Package: phangorn (via r-universe)

October 29, 2026

Title Phylogenetic Reconstruction and Analysis

Version 3.0.0.1

Description Allows for estimation of phylogenetic trees and networks using Maximum Likelihood, Maximum Parsimony, distance methods and Hadamard conjugation (Schliep 2011). Offers methods for tree comparison, model selection and visualization of phylogenetic networks as described in Schliep et al. (2017).

License GPL (>= 2)

URL https://github.com/KlausVigo/phangorn, https://klausvigo.github.io/phangorn/

BugReports https://github.com/KlausVigo/phangorn/issues

Depends ape (>= 5.8), R (>= 4.1.0)

Imports checkmate, digest, future.apply, generics, graphics, grDevices, igraph (>= 1.0), Matrix, methods, osqp, parallel, progressr, Rcpp, stats, utils

Suggests apex, bibtex, Biostrings, codetools, ggseqlogo, ggplot2, knitr, magick, plot3D, rgl, rmarkdown, rsvg, seqinr, tinysnapshot (>= 0.2.0), tinytest, xtable

LinkingTo Rcpp, RcppArmadillo

VignetteBuilder knitr, utils

biocViews Software, Technology, QualityControl

Encoding UTF-8

Roxygen list(old_usage = TRUE, markdown = TRUE, roclets = c (``namespace", ``rd", ``srr::srr_stats_roclet"))

Language en-US

Config/roxygen2/version 8.1.1

Config/pak/sysreqs libglpk-dev make libxml2-dev

Repository https://klausvigo.r-universe.dev

Date/Publication 2026-10-29 16:23:09 UTC

RemoteUrl https://github.com/klausvigo/phangorn

RemoteRef HEAD

RemoteSha fc77eeb77e1cac08d33b7ea1835fa9089a8f81ba

Contents

acctran . . . 3
add.tips . . . 4
add_ci . . . 5
add_edge_length . . . 7
addConfidences . . . 8
allSplits . . . 9
allTrees . . . 11
ancestral.pml . . . 12
as.networx . . . 14
as.pml . . . 15
bab . . . 20
baseFreq . . . 21
bootstrap.pml . . . 22
chloroplast . . . 24
CI . . . 25
cladePar . . . 26
coalSpeciesTree . . . 27
codonTest . . . 28
consensusNet . . . 29
cophenetic.networx . . . 31
delta.score . . . 31
densiTree . . . 32
designTree . . . 35
discrete.gamma . . . 36
dist.hamming . . . 38
dist.p . . . 40
distanceHadamard . . . 41
dna2codon . . . 42
fitch . . . 44
gap_as_state . . . 46
getClans . . . 47
getRoot . . . 50
hadamard . . . 51
heatmapp . . . 53
identify.networx . . . 54
image.phyDat . . . 55
latag2n.phyDat . . . 56
Laurasiatherian . . . 57
ldfactorial . . . 57
lento . . . 58
lli . . . 59

mast . . . 61
maxCladeCred . . . 62
mites . . . 63
modelTest . . . 64
multiphyDat2pmlPart . . . 66
neighborNet . . . 68
NJ . . . 69
nni . . . 70
phyDat . . . 71
plot.networx . . . 74
plot.pml . . . 76
plotAnc . . . 77
plotBS . . . 79
pml.control . . . 81
pml_bb . . . 83
pmlCluster . . . 84
pmlMix . . . 86
print.phyDat . . . 88
read.nexus.partitions . . . 91
read.nexus.splits . . . 92
read.phyDat . . . 93
SH.test . . . 95
simSeq . . . 96
SOWH.test . . . 98
splitsNetwork . . . 100
superTree . . . 101
terraces . . . 102
transferBootstrap . . . 104
treedist . . . 105
upgma . . . 107
write.ancestral . . . 109
write.pml . . . 110
writeDist . . . 111
yeast . . . 113

Index **114**

`acctran` *Assign edge length to tree*

Description

`parsimony_edgelength` and `acctran` assign edge length to a tree where the edge length is the number of mutations. `parsimony_edgelengths` assigns edge lengths using a joint reconstruction based on the sankoff algorithm. Ties are broken at random and trees can be multifurating. `acctran` is based on the fitch algorithm and is faster. However trees need to be bifurcating and ties are split.

Usage

```
acctran(tree, data)

parsimony_edgelength(tree, data)
```

Arguments

tree	a tree, i.e. an object of class pml
data	an object of class phyDat

Value

a tree with edge length.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

pratchet

Examples

```
data(Laurasiatherian)
# lower number of iterations for the example, to run time less than 5 sec.
treeRatchet <- pratchet(Laurasiatherian, minit=5, k=5, trace=0)
# assign edge length (number of substitutions)
treeRatchet <- parsimony_edgelength(treeRatchet, Laurasiatherian)
plot(midpoint(treeRatchet))
add.scale.bar(0,0, length=100)
```

---

add.tips *Add tips to a tree*

Description

This function binds tips to nodes of a phylogenetic trees.

Usage

```
add.tips(tree, tips, where, edge.length = NULL)
```

Arguments

tree	an object of class "phylo".
tips	a character vector containing the names of the tips.
where	an integer or character vector of the same length as tips giving the number of the node or tip of the tree where to add the new tips.
edge.length	optional numeric vector with edge length

Value

an object of class phylo

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

bind.tree

Examples

```
tree <- rcoal(10)
plot(tree)
nodelabels()
tiplabels()
tree1 <- add.tips(tree, c("A", "B", "C"), c(1,2,15))
plot(tree1)
```

---

add_ci *Draw Confidences Intervals on Phylogenies*

Description

These are low-level plotting commands to draw the confidence intervals on the node of a tree as rectangles with coloured backgrounds or add boxplots to ultrametric or tipdated trees.

Usage

```
add_ci(tree, trees = NULL, col95 = "#FF00004D", col50 = "#0000FF4D",
  height = 0.7, legend = TRUE, ...)

add_boxplot(tree, trees = NULL, boxwex = 0.7, ...)
```

Arguments

tree	either an object of class phylo to which the confidences should be added or an object of class pml. In case of the later the tree is extracted from the object.
trees	phylogenetic trees, i.e. an object of class multiPhylo. Can be empty if tree is an object of class pml.
col95	colour used for the 95% intervals; by default: transparent red.
col50	colour used for the 50% intervals; by default: transparent blue.
height	the height of the boxes.
legend	a logical value.
...	arguments passed to other functions, legend or bxp.
boxwex	a scale factor to be applied to all boxes, see bxp.

Details

All trees should to be rooted, either ultrametric or tip dated.

Value

add_ci and add_boxplot return silently the tree object.

Author(s)

Emmanuel Paradis, Santiago Claramunt, Joseph Brown, Klaus Schliep

See Also

plot.phylo, plotBS, add_edge_length, maxCladeCred

Examples

```
data("Laurasiatherian")
dm <- dist.hamming(Laurasiatherian)
tree <- upgma(dm)
set.seed(123)
trees <- bootstrap.phyDat(Laurasiatherian,
                          FUN=function(x)upgma(dist.hamming(x)), bs=100)
tree <- plotBS(tree, trees, "phylogram")
add_ci(tree, trees, bty="n")
plot(tree, direction="downwards")
add_boxplot(tree, trees, boxwex=0.7)
```

add_edge_length *Assign and compute edge lengths from a sample of trees*

Description

This command can infer some average edge lengths and assign them from a (bootstrap/MCMC) sample.

Usage

```
add_edge_length(tree, trees, fun = function(x) median(na.omit(x)),
  rooted = all(is.rooted(trees)))
```

Arguments

tree	a phylogenetic tree or splitnetwork where edge lengths are assigned to.
trees	an object of class multiPhylo, where the average for the edges is computed from.
fun	a function to compute the average (default is median).
rooted	rooted logical, if FALSE edge lengths is a function of the observed splits, if TRUE edge lengths are estimated from height for the observed clades.

Value

The tree with newly assigned edge length.

Author(s)

Klaus Schliep

See Also

node.depth, consensus, maxCladeCred, add_boxplot

Examples

```
data("Laurasiatherian")
set.seed(123)
bs <- bootstrap.phyDat(Laurasiatherian,
                FUN=function(x)upgma(dist.ml(x)), bs=100)
tree_compat <- allCompat(bs, rooted=TRUE) |>
              add_edge_length(bs)
plot(tree_compat)
add_boxplot(tree_compat, bs, boxwex=0.7)
```

addConfidences *Compare splits and add support values to an object*

Description

Add support values to a splits, phylo or networx object (Schliep, Potts, Morrison, and Grimm 2017).

Usage

```
addConfidences(x, y, ...)

## S3 method for class 'phylo'
addConfidences(x, y, rooted = FALSE, ...)

presenceAbsence(x, y)

createLabel(x, y, label_y, type = "edge", nomatch = NA)
```

Arguments

x	an object of class splits, phylo or networx
y	an object of class splits, phylo, multiPhylo or networx
...	Further arguments passed to or from other methods.
rooted	logial, if FALSE bipartitions are considered, if TRUE clades.
label_y	label of y matched on x. Will be usually of length(as.splits(x)).
type	should labels returned for edges (in networx) or splits.
nomatch	default value if no match between x and y is found.

Value

The object x with added bootstrap / MCMC support values.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Schliep K, Potts AJ, Morrison DA, Grimm GW (2017). "Intertwining phylogenetic trees and networks." *Methods in Ecology and Evolution*, **8**(10), 1212-1220. *Methods Ecol Evol*.**8**, 1212–1220. doi:10.1111/2041-210X.12760

See Also

as.splits, as.networx, RF.dist, plot.phylo

Examples

```
data(woodmouse)
woodmouse <- phyDat(woodmouse)
tmpfile <- normalizePath(system.file(
             "extdata/trees/RAxML_bootstrap.woodmouse", package="phangorn"))
boot_trees <- read.tree(tmpfile)

dm <- dist.ml(woodmouse)
tree <- upgma(dm)
nnet <- neighborNet(dm)

tree <- addConfidences(tree, boot_trees)
nnet <- addConfidences(nnet, boot_trees)

plot(tree, show.node.label=TRUE)
plot(nnet, show.edge.label=TRUE)
```

---

allSplits *Splits representation of graphs and trees.*

Description

as.splits produces a list of splits or bipartitions.

Usage

```
allSplits(k, labels = NULL)

allCircularSplits(k, labels = NULL)

allCircularSplitsST(k, labels = NULL, pos = NULL)

as.splits(x, ...)

## S3 method for class 'splits'
as.matrix(x, zero.print = 0L, one.print = 1L, ...)

## S3 method for class 'splits'
as.Matrix(x, ...)

## S3 method for class 'splits'
print(x, maxp = getOption("max.print"), zero.print = ".",
  one.print = "|", ...)

## S3 method for class 'splits'
c(..., recursive = FALSE)
```

```
## S3 method for class 'splits'
unique(x, incomparables = FALSE, unrooted = TRUE, ...)

## S3 method for class 'phylo'
as.splits(x, ...)

## S3 method for class 'multiPhylo'
as.splits(x, ...)

## S3 method for class 'networx'
as.splits(x, ...)

## S3 method for class 'splits'
as.prop.part(x, ...)

## S3 method for class 'splits'
as.bitsplits(x)

## S3 method for class 'bitsplits'
as.splits(x, ...)

compatible(obj1, obj2 = NULL)
```

Arguments

k	number of taxa.
labels	names of taxa.
pos	logical vector indicating which splits to return.
x	An object of class phylo or multiPhylo.
...	Further arguments passed to or from other methods.
zero.print	character which should be printed for zeros.
one.print	character which should be printed for ones.
maxp	integer, default from options(max.print), influences how many entries of large matrices are printed at all.
recursive	logical. If recursive = TRUE, the function recursively descends through lists (and pairlists) combining all their elements into a vector.
incomparables	only for compatibility so far.
unrooted	todo.
obj1, obj2	an object of class splits.

Value

as.splits returns an object of class splits, which is mainly a list of splits and some attributes. Often a splits object will contain attributes confidences for bootstrap or Bayesian support values and weight storing edge weights. 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, as.networx, distanceHadamard, read.nexus.splits

Examples

```
(sp <- as.splits(rtree(5)))
write.nexus.splits(sp)
spl <- allCircularSplits(5)
plot(as.networx(spl))
```

---

allTrees *Compute all trees topologies.*

Description

allTrees computes all bifurcating tree topologies for rooted or unrooted trees with up to 10 tips. The number of trees grows fast.

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>

See Also

rtree, nni, howmanytrees, dfactorial

Examples

```
trees <- allTrees(5)

old.par <- par(no.readonly = TRUE)
par(mfrow = c(3,5))
for(i in 1:15)plot(trees[[i]])
par(old.par)
```

`ancestral.pml` *Ancestral character reconstruction.*

Description

Marginal reconstruction of the ancestral character states.

Usage

```
ancestral.pml(object, type = "marginal", return = "ancestral", ...)

anc_pml(object, type = "marginal", ...)

ancestral.pars(tree, data, type = c("MPR", "ACCTRAN", "POSTORDER"),
  cost = NULL, return = "prob", ...)

anc_pars(tree, data, type = c("MPR", "ACCTRAN", "POSTORDER"), cost = NULL,
  ...)

pace(tree, data, type = c("MPR", "ACCTRAN", "POSTORDER"), cost = NULL,
  return = "prob", ...)
```

Arguments

`object`	an object of class pml
`type`	method used to assign characters to internal nodes, see details.
`return`	return a `phyDat` object or matrix of probabilities.
`...`	Further arguments passed to or from other methods.
`tree`	a tree, i.e. an object of class pml
`data`	an object of class phyDat
`cost`	A cost matrix for the transitions between two states.

Details

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

The function return a list containing the tree with node labels, the original alignment as an `phyDat` object, a data.frame containing the probabilities belonging to a state for all (internal nodes) and the most likely state. For parsimony and nucleotide data the most likely state might be ambiguous. For ML this is very unlikely to be the case.

If the input tree does not contain unique node labels the function `ape::MakeNodeLabel` is used to create them.

With parsimony reconstruction one has to keep in mind that there will be often no unique solution.

The functions use the node labels of the provided tree (also if part of the `pml` object) if these are unique. Otherwise the function `ape::MakeNodeLabel` is used to create them.

For further details see vignette("Ancestral").

Value

An object of class ancestral. This is a list containing the tree with node labels, the original alignment as an `phyDat` object, a `data.frame` containing the probabilities belonging to a state for all (internal nodes) and the most likely state, again a `phyDat` object.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

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

Swofford D, Maddison W (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`, `plotAnc`, `heatmapp`, `latag2n.phyDat`, `latag2n`, `gap_as_state`, `root`, `makeNodeLabel`

Examples

```
example(NJ)
# generate node labels to ensure plotting will work
tree <- makeNodeLabel(tree)
fit <- pml(tree, Laurasiatherian)
anc.ml <- anc_pml(fit)
anc.p <- anc_pars(tree, Laurasiatherian)
# plot ancestral sequences at the root
plotSeqLogo( anc.ml, 48, 1, 20)
```

```
plotSeqLogo( anc.p, 48, 1, 20)
# plot the first character
plotAnc(anc.ml)
# plot the third character
plotAnc(anc.ml, 3)
# plot joint reconstruction as heatmap
heatmapp(anc.ml, select=1:50)
```

`as.networx` *Conversion among phylogenetic network objects*

Description

`as.networx` convert `splits` objects into a `networx` object. And most important there exists a generic `plot` function to plot phylogenetic network or split graphs.

Usage

```
as.networx(x, ...)

## S3 method for class 'splits'
as.networx(x, planar = FALSE, coord = "none", ...)

## S3 method for class 'phylo'
as.networx(x, ...)
```

Arguments

`x`	an object of class `"splits"` or `"phylo"`
`...`	Further arguments passed to or from other methods.
`planar`	logical whether to produce a planar graph from only cyclic splits (may excludes splits).
`coord`	add coordinates of the nodes, allows to reproduce the plot.

Details

A `networx` object hold the information for a phylogenetic network and extends the `phylo` object. Therefore some generic function for `phylo` objects will also work for `networx` objects. The argument `planar = TRUE` will create a planar split graph based on a cyclic ordering. These objects can be nicely plotted in `"2D"`. The argument "coord" allows to create coordinates, with options are "none", "equal angle", "3D", "2D" and "outline" (Bagci et al. 2021).

Value

an object of class `networx`.

Note

The internal representation is likely to change.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Schliep, K., Potts, A. J., Morrison, D. A. and Grimm, G. W. (2017), Intertwining phylogenetic trees and networks. *Methods Ecol Evol*. **8**, 1212–1220. doi:10.1111/2041-210X.12760

Bagci, C., Bryant, D., Cetinkaya, B. and Huson, D.H. (2021), Microbial Phylogenetic Context Using Phylogenetic Outlines. *Genome Biology and Evolution*. Volume 13. Issue 9. evab213

See Also

consensusNet, neighborNet, splitsNetwork, hadamard, distanceHadamard, plot.networx, evonet, as.phylo

Examples

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

spl <- allCircularSplits(5)
plot(as.networx(spl))
plot(as.networx(spl, coord="outline"))
## Not run:
# also see example in consensusNet
example(consensusNet)

## End(Not run)
```

---

as.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. For a more user-friendly interface see pml_bb.

Usage

```
as.pml(x, ...)

## S3 method for class 'pml'
update(object, ...)

pml(tree, data, bf = NULL, Q = NULL, inv = 0, k = 1, shape = 1,
  rate = 1, model = NULL, site.rate = "gamma", ASC = FALSE, ...)

optim.pml(object, optNni = FALSE, optBf = FALSE, optQ = FALSE,
  optInv = FALSE, optGamma = FALSE, optEdge = TRUE, optRate = FALSE,
  optRooted = FALSE, control = pml.control(), model = NULL,
  rearrangement = ifelse(optNni, "NNI", "none"), subs = NULL,
  ratchet.par = ratchet.control(), ...)

