Package 'ratematrix'

June 10, 2018

Title Bayesian Estimation of the Evolutionary Rate Matrix

Version 1.0

Description
Estimates the evolutionary rate matrix (R) using Markov chain Monte Carlo (MCMC) as described in Caetano and Harmon (2017) <doi:10.1111/2041-210X.12826>. The package has functions to run MCMC chains, plot results, evaluate convergence, and summarize posterior distributions.

URL https://github.com/Caetanods/ratematrix

License GPL (>= 2.0)

Encoding UTF-8

LazyData true

Imports ape, geiger, coda, corpcor, MASS, phylolm, readr, mvMORPH, Rcpp, ellipse

Suggests microbenchmark, knitr, rmarkdown, phytools

VignetteBuilder knitr

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

NeedsCompilation yes

Author Daniel Caetano [aut, cre], Luke Harmon [aut]

Maintainer Daniel Caetano <caetanods1@gmail.com>

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2018-06-10 15:27:53 UTC

R topics documented:

`anoles` *Data and phylogenetic tree for Anolis lizards*

Description

See description of the data in Caetano and Harmon (2018). Measurements are log-transformed. Morphological data was compiled from Pinto et al. (2008), Mahler (2010), and Moreno-Arias and Calderon-Espinosa (2016). "Limb_length" data is a composite measurement calculated from the sum of the parts of the front limb.

Usage

```
data(anoles)
```

Format

A list with a dataframe with the morphological data ($data) and a list of stochastic mapped trees with three regimes "island", "mainland" and "mainland.2" ($phy.map).

References

Moreno-Arias, R. A., and M. L. Calderon-Espinosa. 2016. Patterns of morphological diversification of mainland Anolis lizards from northwestern South America. Zool J Linn Soc 176:632–647.

Pinto, G., D. L. Mahler, L. J. Harmon, and J. B. Losos. 2008. Testing the island effect in adaptive radiation: rates and patterns of morphological diversification in Caribbean and mainland Anolis lizards. Proc Biol Sci 275:2749–2757.

Mahler, D. L., L. J. Revell, R. E. Glor, and J. B. Losos. 2010. Ecological Opportunity and the Rate of Morphological Evolution in the Diversification of Greater Antillean Anoles. Evolution 64:2731–2745.

`centrarchidae` *Data and phylogenetic tree for Centrarchidae fishes*

Description

Data from Revell and Collar (2009). Measurements are log-transformed and size-corrected.

Usage

```
data(centrarchidae)
```

Format

A list with a dataframe with the morphological data ($data) and a 'Simmap' format tree with the regime of "narrow_diet" and "wide_diet" mapped on the tree ($phy.map).

Source

http://datadryad.org/resource/doi:10.5061/dryad.4t157/1

References

Revell, L. J., and D. C. Collar. 2009. Phylogenetic Analysis of the Evolutionary Correlation Using Likelihood. Evolution 63:1090–1100.

Revell, L. J. 2013. Ancestral character estimation under the threshold model from quantitative genetics. Evolution 68(3):743–759.

checkConvergence *Performs convergence tests*

Description

Make convergence test for the MCMC chain. We STRONGLY recommend doing at least two independent searches to test convergence. Please see 'Details' for more information.

Usage

```
checkConvergence(...)
```

Arguments

...	posterior(s) distribution(s) of parameter estimates. This can be a single MCMC chain or multiple independent chains from the same model. The type of convergence analysis will be dependent on the number of MCMC chains provided as input. See 'Details'.

Details

Function performs convergence tests using the potential scale reduction factor (Gelman's R) or the Heidelberger test. The Gelman's R test will be performed if two or more MCMC chains are provided as input. If only one MCMC chain is provided, then the function will perform the Heidelberger test (and print a message about it).

Multiple chains need to be replicates of the same analysis (e.g., multiple runs of the 'ratematrixMCMC' function with the same set of arguments and, in the best scenario, with varying starting points). We recommend users to perform the Gelman's R test by providing two or more independent MCMC chains with different starting points. This test is more robust than the Heidelberger test. The advantage of the Heidelberger test is that it can be used with a single MCMC chain, so it can be useful for a preliminary test prior to running a full convergence analysis with multiple chains. (Our experience shows that performing the Heidelberger test alone can return false convergence results.) Convergence can also be investigated using the 'logAnalizer' and 'computeESS'.

The 'Gelman's R' test is based on the potential scale reduction factor which is expected to be equal to 1 when convergence is achieved. If you see values close to 1 (e.g., ~1.01 to 1.05) it means that you just need to get more samples from the MCMC (see 'continueMCMC' function). See more information about each of these tests in the references below and in the documentation for the functions 'coda::gelman.diag' and 'coda::heidel.diag', both from the package 'coda'.

Value

The format of the output depends of the type of test performed. The Gelman's R test will return a list with two elements. The first element is a list with the results for the potential scale reduction factor for the root values and the evolutionary rate matrices. The test for the R matrices is performed element by element, the names of the columns show the number of the row and column for each

element. The length of this list will depend on the number of rate regimes fitted to the phylogenetic tree. The Heidelberger test also returns a list with two elements, the first element is a table with one column for the root values and each evolutionary rate matrix regime fitted to the tree. The colnames show the type of diagnostic used, the values are whether the test passed or not. The second element of the list, independent of the type of convergence test, is a estimate of the Effective Sample Size for each parameter of the model.

Author(s)

Daniel S. Caetano and Luke J. Harmon

References

Gelman, A and Rubin, DB (1992) Inference from iterative simulation using multiple sequences, Statistical Science, 7, 457-511.

Heidelberger, P and Welch, PD (1983) Simulation run length control in the presence of an initial transient. Opns. Res., 31, 1109-44.

Examples

```
data(centrarchidae)
handle1 <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, gen=10000
                          , dir=tempdir())
posterior1 <- readMCMC(handle1, burn=0.25, thin=10)
handle2 <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, gen=10000
                          , dir=tempdir())
posterior2 <- readMCMC(handle2, burn=0.25, thin=1)
## Note that these are short chains used here as example only.
## A convergence test using 'Gelman's R' calculated from two independent MCMC chains.
checkConvergence(posterior1, posterior2)
```

`computeESS` *Compute the ESS for the MCMC samples*

Description

Computes the Effective Sample Size (ESS) for the parameters of the model from the MCMC samples.

Usage

```
computeESS(mcmc, p)
```

Arguments

`mcmc`	Posterior distribution object. Same as output from 'readMCMC' function.
`p`	Number of evolutionary rate matrix regimes fitted to the phylogenetic tree.

Details

Function uses 'coda' function 'effectiveSize' to compute the ESS for each of the parameters of the model separatelly. Values for the ESS is too low indicates poor mixing for the parameter of the model.

Value

A list object with the ESS value for the root, evolutionary rates, and evolutionary correlations among the traits.

