Package 'ips'

November 10, 2014

Version 0.0-7

Date 2014-11-09

Title Interfaces to Phylogenetic Software in R

Author Christoph Heibl <christoph.heibl@gmx.net>

Maintainer Christoph Heibl <christoph.heibl@gmx.net>

Depends ape, colorspace, XML

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

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-11-10 00:38:44

R topics documented:

`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:	0.0-7
Date:	2014-11-09
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`, `write.nex`, and `write.partitioned.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 imple-

ment 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 phylogentic 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 Maintainer: Christoph Heibl (<christoph.heibl@gmx.net>)

See Also

`ape`

`aliscore` *Masking of Sequence Alignments with ALISCORE*

Description

This function provides a wrapper to **Aliscore**, which can be used remove problematic regions of a sequence alignment.

Usage

```
aliscore(x, gaps = "5state", w = 6, r, t, l, s, o,
        path = "/Applications/Aliscore_v.2.0")
```

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.
`path`	A character string, giving the path to the Aliscore script.

Value

`matrix` of class `"DNAbin"`

Note

This function was developed with ALISCORE version 2.

Author(s)

Christoph Heibl

References

http://zfmk.de/web/Forschung/Abteilungen/AG_Wgele/Software/Aliscore/Download/index.en.html

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 sequence alignment; `gblocks` for another alignment masking algorithm.

---

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

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, cutoff)
```

Arguments

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

Value

An object of class `phylo`.

Author(s)

Christoph Heibl

See Also

`help`

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)
```

`deleteEmptyCells` *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. `deleteEmptyCells` will delete all such rows (taxa) and columns (characters) from a DNA sequence alignment.

Usage

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

Arguments

`DNAbin`	an object of class `DNAbin`.
`nset`	vector of mode character; rows or colums that consist **only** of the characters given in `nset` will be deleted from the alignment.
`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. As a consequence, `nset` cannot be extended by the user and the use of IUPAC ambiguity symbols other than `'N'` is currently not possible.

Value

an object of class `DNAbin`.

Author(s)

Christoph Heibl

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)

# delete those rows and colums
x <- deleteEmptyCells(x)
image(x)
```

---

deleteGaps *Remove Gap/Indel Positions from Alignment*

Description

This function removes 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, nmax = nrow(x) - 4)
```

Arguments

x	an object of class DNAbin.
nmax	an integer number between 0 and nrow(x) - 2, which gives the maximum number of gap characters ("-") that will be tolerated at any given alignment position (column).

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.

Author(s)

Christoph Heibl

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)
```

Arguments

phy	an object of class phylo.
node	an integer giving the number of the internal node.
type	a character string, may be "terminal", "internal", "both", 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)
```

`fixNodes` *Standard Node Numbering in Phylo Objects*

Description

The function (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

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`.

Note

`fixNodes` has been completely rewritten for **ips** version 1.0-0. It should now run absolutely stable and is much quicker. Nevertheless, I recommend checking carefully the results of `fixNodes`, until the function has been tested by a number of users. Then this comment will be removed.

Author(s)

Christoph Heibl

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**.

---

gblocks *Masking of Sequence Alignments with GBLOCKS*

Description

This function 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 = .5, b2 = b1, b3 = ncol(x), b4 = 2, b5 = "a", exec)
```

Arguments

x	a matrix of DNA sequences of classes DNAbin or alignment.
b1	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	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	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	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).
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

`matrix` of class `"DNAbin"`

Note

From `phyloch` version 1.4-80 on, the defaults parameters of `gblocks` have been changed to be least conservative.

`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.

Author(s)

Christoph Heibl

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: `http://molevol.cmima.csic.es/castresana/Gblocks.html`

See Also

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

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 resitory at GenBank on February 8, 2014.

References

The nucleotide database on the NCBI website: http://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 resitory at GenBank on February 8, 2014.

References

The nucleotide database on the NCBI website: http://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 resitory at GenBank on February 8, 2014.

References

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

Examples

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

ips.tree *Ips Phylogeny*

Description

Phylogentic 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 *DNA Sequence Alignment with MAFFT*

Description

This function is a wrapper for MAFFT and can be used for sequence and profile aligning.

Usage

```
mafft(x, y, add, method = "auto", maxiterate = 0, op = 1.53,
    ep = 0.0, gt, options, path, quiet)
```

Arguments

x	An object of class DNAbin.
y	An object of class DNAbin, 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" as well as "retree 1" and "retree 2"for speed-oriented alignment. The default is "auto", which lets MAFFT choose an opproriate 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 recommende if no long indels are expected.
gt	An object of class phylo that is to be used as a guide tree during alignment.
options	A vector of mode character specifying addional arguments to MAFFT, that are not included in mafft such as, e.g., --adjustdirection.
path	A character string indicating the path to the MAFFT executable.
quiet	Logical, if set to TRUE, mafft progress is printed out on the screen.

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 aligment; this is the default in MAFFT; recommended for >200 sequences.

Value

A matrix of class "DNAbin".

Note

mafft was last updated and tested to work with MAFFT 7.032. If you have problems getting the function to work with a newer version of MAFFT, please contact the package maintainer.

Author(s)

Christoph Heibl

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.

http://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.

---

mrbayes *Bayesian MCMC Tree Search with MrBayes*

Description

This function is a wrapper for Bayesian phylogenetic tree search through MrBayes (Ronquist & Huelsenbeck, 2003) with either DNA (mrbayes) or morphological (mrbayes.mixed) data.

Usage