## S3 method for class 'pml'
logLik(object, ...)

## S3 method for class 'pml'
anova(object, ...)

## S3 method for class 'pml'
vcov(object, ...)

## S3 method for class 'pml'
print(x, ...)

## S3 method for class 'pml'
glance(x, ...)
```

Arguments

x	So far only an object of class modelTest.
...	Further arguments passed to or from other methods.
object	An object of class pml.
tree	A phylogenetic tree, object of class phylo.
data	An alignment, object of class phyDat.
bf	Base frequencies (see details).
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.

site.rate	Indicates what type of gamma distribution to use. Options are "gamma" approach of Yang 1994 (default), "gamma_quadrature" after the Laguerre quadrature approach of Felsenstein 2001 or "free_rate".
ASC	ascertainment bias correction (ASC), allows to estimate models like Lewis' Mkv.
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.
rearrangement	type of tree tree rearrangements to perform, one of "none", "NNI", "stochastic" or "ratchet"
subs	A (integer) vector same length as Q to specify the optimization of Q
ratchet.par	search parameter for stochastic search

Details

Base frequencies in pml can be supplied in different ways. For amino acid they are usually defined through specifying a model, so the argument bf does not need to be specified. Otherwise if bf=NULL, each state is given equal probability. It can be a numeric vector given the frequencies. Last but not least bf can be string "equal", "empirical" and for codon models additionally "F3x4".

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 optimized, 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 a 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 17 amino acid models are supported ("WAG", "JTT", "LG", "Dayhoff", "cpREV", "mtmam", "mtArt", "MtZoa", "mtREV24", "VT","RtREV", "HIVw", "HIVb", "FLU", "Blosum62", "Dayhoff_DCMut" and "JTT_DCMut") and additionally rate matrices and amino acid frequencies can be supplied.

It is also possible to estimate codon models (e.g. YN98), for details see also the chapter in vignette("phangorn-specials").

If the option 'optRooted' is set to TRUE than the edge lengths of rooted tree are optimized. The tree has to be rooted and by now ultrametric! Optimising rooted trees is generally much slower.

If `rearrangement` is set to `stochastic` a stochastic search algorithm similar to Nguyen et al. (2015). and for `ratchet` the likelihood ratchet as in Vos (2003). This should helps often to find better tree topologies, especially for larger trees.

Value

`pml` or `optim.pml` return a list of class `pml`, some are useful for further computations like

`tree`	the phylogenetic tree.
`data`	the alignment.
`logLik`	Log-likelihood of the tree.
`siteLik`	Site log-likelihoods.
`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. (2001) Taking Variation of Evolutionary Rates Between Sites into Account in Inferring Phylogenies. *Journal of Molecular Evolution*, **53**, 447–455.

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

L.-T. Nguyen, H.A. Schmidt, A. von Haeseler, and B.Q. Minh (2015) IQ-TREE: A fast and effective stochastic algorithm for estimating maximum likelihood phylogenies. *Molecular Biology and Evolution*, **32**, 268–274.

Vos, R. A. (2003) Accelerated Likelihood Surface Exploration: The Likelihood Ratchet. *Systematic Biology*, **52(3)**, 368–373

Yang, Z., and R. Nielsen (1998) Synonymous and nonsynonymous rate variation in nuclear genes of mammals. *Journal of Molecular Evolution*, **46**, 409-418.

Lewis, P.O. (2001) A likelihood approach to estimating phylogeny from discrete morphological character data. *Systematic Biology* **50**, 913–925.

See Also

pml_bb, pml.control, bootstrap.pml, modelTest, pmlPart, pmlMix, plot.phylo, SH.test, ancestral.pml

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=0.2)
# 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, rearrangement = "stochastic",
      optGamma=TRUE, optInv=TRUE, model="GTR")

## 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")
```

bab *Branch and bound for finding all most parsimonious trees*

Description

bab finds all most parsimonious trees (Hendy and Penny 1982).

Usage

```
bab(data, tree = NULL, trace = !getOption("quiet"), ...)
```

Arguments

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

Details

This implementation can be very slow and depending on the data may take very long time. In the worst case all $(2n-5)!! = 1 \times 3 \times 5 \times \ldots \times (2n-5)$ possible trees have to be examined, where n is the number of species / tips. For ten 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. bab uses a very basic heuristic approach of MinMax Squeeze Holland, Huber, Penny, and Moulton (2005) to improve the lower bound and optimtsations from White and Holland (2011).
bab might return multifurcating trees. These multifurcations could be resolved in all ways.
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, Penny D (1982). "Branch and bound algorithms to determine minimal evolutionary trees." *Math. Biosc.*, **59**, 277–290.

Holland BR, Huber KT, Penny D, Moulton V (2005). "The MinMax Squeeze: Guaranteeing a Minimal Tree for Population Data." *Molecular Biology and Evolution*, **22**(2), 235-242. doi:10.1093/molbev/msi010.

White WTJ, Holland BR (2011). "Faster exact maximum parsimony search with XMP." *Bioinformatics*, **27**(10), 1359-1367. ISSN 1367-4803. doi:10.1093/bioinformatics/btr147.

See Also

pratchet, dfactorial

Examples

```
data(yeast)
dfactorial(11)
# choose only the first two genes
gene12 <- yeast[, 1:3158]
trees <- bab(gene12)
```

---

baseFreq *Summaries of alignments*

Description

baseFreq computes the frequencies (absolute or relative) of the states from a sample of sequences. glance computes some useful information about the alignment. composition_test computes a χ^2-test testing if the state composition for a species differs.

Usage

```
baseFreq(obj, freq = FALSE, all = FALSE, drop.unused.levels = FALSE)

## S3 method for class 'phyDat'
glance(x, ...)

composition_test(obj)

## S3 method for class 'phyDat'
summary(object, ...)

## S3 method for class 'summary.phyDat'
print(x, ..., digits = max(3L, getOption("digits") -
  3L))
```

Arguments

freq	logical, if 'TRUE', frequencies or counts are returned otherwise proportions
all	all a logical; if all = TRUE, all counts of bases, ambiguous codes, missing data, and alignment gaps are returned as defined in the contrast.
drop.unused.levels	logical, drop unused levels
...	further arguments passed to or from other methods.
object, obj, x	as object of class phyDat
digits	minimal number of significant digits.

Value

baseFreq returns a named vector and glance a one row data.frame.

Author(s)

Klaus Schliep

See Also

phyDat, base.freq, glance

Examples

```
data(Laurasiatherian)
data(chloroplast)
# base frequencies
baseFreq(Laurasiatherian)
baseFreq(Laurasiatherian, all=TRUE)
baseFreq(Laurasiatherian, freq=TRUE)
baseFreq(chloroplast)

# some statistics
glance(Laurasiatherian)
glance(chloroplast)
composition_test(Laurasiatherian)[1:10,]
summary(Laurasiatherian)
```

---

bootstrap.pml *Bootstrap*

Description

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

Usage

```
bootstrap.pml(x, bs = 100, trees = TRUE, multicore = FALSE,
  mc.cores = NULL, tip.dates = NULL, ...)

bootstrap.phyDat(x, FUN, bs = 100, multicore = FALSE, mc.cores = NULL,
  jumble = TRUE, ...)
```

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, whether models should estimated in parallel.
mc.cores	The number of cores to use during bootstrap. Only supported on UNIX-alike systems.
tip.dates	A named vector of sampling times associated to the tips/sequences. Leave empty if not estimating tip dated phylogenies.
...	further parameters used by optim.pml or plot.phylo.
FUN	the function to estimate the trees.
jumble	logical, jumble the order of the sequences.

Details

It is possible that the bootstrap is performed in parallel using the future package, see example below.

Value

bootstrap.pml returns an object of class multi.phylo or a list where each element is an object of class pml. plotBS returns silently a tree, i.e. an object of class phylo with the bootstrap values as node labels. The argument BStrees is optional and if not supplied the tree with labels supplied in the node.label slot.

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.

Lemoine F, Entfellner JD, Wilkinson E, Correia D, Felipe MD, De Oliveira T, Gascuel O (2018). "Renewing Felsenstein's phylogenetic bootstrap in the era of big data." *Nature*, **556**(7702), 452–456.

Penny D, Hendy M (1985). "Testing methods evolutionary tree construction." *Cladistics*, **1**, 266–278.

Penny D, Hendy M (1986). "Estimating the reliability of evolutionary trees." *Molecular Biology and Evolution*, **3**, 403–417.

See Also

optim.pml, pml, plan, plot.phylo, maxCladeCred nodelabels,consensusNet and SOWH.test for parametric bootstrap

Examples

```
## Not run:
data(Laurasiatherian)
dm <- dist.hamming(Laurasiatherian)
tree <- NJ(dm)
# NJ
set.seed(123)
NJtrees <- bootstrap.phyDat(Laurasiatherian,
     FUN=function(x)NJ(dist.hamming(x)), bs=100)
treeNJ <- plotBS(tree, NJtrees, "phylogram")

# Maximum likelihood
fit <- pml(tree, Laurasiatherian)
fit <- optim.pml(fit, rearrangement="NNI")
set.seed(123)
# compute in parallel
future::plan(multisession, workers = 2)
bs <- bootstrap.pml(fit, bs=100, optNni=TRUE)
future::plan(sequential)

treeBS <- plotBS(fit$tree,bs)

# Maximum parsimony
treeMP <- pratchet(Laurasiatherian)
treeMP <- acctran(treeMP, Laurasiatherian)
set.seed(123)
BStrees <- bootstrap.phyDat(Laurasiatherian, pratchet, bs = 100)
treeMP <- plotBS(treeMP, BStrees, "phylogram")
add.scale.bar()

# export tree with bootstrap values as node labels
# write.tree(treeBS)

## End(Not run)
```

---

chloroplast *Chloroplast alignment*

Description

Amino acid alignment of 15 genes of 19 different chloroplast.

Examples

```
data(chloroplast)
chloroplast
```

CI *Consistency Index and Retention Index*

Description

CI and RI compute the Consistency Index (CI) and Retention Index (RI).

Usage

```
CI(tree, data, cost = NULL, sitewise = FALSE)

RI(tree, data, cost = NULL, sitewise = FALSE)
```

Arguments

tree	a phylogenetic tree, i.e. an object of class phylo.
data	A object of class phyDat containing sequences.
cost	A cost matrix for the transitions between two states.
sitewise	return CI/RI for alignment or sitewise

Details

The Consistency Index is defined as minimum number of changes divided by the number of changes required on the tree (parsimony score). The Consistency Index is equal to one if there is no homoplasy. And the Retention Index is defined as

$$RI = \frac{MaxChanges - ObsChanges}{MaxChanges - MinChanges}$$

Value

a scalar or vector with the CI/RI vector.

See Also

parsimony, pratchet, fitch, sankoff, bab, ancestral.pars

Examples

```
example(as.phylo.formula)
lab <- tr$tip.label
lab[79] <- "Herpestes fuscus"
tr$tip.label <- abbreviateGenus(lab)
X <- matrix(0, 112, 3, dimnames = list(tr$tip.label, c("Canis", "Panthera",
            "Canis_Panthera")))
desc_canis <- Descendants(tr, "Canis")[[1]]
desc_panthera <- Descendants(tr, "Panthera")[[1]]
X[desc_canis, c(1,3)] <- 1
```

```
X[desc_panthera, c(2,3)] <- 1
col <- rep("black", 112)
col[desc_panthera] <- "red"
col[desc_canis] <- "blue"
X <- phyDat(X, "USER", levels=c(0,1))
plot(tr, "f", tip.color=col)
# The first two sites are homoplase free!
CI(tr, X, sitewise=TRUE)
RI(tr, X, sitewise=TRUE)
```

`cladePar` *Utility function to plot.phylo*

Description

cladePar can help you coloring (choosing edge width/type) of clades.

Usage

```
cladePar(tree, node, edge.color = "red", tip.color = edge.color,
  edge.width = 1, edge.lty = "solid", x = NULL, plot = FALSE, ...)
```

Arguments

`tree`	an object of class phylo.
`node`	the node which is the common ancestor of the clade.
`edge.color`	see plot.phylo.
`tip.color`	see plot.phylo.
`edge.width`	see plot.phylo.
`edge.lty`	see plot.phylo.
`x`	the result of a previous call to cladeInfo.
`plot`	logical, if TRUE the tree is plotted.
`...`	Further arguments passed to or from other methods.

Value

A list containing the information about the edges and tips.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

`plot.phylo`

Examples

```
tree <- rtree(10)
plot(tree)
nodelabels()
x <- cladePar(tree, 12)
cladePar(tree, 18, "blue", "blue", x=x, plot=TRUE)
```

---

coalSpeciesTree *Species Tree*

Description

coalSpeciesTree estimates species trees and can handle multiple individuals per species.

Usage

```
coalSpeciesTree(tree, X = NULL, sTree = NULL)
```

Arguments

tree	an object of class multiPhylo
X	A phyDat object to define which individual belongs to which species.
sTree	A species tree which fixes the topology.

Details

coalSpeciesTree estimates a single linkage tree as suggested by Liu, Yu, and Pearl (2010) from the element wise minima of the cophenetic matrices of the gene trees. It extends speciesTree in ape as it allows that have several individuals per gene tree.

Value

The function returns an object of class phylo.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com> Emmanuel Paradies

References

Liu L, Yu L, Pearl DK (2010). "Maximum tree: A consistent estimator of the species tree." *Journal of Mathematical Biology*, **60**(1), 95–106. ISSN 0303-6812. doi:10.1007/s0028500902600.

See Also

speciesTree

Examples

```
## example in Liu et al. (2010)
tr1 <- read.tree(text = "(((B:0.05,C:0.05):0.01,D:0.06):0.04,A:0.1);")
tr2 <- read.tree(text = "(((A:0.07,C:0.07):0.02,D:0.09):0.03,B:0.12);")
TR <- c(tr1, tr2)
sp_tree <- coalSpeciesTree(TR)
```

---

codonTest *codonTest*

Description

Models for detecting positive selection

Usage

```
codonTest(tree, object, model = c("M0", "M1a", "M2a"),
  frequencies = "F3x4", opt_freq = FALSE, codonstart = 1,
  control = pml.control(maxit = 20), ...)
```

Arguments

tree	a phylogenetic tree.
object	an object of class phyDat.
model	a vector containing the substitution models to compare with each other or "all" to test all available models.
frequencies	a character string or vector defining how to compute the codon frequencies
opt_freq	optimize frequencies (so far ignored)
codonstart	an integer giving where to start the translation. This should be 1, 2, or 3, but larger values are accepted and have for effect to start the translation further within the sequence.
control	a list of parameters for controlling the fitting process.
...	further arguments passed to or from other methods.

Details

codonTest allows to test for positive selection similar to programs like PAML Yang (2007) or HyPhy Pond, Frost, and Muse (2005).

There are several options for deriving the codon frequencies. Frequencies can be "equal" (1/61), derived from nucleotide frequencies "F1x4" and "F3x4" or "empirical" codon frequencies. The frequencies taken using the empirical frequencies or estimated via maximum likelihood.

So far the M0 model (Goldman and Yang 2002), M1a and M2a are implemented. The M0 model is always computed the other are optional. The convergence may be very slow and sometimes fails.

Value

A list with an element called summary containing a data.frame with the log-likelihood, number of estimated parameters, etc. of all tested models. An object called posterior which contains the posterior probability for the rate class for each sites and the estimates of the defined models.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Nielsen R, Yang Z (1998). "Likelihood models for detecting positively selected amino acid sites and applications to the HIV-1 envelope gene." *Genetics*, **148**(3), 929–936.

Pond SLK, Frost SD, Muse SV (2005). "HyPhy: hypothesis testing using phylogenies." *Bioinformatics*, **21**(5), 676–679.

Yang Z (2014). *Molecular Evolution: A Statistical Approach*. Oxford University Press, Oxford.

Yang Z (2007). "PAML 4: phylogenetic analysis by maximum likelihood." *Molecular biology and evolution*, **24**(8), 1586–1591. *Genetics*, **148**: 929–936

See Also

pml, pmlMix, modelTest, AIC

Examples

```
## Not run:
# load woodmouse data from ape
data(woodmouse)
dat_codon <- dna2codon(as.phyDat(woodmouse), code=2)
tree <- NJ(dist.ml(dat_codon))
# optimize the model the old way
fit <- pml(tree, dat_codon, bf="F3x4")
M0 <- optim.pml(fit, model="codon1")
# Now using the codonTest function
fit_codon <- codonTest(tree, dat_codon)
fit_codon
plot(fit_codon, "M1a")

## End(Not run)
```

---

consensusNet *Computes a consensusNetwork from a list of trees*

Description

consensusNet computes a split network, i.e. an object of class networx from a list of trees, i.e. an class of class multiPhylo (Holland, Huber, Moulton, and Lockhart 2004).

Usage

```
consensusNet(obj, prob = 0.3, ...)
```

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

Value

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

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Holland BR, Huber KT, Moulton V, Lockhart PJ (2004). "Using Consensus Networks to Visualize Contradictory Evidence for Species Phylogeny." *Molecular Biology and Evolution*, **21**(7), 1459-1461. doi:10.1093/molbev/msh145.

See Also

splitsNetwork, neighborNet, lento, distanceHadamard, plot.networx, maxCladeCred

Examples

```
data(Laurasiatherian)
set.seed(1)
bs <- bootstrap.phyDat(Laurasiatherian, FUN = function(x)nj(dist.hamming(x)),
    bs=50)
cnet <- consensusNet(bs, 0.3)
plot(cnet, angle=-60)
## Not run:
library(rgl)
open3d()
plot(cnet, type = "3D", show.tip.label=FALSE, show.nodes=TRUE)
plot(cnet, type = "equal angle", show.edge.label=TRUE)

tmpfile <- normalizePath(system.file(
              "extdata/trees/RAxML_bootstrap.woodmouse", package="phangorn"))
trees <- read.tree(tmpfile)
cnet_woodmouse <- consensusNet(trees, 0.3)
plot(cnet_woodmouse, type = "equal angle", show.edge.label=TRUE)

## End(Not run)
```

cophenetic.networx *Pairwise Distances from a Phylogenetic Network*

Description

cophenetic.networx computes the pairwise distances between the pairs of tips from a phylogenetic network using its branch lengths.

Usage

```
## S3 method for class 'networx'
cophenetic(x)
```

Arguments

x	an object of class networx.

Value

an object of class dist, names are set according to the tip labels (as given by the element tip.label of the argument x).

Author(s)

Klaus Schliep

See Also

cophenetic for the generic function, neighborNet to construct a network from a distance matrix

Examples

```
example(neighborNet)
cophenetic(nnet)
```

delta.score *Computes the δ score*

Description

Computes the treelikeness

Usage

```
delta.score(x, arg = "mean", ...)
```

Arguments

x	an object of class phyDat
arg	Specifies the return value, one of "all", "mean" or "sd"
...	further arguments passed through dist.hamming

Value

A vector containing the δ scores.

Author(s)

Alastair Potts and Klaus Schliep

References

Gray RD, Bryant D, Greenhill SJ (2010). "On the shape and fabric of human history." *Philosophical Transactions of the Royal Society B: Biological Sciences*, **365**(1559), 3923.

Holland BR, Huber KT, Dress A, Moulton V (2002). "δ Plots: A Tool for Analyzing Phylogenetic Distance Data." *Molecular Biology and Evolution*, **19**(12), 2051-2059. ISSN 0737-4038. doi:10.1093/oxfordjournals.molbev.a004030.

