	4
ancestral.pml	5
as.splits	6
bab	7
bootstrap.pml	8
chloroplast	10
consensusNet	10
designTree	12
dfactorial	13
dist.hamming	14

distanceHadamard	15
getClans	16
hadamard	18
Laurasiatherian	20
lento	20
midpoint	21
modelTest	22
NJ	24
nmi	25
parsimony	26
phyDat	27
plot.networx	29
pml	31
pmlCluster	34
pmlMix	35
pmlPart	37
read.aa	39
SH.test	40
simSeq	41
splitsNetwork	42
treedist	43
upgma	44
yeast	45

Index	46
--------------	-----------

phangorn-package	<i>Phylogenetic analysis in R</i>
------------------	-----------------------------------

Description

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

The complete list of functions can be displayed with `library(help = phangorn)`.

Further information is available in two vignettes.

Trees	Constructing phylogenetic trees (source, pdf)
phangorn-specials	Advanced features (source, pdf)
Ancestral	Ancestral sequence reconstruction (source, pdf)

The first vignette (to display type `vignette('Trees')`) gives an introduction in phylogenetic analysis with phangorn, and the second vignette covers more advanced feature like defining special character spaces.

Author(s)

Klaus Schliep

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

References

Schliep K.P. (2011) phangorn: Phylogenetic analysis in R. *Bioinformatics*, 27(4) 592-593

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]])
```

Ancestors	<i>tree utility function</i>
-----------	------------------------------

Description

Functions for describing relationships among phylogenetic nodes.

Usage

```
Ancestors(x, node, type=c("all", "parent"))
Children(x, node)
Siblings(x, node, include.self=FALSE)
Descendants(x, node, type=c("tips", "children", "all"))
```

Arguments

x	a tree (a phylo object).
node	an integer corresponding to a node ID
type	specify whether to return just direct ancestor / parents or all
include.self	whether to include self in list of siblings

Value

a vector containing the indices of the nodes.

See Also

these functions are inspired by `treewalk` in `phylobase` package

Examples

```
tree = rtree(10)
plot(tree, show.tip.label = FALSE)
nodeLabels()
tiplabels()
Ancestors(tree, 1, "all")
Children(tree, 11)
Descendants(tree, 11, "tips")
Siblings(tree, 3)
```

ancestral.pml *Ancestral character reconstruction.*

Description

Marginal reconstruction of the ancestral character states.

Usage

```
ancestral.pml(object, type = c("ml", "bayes"))
ancestral.pars(tree, data, type = c("MPR", "ACCTRAN"))
pace(tree, data, type = c("MPR", "ACCTRAN"))
plotAnc(tree, data, i, col=NULL, ...)
```

Arguments

object	an object of class pml
tree	a tree, i.e. an object of class pml
data	an object of class phyDat
type	method used to assign characters to internal nodes, see details.
i	plots the i-th character of the data.
col	a vector containing the colors for all possible states.
...	Further arguments passed to or from other methods.

Details

The argument "type" defines the criterion to assign the internal nodes. For `ancestral.pml` so far "ml" and (empirical) "bayes" and for `ancestral.pars` "MPR" and "ACCTRAN" are possible.

With parsimony reconstruction one has to keep in mind that there will be often no unique solution. For further details see `vignette("Ancestral")`.

Value

An object of class "phyDat", containing the ancestral states of all nodes.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

- Felsenstein, J. (2004). *Inferring Phylogenies*. Sinauer Associates, Sunderland.
- Swofford, D.L., Maddison, W.P. (1987) Reconstructing ancestral character states under Wagner parsimony. *Math. Biosci.* **87**: 199–229
- Yang, Z. (2006). *Computational Molecular evolution*. Oxford University Press, Oxford.

See Also

pml, parsimony, ace, root

Examples

```
example(NJ)
fit = pml(tree, Laurasiatherian)
anc.ml = ancestral.pml(fit, type = "ml")
anc.p = ancestral.pars(tree, Laurasiatherian)
## Not run:
require(seqLogo)
seqLogo( t(subset(anc.ml, 48, 1:20)[[1]]), ic.scale=FALSE)
seqLogo( t(subset(anc.p, 48, 1:20)[[1]]), ic.scale=FALSE)

## End(Not run)
plotAnc(tree, anc.ml, 1)
```

as.splits

Splits representation of graphs and trees.

Description

as.splits produces a list of splits or bipartitions.

Usage

```
as.splits(x, ...)
## S3 method for class 'phylo'
as.splits(x, ...)
## S3 method for class 'multiPhylo'
as.splits(x, ...)
## S3 method for class 'splits'
print(x, maxp = getOption("max.print"), zero.print = ".",
      one.print = "|", ...)
compatible(obj)
allSplits(k, labels = NULL)
```

Arguments

x	An object of class phylo or multiPhylo.
maxp	integer, default from options(max.print), influences how many entries of large matrices are printed at all.
zero.print	character which should be printed for zeroes.
one.print	character which should be printed for ones.
...	Further arguments passed to or from other methods.
obj	an object of class splits.
k	number of taxa.
labels	names of taxa.

Value

`as.splits` returns an object of class `splits`, which is mainly a list of splits and some attributes.
`compatible` return a lower triangular matrix where an 1 indicates that two splits are incompatible.

Note

The internal representation is likely to change.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[prop.part](#), [lento](#), [distanceHadamard](#)

Examples

