Package ‘ips’

July 8, 2026

Version 0.1.1

Title Interfaces to Phylogenetic Software in R

URL https://github.com/heibl/ips

Depends R (>= 3.5.0), ape

Imports data.table, dplyr, phangorn, plyr, rlang, tibble, XML

Description Functions that wrap popular phylogenetic software for sequence alignment, masking of sequence alignments, and estimation of phylogenies and ancestral character states.

License GPL-3

Language en-US

Encoding UTF-8

NeedsCompilation no

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

Author Christoph Heibl [aut] (ORCID: <https://orcid.org/0000-0002-7655-3299>),
Natalie Cusimano [aut],
Franz-Sebastian Krah [aut],
Martin R. Smith [ctb, cre] (ORCID:
<https://orcid.org/0000-0001-5660-1727>)

Maintainer Martin R. Smith <martin.smith@durham.ac.uk>

Repository CRAN

Date/Publication 2026-07-08 09:40:02 UTC

Contents

ips-package 3
aliscore 4
aperm.DNAbin 5
blastn 6
code.simple.gaps 6

collapseUnsupportedEdges . . . 7
combMyTree . . . 8
del.miss . . . 9
deleteGaps . . . 9
descendants . . . 10
DNAbin2index . . . 11
EmptyCells . . . 12
eoi . . . 13
fixNodes . . . 15
forceEqualTipHeights . . . 16
gblocks . . . 17
index2DNAbin . . . 19
ips.16S . . . 19
ips.28S . . . 20
ips.cox1 . . . 21
ips.tree . . . 21
mafft . . . 22
mafft.merge . . . 24
mrbayes . . . 25
mrbayes.lset . . . 26
mrbayes.mcmc . . . 28
mrbayes.prset . . . 30
multistate . . . 32
neighboringPairs . . . 34
ntip . . . 34
partitionfinder . . . 35
pathd8 . . . 36
phylo2mafft . . . 37
phylo2mst . . . 38
pis . . . 38
prank . . . 39
raxml . . . 40
raxml.partitions . . . 43
rbeauti . . . 44
rc . . . 46
read . . . 47
read.beast . . . 48
read.beast.table . . . 49
sister . . . 50
splitIntoClades . . . 51
terminalSisters . . . 51
tipHeights . . . 52
traitRate . . . 52
trimEnds . . . 54
unlistFirstLevel . . . 55
write.fas . . . 55

Index **58**

ips-package *Interfaces to Phylogenetic Software*

Description

This package presents a set of functions that were formerly included in the *phyloch* package and which wrap popular phylogenetic software for sequence alignment, masking of sequence alignments, and estimation of phylogenies and ancestral character states.

Details

Package:	ips
Type:	Package
Version:	0.0.12
Date:	2019-11-14
License:	GPL (>= 2)

There are several functions for reading and writing DNA sequences in FASTA, PHYLIP, and NEXUS format: `read.fas`, `read.phy`, `read.nex`, `write.fas`, `write.phy`, and `write.nex`. Some functions are available for integrating BEAST with R. XML input files for BEAST can be generated with `rbeauti`. Two functions are designed to read TreeAnnotator output: `read.beast` will render an object of class `phylo` with additional node statistics appended as list elements. These additional node statistics will be lost be the subsequent use of `ladderize` or `rotate` (or similar functions that change the ordering of internal nodes).`read.beast.table` also parses the TreeAnnotator output, but returns a matrix of node statistics. This package itself does not implement techniques for phylogenetic analyses, but provides a series of wrappers for commonly used software packages. Sequence alignment can be done with the `mafft` and `prank`; cleaning of sequences with `gblocks` and `aliscore`. The function `raxml` and `mrbayes` are intended for phylogenetic tree search. Running `mrbayes` with argument `run = FALSE` can be used to create MrBayes-executable NEXUS files. Finally, wrappers is provided for `Multistate` in the `BayesTraits` package (see `multistateML` and `multistateMCMC`). Several plotting functions (`HPDbars`, `clade.bars`, `box.clades`, `box.tips`, `tip.color`, `edge.color` have been moved to the **viper** package.

Author(s)

Natalie Cusimano, Christoph Heibl, Franz-Sebastian Krah, Maintainer: Christoph Heibl (<christoph.heibl@gmx.net>)

See Also

`ape`

`aliscore` *Masking of Sequence Alignments with ALISCORE*

Description

Provides a interface to **Aliscore**, in order to remove problematic regions of a DNA sequence alignment.

Usage

```
aliscore(x, gaps = "5state", w = 6, r, t, l, s, o, exec)
```

Arguments

`x`	DNA sequences of class `DNAbin`.
`gaps`	A vector of mode `"character"` indicating how gaps shall be treated: as `"5state"` or as `"ambiguous"`.
`w`	An integer giving the size of the sliding window.
`r`	An integer giving the number of random pairwise sequence comparisons; defaults to `4 * N`.
`t`	*Not yet implemented.*
`l`	*Not yet implemented.*
`s`	*Not yet implemented.*
`o`	A vector of mode `"character"` containing outgroup taxon names.
`exec`	A character string, giving the path to the Aliscore script.

Value

A `matrix` of class `"DNAbin"`.

Note

This function was developed with ALISCORE version 2. **ALISCORE is no longer available and seems to be discontinued by the authors.**

References

Misof, B. and K. Misof. 2009. A Monte Carlo approach successfully identifies randomness in multiple sequence alignments: a more objective means of data exclusion. *Syst. Biol.* **58**: 21–34.

Kueck, P., K. Meusemann, J. Dambach, B. Thormann, B.M. von Reumont, J.W. Waegele and B. Misof. 2010. Parametric and non-parametric masking of randomness in sequence alignments can be improved and leads to better resolved trees. *Frontiers in Zoology* **7**: 10.

See Also

mafft and prank for multiple sequence alignment; gblocks for another alignment masking algorithm.

Examples

```
data(ips.28S)
## Not run: aliscore(ips.28S)
```

---

aperm.DNAbin *Array Transposition for DNAbin*

Description

S3 method for aperm on objects of class DNAbin. Strips the class before permuting so that apply receives a plain raw matrix from the internal aperm call, matching the behaviour of aperm.default prior to R-devel revision 90136 (2026-06-12).

Usage

```
## S3 method for class 'DNAbin'
aperm(a, perm = NULL, resize = TRUE, ...)
```

Arguments

a	A matrix of class DNAbin.
perm	An integer vector giving the permutation of the dimensions.
resize	Logical; passed to aperm.default.
...	Further arguments passed to aperm.default.

Value

A raw matrix without the DNAbin class.

blastn — *Nucleotide-Nucleotide BLAST*

Description

Provides an interface to BLASTN

Usage

```
blastn(query, db)
```

Arguments

query	An object of class DNAbin containing the sequences which will be blasted against db.
db	An object of class DNAbin containing the reference sequences, i.e. the sequences against which query will be blasted.

code.simple.gaps — *Simple Gap/Indel Coding*

Description

code.simple.gaps takes an aligned DNA sequence matrix and codes the simple gaps, i.e. gaps that do not overlap with other gaps. The gapped positions are excluded from the matrix and the coded gap characters are appended to the matrix.

Usage

```
code.simple.gaps(x, append = TRUE)
```

Arguments

x	An object of class DNAbin.
append	Logical.

Value

An object of class DNAbin.

Author(s)

Christoph Heibl

References

Simmons, M.P. & H. Ochoterena. 2000. Gaps as characters in sequence-based phylogenetic analyses. *Systematic Biology* **49(2)**: 369–381.

See Also

deleteGaps, deleteEmptyCells, trimEnds

---

collapseUnsupportedEdges

Collapse Unsupported Edges/Branches in a Phylogeny

Description

Given a set of node support values (e.g., bootstrap proportions, posterior probabilities) and a certain threshold, all edges receiving less support than the threshold will be collapsed.

Usage

```
collapseUnsupportedEdges(phy, value = "node.label", cutoff)
```

Arguments

phy	An object of class phylo.
value	A character string giving the name of the list element of the phylo object that contains the support values; default is "node.label".
cutoff	A numeric value giving the threshold below which edges will be collapsed.

Details

Collapsing weekly supported edges can help draw attention to those parts of a phylogeny that are really supported by the underlying data.

Value

An object of class phylo.

Examples

```
## phylogeny of bark beetles
data(ips.tree)

## non-parametric bootstrap proportions (BP)
ips.tree$node.label

## collapse clades with < 70 BP
tr <- collapseUnsupportedEdges(ips.tree, "node.label", 70)
```

```
## show new topology
plot(tr, no.margin = TRUE)
```

---

combMyTree *Graft Polytomies on Tips of Phylogeny*

Description

Graft polytomies on the tips of a class phylo object.

Usage

```
combMyTree(phy, data, tips, brlen = 0, unit = "absolute", annotate = FALSE)
```

Arguments

phy	An object of class phylo.
data	A data frame with at least one column whose contents must be identical to the tip labels of the phylogeny in order to be matched automatically. Optionally, the new tip labels for the extended phylogeny can stem from another column, which is identified with tips.
tips	A character string giving the name of the column in data that contains the new tip labels. Alternatively, if tips is missing, combMyTree will create new tip labels by appending numbers to the former tip labels starting with 0.
brlen	A numeric giving either the absolute branch lengths or a fraction of some other value (depending on "unit").
unit	A character string according to which the number given by brlen will be interpreted. If it is "percent-root-age" (only possible with ultrametric trees), terminal branches will be assigned brlen times the root age and the whole tree will be rescaled so as not to change its age. All other strings will be interpreted as "absolute", i.e. terminal branches will be assigned brlen and the whole tree will not be rescaled.
annotate	Logical, if TRUE, the former tip labels will be turned into node labels. Note, that this will overwrite potentially existing node labels.