See Also

dist.hamming

Examples

```
data(yeast)
hist(delta.score(yeast, "all"))
```

---

densiTree *Plots a densiTree.*

Description

An R function to plot trees similar to those produced by DensiTree Bouckaert (2010).

Usage

```
densiTree(x, type = "phylogram", ..., alpha = 1/length(x),
  consensus = NULL, direction = "rightwards", optim = FALSE,
  scaleX = FALSE, col = 1, width = 1, lty = 1, cex = 0.8, font = 3,
  tip.color = 1, adj = 0, srt = 0, underscore = FALSE,
  label.offset = 0, scale.bar = TRUE, jitter = list(amount = 0, random =
  TRUE), tip.dates = NULL, xlim = NULL, ylim = NULL,
  show.consensus = FALSE, side = NULL)
```

Arguments

x	an object of class multiPhylo or an object of class pml. If the later is the case it need to contain a bootstrap sample is extracted.
type	a character string specifying the type of phylogeny, so far "cladogram" (default) or "phylogram" are supported.
...	further arguments to be passed to plot.
alpha	parameter for semi-transparent colors.
consensus	A tree or character vector which is used to define the order of the tip labels.
direction	a character string specifying the direction of the tree. Four values are possible: "rightwards" (the default), "leftwards", "upwards", and "downwards".
optim	not yet used.
scaleX	scale trees to have identical heights.
col	a scalar or vector giving the colours used to draw the edges for each plotted phylogeny. These are taken to be in the same order than input trees x. If fewer colours are given than the number of trees, then the colours are recycled.
width	edge width.
lty	line type.
cex	a numeric value giving the factor scaling of the tip labels.
font	an integer specifying the type of font for the labels: 1 (plain text), 2 (bold), 3 (italic, the default), or 4 (bold italic).
tip.color	color of the tip labels.
adj	a numeric specifying the justification of the text strings of the labels: 0 (left-justification), 0.5 (centering), or 1 (right-justification).
srt	a numeric giving how much the labels are rotated in degrees.
underscore	a logical specifying whether the underscores in tip labels should be written as spaces (the default) or left as are (if TRUE).
label.offset	a numeric giving the space between the nodes and the tips of the phylogeny and their corresponding labels.
scale.bar	a logical specifying whether add scale.bar to the plot.
jitter	allows to shift trees. a list with two arguments: the amount of jitter and random or equally spaced (see details below)
tip.dates	A named vector of sampling times associated with the tips.
xlim	the x limits of the plot.
ylim	the y limits of the plot.
show.consensus	superimpose the consensus tree?
side	a numeric value specifying the side where the axis is plotted: 1: below, 2: left, 3: above, 4: right. By default, this is taken from the direction of the plot.

Details

If no consensus tree is provided `densiTree` computes a consensus tree, and if the input trees have different labels a mrp.supertree as a backbone. This should avoid too many unnecessary crossings of edges. Trees should be rooted, other wise the output may not be visually pleasing. `jitter` shifts trees a bit so that they are not exactly on top of each other. If `amount == 0`, it is ignored. If `random=TRUE` the result of the permutation is `runif(n, -amount, amount)`, otherwise `seq(-amount, amount, length=n)`, where `n <- length(x)`.

Value

`densiTree` returns silently x.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Bouckaert RR (2010). "DensiTree: making sense of sets of phylogenetic trees." *Bioinformatics*, **26**(10), 1372-1373. doi:10.1093/bioinformatics/btq110.

See Also

`plot.phylo`, `plot.networx`, `jitter`, `rtt`, `axisPhylo`

Examples

```
data(Laurasiatherian)
set.seed(1)
bs <- bootstrap.phyDat(Laurasiatherian, FUN =
   function(x) upgma(dist.hamming(x)), bs=25)
# cladogram nice to show topological differences
densiTree(bs, type="cladogram", col="blue")
densiTree(bs, type="phylogram", col="green", direction="downwards", width=2)
# show consensus tree
tree_compat <- allCompat(bs, rooted=TRUE) |> add_edge_length(bs)
densiTree(bs, type="phylogram", col="green", consensus = tree_compat,
  show.consensus=TRUE)

# plot five trees slightly shifted, no transparent color
densiTree(bs[1:5], type="phylogram", col=1:5, width=2, jitter=
    list(amount=0.3, random=FALSE), alpha=1)
## Not run:
# phylograms are nice to show different age estimates
require(PhyloOrchard)
data(BinindaEmondsEtAl2007)
BinindaEmondsEtAl2007 <- .compressTipLabel(BinindaEmondsEtAl2007)
densiTree(BinindaEmondsEtAl2007, type="phylogram", col="red")

## End(Not run)
```

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. designTree also computes a contrast matrix if the method is "rooted".

Usage

```
designTree(tree, method = "unrooted", sparse = FALSE, tip.dates = NULL,
  calibration = NULL, ...)

nnls.tree(dm, tree, method = c("unrooted", "ultrametric", "tipdated"),
  rooted = NULL, trace = 1, weight = NULL, balanced = FALSE,
  tip.dates = NULL, calibration = NULL)

nnls.phylo(x, dm, method = "unrooted", trace = 0, ...)

nnls.splits(x, dm, trace = 0, eps = 1e-08)

nnls.networx(x, dm, eps = 1e-08)

designSplits(x, splits = "all", ...)
```

Arguments

tree	an object of class phylo
method	compute an "unrooted", "ultrametric" or "tipdated" tree.
sparse	return a sparse design matrix.
tip.dates	a named vector of sampling times associated to the tips of the tree.
calibration	a named vector of calibration times associated to nodes of the tree.
...	further arguments, passed to other methods.
dm	a distance matrix.
rooted	compute a "ultrametric" or "unrooted" tree (better use method).
trace	defines how much information is printed during optimization.
weight	vector of weights to be used in the fitting process. Weighted least squares is used with weights w, i.e., sum(w * e^2) is minimized.
balanced	use weights as in balanced fastME
x	number of taxa.
eps	minimum edge length (default s 1e-8).
splits	one of "all", "star".

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, rtt, distanceHadamard, splitsNetwork, 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)
```

---

discrete.gamma *Discrete Gamma and Beta distribution*

Description

discrete.gamma internally used for the likelihood computations in pml or optim.pml. The functions plot_gamma_plus_inv and plotRates are useful in EDA and understand how discrete rate models work and where it limitations are (Ferretti, Golubchik, Di Lauro, Ghafari, Villabona-Arenas, Atkins, Fraser, and Hall 2026). plotRates plots the cumulative distribution function based on the parsimony score for the data and the discrete rate model.

Usage

```
discrete.gamma(alpha, k, w = NULL)

discrete.beta(shape1, shape2, k)

plot_gamma_plus_inv(w = NULL, g = NULL, shape = 1, inv = 0, k = 4,
  discrete = TRUE, cdf = TRUE, append = FALSE, xlab = "x",
  ylab = ifelse(cdf, "F(x)", "f(x)"), xlim = NULL, verticals = FALSE,
  edge.length = NULL, site.rate = "gamma", ...)

plotRates(obj, cdf.color = "blue", main = "", rug = FALSE, xlim = NULL,
  append = FALSE, ...)
```

Arguments

alpha	Shape parameter of the gamma distribution.
k	Number of intervals of the discrete gamma distribution.
w	proportion of rate g.
shape1, shape2	non-negative parameters of the Beta distribution.
g	rates of discrete distribution.
shape	Shape parameter of the gamma distribution.
inv	Proportion of invariable sites.
discrete	logical whether to plot discrete (default) or continuous pdf or cdf.
cdf	logical whether to plot the cumulative distribution function or density / probability function.
append	logical; if TRUE only add to an existing plot.
xlab	a label for the x axis, defaults to a description of x.
ylab	a label for the y axis, defaults to a description of y.
xlim	the x limits of the plot.
verticals	logical; if TRUE, draw vertical lines at steps.
edge.length	Total edge length (sum of all edges in a tree).
site.rate	Indicates what type of gamma distribution to use. Options are "gamma" (Yang 1994), "gamma_quadrature" using Laguerre quadrature approach of Felsenstein (2001) or "variable_gamma".
...	Further arguments passed to or from other methods.
obj	an object of class pml
cdf.color	color of the cdf.
main	a main title for the plot.
rug	logical; if TRUE a rug is added to the plot.

Value

discrete.gamma returns a matrix.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Ferretti L, Golubchik T, Di Lauro F, Ghafari M, Villabona-Arenas J, Atkins KE, Fraser C, Hall M (2026). "Biased Estimates of Phylogenetic Branch Lengths Resulting from the Discretised Gamma Model of Site Rate Heterogeneity." *Systematic Biology*, syag037. ISSN 1063-5157. doi:10.1093/sysbio/syag037.

See Also

pml.fit, stepfun, pgamma, pbeta

Examples

```
discrete.gamma(1, 4)

old.par <- par(no.readonly = TRUE)
par(mfrow = c(2,1))
plot_gamma_plus_inv(shape=2, discrete = FALSE, cdf=FALSE)
plot_gamma_plus_inv(shape=2, append = TRUE, cdf=FALSE)

plot_gamma_plus_inv(shape=2, discrete = FALSE)
plot_gamma_plus_inv(shape=2, append = TRUE)
par(old.par)

data(Laurasiatherian)
fit <- pml_bb(Laurasiatherian, "JC+G(4)", rearrangement = "none")
plotRates(fit)
```

---

dist.hamming *Pairwise Distances from Sequences*

Description

dist.hamming, dist.ml and dist.logDet compute pairwise distances for an object of class phyDat and additional DNAbin or AAbin from ape. dist.ml uses DNA / AA sequences to compute distances under different substitution models.

Usage

```
dist.hamming(x, ratio = TRUE, exclude = "pairwise")

dist.ml(x, model = "JC69", exclude = "pairwise", bf = NULL, Q = NULL,
  k = 1L, shape = 1, ...)

dist.logDet(x)
```

Arguments

x	An object of class phyDat
ratio	Compute uncorrected ('p') distance or character difference.
exclude	One of "none", "all", "pairwise" indicating whether to delete the sites with gaps, missing data (or ambiguous states). See details below.
model	One of "JC69", "F81" or one of 17 amino acid models see details.
bf	A vector of base frequencies.
Q	A vector containing the lower triangular part of the rate matrix.
k	Number of intervals of the discrete gamma distribution.
shape	Shape parameter of the gamma distribution.
...	Further arguments passed to or from other methods.

Details

So far 17 amino acid models are supported ("WAG", "JTT", "LG", "Dayhoff", "cpREV", "mtmam", "mtArt", "MtZoa", "mtREV24", "VT","RtREV", "HIVw", "HIVb", "FLU", "Blosum62", "Dayhoff_DCMut" and "JTT_DCMut") and additional rate matrices and frequencies can be supplied.

The Jukes-Cantor model "JC69" assumnes equal base frequencies and equal rates and is a sepqecial case of the Mk model. When calling a "JC69" model with different state space a "Mk" model is fitted. The "F81" model uses empirical base frequencies. `dist.dna` offers additional nucleotide models.

The argument `exclude` decides how gaps / ambiguous data / missing data are treated. Usually gaps are treated as ambiguous states, but you can give gaps its on state `gap_as_state`. `exclude="none"` keeps all ambiguous data. The behavior of `dist.ml` is in this case these same you would achieve using `optim.pml` to compute pairwise distances, it might be a bit odd. `exclude="all"` removes all sites with ambiguous states and all gaps if these are coded as ambiguous states. This can lead to the situation that there only few sites if any fo the alignment left. Safer is therefore to use `exclude="pairwise"` which only removes sites which are ambiguous for each pair of sequences.

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.

Jukes TH and Cantor CR (1969). *Evolution of Protein Molecules*. New York: Academic Press. 21–132.

McGuire, G., Prentice, M. J. and Wright, F. (1999). Improved error bounds for genetic distances from DNA sequences. *Biometrics*, **55**, 1064–1070.

See Also

For more distance methods for nucleotide data see `dist.dna` and `dist.p` for pairwise polymorphism p-distances. `writeDist` for export and import distances.

Examples

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

```
dm3 <- dist.ml(Laurasiatherian)
tree3 <- NJ(dm3)
treedist(tree1,tree3)
# F81 + Gamma
dm4 <- dist.ml(Laurasiatherian, model="F81", k=4, shape=0.4)
tree4 <- NJ(dm4)
treedist(tree1,tree4)
treedist(tree3,tree4)
```

`dist.p` *Pairwise Polymorphism P-Distances from DNA Sequences*

Description

This function computes a matrix of pairwise uncorrected polymorphism p-distances. Polymorphism p-distances include intra-individual site polymorphisms (2ISPs; e.g. "R") when calculating genetic distances.

Usage

```
dist.p(x, cost = "polymorphism", ignore.indels = TRUE)
```

Arguments

`x`	a matrix containing DNA sequences; this must be of class "phyDat" (use as.phyDat to convert from DNAbin objects).
`cost`	A cost matrix or "polymorphism" for a predefined one.
`ignore.indels`	a logical indicating whether gaps are treated as fifth state or not. Warning, each gap site is treated as a characters, so an an indel that spans a number of base positions would be treated as multiple character states.

Details

The polymorphism p-distances (Potts et al. 2014) have been developed to analyse intra-individual variant polymorphism. For example, the widely used ribosomal internal transcribed spacer (ITS) region (e.g. Alvarez and Wendel, 2003) consists of 100's to 1000's of units within array across potentially multiple nucleolus organizing regions (Bailey et al., 2003; Goeker and Grimm, 2008). This can give rise to intra-individual site polymorphisms (2ISPs) that can be detected from direct-PCR sequencing or cloning . Clone consensus sequences (see Goeker and Grimm, 2008) can be analysed with this function.

Value

an object of class `dist`.

Author(s)

Klaus Schliep and Alastair Potts

References

Alvarez, I., and J. F. Wendel. (2003) Ribosomal ITS sequences and plant phylogenetic inference. *Molecular Phylogenetics and Evolution*, **29**, 417–434.

Bailey, C. D., T. G. Carr, S. A. Harris, and C. E. Hughes. (2003) Characterization of angiosperm nrDNA polymorphism, paralogy, and pseudogenes. *Molecular Phylogenetics and Evolution* **29**, 435–455.

Goeker, M., and G. Grimm. (2008) General functions to transform associate data to host data, and their use in phylogenetic inference from sequences with intra-individual variability. *BMC Evolutionary Biology*, **8**:86.

Potts, A.J., T.A. Hedderson, and G.W. Grimm. (2014) Constructing phylogenies in the presence of intra-individual site polymorphisms (2ISPs) with a focus on the nuclear ribosomal cistron. *Systematic Biology*, **63**, 1–16

See Also

dist.dna, dist.hamming

Examples

```
data(Laurasiatherian)
laura <- as.DNAbin(Laurasiatherian)

dm <- dist.p(Laurasiatherian, "polymorphism")

#########################################################
# Dealing with indel 2ISPs
# These can be coded using an "x" in the alignment. Note
# that as.character usage in the read.dna() function.
#########################################################
cat("3 5",
    "No305     ATRA-",
    "No304     ATAYX",
    "No306     ATAGA",
    file = "exdna.txt", sep = "\n")
(ex.dna <- read.dna("exdna.txt", format = "sequential", as.character=TRUE))
dat <- phyDat(ex.dna, "USER", levels=unique(as.vector(ex.dna)))
dist.p(dat)

unlink("exdna.txt")
```

---

distanceHadamard *Distance Hadamard*

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.networx, neighborNet

Examples

```
data(yeast)
dm <- dist.hamming(yeast)
dm <- as.matrix(dm)
fit <- distanceHadamard(dm)
lento(fit)
plot(as.networx(fit))
```

---

dna2codon *Translate nucleic acid sequences into codons*

Description

The function transforms dna2codon DNA sequences to codon sequences, codon2dna transform the other way. dna2codon translates nucleotide to amino acids using the function trans.

Usage

```
dna2codon(x, codonstart = 1, code = 1, ambiguity = "---", ...)

codon2dna(x)

dna2aa(x, codonstart = 1, code = 1)
```

Arguments

x	An object containing sequences.
codonstart	an integer giving where to start the translation. This should be 1, 2, or 3, but larger values are accepted and have for effect to start the translation further within the sequence.
code	The ncbi genetic code number for translation (see details). By default the standard genetic code is used.
ambiguity	character for ambiguous character and no contrast is provided.
...	further arguments passed to or from other methods.

Details

The following genetic codes are described here. The number preceding each corresponds to the code argument.

1	standard
2	vertebrate.mitochondrial
3	yeast.mitochondrial
4	protozoan.mitochondrial+mycoplasma
5	invertebrate.mitochondrial
6	ciliate+dasycladaceal
9	echinoderm+flatworm.mitochondrial
10	euplotid
11	bacterial+plantplastid
12	alternativeyeast
13	ascidian.mitochondrial
14	alternativeflatworm.mitochondrial
15	blepharism
16	chlorophycean.mitochondrial
21	trematode.mitochondrial
22	scenedesmus.mitochondrial
23	thraustochytrium.mitochondria
24	Pterobranchia.mitochondrial
25	CandidateDivision.SR1+Gracilibacteria
26	Pachysolen.tannophilus

Alignment gaps and ambiguities are currently ignored and sites containing these are deleted.

Value

The functions return an object of class phyDat.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

https://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/index.cgi?chapter=cgencodes

See Also

trans, phyDat and the chapter 4 in the vignette("phangorn-specials", package="phangorn")

Examples

```
data(Laurasiatherian)
class(Laurasiatherian)
Laurasiatherian
dna2codon(Laurasiatherian)
```

---

fitch *Parsimony tree.*

Description

parsimony returns the parsimony score of a tree using either the sankoff or the fitch algorithm, for more details see for example Felsenstein (2004). pratchet implements the parsimony ratchet Nixon (1999) and is the preferred way to search for the best parsimony tree. For small number of taxa the function bab can be used to compute all most parsimonious trees.

Usage

```
fitch(tree, data, site = "pscore")

random.addition(data, tree = NULL, method = "fitch")

parsimony(tree, data, method = "fitch", cost = NULL, site = "pscore")

optim.parsimony(tree, data, method = "fitch", cost = NULL,
  trace = !getOption("quiet"), rearrangements = "SPR", ...)

pratchet(data, start = NULL, method = "fitch", maxit = 1000,
  minit = 100, k = 10, trace = !getOption("quiet"), all = FALSE,
  rearrangements = "SPR", perturbation = "ratchet", ...)

sankoff(tree, data, cost = NULL, site = "pscore")
```

Arguments

tree	tree to start the nni search from.
data	A object of class phyDat containing sequences.
site	return either 'pscore' or 'site' wise parsimony scores.
method	one of 'fitch' or 'sankoff'.
cost	A cost matrix for the transitions between two states.
trace	defines how much information is printed during optimization.
rearrangements	SPR or NNI rearrangements.
...	Further arguments passed to or from other methods (e.g. model="sankoff" and cost matrix).
start	a starting tree can be supplied.
maxit	maximum number of iterations in the ratchet.
minit	minimum number of iterations in the ratchet.
k	number of rounds ratchet is stopped, when there is no improvement.
all	return all equally good trees or just one of them.
perturbation	whether to use "ratchet", "random_addition" or "stochastic" (nni) for shuffling the tree.

Details

`optim.parsimony` optimizes the topology using either Nearest Neighbor Interchange (NNI) rearrangements or sub tree pruning and regrafting (SPR) and is used inside `pratchet`. `random.addition` can be used to produce starting trees and is an option for the argument perturbation in `pratchet`.

The "SPR" rearrangements are so far only available for the "fitch" method, "sankoff" only uses "NNI". The "fitch" algorithm only works correct for binary trees.

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 Rarsimony Analysis." *Cladistics*, **15**, 407–414.

See Also

bab, CI, RI, acctran, nni, NJ, pml, getClans, ancestral.pars, bootstrap.pml

Examples

```
set.seed(3)
data(Laurasiatherian)
dm <- dist.hamming(Laurasiatherian)
tree <- NJ(dm)
parsimony(tree, Laurasiatherian)
treeRA <- random.addition(Laurasiatherian)
treeSPR <- optim.parsimony(tree, Laurasiatherian)

# lower number of iterations for the example (to run less than 5 seconds),
# keep default values (maxit, minit, k) or increase them for real life
# analyses.
treeRatchet <- pratchet(Laurasiatherian, start=tree, maxit=100,
                        minit=5, k=5, trace=0)
# assign edge length (number of substitutions)
treeRatchet <- acctran(treeRatchet, Laurasiatherian)
# remove edges of length 0
treeRatchet <- di2multi(treeRatchet)

plot(midpoint(treeRatchet))
add.scale.bar(0,0, length=100)

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

---

gap_as_state *Treat gaps as a state*

Description

The function gap_as_state changes the contrast of an phyDat object to treat as its own state. Internally phyDat are stored similar to a factor objects and only the contrast matrix and some attributes change.

