Package: corHMM (via r-universe)

October 21, 2024

Version 2.10

Date 2024-08-21

Title Hidden Markov Models of Character Evolution

Maintainer Jeremy Beaulieu <jmbeauli@uark.edu>

Depends ape, nloptr, GenSA

Suggests testthat, knitr, rmarkdown

Imports expm, numDeriv, corpcor, MASS, nnet, phangorn, parallel, viridis, Rmpfr, igraph, phytools, lhs

Description Fits hidden Markov models of discrete character evolution which allow different transition rate classes on different portions of a phylogeny. Beaulieu et al (2013) <doi:10.1093/sysbio/syt034>.

License GPL (>= 2)

VignetteBuilder knitr

Repository https://thej022214.r-universe.dev

RemoteUrl https://github.com/thej022214/corhmm

RemoteRef HEAD

RemoteSha aa7efef8fbb6b74599c64ca50537b7480f335d07

Contents

---

ancRECON *Ancestral state reconstruction*

Description

Infers ancestral states based on a set of model parameters

Usage

```
ancRECON(phy,data, p, method=c("joint", "marginal", "scaled"),
rate.cat, ntraits=NULL, rate.mat=NULL,
model="ARD", root.p=NULL, get.likelihood=FALSE, get.tip.states = FALSE,
tip.fog=NULL, get.info=FALSE, collapse = TRUE)
```

Arguments

phy	a phylogenetic tree, in ape "phylo" format.
data	a data matrix containing species information (see Details).
p	a vector of transition rates to be used to estimate ancestral states.
method	method used to calculate ancestral states at internal nodes. Can be one of: "joint", "marginal", or "scaled" (see Details).
rate.cat	specifies the number of rate categories in the HRM.
ntraits	currently, this is automaticall detected and can always be set to NULL.
rate.mat	a user-supplied rate matrix index of parameters to be optimized.
model	specifies the underlying model if a rate.mat is not provided ("ER", SYM", or "ARD").
root.p	a vector used to fix the probabilities at the root, but "yang" and "maddfitz" can also be supplied to use the method of Yang (2006) and FitzJohn et al (2009) respectively (see details).
get.likelihood	a logical indicating whether to obtain the likelihood of the rates and states. The default is FALSE.

`get.tip.states` a logical indicating whether just tip reconstructions should be output. The default is `FALSE`.

`tip.fog` provides the probability that an observed state is not actually in the state it is assigned to the reconstruction algorithm. These values are assumed either optimized in "corHMM" or supplied by the user. `tip.fog=NULL` when using this as a standalone function.

`get.info` Whether to return information measures at nodes (Boyko and Beaulieu 2021).

`collapse` a boolean indicating whether to collapse multiple character combinations into only the observed states. For example, if true a two character dataset contained (0,0), (1,0), and (1,1), this would be collapsed into 1,2,3. However, if set to false it would 1,2,4. In combination with a custom rate matrix this allows for the estimation of transitions between the unobserved character combination. The default is `TRUE`

.

Details

This is a stand alone function for computing the marginal, joint, or scaled likelihoods of internal nodes for a given set of transition rates. Like all other functions contained in corHMM, the tree does not have to be bifurcating in order for analyses to be carried out. IMPORTANT: If the corDISC, corHMM, and rayDISC functions are used they automatically provide a tree with the likeliest states as internal node labels. This function is intended for circumstances where the user would like to reconstruct states based on rates estimated elsewhere (e.g. BayesTraits, Mesquite, `ape`).

The algorithm based on Pupko et al. (2000, 2002) is used to calculate the `joint` estimates of ancestral states. The `marginal` method was originally implemented based on a description of an algorithm by Yang (2006). The basic idea is that the tree is rerooted on each internal node, with the marginal likelihood being the probabilities of observing the tips states given that the focal node is the root. However, this takes a ton of time as the number of nodes increase. But, importantly, this does not work easily when the model contains asymmetric rates. Here, we use the same dynamic programming algorithm as Mesquite (Maddison and Maddison, 2011), which is time linear with the number of species and calculates the marginal probability at a node using an additional up and down pass of the tree. If `scaled`, the function uses the same algorithm from ace(). Note that the `scaled` method of ace() is simply the conditional likelihoods of observing everything at or above the focal node and these should NOT be used for ancestral state estimation.

The user can fix the root state probabilities by supplying a vector to `root.p`. For example, in the two trait case, if the hypothesis is that the root is 00, then the root vector would be `root.p=c(1,0,0,0)` for state combinations 00, 01, 10, and 11, respectively. If analyzing a binary or multistate character, the order of root.p is the same order as the traits – e.g., for states 1, 2, 3, a `root.p=c(0,1,0)` would fix the root to be in state 2. If the user supplies the flag `root.p`="yang", then the estimated transition rates are used to set the weights at the root (see pg. 124 Yang 2006), whereas specifying `root.p`="maddfitz" employs the same procedure described by Maddison et al. (2007) and FitzJohn et al. (2009). Note that the default `root.p=NULL` assumes equal weighting among all possible states.

Setting get.likelihood=TRUE will provide the user the joint likelihood of the rates and states.

Value

`$lik.tip.states`
: A matrix of the reconstructed tip values. If the number of rate.cats is greater than 2 then the probability that each observed state is in a particular hidden state is given.

`$lik.anc.states`
: For `joint`, a vector of likeliest states at internal nodes and tips. For either `marginal` or `$scaled`, a matrix of the probabilities of each state for each internal node are returned.

`$info.anc.states`
: A vector containing the amount of information (in bits) that the tip states and model gives to each node. See Boyko and Beaulieu (2021).

Author(s)

Jeremy M. Beaulieu and Jeffrey C. Oliver

References

FitzJohn, R.G., W.P. Maddison, and S.P. Otto. 2009. Estimating trait-dependent speciation and extinction rates from incompletely resolved phylogenies. Systematic Biology 58:595-611.

Maddison, W.P. and D.R. Maddison. 2011. Mesquite: a modular system for evolutionary analysis. Version 2.75 http://mesquiteproject.org

Pupko, T., I. Pe'er, R. Shamir, and D. Graur. 2000. A fast algorithm for joint reconstruction of ancestral amino-acid sequences. Molecular Biology and Evolution 17:890-896.

Pupko, T., I. Pe'er, D. Graur, M. Hasegawa, and N Friedman N. 2002. A branch-and-bound algorithm for the inference of ancestral amino-acid sequences when the replacement rate varies among sites: application to the evolution of five gene families. Bioinformatics 18:1116-1123.

Yang, Z. 2006. Computational Molecular Evolution. London:Oxford.

Boyko, J. D., and J. M. Beaulieu. 2021. Generalized hidden Markov models for phylogenetic comparative datasets. Methods in Ecology and Evolution 12:468-478.

Examples

```
data(primates)
phy <- multi2di(primates[[1]])
data <- primates[[2]]
MK_3state <- corHMM(phy = phy, data = data, rate.cat = 1)

# # one way to get the parameters from your corHMM object in the correct order
p <- sapply(1:max(MK_3state$index.mat, na.rm = TRUE), function(x)
 na.omit(c(MK_3state$solution))[na.omit(c(MK_3state$index.mat) == x)][1])

# using custom params
states_1 <- ancRECON(phy = phy, data = MK_3state$data, p = p, method = "marginal",
 rate.cat <- MK_3state$rate.cat, ntraits = NULL, rate.mat = MK_3state$index.mat,
 root.p = MK_3state$root.p)
```

ancRECON_slice *Ancestral state reconstruction for a particular time*

Description

Infers marginal ancestral states based on a set of model parameters at a particular time

Usage

```
ancRECON_slice(corhmm.obj, time_slice, collapse=TRUE, ncores = 1)
```

Arguments

corhmm.obj	a corhmm object which is the output of the main corhmm function.
time_slice	a vector of times to reconstruct (present = 0, root = max(branching.times(phy)))
collapse	set collapse to be the same as it was during the corhmm run.
ncores	number of cores to use during parallelization.

Details

This is a stand alone function for computing the marginal likelihoods at particular points along a branch for a given set of transition rates. ancRECON has the technical details of ancestral state reconstruction if you're interested. The time_slice argument will specify a time point where marginal reconstructions will be produced. You can imagine the time slice intersecting the branches of the phylogeny and doing a reconstruction there rather than at nodes as is typically done.

Value

a data.frame. Each row of time_slice is the time period that was specified. node is the closest tipward node to the slice on a particular branch. position is the amount of time (rootward) from the nearest node. Remaining columns are the marginal probabilities of each state at that particular node. There is also a plotting function that I don't currently export because it's unfinished. The example shows how to access it.

Author(s)

James D. Boyko

References

Yang, Z. 2006. Computational Molecular Evolution. London:Oxford.

Boyko, J. D., and J. M. Beaulieu. 2021. Generalized hidden Markov models for phylogenetic comparative datasets. Methods in Ecology and Evolution 12:468-478.

Examples

```
# library(corHMM)
# library(viridis)
# data(primates)
# phy <- multi2di(primates[[1]])
# data <- primates[[2]]
# corhmm.obj <- corHMM(phy = phy, data = data, rate.cat = 1)
# test <- ancRECON_slice(corhmm.obj, time_slice = c(1, 10, 20, 30, 40, 49),
#         collapse = TRUE, ncores = 4)
# corHMM:::plot_slice_recon(phy, test, col = viridis(3))
```

`ComputeCI` *Compute confidence around rate estimates*

Description

corHMM gives a single point estimate of rates, but this point estimate could be very uncertain. A traditional way to evaluate confidence is to vary each parameter until the log likelihood gets 2 units worse, while holding other parameters at their maximum likelihood estimates. That's fine and fast, but it can be misled by ridges. So, instead, we want all values that lead to a likelihood within two log likelihood units of the best. The range will be at least as wide as the univariate estimates but probably much larger.