Value

An object of class phylo with nrow(data) tips.

See Also

forceEqualTipHeights

Examples

```
data(ips.tree)
## Simulate varying number of intraspecific observations
s <- sapply(1:Ntip(ips.tree), sample, x = 1:3, size = 1)
x <- rep(ips.tree$tip.label, times = s)
x <- data.frame(old_tips = x,
                new_tips = paste0(x, unlist(lapply(s, function(z) 1:z))))
## Create polytomies, generate tip labels automatically
tre1 <- combMyTree(ips.tree, x, brlen = 0, unit = "absolute")
plot(tre1, no.margin = TRUE, cex =.5)
## Create polytomies, take tip labels from data frame
tre2 <- combMyTree(ips.tree, x, tips = "new_tips", brlen = 0, unit = "absolute")
plot(tre2, no.margin = TRUE, cex =.5)
```

---

del.miss — *Delete Missing Data from DNA Sequences*

Description

Remove gaps ("-") and/or missing and ambiguous data ("N", "?") from a sample of DNA sequences.

Usage

```
del.miss(x)
```

Arguments

x	A matrix, a list, or a vector of class DNAbin containing the DNA sequences.

Value

A list or a vector of class DNAbin.

---

deleteGaps — *Remove Gap Positions From DNA Sequences*

Description

Remove indel positions (or gaps) from a DNA sequence alignment. For faster execution, deleteGaps handles sequences in **ape**'s bit-level coding scheme.

Usage

```
deleteGaps(x, gap.max = nrow(x) - 4)
```

Arguments

x	An object of class DNAbin.
gap.max	An integer, which gives the maximum number of gap characters ("-") that will be tolerated at any given alignment position (column). Only values between 0 and nrow(x) - 4 make sense phylogenetically.

Details

The default, nmax = nrow(x) - 4, removes all those positions from the alignment, which contain at least four non-gap characters, which is the minimum number of sequences needed to produce a non-trivial unrooted topology. All gaps will be excluded by selecting nmax = 0 and half of all gaps with nmax = nrow(x) / 2.

In contrast, del.gaps removes all gap characters from the alignment, so most probably the result will not be a set of sequences of equal length and the matrix will be coerced to a list.

Value

An object of class DNAbin.

See Also

code.simple.gaps for coding of simple gaps, del.gaps for removal of all gap symbols from an alignment, gblocks and aliscore for more sophisticated methods of cleaning/masking alignments.

---

descendants *Descendants of an Internal Node in a Phylogeny*

Description

For any given internal node of a phylogeny, the function returns a vector containing the node numbers descending from that node.

Usage

```
descendants(phy, node, type = "t", ignore.tip = TRUE, labels = FALSE)
```

Arguments

phy	an object of class phylo.
node	an integer giving the number of the internal node.
type	a character string, may be "daughter", "internal", "terminal", "all", or any unambiguous abbreviation of these.
ignore.tip	logical, if ignore.tip = FALSE, the function will issue an error when node is not internal, otherwise the number of the corresponding terminal node will be returned.
labels	logical, determines if node labels are returned instead of node number, currently ignored unless type = "t".

Value

A vector containing terminal node numbers or tip labels.

Author(s)

Christoph Heibl

See Also

sister, noi

Examples

```
# generate a random tree with 12 terminal and 11 internal nodes:
tree <- rtree(12)

# get the descendants of internal node 15:
x <- descendants(tree, 15)
```

DNAbin2index *Conversion of DNAbin to Index*

Description

Extract the indices of non-empty positions in a sample of DNA sequences to zzz.

Usage

```
DNAbin2index(x)
```

Arguments

x	A matrix of class DNAbin.

See Also

index2DNAbin

EmptyCells *Identify/Delete Spurious Rows and Columns from DNA Alignments*

Description

After subsetting (see e.g. DNAbin), DNA sequence alignments can contain rows and columns that consist entirely of missing and/or ambiguous character states. identifyEmptyCells will identify and deleteEmptyCells will delete all such rows (taxa) and columns (characters) from a DNA sequence alignment.

Usage

```
deleteEmptyCells(
  DNAbin,
  margin = c(1, 2),
  nset = c("-", "n", "?"),
  quiet = FALSE
)

identifyEmptyCells(
  DNAbin,
  margin = c(1, 2),
  nset = c("-", "n", "?"),
  quiet = FALSE
)
```

Arguments

DNAbin	An object of class DNAbin.
margin	A vector giving the subscripts the function will be applied over: 1 indicates rows, 2 indicates columns, and c(1,2) indicates rows and columns.
nset	A vector of mode character; rows or columns that consist **only** of the characters given in nset will be deleted from the alignment. Allowed are "-", "?","n", "b", "d","h", "v", "r","y", "s", "w","k", and "m".
quiet	Logical: if set to TRUE, screen output will be suppressed.

Details

For faster execution, deleteEmptyCells handles sequences in **ape**'s bit-level coding scheme.

Value

An object of class DNAbin.

References

Cornish-Bowden, A. 1984. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Res.* **13**: 3021–3030.

See Also

trimEnds, deleteGaps

Examples

```
# COX1 sequences of bark beetles
data(ips.cox1)
# introduce completely ambiguous rows and colums
x <- as.character(ips.cox1[1:6, 1:60])
x[3, ] <- rep("n", 60)
x[, 20:24] <- rep("-", 6)
x <- as.DNAbin(x)
image(x)
# identify those rows and colums
(id <- identifyEmptyCells(x))
xx <- x[-id$row, -id$col]
# delete those rows and colums
x <- deleteEmptyCells(x)
image(x)
identical(x, xx)
```

---

eoi *Identification of Stem-Lineage-Edges and MRCAs*

Description

noi (**n**ode **o**f **i**nterest) identifies the most recent common ancestor (MRCA) and eoi (**e**dge **o**f **i**nterest) its subtending stem-lineage edge of one or more sets of taxa/tips.

Usage

```
eoi(phy, node, group, regex = FALSE, stem = FALSE, monophyletic = FALSE)

noi(phy, group, regex = FALSE, stem = FALSE, monophyletic = FALSE)
```

Arguments

phy	An object of class phylo.
node	A vector of mode "numeric" giving the nodes numbers of the nodes whose subtending stem-lineages will be identified.
group	A vector or list of vectors of mode character specifying the taxon set(s). Will be ignored if node is given.
regex	A logical, if regex = TRUE, taxon sets are matched to the tip labels as regular expressions of the form "taxon1|taxon2"; otherwise strings will be matched exactly (see which).
stem	Logical, ...
monophyletic	Logical, ...

Value

A vector of mode "numeric" containing node numbers.

See Also

mrca; descendants for the contrary operation to noi.

Examples

```
# molecular phylogeny of Ips bark beetles
# ---------------------------------------
data(ips.tree)
ips.tree <- ladderize(ips.tree)
ips.tree <- fixNodes(ips.tree)
clade1 <- descendants(ips.tree, 44, labels = TRUE)
mrca <- noi(ips.tree, clade1)
stem_lineage <- eoi(ips.tree, mrca)
ecol <- rep("black", Nedge(ips.tree))
ecol[stem_lineage] <- "red"
plot(ips.tree, no.margin = TRUE, edge.color = ecol)
nodelabels(node = mrca, pch = 22, col = "blue")
#gen <- sapply(viperidae$tip.label, function(x) unlist(strsplit(x, "_"))[1])
#tax <- data.frame(genus = gen, species = viperidae$tip.label, row.names = NULL)

# group can be a list
# -------------------
#myclades <- split(tax$species, tax$genus)
#nds <- noi(viperidae, myclades)
#plot(viperidae)
#nodeInfo(nds)

# group might contain tip numbers
# -------------------------------
#group <- list(c(17, 22), c(13, 1))
#plot(viperidae)
#append2tips(phy, tips = unlist(group), pch = 19)
#nds <- noi(viperidae, myclades)
#nodeInfo(nds)

# the 'group' argument can also take regular expressions
# ------------------------------------------------------
#rex <- "aspis"
#node <- noi(viperidae, rex, regex = TRUE)
#plot.phylo(viperidae, tip.color = 0, edge.color = 0)
#box.clades(viperidae, nodes = node, col = "#D2A6A7", align = "all")
#plot.phylo.upon(viperidae)
#nodelabels(node = node, pch = 21, cex = 1.2, col = "red", bg = "#D2A6A7")

# if the 'group' argument is a list of elements of length 2,
# n = length(group) nodes of interest will be returned
# ----------------------------------------------------
#group <- list(
```

```
#    c("Vipera_berus", "Vipera_ursinii"),
#    c("Vipera_aspis_ssp._aspis", "Vipera_latastei"),
#    c("Vipera_ammodytes_ssp._ammodytes",
#        "Vipera_ammodytes_ssp._montandoni"),
#    c("Macrovipera_lebetina", "Vipera_wagneri")
#)
#clades <- noi(viperidae, group)
#plot.phylo(viperidae, tip.color = 0, edge.color = 0)
#box.clades(viperidae, nodes = clades, col = c("#FFFFA5", "#D2A6A7",
#    "#A7D2A5", "#A5A6D2"), align = "all")
#plot.phylo.upon(viperidae)
```

`fixNodes` *Standard Node Numbering in Phylo Objects*

Description

(Re-)establishes the standard numbering of terminal and internal nodes in phylogenies represented as objects of class `phylo`.

Usage

```
fixNodes(phy)
```

Arguments

`phy`	An object of class `phylo`.