Usage

```
gap_as_state(obj, gap = "-", ambiguous = "?", latag2n = FALSE)

gap_as_ambiguous(obj, gap = "-")

has_gap_state(obj)
```

Arguments

obj	An object of class phyDat.
gap	a character which codes for the gaps (default is "-").
ambiguous	a character which codes for the ambiguous state
latag2n	transform leading and trailing gaps as ambiguous states.

Value

The functions return an object of class phyDat.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

phyDat, latag2n.phyDat, latag2n, ancestral.pml, gap_as_state

Examples

```
data(Laurasiatherian)
tmp <- gap_as_state(Laurasiatherian)
contr <- attr(tmp, "contrast")
rownames(contr) <- attr(tmp, "allLevels")
contr
```

---

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)

getSlices(tree)

getClips(tree, all = TRUE)

getDiversity(tree, x, norm = TRUE, var.names = NULL, labels = "new")

## S3 method for class 'clanistics'
summary(object, ...)

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.
object	an object for which a summary is desired.
...	Further arguments passed to or from other methods.
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 leave).

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, Lopez, Boucher, Koenig, and Bapteste 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 categorial 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 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, Lopez, Lapointe, and Bapteste (2011) 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@gmail.com>

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

References

Lapointe F, Lopez P, Boucher Y, Koenig J, Bapteste E (2010). "Clanistics: a multi-level perspective for harvesting unrooted gene trees." *Trends in Microbiology*, **18**(8), 341–347. doi:10.1016/j.tim.2010.03.009.

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

Wilkinson M, McInerney JO, Hirt RP, Foster PG, Embley TM (2007). "Of clades and clans: terms for phylogenetic relationships in unrooted trees." *Trends in Ecology and Evolution*, **22**(3), 114–115. doi:10.1016/j.tree.2007.01.002.

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(0.5, 0.4, 0.1)),
   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)
```

---

getRoot *Tree manipulation*

Description

midpoint performs midpoint rooting of a tree. pruneTree produces a consensus tree. pruneTree prunes back a tree and produces a consensus tree, for trees already containing nodelabels. It assumes that nodelabels are numerical or character that allows conversion to numerical, it uses as.numeric(as.character(tree$node.labels)) to convert them. midpoint by default assumes that node labels contain support values. This works if support values are computed from splits, but should be recomputed for clades. keep_as_tip takes a list of tips and/or node labels and returns a tree pruned to those. If node label, then it prunes all descendants of that node until that internal node becomes a tip.

Usage

```
getRoot(tree)

midpoint(tree, node.labels = "support", ...)

## S3 method for class 'phylo'
midpoint(tree, node.labels = "support", ...)

## S3 method for class 'multiPhylo'
midpoint(tree, node.labels = "support", ...)

pruneTree(tree, ..., FUN = ">=")

keep_as_tip(tree, labels, collapse.singles = TRUE)
```

Arguments

tree	an object of class phylo.
node.labels	are node labels 'support' values (edges), 'label' or should labels get 'deleted'?
...	further arguments, passed to other methods.
FUN	a function evaluated on the nodelabels, result must be logical.
labels	tip and node labels to keep as tip labels in the tree
collapse.singles	a logical specifying whether to delete the internal nodes of degree 2. Nodes which are part of the labels are not collapsed.

Value

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

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

consensus, root, multi2di

Examples

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

tree2 <- midpoint(tree)
tree3 <- pruneTree(tree, 0.5)

old.par <- par(no.readonly = TRUE)
par(mfrow = c(3,1))
plot(tree, show.node.label=TRUE)
plot(tree2, show.node.label=TRUE)
plot(tree3, show.node.label=TRUE)
par(old.par)

# prune to node labels
tree <- read.tree(text="((t1:.1,t2:.1)A:.2,(t3:.2,t4:.2)B:.1);")
plot(tree, show.node.label = TRUE)
tree_tip <- keep_as_tip(tree, c("A", "t3", "t4"))
plot(tree_tip, show.node.label = TRUE)
```

---

hadamard *Hadamard Matrices and Fast Hadamard Multiplication*

Description

A collection of functions to perform Hadamard conjugation Hv of a Hadamard matrix H with a vector v using fast Hadamard multiplication.

Usage

```
hadamard(x)

fhm(v)

h4st(obj, levels = c("a", "c", "g", "t"))

h2st(obj, eps = 0.001)
```

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.
levels	levels of the sequences.
eps	Threshold value for splits.

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.networx

Examples

```
H <- hadamard(3)
v <- 1:8
H %*% v
fhm(v)

data(yeast)

# RY-coding
dat_ry <- acgt2ry(yeast)
fit2 <- h2st(dat_ry)
```

```
lento(fit2)

# write.nexus.splits(fit2, file = "test.nxs")
# read this file into Spectronet or SplitsTree to show the network

fit4 <- h4st(yeast)
old.par <- par(no.readonly = TRUE)
par(mfrow=c(3,1))
lento(fit4[[1]], main="Transversion")
lento(fit4[[2]], main="Transition 1")
lento(fit4[[3]], main="Transition 2")
par(old.par)
```

---

heatmapp *heatmapp*

Description

Plots a heat map with ancestral states.

Usage

```
heatmapp(x, ...)

## S3 method for class 'phylo'
heatmapp(x, y, use.edge.length = TRUE, align_label = TRUE,
  clade = NULL, select = NULL, cex.lab = 1, nodes = FALSE, ...)

## S3 method for class 'pml'
heatmapp(x, ...)

## S3 method for class 'ancestral'
heatmapp(x, nodes = TRUE, ...)
```

Arguments

x	an object of class ancestral or phyDat.
...	Further arguments passed to or from other methods.
y	an object of class phyDat.
use.edge.length	a logical indicating whether to use the edge lengths of the phylogeny to draw the branches (the default) or not (if FALSE). This option has no effect if the object of class "phylo" has no 'edge.length' element.
align_label	a logical value or an integer. If TRUE, the tips are aligned and dotted lines are drawn between the nodes of the tree and the labels.
clade	a node number or label to extract the clade from the tree.

select	a subset of characters, columns in the alignment.
cex.lab	font size of the labels.
nodes	Show ancestral sequences at internal nodes.

Details

heatmapp plots a heatmap, a phylogeny together with the alignment. For ancestral reconstructions it additionally plots the joint distribution or the most likely state of the internal nodes. This eases comparison of ancestors with related Descendant sequences.

heatmapp is based on the ape function splot.phylo and image.AAbin or image.AAbin.

Value

heatmapp returns silently a x

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

anc_pml, plotAnc, image.DNAbin, image.AAbin, plot.phylo

Examples

```
data(woodmouse)
wm <- as.phyDat(woodmouse)
tree <- pratchet(wm)
tree <- makeNodeLabel(tree)
anc_mp <- anc_pars(tree, wm)
op <- par(mar=c(4,1,4,5))
heatmapp(anc_mp)
heatmapp(tree, woodmouse)
par(op)
```

---

identify.networx *Identify splits in a network*

Description

identify.networx reads the position of the graphics pointer when the mouse button is pressed. It then returns the split belonging to the edge closest to the pointer. The network must be plotted beforehand.

Usage

```
## S3 method for class 'networx'
identify(x, quiet = FALSE, ...)
```

Arguments

x	an object of class networx
quiet	a logical controlling whether to print a message inviting the user to click on the tree.
...	further arguments to be passed to or from other methods.

Value

identify.networx returns a splits object.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

plot.networx, identify

Examples

```
## Not run:
data(yeast)
dm <- dist.ml(yeast)
nnet <- neighborNet(dm)
plot(nnet)
identify(nnet) # click close to an edge

## End(Not run)
```

---

image.phyDat *Plot of a Sequence Alignment*

Description

This function plots an image of an alignment of sequences.

Usage

```
## S3 method for class 'phyDat'
image(x, ...)

## S3 method for class 'ancestral'
image(x, ...)
```

Arguments

x	an object containing sequences, an object of class phyDat.
...	further arguments passed to or from other methods.

Details

A wrapper for using image.DNAbin and image.AAbin. Codons triplets are transformed and handled as nucleotide sequences. So far it is not yet possible to plot data with type="USER".

Value

image.phyDat returns silently x.

See Also

image.DNAbin, image.AAbin

Examples

```
data("chloroplast")
image(chloroplast[, 1:50], scheme="Clustal", show.aa = TRUE)
```

---

latag2n.phyDat *Replace leading and trailing alignment gaps with an ambiguous state*

Description

Substitutes leading and trailing alignment gaps in aligned sequences into N (i.e., A, C, G, or T) or ?. The gaps in the middle of the sequences are left unchanged.

Usage

```
latag2n.phyDat(x, amb = ifelse(attr(x, "type") == "DNA", "N", "?"),
  gap = "-", ...)
```

Arguments

x	an object of class phyDat.
amb	character of the ambiguous state t replace the gaps.
gap	gap parameter to replace.
...	Further arguments passed to or from other methods.

Value

returns an object of class phyDat.

See Also

latag2n, ancestral.pml, gap_as_state

Examples

```
x <- phyDat(matrix(c("-", "A", "G", "-", "T", "C"), 2, 3))
y <- latag2n.phyDat(x)
image(x)
image(y)
```

Laurasiatherian *Laurasiatherian data (AWCMEE)*

Description

Laurasiatherian RNA sequence data

Source

Data have been taken from the former repository of the Allan Wilson Centre and were converted to R format by <klaus.schliep@gmail.com>.

Examples

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

---

ldfactorial *Arithmetic Operators*

Description

double factorial function

Usage

```
ldfactorial(x)

dfactorial(x)
```

Arguments

x	a numeric scalar or vector

Value

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

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

factorial, howmanytrees

Examples

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

---

lento *Lento plot*

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, trivial = FALSE,
  col = rgb(0, 0, 0, 0.5), ...)
```

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).
col	color for the splits / bipartition.
...	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 pinninpeds. *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)
```

---

lli *Internal maximum likelihood functions.*

Description

These functions are internally used for the likelihood computations in pml or optim.pml.

Usage

```
lli(data, tree = NULL, ...)

pml.fit(tree, data, bf = rep(1/length(levels), length(levels)), shape = 1,
  k = 1, Q = rep(1, length(levels) * (length(levels) - 1)/2),
  levels = attr(data, "levels"), inv = 0, rate = 1, g = NULL,
  w = NULL, eig = NULL, INV = NULL, ll.0 = NULL, llMix = NULL,
  wMix = 0, ..., site = FALSE, ASC = FALSE, site.rate = "gamma")
```

Arguments

data	An alignment, object of class phyDat.
tree	A phylogenetic tree, object of class phylo.
...	Further arguments passed to or from other methods.
bf	Base frequencies.
shape	Shape parameter of the gamma distribution.
k	Number of intervals of the discrete gamma distribution.
Q	A vector containing the lower triangular part of the rate matrix.

levels	The alphabet used e.g. c("a", "c", "g", "t") for DNA
inv	Proportion of invariable sites.
rate	Rate.
g	vector of quantiles (default is NULL)
w	vector of probabilities (default is NULL)
eig	Eigenvalue decomposition of Q
INV	Sparse representation of invariant sites
ll.0	default is NULL
llMix	default is NULL
wMix	default is NULL
site	return the log-likelihood or vector of sitewise likelihood values
ASC	ascertainment bias correction (ASC), allows to estimate models like Lewis' Mkv.
site.rate	Indicates what type of gamma distribution to use. Options are "gamma" approach of Yang 1994 (default), "gamma_quadrature" after the Laguerre quadrature approach of Felsenstein 2001 and "free_rate".

Details

These functions are exported to be used in different packages so far only in the package coalescentMCMC, but are not intended for end user. Most of the functions call C code and are far less forgiving if the import is not what they expect than pml.

Value

pml.fit returns the log-likelihood.

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.

See Also

pml, pml_bb, pmlPart, pmlMix

Examples

```
data(Laurasiatherian)
tree <- NJ(dist.ml(Laurasiatherian))
bf <- rep(0.25, 4)
eig <- edQt()
pml.init(Laurasiatherian)
pml.fit(tree, Laurasiatherian, bf=bf, eig=eig)
pml.free()
pml(tree, Laurasiatherian) |> logLik()
```

---

mast *Maximum agreement subtree*

Description

`mast` computes the maximum agreement subtree (MAST).

Usage

```
mast(x, y, tree = TRUE, rooted = TRUE)
```

Arguments

`x`	a tree, i.e. an object of class `phylo`.
`y`	a tree, i.e. an object of class `phylo`.
`tree`	a logical, if TRUE returns a tree other wise the tip labels of the the maximum agreement subtree.
`rooted`	logical if TRUE treats trees as rooted otherwise unrooted.

Details

The code is derived from the code example in Valiente (2009). The version for the unrooted trees is much slower.

Value

`mast` returns a vector of the tip labels in the MAST.

Author(s)

Klaus Schliep <`klaus.schliep@gmail.com`> based on code of Gabriel Valiente

References

G. Valiente (2009). *Combinatorial Pattern Matching Algorithms in Computational Biology using Perl and R*. Taylor & Francis/CRC Press

See Also

SPR.dist

Examples

```
tree1 <- rtree(100)
tree2 <- rNNI(tree1, 20)
tips <- mast(tree1, tree2)
```

---

maxCladeCred *Maximum clade credibility tree*

Description

maxCladeCred computes the maximum clade credibility tree from a sample of trees. So far just the best tree is returned. No annotations or transformations of edge length are performed and the edge length are taken from the tree.

Usage

```
maxCladeCred(x, tree = TRUE, part = NULL, rooted = TRUE)

mcc(x, tree = TRUE, part = NULL, rooted = TRUE)

allCompat(x, rooted = FALSE)
```

Arguments

x	x is an object of class multiPhylo or phylo
tree	logical indicating whether return the tree with the clade credibility (default) or the clade credibility score for all trees.
part	a list of partitions as returned by prop.part
rooted	logical, if FALSE the tree with highest maximum bipartition credibility is returned.

Details

If a list of partition is provided then the clade credibility is computed for the trees in x.

allCompat returns a 50% majority rule consensus tree with added compatible splits similar to the option allcompat in MrBayes. This tree has no edge length.

add_edge_length can be used to add edge lengths computed from the sample of trees.

Value

a tree (an object of class phylo) with the highest clade credibility or a numeric vector of clade credibilities for each tree.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

consensus, consensusNet, prop.part, bootstrap.pml, plotBS, transferBootstrap, add_edge_length, add_boxplot

Examples

```
data(Laurasiatherian)
set.seed(42)
bs <- bootstrap.phyDat(Laurasiatherian,
  FUN = function(x)upgma(dist.hamming(x)), bs=100)

strict_consensus <- consensus(bs)
majority_consensus <- consensus(bs, p=0.5)
all_compat <- allCompat(bs)
max_clade_cred <- maxCladeCred(bs)

old.par <- par(no.readonly = TRUE)
par(mfrow = c(2,2), mar = c(1,4,1,1))
plot(strict_consensus, main="Strict consensus tree")
plot(majority_consensus, main="Majority consensus tree")
plot(all_compat, main="Majority consensus tree with compatible splits")
plot(max_clade_cred, main="Maximum clade credibility tree")

par(mfrow = c(2,1))
plot(max_clade_cred, main="Edge length from tree")
add_boxplot(max_clade_cred, bs)
max_clade_cred_2 <- add_edge_length(max_clade_cred, bs)
plot(max_clade_cred_2, main="Edge length computed from sample")
add_boxplot(max_clade_cred_2, bs)

par(old.par)

# compute clade credibility for trees given a prop.part object
pp <- prop.part(bs)
tree <- rNNI(bs[[1]], 20)
maxCladeCred(c(tree, bs[[1]]), tree=FALSE, part = pp)
# first value likely be -Inf
```

---

mites *Morphological characters of Mites (Schäffer et al. 2010)*

Description

Matrix for morphological characters and character states for 12 species of mites. See vignette '02_Phylogenetic trees from morphological data' for examples to import morphological data.

References

Schäffer, S., Pfingstl, T., Koblmüller, S., Winkler, K. A., Sturmbauer, C., & Krisper, G. (2010). Phylogenetic analysis of European Scutovertex mites (Acari, Oribatida, Scutoverticidae) reveals paraphyly and cryptic diversity: a molecular genetic and morphological approach. *Molecular Phylogenetics and Evolution*, **55(2)**, 677–688.