```
as.splits(rtree(5))
```

bab

Branch and bound for finding all most parsimonious trees

Description

bab finds all most parsimonious trees.

Usage

```
bab(data, tree = NULL, trace = 1, ...)
```

Arguments

<code>data</code>	an object of class <code>phyDat</code> .
<code>tree</code>	a phylogenetic tree an object of class <code>phylo</code> , otherwise a pratchet search is performed.
<code>trace</code>	defines how much information is printed during optimisation.
<code>...</code>	Further arguments passed to or from other methods

Details

This implementation is very slow and depending on the data may take very long time. In the worst case all $(2n-5)!!$ possible trees have to be examined. For 10 species there are already 2027025 tip-labelled unrooted trees. It only uses some basic strategies to find a lower and upper bounds similar to penny from `phylip`. It uses a very basic heuristic approach of MinMax Squeeze (Holland et al. 2005) to improve the lower bound. On the positive side `bab` is not like many other implementations restricted to binary or nucleotide data.

Value

bab returns all most parsimonious trees in an object of class multiPhylo.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com> based on work on Liam Revell

References

Hendy, M.D. and Penny D. (1982) Branch and bound algorithms to determine minimal evolutionary trees. *Math. Biosc.* **59**, 277-290

Holland, B.R., Huber, K.T. Penny, D. and Moulton, V. (2005) The MinMax Squeeze: Guaranteeing a Minimal Tree for Population Data, *Molecular Biology and Evolution*, **22**, 235–242

White, W.T. and Holland, B.R. (2011) Faster exact maximum parsimony search with XMP. *Bioinformatics*, **27(10)**, 1359–1367

See Also

[pratchet](#), [dfactorial](#)

Examples

```
data(yeast)
dfactorial(11)
trees <- bab(yeast)
```

bootstrap.pml

Bootstrap

Description

bootstrap.pml performs (non-parametric) bootstrap analysis and bootstrap.phyDat produces a list of bootstrapped data sets. plotBS plots a phylogenetic tree with the with the bootstrap values assigned to the (internal) edges.

Usage

```
bootstrap.pml(x, bs = 100, trees = TRUE, multicore=FALSE, ...)
bootstrap.phyDat(x, FUN, bs = 100, multicore=FALSE, ...)
plotBS(tree, BStrees, type="unrooted", bs.col="black", bs.adj=NULL, ...)
```

Arguments

x	an object of class pml or phyDat.
bs	number of bootstrap samples.
trees	return trees only (default) or whole pml objects.
multicore	logical, if TRUE analysis is performed in parallel (see details).
...	further parameters used by optim.pml or plot.phylo.
FUN	the function to estimate the trees.
tree	The tree on which edges the bootstrap values are plotted.
BStrees	a list of trees (object of class "multiPhylo").
type	the type of tree to plot, so far "cladogram", "phylogram" and "unrooted" are supported.
bs.col	color of bootstrap support labels.
bs.adj	one or two numeric values specifying the horizontal and vertical justification of the bootstrap labels.

Details

It is possible that the bootstrap is performed in parallel, with help of the multicore package. Unfortunately the multicore package does not work under windows or with GUI interfaces ("aqua" on a mac). However it will speed up nicely from the command line ("X11").

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>

References

- Felsenstein J. (1985) Confidence limits on phylogenies. An approach using the bootstrap. *Evolution* **39**, 783–791
- Penny D. and Henny M.D. (1985) Testing methods evolutionary tree construction. *Cladistics* **1**, 266–278
- Penny D. and Henny M.D. (1986) Estimating the reliability of evolutionary trees. *Molecular Biology and Evolution* **3**, 403–417

See Also

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

Examples

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

set.seed(1)
bs <- bootstrap.pml(fit, bs=100, optNni=TRUE)
plotBS(fit$tree,bs)

## End(Not run)
```

chloroplast

Chloroplast alignment

Description

Amino acid alignment of 15 genes of 19 different chloroplast.

Usage

```
data(yeast)
```

Examples

```
data(chloroplast)
chloroplast
```

consensusNet

Computes a network object from a collection of splits.

Description

Computes a network object from a collection of splits.

Usage

```
consensusNet(obj, prob=.3, ...)
as.network(x, ...)
## S3 method for class 'splits'
as.network(x, ...)
```

Arguments

obj	An object of class multiPhylo.
prob	the proportion a split has to be present in all trees to be represented in the network.
x	An object of class splits.
...	Further arguments passed to or from other methods.

Value

as.networx returns an object of class networkx. This is just an intermediate to plot phylogenetic networks with igraph.