Author(s)

Daniel Caetano and Luke Harmon

Examples

```
data( centrarchidae )
dt.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( dt.range[,2] - dt.range[,1] ) / 10
par.sd <- cbind(c(0,0), sqrt( c(10,10) ))
prior <- makePrior(r=2, p=2, den.mu="unif", par.mu=dt.range, den.sd="unif", par.sd=par.sd)
prior.samples <- samplePrior(n = 1000, prior = prior)
start.point <- samplePrior(n=1, prior=prior)
## Plot the prior. Red line shows the sample from the prior that will set the
##      starting point for the MCMC.
plotRatematrix(prior.samples, point.matrix = start.point$matrix, point.color = "red"
               , point.wd = 2)
plotRootValue(prior.samples)
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, prior=prior
                         , gen=10000, w_mu=w_mu, dir=tempdir())
posterior <- readMCMC(handle, burn = 0.2, thin = 10)
## Again, here the red line shows the starting point of the MCMC.
plotRatematrix( posterior, point.matrix = start.point$matrix, point.color = "red"
               , point.wd = 2)
plotRootValue(posterior)
computeESS(mcmc=posterior, p=2)
```

`continueMCMC` *Continue unfinished MCMC chain or add more generations*

Description

Function to continue an unfinished MCMC chain or to append more generations to a previously finished MCMC. It works by reading the last state of the chain and the tunning parameters of the previous chain, then restarting it from this step.

Usage

```
continueMCMC(handle, add.gen = NULL, save.handle = TRUE, dir = NULL)
```

Arguments

handle	the output of 'ratematrixMCMC'.
add.gen	number of generations to be added to a finished chain. If 'NULL' (default), the function will only continue unfinished chains.
save.handle	whether to save the updated 'handle' object to the directory. This can overwrite the previous handle file.
dir	an optional path to the output files. See 'Details'.

Details

The function will append the new generations to the same files created by the prior run of the 'ratematrixMCMC' function. The function will, by default, search for files in the same directory of the previous run (see 'handle$dir'). However, you can provide a new path (relative or absolute path) to the argument 'dir'. The path provided to 'dir' will override the path pointed by 'handle$dir'. The new 'handle' output from 'continueMCMC' will have an updated total number of generations and will also update the directory path, if required.

Value

Function will write the parameter values for each generation and the log to files. The new generations will be appended to the same files created by 'ratematrixMCMC'.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
## Continue unfinished run.
data(centrarchidae)
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, gen=10000
                         , dir=tempdir())
## Now add generations to the same MCMC chain.
handle.add <- continueMCMC(handle=handle, add.gen=10000)
```

`estimateTimeMCMC` *Time estimate to complete a MCMC chain*

Description

Estimate time minimum time needed to run the MCMC.

Usage

```
estimateTimeMCMC(data, phy, gen, eval.times = 5, singlerate = FALSE)
```

Arguments

`data`	a matrix with the data. Each column is a different trait and species names need to be provided as rownames (rownames(data) == phy$tip.label).
`phy`	a phylogeny of the class "simmap" with the mapped regimes for two or more R regimes OR a phylogeny of the class "phylo" for a single regime. The number of evolutionary rate matrices fitted to the phylogeny is equal to the number of regimes in 'phy'. Regime names will also be used.
`gen`	number of generations of the complete MCMC chain. This is used to create the time estimate for the analysis.
`eval.times`	number of replicates to compute the likelihood (default is 5). A time average across replicates will be used in order to account for the uncertainty associated with computing times.
`singlerate`	whether the function should fit a single regime and ignore the number of regimes painted to the tree (default is FALSE).

Details

Function will estimate the time based on the computation of the log-likelihood, prior density, and the Jacobian of the proposal step. The time estimated is a minimum bound based on the processing power of the current computer. Running the MCMC in different computers might change the time. Other factors, such as writing the posterior samples to large files, can influence the time to run the MCMC.

Value

Function returns a numeric value with the time estimate in hours and prints a message to the screen with the result.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
data(centrarchidae)
estimateTimeMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, gen=10000)
```

`extractCorrelation` *Extract the posterior distribution of evolutionary correlation*

Description

Function extracts the posterior distribution of evolutionary correlation among traits.

Usage

```
extractCorrelation(post)
```

Arguments

`post`	a posterior distribution object as returned by the function 'readMCMC' or a merged posterior generated by 'mergePosterior'.

Details

Returns a list with length equal to the number of regimes. Each list element is composed by a matrix with trait correlation types in the columns and the evolutionary correlations for each sample at the rows.

One can plot the correlation values using boxplots and compare their distribution. Pairwise statistical tests across the samples is also possible.

Value

a list with the posterior distribution of evolutionary correlations among traits.

Author(s)

Daniel Caetano and Luke Harmon

Examples

```
data( centrarchidae )
dt.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( dt.range[,2] - dt.range[,1] ) / 10
par.sd <- cbind(c(0,0), sqrt( c(10,10) ))
prior <- makePrior(r=2, p=2, den.mu="unif", par.mu=dt.range, den.sd="unif", par.sd=par.sd)
```

```
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, prior=prior
                         , gen=10000, w_mu=w_mu, dir=tempdir())
posterior <- readMCMC(handle, burn = 0.2, thin = 10)
## Get the correlations:
cor.list <- extractCorrelation(post = posterior)
## Plot the results:
boxplot(cor.list[[1]], main = "Regime 1") ## Regime 1
boxplot(cor.list[[2]], main = "Regime 2") ## Regime 2
```

`likelihoodFunction` *Likelihood function for the multivariate Brownian motion model*

Description

Returns the log-likelihood for the multivariate Brownian motion model with 1 or more rate regimes mapped to the tree.

Usage

```
likelihoodFunction(data, phy, root, R)
```

Arguments

`data`	a matrix with the data. Species names need to be provided as rownames (rownames(data) == phy$tip.label).
`phy`	a phylogeny of the class "simmap" with the mapped regimes or "phylo" for a single rate model.
`root`	a numeric vector with the root value (phylogenetic mean).
`R`	a matrix or a list of matrices. If 'R' is a matrix then the likelihood for a single regime is calculated. If 'R' is a list of matrices, then each matrix will be fitted to a regime in 'phy' and the length of the list need to match the number of regimes fitted to the tree.

Details

If two or more rate regimes are mapped to the phylogenetic tree, then the function calculates the likelihood using the new prunning algorithm adapted to fit multiple rate regimes. The prunning algorithm is implemented in C++ using 'Rcpp' and 'RcppArmadillo'. Otherwise the function uses the three point algorithm (Ho and Ané, 2014) to make calculations for the single regime case.

Value

The log likelihood for the multivariate Brownian motion model.

Author(s)

Daniel S. Caetano and Luke J. Harmon

References

Ho, L. S. T. and Ané, C. (2014). "A linear-time algorithm for Gaussian and non-Gaussian trait evolution models". Systematic Biology *63*(3):397-408.

Examples

```
data( centrarchidae )
root <- colMeans( centrarchidae$data )
Rlist <- list( rbind(c(0.5, 0.1),c(0.1,0.5)), rbind(c(0.5, 0),c(0,0.5)) )
likelihoodFunction(data = centrarchidae$data, phy = centrarchidae$phy.map, root = root
                   , R = Rlist)
## Get the likelihood for a single regime model:
phy.single <- mergeSimmap(phy = centrarchidae$phy.map, drop.regimes = TRUE)
Rsingle <- rbind(c(0.5, 0.1),c(0.1,0.5))
likelihoodFunction(data = centrarchidae$data, phy = phy.single, root = root, R = Rsingle)
```

---

logAnalyzer *Make analysis of the log file of the MCMC chain*

Description

Reads the log file produced by the 'ratematrixMCMC' function. Calculates acceptance ratio and shows the trace plot. Check the function 'computeESS' to compute the Effective Sample Size of the posterior distribution.