Examples

```
data(mites)
mites
# infer all maximum parsimony trees
trees <- bab(mites)
# For larger data sets you might use pratchet instead bab
# trees <- pratchet(mites, minit=200, trace=0, all=TRUE)
# build consensus tree
ctree <- root(consensus(trees, p=0.5), outgroup = "C._cymba",
              resolve.root=TRUE, edgelabel=TRUE)
plotBS(ctree, trees)
cnet <- consensusNet(trees)
plot(cnet)
```

`modelTest` *ModelTest*

Description

Comparison of different nucleotide or amino acid substitution models

Usage

```
modelTest(object, tree = NULL, model = NULL, G = TRUE, I = TRUE,
  FREQ = FALSE, k = 4, control = pml.control(), RHAS = "gamma", ...,
  mt_control = mt.control(crit = "BIC", n_model = 5, n_rhas = 100))

write.modelTest(x, file = "modelTest", digits = 10)
```

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 or "all" to test all available models.
`G`	logical, TRUE (default) if (discrete) Gamma model should be tested.
`I`	logical, TRUE (default) if invariant sites should be tested.
`FREQ`	logical, FALSE (default) if TRUE amino acid frequencies will be estimated.
`k`	number of rate classes. Can be a list with a vector for each RHAS term.

control	A list of parameters for controlling the fitting process.
RHAS	a character vector specifying the rate heterogeneity among sites models. Option are "gamma" for discrete gamma model with equal weights, "variable_gamma" for discrete gamma model with estimated weights, "free_rate" and "gamma_quadrature".
...	Further arguments passed to or from other methods.
mt_control	a list with some options.
x	an object of class modelTest.
file	a file name. File endings are added.
digits	default is 10, i.e. edge length for the bootstrap trees are exported. For digits larger smaller than zero no edge length are exported.

Details

modelTest estimates all the specified models for a given tree and data similar to (Posada and Crandall 1998; Posada 2008).

model="3Di" tests the three models for structural alignments mention in Garg and Hochberg (2025).

Value

A data.frame containing the log-likelihood, number of estimated parameters, AIC, AICc 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

Garg SG, Hochberg GKA (2025). "A General Substitution Matrix for Structural Phylogenetics." *Molecular Biology and Evolution*, **42**(6), msaf124. ISSN 1537-1719. doi:10.1093/molbev/msaf124.

Posada D (2008). "jModelTest: Phylogenetic Model Averaging." *Molecular Biology and Evolution*, **25**(7), 1253–1256. doi:10.1093/molbev/msn083.

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

Burnham, K. P. and Anderson, D. R (2002) *Model selection and multimodel inference: a practical information-theoretic approach*. 2nd ed. Springer, New York

Darriba D., Taboada G.L., Doallo R and Posada D. (2011) ProtTest 3: fast selection of best-fit models of protein evolution. . *Bioinformatics* **27**: 1164-1165

Puente-Lelievre, Caroline, et al. (2023) "Tertiary-interaction characters enable fast, model-based structural phylogenetics beyond the twilight zone." *bioRxiv*: 2023-12

See Also

pml, anova, AIC, codonTest, plan

Examples

```
## Not run:
data(Laurasiatherian)
mT <- modelTest(Laurasiatherian, model = c("JC", "K80", "HKY", "GTR"))

# Some exploratory data analysis
plot(mT$TL, mT$logLik, xlim=c(3,6.5),
   xlab="total tree length", ylab="log likelihood")
text(mT$TL, mT$logLik, labels=mT$Model, pos=4)

fit_GTR_G <- as.pml(mt, "GTR+G(4)")
fit_GTR_VG <- as.pml(mt, "GTR+VG(4)")
fit_GTR_R <- as.pml(mt, "GTR+R(4)")
plotRates(fit_GTR_G)
plotRates(fit_GTR_VWG, append=TRUE, col="red")
plotRates(fit_GTR_R, append=TRUE, col="green")

# extract best model
(best_model <- as.pml(mT))


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

# test all available amino acid models
plan(multisession, workers = 2)
(mTAA_all <- modelTest(chloroplast, model="all"))
plan(sequential)

## End(Not run)
```

`multiphyDat2pmlPart` *Partition model.*

Description

Model to estimate phylogenies for partitioned data.

Usage

```
multiphyDat2pmlPart(x, method = "unrooted", tip.dates = NULL, ...)

pmlPart2multiPhylo(x)

pmlPart(formula, object, control = pml.control(epsilon = 1e-08, maxit = 10),
  model = NULL, method = "unrooted", ...)
```

Arguments

x	an object of class pmlPart
method	One of "unrooted", "ultrametric" or "tiplabeled". Only unrooted is properly supported right now.
tip.dates	A named vector of sampling times associated to the tips/sequences. Leave empty if not estimating tip dated phylogenies.
...	Further arguments passed to or from other methods.
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.
model	A vector containing the models containing a model for each partition.

Details

The formula object allows to specify which parameter get optimized. The formula is generally of the form edge + bf + Q ~ rate + shape + ...{}, 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" is 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).

pmlPart2multiPhylo is a convenience function to extract the trees out of a pmlPart object.

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

## Not run:
sp2 <- pmlPart(~ edge + inv, fits, weight=weight)
sp2
AIC(sp2)

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

## End(Not run)
```

`neighborNet` *Computes a neighborNet from a distance matrix*

Description

Computes a neighborNet, i.e. an object of class `networx` from a distance matrix.

Usage

```
neighborNet(x, ord = NULL)
```

Arguments

`x`	a distance matrix.
`ord`	a circular ordering.

Details

`neighborNet` is still experimental. The cyclic ordering sometimes differ from the SplitsTree implementation, the *ord* argument can be used to enforce a certain circular ordering.

Value

`neighborNet` returns an object of class networx.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Bryant D, Huson DH (2023). "NeighborNet: improved algorithms and implementation." *Frontiers in bioinformatics*, **3**, 1178600. doi:10.3389/fbinf.2023.1178600.

Bryant D, Moulton V (2004). "Neighbor-Net: An Agglomerative Method for the Construction of Phylogenetic Networks." *Molecular Biology and Evolution*, **21**(2), 255-265. doi:10.1093/molbev/msh018.

See Also

splitsNetwork, consensusNet, plot.networx, lento, cophenetic.networx, distanceHadamard

Examples

```
data(yeast)
dm <- dist.ml(yeast)
nnet <- neighborNet(dm)
plot(nnet)
plot(nnet, type="outline")
```

NJ *Neighbor-Joining*

Description

This function performs the neighbor-joining tree estimation of Saitou and Nei (1987); Studier and Keppler (1988). UNJ is the unweighted version from Gascuel (1997).

Usage

```
NJ(x)

UNJ(x)
```

Arguments

x	A distance matrix.

Details

NJ is a wrapper around nj from ape.

Value

an object of class "phylo".

Author(s)

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

References

Gascuel O (1997). "Concerning the NJ algorithm and its unweighted version UNJ. 21 Pages 149-171 in: Mathematical hierarchies and biology. B. Mirkin, FR McMorris, 22 F. Roberts and A. Rzhetsky, eds." *American Mathematical Society, Providence, RI*, **23**.

Saitou N, Nei M (1987). "The Neighbor-Joining Method - a New Method for Reconstructing Phylogenetic Trees." *Molecular Biology and Evolution*, **4**(4), 406–425.

Studier JA, Keppler KJ (1988). "A Note on the Neighbor-Joining Algorithm of Saitou and Nei." *Molecular Biology and Evolution*, **5**(6), 729–731.

See Also

nj, dist.dna, dist.hamming, upgma, fastme

Examples

```
data(Laurasiatherian)
dm <- dist.ml(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)

rNNI(tree, moves = 1, n = length(moves))

rSPR(tree, moves = 1, n = length(moves), k = NULL)
```

Arguments

tree	A phylogenetic tree, object of class phylo.
moves	Number of tree rearrangements to be transformed on a tree. Can be a vector
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>

See Also

allTrees, SPR.dist

Examples

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

---

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, ...)

as.phyDat(x, ...)

## S3 method for class 'factor'
as.phyDat(x, ...)

## S3 method for class 'DNAbin'
as.phyDat(x, ...)

## S3 method for class 'AAbin'
as.phyDat(x, ...)
```

```
## S3 method for class 'alignment'
as.phyDat(x, type = "DNA", ...)

phyDat2alignment(x)

## S3 method for class 'MultipleAlignment'
as.phyDat(x, ...)

## S3 method for class 'AAStringSet'
as.phyDat(x, ...)

## S3 method for class 'DNAStringSet'
as.phyDat(x, ...)

as.StringSet(x, ...)

## S3 method for class 'phyDat'
as.StringSet(x, ...)

## S3 method for class 'phyDat'
as.MultipleAlignment(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'
as.AAbin(x, ...)

genlight2phyDat(x, ambiguity = NA)

acgt2ry(obj)

unalign(x)
```

Arguments

data	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.
...	further arguments passed to or from other methods.

x	An object containing sequences.
allLevels	return original data.
ambiguity	character for ambiguous character and no contrast is provided.
obj	as object of class phyDat

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. There is a more detailed example for specifying "USER" defined data formats in the vignette "phangorn-specials".

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

unalign converts a phyDat object of nucleotides or amino acids into a DNAbin or AAbin object in list form removing all gaps. These objects can be exported using write.FASTA.

Value

The functions return an object of class phyDat.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

DNAbin, as.DNAbin, baseFreq, glance.phyDat, read.dna, read.nexus.data and the chapter 1 in the vignette("phangorn-specials",package="phangorn") and the example of pmlMix for the use of allSitePattern

Examples

```
data(Laurasiatherian)
class(Laurasiatherian)
Laurasiatherian
# transform as characters
LauraChar <- as.character(Laurasiatherian)
# and back
Laura <- phyDat(LauraChar)
all.equal(Laurasiatherian, Laura)
LauraDNAbin <- as.DNAbin(Laurasiatherian)
all.equal(Laurasiatherian, as.phyDat(LauraDNAbin))
```

plot.networx *plot phylogenetic networks*

Description

So far not all parameters behave the same on the the rgl "3D" and basic graphic "2D" device.

Usage

```
## S3 method for class 'networx'
plot(x, type = "equal angle", use.edge.length = TRUE,
  show.tip.label = TRUE, show.edge.label = FALSE, edge.label = NULL,
  show.node.label = FALSE, node.label = NULL, show.nodes = FALSE,
  tip.color = "black", edge.color = "black", edge.width = 1,
  edge.lty = 1, split.color = NULL, split.width = NULL,
  split.lty = NULL, font = 3, cex = par("cex"), cex.node.label = cex,
  cex.edge.label = cex, col.node.label = tip.color,
  col.edge.label = tip.color, font.node.label = par("font"),
  font.edge.label = par("font"), underscore = FALSE, angle = 0,
  digits = 2, ...)
```

Arguments

x	an object of class "networx"
type	"3D" to plot using rgl or "equal angle", "2D" or "outline" in the normal device.
use.edge.length	a logical indicating whether to use the edge weights of the network to draw the branches (the default) or not.
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.
edge.label	an additional vector of edge labels (normally not needed).
show.node.label	a logical indicating whether to show the node labels (see example).
node.label	an additional vector of node labels (normally not needed).
show.nodes	a logical indicating whether to show the nodes (see example).
tip.color	the colors used for the tip labels.
edge.color	the colors used to draw edges.
edge.width	the width used to draw edges.
edge.lty	a vector of line types.
split.color	the colors used to draw edges.
split.width	the width used to draw edges.

split.lty	a vector of line types.
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.
cex.node.label	a numeric value giving the factor scaling of the node labels.
cex.edge.label	a numeric value giving the factor scaling of the edge labels.
col.node.label	the colors used for the node labels.
col.edge.label	the colors used for the edge labels.
font.node.label	the font used for the node labels.
font.edge.label	the font used for the edge labels.
underscore	a logical specifying whether the underscores in tip labels should be written as spaces (the default) or left as are (if TRUE).
angle	rotate the plot.
digits	if edge labels are numerical a positive integer indicating how many significant digits are to be used.
...	Further arguments passed to or from other methods.

Details

Often it is easier and safer to supply vectors of graphical parameters for splits (e.g. splits.color) than for edges. These overwrite values edge.color.

Value

plot.networx returns invisibly a list with paramters of the plot.

Note

The internal representation is likely to change.

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

Schliep, K., Potts, A. J., Morrison, D. A. and Grimm, G. W. (2017), Intertwining phylogenetic trees and networks. *Methods Ecol Evol*. **8**, 1212–1220. doi:10.1111/2041-210X.12760

Bagci, C., Bryant, D., Cetinkaya, B. and Huson, D.H. (2021), Microbial Phylogenetic Context Using Phylogenetic Outlines. *Genome Biology and Evolution*. Volume 13. Issue 9. evab213

See Also

consensusNet, neighborNet, splitsNetwork, hadamard, distanceHadamard, as.networx, evonet, as.phylo, densiTree, nodelabels

Examples

```
set.seed(1)
tree1 <- rtree(20, rooted=FALSE)
sp <- as.splits(rNNI(tree1, n=10))
net <- as.networx(sp)
plot(net)
plot(net, direction="horizontal")

circ_net <- allCircularSplits(5) |> as.networx()
plot(circ_net)
plot(circ_net, type = "outline")
## Not run:
# also see example in consensusNet
example(consensusNet)

## End(Not run)
```

---

plot.pml *Plot phylogeny of a pml object*

Description

plot.pml is a wrapper around plot.phylo with different default values for unrooted, ultrametric and tip dated phylogenies.

Usage

```
## S3 method for class 'pml'
plot(x, type = "phylogram", direction = "rightwards", ...,
  adj = NULL, digits = 2, method = "FBP")
```

Arguments

x	an object of class pml.
type	a character string specifying the type of phylogeny to be drawn; it must be one of "phylogram" (the default), "cladogram", "fan", "unrooted", "radial", "tidy", or any unambiguous abbreviation of these.
direction	a character string specifying the direction of the tree. Four values are possible: "rightwards" (the default), "leftwards", "upwards", and "downwards".
...	further parameters to be passed to plot.phylo.
adj	one or two numeric values specifying the horizontal and vertical justification of the text or symbols of the support values.

digits	integer indicating the number of decimal places.
method	either "FBP" the classical bootstrap (default), "TBE" (transfer bootstrap) or "MCC" for assigning clade credibilities.

Value

plot.pml returns the pml object x.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

plot.phylo, axisPhylo, add.scale.bar

Examples

```
fdir <- system.file("extdata/trees", package = "phangorn")
tmp <- read.csv(file.path(fdir,"H3N2_NA_20.csv"))
H3N2 <- read.phyDat(file.path(fdir,"H3N2_NA_20.fasta"), format="fasta")
dates <- setNames(tmp$numdate_given, tmp$name)

fit_td <- pml_bb(H3N2, model="JC", method="tipdated", tip.dates=dates,
                 rearrangement="none", control = pml.control(trace = 0))
plot(fit_td, show.tip.label = FALSE)
# Same as:
# root_time <- max(dates) - max(node.depth.edgelength(fit_td$tree))
# plot(fit_td$tree, show.tip.label = FALSE)
# axisPhylo(root.time = root_time, backward = FALSE)
plot(fit_td, show.tip.label = FALSE, direction="up")

fit_unrooted <- pml_bb(H3N2, model="JC", rearrangement="none",
                       control = pml.control(trace = 0))
plot(fit_unrooted, cex=0.5)
```

---

plotAnc *Plot ancestral character on a tree*

Description

plotAnc plots a phylogeny and adds character to the nodes. Either takes output from ancestral.pars or ancestral.pml or from an alignment where there are node labels in the tree match the constructed sequences in the alignment.

Usage

```
plotAnc(x, i = 1, type = "phylogram", ..., col = NULL, cex.pie = 0.5,
  pos = "bottomright", scheme = NULL)

plotSeqLogo(x, node = getRoot(x$tree), start = 1, end = 10,
  scheme = "Ape_NT", ...)

add_mutations(x, pos = NULL, frame = "none", ...)
```

Arguments

x	an object of class ancestral.
i	plots the i-th site.
type	a character string specifying the type of phylogeny to be drawn; it must be one of "phylogram" (the default), "cladogram", "fan", "unrooted", "radial", "tidy", or any unambiguous abbreviation of these.
...	Further arguments passed to or from other methods.
col	a vector containing the colors for all possible states.
cex.pie	a numeric defining the size of the pie graphs.
pos	position or range in the alignment to add mutations. If NULL all mutations are written out.
scheme	a predefined color scheme. For amino acid options are "Ape_AA", "Zappo_AA", "Clustal", "Polarity" and "Transmembrane_tendency", for nucleotides "Ape_NT" and"RY_NT". Names can be abbreviated.
node	the node for which the probabilities should be plotted.
start	start position to plot.
end	end position to plot.
frame	a character string specifying the kind of frame to be printed around the text. See edgelabels.

Details

For further details see vignette("Ancestral").

Value

plotAnc returns silently x.

plotSeqLogo returns a ggplot object.

add_mutations adds the position and changes of possible mutations to a phylogeny.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

anc_pml, heatmapp, plot.phylo, image.DNAbin, image.AAbin ggseqlogo, edgelabels

Examples

```
example(NJ)
# generate node labels to ensure plotting will work
tree <- makeNodeLabel(tree)
anc.p <- anc_pars(tree, Laurasiatherian)
# plot the third character
plotAnc(anc.p, 3, pos="topright")
plotSeqLogo(anc.p, node="Node10", 1, 25)

data(chloroplast)
tree <- pratchet(chloroplast,  maxit=10, trace=0)
tree <- makeNodeLabel(tree)
anc.ch <- anc_pars(tree, chloroplast)
image(as.phyDat(anc.ch)[, 1:25])
plotAnc(anc.ch, 21, scheme="Ape_AA", pos="topleft")
plotAnc(anc.ch, 21, scheme="Clustal", pos="topleft")
plotSeqLogo(anc.ch, node="Node1", 1, 25, scheme="Clustal")


data(woodmouse)
tree <- pml_bb(woodmouse, "JC", rearrangement="NNI")$tree |> midpoint()
woodmouse_aa <- trans(woodmouse, 2) |> as.phyDat()
anc_aa <- anc_pars(tree, woodmouse_aa)
plot(tree)
add_mutations(anc_aa, adj = c(0.5, -0.3), col=2)
```

---

plotBS *Plotting trees with bootstrap values*

Description

plotBS plots a phylogenetic tree with the bootstrap values assigned to the (internal) edges. It can also used to assign bootstrap values to a phylogenetic tree. add_support adds support values to a plot.