Note

The internal representation is likely to change.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Holland B.R., Huber K.T., Moulton V., Lockhart P.J. (2004) Using consensus networks to visualize contradictory evidence for species phylogeny. *Molecular Biology and Evolution*, **21**, 1459–61

See Also

[splitsNetwork](#), [lento](#), [distanceHadamard](#)

Examples

```
data(Laurasiatherian)
set.seed(1)
bs <- bootstrap.phyDat(Laurasiatherian, FUN = function(x)nj(dist.hamming(x)), bs=100, multicore=FALSE)
class(bs) <- 'multiPhylo'
cnet = consensusNet(bs, .3)
plot(cnet, show.edge.label=TRUE)
open3d()
plot(cnet, show.tip.label=FALSE, show.nodes=TRUE)

set.seed(1)
tree1 = rtree(20, rooted=FALSE)
sp = as.splits(rNNI(tree1, n=10))
net = as.networx(sp)
open3d()
plot(net)
```

`designTree`*Compute a design matrix or non-negative LS*

Description

`nnls.tree` estimates the branch length using non-negative least squares given a tree and a distance matrix. `designTree` and `designSplits` compute design matrices for the estimation of edge length of (phylogenetic) trees using linear models. For larger trees a sparse design matrix can save a lot of memory.

Usage

```
designTree(tree, method = "unrooted", sparse=FALSE, ...)
designSplits(x, splits = "all", ...)
nnls.tree(dm, tree, rooted=FALSE, trace=1)
```

Arguments

<code>tree</code>	an object of class <code>phylo</code>
<code>method</code>	design matrix for an "unrooted" or "rooted" ultrametric tree.
<code>sparse</code>	return a sparse design matrix.
<code>x</code>	number of taxa.
<code>splits</code>	one of "all", "star".
<code>dm</code>	a distance matrix.
<code>rooted</code>	compute a "rooted" or "unrooted" tree.
<code>trace</code>	defines how much information is printed during optimisation.
<code>...</code>	further arguments, passed to other methods.

Value

`nnls.tree` return a tree, i.e. an object of class `phylo`. `designTree` and `designSplits` a matrix, possibly sparse.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

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

Examples

```
example(NJ)
dm <- as.matrix(dm)
y <- dm[lower.tri(dm)]
X <- designTree(tree)
lm(y~X-1)
# avoids negative edge weights
tree2 = nnls.tree(dm, tree)
```

dfactorial

Arithmetic Operators

Description

double factorial function

Usage

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

Arguments

x a numeric scalar 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", pairwise.deletion = FALSE, bf=NULL, Q=NULL, ...)
```

Arguments

x	An object of class phyDat
ratio	Compute uncorrected ('p') distance or character difference.
model	One of "JC69", "WAG", "JTT", "LG" or "Dayhoff"
pairwise.deletion	A logical indicating whether to delete the sites with missing data (or ambiguous states) in a pairwise way. The default is to delete the sites with at least one missing data for all sequences.
bf	A vector of base frequencies.
Q	A vector containing the lower triangular part of the rate matrix.
...	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	<i>Distance Hadamard</i>
------------------	--------------------------

Description

Distance Hadamard produces spectra of splits from a distance matrix.

Usage

```
distanceHadamard(dm, eps=0.001)
```

Arguments

dm	A distance matrix.
eps	Threshold value for splits.

Value

distanceHadamard returns a matrix. The first column contains the distance spectra, the second one the edge-spectra. If eps is positive an object of with all splits greater eps is returned.

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](#), [lento](#), [plot.networkx](#)

Examples

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

getClans

*Clans, slices and clips***Description**

Functions for clanistics to compute clans, slices, clips for unrooted trees and functions to quantify the fragmentation of trees.

Usage