Details

Internal and terminal nodes of `phylo` objects have a standard, "canonical" node numbering. It consists of the following rules:

1. Number tip nodes from left/bottom to right/top by 1 to `ntip()`
2. Number internal nodes by preorder traversal (see Examples) consecutively from `ntip() + 1` to `nnode() + 1`

When reading phylogenetic trees from a NEXUS file that contains a `translate` section, it can happen that the terminal nodes (tips, leaves) of the corresponding `phylo` object are not numbered consecutively, which can be a problem in some downstream applications. You can use `fixNodes` to get the correct order of terminal node numbers.

`fixNodes` is also intended to re-establish the standard numbering of internal nodes and reorder all node value elements (e.g. node.label, posterior, ...) if a `phylo` object has been modified by either `root`, `ladderize`, or `rotate`.

Value

An object of class `phylo`.

See Also

read.tree, read.nexus, read.beast for reading trees in NEWICK and NEXUS format; ladderize and rotate for tree manipulation; node.support for plotting node support values has been moved to package **viper**. non-standard node elements in class phylo.

Examples

```
set.seed(9999)

# Create random topology with random node numbering
phy1 <- rtopology(10)

# Impose 'canonical' node numbering
phy2 <- fixNodes(phy1)

# Compare node numbering schemes
op = par(no.readonly = TRUE)
par(mfcol = c(1, 2), mar = c(0, 0, 3, 0))
plot(phy1, use.edge.length = FALSE, show.tip.label = FALSE)
nodelabels(); tiplabels()
title("random node numbers")
plot(phy2, use.edge.length = FALSE, show.tip.label = FALSE)
nodelabels(); tiplabels()
title("preorder traversal of internal nodes")
par(op)
```

---

forceEqualTipHeights *Equal Tip Heights*

Description

Modify terminal edge lengths to create "exactly" (see Details) equal tip heights (sum of edge lengths from root to tip)

Usage

```
forceEqualTipHeights(phy, baseline = "mean")
```

Arguments

phy	An object of class phylo.
baseline	A character string giving a function to calculate the baseline tip height, e.g. "min", "max" or "mean".

Details

What is "exactly" equal depends on the precision of the system (.Machine); in any case the resulting phylogeny will pass is.ultrametric with default arguments.

Value

An object of class phylo with changed terminal edge lengths.

Note

forceEqualTipHeights is only intended to correct small rounding errors in edge lengths, not to make an additive phylogeny ultrametric. For the latter, see e.g. chronos.

See Also

tipHeights

---

gblocks *Masking of Sequence Alignments with GBLOCKS*

Description

Provides a wrapper to Gblocks, a computer program written in ANSI C language that eliminates poorly aligned positions and divergent regions of an alignment of DNA or protein sequences. Gblocks selects conserved blocks from a multiple alignment according to a set of features of the alignment positions.

Usage

```
gblocks(
  x,
  b1 = 0.5,
  b2 = b1,
  b3 = ncol(x),
  b4 = 2,
  b5 = "a",
  target = "alignment",
  exec
)
```

Arguments

x	A matrix of DNA sequences of classes DNAbin.
b1	A real number, the **minimum number of sequences for a conserved position** given as a fraction. Values between 0.5 and 1.0 are allowed. *Larger* values will *decrease* the number of selected positions, i.e. are more *conservative*. Defaults to 0.5
b2	A real number, the **minimum number of sequences for a flank position** given as a fraction. Values must be equal or larger than b1. *Larger* values will *decrease* the number of selected positions, i.e. are *more conservative*. Defaults to 0.5

`b3` An integer, the **maximum number of contiguous nonconserved positions**; any integer is allowed. *Larger* values will *increase* the number of selected position, i.e. are *less conservative*. Defaults to the number of positions in the alignment.

`b4` An integer, the **minimum length of a block**, any integer equal to or bigger than 2 is allowed. *Larger* values will *decrease* the number of selected positions, i.e. are *more conservative*. Defaults to 2.

`b5` A character string indicating the **treatment of gap positions**. Three choices are possible. 1. `"n"`: *No* gap positions are allowed in the final alignment. All positions with a single gap or more are treated as a gap position for the block selection procedure, and they and the adjacent nonconserved positions are eliminated. 2. `"h"`: Only positions where *50% or more* of the sequences have a gap are treated as a gap position. Thus, positions with a gap in less than 50% of the sequences can be selected in the final alignment if they are within an appropriate block. 3. `"a"`: *All* gap positions can be selected. Positions with gaps are not treated differently from other positions (default).

`target` A vector of mode `"character"` giving the output format: `"alignment"` will return the alignment with only the selected positions, `"index"` will return the indices of the selected position, and `"score"` will provide a score for every position in the original alignment (0 for excluded, 1 for included).

`exec` A character string indicating the path to the GBLOCKS executable.

Details

Explanation of the routine taken from the Online Documentation: First, the degree of conservation of every positions of the multiple alignment is evaluated and classified as *nonconserved*, *conserved*, or *highly conserved*. All stretches of contiguous nonconserved positions bigger than a certain value (**b3**) are rejected. In such stretches, alignments are normally ambiguous and, even when in some cases a unique alignment could be given, multiple hidden substitutions make them inadequate for phylogenetic analysis. In the remaining blocks, flanks are examined and positions are removed until blocks are surrounded by highly conserved positions at both flanks. This way, selected blocks are anchored by positions that can be aligned with high confidence. Then, all gap positions -that can be defined in three different ways (**b5**)- are removed. Furthermore, nonconserved positions adjacent to a gap position are also eliminated until a conserved position is reached, because regions adjacent to a gap are the most difficult to align. Finally, small blocks (falling below the limit of **b4**) remaining after gap cleaning are also removed.

Value

A `matrix` of class `"DNAbin"`

Note

`gblocks` was last updated and tested to work with Gblocks 0.91b. If you have problems getting the function to work with a newer version of Gblocks, please contact the package maintainer.

References

Castresana, J. 2000. Selection of conserved blocks from multiple alignments for their use in phylogenetic analysis. *Molecular Biology and Evolution* **17**, 540-552.

Talavera, G., and J. Castresana. 2007. Improvement of phylogenies after removing divergent and ambiguously aligned blocks from protein sequence alignments. *Systematic Biology* **56**, 564-577.

Gblocks website: https://home.cc.umanitoba.ca/~psgendb/doc/Castresana/Gblocks_documentation.html

See Also

mafft and prank for sequence alignment; aliscore for another alignment masking algorithm.

Examples

```
data(ips.28S)
## Not run: gblocks(ips.28S)
```

---

index2DNAbin — *Conversion of Index to DNAbin*

Description

Use indices of non-empty positions to convert a list of DNA sequences into a matrix.

Usage

```
index2DNAbin(DNAbin, index)
```

Arguments

DNAbin	A list of class DNAbin.
index	A list of integers containing the indices of base positions.

See Also

DNAbin2index

---

ips.16S — *Bark Beetle 16S Sequences*

Description

This DNA alignment contains 376 positions of 42 sequences of 16S ribosomal DNA of the bark beetle genera *Ips*, *Orthotomicus*, and *Pityogenes* (Scolytinae, Curculionidae, Coleoptera).

Usage

```
data(ips.16S)
```

Format

The sequences are stored in binary format (see DNAbin).

Source

The sequences were downloaded and assembled from the Nucleotide repository at GenBank on February 8, 2014.

References

The nucleotide database on the NCBI website: https://www.ncbi.nlm.nih.gov/nuccore

Examples

```
data(ips.16S)
```

---

ips.28S *Bark Beetle 28S Sequences*

Description

This DNA alignment contains 562 positions of 28 sequences of 28S ribosomal DNA of the bark beetle genus *Ips* (Scolytinae, Curculionidae, Coleoptera).

Usage

```
data(ips.28S)
```

Format

The sequences are stored in binary format (see DNAbin).

Source

The sequences were downloaded and assembled from the Nucleotide repository at GenBank on February 8, 2014.

References

The nucleotide database on the NCBI website: https://www.ncbi.nlm.nih.gov/nuccore

Examples

```
data(ips.28S)
```

ips.cox1 — *Bark Beetle COX1 Sequences*

Description

This DNA alignment contains 770 positions of 26 sequences of cox1 of the bark beetle genera *Ips*, *Orthotomicus*, and *Pityogenes* (Scolytinae, Curculionidae, Coleoptera).

Usage

```
data(ips.cox1)
```

Format

The sequences are stored in binary format (see DNAbin).

Source

The sequences were downloaded and assembled from the Nucleotide repository at GenBank on February 8, 2014.

References

The nucleotide database on the NCBI website: https://www.ncbi.nlm.nih.gov/nuccore

Examples

```
data(ips.cox1)
```

ips.tree — *Ips Phylogeny*

Description

Phylogenetic tree of bark beetles (genus *Ips*).

Usage

```
data(ips.tree)
```

Format

The format is: List of 5 $ edge : int [1:72, 1:2] 38 39 39 40 41 42 42 43 44 45 ... $ Nnode : int 36 $ tip.label : chr [1:37] "Ips_acuminatus" "Ips_duplicatus" "Ips_integer" "Ips_plastographus" ... $ edge.length: num [1:72] 0.2806 0.0727 0.0295 0.0097 0.021 ... $ node.label : chr [1:36] "" "100" "21" "12" ... - attr(*, "class")= chr "phylo" - attr(*, "order")= chr "cladewise"

Examples

```
data(ips.tree)
plot(ips.tree)
```

---

mafft *Sequence Alignment with MAFFT*

Description

This function is a wrapper for MAFFT and can be used for (profile) aligning of DNA and amino acid sequences.