Usage

```
logAnalyzer(handle, burn = 0.25, thin = 100, show.plots = TRUE,
  print.result = TRUE, dir = NULL)
```

Arguments

handle	the output object from the 'ratematrixMCMC' function.
burn	the proportion of burn-in. A numeric value between 0 and 1.
thin	the number of generations to skip when reading the posterior distributionfrom the files. Since the files contain each step of the sampler, one can check the posterior with different 'thin' values without the need of reanalyses.
show.plots	whether to show a trace plot of the log-likelihood and the acceptance ratio. Default is TRUE.
print.result	whether to print the results of the acceptance ratio to the screen. Default is TRUE.
dir	the directory where to find the log file. If set to 'NULL' (default), the function will search for the files in the same directory that the MCMC chain was made (stored in handle$dir).

Details

The log shows the acceptance ratio for the parameters of the model and also for each of the phylogenies provided to the 'ratematrixMCMC' function (if more than one was provided as input). Also see function 'ratematrixMCMC' for a brief discussion about acceptance ratio for the parameters in 'Details'.

The acceptance ratio is the frequency in which any proposal step for that parameter was accepted by the MCMC sampler. When this frequency is too high, then proposals are accepted too often, which might decrease the efficiency of the sampler to sample from a wide range of the parameter space (the steps are too short). On the other hand, when the acceptance ratio is too low, then the steps of the sampler propose new values that are often outside of the posterior distribution and are systematically rejected by the sampler. Statisticians often suggest that a good acceptance ratio for a MCMC is something close to '0.24'. Our experience is that acceptance ratios between 0.15 and 0.4 will work just fine. Much lower or higher than this might create mixing problems or be too inefficient.

If you provided a list of phylogenies to the MCMC chain, then the sampler will randomly sample one of these phylogenies and use it to compute the likelihood of the model at each step of the MCMC. The pool of trees and/or regime configurations provided effectivelly works as a prior distribution. It is important to note that this is not equivalent to a joint estimation of the comparative model of trait evolution and phylogenetic tree, since the moves proposed by the MCMC chain are restricted to the parameters of the phylogenetic comparative model. Some of the phylogenies provided in the pool might be accepted more than others during the MCMC. When this happens, the acceptance ratio for a given tree, or set of trees, will be relativelly lower when compared to the rest. This means that the information presented in such a tree (or trees) is less represented in the posterior distribution than other trees. If this issue happens, we advise users to investigate whether these trees show a different pattern (potentially biologically informative) when compared to the other set of trees. Additionally, one might also repeat the analysis with these trees in separate in order to check whether parameters estimates are divergent.

Value

A named vector with the acceptance ratio for the whole MCMC and each of the parameters of the model. If a list of phylogenetic trees was provided to the MCMC chain, then the output is a list with the acceptance ratio for the parameters and a table showing the frequency in which each of the phylogenies was accepted in a move step.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
## Load data
data(centrarchidae)
## Run MCMC. This is just a very short chain.
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, gen=10000
                         , dir=tempdir())
```

```
## Load posterior distribution, make plots and check the log.
posterior <- readMCMC(handle, burn=0.1, thin=10)
plotRatematrix(posterior)
logAnalyzer(handle, burn=0.1, thin=10)
```

`makePrior` *Generate prior distributions for the multivariate Brownian motion model*

Description

Generates prior densities for the MCMC sampler.

Usage

```
makePrior(r, p, den.mu = "unif", par.mu, den.sd = "unif", par.sd,
  unif.corr = TRUE, Sigma = NULL, nu = NULL)
```

Arguments

`r`	number of traits in the model.
`p`	number of evolutionary rate matrix regimes fitted to the phylogeny.
`den.mu`	one of "unif" (uniform prior, default) or "norm" (normal prior).
`par.mu`	the parameters for the prior density on the vector of phylogenetic means. Matrix with 2 columns and number of rows equal to the number of traits (r). When the density ('den.mu') is set to "unif" then par.mu[,1] is the minimum values and par.mu[,2] is the maximum values for each trait. When the density is set to "norm" then par.mu[,1] is the mean values and par.mu[,2] is the standard deviation values for the set of normal densities around the vector of phylogenetic means.
`den.sd`	one of "unif" (uniform prior, default) or "lnorm" (log-normal prior).
`par.sd`	the parameters for the density of standard deviations. Matrix with 2 columns and number of rows equal to the number of evolutionary rate matrix regimes fitted to the phylogenetic tree (p). When "den.sd" is set to "unif", then 'par.sd[,1]' (the minimum) need to be a vector of positive values and 'par.sd[,2]' is the vector of maximum values. When "den.sd" is set to "lnorm" then 'par.sd[,1]' is the vector of log(means) for the density and 'par.sd[,2]' is the vector of log(standard deviations) for the distributions. If there is only one regime fitted to the tree, then 'par.sd' is a vector with length 2 (e.g., c(min, max)).
`unif.corr`	whether the correlation structure of the prior distribution on the Sigma matrix is flat. This sets an uniformative prior (as uninformative as possible) to the evolutionary correlations among the traits.

`Sigma` the scale matrix to be used as a parameter for the inverse-Wishart distribution responsible for the generation of the correlation matrices. Need to be a list of matrices with number of elements equal to the number of evolutionary rate regimes fitted to the phylogeny, in the case of a single regime, this needs to be a matrix (not a list). The scale matrix is somewhat analogous to the mean of a normal distribution. Thus, if 'Sigma' show strong positive correlation among traits, then the distribution generated by the prior will vary around positive correlations. This parameter will be ignored if 'unif.corr' is set to TRUE.

`nu` the degrees of freedom parameter for the inverse-Wishart distribution. Need to be a numeric vector with length equal to the number of evolutionary rate matrix regimes fitted to tree. Larger values of 'nu' will make the prior distribution closer around 'Sigma' and small values will increase the variance. This parameter is analogous to the variance parameter of a normal distribution, however, it has an inverted relationship.

Details

This function is integrated within the 'ratematrixMCMC' function that runs the MCMC chain. However, this implementation allows for more control over the prior distribution for the analysis. The prior functions produced here can be easily passed to the 'ratematrixMCMC' function. See examples and more information in the 'ratematrixMCMC' function.

One can use the output of this function in order to sample from the prior using the 'samplePrior' function. A sample from the prior can be set as the starting point of the MCMC sampler.

Independent priors are defined for the phylogenetic mean, the vector of standard deviations and the structure of correlation, allowing for a wide range of configurations. Priors for the phylogenetic mean and the standard deviations can be uniform or normal (lognormal in the case of the standard deviations). The prior on the matrix of correlations is distributed as an inverse-Wishart and can be set to a marginally uniform prior or to be centered around a given variance-covariace matrix.

The prior for the model has two elements, one is the vector of phylogenetic means (or the root values) and the other is the evolutionary rate matrices (the vcv matrices for the rate of the multivariate BM model). The vector of root values can be distributed as any continuous distribution. In this implementation the two options are the uniform and the normal distribution. On the other hand, the prior distribution for the rate matrices need to be more elaborated. Here we divide the variance-covariance matrix into two elements, a correlation matrix and a vector of standard deviations. Standard deviations can be modelled as any continuous distrbuted of positive values. Here we use a uniform or a log-normal distribution. The correlation matrix need to be derived from a distribution of covariance matrices known as the inverse-Wishart. The inverse-Wishart is controlled by two parameters; the scale matrix (Sigma) and the degrees of freedom (nu). Any variance-covariance matrix can be used as the scale matrix. To set a marginally uniform prior for the correlation structure of the evolutionary rate matrices sampled for the model one need to set 'Sigma' as an identity matrix and 'nu' as the dimension of the matrix +1. This is performed automatically by the function when the option 'unif.corr' is set to TRUE.

Value

List of density functions to compute the log prior probability of parameter values.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
data( centrarchidae )
## Set the limits of the uniform prior on the root based on the observed traits
data.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( data.range[,2] - data.range[,1] ) / 10
## Set a reasonable value for the uniform prior distribution for the standard deviation.
## Here the minimum rate for the traits is 0 and the maximum is 10 ( using 'sqrt(10)' to
##      transform to standard deviation).
par.sd <- cbind(c(0,0), sqrt( c(10,10) ))
## The proposal step on the standard deviation is a multiplier. So 0.2 is good enough for
##     most cases.
w_sd <- matrix(0.2, ncol = 2, nrow = 2)
prior <- makePrior(r = 2, p = 2, den.mu = "unif", par.mu = data.range, den.sd = "unif"
                   , par.sd = par.sd)
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, prior=prior
                         , gen=50000, w_mu=w_mu, w_sd=w_sd, dir=tempdir())
posterior <- readMCMC(handle, burn = 0.2, thin = 10)
plotRatematrix( posterior )
```