Usage

```
ComputeCI(corhmm.object, desired.delta = 2, n.points=5000, verbose=TRUE,
print_freq=50, ...)
```

Arguments

`corhmm.object`	The result of a corHMM search.
`desired.delta`	How many log likelihood units to deviate from the optimal likelihood.
`n.points`	How many points to use.
`print_freq`	Output progress every print_freq steps.
`verbose`	Other arguments to pass into the likelihood function.
`...`	further arguments to be passed dentist.

Details

The algorithm tunes: if it is moving too far away from the desired likelihoods, it will decrease the proposal width; if it staying in areas better than the desired likelihood, it will increase the proposal width. It will also expand the proposal width for parameters where the extreme values still appear good enough to try to find out the full range for these values.

In general, the idea of this is not to give you a pleasingly narrow range of possible values – it is to try to find the actual uncertainty, including finding any ridges that would not be seen in univariate space.

Value

A dentist object containing results, the data.frame of negative log likelihoods and the parameters associated with them; acceptances, the vector of whether a proposed move was accepted each step; best_neglnL, the best value passed into the analysis; delta, the desired offset; all_ranges, a summary of the results.

Author(s)

Brian O'Meara (see also the R package Dentist) & exported by James D. Boyko

Examples

```
# data(primates)
# phy <- multi2di(primates[[1]])
# data <- primates[[2]]
# MK_3state <- corHMM(phy = phy, data = data, rate.cat = 1)
# confidence_results <- ComputeCI(MK_3state, desired.delta = 2, 200)
# print(confidence_results)
# plot.dentist(confidence_results)
```

ConvertPhangornReconstructions

Convert phangorn reconstruction to a vector

Description

Converts a character reconstruction from phangorn into a vector of tip and node states. Nodes where there are equal weights among states, ties are broken at random.

Usage

```
ConvertPhangornReconstructions(x, site = 1, best = TRUE)
```

Arguments

x	The phyDat object that contains a character reconstruction from phangorn
site	The character number to convert into a vector
best	A logical indicating whether the state that maximizes some function (likelihood, parsimony, etc.) is to be returned.

Details

Creates a vector that contains the best tips and node state from a phangorn reconstruction.

corDISC *Correlated evolution binary traits*

Description

Fits a model of correlated evolution between two or three binary traits

Usage

```
corDISC(phy,data, ntraits=2, rate.mat=NULL, model=c("ER","SYM","ARD"),
node.states=c("joint", "marginal", "scaled", "none"), lewis.asc.bias=FALSE, p=NULL,
root.p=NULL, ip=NULL, lb=0, ub=100, diagn=FALSE)
```

Arguments

phy	a phylogenetic tree, in ape "phylo" format.
data	a data matrix containing species information (see Details).
ntraits	specifies the number of traits to be included in the analysis.
rate.mat	a user-supplied rate matrix index of parameters to be optimized.
model	specifies the underlying model.
node.states	method used to calculate ancestral states at internal nodes (see Details).
lewis.asc.bias	a logical indicating whether the ascertainment bias correction of Lewis et al. 2001 should be used. The default is FALSE.
p	a vector of transition rates. Allows the user to calculate the likelihood given a specified set of parameter values to specified as fixed and calculate the likelihood.
root.p	a vector used to fix the probabilities at the root, but "yang" and "maddfitz" can also be supplied to use the method of Yang (2006) and FitzJohn et al (2009) respectively (see details).
ip	initial values used for the likelihood search. Can be a single value or a vector of unique values for each parameter. The default is ip=1.
lb	lower bound for the likelihood search. The default is lb=0.
ub	upper bound for the likelihood search. The default is ub=100.
diagn	logical indicating whether diagnostic tests should be performed. The default is FALSE.

Details

__THIS FUNCTION IS NO LONGER NECESSARY AS IT IS NOW ENTIRELY SUBSUMED WITHIN__ corHMM (see _Generalized corHMM_ vignette). But we still provide it for those that are more comfortable using it than exploring the new corHMM function. As before, corDISC takes a tree and a trait file and estimates transition rates and ancestral states for two or three binary characters (see Pagel 1994). Note, however, that rayDISC can be used to evaluate the same models as

in corDISC, with the major difference being that, with rayDISC, the rate matrix would have to be manipulated using `rate.mat.maker` in order to remove parameters associated with dual transitions. With corDISC, the input phylogeny need not be bifurcating as the algorithm is implemented to handle multifucations. Polytomies are allowed by generalizing Felsenstein's (1981) pruning algorithm to be the product of the probability of observing the tip states of n descendant nodes, rather than two, as in the completely bifurcating case. For the trait file, the first column of the trait file must contain the species labels to match to the tree, with the second column onwards corresponding to the binary traits of interest.

The user can fix the root state probabilities by supplying a vector to `root.p`. For example, in the two trait case, if the hypothesis is that the root is 00, then the root vector would be `root.p=c(1,0,0,0)` for state combinations 00, 01, 10, and 11, respectively. If the user supplies the flag `root.p=`"yang", then the estimated transition rates are used to set the weights at the root (see pg. 124 Yang 2006), whereas specifying `root.p=`"maddfitz" employs the same procedure described by Maddison et al. (2007) and FitzJohn et al. (2009). Note that the default `root.p=NULL` assumes equal weighting among all possible states.

We also note that scoring information that is missing for a species can be incorporated in the analysis by including an NA for that particular trait. corDISC will then set the trait vector so that the tip vector will reflect the probabilities that are compatible with our observations. For example, if the scoring for trait 1 is missing, but trait 2 is scored as 0, then the tip vector would be (1,0,1,0), for state combinations 00, 01, 10, and 11 respectively, given our observation that trait 2 is scored 0 (for a good discussion see Felsenstein 2004, pg. 255).

Value

`corDISC` returns an object of class `corDISC`. This is a list with elements:

`$loglik`	the maximum negative log-likelihood.
`$AIC`	Akaike information criterion.
`$AICc`	Akaike information criterion corrected for sample size.
`$ntraits`	The number of traits specified.
`$solution`	a matrix containing the maximum likelihood estimates of the transition rates.
`$solution.se`	a matrix containing the approximate standard errors of the transition rates. The standard error is calculated as the square root of the diagonal of the inverse of the Hessian matrix.
`$index.mat`	The indices of the parameters being estimated are returned. The numbers correspond to the row in the `eigvect` and can useful for identifying the parameters that are causing the objective function to be at a saddlepoint.
`$lewis.asc.bias`	The setting describing whether or not the Lewis ascertainment bias correction was used.
`$opts`	Internal settings of the likelihood search
`$data`	User-supplied dataset.
`$phy`	User-supplied tree.
`$states`	The likeliest states at each internal node.
`$tip.states`	NULL

`$iterations`	The number of iterations used by the optimization routine.
`$eigval`	The eigenvalues from the decomposition of the Hessian of the likelihood function. If any `eigval<0` then one or more parameters were not optimized during the likelihood search
`$eigvect`	The eigenvectors from the decomposition of the Hessian of the likelihood function is returned

Author(s)

Jeremy M. Beaulieu

References

Beaulieu J.M., and M.J. Donoghue 2013. Fruit evolution and diversification in campanulid angiosperms. Evolution, 67:3132-3144.

Felsenstein, J. 1981. A likelihood approach to character weighting and what it tells us about parsimony and compatibility. Biological Journal of the Linnean Society 16: 183-196.

Felsenstein J. 2004. Inferring phylogenies. Sunderland MA: Sinauer Associates.

FitzJohn, R.G., W.P. Maddison, and S.P. Otto. 2009. Estimating trait-dependent speciation and extinction rates from incompletely resolved phylogenies. Systematic Biology 58:595-611.

Lewis, P.O. 2001. A likelihood approach to estimating phylogeny from discrete morphological character data. Systematic Biology 50:913-925.

Maddison, W.P., P.E. Midford, and S.P. Otto. 2007. Estimating a binary characters effect on speciation and extinction. Systematic Biology 56:701-710.

Pagel, M. 1994. Detecting correlated evolution on phylogenies: a general method for the comparative analysis of discrete characters. Proceedings of the Royal Society, B. 255:37-45.