```
getClans(tree)
getClips(tree, all=TRUE)
getSlices(tree)
getDiversity(tree, x, norm=TRUE, var.names = NULL, labels="new")
diversity(tree, X)
```

Arguments

tree	An object of class phylo or multiPhylo (getDiversity).
all	A logical, return all or just the largest clip.
x	An object of class phyDat.
norm	A logical, return Equitability Index (default) or Shannon Diversity.
var.names	A vector of variable names.
labels	see details.
X	a data.frame

Details

Every split in an unrooted tree defines two complementary clans. Thus for an unrooted binary tree with n leaves there are $2n - 3$ edges, and therefore $4n - 6$ clans (including n trivial clans containing only one leaf).

Slices are defined by a pair of splits or tripartitions, which are not clans. The number of distinguishable slices for a binary tree with n tips is $2n^2 - 10n + 12$.

A clip is a different type of partition, defining groups of leaves that are related in terms of evolutionary distances and not only topology. Namely, clips are groups of leaves for which all pairwise path-length distances are smaller than a given threshold value (Lapointe et al. 2010). There exists different numbers of clips for different thresholds, the largest (and trivial) one being the whole tree. There is always a clip containing only the two leaves with the smallest pairwise distance.

Clans, slices and clips can be used to characterize how well a vector of categorical characters (natives/intruders) fit on a tree. We will follow the definitions of Lapointe et al.(2010). A complete clan is a clan that contains all leaves of a given state/color, but can also contain leaves of another state/color. A clan is homogeneous if it only contains leaves of one state/color.

getDiversity computes either the

Shannon Diversity: $H = -\sum_{i=1}^k (N_i/N) \log(N_i/N)$, $N = \sum_{i=1}^k N_i$

or the

Equitability Index: $E = H/\log(N)$

where N_i are the sizes of the k largest homogeneous clans of intruders. If the categories of the data can be separated by an edge of the tree then the E-value will be zero, and maximum equitability ($E=1$) is reached if all intruders are in separate clans. `getDiversity` computes these Intruder indices for the whole tree, complete clans and complete slices. Additionally the parsimony scores (p-scores) are reported. The p-score indicates if the leaves contain only one color (p-score=0), if the leaves can be separated by a single split (perfect clan, p-score=1) or by a pair of splits (perfect slice, p-score=2).

So far only 2 states are supported (native, intruder), however it is also possible to recode several states into the native or intruder state using contrasts, for details see section 2 in vignette("phangorn-specials"). Furthermore unknown character states are coded as ambiguous character, which can act either as native or intruder minimizing the number of clans or changes (in parsimony analysis) needed to describe a tree for given data.

Set attribute labels to "old" for analysis as in Schliep et al. (2010) or to "new" for names which are more intuitive.

`diversity` returns a data.frame with the parsimony score for each tree and each levels of the variables in X . X has to be a data.frame where each column is a factor and the rownames of X correspond to the tips of the trees.

Value

`getClans`, `getSlices` and `getClips` return a matrix of partitions, a matrix of ones and zeros where rows correspond to a clan, slice or clip and columns to tips. A one indicates that a tip belongs to a certain partition.

`getDiversity` returns a list with tree object, the first is a data.frame of the equitability index or Shannon divergence and parsimony scores (p-score) for all trees and variables. The data.frame has two attributes, the first is a splits object to identify the taxa of each tree and the second is a splits object containing all partitions that perfectly fit.

Author(s)

Klaus Schliep <klaus.schliep@snv.jussieu.fr>

Francois-Joseph Lapointe <francois-joseph.lapointe@umontreal.ca>

References

Lapointe, F.-J., Lopez, P., Boucher, Y., Koenig, J., Bapteste, E. (2010) Clanistics: a multi-level perspective for harvesting unrooted gene trees. *Trends in Microbiology* 18: 341-347

Wilkinson, M., McInerney, J.O., Hirt, R.P., Foster, P.G., Embley, T.M. (2007) Of clades and clans: terms for phylogenetic relationships in unrooted trees. *Trends in Ecology and Evolution* 22: 114-115

Schliep, K., Lopez, P., Lapointe F.-J., Bapteste E. (2011) Harvesting Evolutionary Signals in a Forest of Prokaryotic Gene Trees, *Molecular Biology and Evolution* 28(4): 1393-1405

See Also

[parsimony](#), Consistency index [CI](#), Retention index [RI](#), [phyDat](#)

Examples

```

set.seed(111)
tree = rtree(10)
getClans(tree)
getClips(tree, all=TRUE)
getSlices(tree)

set.seed(123)
trees = rmtree(10, 20)
X = matrix(sample(c("red", "blue", "violet"), 100, TRUE, c(.5,.4, .1)), ncol=5,
           dimnames=list(paste('t',1:20, sep=""), paste('Var',1:5, sep="_")))
x = phyDat(X, type = "USER", levels = c("red", "blue"), ambiguity="violet")
plot(trees[[1]], "u", tip.color = X[trees[[1]]$tip,1]) # intruders are blue

(divTab <- getDiversity(trees, x, var.names=colnames(X)))
summary(divTab)

```

hadamard

*Hadamard Matrices and Fast Hadamard Multiplication***Description**

A collection of functions to perform Hadamard conjugation.

Usage

```

hadamard(x)
fhm(v)
h2st(obj, eps=0.001)
h4st(obj, levels = c("a", "c", "g", "t"))
write.nexus.splits(obj, file="", weights=NULL)

```

Arguments

x	a vector of length 2^n , where n is an integer.
v	a vector of length 2^n , where n is an integer.
obj	a data.frame or character matrix, typical a sequence alignment.
eps	Threshold value for splits.
levels	levels of the sequences.
file	a file name.
weights	Edge weights.

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](#), [lento](#), [plot.networkx](#)

Examples

```
H = hadamard(3)
v = 1:8
H
fhm(v)

data(yeast)
dat = as.character(yeast)
# RY-coding
dat2 = dat
dat2[dat=="a" | dat=="g"] = "r"
dat2[dat=="c" | dat=="t"] = "y"
dat2 = phyDat(dat2, type="USER", levels=c("r","y"), ambiguity=NULL)
fit2 = h2st(dat2)
lento(fit2)

# write.nexus.splits(fit2, file = "test.nxs")
# read this file into Spectronet or Splitstree to show the network
## Not run:
dat4 = phyDat(dat, type="USER", levels=c("a","c", "g", "t"), ambiguity=NULL)
fit4 = h4st(dat4)

par(mfrow=c(3,1))
lento(fit4[[1]], main="Transversion")
lento(fit4[[2]], main="Transition 1")
lento(fit4[[3]], main="Transition 2")
```

```
## End(Not run)
```

Laurasiatherian	<i>Laurasiatherian data (AWCMEE)</i>
-----------------	--------------------------------------

Description

Laurasiatherian RNA sequence data

Usage

```
data(Laurasiatherian)
```

Source

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

Examples

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

lento	<i>Lento plot</i>
-------	-------------------

Description

The lento plot represents support and conflict of splits/bipartitions.

Usage