---

mergePosterior *Merge posterior distributions*

Description

Join two or more independent MCMC chains from the same data and phylogenetic trees by appending them together into a single chain.

Usage

```
mergePosterior(...)
```

Arguments

...	any number of posterior distributions as produced by the function 'readMCMC'.

Value

A merged posterior distribution in the same format.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
data( centrarchidae )
## Set the limits of the uniform prior on the root based on the observed traits
data.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( data.range[,2] - data.range[,1] ) / 10
## Set a reasonable value for the uniform prior distribution for the standard deviation.
## Here the minimum rate for the traits is 0 and the maximum is 10 ( using 'sqrt(10)' to
##      transform to standard deviation).
par.sd <- cbind(c(0,0), sqrt( c(10,10) ))
## The proposal step on the standard deviation is a multiplier. So 0.2 is good enough
##       for most cases.
w_sd <- matrix(0.2, ncol = 2, nrow = 2)
prior <- makePrior(r = 2, p = 2, den.mu = "unif", par.mu = data.range, den.sd = "unif"
                   , par.sd = par.sd)
## Run multiple MCMC chains.
handle.list <- lapply(1:4, function(x) ratematrixMCMC(data=centrarchidae$data
                      , phy=centrarchidae$phy.map, prior=prior, gen=10000
                      , w_mu=w_mu, w_sd=w_sd, dir=tempdir()) )
## Read all to a list
posterior.list <- lapply(handle.list, readMCMC)
## Merge all posteriors in the list.
merged.four <- mergePosterior(posterior.list)
## Merge some of the posteriors.
merged.two <- mergePosterior(posterior.list[[1]], posterior.list[[3]])
```

`mergeSimmap` *Merge two or more regimes of a 'simmap' tree*

Description

Function will merge stochastic mapped regimes together to form a new regime. This can be used to decrease the number of regimes in the phylogeny. Additionally, the function can drop all regimes and return a phylogeny of the class 'phylo'.

Usage

```
mergeSimmap(phy, merge.regimes = NULL, new.regime = NULL,
  drop.regimes = FALSE)
```

Arguments

`phy`	a phylogeny of the 'simmap' format.
`merge.regimes`	a vector with the names of the regimes to be merged.
`new.regime`	the name of the new regime.
`drop.regimes`	whether to simply drop all information about the regimes and return a phylogeny of class 'phylo'.

Details

The distribution of the regimes across the tree will not change. The function only modify the labels of the regimes such that two or more regimes become one (with a new label).

Function takes the elements of the 'merge.regimes' vector and collapse all those regimes into a single one. The branch length associated with 'merge.regimes' are summed and assigned to the regime correspondent to the first element of the 'merge.regimes' vector. Then this new regime is renamed as 'new.regime'.

If the original phylogeny has only two regimes or if 'drop.regimes' is set to TRUE, then the output will be of class 'phylo' with no regime information.

Value

A phylogeny of the format 'simmap' with merged regimes or a phylogeny of class 'phylo' with no regime information.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
library( phytools ) ## Need phytools for this example.
data(centrarchidae)
plot( centrarchidae$phy.map )
class( centrarchidae$phy.map )
## Now drop all regime information:
no.regime.phy <- mergeSimmap(centrarchidae$phy.map, drop.regimes=TRUE)
plot( no.regime.phy )
class( no.regime.phy )
## Create a new regime with three states:
dt <- c(rep( c("water","earth"), each=10 ), rep("fire", times=7))
names(dt) <- no.regime.phy$tip.label
map.phy <- phytools::make.simmap(tree=no.regime.phy, x=dt)
plot( map.phy )
## Merge two regimes:
merged.phy <- mergeSimmap(phy=map.phy, merge.regimes=c("water","earth"), new.regime="mud")
plot( merged.phy )
```

---

plotPrior *Plot the prior distribution used in the MCMC analysis*

Description

Function plots the prior distribution used in the MCMC analysis.

Usage

```
plotPrior(handle, n = 1000, root = FALSE, color = "black", ...)
```

Arguments

handle	the output object from the 'ratematrixMCMC' function.
n	number of samples from the prior to be plotted (default is 1000).
root	whether to plot the prior for the root value instead of the evolutionary rate matrix (default is FALSE).
color	color for the plot (default is "black").
...	other parameters to be passed to the function 'plotRatematrix' or 'plotRootValue'. See help page for list of possible parameters.

Details

Function will make a plot of the prior for the evolutionary rate matrix by default. One can plot the prior for the root value instead by setting 'root' to TRUE.

The prior distribution often has a different range of parameter values when compared to the posterior distribution. Depending on the prior configuration the range of the prior can be orders of magnitude larger than the posterior distribution. In this case, it is important to observe the scale of the x axis when comparing the prior and the posterior distribution. One can use the 'set.xlim' parameter to restrict the x axis for plotting the prior to be similar to the posterior distribution. However, often the region of parameter space of the posterior distribution has a low likelihood under the prior. This results in problems to take samples from that region to make the plot. This problem can be identified when the 'set.xlim' argument is changed and the plot shows only a few samples.