Usage

```
plotBS(tree, trees, type = "phylogram", method = "FBP", bs.col = "black",
  bs.adj = NULL, digits = 3, p = 0, frame = "none", tol = 1e-06,
  sep = "/", ...)

add_support(tree, trees, method = "FBP", tol = 1e-08, scale = TRUE,
  frame = "none", digits = 3, sep = "/", ...)
```

Arguments

tree	The tree on which edges the bootstrap values are plotted.
trees	a list of trees (object of class "multiPhylo").
type	the type of tree to plot, one of "phylogram", "cladogram", "fan", "unrooted", "radial" or "none". If type is "none" the tree is returned with the bootstrap values assigned to the node labels.
method	either "FBP" the classical bootstrap (default), "TBE" (transfer bootstrap) or "MCC" for assigning clade credibilities. In case of "MCC" all trees need to be rooted.
bs.col	color of bootstrap support labels.
bs.adj	one or two numeric values specifying the horizontal and vertical justification of the bootstrap labels.
digits	integer indicating the number of decimal places.
p	only plot support values higher than this percentage number (default is 0).
frame	a character string specifying the kind of frame to be printed around the bootstrap values. This must be one of "none" (the default), "rect" or "circle".
tol	a numeric value giving the tolerance to consider a branch length significantly greater than zero.
sep	seperator between the different methods.
...	further parameters used by plot.phylo.
scale	return ratio or percentage.

Details

The functions can either assign the classical Felsenstein's bootstrap proportions (FBP) (Felsenstein 1985; Penny and Hendy 1985; Penny and Hendy 1986), the transfer bootstrap expectation (TBE) of (Lemoine, Entfellner, Wilkinson, Correia, Felipe, De Oliveira, and Gascuel 2018) or "MCC" for assigning clade credibilities if trees are rooted. Using the option type=="n" just assigns the bootstrap values and return the tree without plotting it.

Value

plotBS returns silently a tree, i.e. an object of class phylo with the bootstrap values as node labels. The argument trees is optional and if not supplied the labels supplied in the node.label slot will be used.

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.

Lemoine F, Entfellner JD, Wilkinson E, Correia D, Felipe MD, De Oliveira T, Gascuel O (2018). "Renewing Felsenstein's phylogenetic bootstrap in the era of big data." *Nature*, **556**(7702), 452–456.

Penny D, Hendy M (1985). "Testing methods evolutionary tree construction." *Cladistics*, **1**, 266–278.

Penny D, Hendy M (1986). "Estimating the reliability of evolutionary trees." *Molecular Biology and Evolution*, **3**, 403–417.

See Also

plot.phylo, add_ci, nodelabels, prop.clades, maxCladeCred, transferBootstrap, consensus, consensusNet

Examples

```
fdir <- system.file("extdata/trees", package = "phangorn")
# RAxML best-known tree with bipartition support (from previous analysis)
raxml.tree <- read.tree(file.path(fdir,"RAxML_bipartitions.woodmouse"))
# RAxML bootstrap trees (from previous analysis)
raxml.bootstrap <- read.tree(file.path(fdir,"RAxML_bootstrap.woodmouse"))
par(mfrow=c(1,2))
plotBS(raxml.tree,  raxml.bootstrap, "p")
plotBS(raxml.tree,  raxml.bootstrap, "p", "TBE")
```

---

pml.control *Auxiliary for Controlling Fitting*

Description

Auxiliary functions for providing optim.pml, pml_bb fitting. Use it to construct a control or ratchet.par argument.

Usage

```
pml.control(epsilon = 1e-08, maxit = 10, trace = !getOption("quiet"),
  tau = 1e-08, statefreq = "empirical")

ratchet.control(iter = 20L, maxit = 200L, minit = 100L, prop = 1/2,
  rell = TRUE, bs = 100L)

mt.control(crit = "BIC", n_model = 100, n_rhas = 7)
```

Arguments

epsilon	Stop criterion for optimization (see details).
maxit	Maximum number of iterations (see details).
trace	Show output during optimization (see details).

tau	minimal edge length.
statefreq	take "empirical" or "estimate" state frequencies.
iter	Number of iterations to stop if there is no change.
minit	Minimum number of iterations.
prop	Only used if rearrangement=stochastic. How many NNI moves should be added to the tree in proportion of the number of taxa.´
rell	logical, if TRUE approximate bootstraping similar Minh et al. (2013) is performed.
bs	number of approximate bootstrap samples.
crit	criterion to choose for best models, one of "AIC", "AICc", "BIC".
n_model	how many of the best transition models to choose to compute the rate heterogeneity for.
n_rhas	number of rate heterogeneity models to compute for each transition model.

Details

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 not an absolute difference between, but instead is defined as (logLik(k)-logLik(k+1))/logLik(k+1). This seems to be a good compromise and to work reasonably well 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, so a higher of trace produces more feedback. It can be useful to figure out how long an run will take and for debugging.

statefreq controls if base/state frequencies are optimized or empirical estimates are taken, when this applies. For some nucleotide models (e.g. JC, SYM) equal base frequencies and for amino acid models precomputed state frequencies are used, if not '+F' is specified.

tau might be exactly zero if duplicated sequences in the alignment are observed. In this case the analysis is performed only on unique sequences and duplicated taxa are added to the tree with zero edge length. This may lead to multifurcations if there are three or more identical sequences. After optimization it is good practice to prune away edges of length tau using di2multi. See also Janzen et al. (2021).

Value

A list with components named as the arguments for controlling the fitting process.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Minh, B. Q., Nguyen, M. A. T., & von Haeseler, A. (2013). Ultrafast approximation for phylogenetic bootstrap. *Molecular biology and evolution*, **30(5)**, 1188-1195.

Janzen, T., Bokma, F.,Etienne, R. S. (2021) Nucleotide Substitutions during Speciation may Explain Substitution Rate Variation, *Systematic Biology*, **71(5)**, 1244–1254.

See Also

pml_bb, optim.pml

Examples

```
pml.control()
pml.control(maxit=25)
```

---

pml_bb *Likelihood of a tree.*

Description

pml_bb for pml black box infers a phylogenetic tree infers a tree using maximum likelihood (ML).

Usage

```
pml_bb(x, model = NULL, rearrangement = ifelse(method == "unrooted",
  "stochastic", "ratchet"), method = "unrooted", start = NULL,
  tip.dates = NULL, ...)
```

Arguments

x	An alignment of class (either class phyDat, DNAbin or AAbin) or an object of class modelTest.
model	A string providing model (e.g. "GTR+G(4)+I"). Not necessary if a modelTest object is supplied.
rearrangement	Type of tree tree rearrangements to perform, one of "none", "NNI", "stochastic" or "ratchet"
method	One of "unrooted", "ultrametric" or "tiplabeled".
start	A starting tree can be supplied.
tip.dates	A named vector of sampling times associated to the tips / sequences.
...	Further arguments passed to or from other methods.

Details

pml_bb is a convenience function combining pml and optim.pml. If no tree is supplied, the function will generate a starting tree. If a modelTest object is supplied the model will be chosen according to BIC.

tip.dates should be a named vector of sampling times, in any time unit, with time increasing toward the present. For example, this may be in units of “days since study start” or “years since 10,000 BCE”, but not “millions of years ago”.

model takes a string and tries to extract the model. When an modelTest object the best BIC model is chosen by default. The string should contain a substitution model (e.g. JC, GTR, WAG) and can additional have a term "+I" for invariant sites, "+G(4)" for a discrete gamma model, "+R(4)"

for a free rate model. In case of amino acid models a term "+F" for estimating the amino acid frequencies. Whether nucleotide frequencies are estimated is defined by pml.control.

Currently very experimental and likely to change.

Value

pml_bb returns an object of class pml.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

optim.pml, modelTest, rtt, pml.control

Examples

```
data(woodmouse)
tmp <- pml_bb(woodmouse, model="HKY+I", rearrangement="NNI")

## Not run:
data(Laurasiatherian)
mt <- modelTest(Laurasiatherian)
fit <- pml_bb(mt)

# estimate free rate model with 2 rate categories
fit_HKY_R2 <- pml_bb(woodmouse, model="HKY+R(2)")

## End(Not run)
```

---

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-08, maxit = 10), ...)
```

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 ~ rate + shape + ...{}, 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>

References

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

Lanfear, R., Calcott, B., Ho, S.Y.W. and Guindon, S. (2012) PartitionFinder: Combined Selection of Partitioning Schemes and Substitution Models for Phylogenetic Analyses. *Molecular Biology and Evolution*, **29(6)**, 1695-1701

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), ...)
```

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 edge + bf + Q ~ rate + shape + ...{}, 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

```
## Not run:
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$siteLik)
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))


data(Laurasiatherian)
dm <- dist.logDet(Laurasiatherian)
tree <- NJ(dm)
fit <- pml(tree, Laurasiatherian)
fit <- optim.pml(fit)

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

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


#
# simulation of mixture models
#
X <- allSitePattern(5)
tree0 <- read.tree(text = "((t1:0.3,t2:0.3):0.1,(t3:0.3,t4:0.3):0.1,t5:0.5);")
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$siteLik) + 1000*exp(fit2$siteLik)
attr(X, "weight") <- weights

fit0 <- pml(tree0, X)
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, 0.25)
logLik(fit.p)
AIC(fit.p, k=log(3000))

## End(Not run)
```

`print.phyDat` *Generic functions for class phyDat*

Description

These functions help to manipulate alignments of class phyDat.

Usage

```
## S3 method for class 'phyDat'
print(x, ...)

## S3 method for class 'phyDat'
cbind(..., gaps = "-", compress = TRUE)

## S3 method for class 'phyDat'
rbind(...)

## S3 method for class 'phyDat'
c(..., gaps = "-", compress = TRUE)

## S3 method for class 'phyDat'
subset(x, subset, select, site.pattern = TRUE, ...)

## S3 method for class 'phyDat'
```

```
x[i, j, ..., drop = FALSE]

## S3 method for class 'phyDat'
unique(x, incomparables = FALSE, identical = TRUE, ...)

removeUndeterminedSites(x, ...)

removeAmbiguousSites(x)

allSitePattern(n, levels = NULL, names = NULL, type = "DNA", code = 1)
```

Arguments

x	An object containing sequences.
...	further arguments passed to or from other methods.
gaps	character for gaps (default is '-').
compress	logical, compress data to store only site patterns.
subset	a subset of taxa.
select	a subset of characters.
site.pattern	select site pattern or sites (see details).
i, j	indices of the rows and/or columns to select or to drop. They may be numeric, logical, or character (in the same way than for standard R objects).
drop	for compatibility with the generic (unused).
incomparables	for compatibility with unique.
identical	if TRUE (default) sequences have to be identical, if FALSE sequences are considered duplicates if distance between sequences is zero (happens frequently with ambiguous sites).
n	Number of sequences.
levels	Level attributes.
names	Names of sequences.
type	Type of sequences ("DNA", "AA" or "USER").
code	The ncbi genetic code number for translation. By default the standard genetic code is used.

Details

allSitePattern generates all possible site patterns and can be useful in simulation studies. For further details see the vignette AdvancedFeatures.

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

phyDat stores identical columns of an alignment only once and keeps an index of the original positions. This saves memory and especially computations as these are usually need to be done only once for each site pattern. In the example below the matrix x in the example has 8 columns, but column 1 and 2 and also 3 and 5 are identical. The phyDat object y has only 6 site pattern. If argument site.pattern=FALSE the indexing behaves like on the original matrix x. site.pattern=TRUE can be useful inside functions.

Value

The functions return an object of class phyDat.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

DNAbin, as.DNAbin, baseFreq, glance.phyDat, dna2codon, read.dna, read.nexus.data and the chapter 1 in the vignette("AdvancedFeatures",package="phangorn") and the example of pmlMix for the use of allSitePattern.

Examples

```
data(Laurasiatherian)
class(Laurasiatherian)
Laurasiatherian
# base frequencies
baseFreq(Laurasiatherian)
# summary statistics
summary(Laurasiatherian)

# subsetting phyDat objects
# the first 5 sequences
subset(Laurasiatherian, subset=1:5)
# the first 5 characters
subset(Laurasiatherian, select=1:5, site.pattern = FALSE)
# subsetting with []
Laurasiatherian[1:5, 1:20]
# short for
subset(Laurasiatherian, subset=1:5, select=1:20, site.pattern = FALSE)
# the first 5 site patterns (often more than 5 characters)
subset(Laurasiatherian, select=1:5, site.pattern = TRUE)

x <- matrix(c("a", "a", "c", "g", "c", "t", "a", "g",
              "a", "a", "c", "g", "c", "t", "a", "g",
              "a", "a", "c", "c", "c", "t", "t", "g"), nrow=3, byrow = TRUE,
            dimnames = list(c("t1", "t2", "t3"), 1:8))
(y <- phyDat(x))

subset(y, 1:2)
subset(y, 1:2, compress=TRUE)

subset(y, select=1:3, site.pattern = FALSE) |> as.character()
subset(y, select=1:3, site.pattern = TRUE) |> as.character()
y[,1:3] # same as subset(y, select=1:3, site.pattern = FALSE)

# Compute all possible site patterns
# for nucleotides there $4 ^ (number of tips)$ patterns
allSitePattern(5)
```

read.nexus.partitions *Function to import partitioned data from nexus files*

Description

read.nexus.partitions reads in sequences in NEXUS format and splits the data according to the charsets given in the SETS block.

Usage

```
read.nexus.partitions(file, return = "list", ...)
```

Arguments

file	a file name.
return	either returns a list where each element is a 'phyDat' object or an object of class 'multiphyDat'
...	Further arguments passed to or from other methods.

Value

a list where each element is a 'phyDat' object or an object of class 'multiphyDat'.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

read.nexus.data, read.phyDat

Examples

```
tree <- rtree(10)
dat <- simSeq(tree, l=24)
fcat <- function(..., file = zz) cat(..., file=file, sep="", append=TRUE)
zz <- tempfile(pattern="file", tmpdir=tempdir(), fileext=".nex")
write.phyDat(dat, file=zz, format="nexus")
fcat("BEGIN SETS;\n")
fcat("  Charset codon1 = 1-12/3;\n")
fcat("  Charset codon2 = 2-12/3;\n")
fcat("  Charset codon3 = 3-12/3;\n")
fcat("  Charset range = 16-18;\n")
fcat("  Charset range2 = 13-15 19-21;\n")
fcat("  Charset singles = 22 23 24;\n")
fcat("END;\n")

tmp <- read.nexus.partitions(zz)
```

```
tmp
unlink(zz)
```

`read.nexus.splits` *Function to import and export splits and networks*

Description

`read.nexus.splits`, `write.nexus.splits`, `read.nexus.networx`, `write.nexus.networx` can be used to import and export splits and networks with nexus format and allow to exchange these object with other software like SplitsTree. `write.splits` returns a human readable output.

Usage

```
read.nexus.splits(file)

write.nexus.splits(obj, file = "", weights = NULL, taxa = TRUE,
  append = FALSE)

write.nexus.networx(obj, file = "", taxa = TRUE, splits = TRUE,
  append = FALSE)

read.nexus.networx(file, splits = TRUE)

write.splits(x, file = "", zero.print = ".", one.print = "|",
  print.labels = TRUE, ...)
```

Arguments

`file`	a file name.
`obj`	An object of class splits.
`weights`	Edge weights.
`taxa`	logical. If TRUE a taxa block is added
`append`	logical. If TRUE the nexus blocks will be added to a file.
`splits`	logical. If TRUE the nexus blocks will be added to a file.
`x`	An object of class splits.
`zero.print`	character which should be printed for zeros.
`one.print`	character which should be printed for ones.
`print.labels`	logical. If TRUE labels are printed.
`...`	Further arguments passed to or from other methods.
`labels`	names of taxa.

Value

write.nexus.splits and write.nexus.networx write out the splits and networx object to read with other software like SplitsTree. read.nexus.splits and read.nexus.networx return an splits and networx object.

Note

read.nexus.splits reads in the splits block of a nexus file. It assumes that different co-variables are tab delimited and the bipartition are separated with white-space. Comments in square brackets are ignored.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

See Also

prop.part, lento, as.splits, as.networx

Examples

```
(sp <- as.splits(rtree(5)))
write.nexus.splits(sp)
spl <- allCircularSplits(5)
plot(as.networx(spl))
write.splits(spl, print.labels = FALSE)
```

---

read.phyDat *Import and export sequence alignments*

Description

These functions read and write sequence alignments.

Usage

```
read.phyDat(file, format = "fasta", type = "DNA", ...)

write.phyDat(x, file, format = "fasta", colsep = "", nbcol = -1, ...)
```

Arguments

file	a file name specified by either a variable of mode character, or a double-quoted string.
format	File format of the sequence alignment (see details). Several popular formats are supported: "phylip", "interleaved", "sequential", "clustal", "fasta" or "nexus", or any unambiguous abbreviation of these.

type	Type of sequences ("DNA", "AA", "CODON" or "USER").
...	further arguments passed to or from other methods.
x	An object of class phyDat.
colsep	a character used to separate the columns (a single space by default).
nbcol	a numeric specifying the number of columns per row (-1 by default); may be negative implying that the nucleotides are printed on a single line.

Details

write.phyDat calls the function write.dna or write.nexus.data and read.phyDat calls the function read.dna or read.nexus.data, so 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

read.phyDat returns an object of class phyDat, write.phyDat write an alignment to a file.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Anonymous. FASTA format description. https://www.ncbi.nlm.nih.gov/blast/fasta.shtml
Felsenstein, J. (1993) Phylip (Phylogeny Inference Package) version 3.5c. Department of Genetics, University of Washington. https://phylipweb.github.io/phylip/

See Also

read.dna, read.GenBank, phyDat, read.alignment

Examples

```
fdir <- system.file("extdata/trees", package = "phangorn")
primates <- read.phyDat(file.path(fdir, "primates.dna"),
                        format = "interleaved")
```

SH.test *Shimodaira-Hasegawa Test*

Description

This function computes the Shimodaira–Hasegawa test Shimodaira and Hasegawa (1999) for a set of trees.

Usage

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

Arguments

...	either a series of objects of class "pml" separated by commas, a list containing such objects or an object of class "pmlPart" or a matrix containing the site-wise likelihoods in columns.
B	the number of bootstrap replicates.
data	an object of class "phyDat".
weight	if a matrix with site (log-)likelihoods is is supplied an optional vector containing the number of occurrences of each site pattern.

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, 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, SOWH.test

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)
# with pml objects as input
SH.test(fit1, fit2, B=1000)
# in real analysis use larger B, e.g. 10000

# with matrix as input
X <- matrix(c(fit1$siteLik, fit2$siteLik), ncol=2)
SH.test(X, weight=attr(Laurasiatherian, "weight"), B=1000)
## Not run:
example(pmlPart)
SH.test(sp, B=1000)

## End(Not run)
```

---

simSeq *Simulate sequences.*

Description

Simulate sequences from a given evolutionary tree or an pml object.

Usage

```
simSeq(x, ...)