Usage

```
mafft(
  x,
  y,
  add,
  method = "auto",
  maxiterate = 0,
  op = 1.53,
  ep = 0,
  gt = NULL,
  options,
  thread = -1,
  exec,
  quiet,
  file
)
```

Arguments

x	An object of class DNAbin or AAbin.
y	An object of class DNAbin or AAbin, if given both x and y are preserved and aligned to each other ("profile alignment").
add	A character string giving the method used for adding y to x: "add", "addprofile" (default), or any unambiguous abbreviation of these.
method	A character string giving the alignment method. Available accuracy-oriented methods for less than 200 sequences are "localpair", "globalpair", and "genafpair"; "retree 1" and "retree 2" are for speed-oriented alignment. The default is "auto", which lets MAFFT choose an appropriate alignment method.
maxiterate	An integer giving the number of cycles of iterative refinement to perform. Possible choices are 0: progressive method, no iterative refinement (default); 2: two cycles of iterative refinement; 1000: at most 1000 cycles of iterative refinement.

op	A numeric giving the gap opening penalty at group-to-group alignment; default 1.53.
ep	A numeric giving the offset value, which works like gap extension penalty, for group-to-group alignment; default 0.0, but 0.123 is recommended if no long indels are expected.
gt	An object of class phylo that is to be used as a guide tree during alignment. The default is to let MAFFT use its own internal guide tree
options	A vector of mode character specifying additional arguments to MAFFT, that are not included in mafft such as, e.g., --adjustdirection.
thread	Integer giving the number of physical cores MAFFT should use; with thread = -1 the number of cores is determined automatically.
exec	A character string giving the path to the MAFFT executable including its name, e.g. something like /user/local/bin/mafft under UNIX-alikes.
quiet	Logical, if set to TRUE, mafft progress is printed out on the screen.
file	A character string indicating the filename of the output FASTA file; if this is missing the the alignment will be returned as matrix of class DNAbin or AAbin.

Details

"localpair" selects the **L-INS-i** algorithm, probably most accurate; recommended for <200 sequences; iterative refinement method incorporating local pairwise alignment information.

"globalpair" selects the **G-INS-i** algorithm suitable for sequences of similar lengths; recommended for <200 sequences; iterative refinement method incorporating global pairwise alignment information.

"genafpair" selects the **E-INS-i** algorithm suitable for sequences containing large unalignable regions; recommended for <200 sequences.

"retree 1" selects the **FFT-NS-1** algorithm, the simplest progressive option in MAFFT; recommended for >200 sequences.

"retree 2" selects the **FFT-NS-2** algorithm that uses a second iteration of alignment based on a guide tree computed from an FFT-NS-1 alignment; this is the default in MAFFT; recommended for >200 sequences.

Value

A matrix of class "DNAbin" or "AAbin".

Note

mafft was last updated and tested to work with MAFFT 7.205. If you have problems getting the function to work with a newer version of MAFFT, please contact the package maintainer.

References

Katoh, K. and H. Toh. 2008. Recent developments in the MAFFT multiple sequence alignment program. *Briefings in Bioinformatics* **9**: 286-298.

Katoh, K., K.-i. Kuma, H. Toh, and T. Miyata. 2005. Mafft version 5: improvement in accuracy of multiple sequence alignment. *Nucleic Acids Research* **33**: 511–518.

Katoh, K., K. Misawa, K.-i. Kuma, and T. Miyata. 2002. Mafft: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleid Acids Research* **30**: 3059–3066.

https://mafft.cbrc.jp/alignment/software/index.html

See Also

`read.fas` to import DNA sequences; `prank` for another alignment algorithm; `gblocks` and `aliscore` for alignment cleaning.

`mafft.merge` *Profile Alignment with MAFFT*

Description

Merge two or more DNA or amino acid sequence alignments by profile alignment with MAFFT.

Usage

```
mafft.merge(subMSA, method = "auto", gt, thread = -1, exec, quiet = TRUE)
```

Arguments

`subMSA`	A list of objects of class `DNAbin` or `AAbin`.
`method`	A character string giving the alignment method. Available accuracy-oriented methods for less than 200 sequences are `"localpair"`, `"globalpair"`, and `"genafpair"`; `"retree 1"` and `"retree 2"` are for speed-oriented alignment. The default is `"auto"`, which lets MAFFT choose an appropriate alignment method.
`gt`	An object of class `phylo` that is to be used as a guide tree during alignment.
`thread`	Integer giving the number of physical cores MAFFT should use; with `thread = -1` the number of cores is determined automatically.
`exec`	A character string giving the path to the MAFFT executable including its name, e.g. something like `/user/local/bin/mafft` under UNIX-alikes.
`quiet`	Logical, if set to `TRUE`, mafft progress is printed out on the screen.

Value

An object of class `"DNAbin"` or `"AAbin"`.

mrbayes *Bayesian MCMC Tree Search with MrBayes*

Description

Provides a wrapper for Bayesian phylogenetic tree search through MrBayes (Huelsenbeck & Ronquist, 2001; Ronquist & Huelsenbeck, 2003).

Usage

```
mrbayes(
  x,
  file = "",
  lset,
  prset,
  mcmc,
  unlink,
  constraint,
  burnin = 10,
  contype = "allcompat",
  exec,
  run = FALSE
)
```

Arguments

x	An object of class DNAbin in the case of mrbayes or a matrix of mode character in the case of mrbayes.mixed.
file	A character string, giving the name of the MrBayes input file.
lset	A list as returned by mrbayes.prset containing the parameter setting for the prior distributions.
prset	A list as returned by mrbayes.prset containing the parameter setting for the prior distributions.
mcmc	A list as returned by mrbayes.mcmc containing the parameter setting for the Markov chain Monte Carlo (MCMC).
unlink	xxx
constraint	xxx
burnin	An integer; the number of samples from the MCMC to be discarded prior to further analysis.
contype	A character string; the type of consensus tree calculated from the posterior distribution of trees: either "halfcompat" (majority-rule consensus tree) or "allcombat" (strict consensus tree).
exec	A character string giving the full path of the MrBayes program.
run	Logical; run = FALSE will only print the NEXUS file, run = TRUE will also start the MCMC runs, if exec is correctly specified.

Details

mrbayes was last updated and tested with MrBayes **v3.2.2** under R 3.1.0 on a x86_64-apple-darwin10.8.0 (64-bit) platform. It is intended to offer a simply parameterized building block for larger scripts.

Value

None; a NEXUS file with MrBayes block is written to a file and, if run = TRUE, the MCMC runs in MrBayes are started.

References

J. P. Huelsenbeck & Ronquist F. 2001. MrBayes: Bayesian inference of phylogenetic trees. *Bioinformatics* **17**: 754-755.

Ronquist F. & J. P. Huelsenbeck. 2003. MrBayes 3: Bayesian phylogenetic inference under mixed models. *Biometrics* **19**: 1572-1574. MrBayes website: https://mrbayes.sourceforge.net/.

See Also

mafft and prank for sequence alignment; raxml for maximum likelihood tree search.

Examples

```
data(ips.cox1)
x <- ips.cox1[, 100:140] # tiny alignment
mrbayes(x, file = "", mcmc = mrbayes.mcmc(ngen = 100), run = FALSE)
## Not run:
library(phangorn)
tree <- rtree(10)
Y1 <- simSeq(tree, l = 20)
Y2 <- simSeq(tree, l = 20, type = "USER", levels=c("0", "1"))
Y <- cbind(as.character(Y1), as.character(Y2))

## End(Not run)
```

---

mrbayes.lset *MrBayes model settings*

Description

Specify a model of DNA, Protein or trait evolution for mrbayes.

Usage

```
mrbayes.lset(
  nucmodel = "4by4",
  nst = 1,
  code = "universal",
  ploidy = "diploid",
  rates = "equal",
  ngammacat = 4,
  nlnormcat = 4,
  nmixtcat = 4,
  nbetacat = 5,
  omegavar = "equal",
  covarion = "no",
  coding = "all",
  parsmodel = "no",
  partition = 1
)
```

Arguments

nucmodel	A character string
nst	xxx
code	xxx
ploidy	xxx
rates	xxx
ngammacat	xxx
nlnormcat	xxx
nmixtcat	xxx
nbetacat	xxx
omegavar	xxx
covarion	xxx
coding	xxx
parsmodel	xxx
partition	A character string serving as an identifier in a multi-partition model.

Details

xxx

Value

A list of model parameters.

See Also

mrbayes, mrbayes.prset , mrbayes.mcmc.

mrbayes.mcmc *MrBayes MCMC settings*

Description

Specify the MCMC settings for mrbayes.

Usage

```
mrbayes.mcmc(
  ngen = 1e+06,
  nruns = 2,
  nchains = 4,
  temp = 0.1,
  reweight = c(0, 0),
  swapfreq = 1,
  nswaps = 1,
  samplefreq = 500,
  printfreq = 1000,
  printall = TRUE,
  printmax = 8,
  mcmcdiagn = TRUE,
  diagnfreq = 5000,
  diagnstat = "avgstddev",
  minpartfreq = 0.1,
  allchains = FALSE,
  allcomps = FALSE,
  relburnin = TRUE,
  burnin = 0,
  burninfrac = 0.25,
  stoprule = FALSE,
  stopval = 0.05,
  savetrees = FALSE,
  checkpoint = TRUE,
  checkfreq = 2000,
  startparams = "current",
  starttree = "current",
  nperts = 0,
  data = TRUE,
  ordertaxa = FALSE,
  append = FALSE,
  autotune = TRUE,
  tunefreq = 100
)
```

Arguments

ngen	A character string

nruns	xxx
nchains	xxx
temp	xxx
reweight	xxx
swapfreq	xxx
nswaps	xxx
samplefreq	xxx
printfreq	xxx
printall	xxx
printmax	xxx
mcmcdiagn	xxx
diagnfreq	xx
diagnstat	xxx
minpartfreq	xxx
allchains	xxx
allcomps	xxx
relburnin	xxx
burnin	xxx
burninfrac	xxx
stoprule	xxx
stopval	xxx
savetrees	xxx
checkpoint	xxx
checkfreq	xxx
startparams	xxx
starttree	xxx
nperts	xxx
data	xxx
ordertaxa	xxx
append	xxx
autotune	xxx
tunefreq	xxx

Details

xxx

Value

A list of MCMC parameters.