Value

A plot similar to 'plotRatematrix'.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
## Load data
data(centrarchidae)
## Run MCMC. This is just a very short chain.
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, gen=1000
                         , dir=tempdir())
## Load posterior distribution, make plots and check the log.
posterior <- readMCMC(handle, burn=0.25, thin=1)
plotRatematrix(posterior)
plotRootValue(posterior)
plotPrior(handle)
```

```
plotPrior(handle, root=TRUE)
logAnalyzer(handle)
```

---

plotRatematrix *Plot the distribution of evolutionary rate matrices*

Description

Generates a plate with plots showing the posterior distribution of evolutionary rate matrices.

Usage

```
plotRatematrix(chain, p = NULL, colors = NULL, set.xlim = NULL,
  set.leg = NULL, l.cex = 0.7, ell.wd = 0.5, alphaOff = 1,
  alphaDiag = 1, alphaEll = 1, hpd = 100, show.zero = FALSE,
  n.lines = 50, n.points = 200, point.matrix = NULL, point.color = NULL,
  point.wd = 0.5)
```

Arguments

chain	the posterior distribution of parameter estimates as produced by 'readMCMC' or samples from the prior using 'samplePrior'..
p	a numeric vector with the regimes to be plotted. This parameter can be used to subset the rate regimes to be plotted as well as to control the order of the plotting. If 'NULL' (default), then all rate regimes are plotted in the same order as in the data.
colors	a vector with colors for each rate regime with length equal to the number of regimes or to the number of regimes provided to the argument 'p'. If not provided the function will use pre-selected colors up to 8 regimes.
set.xlim	user limits for the x axes. Need to be a vector with two elements, the minimum and the maximum.
set.leg	user defined legends for the trait names. A character vector with same length as the number of traits in the model.
l.cex	the 'cex' parameter for legends of the plot. See 'help(par)' for more information on 'cex'. Default is 0.7 .
ell.wd	a number for the width of the ellipse lines. Default is 0.5 .
alphaOff	a number between 0 and 1 with the transparency of the color used for the off-diagonal plots. Default is 1.
alphaDiag	a number between 0 and 1 with the transparency of the color used for the diagonal plots. Default is 1.
alphaEll	a number between 0 and 1 with the transparency of the color used for the lines of the ellipse plots. Using transparency in the lines might enhance the visualization of regions with more or less density of samples. Default is 1.

hpd	a number between 0 and 100 to set the proportion of the highest posterior density (HPD) to be highlighted in the plot.
show.zero	whether to plot a thin blue line showing the position of the 0 value on the histograms.
n.lines	number of lines to be displayed in the ellipse plots. Default is 50 lines.
n.points	number of points used to approximate the shape of the ellipses.
point.matrix	optional argument. A list of variance-covariance matrices with length equal to p. If p=NULL then length need to be equal to the number of rate regimes fitted to the data. Each element of the list will be plotted as a single line on top of the distribution of parameter estimates.
point.color	optional argument. A vector with color names for the matrices set in 'point.matrix'. The vector need to have same length as 'point.matrix. If not provided, the colors of the lines will be equal to the colors of the distribution (argument 'colors').
point.wd	optional argument. The width of the lines plotted using 'point.matrix'. Default is 0.5 .

Details

The function provides the option to plot a single evolutionary rate matrix on top of the posterior distribution of each regime as a vertical line on the upper-diagonal and diagonal histogram plots and as an ellipse on the lower-diagonal plots. This can be set using the argument 'point.matrix' (as well as the 'point.color' and 'point.wd' options). One can use this option to contrast the posterior with some point estimate or summary statistics.

Colors can be provided either as color names recognized by R-base plot functions or in the HEX format.

The lines showed by the ellipse plots (lower-diagonal) are a sample from the posterior distribution. The user can set the number of lines plotted using the argument 'n.lines'. Note that more lines will take more time to plot.

The 'hpd' argument can be used to set some regions of the plot to be colored in white. For example, if 'hpd=95' the histograms will plot the region outside the 95% HPD (Highest Posterior Density) in white and ellipse lines will only be showed if within this 95% HPD interval. If the region chosen is too small (~10% or lower), the plot might return an error. This happens because the function take random samples from the posterior distribution to plot as ellipse lines and exclude the samples that are outside the defined HPD interval. If this happens, try to choose a more inclusive percentage or increase the number of samples taken for the ellise lines (see argument 'n.lines') or repeat the plot until sucessful. [A better solution for this issue will be provided soon.] The default is 100 (no highlight is performed and ellipse lines are not restricted).

The plots are divided into three groups. Upper-diagonal plots show histograms with the posterior distribution for the covariance values between each pairwise combination of the traits. Plot in the diagonal show histograms with the posterior distribution of evolutionary rates for each trait. Plots on the lower-diagonal slots show a collection of ellipses sampled from the posterior distribution of the model. Each ellipse line represents a bivariate distribution for the 95
Lower-diagonal plots are ideal to visualize the evolutionary correlation and variance between two

traits. The orientation of the ellipses show whether there is a positive, negative or lack of correlation (horizontal or vertical orientation) between traits. The shape of the ellipses show the major axis of variation between traits. A 'cigar-shaped' ellipse indicates that one of the traits show faster evolutionary rates than the other, so one axis of variation is much larger than the other whereas a more circular (round) ellipse is a result of comparable rates of evolution between the two traits. A completely circular shape denotes lack of evolutionary correlation between two traits. It might help to understand the meaning of the ellipses lines by imagining each ellipse line marks the spread of the dots in a scatterplot with data generated with a particular covariance value (i.e., the covariance value the ellipse is representing).

Value

A plate with a grid of plots with dimension equal to the number of traits fitted to the data.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
data( centrarchidae )
dt.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( dt.range[,2] - dt.range[,1] ) / 10
par.sd <- cbind(c(0,0), sqrt( c(1,1) ))
prior <- makePrior(r=2, p=2, par.mu=dt.range, par.sd=par.sd)
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, prior=prior
                         , gen=50000, w_mu=w_mu, dir=tempdir())
posterior <- readMCMC(handle, burn = 0.2, thin = 10)
plotRatematrix( posterior )
plotRatematrix( posterior, colors = c("black","red"))
plotRatematrix( posterior, colors = c("black","red"), alphaOff = 0.5)
plotRatematrix( posterior, colors = c("black","red"), alphaOff = 0.5, alphaDiag = 0.5)
plotRatematrix( posterior, colors = c("black","red"), alphaOff = 0.5, alphaDiag = 0.5)
ref.matrix <- list( rbind(c(0.5,0),c(0,0.5)), rbind(c(0.5,0),c(0,0.5)) )
plotRatematrix( posterior, colors = c("black","red"), alphaOff = 0.5, alphaDiag = 0.5
               , point.matrix = ref.matrix)
plotRatematrix( posterior, colors = c("black","red"), alphaOff = 0.5, alphaDiag = 0.5
               , point.matrix = ref.matrix, point.color = "orange", point.wd = 3)
plotRatematrix( posterior, colors = c("black","red"), alphaOff = 0.5, alphaDiag = 0.5
               , point.matrix = ref.matrix, point.color = "orange", point.wd = 3
               , alphaEll = 0.05, n.lines = 2000)
plotRatematrix( posterior, colors = c("black","red"), alphaOff = 0.5, alphaDiag = 0.5
               , point.matrix = ref.matrix, point.color = "orange", point.wd = 3
               , alphaEll = 0.5, n.lines = 200, hpd = 90)
```

plotRootValue *Plot posterior distribution of root values for the traits*

Description

Plot the posterior distribution of root values sampled from the MCMC analysis or samples from the prior distribution.