`corHMM` *Hidden Rates Model*

Description

Estimates hidden rates underlying the evolution of any number of multi-state characters

Usage

```
corHMM(phy, data, rate.cat, rate.mat=NULL, model = "ARD",
node.states = "marginal", fixed.nodes=FALSE, p=NULL, root.p="yang",
tip.fog=NULL, ip=NULL, fog.ip=0.01, nstarts=0, n.cores=1,
get.tip.states = FALSE, lewis.asc.bias = FALSE, collapse = TRUE,
lower.bound = 1e-9, upper.bound = 100, opts=NULL)
```

Arguments

phy	a phylogenetic tree, in ape "phylo" format.
data	a data.frame containing species information. The first column must be species names matching the phylogeny. Additional columns contain discrete character data. All columns are converted to class factor unless already is.factor. Order of states is determined by the levels of column. Unobserved levels may be modeled if collapse=FALSE.
rate.cat	specifies the number of hidden rate categories (if 1, there are no hidden states, see Details).
rate.mat	a user-supplied matrix containing indexes of parameters to be optimized (see Details).
model	One of "ARD", "SYM", or "ER". ARD: all rates differ. SYM: rates between any two states do not differ. ER: all rates are equal.
node.states	method used to calculate ancestral states at internal nodes (see Details).
fixed.nodes	specifies that states for nodes in the phylogeny are assumed fixed. These are supplied as node labels in the "phylo" object.
p	a vector of transition rates. Allows the user to calculate the likelihood given a specified set of parameter values to specified as fixed and calculate the likelihood (see Details).
root.p	a vector used to fix the probabilities at the root, but "yang" and "maddfitz" can also be supplied to use the method of Yang (2006) and FitzJohn et al (2009) respectively (see details).
tip.fog	a fixed value or vector of free parameters to estimate the probability that an observed state is not actually in the state it is assigned. By default tip.fog=NULL. To estimate tip fog probabilities a vector of integers designating the number of free parameters is needed as input. See vignette for details on how to carry this out.
ip	initial values used for the likelihood search. Can be a single value or a vector of unique values for each parameter. The default is ip=1.
fog.ip	initial values used for the likelihood search of tip fog. It is always a single value. The default is fog.ip=0.01.
nstarts	the number of random restarts to be performed. The default is nstarts=0.
n.cores	the number of processor cores to spread out the random restarts.
get.tip.states	a boolean indicating whether tip reconstructions should be output. The default is FALSE.
lewis.asc.bias	a boolean indicating whether to correct for observing a dataset that is not univariate. The default is FALSE
.	
collapse	a boolean indicating whether to collapse multiple character combinations into only the observed states. For example, if true a two character dataset contained (0,0), (1,0), and (1,1), this would be collapsed into 1,2,3. However, if set to false it would 1,2,4. In combination with a custom rate matrix this allows for the estimation of transitions between the unobserved character combination. The default is TRUE

.

lower.bound	lower bound for the likelihood search. The default is lower.bound=1e-9.
upper.bound	upper bound for the likelihood search. The default is upper.bound=100.
opts	options to pass to nloptr. default is NULL.

Details

This function takes a tree and a trait file and estimates transition rates and ancestral states for any number of discrete characters using a Markov model with or without "hidden" states. Users are advised to read the _Generalized corHMM_ vignette for details on how to make full use of corHMM's new functionality. In general, these models describe evolution as discrete transitions between observed states. If rate.cat > 1, then the model is a hidden Markov model (HMM; also known as a hidden rates model (HRM)). The HRM is a generalization of the covarion model that allows different rate classes to be treated as "hidden" states. Essentially a hidden Markov model allows for multiple processes to describe the evolution of your observed character. This could be another (hidden) state or a large group of them. Regardless of the reason, an HMM is saying that not all observed characters are expected to act the same way.

The first column of the input data must be species names (as in the previous version), but there can be any number of data columns. If your dataset does have 2 or more columns of trait information, each column is taken to describe a separate character. The separation of character and state is an important one because corHMM will automatically remove dual transitions from your model. For example, say you had 3 characters each with 2 states (0 or 1), but only three of these combinations were ever observed 0_0_1, 0_1_0, or 1_0_0. With dual transitions disallowed, it is impossible to move between these combinations because it would mean simultaneously losing and gaining a state (0_0_1 -> 0_0_0 -> 0_1_0 in one step.) One way around this is to provide a custom rate matrix to corHMM where transitions are allowed between these states. However, this is also a case where it would seem appropriate to code the data as a single character with 3 states.

Ambiguities (polymorphic taxa or taxa missing data) are assigned likelihoods following Felsenstein (2004, p. 255). Taxa with missing data are coded "?" with all states observed at a tip. Polymorphic taxa are coded with states separated by an "&". For example, if a trait has four states and taxonA is observed to be in state 1 and 3, the character would be coded as "1&3". corHMM then uses this information to assign a likelihood of 1.0 to both states. Missing data are treated as ambiguous for all states, thus all states for taxa missing data are assigned a likelihood of 1.0. For example, for a four-state character (i.e. DNA), a taxon missing data will have likelihoods of all four states equal to 1.0 [e.g. L(A)=1.0, L(C)=1.0, L(G)=1.0, L(T)=1.0].

Rate matrices are usually generated via the specified model, but custom matrices can be used by supplying rate.mat. Similar to the output of model, the required format is a matrix of dimensions nTraitStates x nTraitStates, in which each transition is given an index value. Transitions with the same value will be optimised to have the same rate, so e.g. a vector with all 1's would correspond to the ER model. Transitions which are not allowed (e.g. representing two changes at once, like 00 -> 11) or not possible (e.g. the diagonal) can be set as 0 or NA.

If a user wants to calculate the likelihood for a fixed set of transition rates, without optimising them, a vector p of those values has to be provided. The positon of those values in the vector should correspond the the index numbers in rate.mat, if provided, or to those resulting from the specified model. If for any reason, any index numbers are not present in rate.mat, those positions in the vector will have to be populated with a placeholder (any number or NA will do), to prevent rates from shifting when being assigned to the transitions.

The likelihood function is maximized using the bounded subplex optimization routine implemented in the R package `nloptr`, which provides a common interface to NLopt, an open-source library for nonlinear optimization. In the former case, however, it is recommended that `nstarts` is set to a large value (e.g. 100) to ensure that the maximum likelihood solution is found. Users can set `n.cores` to parse the random restarts onto multiple processors.

The user can fix the root state probabilities by supplying a vector to `root.p`. For example, if the hypothesis is that the root is 0_S in a model with two hidden rates (their categories being denoted "S" and "F" respectively), then the root vector would be `root.p=c(1,0,0,0)` for state combinations 0_S, 1_S, 0_F, and 1_F, respectively. If the user supplies the flag `root.p=`"NULL", then there is equal weighting among all possible states in the model. If the user supplies the flag `root.p=`"yang", then the estimated transition rates are used to set the weights at the root (see pg. 124 Yang 2006), whereas specifying `root.p=`"maddfitz" employs the same procedure described by Maddison et al. (2007) and FitzJohn et al. (2009). Note that the default `root.p="yang"`.

Ancestral states can be estimated using marginal, joint, scaled, or none approaches. Marginal gives the likelihood of state at each node, integrating over the states at other nodes. Joint gives the optimal state at each node for the entire tree at once (it can only return the most likely state, i.e. it is not a probability like the marginal reconstruction). Scaled is included for compatibility with ape's ace() function. None suppresses calculation of ancestral states, which can dramatically speed up calculations if you're comparing models but make plotting difficult.

Value

`corHMM` returns an object of class `corHMM`. This is a list with elements:

`$loglik`	the maximum negative log-likelihood.
`$AIC`	Akaike information criterion.
`$AICc`	Akaike information criterion corrected for sample size.
`$rate.cat`	The number of rate categories specified.
`$solution`	a matrix containing the maximum likelihood estimates of the transition rates. Note that the rate classes are ordered from slowest (R1) to fastest (Rn) with respect to state 0.
`$index.mat`	The indices of the parameters being estimated are returned. This also is a way to allow the estimation of transition rates for parameters not oberved in the dataset. Say you have 2 traits X and Y, where the combinations 00, 01, and 11 are observed (10 is not). A 4 by 4 index matrix could be used to force 10 into the model.
`$data`	User-supplied dataset.
`$data.legend`	User-supplied dataset with an extra column of trait values corresponding to how corHMM calls the user data.
`$phy`	User-supplied tree.
`$states`	The likeliest states at each internal node. The state and rates reconstructed at internal nodes are in the order of the column headings of the rates matrix.
`$tip.states`	The likeliest state at each tip. The state and rates reconstructed at the tips are in the order of the column headings of the rates matrix.
`$states.info`	a vector containing the amount of information (in bits) that the tip states and model gives to each node.

$iterations	The number of iterations used by the optimization routine.
$root.p	The root prior used in model estimation.
pen.type	The penalty type used in model estimation.
lambda	The hyperparameter that adjusts penalty severity used in model estimation.

Author(s)

Jeremy M. Beaulieu and James D. Boyko

References

Beaulieu J.M., B.C. O'Meara, and M.J. Donoghue. 2013. Identifying hidden rate changes in the evolution of a binary morphological character: the evolution of plant habit in campanulid angiosperms. Systematic Biology 62:725-737.

Boyko, J. D., and J. M. Beaulieu. 2021. Generalized hidden Markov models for phylogenetic comparative datasets. Methods in Ecology and Evolution 12:468-478.

Felsenstein, J. 1981. A likelihood approach to character weighting and what it tells us about parsimony and compatibility. Biological Journal of the Linnean Society 16: 183-196.

Felsenstein J. 2004. Inferring phylogenies. Sunderland MA: Sinauer Associates.

FitzJohn, R.G., W.P. Maddison, and S.P. Otto. 2009. Estimating trait-dependent speciation and extinction rates from incompletely resolved phylogenies. Systematic Biology 58:595-611.

Maddison, W.P., P.E. Midford, and S.P. Otto. 2007. Estimating a binary characters effect on speciation and extinction. Systematic Biology 56:701-710.

Pagel, M. 1994. Detecting correlated evolution on phylogenies: a gneeral method for the comparative analysis of discrete characters. Proc. R. Soc. Lond. B 255:37-45.

Yang, Z. 2006. Computational Molecular Evolution. Oxford Press:London.