See Also

mrbayes, mrbayes.lset , mrbayes.prset.

---

mrbayes.prset *MrBayes prior settings*

Description

Specify parameters for prior distributions for mrbayes.

Usage

```
mrbayes.prset(
  traitiopr,
  revmatpr,
  aamodelpr,
  aarevmatpr,
  omegapr,
  ny98omega1pr,
  ny98omega3pr,
  m3omegapr,
  codoncatfreqs,
  statefreqpr,
  shapepr,
  ratecorrpr,
  pinvarpr,
  covswitchpr,
  symdirihyperpr,
  topologypr,
  brlenspr,
  treeagepr,
  speciationpr,
  extinctionpr,
  fossilizationpr,
  samplestrat,
  sampleprop,
  popsizepr,
  popvarpr,
  nodeagepr,
  clockratepr,
  clockvarpr,
  cppratepr,
  cppmultdevpr,
  tk02varpr,
  wnvarpr,
  igrvarpr,
```

```
  ilnvarpr,
  ratepr,
  generatepr
)
```

Arguments

traitiopr	A character string
revmatpr	xxx
aamodelpr	xxx
aarevmatpr	xxx
omegapr	xxx
ny98omega1pr	xxx
ny98omega3pr	xxx
m3omegapr	xxx
codoncatfreqs	xxx
statefreqpr	xxx
shapepr	xxx
ratecorrpr	xxx
pinvarpr	xxx
covswitchpr	xxx
symdirihyperpr	xxx
topologypr	xxx
brlenspr	xxx
treeagepr	xxx
speciationpr	xxx
extinctionpr	xxx
fossilizationpr	xxx
samplestrat	xxx
sampleprop	xxx
popsizepr	xxx
popvarpr	xxx
nodeagepr	xxx
clockratepr	xxx
clockvarpr	xxx
cppratepr	xxx
cppmultdevpr	xxx
tk02varpr	xxx

wnvarpr	xxx
igrvarpr	xxx
ilnvarpr	xxx
ratepr	xxx
generatepr	xxx

Details

xxx

Value

A list of parameters for prior distributions.

See Also

mrbayes, mrbayes.lset , mrbayes.mcmc.

---

multistate *MULTISTATE*

Description

These functions provide wrappers to BayesMultiState in the BayesTraits package written by Mark Pagel and Andrew Meade.

Usage

```
multistateML(phy, traits, model = "ARD", anc.states = TRUE,
  path = "/Applications/BayesTraits", dir = NULL)

multistateMCMC(phy, traits, model = "ARD", anc.states = TRUE,
    rd = 2, rjhp = NULL, fixNodes = NULL, it = 1e+05, bi = 10000,
    sa = 1000, path = "/Applications/BayesTraits", dir = NULL)
```

Arguments

phy	an object of class phylo.
traits	a data.frame with two columns. The first column contains the taxon labels, the second column contains the character states.
model	a character string to select a model of rate evolution. One of "ER" (equal rates), "FB", "ROW", "SYM", or "ARD" (all rates different).
anc.states	either logical or a list, the latter containing the tip labels of those internal nodes, for which the likelihood of ancestral character states should be estimated.

`rd` a real number, giving the `RateDev` parameter, i.e., the deviation of the normal distribution, that changes to the rates are drawn from. Should be set such that acceptance of the rate parameters is about 0.2.

`rjhp` a character string giving the details of priors and hyperpriors for the reversible jump MCMC (rjMCMC). If left `NULL`, a conventional MCMC is used. In order to use the rjMCMC, you must specify the distribution of the prior and the interval of the uniform hyperprior distribution that seeds it. For example, `exp 0 30` specifies an exponential distribution seeded from a uniform distribution on the interval 0 to 30, and `gamma 0 10 0 10` specifies a gamma prior with its mean and standard deviation seeded from uniform distributions on the interval 0 to 10.

`fixNodes` a list giving fixed character states of certain internal nodes. This argument corresponds to the `fossil` command in the MultiState manual.

`it` numeric, sets the number of iterations to run the MCMC for.

`bi` numeric, sets the number of iterations of the MCMC that will be discarded as burn-in.

`sa` numeric, sets the the sample period in the MCMC.

`path` a character string giving the path to executables in the BayesTraits package.

`dir` a character string giving a directory name where the input and output files will be stored. The directory will be created by `multistateML` and must not exist already. If `dir = NULL` (default) input and output is written to the working directory (thereby overwriting existing output).

Author(s)

Christoph Heibl

References

The BayesTraits manual: http://www.evolution.reading.ac.uk/BayesTraitsV4.1.1/Files/BayesTraitsV4.1.1-Manual.pdf.

Pagel, M., A. Meade, and D. Barker. 2004. Bayesian estimation of ancestral character states on phylogenies. *Syst. Biol.* **53**: 673-684.

Pagel, M. and A. Meade. 2006. Bayesian analysis of correlated evolution of discrete characters by reversible-jump Markov chain Monte Carlo. *Am. Nat.* **167**: 808-825.

See Also

`ace`

neighboringPairs *Neighboring Nodes in a Minimum Spanning Tree*

Description

Finds all pairs of adjacent nodes, i.e. nodes separated by only one edge, in a minimum spanning tree

Usage

```
neighboringPairs(mst)
```

Arguments

mst	An object of class mst.

ntip *Numbers of Tips of (Sub)trees*

Description

Counts the number of tips of a given clade of a phylogenetic tree.

Usage

```
ntip(phy, node)
```

Arguments

phy	An object of class phylo.
node	An integer given the number of an internal node.

Value

An integer giving the number of tips.

Examples

```
set.seed(1234)
tr <- rtree(12)
plot(tr); nodelabels()
ntip(tr, 16)
```

partitionfinder *PartitionFinder*

Description

Provides a wrapper to the PartitionFinder software.

Usage

```
partitionfinder(
  alignment,
  user.tree,
  branchlengths = "linked",
  models = "all",
  model.selection = "BIC",
  search = "greedy",
  exec = "/Applications/PartitionFinderV1.1.1_Mac/PartitionFinder.py"
)
```

Arguments

alignment	A
user.tree	A
branchlengths	A
models	A
model.selection	A
search	A
exec	A character string giving the path to the executable (python script).

References

Lanfear, R., B. Calcott, S.Y.W. Ho, and S. Guindon. 2012. PartitionFinder: combined selection of partitioning schemes and substitution models for phylogenetic analyses. *Molecular Biology and Evolution* **29**: 1695-1701.

Lanfear, R., B. Calcott, K. David, C. Mayer, and A. Stamatakis. 2014. Selecting optimal partitioning schemes for phylogenomic datasets. *BMC Evolutionary Biology* **14**: 82.

pathd8 *PATHd8*

Description

This function is a wrapper for PATHd8 and can be used for phylogenetic dating, especially of large trees

Usage

```
pathd8(phy, exec = "/Applications/PATHd8/PATHd8", seql, calibration)
```

Arguments

phy	An object of class phylo.
exec	A character string giving the path to the PATHd8 program.
seql	sequence length of alignment
calibration	A data frame with 4 columns and as many rows as calibration points. Columns are: taxon 1; taxon 2; one of c("minage", "maxage", "fixage"); age.

Value

tree list of ultrametric trees returned from PATHd8 of class phylo. First tree is PATHd8 chronogram, which is a calibrated ultrametric tree. Second is a PATH tree, which is a ultrametric tree without calibration.

Author(s)

Franz-Sebastian Krah

References

Britton et al (2006). PATHd8—a new method for estimating divergence times in large phylogenetic trees without a molecular clock. Available from the authors (www.math.su.se/PATHd8)

Britton et al. (2007). Estimating divergence times in large phylogenetic trees. Systematic biology. 56:741–752

Examples

```
## Not run:
## This example is taken from the PATHD8 manual
cal <- rbind(cal1 = c("Rat", "Ostrich", "minage", 260),
cal2 = c("Human", "Platypus", "fixage", 125),
cal3 = c("Alligator", "Ostrich", "minage", 150))
colnames(cal) = c("tax1", "tax2", "age_type", "age")
phy <- read.tree(text = paste0("((((Rat:0.007148,Human:0.001808):0.024345,",
                               "Platypus:0.016588):0.012920,(Ostrich:0.018119,",
```

```
                                "Alligator:0. 006232):0.004708):0.028037,Frog:0);")
seql <- 1823
pathd8(phy, exec = "/Applications/PATHd8/PATHd8", seql = seql, calibration = cal)

## End(Not run)
```

---

phylo2mafft *Convert Trees for MAFFT*

Description

Converts a phylogenetic tree of class "phylo" to a format usable as a guide tree by MAFFT. This function is called internally by mafft.