## S3 method for class 'phylo'
simSeq(x, l = 1000, Q = NULL, bf = NULL,
  rootseq = NULL, type = "DNA", model = NULL, levels = NULL,
  rate = 1, ancestral = FALSE, code = 1, ...)

## S3 method for class 'pml'
simSeq(x, ancestral = FALSE, ...)
```

Arguments

x	a phylogenetic tree tree with edge lengths, i.e. an object of class phylo or and object of class pml.
...	Further arguments passed to or from other methods.
l	The length of the sequence to simulate.
Q	Either a numeric matrix of size Nstates × Nstates, giving the transition rates between states or a vector representing the lower triangular of these matrix (see details).
bf	Base frequencies.
rootseq	A vector of length l containing the root sequence. If not provided, the root sequence is randomly generated.
type	Type of sequences ("DNA", "AA", "CODON" or "USER").

model	Amino acid model of evolution to employ, for example "WAG", "JTT", "Dayhoff" or "LG" or structural 3Di alphabet (e.g. "Q_3Di"). For a full list of supported models, type phangorn:::.aa_3Di_models. Ignored if type is not equal to "AA".
levels	A character vector of the different character tokens. Ignored unless type = "USER".
rate	A numerical value greater than zero giving the mutation rate or scaler for edge lengths.
ancestral	Logical specifying whether to return ancestral sequences.
code	The ncbi genetic code number for translation (see details). By default the standard genetic code is used.

Details

simSeq is a generic function to simulate sequence alignments along a phylogeny. It is quite flexible and can generate DNA, RNA, amino acids, codon, morphological or binary sequences. simSeq can take as input a phylogenetic tree of class phylo, or a pml object; it will return an object of class phyDat. There is also a more low level version, which lacks rate variation, but one can combine different alignments with their own rates (see example). The rate parameter acts like a scaler for the edge lengths.

For codon models type="CODON", two additional arguments dnds for the dN/dS ratio and tstv for the transition transversion ratio can be supplied.

So far simSeq is limited to time reversible models. simSeq will normalize the rate matrix A composed from Q and bf so that every row of A sums to zero and expected rate is one, see formulas 13.14 and 13.15 on page 205 in Felsenstein (2004). The edge lengths are should represent the expected number of mutations per site.

Defaults:

If x is a tree of class phylo, then sequences will be generated with the default Jukes-Cantor DNA model ("JC").

If bf is not specified, then all states will be treated as equally probable.

If Q is not specified, then a uniform rate matrix will be employed.

Value

simSeq returns an object of class phyDat.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

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

See Also

phyDat, pml, SOWH.test

Examples

```
## Not run:
data(Laurasiatherian)
tree <- nj(dist.ml(Laurasiatherian))
fit <- pml(tree, Laurasiatherian, k=4)
fit <- optim.pml(fit, optNni=TRUE, model="GTR", optGamma=TRUE)
data <- simSeq(fit)

## End(Not run)


tree <- rtree(5)
plot(tree)
nodelabels()

# Example for simple DNA alignment
data <- simSeq(tree, l = 10, type="DNA", bf=c(0.1, 0.2, 0.3, 0.4), Q=1:6,
               ancestral=TRUE)
as.character(data)


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

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

SOWH.test *Swofford-Olsen-Waddell-Hillis Test*

Description

This function computes the Swofford–Olsen–Waddell–Hillis (SOWH) test Swofford, Olsen, Waddell, and Hillis (1996); Goldman, Anderson, and Rodrigo (2000), a parametric bootstrap test. The function is computational very demanding and likely to be very slow.

Usage

```
SOWH.test(x, n = 100, restricted = list(optNni = FALSE), optNni = TRUE,
  trace = !getOption("quiet"), ...)
```

Arguments

x	an object of class "pml".
n	the number of bootstrap replicates.
restricted	list of restricted parameter settings.
optNni	Logical value indicating whether topology gets optimized (NNI).
trace	Show output during computations.
...	Further arguments passed to "optim.pml".

Details

SOWH.test performs a parametric bootstrap test to compare two trees. It makes extensive use simSeq and optim.pml and can take quite long.

Value

an object of class SOWH. That is a list with three elements, one is a matrix containing for each bootstrap replicate the (log-) likelihood of the restricted and unrestricted estimate and two pml objects of the restricted and unrestricted model.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Goldman N, Anderson JP, Rodrigo AG (2000). "Likelihood-based tests of topologies in phylogenetics." *Systematic biology*, **49**(4), 652–670.

Swofford D, Olsen G, Waddell P, Hillis D (1996). "Phylogenetic Inference." In Hillis D, Moritz C, Mable B (eds.), *Molecular Systematics*, Second edition. Sinauer, Sunderland, MA.

See Also

pml, pmlPart, pmlCluster, simSeq, SH.test

Examples

```
# in real analysis use larger n, e.g. 500 preferably more
## Not run:
data(Laurasiatherian)
dm <- dist.logDet(Laurasiatherian)
tree <- NJ(dm)
fit <- pml(tree, Laurasiatherian)
fit <- optim.pml(fit, TRUE)
set.seed(6)
tree <- rNNI(fit$tree, 1)
fit <- update(fit, tree = tree)
(res <- SOWH.test(fit, n=100))
summary(res)
```

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

`splitsNetwork` *Phylogenetic Network*

Description

`splitsNetwork` estimates weights for a splits graph from a distance matrix.

Usage

```
splitsNetwork(splits, dm, gamma = 0.1, lambda = 1e-06, weight = NULL)
```

Arguments

`splits`	a splits object, containing all splits to consider.
`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 non-negative least-squares phylogenetic networks using L1 (LASSO), L2(ridge regression) constraints. The function minimizes the penalized least squares

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

with respect to

$$\|\beta\|_1 <= \gamma, \beta >= 0$$

where X is a design matrix constructed with `designSplits`. External edges are fitted without L1 or L2 constraints.

Value

`splitsNetwork` returns a splits object with a matrix added. 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

Efron, Hastie, Johnstone and Tibshirani (2004) Least Angle Regression (with discussion) *Annals of Statistics* **32(2)**, 407–499

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

See Also

`distanceHadamard`, `designTree` `consensusNet`, `plot.networx`

Examples

```
data(yeast)
dm <- dist.ml(yeast)
spl <- allSplits(8, names(yeast))
fit <- splitsNetwork(spl, dm)
net <- as.networx(fit)
plot(net)
write.nexus.splits(fit)
```

`superTree` *Super Tree methods*

Description

These function `superTree` allows the estimation of a supertree from a set of trees using either Matrix representation parsimony, Robinson-Foulds or SPR as criterion.

Usage

```
superTree(tree, method = "MRP", rooted = FALSE, trace = 0,
  start = NULL, multicore = FALSE, mc.cores = NULL, ...)
```

Arguments

`tree`	an object of class `multiPhylo`
`method`	An argument defining which algorithm is used to optimize the tree. Possible are "MRP", "RF", and "SPR".
`rooted`	should the resulting supertrees be rooted.
`trace`	defines how much information is printed during optimization.
`start`	a starting tree can be supplied.
`multicore`	logical, whether models should estimated in parallel.
`mc.cores`	The number of cores to use, i.e. at most how many child processes will be run simultaneously.
`...`	further arguments passed to or from other methods.

Details

The function superTree extends the function mrp.supertree from Liam Revells, with artificial adding an outgroup on the root of the trees. This allows to root the supertree afterwards. The functions is internally used in DensiTree. The implementation for the RF- and SPR-supertree are very basic so far and assume that all trees share the same set of taxa.

Value

The function returns an object of class phylo.

Author(s)

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

References

Baum, B. R., (1992) Combining trees as a way of combining data sets for phylogenetic inference, and the desirability of combining gene trees. *Taxon*, **41**, 3-10.

Ragan, M. A. (1992) Phylogenetic inference based on matrix representation of trees. *Molecular Phylogenetics and Evolution*, **1**, 53-58.

See Also

mrp.supertree, densiTree, RF.dist, SPR.dist

Examples

```
data(Laurasiatherian)
set.seed(1)
bs <- bootstrap.phyDat(Laurasiatherian,
                       FUN = function(x) upgma(dist.hamming(x)), bs=50)

mrp_st <- superTree(bs, minit = 25, maxit=50)
plot(mrp_st)

rf_st <- superTree(bs, method = "RF")
spr_st <- superTree(bs, method = "SPR")
```

---

terraces *Explore the likelihood / parsimony / distance surface*

Description

terraces visualizes in likelihood surface for the tree space (Sanderson, McMahon, and Steel 2011; Sanderson, McMahon, Stamatakis, Zwickl, and Steel 2015). Here trees usually are from a bootstrap or MCMC sample. There the first two axis are the principle components of distances between trees and the third axis is the likelihood value. We also allow parsimony score, minimum evolution criteria or least squares as criterion.

Usage

```
terraces(x, ...)

## S3 method for class 'pml'
terraces(x, trees = x$bs, dist_fun = "RF.dist",
  di2multi = FALSE, tol = 2e-08, plot = TRUE, ...)

## S3 method for class 'phyDat'
terraces(x, trees, dist_fun = "RF.dist", di2multi = FALSE,
  tol = 2e-08, plot = TRUE, ...)

## S3 method for class 'dist'
terraces(x, trees, dist_fun = "RF.dist", di2multi = FALSE,
  tol = 2e-08, plot = TRUE, crit = "ME", ...)
```

Arguments

x	an object of class pml
...	Further arguments passed to or from other methods.
trees	an object of class multiPhylo
dist_fun	a function to compute distances between trees see e.g. RF.dist
di2multi	logical, should polytomies get collapsed. Useful for Robinson-Foulds distance. If edge length are used to compute the distance, e.g. Kuhner-Felsenstein distance, this is not needed.
tol	a numeric value giving the tolerance to consider a branch length significantly greater than zero.
plot	logical if TRUE a 3D scatter is shown.
crit	either "ME" (minimum) or "RSS" (residual sum of squares) if x is a dist object.

Value

terraces silently returns a matrix.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Sanderson MJ, McMahon MM, Stamatakis A, Zwickl DJ, Steel M (2015). "Impacts of Terraces on Phylogenetic Inference." *Systematic Biology*, **64**(5), 709-726. ISSN 1063-5157. doi:10.1093/sysbio/syv024.

Sanderson MJ, McMahon MM, Steel M (2011). "Terraces in Phylogenetic Tree Space." *Science*, **333**(6041), 448-450. doi:10.1126/science.1206357.

See Also

pml_bb, optim.pml, pratchet,RF.dist, di2multi, cmdscale.

Examples

```
data(woodmouse)
trs <- pratchet(woodmouse, all=TRUE)
start_trs <- get("start_trees", envir = attr(trs, "env"))
terraces(as.phyDat(woodmouse), c(trs, start_trs))

## Not run:
fit <- pml_bb(woodmouse, model="JC")
terraces(fit, dist_fun="KF.dist")
terraces(fit, pkg="scatterplot3d")
terraces(fit, pkg="plot3D")
terraces(fit, pkg="rgl")

## End(Not run)
```

---

transferBootstrap *Transfer Bootstrap*

Description

transferBootstrap assigns transfer bootstrap (Lemoine, Entfellner, Wilkinson, Correia, Felipe, De Oliveira, and Gascuel 2018) values to the (internal) edges.

Usage

```
transferBootstrap(tree, trees, phylo = TRUE, scale = TRUE)
```

Arguments

tree	The tree on which edges the bootstrap values are plotted.
trees	a list of trees (object of class "multiPhylo").
phylo	Logical, return a phylogentic tree with support value or a vector of bootstrap values.
scale	scale the values.

Value

a phylogentic tree (a phylo object) with bootstrap values assigned to the node labels.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Lemoine F, Entfellner JD, Wilkinson E, Correia D, Felipe MD, De Oliveira T, Gascuel O (2018). "Renewing Felsenstein's phylogenetic bootstrap in the era of big data." *Nature*, **556**(7702), 452–456.

See Also

plotBS, maxCladeCred, drawSupportOnEdges

Examples

```
fdir <- system.file("extdata/trees", package = "phangorn")
# RAxML best-known tree with bipartition support (from previous analysis)
raxml.tree <- read.tree(file.path(fdir,"RAxML_bipartitions.woodmouse"))
# RAxML bootstrap trees (from previous analysis)
raxml.bootstrap <- read.tree(file.path(fdir,"RAxML_bootstrap.woodmouse"))

tree_tbe <- transferBootstrap(raxml.tree,  raxml.bootstrap)
par(mfrow=c(1,2))
plotBS(tree_tbe)
# same as
plotBS(raxml.tree,  raxml.bootstrap, "p", "TBE")
```

---

treedist *Distances between trees*

Description

`treedist` computes different tree distance methods and `RF.dist` the Robinson-Foulds or symmetric distance. The Robinson-Foulds distance (Robinson and Foulds 1979; Robinson and Foulds 1981) only depends on the topology of the trees. If edge weights should be considered `wRF.dist` calculates the weighted RF distance (Robinson and Foulds 1981) and `KF.dist` calculates the branch score distance (Kuhner and Felsenstein 1994). `path.dist` computes the path difference metric as described in Steel and Penny (1993). `sprdist` computes the approximate SPR distance (de Oliveira Martins, Leal, and Kishino 2008; De Oliveira Martins, Mallo, and Posada 2016).

Usage

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

sprdist(tree1, tree2)

SPR.dist(tree1, tree2 = NULL)

RF.dist(tree1, tree2 = NULL, normalize = FALSE, check.labels = TRUE,
  rooted = FALSE)

wRF.dist(tree1, tree2 = NULL, normalize = FALSE, check.labels = TRUE,
```

```
  rooted = FALSE)

KF.dist(tree1, tree2 = NULL, check.labels = TRUE, rooted = FALSE)

path.dist(tree1, tree2 = NULL, check.labels = TRUE, use.weight = FALSE)
```

Arguments

tree1	A phylogenetic tree (class phylo) or vector of trees (an object of class multiPhylo). See details
tree2	A phylogenetic tree.
check.labels	compares labels of the trees.
normalize	compute normalized RF-distance, see details.
rooted	take bipartitions for rooted trees into account, default is unrooting the trees.
use.weight	use edge.length argument or just count number of edges on the path (default)

Details

The Robinson-Foulds distance between two trees T_1 and T_2 with n tips is defined as (following the notation Steel and Penny 1993):

$$d(T_1, T_2) = i(T_1) + i(T_2) - 2v_s(T_1, T_2)$$

where $i(T_1)$ denotes the number of internal edges and $v_s(T_1, T_2)$ denotes the number of internal splits shared by the two trees. The normalized Robinson-Foulds distance is derived by dividing $d(T_1, T_2)$ by the maximal possible distance $i(T_1) + i(T_2)$. If both trees are unrooted and binary this value is $2n - 6$.

Functions like RF.dist returns the Robinson-Foulds distance between either 2 trees or computes a matrix of all pairwise distances if a multiPhylo object is given.

For large number of trees the distance functions can use a lot of memory!

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>, Leonardo de Oliveira Martins

References

de Oliveira Martins L, Leal É, Kishino H (2008). "Phylogenetic Detection of Recombination with a Bayesian Prior on the Distance between Trees." *PLoS ONE*, **3**(7), 1-13. doi:10.1371/journal.pone.0002651.

De Oliveira Martins L, Mallo D, Posada D (2016). "A Bayesian Supertree Model for Genome-Wide Species Tree Reconstruction." *Systematic Biology*, **65**(3), 397-416. doi:10.1093/sysbio/syu082.

Kuhner MK, Felsenstein J (1994). "A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates." *Molecular Biology and Evolution*, **11**(3), 459-468. doi:10.1093/oxfordjournals.molbev.a040126.

Robinson D, Foulds L (1981). "Comparison of phylogenetic trees." *Mathematical Biosciences*, **53**(1), 131 - 147. ISSN 0025-5564. doi:10.1016/00255564(81)900432.

Robinson DF, Foulds LR (1979). "Comparison of weighted labelled trees." In Horadam AF, Wallis WD (eds.), *Combinatorial Mathematics VI: Proceedings of the Sixth Australian Conference on Combinatorial Mathematics, Armidale, Australia, August 1978*, 119–126. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-34857-3. doi:10.1007/BFb0102690.

Steel MA, Penny D (1993). "Distributions of Tree Comparison Metrics—Some New Results." *Systematic Biology*, **42**(2), 126-141. ISSN 1063-5157. doi:10.1093/sysbio/42.2.126.

See Also

`dist.topo`, `nni`, `superTree`, `mast`

Examples

```
tree1 <- rtree(100, rooted=FALSE)
tree2 <- rSPR(tree1, 3)
RF.dist(tree1, tree2)
treedist(tree1, tree2)
sprdist(tree1, tree2)
trees <- rSPR(tree1, 1:5)
SPR.dist(tree1, trees)
```

---

upgma *UPGMA, WPGMA and sUPGMA*

Description

UPGMA and WPGMA clustering. UPGMA (Sokal and Michener 1958) and WPGMA (McQuitty 1966) are a wrapper function around `hclust` returning a `phylo` object. `supgma` perform serial sampled UPGMA similar to Drummond and Rodrigo (2000).

Usage

```
upgma(D, ...)

wpgma(D, ...)

supgma(D, tip.dates, trace = 0, ...)
```

Arguments

D	A distance matrix, i.e. an object of class dist. If an matrix is supplied it is tried to covert it do a dist object.
...	Further arguments passed to or from other methods.
tip.dates	A named vector of sampling times associated to the tips.
trace	Show output during optimization (see details).

Details

UPGMA and WPGMA return ultrametric trees, it is implicitly assumed that the distances supplied are close to ultrametric, e.g. hold the molecular clock assumption. Neighbor Joining (NJ) nj and fastME fastme relax this assumption to additive distances. sUPGMA assumes tip dated data.

tip.dates should be a vector of sampling times, in any time unit, with time increasing toward the present. For example, this may be in units of "days since study start" or "years since 10.000 BCE", but not "millions of years ago".

Value

A phylogenetic tree of class phylo.

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Drummond A, Rodrigo AG (2000). "Reconstructing Genealogies of Serial Samples Under the Assumption of a Molecular Clock Using Serial-Sample UPGMA." *Molecular Biology and Evolution*, **17**(12), 1807-1815. ISSN 0737-4038. doi:10.1093/oxfordjournals.molbev.a026281.

McQuitty LL (1966). "Similarity analysis by reciprocal pairs for discrete and continuous data." *Educational and Psychological measurement*, **26**(4), 825–831.

Sokal R, Michener C (1958). "A statistical method for evaluating systematic relationships." *University of Kansas Science Bulletin*, **38**, 1409–1438.

Sneath, P. H., & Sokal, R. R. (1973). *Numerical taxonomy. The principles and practice of numerical classification.*

See Also

hclust, dist.hamming, NJ, as.phylo, fastme, nnls.tree, rtt

Examples

```
data(Laurasiatherian)
dm <- dist.ml(Laurasiatherian)
tree <- upgma(dm)
plot(tree)
```

`write.ancestral` *Export and convenience functions for ancestral reconstructions*

Description

`write.ancestral` allows to export ancestral reconstructions. It writes out the tree, a tab delimited text file with the probabilities, an alignment of the the extant sequences and the most likely ancestral states. And a unaligned fasta file containg the most likely ancestral states. `ancestral` generates an object of class ancestral.

Usage

```
write.ancestral(x, file = "ancestral")

as.ancestral(tree, prob, align, ...)