```
lento(obj, xlim = NULL, ylim = NULL, main = "Lento plot", sub = NULL, xlab = NULL, ylab = NULL, bipart=TRUE)
```

Arguments

obj	an object of class phylo, multiPhylo or splits
xlim	graphical parameter
ylim	graphical parameter
main	graphical parameter
sub	graphical parameter
xlab	graphical parameter
ylab	graphical parameter
bipart	plot bipartition information.
trivial	logical, whether to present trivial splits (default is FALSE).
...	Further arguments passed to or from other methods.

Value

lento returns a plot.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Lento, G.M., Hickson, R.E., Chambers G.K., and Penny, D. (1995) Use of spectral analysis to test hypotheses on the origin of pinnipeds. *Molecular Biology and Evolution*, **12**, 28-52.

See Also

[as.splits](#), [hadamard](#)

Examples

```
data(yeast)
yeast.ry = acgt2ry(yeast)
splits.h = h2st(yeast.ry)
lento(splits.h, trivial=TRUE)
```

midpoint

Tree manipulation

Description

midpoint performs midpoint rooting of a tree. pruneTree produces a consensus tree.

Usage

```
midpoint(tree)
pruneTree(tree, ..., FUN = ">=")
getRoot(tree)
```

Arguments

tree	an object of class phylo
FUN	a function evaluated on the nodelabels, result must be logical.
...	further arguments, passed to other methods.

Details

pruneTree prunes back a tree and produces a consensus tree, for trees already containing node-labels. It assumes that nodelabels are numerical or character generated from numerals, it uses `as.numeric(as.character(tree$node.labels))` to convert them. midpoint so far does not transform node.labels properly.

Value

pruneTree and midpoint a tree. getRoot returns the root node.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

[consensus](#), [root](#), [di2multi](#)

Examples

```
tree = unroot(rtree(10))
tree$node.label = c("", round(runif(tree$Nnode-1), 3))

tree2 = midpoint(tree)
tree3 = pruneTree(tree, .5)

par(mfrow = c(3,1))
plot(tree, show.node.label=TRUE)
plot(tree2, show.node.label=TRUE)
plot(tree3, show.node.label=TRUE)
```

modelTest

ModelTest

Description

Comparison of different substitution models

Usage

```
modelTest(object, tree=NULL, model = c("JC", "F81", "K80", "HKY", "SYM", "GTR"), G = TRUE, I = TRUE, k =
```

Arguments

object	an object of class phyDat or pml
tree	a phylogenetic tree.
model	a vector containing the substitution models to compare with each other
G	logical, TRUE (default) if (discrete) Gamma models should be tested
I	logical, TRUE (default) if invariant sites should be tested
k	number of rate classes
control	A list of parameters for controlling the fitting process.
multicore	logical, whether models should be estimated in parallel.

Details

modelTest estimates all the specified models for a given tree and data. When the multicore package is available, the computations are done in parallel. This is only possible without GUI interface and under linux. Only nucleotide models are tested so far.

Value

A data.frame containing the log-likelihood, AIC and BIC all tested models. The data.frame has an attributes "env" which is an environment which contains all the trees, the data and the calls to allow get the estimated models, e.g. as a starting point for further analysis (see example).

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Posada, D. and Crandall, K.A. (1998) MODELTEST: testing the model of DNA substitution. *Bioinformatics* **14**(9): 817-818

Posada, D. (2008) jModelTest: Phylogenetic Model Averaging. *Molecular Biology and Evolution* **25**: 1253-1256

See Also

[pml](#), [anova](#)

Examples

```
## Not run:
example(NJ)
(mT <- modelTest(Laurasiatherian, tree))

# some R magic
env = attr(mT, "env")
ls(env=env)
(F81 <- get("F81+G", env)) # a call
eval(F81, env=env)

data(chloroplast)
(mTAA <- modelTest(chloroplast, model=c("JTT", "WAG", "LG")))

## End(Not run)
```

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 apart 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, k=NULL)
rNNI(tree, moves=1, n=1)
```

Arguments

tree	A phylogenetic tree, object of class phylo.
moves	Number of tree rearrangements to be transformed on a tree.
n	Number of trees to be simulated.
k	If defined just SPR of distance k are performed.

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 Nearest Neighbor Interchange (NNI) rearrangements. pace returns a (logical) matrix of the ancestral states of the root node. CI and RI computes the consistency and retention index.

Usage

```
parsimony(tree, data, method="fitch", ...)
optim.parsimony(tree, data, method="fitch", cost=NULL, trace=1,...)
pratchet(data, start=NULL, k=20, np=1, trace=1, all=FALSE, method="fitch", multicore=FALSE, ...)
fitch(tree, data, site = "pscore")
sankoff(tree, data, cost = NULL, site = "pscore")
CI(tree, data)
RI(tree, data)
acctrans(tree, data)
```

Arguments

data	A object of class phyDat containing sequences.
tree	tree to start the nni search from.
method	one of 'fitch' or 'sankoff'.
cost	A cost matrix for the transitions between two states.
site	return either 'pscore' or 'site' wise parsimony scores.
trace	defines how much information is printed during optimisation.
start	a starting tree can be supplied.
k	number of rounds in the ratchet.
np	number of (parallel) bootstraps (see details).
multicore	logical, if TRUE analysis is performed in parallel (see details).
all	return all equally good trees or just one of them.
...	Further arguments passed to or from other methods (e.g. model="sankoff" and cost matrix).

Details

On platforms where the multicore package is available the parsimony ratchet may find solutions faster. To do this set the parameter np to the number of cores available. See also details in [bootstrap.pml](#).