Usage

```
phylo2mafft(phy, file)
```

Arguments

phy	A phylogenetic tree of class phylo.
file	A character string giving a filename. May be missing, in which case the results are only printed on the screen.

Value

A matrix coding the MAFFT-formatted tree, as a side effect the same matrix is written to file.

References

The MAFFT website: https://mafft.cbrc.jp/alignment/software/index.html

See Also

mafft for an interface to MAFFT.

phylo2mst — *Conversion from PHYLO to MST Object*

Description

Converts a phylogenetic tree (class `phylo`) into a minimum spanning tree (class `mst`).

Usage

```
phylo2mst(phy)
```

Arguments

`phy`	An object of class `phylo`.

Details

The current version of `phylo2mst` does not handle polytomies and does not incorporate branch length information. Note that topological information is lost during the conversion.

Examples

```
phy <- rtree(12)
plot(phy)
mst <- phylo2mst(phy)
plot(mst)
```

pis — *Number of Potentially-Informative Sites*

Description

Returns the number or positions of potentially-informative (parsimony-informative, phylogenetically-informative) sites in DNA sequence alignment.

Usage

```
pis(x, what = "fraction", use.ambiguities = FALSE)
```

Arguments

`x`	An object of class `DNAbin`.
`what`	Either of `"absolute"`, `"fraction"`, or `"index"`, which will return the absolute number, the relative number or the indices of the potentially-informative sites.
`use.ambiguities`	*Not yet available*.

Value

Numeric (depending on `what`, the number, fraction, or indices of potentially-informative nucleotide sites).

Examples

```
data(ips.16S)

# number of potentially-informative sites:
pis(ips.16S, what = "abs")

# proportion of potentially-informative sites:
pis(ips.16S, what = "frac")

# indices of potentially-informative sites:
pis(ips.16S, what = "ind")
```

---

prank *PRANK*

Description

DNA sequence Alignment Using the program PRANK.

Usage

```
prank(x, outfile, guidetree = NULL, gaprate = 0.025,
    gapext = 0.75, path)
```

Arguments

`x`	an object of class `DNAbin`.
`outfile`	a character string giving a name for the output file.
`guidetree`	an object of class `phylo` to be used as guidetree in alignment.
`gaprate`	numeric giving the gap opening rate; defaults to 0.025.
`gapext`	numeric giving the gap extension penalty; defaults to 0.75.
`path`	a character string indicating the path to the PRANK executable.

Value

`matrix` of class `"DNAbin"`

Note

`prank` was last updated and tested to work with PRANK v. 120814 on Windows XP. If you have problems getting the function to work with a newer version of PRANK, contact the package maintainer.

References

http://wasabiapp.org/software/prank/

See Also

read.fas to import DNA sequences; mafft for another alignment algorithm; gblocks and aliscore for alignment cleaning.

---

raxml *Maximum Likelihood Tree Estimation with RAxML*

Description

Provides an interface to the C program **RAxML** (see Reference section) for maximum likelihood estimation of tree topology and/or branch lengths, rapid and conventional non-parametric bootstrapping, mapping splits onto individual topologies, and a lot more. See the RAxML manual for details, especially if you are a new user of RAxML.

Usage

```
raxml(
  DNAbin,
  m = "GTRCAT",
  f,
  N,
  p,
  b,
  x,
  k,
  weights,
  partitions,
  outgroup,
  backbone = NULL,
  file = paste0("fromR_", Sys.Date()),
  exec,
  threads
)
```

Arguments

DNAbin	A matrix of DNA sequences of class DNAbin.
m	A vector of mode "character" defining a model of molecular evolution; currently only GTR model available.
f	A vector of mode "character" selecting an RAxML algorithm analogous to the -f flag (see Detail section and RAxML manual).

N	Either of mode "integer" or "character". Integers give the number of independent searches on different starting tree or replicates in bootstrapping. Alternatively, one of four bootstopping criteria can be chosen: "autoFC", "autoMR", "autoMRE", or "autoMRE_IGN".
p	Integer, setting a random seed for the parsimony starting trees.
b	Integer, setting a random seed for bootstrapping.
x	Integer, setting a random seed for rapid bootstrapping.
k	Logical, if TRUE, the branch lengths of bootstrapped trees are recorded.
weights	A vector of mode "numeric" giving integers to assign individual weights to each column of the alignment. (-a)
partitions	A data frame giving the partitions of the alignment.
outgroup	A vector of mode "character" containing the names of the outgroup taxa.
backbone	A phylo object representing a backbone tree.
file	A vector of mode "character" giving a name for the output files.
exec	A vector of mode "character" giving the path to the directory containing the RAxML executable. The default value will work on Mac OS X if the folder containing the executable is renamed to "RAxML-8.0.3".
threads	Integer, giving the number of parallel threads to use (PTHREADS only).

Details

There are some limitations of this wrapper compared to RAxML run directly from the command line.

1. Only DNA is allowed as data type.
2. Option f can only take a limited number of values (d, a).

RAxML needs the specification of random seeds for parsimony estimation of starting trees and for bootstrap resampling. The corresponding argument names in raxml are identical to the flags used by RAxML (-p, -b, and -x). If you choose not to give any values, raxml will generate a (different) value for each required random seed every time it is called. Be aware that set.seed will work only for p, but not for b or x.

Value

A list with a variable number of elements, depending on the analysis chosen:

"info"	RAxML log file as character string
"bestTree"	MLE of tree
"bipartitions"	MLE of tree annotated with bootstrap proportions
"bootstrap"	bootstrapped trees

Note

RAxML is a C program and the source code is not contained in this package. This means that in order to run this function you will need to install RAxML yourself. See https://cme.h-its.org/exelixis/web/software/raxml/ for the most recent documentation and source code of RAxML. Depending on where you chose to install RAxML, you need to adjust the `exec` argument.

`raxml` was last tested and running fine on Mac OS X with RAxML 8.0.29. Please be aware that calling third-party software from within R is a platform-specific process and I cannot guarantee that `raxml` will behave properly on any system.

References

(in chronological order)

Stamatakis, A., T. Ludwig and H. Meier. 2004. RAxML-III: A fast program for maximum likelihood-based inference of large phylogenetic trees. *Bioinformatics* **1**: 1–8.

Stamatakis, A. 2006. RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. *Bioinformatics* **22**: 2688–2690.

Stamatakis, A., P. Hoover, and J. Rougemont. 2008. A rapid bootstrap algorithm for the RAxML web-servers. *Syst. Biol.* **75**: 758–771.

Pattengale, N. D., M. Alipour, O. R. P. Bininda-Emonds, B. M. E. Moret, and A. Stamatakis. 2010. How many bootstrap replicates are necessary? *Journal of Computational Biology* **17**: 337-354.

Stamatakis, A. 2014. RAxML Version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* Advance Access.

See Also

`raxml.partitions` to store partitioning information in a data frame suitable for input as `partitions` argument in `raxml`.

Examples

```
## bark beetle sequences
data(ips.cox1)
data(ips.16S)
data(ips.28S)

ips <- cbind(ips.cox1, ips.16S, ips.28S,
            fill.with.gaps = TRUE)

exec <- "/Applications/RAxML-code/standard-RAxML/raxmlHPC-PTHREADS-AVX"
w <- sample(1:5, ncol(ips.cox1), replace = TRUE)

## Not run:

# Simple tree search with GTRCAT and GTRGAMMA
tr <- raxml(ips.cox1, f = "d", N = 2, p = 1234,
           exec = exec) # -1743.528461
tr <- raxml(ips.cox1, m = "GTRGAMMA", f = "d", N = 2, p = 1234,
           exec = exec)
```

```
# Applying weights to columns
tr <- raxml(ips.cox1, f = "d", N = 2, p = 1234,
           weights = w, exec = exec) # -1743.528461

# Rapid bootstrap
tr <- raxml(ips.cox1, m = "GTRGAMMA",
           f = "a", N = 10, p = 1234, x = 1234,
           exec = exec)

# Rapid bootstrap with automatic halt
tr <- raxml(ips.cox1, m = "GTRGAMMA",
           f = "a", N = "autoMRE", p = 1234, x = 1234,
           exec = exec)

## End(Not run)
```

---

raxml.partitions *Partition scheme for RAxML*

Description

Given a set of DNA sequence alignments, raxml.partitions creates a data frame with partition boundaries that can be input into raxml.

Usage

```
raxml.partitions(...)
```

Arguments

...	Two or more DNA sequence alignments of class DNAbin.

Details

For raxml.partitions to make sense, the DNA sequence alignments must be given exactly in the same order in which they are concatenated into a supermatrix (see Examples section). Without any testing, the type of sequences is supposed to be DNA.

Value

A data frame with four columns (type, locus, begin, and end) and number of rows corresponding to the number of partitions.

See Also

cbind.DNAbin to concatenate multiple alignments; raxml for an interface to RAxML.

Examples

```
## bark beetle sequences
data(ips.cox1)
data(ips.16S)
data(ips.28S)

## Note the same order of individual
## alignments in both functions:
## ----------------------------
raxml.partitions(cox1 = ips.cox1,
                 r16S = ips.16S,
                 r28S = ips.28S)

cbind(ips.cox1, ips.16S, ips.28S,
  fill.with.gaps = TRUE)
```

`rbeauti` *XML Input Files for BEAST*

Description

Prepare XML files for BEAST with R. BEAST uses an MCMC approach to estimate rooted phylogenies from molecular data (Drummond & Rambaut, 2007).