## S3 method for class 'ancestral'
print(x, ...)

## S3 method for class 'ancestral'
as.phyDat(x, ...)

## S3 method for class 'ancestral'
as.data.frame(x, ..., terminal = FALSE)
```

Arguments

`x`	an object of class ancestral
`file`	a file name. File endings are added.
`tree`	an object of class phylo.
`prob`	an data.frame containing a matrix of posterior probabilities for each state and site.
`align`	an object of class phyDat.
`...`	Further arguments passed to or from other methods.
`terminal`	logical, should terminal nodes included.

Details

This allows also to read in reconstruction made by iqtree to use the plotting capabilities of R.

Value

`write.ancestral` returns the input x invisibly.

See Also

anc_pml, plotAnc

Examples

```
data(Laurasiatherian)
fit <- pml_bb(Laurasiatherian[,1:100], "JC", rearrangement = "none")
anc_ml <- anc_pml(fit)
write.ancestral(anc_ml)
# Can be also results from iqtree
align <- read.phyDat("ancestral_all.fasta", format="fasta")
tree <- read.tree("ancestral_tree.nwk")
df <- read.table("ancestral_state.tsv.gz", header=TRUE)
anc_ml_disc <- as.ancestral(tree, df, align)

plotAnc(anc_ml_disc, 20)

heatmapp(tree, align, nodes=TRUE)

unlink(c("ancestral_all.fasta", "ancestral_tree.nwk",
         "ancestral_state.tsv.gz", "ancestral_state.fasta"))
```

---

write.pml *Export pml objects*

Description

`write.pml` writes out the ML tree and the model parameters.

Usage

```
write.pml(x, file = "pml", save_rds = TRUE, digits = 10, ...)
```

Arguments

x	an object of class pml.
file	a file name. File endings are added.
save_rds	logical, if TRUE saves the pml object as a rds file, otherwise the alignment is saved as a fasta file.
digits	default is 10, i.e. edge length for the bootstrap trees are exported. For digits larger smaller than zero no edge length are exported.
...	Further arguments passed to or from other methods.

Details

write.pml creates several files. It exports the alignment as fasta file. It writes out the ML tree in a newick file and the estimates parameters in a txt file. It should be possible to (re-)create the pml object up to numerical inaccuracies and this is possible with the *.rds file. If bootstrap trees exist these are additionally exported in a compressed nexus file. Additionally several plots are returned. The maximum likelihood tree, with support values, if these are available. If an bootstrapped trees exist, a consensus tree, a consensus network (< 200 tips) and terrace plot. And last but not least the distribution of the rates. It might be better to adopt these on the dataset.

Value

write.pml returns the input x invisibly.

See Also

ancestral.pml, plotAnc

Examples

```
data(woodmouse)
fit <- pml_bb(woodmouse, "JC", rearrangement = "none")
write.pml(fit, "woodmouse")
unlink(c("woodmouse.txt", "woodmouse_tree.nwk", "woodmouse_align.fasta",
       "woodmouse_tree.pdf", "woodmouse.rds", "woodmouse_rates.pdf"))
```

---

writeDist *Writing and reading distances in phylip and nexus format*

Description

readDist, writeDist and write.nexus.dist are useful to exchange distance matrices with other phylogenetic programs.

Usage

```
writeDist(x, file = "", format = "phylip", ...)

write.nexus.dist(x, file = "", append = FALSE, upper = FALSE,
  diag = TRUE, digits = getOption("digits"), taxa = !append)

readDist(file, format = "phylip")

read.nexus.dist(file)

## S3 method for class 'dist'
unique(x, incomparables, ...)
```

Arguments

x	A dist object.
file	A file name.
format	file format, default is "phylip", only other option so far is "nexus".
...	Further arguments passed to or from other methods.
append	logical. If TRUE the nexus blocks will be added to a file.
upper	logical value indicating whether the upper triangle of the distance matrix should be printed.
diag	logical value indicating whether the diagonal of the distance matrix should be printed.
digits	passed to format inside of write.nexus.dist.
taxa	logical. If TRUE a taxa block is added.
incomparables	Not used so far.

Value

an object of class dist

Author(s)

Klaus Schliep <klaus.schliep@gmail.com>

References

Maddison, D. R., Swofford, D. L. and Maddison, W. P. (1997) NEXUS: an extensible file format for systematic information. *Systematic Biology*, **46**, 590–621.

See Also

To compute distance matrices see dist.ml dist.dna and dist.p for pairwise polymorphism p-distances

Examples

```
data(yeast)
dm <- dist.ml(yeast)
writeDist(dm)
write.nexus.dist(dm)
```

yeast *Yeast alignment (Rokas et al.)*

Description

Alignment of 106 genes of 8 different species of 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

∗ **IO**
read.nexus.partitions, 91
read.nexus.splits, 92
read.phyDat, 93
∗ **aplot**
add_ci, 5
add_edge_length, 7
∗ **classif**
ldfactorial, 57
treedist, 105
∗ **cluster**
add.tips, 4
addConfidences, 8
allSplits, 9
allTrees, 11
ancestral.pml, 12
as.pml, 15
bab, 20
bootstrap.pml, 22
coalSpeciesTree, 27
codonTest, 28
delta.score, 31
designTree, 35
dist.hamming, 38
dist.p, 40
distanceHadamard, 41
dna2codon, 42
fitch, 44
gap_as_state, 46
getClans, 47
getRoot, 50
hadamard, 51
latag2n.phyDat, 56
lento, 58
lli, 59
mast, 61
maxCladeCred, 62
modelTest, 64
multiphyDat2pmlPart, 66
NJ, 69
nni, 70
phyDat, 71
pml_bb, 83
pmlCluster, 84
pmlMix, 86
print.phyDat, 88
simSeq, 96
splitsNetwork, 100
superTree, 101
upgma, 107
writeDist, 111
∗ **datasets**
chloroplast, 24
Laurasiatherian, 57
mites, 63
yeast, 113
∗ **distribution**
discrete.gamma, 36
∗ **dplot**
as.networx, 14
∗ **hplot**
cladePar, 26
consensusNet, 29
densiTree, 32
discrete.gamma, 36
heatmapp, 53
neighborNet, 68
plot.networx, 74
plot.pml, 76
plotAnc, 77
∗ **manip**
cophenetic.networx, 31
gap_as_state, 46
∗ **models**
SH.test, 95
SOWH.test, 98
∗ **plot**
lento, 58

∗ **univar**
getClans, 47
[.phyDat *(*print.phyDat*)*, 88

acctran, 3, *45*
ace, *13*
acgt2ry *(*phyDat*)*, 71
add.scale.bar, *77*
add.tips, 4
add_boxplot, *7*, *63*
add_boxplot *(*add_ci*)*, 5
add_ci, 5, *81*
add_edge_length, *6*, 7, *62, 63*
add_mutations *(*plotAnc*)*, 77
add_support *(*plotBS*)*, 79
addConfidences, 8
addTrivialSplits *(*allSplits*)*, 9
AIC, *29*, *65*
AICc *(*modelTest*)*, 64
allCircularSplits *(*allSplits*)*, 9
allCircularSplitsST *(*allSplits*)*, 9
allCompat *(*maxCladeCred*)*, 62
allSitePattern, *90*
allSitePattern *(*print.phyDat*)*, 88
allSplits, 9
allTrees, 11, *71*
anc_pars *(*ancestral.pml*)*, 12
anc_pml, *54*, *79*, *110*
anc_pml *(*ancestral.pml*)*, 12
ancestral.pars, *25*, *45*
ancestral.pars *(*ancestral.pml*)*, 12
ancestral.pml, 12, *19*, *47*, *56*, *111*
anova, *65*
anova.pml *(*as.pml*)*, 15
as.AAbin.phyDat *(*phyDat*)*, 71
as.ancestral *(*write.ancestral*)*, 109
as.bitsplits.splits *(*allSplits*)*, 9
as.character.phyDat *(*phyDat*)*, 71
as.data.frame.ancestral *(*write.ancestral*)*, 109
as.data.frame.phyDat *(*phyDat*)*, 71
as.DNAbin, *73*, *90*
as.DNAbin.phyDat *(*phyDat*)*, 71
as.Matrix *(*allSplits*)*, 9
as.Matrix.splits *(*allSplits*)*, 9
as.matrix.splits *(*allSplits*)*, 9
as.MultipleAlignment *(*phyDat*)*, 71
as.networx, *8*, *11*, 14, *76*, *93*
as.phyDat *(*phyDat*)*, 71
as.phyDat.ancestral *(*write.ancestral*)*, 109
as.phylo, *15*, *76*, *108*
as.phylo.splits *(*allSplits*)*, 9
as.pml, 15
as.prop.part.splits *(*allSplits*)*, 9
as.splits, *8*, *59*, *93*
as.splits *(*allSplits*)*, 9
as.StringSet *(*phyDat*)*, 71
axisPhylo, *34*, *77*

bab, 20, *25*, *44, 45*
base.freq, *22*
baseFreq, 21, *73*, *90*
bind.tree, *5*
bootstrap.phyDat *(*bootstrap.pml*)*, 22
bootstrap.pml, *19*, 22, *45*, *63*
BranchAndBound *(*bab*)*, 20
bxp, *6*

c.phyDat *(*print.phyDat*)*, 88
c.splits *(*allSplits*)*, 9
cbind.phyDat *(*print.phyDat*)*, 88
chloroplast, 24
CI, 25, *45*, *49*
cladePar, 26
cmdscale, *104*
coalSpeciesTree, 27
codon2dna *(*dna2codon*)*, 42
codonTest, 28, *65*
compatible *(*allSplits*)*, 9
composition_test *(*baseFreq*)*, 21
consensus, *7*, *51*, *63*, *81*
consensusNet, *15*, *23*, 29, *63*, *69*, *76*, *81*, *101*
cophenetic, *31*
cophenetic.networx, 31, *69*
cophenetic.splits *(*cophenetic.networx*)*, 31
createLabel *(*addConfidences*)*, 8

delta.score, 31
densiTree, 32, *76*, *102*
designSplits *(*designTree*)*, 35
designTree, 35, *101*
dfactorial, *11*, *21*
dfactorial *(*ldfactorial*)*, 57
di2multi, *104*
discrete.beta *(*discrete.gamma*)*, 36
discrete.gamma, 36

dist.dna, *39*, *41*, *70*, *112*
dist.hamming, *32*, 38, *41*, *70*, *108*
dist.logDet *(*dist.hamming*)*, 38
dist.ml, *112*
dist.ml *(*dist.hamming*)*, 38
dist.p, *39*, 40, *112*
dist.topo, *107*
distanceHadamard, *11*, *15*, *30*, *36*, 41, *52*, *69*, *76*, *101*
distinct.splits *(*allSplits*)*, 9
diversity *(*getClans*)*, 47
dna2aa *(*dna2codon*)*, 42
dna2codon, 42, *90*
DNAbin, *73*, *90*
drawSupportOnEdges, *105*

edgelabels, *78, 79*
evonet, *15*, *76*

factorial, *58*
fastme, *36*, *70*, *108*
fhm *(*hadamard*)*, 51
fitch, *25*, 44

gap_as_ambiguous *(*gap_as_state*)*, 46
gap_as_state, *13*, *39*, 46, *47*, *56*
genlight2phyDat *(*phyDat*)*, 71
getClans, *45*, 47
getClips *(*getClans*)*, 47
getDiversity *(*getClans*)*, 47
getRoot, 50
getSlices *(*getClans*)*, 47
ggseqlogo, *79*
glance, *22*
glance.phyDat, *73*, *90*
glance.phyDat *(*baseFreq*)*, 21
glance.pml *(*as.pml*)*, 15

h2st *(*hadamard*)*, 51
h4st *(*hadamard*)*, 51
hadamard, *15*, *42*, 51, *59*, *76*
has_gap_state *(*gap_as_state*)*, 46
hclust, *107, 108*
heatmapp, *13*, 53, *79*
howmanytrees, *11*, *58*

identify, *55*
identify.networx, 54
image.AAbin, *54*, *56*, *79*
image.ancestral *(*image.phyDat*)*, 55
image.DNAbin, *54*, *56*, *79*
image.phyDat, 55

jitter, *34*

keep_as_tip *(*getRoot*)*, 50
KF.dist *(*treedist*)*, 105

latag2n, *13*, *47*, *56*
latag2n.phyDat, *13*, *47*, 56
Laurasiatherian, 57
ldfactorial, 57
legend, *6*
lento, *11*, *30*, *42*, *52*, 58, *69*, *93*
lli, 59
logLik.pml *(*as.pml*)*, 15

makeNodeLabel, *13*
mast, 61, *107*
matchSplits *(*allSplits*)*, 9
maxCladeCred, *6, 7*, *23*, *30*, 62, *81*, *105*
mcc *(*maxCladeCred*)*, 62
midpoint *(*getRoot*)*, 50
mites, 63
modelTest, *19*, *29*, 64, *84*
mt.control *(*pml.control*)*, 81
multi2di, *51*
multiphyDat2pmlPart, 66

neighborNet, *15*, *30*, *42*, 68, *76*
networx *(*as.networx*)*, 14
NJ, *45*, 69, *108*
nj, *70*, *108*
nni, *11*, *45*, 70, *107*
nnls.networx *(*designTree*)*, 35
nnls.phylo *(*designTree*)*, 35
nnls.splits *(*designTree*)*, 35
nnls.tree, *108*
nnls.tree *(*designTree*)*, 35
node.depth, *7*
nodelabels, *23*, *76*, *81*

optim.parsimony *(*fitch*)*, 44
optim.pml, *23*, *81*, *83, 84*, *104*
optim.pml *(*as.pml*)*, 15

pace *(*ancestral.pml*)*, 12
parsimony, *13*, *25*, *49*
parsimony *(*fitch*)*, 44

parsimony_edgelength (acctran), 3
path.dist (treedist), 105
pbeta, *37*
pgamma, *37*
phyDat, *22*, *44*, *47*, *49*, 71, *94*, *97*
phyDat2alignment (phyDat), 71
phyDat2MultipleAlignment (phyDat), 71
plan, *23*, *65*
plot.networx, *15*, *30*, *34*, *42*, *52*, *55*, *69*, 74, *101*
plot.phylo, *6*, *8*, *19*, *23*, *26*, *34*, *54*, *77*, *79*, *81*
plot.pml, 76
plot_gamma_plus_inv (discrete.gamma), 36
plotAnc, *13*, *54*, 77, *110*, *111*
plotBS, *6*, *63*, 79, *105*
plotRates (discrete.gamma), 36
plotSeqLogo (plotAnc), 77
pml, *13*, *23*, *29*, *45*, *60*, *65*, *67*, *85*, *87*, *95*, *97*, *99*
pml (as.pml), 15
pml.control, *19*, 81, *84*
pml.fit, *37*
pml.fit (lli), 59
pml_bb, *15*, *19*, *60*, *81*, 83, *83*, *104*
pmlCluster, *67*, 84, *87*, *95*, *99*
pmlMix, *19*, *29*, *60*, *67*, *73*, *85*, 86, *90*
pmlPart, *19*, *60*, *85*, *87*, *95*, *99*
pmlPart (multiphyDat2pmlPart), 66
pmlPart2multiPhylo (multiphyDat2pmlPart), 66
pratchet, *4*, *21*, *25*, *104*
pratchet (fitch), 44
presenceAbsence (addConfidences), 8
print.ancestral (write.ancestral), 109
print.phyDat, 88
print.pml (as.pml), 15
print.splits (allSplits), 9
print.summary.phyDat (baseFreq), 21
prop.clades, *81*
prop.part, *11*, *63*, *93*
pruneTree (getRoot), 50

random.addition (fitch), 44
ratchet.control (pml.control), 81
rbind.phyDat (print.phyDat), 88
read.alignment, *94*
read.dna, *73*, *90*, *94*
read.GenBank, *94*
read.nexus.data, *73*, *90*, *91*, *94*
read.nexus.dist (writeDist), 111
read.nexus.networx (read.nexus.splits), 92
read.nexus.partitions, 91
read.nexus.splits, *11*, 92
read.phyDat, *91*, 93
readDist (writeDist), 111
removeAmbiguousSites (print.phyDat), 88
removeTrivialSplits (allSplits), 9
removeUndeterminedSites (print.phyDat), 88
RF.dist, *8*, *102–104*
RF.dist (treedist), 105
RI, *45*, *49*
RI (CI), 25
rNNI (nni), 70
root, *13*, *51*
rSPR (nni), 70
rtree, *11*
rtt, *34*, *36*, *84*, *108*
rug, *37*

sankoff, *25*
sankoff (fitch), 44
SH.test, *19*, *67*, *85*, 95, *99*
simSeq, 96, *99*
SOWH.test, *23*, *95*, *97*, 98
speciesTree, *27*
splits (allSplits), 9
splitsNetwork, *15*, *30*, *36*, *69*, *76*, 100
SPR.dist, *62*, *71*, *102*
SPR.dist (treedist), 105
sprdist (treedist), 105
stepfun, *37*
subset.phyDat (print.phyDat), 88
summary.clanistics (getClans), 47
summary.phyDat (baseFreq), 21
superTree, 101, *107*
supgma (upgma), 107

terraces, 102
trans, *42*, *44*
transferBootstrap, *63*, *81*, 104
treedist, 105

unalign (phyDat), 71
unique.dist (writeDist), 111
unique.phyDat (print.phyDat), 88
unique.splits (allSplits), 9

UNJ *(NJ)*, 69
update.pml *(as.pml)*, 15
upgma, *36*, *70*, 107

vcov.pml *(as.pml)*, 15

wpgma *(upgma)*, 107
wRF.dist *(treedist)*, 105
write.ancestral, 109
write.dna, *94*
write.FASTA, *73*
write.modelTest *(modelTest)*, 64
write.nexus.data, *94*
write.nexus.dist *(writeDist)*, 111
write.nexus.networx
(read.nexus.splits), 92
write.nexus.splits *(read.nexus.splits)*, 92
write.phyDat *(read.phyDat)*, 93
write.pml, 110
write.splits *(read.nexus.splits)*, 92
writeDist, *39*, 111

yeast, 113