Usage

```
plotRootValue(chain, color = "black", set.xlab = NULL, set.cex.lab = 1,
  set.cex.axis = 1.5, set.xlim = NULL, hpd = 100, mfrow = 1,
  vline.values = NULL, vline.color = NULL, vline.wd = NULL,
  show.zero = FALSE)
```

Arguments

chain	the posterior distribution loaded from the files using 'readMCMC' or samples from the prior generated with the 'samplePrior' function.
color	the color for the histograms.
set.xlab	a vector with legends for the x axes. If 'NULL' (default), the names are 'trait_1' to 'trait_n".
set.cex.lab	the cex value for the labels (default is 1).
set.cex.axis	the cex value for the axes numbers (default is 1.5).
set.xlim	the xlim for the plot. Need to be a vector with the lower and higher bound.
hpd	the Highest Posterior Density interval to highlight in the plot. Parameter values outside this interval will be colored in white. A numeric value between 0 and 100 (default is 100).
mfrow	the number of rows to use in the figure (default is 1).
vline.values	numeric values for plotting vertical lines. Can be a single value recycled for each of the plots or a vector with length equal to the number of traits.
vline.color	character vector with colors for the vertical lines. Can be a single color if length of 'vline.values' is 1 otherwise need to have length equal to the number of traits.
vline.wd	numeric value for the width of the vertical lines. Can be a single value if length of 'vline.values' is 1 otherwise need to have length equal to the number of traits.
show.zero	whether a vertical line should be plotted showing the position of the value 0 in the plot.

Value

A plot with the posterior density of root values or distribution of root values sampled from the prior.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
data( centrarchidae )
dt.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( dt.range[,2] - dt.range[,1] ) / 10
par.sd <- cbind(c(0,0), sqrt( c(10,10) ))
prior <- makePrior(r=2, p=2, den.mu="unif", par.mu=dt.range, den.sd="unif", par.sd=par.sd)
prior.samples <- samplePrior(n = 1000, prior = prior)
start.point <- samplePrior(n=1, prior=prior)
## Plot the prior. Red line shows the sample from the prior that will set the starting
##      point for the MCMC.
plotRatematrix(prior.samples, point.matrix = start.point$matrix, point.color = "red"
               , point.wd = 2)
plotRootValue(prior.samples)
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, prior=prior
                         , gen=10000, w_mu=w_mu, dir=tempdir())
posterior <- readMCMC(handle, burn = 0.2, thin = 10)
## Again, here the red line shows the starting point of the MCMC.
plotRatematrix( posterior, point.matrix = start.point$matrix, point.color = "red"
               , point.wd = 2)
plotRootValue(posterior)
```

---

print.ratematrix_multi_chain

Print method for the "ratematrix_multi_chain" class.

Description

Print method for the "ratematrix_multi_chain" class.

Usage

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

Arguments

x	The object.
...	Additional arguments. Not used here.

Details

Print information about object.

`print.ratematrix_multi_mcmc` *Print method for the "ratematrix_multi_mcmc" class.*

Description

Print method for the "ratematrix_multi_mcmc" class.

Usage

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

Arguments

`x`	The object.
`...`	Additional arguments. Not used here.

Details

Print information about object.

`print.ratematrix_prior_function` *Print method for the "ratematrix_prior_function" class.*

Description

Print method for the "ratematrix_prior_function" class.

Usage

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

Arguments

`x`	The object.
`...`	Additional arguments. Not used here.

Details

Print information about object.

print.ratematrix_prior_sample *Print method for the "ratematrix_prior_sample" class.*

Description

Print method for the "ratematrix_prior_sample" class.

Usage

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

Arguments

x	The object.
...	Additional arguments. Not used here.

Details

Print information about object.

print.ratematrix_single_chain *Print method for the "ratematrix_single_chain" class.*

Description

Print method for the "ratematrix_single_chain" class.

Usage

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

Arguments

x	The object.
...	Additional arguments. Not used here.

Details

Print information about object.

`print.ratematrix_single_mcmc` *Print method for the "ratematrix_single_mcmc" class.*

Description

Print method for the "ratematrix_single_mcmc" class.

Usage

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

Arguments

`x`	The object.
`...`	Additional arguments. Not used here.

Details

Print information about object.

`ratematrix` *ratematrix.*

Description

Package to estimate the evolutionary rate matrix for multiple regimes fitted to the phylogenetic tree using Bayesian Markov-chain Monte Carlo.

`ratematrixMCMC` *Estimate the evolutionary rate matrix using Markov-chain Monte Carlo*

Description

Function runs a MCMC chain to estimate the posterior distribution of the evolutionary rate matrix (R) and the root value (phylogenetic mean). Prior distribution and starting state for the chain can be chosen among pre-defined options or manually set by the user using accompanying functions (see function 'makePrior' for more information). User NEED to provide a directory to write the files (See 'Details'). Use dir="." option to write files to the current directory or provide a name of a folder to be created.

Usage