```
mrbayes(x, file = "", nst = 6, rates = "invgamma", ngammacat = 4,
    nruns = 2, ngen = 1e+06, printfreq = 100, samplefreq = 10,
    nchains = 4, savebrlens = "yes", temp = 0.2, burnin = 10,
    contype = "allcompat", run = FALSE)


mrbayes.mixed(x, file, nst = 6, rates = "invgamma", ngammacat = 4, nruns = 2,
ngen = 1e+06, printfreq = 100, samplefreq = 10, nchains = 4,
savebrlens = "yes", temp = 0.2, burnin = 10, contype = "allcompat", run = TRUE)
```

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.
nst	An integer giving the number of rates in the model of sequence evolution.
rates	A character string; allowed are "equal", "gamma", "propinv", "invgamma", and "adgamma"; the default is "equal".
ngammacat	An integer; the number rate categories for the discretized Gamma distribution; the default is 4.

nruns	An integer; the number of runs.
ngen	An integer; the number of states of the MCMC.
printfreq	An integer; the interval between states of the MCMC to be printed on the screen
samplefreq	An integer; the interval between states of the MCMC to be **sampled**.
nchains	An integer; number of Metropolis coupled MCMCs in each run.
savebrlens	Logical; shall branch lengths be saved.
temp	
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).
run	Logical; run = FALSE will only print the NEXUS file, run = TRUE will also start the MCMC runs, if the path argument 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.

Author(s)

Christoph Heibl

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: http://mrbayes.sourceforge.net/.

See Also

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

Examples

```
# DNA sequence data:
# ------------------
data(ips.cox1)
x <- ips.cox1[, 100:140] # tiny alignment

# print NEXUS file with MrBayes block to working directory
# --------------------------------------------------------
mrbayes(x, file = "", ngen = 100, run = FALSE)
```

`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 colums. The first column contains the taxon labels, the second column contains the character states.
`model`	
`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.rdg.ac.uk/Files/BayesTraits-V1.0-Manual.pdf.

Pagel, M., A. Meade, and D. Barker. 2004. Bayesian estiamation 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

---

noi *Identification of MRCAs for Clades*

Description

This function identifies the most recent common ancestor (MRCA) nodes for one or more sets of taxa/tips.

Usage

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

Arguments

phy	an object of class phylo.
group	a vector or list of vectors of mode character specifying the taxon set(s).
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"

Author(s)

Christoph Heibl

See Also

mrca; descendants for the contrary operation to noi.

Examples

```
# molecular phylogeny of Eurasian vipers:
# ----------------------------------------
#data(viperidae)
#gen <- sapply(viperidae$tip.label, function(x) unlist(strsplit(x, "_"))[1])
#tax <- data.frame(genus = gen, species = viperidae$tip.label, row.names = NULL)

# group be a data frame
# ----------------------------------------------------
## .. to be added ..

# 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)
```

`pis` *Number of Potentially-Informative Sites*

Description

This function 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 indeces 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).

Author(s)

Christoph Heibl

Examples

```
# example data:
# -------------
data(ips.16S)

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

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

# indeces 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.

Author(s)

Christoph Heibl

References

http://www.ebi.ac.uk/goldman-srv/prank/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

This function calls **RAxML** (see Reference section) for the maximum likelihood estimation of tree topology and/or branch lengths for a given DNA sequence alignment, 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, f, N, p, b, x, k,
      partitions, outgroup, backbone = NULL,
      file = "fromR", exec, threads)
```

Arguments

DNAbin	a matrix of DNA sequences of class DNAbin.
m	a vector of mode "character" choosing a model of subsitution; currently only GTR models available
f	a vector of mode "character" choosing from the 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 independant 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 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
partitions	a data frame specifying partitions of the alignment.
outgroup	a vector of mode "character" containing the names of the outgroup taxa.
backbone	a phylo object handed as a backbone tree to RAxML.
file	a vector of mode "character" giving a name to 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 executale 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 requiered 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 the 'Software' tab on http://www.exelixis-lab.org/ 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.

Author(s)

Christoph Heibl

References

(in chronolocigal 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 <- NULL # replace by your RAxML path

if ( !is.null(exec) ){

## normal tree search with GTRCAT and GTRGAMMA
tr <- raxml(woodmouse, f = "d", N = 2, p = 1234,
            exec = exec) # -1743.528461
tr <- raxml(woodmouse, m = "GTRGAMMA", f = "d", N = 2, p = 1234,
            exec = exec)

## rapid bootstrap
tr <- raxml(woodmouse, m = "GTRGAMMA",
            f = "a", N = 10, p = 1234, x = 1234,
            exec = exec)

## rapid bootstrap with automatic halt
tr <- raxml(woodmouse, m = "GTRGAMMA",
```

```
            f = "a", N = "autoMRE", p = 1234, x = 1234,
            exec = exec)
}
```

`raxml.partitions` *Partition scheme for RAxML*

Description

Given a set of DNA sequence alignments, `raxml.partitions` creates a data frame with partition bounderies 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

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

Usage

```
rbeauti(s, file, taxonset)
```

Arguments

s	An object 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).
taxonset	A list containing one or more taxon sets.