Usage

```
rbeauti(
  ...,
  file,
  template = "standard",
  link.clocks = TRUE,
  link.trees = TRUE,
  subst.model,
  clock,
  tree,
  taxonset,
  chain.length = 1e+07,
  log.every = 1000
)
```

Arguments

`...`	One or more object(s) of class `DNAbin`.
`file`	A connection, or a character string naming the file to write to. If left empty the XML tree will be printed to the screen (see Examples).
`template`	*Currently unused.*
`link.clocks`	Logical, indicating if clock models should be linked over partitions.

link.trees	Logical, indicating if tree models should be linked over partitions.
subst.model	A character string defining a substitution model, either "JC69", "HKY","TN93", or "GTR".
clock	A character string defining a clock model, either "Strict Clock", "Relaxed Clock Exponential","Relaxed Clock Log Normal", or "Random Local Clock".
tree	A character string defining a tree model.
taxonset	A list containing one or more taxon sets.
chain.length	Integer, the number of generations to run the MCMC.
log.every	Integer, defining how often samples from the posterior will be written to log files and shown on screen.

Details

rbeauti has been completely rewritten to work with **BEAST 2**. Currently rbeauti offers few options, because the idea is not to create ready-to-use XML file. That can be done conveniently with **BEAUti** (the BEAST package's genuine XML generator). Instead, rbeauti is intended to make the definition of large numbers of taxon sets easy. The creation of taxon sets can be done via R scripts and the resulting XML files can be further modified with BEAUti.

References

The XML reference at the BEAST 2 website: http://beast.community/xml_reference Drummond, A.J. & A. Rambaut. 2007. BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology* **7**: 240.

See Also

read.beast, read.beast.table

Examples

```
data(ips.16S)

## define taxon sets
spec <- rownames(ips.16S)
ingroup <- spec[grep("Ips|Orthomotomicus", spec)]
outgroup <- spec[grep("Pityogenes", spec)]
ts <- list(list(id = "ingroup", taxon = ingroup),
           list(id = "outgroup", taxon = outgroup))

## print XML file to screen
# rbeauti(ips.16S, taxonset = ts)
```

rc *Reverse-Complement of DNA sequences*

Description

Reverse, complement or reverse-complement of DNA sequences.

Usage

```
rc(seqs, i, complement = TRUE, reverse = TRUE, delete.gaps = FALSE)
```

Arguments

seqs	An object of class DNAbin.
i	Logical or numeric index to indicate a subset of sequences to manipulate. If not given the entire set of sequences will be manipulated.
complement	Logical, indicating if sequences will be turned into their complement.
reverse	Logical, indication if sequences will be reversed.
delete.gaps	Logical, indicating if gap character will be removed prior to sequence manipulation.

Value

An object of the same class and dimension as seqs.

Examples

```
## A minimal sequence alignment:
x <- list(
  seqA = c("a", "a", "c", "c", "g", "t"),
  seqB = c("n", "-", "r", "y", "g", "t"))
x <- as.DNAbin(x)

## Three choices of manipulation:
as.character(x)
as.character(rc(x))                      ## reverse-complement
as.character(rc(x, complement = FALSE))  ## only reverse
as.character(rc(x, reverse = FALSE))     ## only complement

## You can remove gaps:
as.character(rc(x, delete.gaps = TRUE))  ## gaps/indels removed
```

read *Reading Sequence Files*

Description

Read DNA and amino acid sequences from FASTA, PHILIP, and NEXUS formatted files.

Usage

```
read.fas(x, text)

read.nex(x)

read.phy(x)
```

Arguments

x	A character string, giving the file name.
text	A character string in FASTA format.

Value

An matrix (aligned sequences) or list (unaligned sequences) of class DNAbin or AAbin.

References

Maddison, D.R., D.L. Swofford, and W.P. Maddison. 1997. NEXUS: an extensible file format for systematic information. *Syst. Biol.* **46**: 590-621.

See Also

mafft and prank for sequence alignment, gblocks and aliscore for quality check and cleaning of sequence alignments, cbind.DNAbin for concatenation of sequence alignments.

Examples

```
## bark beetle COX1 sequences
data(ips.cox1)
## create temporary file names
format <- c(".fas", ".phy", ".nex")
fn <- sapply(format, tempfile,
             pattern = "ips", tmpdir = tempdir())
## write sequences files
write.fas(ips.cox1, fn[".fas"])
write.phy(ips.cox1, fn[".phy"])
write.nex(ips.cox1, fn[".nex"])
## read sequence files
fas <- read.fas(fn[".fas"])
```

```
phy <- read.phy(fn[".phy"])
nex <- read.nex(fn[".nex"])
## remove sequence files
unlink(fn)
```

`read.beast` *Read Bayesian Trees*

Description

Thess functions parse chronograms in NEXUS format as produced by TreeAnnotator or output by MrBayes.

Usage

```
read.mrbayes(file, digits = NULL)

read.beast(file, digits = NULL)

read.starbeast(file)
```

Arguments

`file`	A character string giving the input file, which must be a TreeAnnotator-generated chronogram in NEXUS format.
`digits`	NULL or integer, if `!is.null(digits)` values are rounded to the given integer.

Value

An object of class `phylo`.

Note

`read.starbeast` currently parses only scalars and ranges; node statistics with more than two values will be deleted and a warning message will be issued. Future version of `read.starbeast` will hopefully be able to append list or data frames to `phylo` objects. If you have any opinion or wishes regarding the question of how this exactly should be managed, send me a message.

`read.mrbayes` is intended to read single trees with annotated nodes. For reading tree samples from the posterior distribution there is a collection of perl script written by Johan Nylander (https://github.com/nylander/Burntrees).

Author(s)

Christoph Heibl

References

TreeAnnotator: http://beast.community/treeannotator

Metacomments in NEXUS: https://code.google.com/archive/p/beast-mcmc/wikis/NexusMetacommentFormat.wiki

See Also

read.beast.table to extract internal node data from NEXUS file, rbeauti to create XML input for BEAST. HPDbars for plotting highest posterior densities on phylogenies has been moved to package **viper**.

---

read.beast.table *Extract node data from BEAST chronogram*

Description

This function reads a BEAST chronogram such as produced by TreeAnnotator and extracts time, rate, and support values for internal and external nodes. Nodes in the resulting data frame are ordered exactly like in the NEXUS file.

Usage

```
read.beast.table(file, digits = 2)
```

Arguments

file	character string giving the input file, which must be a TreeAnnotator-generated chronogram in NEXUS format
digits	NULL or integer, if !is.null(digits) values are rounded to the given integer

Value

A matrix; each row corresponds to an internal node, the (ape!)number of which is given in the first column; the remaining columns list the node values extracted from the chronogram.

Author(s)

Christoph Heibl

See Also

read.beast to parse TreeAnnotator output, rbeauti to create XML input for BEAST. HPDbars for plotting highest posterior densities on phylogenies has been moved to package **viper**.

sister *Identification of Sister Nodes and Clades*

Description

For any given internal node in a phylogeny, this function returns the sister clade.

Usage

```
sister(phy, node, type = "terminal", label = FALSE)
```

Arguments

phy	An object of class phylo.
node	A vector of mode "numeric" or "character" giving the number(s) or name(s) of the tiplabel(s); these must be monophyletic.
type	A character string, may be "terminal", "internal", "daughter", "all", or any unambiguous abbreviation of these; "daughter" will return the MRCA of the sister clade of "node".
label	Logical, determining if tip number or tip labels will be returned.

Value

A vector of mode "numeric" or "character", containing either tip numbers or labels, respectively.

See Also

descendants, noi.

Examples

```
# A phylogeny of bark beetles ...
data(ips.tree)
tcol <- rep("black", Ntip(ips.tree))
tcol[ips.tree$tip.label %in% c("Ips_typographus", "Ips_nitidus")] <- "blue"
tcol[ips.tree$tip.label %in% c("Ips_duplicatus")] <- "red"
plot(ips.tree, no.margin = TRUE, tip.color = tcol)
# What is the sister species of Ips typographus?
sister(ips.tree, "Ips_typographus", label = TRUE)
# Return the MRCA of the sister clade of Ips duplicatus
x <- sister(ips.tree, "Ips_duplicatus", "daughter")
nodelabels(node = x, pch = 21, bg = "red")
```

splitIntoClades — *Find Monophyletic Subsets in Species Lists*

Description

Takes a phylogeny and a subset of its tiplabels and splits the list of tiplabels into monophyletic groups (clades).

Usage

```
splitIntoClades(phy, tips)
```

Arguments

phy	An object of class phylo.
tips	A vector of mode "character" containing any subset of the tiplabels in phy.

Value

A list.

terminalSisters — *Find Pairs of Sister Species*

Description

Finds pairs of sister species in a phylogenetic tree.

Usage

```
terminalSisters(phy, labels = TRUE)
```

Arguments

phy	An object of class phylo.
labels	Logical, indicating whether to return tip labels or tip numbers.

Value

A list of which each element contains the tip labels of a sister species pair.

Examples

```
set.seed(1234)
tr <- rtree(12)
plot(tr)
terminalSisters(tr)
```

tipHeights — *Tip Heights in a Phylogenetic Tree*

Description

For each tip (leave, terminal node) in the phylogenetic tree the edge lengths (branch lengths) from root to tip, be it units of time or divergence, is summed up.

Usage

```
tipHeights(phy)
```

Arguments

phy	an object of class phylo.

Value

a numeric vector with distances from root to tip for each tip in the phylogenetic tree.

Author(s)

Christoph Heibl

See Also

branching.times

traitRate — *Trait-Dependent Shifts in Molecular Rate*

Description

Detection of trait-dependent shifts in the rate of molecular evolution with **traitRate** (Mayrose & Otto, 2011).