Value

parsimony returns the maximum parsimony score (pscore). optim.parsimony returns a tree after NNI rearrangements. pratchet returns a tree or list of trees containing the best tree(s) found during the search. acctran returns a tree with edge length according to the ACCTRAN criterion.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

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

Nixon, K. (1999) The Parsimony Ratchet, a New Method for Rapid Parsimony Analysis. *Cladistics* **15**, 407-414

See Also

[bab](#), [ancestral.pml](#), [nni](#), [NJ](#), [pml](#), [getClans](#), [ancestral.pars](#), [bootstrap.pml](#)

Examples

```
data(Laurasiatherian)
dm = dist.logDet(Laurasiatherian)
tree = NJ(dm)
parsimony(tree, Laurasiatherian)
treeNNI <- optim.parsimony(tree, Laurasiatherian)
treeRatchet <- pratchet(Laurasiatherian, start=tree)
treeRatchet <- acctran(treeRatchet, Laurasiatherian)

plot(midpoint(treeRatchet))
add.scale.bar()

parsimony(c(tree,treeNNI, treeRatchet), Laurasiatherian)
```

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, allLevels = TRUE, ...)
## S3 method for class 'phyDat'
as.data.frame(x, ...)
## S3 method for class 'phyDat'
as.DNABin(x, ...)
## S3 method for class 'phyDat'
subset(x, subset, select, site.pattern = TRUE, ...)
allSitePattern(n, levels=c("a","c","g","t"), names=NULL)
acgt2ry(obj)

```

Arguments

data	An object containing sequences.
x	An object containing sequences.
type	Type of sequences ("DNA", "AA", "CODON" or "USER").
levels	Level attributes.
return.index	If TRUE returns a index of the site patterns.
file	A file name.
format	File format of the sequence alignment (see details).
n	Number of sequences.
names	Names of sequences.
subset	a subset of taxa.
select	a subset of characters.
site.pattern	select site pattern or sites.
allLevels	return original data.
obj	as object of class phyDat
...	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 useful in simulation studies. For further details see the vignette `phangorn-specials`.

`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.

The generic function `c` can be used to combine sequences and `unique` to get all unique sequences or unique haplotypes.

`acgt2ry` converts a phyDat object of nucleotides into an binary ry-coded dataset.

There is a more detailed example for specifying USER defined data formats in the vignette advanced features.

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 [pm1Mix](#) for the use of `allSitePattern`

Examples

```
data(Laurasiatherian)
class(Laurasiatherian)
Laurasiatherian
subset(Laurasiatherian, subset=1:5)
# transform into old ape format
LauraChar <- as.character(Laurasiatherian)
# and back
Laura <- phyDat(LauraChar, return.index=TRUE)
all.equal(Laurasiatherian, Laura)
allSitePattern(5)
```

plot.networx

Plot phylogenetic networks

Description

`plot.networx` plot phylogenetic network or split graphs.

Usage

```
## S3 method for class 'networx'
plot(x, type = "3D", show.tip.label = TRUE, show.edge.label=FALSE, show.nodes = FALSE, tip.color = "blue")
```

Arguments

x	an object of class "networkx"
type	"3D" to plot using rgl or "2D" in the normal device.
show.tip.label	a logical indicating whether to show the tip labels on the graph (defaults to TRUE, i.e. the labels are shown).
show.edge.label	a logical indicating whether to show the tip labels on the graph.
show.nodes	a logical indicating whether to show the nodes (see example).
tip.color	the colours used for the tip labels.
edge.color	the colours used to draw edges.
edge.width	the width used to draw edges.
font	an integer specifying the type of font for the labels: 1 (plain text), 2 (bold), 3 (italic, the default), or 4 (bold italic).
cex	a numeric value giving the factor scaling of the labels.
...	Further arguments passed to or from other methods.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Dress, A.W.M. and Huson, D.H. (2004) Constructing Splits Graphs *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, **1(3)**, 109–115

See Also

[consensusNet](#), [hadamard](#), [distanceHadamard](#), [layout.kamada.kawai](#)

Examples

```
## Not run:  
see example in consensusNet  
example(consensusNet)  
  
## End(Not run)
```

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, optRooted=FALSE,
          control = pml.control(epsilon=1e-08, maxit=10, trace=1), model = NULL, subs = NULL, ...)
pml.control(epsilon = 1e-08, maxit = 10, trace = 1)
```

Arguments

tree	A phylogenetic tree, object of class phylo.
data	The (DNA) alignment.
bf	Base frequencies.
Q	A vector containing the lower triangular part of the rate matrix.
inv	Proportion of invariable sites.
k	Number of intervals of the discrete gamma distribution.
shape	Shape parameter of the gamma distribution.
rate	Rate.
model	allows to choose an amino acid models or nucleotide model, see details.
object	An object of class pml.
optNni	Logical value indicating whether topology gets optimized (NNI).
optBf	Logical value indicating whether base frequencies gets optimized.
optQ	Logical value indicating whether rate matrix gets optimized.
optInv	Logical value indicating whether proportion of variable size gets optimized.
optGamma	Logical value indicating whether gamma rate parameter gets optimized.
optEdge	Logical value indicating the edge lengths gets optimized.
optRate	Logical value indicating the overall rate gets optimized.
optRooted	Logical value indicating if the edge lengths of a rooted tree get optimized.
control	A list of parameters for controlling the fitting process.
subs	A (integer) vector same length as Q to specify the optimization of Q
...	Further arguments passed to or from other methods.
epsilon	Stop criterion for optimisation (see details).
maxit	Maximum number of iterations (see details).
trace	Show output during optimization (see details).