```
ratematrixMCMC(data, phy, prior = "uniform_scaled", start = "prior_sample",
  gen, v = 50, w_sd = 0.2, w_mu = 0.5, prop = c(0.05, 0.475, 0.475),
  dir = NULL, outname = "ratematrixMCMC", IDlen = 5, save.handle = TRUE)
```

Arguments

data	a matrix with the data. Species names need to be provided as rownames (rownames(data) == phy$tip.label). Each column is a different trait. Names for the columns is used as trait labels. If labels are not provided, the function will use default labels.
phy	a phylogeny of the class "simmap" with the mapped regimes for two or more R regimes OR a phylogeny of the class "phylo" for a single regime. The number of evolutionary rate matrices fitted to the phylogeny is equal to the number of regimes in 'phy'. Regime names will also be used. 'phy' can also be a list of phylogenies. See 'Details'.
prior	the prior densities for the MCMC. Must be one of "uniform", "uniform_scaled" (the default, see 'Details'), "empirical_mean", or the output of the "makePrior" function. See more information on 'makePrior' and in the examples below.
start	the starting state for the MCMC chain. Must be one of "prior_sample" (the default), "mle", or a sample from the prior generated with the "samplePrior" functions.
gen	number of generations for the chain.
v	value for the degrees of freedom parameter of the inverse-Wishart proposal distribution for the correlation matrix. Smaller values provide larger steps and larger values provide smaller steps. (Yes, it is counterintuitive.) This needs to be a single value applied to all regimes or a vector with the same length as the number of regimes.
w_sd	the multiplying factor for the multiplier proposal on the vector of standard deviations. This can be a single value to be used for the sd of all traits for all regimes or a matrix with number of columns equal to the number of regimes and number of rows equal to the number of traits. If a matrix, then each element will be used to control the correspondent width of the standard deviation.
w_mu	value for the width of the sliding window proposal for the vector of root values (phylogenetic mean). This can be a single value to be used for the root value of all traits or a vector of length equal to the number of traits. If a vector, then each element will be used as the width of the proposal distribution for each trait in the same order as the columns in 'data'. When 'prior="uniform_scaled"' (the default) this parameter is computed from the data.
prop	a numeric vector of length 3 with the proposal frequencies for each parameter of the model. The vector need to sum to 1. These values are the probability that the phylogenetic mean (prop[1]), the vector of standard deviations (prop[2]), and the correlation matrix (prop[3]) will be updated at each step of the MCMC chain, respectively. Default value is 'c(0.05, 0.475, 0.475)'.

`dir`	path of the directory to write the files. Has no default value (due to RCran policy). The path can be provided both as relative or absolute. It should accept Linux, Mac and Windows path formats.
`outname`	name for the MCMC chain (default is 'ratematrixMCMC'). Name will be used in all the files alongside a unique ID of numbers with length of 'IDlen'.
`IDlen`	length of digits of the numeric identifier used to name output files (default is 5).
`save.handle`	whether the handle for the MCMC should be saved to the directory in addition to the output files.

Details

The MCMC chain works by proposing values for the evolutionary rate matrices (R) fitted to the tree and the vector of root values (or phylogenetic mean). The proposal for the R matrices works by separating the variance-covariance matrix into a correlation matrix and a vector of standard deviations and making independent proposals for each. This scheme is called the 'separation strategy' and significantly improves the mix of the chain and also provide a intuitive distinction between the evolutionary correlation among the traits (correlation matrix) and the rates of evolution (standard deviation vector). The proposal for the root values are made all in a single step.

The function will print a series of messages to the screen. Those provide details of the setup of the chain, the unique identifier for the files and the log-likelihood of the starting value of the chain. Up to now these messages cannot be disabled.

DIRECTORY TO WRITE FILES: User need to specify the directory to write the files. Use "." to write to the current directoy. Or, for example, use "MCMC_files" to create the folder "MCMC_files" in the current directory. RCran policy prohibits the package to automaticaly write to the current directory.

DEFAULT PRIOR: The default prior distribution ('uniform_scaled') is composed by a uniform distribution on the root values with range equal to the range observed at the tip data. The size of the window used at each proposal step for the root values is equal to the width of the prior divided by 10 units. For the evolutionary rate matrix, this prior sets a uniform distribution on the correlations (spanning all possible correlation structures) and also a uniform distribution on the vector of standard deviations. The limits of the prior on the standard deviation is computed by first doing a quick Maximum Likelihood estimate of each trait under a single rate BM model and using the results to inform the magnitude of the rates. This default prior distribution might not be the best for your dataset. Keep in mind that the default behavior of the MCMC is to draw a starting point from the prior distribution. Please check the 'makePrior' function for more information on priors and how to make a custom prior distribution.

SAMPLE OF TREES: The MCMC chain can integrate the phylogenetic uncertainty or the uncertainty in the rate regimes by randomly sampling a phylogenetic tree from a list of trees. To activate this option, provide a list of 'simmap' or 'phylo' trees as the 'phy' argument. The MCMC will randomly sample a tree each proposal step. Check the 'logAnalyzer' function for more information.

MCMC DOES NOT START: It is possible that the starting point shows a very low likelihood value, resulting in the collapse of the chain. This might be a result of a random sample from a very unlikely region of the prior. We suggest that another sample of the prior is taken or that the user make

a more suitable prior using the function 'makePrior'.

MCMC DOES NOT CONVERGE OR MIX: If the MCMC is taking too long to converge then the parameters of the chain might not be good for your data. First check the 'logAnalyzer' function as well as the 'computeESS'. The recommended acceptance ratio is ~ 0.24, if it is too high, then the step size of the proposals might be too small, try increasing the step size. In contrast, low acceptance ratio might be due to step sizes too large. Try to decrease the size of the steps. If the effective sample size (ESS) for the chain (see 'checkConvergence' and 'computeESS' functions) is to low for some parameter, then try to increase the proportion of times that the parameter is proposed in the MCMC.

CANNOT FIND THE POSTERIOR: The function writes the posterior into two files: The '.log' file has the log-likelihood and information about which phylogeny was used, which parameter was proposed and whether the step was accepted or not. The '.mcmc' file has the posterior for the parameters of the model. Those are identified by a name for the chain set by "outname" and an unique series of numbers set by "IDlen". Note that you will need the handle object provided as the output for the function (or saved to the directory if 'save.handle' is TRUE) to be able to load, plot and analyze the posterior distribution.

Value

Function returns the 'handle' object and writes the posterior distribution and log as files in the directory (see 'dir'). The handle is a list with the details of the MCMC chain. It is composed by: *k* the number of traits; *p* the number of R regimes fitted to the tree; *ID* the unique identifier of the run; *dir* the directory where the posterior and log files were saved; *outname* the name for the chain; *trait.names* a vector with the label for the traits; *regime.names* a vector with the label for the rate regimes; *data* the data used in the analysis; *phy* a single phylogeny or the list of phylogenies; *prior* a list with the prior functions; *start* a list with the starting parameters for the chain; *gen* the number of generations for the chain; *mcmc.par* a list with the tunning parameters for the MCMC.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
data( centrarchidae )
## Set the limits of the uniform prior on the root based on the observed traits
data.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( data.range[,2] - data.range[,1] ) / 10
## Set a reasonable value for the uniform prior distribution for the standard deviation.
## Here the minimum rate for the traits is 0 and the maximum is 10 ( using 'sqrt(10)' to
##      transform to standard deviation).
par.sd <- cbind(c(0,0), sqrt( c(10,10) ))
## The proposal step on the standard deviation is a multiplier. So 0.2 is good enough
##       for most cases.
w_sd <- matrix(0.2, ncol = 2, nrow = 2)
```

```
prior <- makePrior(r = 2, p = 2, den.mu = "unif", par.mu = data.range, den.sd = "unif"
                   , par.sd = par.sd)
## Run multiple MCMC chains.
handle.list <- lapply(1:4, function(x) ratematrixMCMC(data=centrarchidae$data
                     , phy=centrarchidae$phy.map, prior=prior, gen=10000
                     , w_mu=w_mu, w_sd=w_sd, dir=tempdir()) )
## Read all to a list
posterior.list <- lapply(handle.list, readMCMC)
## Check for convergence (it might not converge with only 10000 steps)
checkConvergence(posterior.list)
## Merge all posteriors in the list.
merged.posterior <- mergePosterior(posterior.list)
## PLot results:
plotRatematrix(merged.posterior)
plotRootValue(merged.posterior)
```

`readMCMC` *Read the MCMC output files*

Description

Reads the output files from the MCMC with the posterior distribution of the chains.

Usage

```
readMCMC(handle, burn = 0.25, thin = 100, dir = NULL)
```

Arguments

`handle`	the output object from the 'ratematrixMCMC' function.
`burn`	the proportion of the burnin to be pruned from the MCMC chain. A number between 0 and 1 (default is 0.25).
`thin`	the thinning of the posterior distribution. A number with the interval between each MCMC step to be kept in the posterior distribution (default is 100).
`dir`	directory with the output files. If 'NULL' (default), then files are read from the directory chosen when running the MCMC chain using the argument 'dir' of the 'ratematrixMCMC' function (stored on handle). Otherwise function will read files from 'dir'.