Examples

```
data(primates)
phy <- multi2di(primates[[1]])
data <- primates[[2]]
MK_3state <- corHMM(phy = phy, data = data, rate.cat = 1)
MK_3state


### custom rate.mat and user supplied transition rates not to be optimised

# this is the rate matrix used in the above example, which represents the ARD
#model with four different rates for all four possible transitions
MK_3state$index.mat

# now we specify our own one where the transitions in and out of state 3 are
#forced to be the same:
rate.mat <- matrix(c(NA, 1, NA, 2, NA, 3, NA, 3, NA), 3, 3)

# we supply the fixed transition rates for which we want the likelihood
#calculated. Since we optimise 3 states in the above matrix, we have to
```

```
#give a vector with three rates, which will be applied to the transition
#with the corresponding rate above (e.g. the transition 2 -> 1 happens at
#rate 0.05)
p <- c(0.05, 0.02, 0.03)

# run
MK_3state.fixed <- corHMM(phy = phy, data = data, rate.cat = 1, rate.mat = rate.mat, p = p)
MK_3state.fixed
```

---

corHMMDredge *Automatic search for optimal discrete character model with regularization with Penalization Options*

Description

corHMMDredge fits a series of hidden Markov models (HMMs) to a given phylogenetic tree and discrete character data to automatically find the optimal model structure. It offers additional options for penalization and optimization compared to the standard corHMM function.

Usage

```
corHMMDredge(phy, data, max.rate.cat, root.p="maddfitz", pen.type = "l1", lambda = 1, drop.par = TRUE, c
```

Arguments

phy	An object of class phylo, representing the phylogenetic tree.
data	A data frame containing character states for the tips of the phylogeny. The first column should match the tip labels in phy, and the second onwards column should contain the observed states.
max.rate.cat	The maximum number of rate categories to try.
root.p	A vector of probabilities for the root states or a method to estimate them. The default is the "maddfitz" method.
pen.type	The type of penalization applied to the model. Options include "l1", "l2", "er", and "unreg". See Details.
lambda	A hyper-parameter that adjusts the severity of the penalty, ranging from 0 (no regularization) to 1 (full penalization). Default is 1.
drop.par	Logical. Whether to drop parameters during optimization based on a threshold. Default is TRUE.
drop.threshold	A numeric value determining the threshold below which parameters should be dropped. Default is 1e-7.
info.threshold	A numeric value specifying the threshold for the amount of information required for parameter estimation. Default is 2.
criterion	The model selection criterion to use. Options are "AIC", "AICc", or "BIC". Default is "AIC".

merge.params	Logical. Whether to merge similar parameters during the model search. Default is TRUE.
merge.threshold	A numeric threshold to determine when parameters should be merged. Default is 0.
rate.mat	A user-supplied matrix containing indexes of parameters to be optimized. If NULL, an all-rates-different model is estimated.
node.states	A method for estimating node states. Options include "marginal", "joint", and "none".
fixed.nodes	Logical. Specifies whether the states for nodes in the phylogeny are assumed fixed. These are supplied as node labels in the phylo object. Default is FALSE.
ip	Initial values used for the likelihood search. Can be a single value or a vector of unique values for each parameter. The default is ip = 1.
nstarts	The number of random restarts to be performed. Default is nstarts = 0.
n.cores	The number of processor cores to spread out the random restarts. Default is n.cores = 1.
get.tip.states	Logical. Indicates whether tip reconstructions should be output. Default is FALSE.
lewis.asc.bias	Logical. Indicates whether to correct for observing a dataset that is not univariate. Default is FALSE.
collapse	Logical. Indicates whether to collapse branches with no variation in states. Default is FALSE.
lower.bound	The lower bound for the rate parameters during optimization. Default is 1e-10.
upper.bound	The upper bound for the rate parameters during optimization. Default is 100.
opts	A list of options to pass to nloptr for controlling optimization behavior.
verbose	Logical. If TRUE, detailed messages about the model fitting process will be printed. Default is TRUE.
p	A vector of transition rates. Allows the user to calculate the likelihood given a specified set of parameter values to be fixed and calculate the likelihood.
rate.cat	An integer specifying the number of rate categories in the model. Only useful if fitting a fixed value p.
grad	Logical. If TRUE, numerical gradient-based optimization will be used. Default is FALSE. This is useful for highly parameterized models, but because it is a numerical gradient it is slow.

Details

corHMMDredge will automatically search different model structures dropping parameters which are estimated near 0 and/or equating parameter values which are near one another. This can be combined with a regularization term (see below) to encrouage lower rate values and thus lead to more parameters being dropped. It will do this iteratively until a stopping criterion is met. The stopping criteria is currently a dAIC of 2, meaning if the next step has made the model worse as indicated by dAIC > 2, the dredge will stop. No model averaging is conducted and only the best model should be used from a dredge search. I explain this in more detail in Boyko (2024), but by dredging we

are not specifying a model set ourselves of distinct hypotheses. Model averaging is useful in that case, but not in the dredge case, because each model as it relates to a hypothesis provides unique information about the system, but the dredge model fits can be very similar to one another, differing in only one or two parameters. These do not really provide unique information and are just minor variations of essentially the same model.

There are 3 penalty types available for users (though this may be expanded in the future). They are `l1`, `l2`, and `er`. The first two penalty types are analagous to lasso and ridge regression. Whereas the `er` penalization is analagous to Zhou et al.'s (2024) McLasso and penalizes the distance between rate values (i.e., it prefers rate matrices that are closer to an equal-rates model).

Under an `l1` regularization scheme, the likelihood is penalized by the mean transition rate. The mean is used instead of the sum because a sum would overly penalize more complex models by virtue of them having more parameters. This leads to scenarios where simpler models have much better likelihoods than more complex models

Under an `l2` regularization scheme, the likleihood is penalized by the squared mean transition rate.

Under an `er` regularization scheme, the likleihood is penalized by the average distance between parameters. If all the parameters were the same (an equal rates model), the penalty would be 0.

A user can also set `pen.type = 'unreg'` for an unpenalized likelihood which would be identical to the original implementation of `corHMM`. More work needs to be done to determine when to use each type of penalization, but generally using any penalization will be good for smaller datasets which tend to be high variance. `l2` is the most aggresive penalization, shrinking paramaters more quickly than other methods and leading to more dropped (and potentially finding more unecessary) parameters. `er` is the most similar to an unregularized model as it does not necessarily penalize high parameter values. It will however penalize a model that has one parameter that is much higher than the rest unless there is significant evidence that this outlier parameter is needed. In practice, `er`, behaves very similarly to unregularized models. `l1` regularization is an intermediate penalization between `l2` and `er`.

The `grad` option employs a numerical gradient for the optimization. This is a particularly inefficient way to find a gradient as it will require at least `k` iterations per likelihood calculation. However, I have found that this improves the search stability and speed as the number of iterations is greatly reduced when a gradient is provided. This is also important in cases where there are a lot of parameters (`k` is large). In these cases the parameter space is so highly dimensional that many search algorithms struggle. In the future I hope to implement a more efficient gradient calculation and combine a gradient based local optimizaiton with a global optimization scheme.

*Note: Many details of `corHMMDredgeBase` and `corHMM` are the same and will not be repeated here. If an arguement is unclear check the Details section of `corHMM`. The focus of these details will be on the unique aspects of the dredge fitting approach.

Value

`fit_set` returns an object of class `corhmm.dredge`. This is a list of the models being fit. Each element of that list is of class `corhmm`.

Author(s)

James D. Boyko

References

Boyko, J. D. 2024. Automatic Discovery of Optimal Discrete Character Models through Regularization. In prep.

Zhou, Y., Gao, M., Chen, Y., Shi, X., 2024. Adaptive Penalized Likelihood method for Markov Chains.

Examples

```
data(primates)
phy <- multi2di(primates[[1]])
data <- primates[[2]]
dredge_fits <- corHMMDredge(phy = phy, data = data, max.rate.cat = 1, pen.type = "l1", root.p = "maddfitz", lambda =
getModelTable(dredge_fits)
```

`corHMMDredgeBase` *Hidden Rates Model with regularization with Penalization Options*

Description

`corHMMDredgeBase` fits a hidden Markov model (HMM) to a given phylogenetic tree and character data. It offers additional options for penalization and optimization compared to the standard `corHMM` function.

Usage

```
corHMMDredgeBase(phy, data, rate.cat, root.p="maddfitz", pen.type = "l1", lambda = 1, rate.mat=NULL, no
```

Arguments

`phy`	An object of class `phylo`, representing the phylogenetic tree.
`data`	A data frame containing character states for the tips of the phylogeny. The first column should match the tip labels in `phy`, and the second onwards column should contain the observed states.
`rate.cat`	An integer specifying the number of rate categories in the model.
`root.p`	A vector of probabilities for the root states or a method to estimate them. The default is the `"maddfitz"` method.
`pen.type`	The type of penalization applied to the model. Options include `"l1"`, `"l2"`, `"er"`, and `"unreg"`. See Details.
`lambda`	A hyper-parameter that adjusts the severity of the penalty, ranging from 0 (no regularization) to 1 (full penalization). Default is 1.
`rate.mat`	A user-supplied matrix containing indexes of parameters to be optimized. If NULL, an all rates different model is estimated.
`node.states`	A method for estimating node states. Options include `"marginal"`, `"joint"`, and `"none"`.

fixed.nodes	Specifies that states for nodes in the phylogeny are assumed fixed. These are supplied as node labels in the <e2><80><9c>phylo<e2><80><9d> object. Default is FALSE.
ip	Initial values used for the likelihood search. Can be a single value or a vector of unique values for each parameter. The default is ip=1.
nstarts	The number of random restarts to be performed. The default is nstarts=0.
n.cores	The number of processor cores to spread out the random restarts.
get.tip.states	Logical value indicating whether tip reconstructions should be output. The default is FALSE.
lewis.asc.bias	Logical value indicating whether to correct for observing a dataset that is not univariate. The default is FALSE
collapse	A logical value indicating whether to collapse branches with no variation in states. Default is FALSE.
lower.bound	The lower bound for the rate parameters during optimization. Default is 1e-10.
upper.bound	The upper bound for the rate parameters during optimization. Default is 100.
opts	A list of options to pass to nloptr.
p	A vector of transition rates. Allows the user to calculate the likelihood given a specified set of parameter values to specified as fixed and calculate the likelihood.
grad	A logical value indicating whether to use gradient-based optimization. Default is FALSE.