Details

The topology search uses a nearest neighbor interchange (NNI) and the implementation is similar to phyML. The option `model` in `pml` is only used for amino acid models. The option `model` defines the nucleotide model which is getting optimised, all models which are included in `modeltest` can be chosen. Setting this option (e.g. "K81" or "GTR") overrules options `optBf` and `optQ`. Here is an overview how to estimate different phylogenetic models with `pml`:

model	optBf	optQ
Jukes-Cantor	FALSE	FALSE
F81	TRUE	FALSE
symmetric	FALSE	TRUE
GTR	TRUE	TRUE

Via `model` in `optim.pml` the following nucleotide models can be specified: JC, F81, K80, HKY, TrNe, TrN, TPM1, K81, TPM1u, TPM2, TPM2u, TPM3, TPM3u, TIM1e, TIM1, TIM2e, TIM2, TIM3e, TIM3, TVMe, TVM, SYM and GTR. These models are specified as in Posada (2008).

So far 9 amino acid models are supported ("WAG", "JTT", "Dayhoff", "LG", "cpREV", "mtmam", "mtArt", "MtZoa" and "mtREV24") and additionally rate matrices and amino acid frequencies can be supplied to .

If the option `'getRooted'` is set to TRUE than the edge lengths of rooted tree are optimized. The tree has to be rooted and ultrametric! No tree rearrangements are yet supported. If `'getRooted=FALSE'` any rooted tree is getting unrooted.

`pml.control` controls the fitting process. `epsilon` and `maxit` are only defined for the most outer loop, this affects `pmlCluster`, `pmlPart` and `pmlMix`. `epsilon` is defined as $(\log\text{Lik}(k) - \log\text{Lik}(k+1)) / \log\text{Lik}(k+1)$, this seems to be a good heuristics which works reasonably for small and large trees or alignments. If `trace` is set to zero than no out put is shown, if functions are called internally than the trace is decreased by one.

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.
<code>weight</code>	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.
- Adachi, J., P. J. Waddell, W. Martin, and M. Hasegawa (2000) Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA. *Journal of Molecular Evolution*, **50**, 348–358
- Rota-Stabelli, O., Z. Yang, and M. Telford. (2009) MtZoa: a general mitochondrial amino acid substitutions model for animal evolutionary studies. *Mol. Phyl. Evol*, **52(1)**, 268–72
- 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
- Yang, Z., R. Nielsen, and M. Hasegawa (1998) Models of amino acid substitution and applications to Mitochondrial protein evolution. *Molecular Biology and Evolution*, **15**, 1600–1611
- Abascal, F., D. Posada, and R. Zardoya (2007) MtArt: A new Model of amino acid replacement for Arthropoda. *Molecular Biology and Evolution*, **24**, 1–5
- Kosiol, C, and Goldman, N (2005) Different versions of the Dayhoff rate matrix - *Molecular Biology and Evolution*, **22**, 193–199

See Also

[bootstrap.pml](#), [pmlPart](#), [pmlMix](#), [plot.phylo](#)

Examples

```
example(NJ)
# Jukes-Cantor (starting tree from NJ)
fitJC <- pml(tree, Laurasiatherian)
# optimize 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)
# optimize 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 <- acgt2ry(Laurasiatherian)
```

```

fit2ST <- pml(tree, dat)
fit2ST <- optim.pml(fit2ST,optNni=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=1:5, part=NULL, nrep = 10, control=pml.control(epsilon=1e-8, maxit=
```

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 partition is generated.
nrep	number of replicates for each p.
control	A list of parameters for controlling the fitting process.
...	Further arguments passed to or from other methods.

Details

The formula object allows to specify which parameter get optimized. The formula is generally of the form $edge + bf + Q \sim rate + shape + \dots$, on the left side are the parameters which get optimized over all cluster, on the right the parameter which are optimized 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 specified 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 optimization.
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=1:4)
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), control=pml.control(epsilon=1e-08, maxit=20, trace=1),...
```

Arguments

formula	a formula object (see details).
fit	an object of class pml.
m	number of mixtures.
omega	mixing weights.
control	A list of parameters for controlling the fitting process.
...	Further arguments passed to or from other methods.

Details

The formula object allows to specify which parameter get optimized. 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 optimized over all mixtures, on the right the parameter which are optimized 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 left-hand 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, control = pml.control(epsilon=1e-8, maxit=10, trace=1),...)
```

Arguments

formula	a formula object (see details).
object	an object of class pml or a list of objects of class pml .

control A list of parameters for controlling the fitting process.
 ... Further arguments passed to or from other methods.

Details

The formula object allows to specify which parameter get optimized. 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 optimized over all partitions, on the right the parameter which are optimized 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 optimization.
 object an object of class "pml" or "pmlPart"

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

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

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]

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	<i>Read Amino Acid Sequences in a File</i>
---------	--

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 row names.