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 convieniently 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.

Author(s)

Christoph Heibl

References

The BEAST 2 website: http://beast.bio.ed.ac.uk/BEAST_v1.5.x_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)
```

`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 skalars 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.

Author(s)

Christoph Heibl

References

TreeAnnotator: http://beast.bio.ed.ac.uk/TreeAnnotator

Metacomments in NEXUS: http://code.google.com/p/beast-mcmc/wiki/NexusMetacommentFormat

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 TreeAnnotaror-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**.

`read.fas` *Read DNA Sequences*

Description

This functions parse DNA sequences in 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 object of class `DNAbin` and of mode `matrix` if all sequences are of same length or `matrix` if they are not.

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

`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)
```

`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")
```

Arguments

`phy`	an object of class `phylo`.
`node`	an integer giving the number of the node.
`type`	a character string, may be `"terminal"`, `"internal"`, `"both"`, or any unambiguous abbreviation of these.

Value

A vector containing node numbers.

Author(s)

Christoph Heibl

See Also

`descendants`, `noi`

Examples

```
# data
# ----
#data(vipera)
#node <- noi(vipera.ml, c("Vipera_latastei", "Vipera_aspis"))

# get the sister node of 'node'
# -----------------------------
#(sn <- sister(vipera.ml, node, type = "internal"))

# ... and the sister nodes's tips (i.e. the sister clade)
# -------------------------------------------------------
#(sc <- sister(vipera.ml, node, type = "terminal"))

# results graphically:
# --------------------
#plot(vipera.ml)
#nodelabels(text = "15", node = 15, frame = NULL, col = "white", bg = "black")
#nodelabels(text = sn, node = sn, col = "black", bg = "red")
#append2tiplabel(tree, tips = sc, pch = 21, offset = strwidth("."))
```

---

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 detectiong 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

This function trims both ends of a DNA sequence alignment to the first and last alignment positions that contain a minimum number of non-ambiguous characters (`a`, `c`, `g`, `t`). In addition, all gap characters (`"-"`) beyond the first and last non-ambiguous 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.

Value

An object of class `DNAbin`.

Author(s)

Christoph Heibl

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, ])
```

`write.fas` *Write DNA Sequences to File*

Description

These functions write DNA sequences to FASTA, PHYLIP, or NEXUS formatted files.

Usage

```
write.fas(x, file = "", interleave = FALSE, truncate = FALSE, append = FALSE)

write.phy(x, file = "", interleave = FALSE, strict = FALSE)

write.nex(x, file = "", interleave = 60, taxblock = FALSE)
```

Arguments

`x`	a list or matrix of DNA sequences.
`file`	a character string giving the filename; if `file = ""`, the file is written on the standard output connection (i.e. the console).
`interleave`	an integer, giving the number of bases 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.

Value

None.

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

```
## bark beetle COX1 sequences
## --------------------------
data(ips.cox1)
ips.cox1 <- ips.cox1[1:4, 1:120]

## Examples for FASTA files
## ------------------------
write.fas(ips.cox1, interleave = 60)

## Examples for PHYLIP files
## ------------------------
write.phy(ips.cox1, interleave = 40)

## Examples for NEXUS files
## ------------------------
# write nexus file with taxon block
write.nex(ips.cox1, taxblock = TRUE)
# write non-interleaved nexus file without taxon block
write.nex(ips.cox1, interleave = FALSE)

# 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)
```

`write.partitioned.nex` *Write partitioned and commented NEXUS files*

Description

This function takes one or more objects of class `"DNAbin"` and writes them to an interleaved NEXUS file.

Usage

```
write.partitioned.nex(..., file, labels)
```

Arguments

`...`	One or more objects of class `"DNAbin"`
`file`	character string giving a file name
`labels`	vector of mode `"character"`; used to label partitions; its length must match the number of

Author(s)

Christoph Heibl

See Also

`write.nex`, `write.phy`, `write.fas`

Examples

```
# load sequence data and split in three parts
# -------------------------------------------
data(woodmouse)
w1 <- woodmouse[, 1:250]
w2 <- woodmouse[, 251:750]
w3 <- woodmouse[, 750:965]

# write nexus file with taxon block
# ---------------------------------
markers <- c("gene1", "gene2", "gene3")
write.partitioned.nex(w1, w2, w3, labels = markers)
```

Index