Details

There are 3 penalty types available for users (though this may be expanded in the future). They are l1, l2, and er. The first two penalty types are analagous to lasso and ridge regression. Whereas the er penalization is analagous to Zhou et al.'s (2024) McLasso and penalizes the distance between rate values (i.e., it prefers rate matrices that are closer to an equal-rates model).

Under an l1 regularization scheme, the likelihood is penalized by the mean transition rate. The mean is used instead of the sum because a sum would overly penalize more complex models by virtue of them having more parameters. This leads to scenarios where simpler models have much better likelihoods than more complex models

Under an l2 regularization scheme, the likleihood is penalized by the squared mean transition rate.

Under an er regularization scheme, the likleihood is penalized by the average distance between parameters. If all the parameters were the same (an equal rates model), the penalty would be 0.

A user can also set pen.type = 'unreg' for an unpenalized likelihood which would be identical to the original implementation of corHMM. More work needs to be done to determine when to use each type of penalization, but generally using any penalization will be good for smaller datasets which tend to be high variance. l2 is the most aggresive penalization, shrinking paramaters more quickly than other methods and leading to more dropped (and potentially finding more unecessary) parameters. er is the most similar to an unregularized model as it does not necessarily penalize high parameter values. It will however penalize a model that has one parameter that is much higher than the rest unless there is significant evidence that this outlier parameter is needed. In practice, er,

behaves very similarly to unregularized models. l1 regularization is an intermediate penalization between l2 and er.

The grad option employs a numerical gradient for the optimization. This is a particularly inefficient way to find a gradient as it will require at least k iterations per likelihood calculation. However, I have found that this improves the search stability and speed as the number of iterations is greatly reduced when a gradient is provided. This is also important in cases where there are a lot of parameters (k is large). In these cases the parameter space is so highly dimensional that many search algorithms struggle. In the future I hope to implement a more efficient gradient calculation and combine a gradient based local optimizaiton with a global optimization scheme.

*Note: Many details of corHMMDredgeBase and corHMM are the same and will not be repeated here. If an arguement is unclear check the Details section of corHMM. The focus of these details will be on the unique aspects of the dredge fitting approach.

Value

corHMM returns an object of class corHMM. This is a list with elements:

$loglik	the maximum negative log-likelihood.
$AIC	Akaike information criterion.
$AICc	Akaike information criterion corrected for sample size.
$rate.cat	The number of rate categories specified.
$solution	a matrix containing the maximum likelihood estimates of the transition rates. Note that the rate classes are ordered from slowest (R1) to fastest (Rn) with respect to state 0.
$index.mat	The indices of the parameters being estimated are returned. This also is a way to allow the estimation of transition rates for parameters not oberved in the dataset. Say you have 2 traits X and Y, where the combinations 00, 01, and 11 are observed (10 is not). A 4 by 4 index matrix could be used to force 10 into the model.
$data	User-supplied dataset.
$data.legend	User-supplied dataset with an extra column of trait values corresponding to how corHMM calls the user data.
$phy	User-supplied tree.
$states	The likeliest states at each internal node. The state and rates reconstructed at internal nodes are in the order of the column headings of the rates matrix.
$tip.states	The likeliest state at each tip. The state and rates reconstructed at the tips are in the order of the column headings of the rates matrix.
$states.info	a vector containing the amount of information (in bits) that the tip states and model gives to each node.
$iterations	The number of iterations used by the optimization routine.
$root.p	The root prior used in model estimation.

Author(s)

James D. Boyko

References

Boyko, J. D. 2024. Automatic Discovery of Optimal Discrete Character Models through Regularization. In prep.

Zhou, Y., Gao, M., Chen, Y., Shi, X., 2024. Adaptive Penalized Likelihood method for Markov Chains.

Examples

```
data(primates)
phy <- multi2di(primates[[1]])
data <- primates[[2]]
MK_3state <- corHMMDredgeBase(phy = phy, data = data, rate.cat = 1, pen.type = "l1", root.p = "maddfitz", lambda = 1)
MK_3state
```

`examples` *Example datasets*

Description

Example files for running various functions in `corHMM`. The "primates" dataset comes from the example files provided by BayesTraits, though here we only include a single tree with branch lengths scaled to time. The "primates.paint" dataset is the same, but with the tree painted according to hypothetical regimes. Finally, the "rayDISC.example" dataset provides an example on how polymorphic data can be coded for `rayDISC`.

Format

a list object that contains a tree of class "phylo" and a dataframe that contains the trait data

References

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

`fitCorrelationTest` *Test for correlation*

Description

Automatically fits a set of independent and dependent models to test for correlation between characters.

Usage

```
fitCorrelationTest(phy, data, simplified_models=FALSE)
```

Arguments

phy	a phylogenetic tree, in ape "phylo" format.
data	a data.frame containing species information. The first column must be species names matching the phylogeny. Additional columns contain discrete character data.
simplified_models	A boolean which indicates whether to include simplified independent and dependent models (currently only works for two binary-state characters; see Details).

Details

This function automatically fit a set of multi-rate independent and dependent models (with default corHMM options) to drastically reduce false support for correlation. Currently, the simplified models are only available for two binary-state characters, but it is straightforward for users to use the tools available in corHMM to create model structures specific to their questions when the datasets are more complex.

The correlation between two characters is often interpreted as evidence that there exists a significant and biologically important relationship between them. However, Maddison and FitzJohn (2015) recently pointed out that in certain situations find evidence of correlated evolution between two categorical characters is often spurious, particularly, when the dependent relationship stems from a single replicate deep in time. In Boyko and Beaulieu (in prep) we showed that there is, in fact, a statistical solution to the problem posed by Maddison and FitzJohn (2015) naturally embedded within the expanded model space afforded by the hidden Markov model (HMM) framework.

The problem of single unreplicated evolutionary events manifests itself as rate heterogeneity within our models and that this is the source of the false correlation. Therefore, we argue that this problem is better understood as model misspecification rather than a failure of comparative methods to account for phylogenetic pseudoreplication. We utilize HMMs to develop a multi-rate independent model which, when implemented, drastically reduces support for correlation.

Value

fitCorrelationTest returns an object of class corhmm_list. This is a list with elements:

$independent_model_fit	A corHMM object of the standard independent model ala Pagel (1994).
$correlated_model_fit	A corHMM object of the standard dependent model ala Pagel (1994).
$hidden_Markov_independent_model_fit	A corHMM object of the hidden Markov independent model which allows for rate heterogeneity independent of the focal character.
$hidden_Markov_correlated_model_fit.cat	A corHMM object of the hidden Markov dependent model which allows for rate heterogeneity independent of the focal character as well as correlation between characters.
$simplified_*	If simplified was set to TRUE, then the function will also return simplified versions of the above models. These models have fewer parameters than the above models while still being either dependent or independent models.

Author(s)

James D. Boyko

References

Maddison W.P., FitzJohn R.G. 2015. The Unsolved Challenge to Phylogenetic Correlation Tests for Categorical Characters. Syst Biol. 64:127-136.

Examples

```
data(primates)
phy <- multi2di(primates[[1]])
data <- primates[[2]]
# not run because of long run times
#corr_test_fits <- fitCorrelationTest(phy = phy, data = data, simplified_models = TRUE)
#corr_test_fits
```

---

getCVTable *Print Method for corhmm.kfold Class Objects*

Description

Summarize the scores for each fold and the average cross-validation score for objects of class `"corhmm.kfold"`.

Usage

```
getCVTable(x)
```

Arguments

x	An object of class `"corhmm.kfold"` returned from `kFoldCrossValidation`.

Value

Returns a list of scores per fold for all lambdas and an average score for each lambda.

Author(s)

James D. Boyko

getFullMat *Combines several rate class index matrices*

Description

Combines several index matrices which describe transitions between observed states into output a single index matrix for use in corHMM

Usage

```
getFullMat(StateMats, RateClassMat = NULL)
```

Arguments

StateMats	A list of index matrices describing transitions between observed states. Each unique number from 1 to n, will be independently estimated. Values of 0 are not estimated. Matrix entries of the same value are estimated to be the same rate.
RateClassMat	An optional index matrix which describes how StateMats are related to one another. This will be a matrix of size: length(StateMats) by length(StateMats). By default, all transitions between StateMats are allowed and independently estimated.

Details

This function is the final step in creating a custom hidden Markov model. It takes a list of index matrices (StateMats) which describe different ways that the observed states are related to one another and creates a single matrix to describe the model. The matrices are combined following Eq. 2 of Tarasov (2019). getFullMat is part of several functions which help the user efficiently create custom index matrices. Often, it will be more practical to begin constructing a custom model with `getRateMat4Dat`.

getStateMat will generate an index matrix of size n by n in which all transitions between the n states are allowed and independently estimated. That index matrix can then be manipulated by dropStateMatPars and equateStateMatPars. dropStateMatPars will drop specific rates from an index matrix. dropStateMatPars requires an index matrix and a vector of which rates should be dropped. equateStateMatPars will equates rates within an index matrix. equateStateMatPars requires an index matrix and a list of vectors each element of which should correspond to two or more rates to be equated.