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	<i>Shimodaira-Hasegawa Test</i>
---------	---------------------------------

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) # optimize 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",nbcol = -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 minimizes the penalized 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 constrained 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)
net = as.network(fit)
plot(net)
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 and WPGMA

Description

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

Usage

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

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](#), [fastme](#), [nnls.tree](#)

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, 39

*Topic **\textasciitildekwd1**

ancestral.pml, 5

*Topic **\textasciitildekwd2**

ancestral.pml, 5

bab, 7

*Topic **classif**

dfactorial, 13

treedist, 43

*Topic **cluster**

allTrees, 3

as.splits, 6

bab, 7

bootstrap.pml, 8

designTree, 12

dist.hamming, 14

distanceHadamard, 15

getClans, 16

hadamard, 18

lento, 20

midpoint, 21

modelTest, 22

NJ, 24

nni, 25

parsimony, 26

phyDat, 27

pml, 31

pmlCluster, 34

pmlMix, 35

pmlPart, 37

simSeq, 41

splitsNetwork, 42

upgma, 44

*Topic **datasets**

chloroplast, 10

Laurasiatherian, 20

yeast, 45

*Topic **hplot**

consensusNet, 10

*Topic **misc**

Ancestors, 4

*Topic **models**

SH.test, 40

*Topic **package**

phangorn-package, 2

*Topic **plot**

lento, 20

plot.networx, 29

acctran (parsimony), 26

acgt2ry (phyDat), 27

allSitePattern (phyDat), 27

allSplits (as.splits), 6

allTrees, 3

Ancestors, 4

ancestral.pars, 27

ancestral.pars (ancestral.pml), 5

ancestral.pml, 5, 27

anova, 23

as.character.phyDat (phyDat), 27

as.data.frame.phyDat (phyDat), 27

as.DNAbin, 29

as.DNAbin.phyDat (phyDat), 27

as.matrix.splits (as.splits), 6

as.networx (consensusNet), 10

as.phyDat (phyDat), 27

as.phylo, 45

as.prop.part (as.splits), 6

as.splits, 6, 21

bab, 7, 27

bootstrap.phyDat (bootstrap.pml), 8

bootstrap.pml, 8, 26, 27, 33

BranchAndBound (bab), 7

Children (Ancestors), 4

chloroplast, 10

CI, 17

- CI (parsimony), 26
- compatible (as.splits), 6
- consensus, 22
- consensusNet, 10, 30
- Descendants (Ancestors), 4
- designSplits (designTree), 12
- designTree, 12, 43
- dfactorial, 8, 13
- di2multi, 22
- dist.dna, 14, 24
- dist.hamming, 14, 24
- dist.logDet (dist.hamming), 14
- dist.ml (dist.hamming), 14
- distanceHadamard, 7, 11, 12, 15, 18, 19, 30, 43
- diversity (getClans), 16
- DNABin, 29
- factorial, 13
- fastme, 12, 24, 45
- fhm (hadamard), 18
- fitch (parsimony), 26
- getClans, 16, 27
- getClips (getClans), 16
- getDiversity (getClans), 16
- getRoot (midpoint), 21
- getSlices (getClans), 16
- h2st (hadamard), 18
- h4st (hadamard), 18
- hadamard, 15, 18, 21, 30
- hclust, 44, 45
- Laurasiatherian, 20
- layout.kamada.kawai, 30
- ldfactorial (dfactorial), 13
- lento, 7, 11, 15, 19, 20
- midpoint, 21
- modelTest, 22
- NJ, 24, 27, 45
- nj, 24
- nni, 25, 27
- npls.tree, 45
- npls.tree (designTree), 12
- optim.parsimony (parsimony), 26
- optim.pml, 9
- optim.pml (pml), 31
- pace (ancestral.pml), 5
- parsimony, 17, 26
- phangorn (phangorn-package), 2
- phangorn-package, 2
- phyDat, 17, 27, 39, 42
- plot.networx, 15, 19, 29
- plot.phylo, 9, 33
- plotAnc (ancestral.pml), 5
- plotBS (bootstrap.pml), 8
- pml, 9, 23, 27, 31, 35, 36, 38, 40
- pmlCluster, 34, 36, 38, 40
- pmlMix, 29, 33, 35, 35, 38
- pmlPart, 33, 35, 36, 37, 40
- pmlPen (pmlMix), 35
- PNJ (parsimony), 26
- pratchet, 8
- pratchet (parsimony), 26
- print.splits (as.splits), 6
- prop.part, 7
- pruneTree (midpoint), 21
- random.addition (parsimony), 26
- read.aa, 29, 39
- read.alignment, 39
- read.dna, 29, 39
- read.GenBank, 39
- read.nexus.data, 29
- read.phyDat (phyDat), 27
- RF.dist (treedist), 43
- RI, 17
- RI (parsimony), 26
- rNNI (nni), 25
- root, 22
- rSPR (nni), 25
- sankoff (parsimony), 26
- SH.test, 35, 38, 40
- Siblings (Ancestors), 4
- simSeq, 41
- splitsNetwork, 11, 42
- subset.phyDat (phyDat), 27
- treedist, 43
- UNJ (NJ), 24
- upgma, 12, 24, 44

wpgma (upgma), [44](#)
write.nexus.splits (hadamard), [18](#)
write.phyDat (phyDat), [27](#)
write.splits (as.splits), [6](#)

yeast, [45](#)