Value

List with the MCMC chain for the phylogenetic mean (root value) and evolutionary rate matrices (R). *root* are the values for the phylogenetic mean in matrix format; *matrix* is a list of length equal to the number of rate regimes fitted to the tree, each of those are lists with the chain of respective R matrices.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
## Load data
data(centrarchidae)
## Run MCMC. This is just a very short chain.
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, gen=1000
                         , dir=tempdir())
## Load posterior distribution, make plots and check the log.
posterior <- readMCMC(handle, burn=0.25, thin=1)
plotRatematrix(posterior)
plotRootValue(posterior)
plotPrior(handle)
plotPrior(handle, root=TRUE)
logAnalyzer(handle)
```

---

samplePrior *Take samples from the prior distribution*

Description

Take samples from the prior distribution.

Usage

```
samplePrior(n, prior, sample.sd = TRUE, rebuild.R = FALSE)
```

Arguments

n	number of samples to be generated.
prior	the object with the prior function. See 'makePrior' for more information.
sample.sd	whether the function should sample the vector of standard deviations independently from the correlation matrices. See 'Details'.
rebuild.R	whether the prior sample should return an evolutionary rate matrix rather than a correlation matrix and a vector of standard deviations (default is FALSE). See 'Details'.

Details

The prior samples from this function can be used to start the MCMC sampler. See the examples below.

If 'sample.sd' is set to FALSE the samples from the standard deviations will be derived from the

covariance matrices. If 'sample.sd' is set to TRUE (default) then standard deviations are independently sampled from their own prior distribution and are not derived from the samples of the correlation matrix. Option 'sample.sd = TRUE' is the one used during the MCMC.

The option 'rebuild.R' controls if the samples from the posterior distribution should return the standard deviation separated from the correlation matrix or if these elements should be used to rebuild the covariance matrix. Set 'rebuild.R' to TRUE if you want to obtain the covariance matrices. Otherwise, the 'plotPrior' function works better when 'rebuild.R' is set to FALSE.

Value

A list with samples from the prior distribution. The structure of this list is the same as required by the parameter 'start' of the 'ratematrixMCMC'.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
data( centrarchidae )
dt.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( dt.range[,2] - dt.range[,1] ) / 10
par.sd <- cbind(c(0,0), sqrt( c(10,10) ))
prior <- makePrior(r=2, p=2, den.mu="unif", par.mu=dt.range, den.sd="unif", par.sd=par.sd)
prior.samples <- samplePrior(n = 1000, prior = prior)
start.point <- samplePrior(n=1, prior=prior)
## Plot the prior. Red line shows the sample from the prior that will set the starting
##        point for the MCMC.
plotRatematrix(prior.samples, point.matrix = start.point$matrix, point.color = "red"
               , point.wd = 2)
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, prior=prior
                         , gen=10000, w_mu=w_mu, dir=tempdir())
posterior <- readMCMC(handle, burn = 0.2, thin = 10)
## Again, here the red line shows the starting point of the MCMC.
plotRatematrix( posterior, point.matrix = start.point$matrix, point.color = "red"
               , point.wd = 2)
```

`simRatematrix` *Simulates multivariate trait evolution using a Brownian motion model*

Description

Simulates correlated traits under a multivariate Brownian motion model. The function uses a covariance matrix (evolutionary rate matrix-R) to indicate the rates of the traits.

Usage

```
simRatematrix(tree, vcv, anc = NULL, internal = FALSE)
```

Arguments

tree	a phylogenetic tree of 'phylo' format.
vcv	a variance covariance matrix (the evolutionary rate matrix).
anc	a vector of the same length as the number of traits to be simulated (same as the dimension of the 'vcv' matrix). This is used as the values for the root in the simulations. If 'NULL' then all traits have root value of 0.
internal	whether to return the values simulated for the nodes in the phylogeny. If FALSE (default), then only returns the values simulated for the tips.

Details

This is a function derived from 'sim.corrs' in the package 'phytools'. This version has some edits to make the simulations more efficient for this particular use. For all other applications please refer to the original implementation of 'sim.corrs' in the package 'phytools' wrote by Liam Revell.

Value

Returns a matrix with each trait values for the tips. Traits are distributed in the rows and tips are distributed in the columns.

Author(s)

Daniel S. Caetano and Luke J. Harmon

References

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

---

testRatematrix *Test for difference between evolutionary rate matrix estimates*

Description

Function uses summary statistics to test for differences between the posterior distribution of parameter estimates for the evolutionary rate matrix regimes.

Usage

```
testRatematrix(chain, par = c("all", "correlation", "rates"),
  median.test = FALSE, regimes = NULL, plot = FALSE)
```

Arguments

chain	the posterior distribution of parameter estimates as output of the function 'readMCMC'.
par	the attribute of the rate matrices that are checked by the test. One of 'all', 'correlation', or 'rates' (default is 'all'). Choose 'all' to compute the summary statistics for the overall matrix. Choose 'rates' to check the rates of evolution among the traits. Choose 'correlation' to compute the summary statistics for the evolutionary correlations.
median.test	whether to return a median of the summary statistics across all elements of the evolutionary rate matrices. The default is FALSE.
regimes	a numeric vector or character vector. This is the set of regimes to be compared. If numeric, then the regimes are in the same order as in the 'chain' argument. If character, than names need to match the names of the rate regimes fitted to the phylogenetic tree.
plot	whether to plot the results of the summary statistics test applied to every element of the matrix. Default is FALSE.

Details

This functions performs a test to check whether the posterior distribution of the fitted matrices are different. It returns the proportion of overlap between regimes. When this proportion is less than 0.05 this means that the posterior distribution of the elements of the evolutionary rate matrices does not overlap more than 5%. This test statistics is NOT a p value! This is not an estimate of the probability of deviance from a null distribution. It assumes that when the posterior distribution of two or more paramaters do not overlap, then there is a relevant difference between the parameters.

The test can be performed using the median overlap of the posterior distribution across all elements of the ratematrix or by contrasting each element separatelly. Checking each element independently provides more information. Using the median overlap will result in a single value returned, but it can be insensitive to important changes in the evolutionary rate matrices between regimes. When a posterior distribution with more than two rate regimes is fitted to the data, the function performs tests for all pairwise combinations.

Value

Return a matrix or a list with the value of the test statistics.

Author(s)

Daniel S. Caetano and Luke J. Harmon

Examples

```
data( centrarchidae )
dt.range <- t( apply( centrarchidae$data, 2, range ) )
## The step size for the root value can be set given the range we need to sample from:
w_mu <- ( dt.range[,2] - dt.range[,1] ) / 10
```

```
par.sd <- cbind(c(0,0), sqrt( c(1,1) ))
prior <- makePrior(r=2, p=2, par.mu=dt.range, par.sd=par.sd)
handle <- ratematrixMCMC(data=centrarchidae$data, phy=centrarchidae$phy.map, prior=prior
                         , gen=50000, w_mu=w_mu, dir=tempdir())
posterior <- readMCMC(handle, burn = 0.2, thin = 10)
testRatematrix(posterior, par = "all")
testRatematrix(posterior, par = "correlation")
testRatematrix(posterior, par = "rates")
testRatematrix(posterior, par = "correlation", plot = TRUE)
```

Index