Value

Returns an index matrix.

Author(s)

James D. Boyko

References

Boyko, J. D., and J. M. Beaulieu. 2021. Generalized hidden Markov models for phylogenetic comparative datasets. Methods in Ecology and Evolution 12:468-478.

Tarasov, S. 2019. Integration of Anatomy Ontologies and Evo-Devo Using Structured Markov Models Suggests a New Framework for Modeling Discrete Phenotypic Traits. Systematic Biology, 68(5) 698-716. doi:10.1093/sysbio/syz005

See Also

`getRateMat4Dat`

Examples

```
data(primates)
phy <- primates[[1]]
phy <- multi2di(phy)
data <- primates[[2]]
# create a legend and rate mat from a multi-character dataset.
LegendAndRateMat <- getStateMat4Dat(data)
rate.mat <- LegendAndRateMat$rate.mat
legend <- LegendAndRateMat$legend

# To create a hidden markov model first define your rate classes (state-dependent processes)
# R1 will be a manually created SYM model
R1 <- equateStateMatPars(rate.mat, c(1:6))
# R2 will only allow transitions between 1 and 2
R2 <- dropStateMatPars(rate.mat, c(3,4))
# R1 and R2 will transtion at equal rates (i.e. the parameter process will be ER)
P <- getRateCatMat(2)
P <- equateStateMatPars(P, c(1,2))
# combine our state-dependnet processes and parameter process
HMM <- getFullMat(list(R1, R2), P)

# This can now be used in a corHMM model
CustomModel <- corHMM(phy = phy, data = data, rate.cat = 2, rate.mat = HMM, node.states = "none")
```

---

getModelTable *Summarize Model Statistics for a List of corHMM Objects*

Description

`getModelTable` extracts key statistics from a list of fitted `corHMM` models and returns a summary table. The table includes the number of parameters, log-likelihood, and model selection criteria such as AIC, delta AIC, and AIC weights.

Usage

```
getModelTable(model_list, type = "AIC")
```

Arguments

model_list	A list of corHMM objects, each representing a fitted hidden Markov model to phylogenetic and character data.
type	The type of model selection criterion to use. Options are "AIC", "AICc", or "BIC". Default is "AIC".

Details

This function takes a list of models fitted using corHMM and calculates key statistics for comparison across models. Specifically, it calculates the number of parameters, log-likelihood, the chosen model selection criterion (e.g., AIC), the difference in the criterion relative to the best model (delta AIC), and the relative model weight based on the criterion.

getModelTable can handle different model selection criteria such as AIC, AICc, and BIC by specifying the type argument.

Value

A data frame with the following columns:

- np: The number of parameters in the model.
- lnLik: The log-likelihood of the model.
- AIC (or the value of type): The model selection criterion value.
- dAIC: The difference in the criterion between the model and the best model (i.e., the model with the minimum criterion value).
- AICwt: The Akaike weights, representing the relative likelihood of the model given the data.

Author(s)

James D. Boyko

See Also

corHMM for fitting hidden Markov models to phylogenetic data.

Examples

```
# Assuming you have a list of fitted corHMM models:
models <- list(model1, model2, model3)
getModelTable(models)

# To use BIC instead of AIC:
getModelTable(models, type = "BIC")
```

getStateMat4Dat *Produce an index matrix and legend from a dataset*

Description

Takes a dataset to produce an index matrix that corresponds to a single state-dependent process (i.e. a single rate category) and a legend which matches input data to the rows and columns of the index matrix and corHMM solution.

Usage

```
getStateMat4Dat(data, model = "ARD", dual = FALSE, collapse = TRUE, indep = FALSE)
```

Arguments

data	A data matrix containing species information in the same format as the main corHMM function: column[,1] is species names, column[,2:n] are the discrete states.
model	One of "ARD", "SYM", or "ER". ARD: all rates differ. SYM: rates between any two states do not differ. ER: all rates are equal.
dual	A boolean indicating whether or not to include dual transitions.
collapse	a boolean indicating whether to collapse multiple character combinations into only the observed states. For example, if true a two character dataset contained (0,0), (1,0), and (1,1), this would be collapsed into 1,2,3. However, if set to false it would 1,2,4. In combination with a custom rate matrix this allows for the estimation of transitions between the unobserved character combination. The default is TRUE
.	
indep	A boolean indicating whether or not to return an independent or correlated model.

Details

This function will generate an index matrix based on user provided data. It provides a useful starting point for further modifications using dropStateMatPars, equateStateMatPars, and getFullMat. If more than a single column of data is given double transitions between characters are disallowed. For example, if character 1 is the presence or absence of limbs, and character 2 is the presence or absence of fingers, then the transition from absence of limbs and fingers to presence of limbs and fingers is automatically disallowed. This is consistent with Pagel's (1994) model of correlated character evolution.

Value

$legend	A named vector. The elements of the vector are all the unique state combinations in the user data. The names of the vector are the state number assigned to each combination.
$rate.mat	A rate index matrix describing a single rate class.

Author(s)

James D. Boyko

References

Pagel, M. 1994. Detecting correlated evolution on phylogenies: a gneeral method for the comparative analysis of discrete characters. Proc. R. Soc. Lond. B 255:37-45.

See Also

`getFullmat`

Examples

```
data(primates)
phy <- primates[[1]]
phy <- multi2di(phy)
data <- primates[[2]]
# create a legend and rate mat from a multi-character dataset.
LegendAndRateMat <- getStateMat4Dat(data)
rate.mat <- LegendAndRateMat$rate.mat
legend <- LegendAndRateMat$legend
```

`get_batch_profile_lik` *Perform Batch Profile Likelihood Analysis for Multiple Parameters*

Description

`get_batch_profile_lik` is a wrapper function that performs profile likelihood analysis for multiple parameters of a fitted `corHMM` object. It allows exploration of the likelihood surface by evaluating the likelihood at various points along the parameter space.

Usage

```
get_batch_profile_lik(corhmm_obj, range_factor, n_points, verbose=FALSE, ncores = NULL, dredge = FALSE
```

Arguments

`corhmm_obj`	A fitted `corHMM` object representing the model for which profile likelihood analysis is to be performed.
`range_factor`	A numeric factor determining the range over which to generate points for the profile likelihood analysis. This value is used to calculate the bounds around the maximum likelihood estimates (MLEs) for each parameter.
`n_points`	The number of points to generate for each parameter along its profile likelihood curve.
`verbose`	Whether to print messages about which parameter is being optimized.

ncores	The number of processor cores to be used for parallel computation. Default is NULL, which uses a single core.
dredge	Logical value indicating whether to include model penalization factors such as pen_type and lambda from the fitted corHMM object. Default is FALSE.

Details

This function performs a profile likelihood analysis for each parameter in a fitted corHMM model. It evaluates the likelihood at logarithmically spaced points around the maximum likelihood estimates (MLEs) for the parameters. If dredge is set to TRUE, the function also considers model penalization terms.

The function works by first generating a set of points along a logarithmic scale for each parameter, then fixing one parameter at a time while optimizing over the others. The resulting profile likelihood values are returned for each parameter.

Parallel computation can be enabled using the ncores argument to speed up the analysis.

Value

A list containing the profile likelihood results for each parameter. Each entry in the list corresponds to a parameter and contains the profile likelihood values across the range of points evaluated. The original corHMM object is also included in the output.

Author(s)

Your Name

See Also

corHMM for fitting hidden Markov models to phylogenetic data.

Examples

```
# Assuming you have a fitted corHMM object:
data(primates)
phy <- multi2di(primates[[1]])
data <- primates[[2]]
corhmm_fit <- corHMM(phy = phy, data = data, rate.cat = 1)
profile_results <- get_batch_profile_lik(corhmm_fit, range_factor = 2, n_points = 50)
plot_batch_profile_lik(profile_results)
```

kFoldCrossValidation *Perform K-Fold Cross-Validation for corHMM Models*

Description

This function performs k-fold cross-validation on a given corHMM model by dividing the data into k equally sized subsets. The function evaluates model performance across multiple lambda regularization values, if provided. Optionally, it can save the trained models for each fold and return the cross-validation results.

Usage

```
kFoldCrossValidation(corhmm_obj, k, lambdas = NULL, return_model = TRUE, save_model_dir = NULL, model_n
```

Arguments

corhmm_obj	A corHMM object that contains a fitted model.
k	An integer specifying the number of folds to divide the data into for cross-validation.
lambdas	A numeric vector of lambda regularization values to evaluate during cross-validation. If NULL, the lambda value from corhmm_obj will be used. Defaults to NULL.
return_model	A logical value indicating whether to return the trained models for each fold. Defaults to TRUE.
save_model_dir	A character string specifying the directory to save the trained models for each fold. If NULL, models will not be saved. Defaults to NULL.
model_name	A character string specifying the base name for saved model files. If NULL, a default name "corhmm.obj" is used. Defaults to NULL.

Details

The function splits the data into k folds and trains a separate corHMM model for each fold by leaving one fold out as the test set. The remaining folds are used for training the model. The performance of the model is evaluated on the test set using a divergence-based (Jensen-Shannon Divergence) scoring method. Evaluations are based on estimating the tips which were removed for that particular fold given the newly fitted model.

The function supports evaluating models across different lambda regularization values. If lambdas are provided, models are trained and evaluated for each lambda value. The results, including the models (if return_model = TRUE) and cross-validation scores, are returned as a list.