Usage

```
traitRate(phy, seq, x, mainType = "Optimize_Model",
   n, charModelParam1 = 0.5, charModelParam2 = 1,
   gammaParam = 0.5, seqModelParam1 = 2,
   exec = "/Applications/traitRate-1.1/programs/traitRate")
```

Arguments

phy	a ultrametric phylogenetic tree of class phylo.
seq	a multiple sequence alignment of class DNAbin.
x	data frame containing a binary character in the first column.
mainType	character string giving the type of analysis; two choices are possible: "Optimize_Model" will produce MLE of parameters and "runTraitBootstrap" will perform a parametric bootstrap analysis.
n	numeric, the number of bootstrap replicates. Will be ignored if mainType = "Optimize_Model".
charModelParam1	numeric, giving an initial value for the rate of transitions of character state 0 to 1.
charModelParam2	numeric, giving an initial value for the rate of transitions of character state 1 to 0.
gammaParam	numeric, giving an initial value for the **alpha** parameter of the model of sequence evolution.
seqModelParam1	numeric, giving an initial value for the **kappa** parameter of the model of sequence evolution.
exec	character string giving the path to the program directory.

Value

Currently none, but look for the output files in the 'RESULTS' subdirectory in the current working directory.

Note

This function is under development!

Author(s)

Christoph Heibl

References

Mayrose, I. & S.P. Otto. 2011. A likelihood method for detecting trait-dependent shifts in the rate of molecular evolution. *Mol. Biol. Evol.* **28**: 759-770

See Also

read.tree for reading phylogenetic trees, read.fas for reading multiple sequence alignments in FASTA format.

trimEnds *Trim Alignment Ends*

Description

Trims both ends of a DNA sequence alignment to the first and last alignment positions that contain a minimum number of IUPAC base characters ("a", "c", "g", "t", "r", "y", "s", "w", "k", "m", "b", "d", "h", "v"). In addition, all gap characters ("-") beyond the first and last base characters of each sequence are replaced by the character "n".

Usage

```
trimEnds(x, min.n.seq = 4)
```

Arguments

x	An object of class DNAbin.
min.n.seq	A numeric giving the required minimum number of sequences having an non-ambiguous base character (a, c, g, t) in the first and last position of the alignment; defaults to 4, which is the minimum number of sequences needed to produce a non-trivial unrooted topology. Can also be given as a fraction.

Value

An object of class DNAbin.

See Also

deleteEmptyCells, deleteGaps

Examples

```
# simple example alignment:
x <- structure(list(nb = 5, seq = c("acaaggtaca", "-caaggtac-",
"acaaggtaca", "aca--gtaca", "-ccaggta--"), nam = LETTERS[1:5]),
.Names = c("nb", "seq", "nam"), class = "alignment")
# convert to DNAbin:
x <- as.DNAbin(x)
# fill missing nucleotides:
x <- trimEnds(x)
# show results:
as.character(x[2, ])
```

unlistFirstLevel *Unlist To First Level Only*

Description

Does the same as `unlist`, but recurses only one level.

Usage

```
unlistFirstLevel(z, use.names = TRUE)
```

Arguments

`z`	A list of lists.
`use.names`	Logical, indicating if element names from the element should be preserved.

write.fas *Write DNA Sequences to File*

Description

Write DNA sequences and morphological data to FASTA, PHYLIP, or NEXUS formatted files.

Usage

```
write.fas(x, file, block.width = FALSE,
    truncate = FALSE, append = FALSE)

write.phy(x, file, block.width = FALSE,
    strict = FALSE)

write.nex(x, file, block.width = 60,
    taxblock = FALSE)
```

Arguments

`x`	an object of class `DNAbin` (usually as matrix, but `write.fas` also accepts lists) or a list of objects of class `DNAbin` (only `write.nex`) or a data frame containing standard (morphological, etc.) data (only `write.nex`).
`file`	a character string giving the filename; a special case is `file = ""`, which causes the file content to be written on the standard output connection (i.e. the console). If `file` is left unspecified (default), the file content is returned as a vector of mode `"character"` and can be used as a building block for more complex data files.
`block.width`	an integer, giving the number of characters per line.

truncate	truncation of taxon names to the number of characters given as a integer, otherwise (default) taxon names will not be changed.
append	logical, if TRUE the sequences will be appended to file (if it exists).
strict	logical, if TRUE the names of the sequences will be truncated to 10 strings.
taxblock	logical, if TRUE, a tax block will be added to the NEXUS file.

Details

write.nex can handle multiple DNA sequence alignments, which are handed over as a list of objects of class DNAbin. Correct matching of the rows in the alignments is cared for automatically, hence the individual alignments can contain different numbers of samples and samples need not be in the same order.

Value

None, except when called with file left unspecified, which causes the file content to be returned as a vector of mode "character". This is particularly useful for constructing special types of input files, e.g. for MrBayes (mrbayes).

Author(s)

Christoph Heibl

References

Maddison, D.R., D.L. Swofford, and W.P. Maddison. 1997. NEXUS: an extensible file format for systematic information. *Syst. Biol.* **46**: 590-621.

See Also

read.fas, read.phy, and read.nex for reading of DNA sequence files.

Examples

```
data(ips.cox1)
data(ips.28S)

## Examples for FASTA files
## ------------------------
write.fas(ips.cox1[1:5, 1:120], block.width = 60)

## Examples for PHYLIP files
## ------------------------
write.phy(ips.cox1[1:5, 1:20], block.width = 40)

## Examples for NEXUS files
## ------------------------
x <- list(cox1 = ips.cox1[1:5, 1:10],
          rna28S = ips.28S[1:5, 1:30])
write.nex(x, block.width = 20)
```

```
# Truncation of taxonnames:
# -------------------------
rownames(ips.cox1)[1] <- "AVeeeeeeeeeeeeeeeeeryLongName"
write.fas(ips.cox1, truncate = 10)

# If truncation leads to identical taxonnames,
# a warning will be issued:
# -------------------------
rownames(ips.cox1)[1:2] <- "AVeeeeeeeeeeeeeeeeeryLongName"
write.fas(ips.cox1, truncate = 10)
```

Index

∗ **datasets**
    ips.16S, 19
    ips.28S, 20
    ips.cox1, 21
    ips.tree, 21
∗ **package**
    ips-package, 3
.Machine, *16*

AAbin, *24*
ace, *33*
aliscore, *3*, 4, *10*, *19*, *24*, *40*, *47*
ape, *3*
aperm, *5*
aperm.DNAbin, 5
apply, *5*

blastn, 6
branching.times, *52*

cbind.DNAbin, *43*, *47*
chronos, *17*
code.simple.gaps, 6, *10*
collapseUnsupportedEdges, 7
combMyTree, 8

del.gaps, *10*
del.miss, 9
deleteEmptyCells, *7*, *54*
deleteEmptyCells *(*EmptyCells*)*, 12
deleteGaps, *7*, 9, *13*, *54*
descendants, 10, *14*, *50*
DNAbin, *5, 6*, *9–12*, *17*, *19–21*, *24, 25*, *38*, *40*, *43, 44*, *53*, *55*
DNAbin2index, 11, *19*

EmptyCells, 12
eoi, 13

fixNodes, 15
forceEqualTipHeights, *8*, 16

gblocks, *3*, *5*, *10*, 17, *24*, *40*, *47*

identifyEmptyCells *(*EmptyCells*)*, 12
index2DNAbin, *11*, 19
ips *(*ips-package*)*, 3
ips-package, 3
ips.16S, 19
ips.28S, 20
ips.cox1, 21
ips.tree, 21
is.ultrametric, *16*

ladderize, *3*, *15, 16*

mafft, *3*, *5*, *19*, 22, *26*, *37*, *40*, *47*
mafft.merge, 24
mrbayes, *3*, 25, *26–28*, *30*, *32*, *56*
mrbayes.lset, 26, *30*, *32*
mrbayes.mcmc, *25*, *27*, 28, *32*
mrbayes.prset, *25*, *27*, *30*, 30
mrca, *14*
mst, *34*
multistate, 32
multistateMCMC, *3*
multistateMCMC *(*multistate*)*, 32
multistateML, *3*
multistateML *(*multistate*)*, 32

neighboringPairs, 34
noi, *11*, *50*
noi *(*eoi*)*, 13
ntip, 34

oi *(*eoi*)*, 13

partitionfinder, 35
pathd8, 36
phylo, *7, 8*, *10*, *13*, *15–17*, *23, 24*, *34*, *36–38*, *41*, *50–53*
phylo2mafft, 37
phylo2mst, 38

pis, 38
prank, *3*, *5*, *19*, *24*, *26*, 39, *47*

raxml, *3*, *26*, 40, *43*
raxml.partitions, *42*, 43
rbeauti, *3*, 44, *49*
rc, 46
read, 47
read.beast, *3*, *16*, *45*, 48, *49*
read.beast.table, *3*, *45*, *49*, 49
read.fas, *3*, *24*, *40*, *53*, *56*
read.mrbayes *(*read.beast*)*, 48
read.nex, *3*, *56*
read.nexus, *16*
read.phy, *3*, *56*
read.starbeast *(*read.beast*)*, 48
read.tree, *16*, *53*
root, *15*
rotate, *3*, *15, 16*

set.seed, *41*
sister, *11*, 50
splitIntoClades, 51

terminalSisters, 51
tipHeights, *17*, 52
traitRate, 52
trimEnds, *7*, *13*, 54

unlistFirstLevel, 55

which, *13*
write.fas, *3*, 55
write.nex, *3*
write.nex *(*write.fas*)*, 55
write.phy, *3*
write.phy *(*write.fas*)*, 55