Value

A list of cross-validation results, including the following components:

models	A list of the trained models for each fold (if return_model = TRUE).
scores	A numeric vector of the cross-validation scores for each fold.
averageScore	The average cross-validation score across all folds.

Author(s)

James D. Boyko

Examples

```
data(primates)
phy <- multi2di(primates[[1]])
data <- primates[[2]]
dredge_fits <- corHMMDredge(phy = phy, data = data, max.rate.cat = 1, pen.type = "l1", root.p = "maddfitz", lambda =
model_table <- getModelTable(dredge_fits)
dredge_model <- dredge_fits[[which.min(model_table$dAIC)]]
k_fold_res <- kFoldCrossValidation(dredge_model, k = 5, lambdas = c(0,0.25,0.5,0.75,1))
getCVTable(k_fold_res)
```

---

makeSimmap *Simulate a character history*

Description

Produces a character history given some of the outputs of a corHMM object.

Usage

```
makeSimmap(tree, data, model, rate.cat, root.p="yang", nSim=1, nCores=1, fix.node=NULL,
fix.state=NULL, parsimony = FALSE, max.attempt = 1000, collapse=TRUE)
```

Arguments

tree	A phylogeny of class phylo.
data	a data.frame containing species information. The first column must be species names matching the phylogeny. Additional columns contain discrete character data.
model	The transition rate matrix.
rate.cat	The number of rate categories.
root.p	The root prior to begin the sampling at the root.
nSim	The number of simmaps to be simulated.
nCores	The number of cores to be used.
fix.node	A vector specifying node numbers to be fixed. Also possible to fix tips if using a hidden Markov model. Tips are in the order of tree$tip.label.
fix.state	Specifies which states to fix the nodes. States are specified according to position in the rate matrix. E.g. If I had binary observed characters 0/1 and two hidden rate classes A/B and wanted to fix a node as 1B, I would set this to 4.

`parsimony` A boolean indicating whether node states should be based on conditional likelihood (per Bollback 2006), or if they should be consistent with a parsimonious model (if TRUE). Parsimony states are evaluted by dividing the rates present in the variable, `model`, by 1000 and evaluating the conditional likelihood of each state. However, by lowering the rates we can approximate a parsimony reconstruction (Steel and Penny 2000).

`max.attempt` A numeric value indicating the maximum number of attempts to create a possible path between an initial and final state on a branch. When the maximum value is reached we use the Floyd-Walsh algorithm to produce the shortest path between the two states and divide the branch into equal segments.

`collapse` a boolean indicating whether to collapse multiple character combinations into only the observed states. For example, if true a two character dataset contained (0,0), (1,0), and (1,1), this would be collapsed into 1,2,3. However, if set to false it would 1,2,4. In combination with a custom rate matrix this allows for the estimation of transitions between the unobserved character combination. The default is `TRUE`

.

Details

This function will generate a character history given a model and dataset. It has a similar structure to the simmap generated in phytools and follows the methods of Bollback (2006). If using hidden states, then it is necessary to reconstruct the tip probabilities as well as the node probabilities (i.e. get.tip.states must be `TRUE` when running `corHMM`). We chose not to implement any new plotting functions, instead `makeSimmap` produces a simmap object which is formatted so it can used with other R packages such as phytools (Revell, 2012). For additional capabilities, options, and biological examples we refer readers to the detailed _Generalized corHMM_ vignette.

Value

A list of simmaps.

Author(s)

James D. Boyko

References

Boyko, J. D., and J. M. Beaulieu. 2021. Generalized hidden Markov models for phylogenetic comparative datasets. Methods in Ecology and Evolution 12:468-478.

Bollback, J. P. 2006. SIMMAP: stochastic character mapping of discrete traits on phylogenies. BMC Bioinformatics 7:88.

Revell, L. J. 2012. phytools: an R package for phylogenetic comparative biology (and other things). Methods in Ecology and Evolution, 3(2), 217-223.

Steel, M., and D. Penny. 2000. Parsimony, Likelihood, and the Role of Models in Molecular Phylogenetics. Molecular Biology and Evolution 17:839-850.

Examples

```
data(primates)
phy <- primates[[1]]
phy <- multi2di(phy)
data <- primates[[2]]

##run corhmm
MK <- corHMM(phy, data, 1)

##get simmap from corhmm solution
model <- MK$solution
simmap <- makeSimmap(tree=phy, data=data, model=model, rate.cat=1, nSim=1, nCores=1)

## we import phytools plotSimmap for plotting
# library(phytools)
# plotSimmap(simmap[[1]])
```

---

plotMKmodel *Plot a Markov model*

Description

Plots a diagram of a Markov model from the output of corHMM or a custom index matrix

Usage

```
plotMKmodel(corhmm.obj, rate.cat = NULL, display = "column", color = c("blue", "red"),
arrow.scale = 1, text.scale = 1, vertex.scale = 1)
```

Arguments

corhmm.obj	an object of class corHMM or matrix.
rate.cat	if using a custom matrix then the number of rate categories must be indicated.
display	the structure of the plot. one of "column", "square", or "row".
color	Either, 1. a vector of 2 colors to create a gradient from low transition rates (first element) to high transition rates (second element), or 2. "col.blind" which will use the color pallete "plasma" from viridis.
arrow.scale	determines the size of the arrows for the Markov diagram.
text.scale	determines the size of the text for the plotted matrix.
vertex.scale	determines the size of the text for the Markov diagram.

Details

Plots Markov models in a ball and stick type diagram next to its corresponding matrix. If plotting a hidden rates model it will produce a compound plot describing how the different rate classes are related to one another. If the input is a corHMM result then arrows are colored by relative rate. If the input is a custom matrix arrows are colored by the paramater index.

Value

Returns a ball and stick diagram of the input model.

Author(s)

James D. Boyko

References

Boyko, J. D., and J. M. Beaulieu. 2021. Generalized hidden Markov models for phylogenetic comparative datasets. Methods in Ecology and Evolution 12:468-478.

Examples

```
data(primates)
phy <- primates[[1]]
phy <- multi2di(phy)
data <- primates[[2]]
# create a legend and rate mat from a multi-character dataset.
LegendAndRateMat <- getStateMat4Dat(data)
rate.mat <- LegendAndRateMat$rate.mat
legend <- LegendAndRateMat$legend

# To create a hidden markov model first define your rate classes (state-dependent processes)
# R1 will be a manually created SYM model
R1 <- equateStateMatPars(rate.mat, c(1:6))
# R2 will only allow transitions between 1 and 2
R2 <- dropStateMatPars(rate.mat, c(3,4))
# R1 and R2 will transtion at equal rates (i.e. the parameter process will be ER)
P <- getRateCatMat(2)
P <- equateStateMatPars(P, c(1,2))
# combine our state-dependnet processes and parameter process
HMM <- getFullMat(list(R1, R2), P)
# plot the input
plotMKmodel(HMM, rate.cat = 2)

# This can now be used in a corHMM model
CustomModel <- corHMM(phy = phy, data = data, rate.cat = 2, rate.mat = HMM, node.states = "none")
# plot the output
plotMKmodel(CustomModel)
```

---

plotRECON *Plot ancestral state reconstructions*

Description

Plots maximum likelihood ancestral state estimates on tree

Usage

```
plotRECON(phy, likelihoods, piecolors=NULL, cex=0.5, pie.cex=0.25, file=NULL,
height=11, width=8.5, show.tip.label=TRUE, title=NULL, ...)
```

Arguments

phy	a phylogenetic tree, in ape "phylo" format.
likelihoods	likelihoods for ancestral states (see Details).
piecolors	a vector of colors for states.
cex	specifies the size of the font for labels (if used).
pie.cex	specifies the size of the symbols to plot on tree.
file	filename to which a pdf is saved.
height	height of plot.
width	width of plot.
show.tip.label	a logical indicating whether to draw tip labels to tree. The default is TRUE.
title	an optional title for the plot.
...	Additional arguments to be passed to the plot device

Details

Plots ancestral state estimates on provided tree. The likelihoods can be the states of an object of class rayDISC or class corDISC, or the lik.anc of an object of class ace (from the ape package).

Value

A plot indicating the maximum likelihood ancestral states at each internal node.

Author(s)

Jeffrey C. Oliver

See Also

corDISC, rayDISC

Examples

```
data(rayDISC.example)
## Perform ancestral state estimation, using a single rate of evolution and marginal
## reconstruction of ancestral states
recon <- rayDISC(rayDISC.example$tree,rayDISC.example$trait,model="ER",
node.states="marginal")
## Plot reconstructions on tree
plotRECON(rayDISC.example$tree,recon$states,title="rayDISC Example")
```

plot_batch_profile_lik *Plot Batch Profile Likelihoods*

Description

Plots profile likelihoods for each parameter in a batch from the output of a profile corHMM object, displaying the log-likelihoods across parameter values and indicating the maximum likelihood estimate (MLE) and the 95% confidence interval.

Usage

```
plot_batch_profile_lik( corhmm_profile, n_cols = NULL, n_rows = NULL, mar = c(5, 4, 4, 1) + 0.1, ci_level
```

Arguments

corhmm_profile	a list containing profile likelihood tables for each parameter and the corHMM object with MLE and loglik attributes.
n_cols	optional; number of columns for the plotting layout. If NULL, automatically calculated based on the number of parameters.
n_rows	optional; number of rows for the plotting layout. If NULL, automatically calculated based on the number of parameters.
mar	margins around the plot. Defaults to c(5, 4, 4, 1) + 0.1.
ci_level	z-value for the 95% confidence interval. Defaults to 1.96.
polygon_col	color of the polygon under the curve. Defaults to "lightgrey".
line_col	color of the profile likelihood curve. Defaults to "black".
line_type	type of the profile likelihood curve. Defaults to "l".
mle_col	color of the MLE point. Defaults to "blue".
ci_line_col	color of the 95% CI line. Defaults to "black".
ci_line_type	line type for the 95% CI line. Defaults to "dashed".
axis_tick_length	length of the axis ticks, with negative values indicating ticks pointing inwards. Defaults to -0.2.
label_cex	character expansion size for labels, affecting the size of text labels. Defaults to 0.7.
ylim	a user-specified upper and lower limit to the y-axis.

Details

This function generates a series of plots for the profile likelihood of each parameter in the input corHMM model object. It visualizes the log-likelihood across the range of parameter values, highlights the maximum likelihood estimate (MLE), and denotes the 95% confidence interval with a horizontal dashed line. The function is designed to accommodate the batch analysis of multiple parameters, organizing the plots in a specified layout and allowing for extensive customization of plot aesthetics.

Value

Generates and displays a grid of profile likelihood plots, one for each parameter in the input model, with visual cues for MLE and confidence intervals.

Author(s)

James D. Boyko

`rayDISC` *Evolution of categorical traits*

Description

Fits a model of evolution for categorical traits, allowing for multi-state characters, polymorphisms, missing data, and incompletely resolved trees

Usage

```
rayDISC(phy,data, ntraits=1, charnum=1, rate.mat=NULL, model=c("ER","SYM","ARD"),
node.states=c("joint", "marginal", "scaled", "none"), state.recon=c("subsequently"),
lewis.asc.bias=FALSE, p=NULL, root.p="yang", ip=NULL, lb=1e-9, ub=100, verbose=TRUE,
diagn=FALSE)
```

Arguments

`phy`	a phylogenetic tree, in `ape` "phylo" format.
`data`	a data matrix containing species information (see Details).
`ntraits`	specifies the number of traits to included in the analysis.
`charnum`	specified the character to analyze.
`rate.mat`	a user-supplied rate matrix index of parameters to be optimized.
`model`	specifies the underlying model.
`node.states`	method used to calculate ancestral states at internal nodes.
`state.recon`	whether to reconstruct states jointly with the rates or subsequent to the rates being optimized.
`lewis.asc.bias`	a logical indicating whether the ascertainment bias correction of Lewis et al. 2001 should be used. The default is `FALSE`.
`p`	a vector of transition rates. Allows the user to calculate the likelihood given a specified set of parameter values to specified as fixed and calculate the likelihood.
`root.p`	a vector used to fix the probabilities at the root, but "yang" and "maddfitz" can also be supplied to use the method of Yang (2006) and FitzJohn et al (2009), respectively (see details).
`ip`	initial values used for the likelihood search. Can be a single value or a vector of unique values for each parameter. The default is `ip=1`.

lb	lower bound for the likelihood search. The default is lb=0.
ub	upper bound for the likelihood search. The default is ub=100.
verbose	a logical indicating whether progress should be printed to the screen.
diagn	logical indicating whether diagnostic tests should be performed. The default is FALSE.

Details

__THIS FUNCTION IS NO LONGER NECESSARY AS IT IS NOW ENTIRELY SUBSUMED WITHIN__ corHMM (see _Generalized corHMM_ vignette). But we still provide it for those that are more comfortable using it than exploring the new corHMM function. As before, rayDISC takes a tree and a trait file and estimates transition rates and ancestral states for binary or multistate characters. The first column of the trait file must contain the species labels to match to the tree, with the second, third, fourth, and so on, corresponding to the traits of interest. Use the charnum variable to select the trait for analysis. Also, the input phylogeny need not be bifurcating as the algorithm is implemented to handle multifucations. Polytomies are allowed by generalizing Felsenstein's (1981) pruning algorithm to be the product of the probability of observing the tip states of n descendant nodes, rather than two, as in the completely bifurcating case.

The user can fix the root state probabilities by supplying a vector to the root.p. If the user supplies the flag root.p="yang", then the estimated transition rates are used to set the weights at the root (see pg. 124 Yang 2006), whereas specifying root.p="maddfitz" employs the same procedure described by Maddison et al. (2007) and FitzJohn et al. (2009). Note that the default root.p=NULL assumes equal weighting among all possible states.

Ambiguities (polymorphic taxa or taxa missing data) are assigned likelihoods following Felsenstein (2004, p. 255). Taxa with missing data are coded "?" with all states observed at a tip. Polymorphic taxa are coded with states separated by an "&". For example, if a trait has four states and taxonA is observed to be in state 1 and 3, the character would be coded as "1&3". corHMM then uses this information to assign a likelihood of 1.0 to both states. Missing data are treated as ambiguous for all states, thus all states for taxa missing data are assigned a likelihood of 1.0. For example, for a four-state character (i.e. DNA), a taxon missing data will have likelihoods of all four states equal to 1.0 [e.g. L(A)=1.0, L(C)=1.0, L(G)=1.0, L(T)=1.0].

In all ancestral state reconstruction implementations, the rates are first estimated, and subsequently, the MLE estimates of the rates are used to determine either the state probabilities (i.e., marginal or "scaled") or maximum likelihood states at nodes. This is the default – i.e., the state.recon="subsequently" argument. However, for this function only, we also allow for both rates and states to be estimated jointly. This can be done with state.recon="estimate". We also allow for a hypothesis about states at all or even some nodes to help fixed, with the rates (and in some cases some of the states) being estimated. This is state.recon="given". For more information please see Vignette "Getting Likelihoods From Reconstructions".

Value

rayDISC returns an object of class rayDISC. This is a list with elements:

$loglik	the maximum negative log-likelihood.
$AIC	Akaike information criterion.
$AICc	Akaike information criterion corrected for sample size.

$ntraits	The number of traits specified.
$solution	a matrix containing the maximum likelihood estimates of the transition rates.
$solution.se	a matrix containing the approximate standard errors of the transition rates. The standard error is calculated as the square root of the diagonal of the inverse of the Hessian matrix.
$index.mat	The indices of the parameters being estimated are returned. The numbers correspond to the row in the eigvect and can useful for identifying the parameters that are causing the objective function to be at a saddlepoint.
$lewis.asc.bias	The setting describing whether or not the Lewis ascertainment bias correction was used.
$opts	Internal settings of the likelihood search.
$data	User-supplied dataset.
$phy	User-supplied tree.
$states	The likeliest states at each internal node.
$tip.states	NULL
$iterations	The number of iterations used by the optimization routine.
$eigval	The eigenvalues from the decomposition of the Hessian of the likelihood function. If any eigval<0 then one or more parameters were not optimized during the likelihood search.
$eigvect	The eigenvectors from the decomposition of the Hessian of the likelihood function is returned.
$bound.hit	A logical for diagnosing if rate parameters were constrained by lb or ub values during optimization.
$message.tree	A list of taxa which were listed in the data matrix, but were not present in the passed phylo object. These taxa will be excluded from the analysis. message.tree is null if all taxa in data are included in tree.
$message.data	A list of taxa which were present in the passed phylo object, but lacked data in the passed data matrix. These taxa will be coded as missing data (all states equally likely). message.data is null if all taxa in tree have entries in data matrix.

Author(s)

Jeffrey C. Oliver and Jeremy M. Beaulieu

References

Felsenstein, J. 1981. A likelihood approach to character weighting and what it tells us about parsimony and compatibility. Biological Journal of the Linnean Society 16: 183-196.

Felsenstein J. 2004. Inferring phylogenies. Sunderland MA: Sinauer Associates.

FitzJohn, R.G., W.P. Maddison, and S.P. Otto. 2009. Estimating trait-dependent speciation and extinction rates from incompletely resolved phylogenies. Systematic Biology 58:595-611.

Lewis, P.O. 2001. A likelihood approach to estimating phylogeny from discrete morphological character data. Systematic Biology 50:913-925.

Maddison, W.P., P.E. Midford, and S.P. Otto. 2007. Estimating a binary characters effect on speciation and extinction. Systematic Biology 56:701-710.

See Also

plotRECON

Examples

```
### Example 1
data(rayDISC.example)
## Perform ancestral state estimation, using an asymmetric model of evolution and marginal
## reconstruction of ancestral states
recon <- rayDISC(rayDISC.example$tree,rayDISC.example$trait,model="ARD",
node.states="marginal")

## Plot reconstructions on tree
plotRECON(rayDISC.example$tree,recon$states)

### Example 2
## Perform ancestral state estimation on second character, using a single-rate model of
## evolution, marginal reconstruction of ancestral states, and setting the lower bound for
## parameter estimates to 0.01
recon <- rayDISC(rayDISC.example$tree,rayDISC.example$trait,charnum=2,model="ER",
node.states="marginal",lb=0.01)

### Example 3
## Perform ancestral state estimation on third character, using a single-rate model of
## evolution and joint reconstruction of ancestral states
recon <- rayDISC(rayDISC.example$tree,rayDISC.example$trait,charnum=3,
model="ER",node.states="joint")
```

---

simMarkov *Simulate a character on the tree*

Description

Simulates a character using an instanteous rate matrix, storing the states at nodes and at the tips of the tree.

Usage

```
simMarkov(phy, Q, root.freqs)
```

Arguments

phy	A phylogeny of class phylo.
Q	an instantaneous rate matrix.
root.freqs	A vector of root frequencies.

Details

This function will generate a character at tips and at interior nodes given a tree and Q matrix.

Value

A two element list containing ‘TipStates‘ and ‘NodeStates‘

Author(s)

James D. Boyko

Index