Set custom starting point with 'ratematrix'

Daniel S. Caetano

Set custom starting point with 'ratematrix'

Here we are going to set a custom starting point distribution for the Markov-chain Monte Carlo (MCMC) run. Please note that model estimation is based on MCMC, so the analyses described here might take some time to run.

General comments

The default behavior of the 'ratematrix' package is randomly draw staring points from the prior do the model. This is defined by the argument `start` of the function `ratematrixMCMC`. This argument accepts three possible values: `"prior_sample"`, `"mle"`, and a list object with a custom starting point. In this tutorial I will show how to set up this list object with a custom starting point. As an exercise we will use the best posterior value from a previous MCMC chain to start a new chain.

Structure of the starting point list

The starting point is a object of type list with three elements: a vector of root values, a correlation matrix, and a vector of variance elements. The correlation matrix and the vector of variance elements will be used to recompose the evolutionary rate matrix (**R**).

Here we will take a sample from a prior distribution using the function `samplePrior` to explore the format of the starting point used in the package.

Load package:

```
library( ratematrix )
```

Generate a sample from the prior:

```
data( "centrarchidae" )
## Using a function to drop all the regimes in the phylogeny to create a simpler case.
phy <- mergeSimmap( centrarchidae$phy.map, drop.regimes = TRUE)
par_mu <- t( apply(centrarchidae$data, 2, range) )
prior <- makePrior(r = 2, p = 1, par.mu = par_mu, par.sd = c(0, sqrt(10)))
one.regime.sample <- samplePrior(n = 1, prior = prior)
```

Note that the result is a list with three elements. `one.regime.sample[[1]]` is a vector of root values, `one.regime.sample[[2]]` is a matrix with a correlation matrix and, finally, `one.regime.sample[[3]]` is a vector of variance components. This is the format for a model with 2 traits (`k=2` or `r=2`) and 1 regime (`p=1`). Now let's increase the number of regimes to 2 and see the format of the sample.

```
par.sd <- rbind( c(0, sqrt(10)), c(0, sqrt(10)) )
two.regime.prior <- makePrior(r = 2, p = 2, par.mu = par_mu, par.sd = par.sd)
two.regime.sample <- samplePrior(n = 1, prior = two.regime.prior)
```

Now the format looks a little different, but the structure is the same. The first element of the list is the vector of root values. Since we will always have just one root value for each trait in the model independent of the number of rate regimes, this is still a numeric vector. The second element now is a list. `two.regime.sample[[2]]` is a list with two correlation matrices; one for each fitted regime in the model. Of course, all correlation

matrices for the p regimes need to have the same dimension; with the number of rows and number of columns equal to the number of traits in the model. The third element is also now a list. `two.regime.sample[[3]]` is a list with two elements; the vector of variance components for each of the p regimes of the model. The order of the list elements of `two.regimes.sample[[1]]` and `two.regimes.sample[[2]]` need to match.

This structure holds for any number of regimes and any number of trait in the model. With a larger number of regimes the number of elements will increase.

Order of the regimes in the start point object

The order of the regimes is the same as in the `simmap` format phylogeny. Here I will use the dataset provided with the package as an example:

```
data( anoles )
anoles.phy <- anoles$phy[[1]]
```

Here `anoles.phy` is a `simmap` tree with three regimes. The regimes are `island`, `mainland` and `mainland.2`. The order of the correlation matrices and vectors of variance components is the same; first the correlation and variance associated with `island`, second the one for `mainland`, and third the one for `mainland.2`.

Although the order for the regimes is the same as the `simmap` tree, there is no need to name the elements of the list. As a matter of fact, the package `ratematrix` will not verify if the names of the list match the regimes in the tree. (Although this seems an intelligent thing to implement in the future!)

Worked example

Here I will walk through an application example. Imagine we have a large phylogeny with several regimes. There is a chance that the MCMC will take a long time to reach convergence, because the parameter space and number of dimensions of the model is very large. In this case, starting from a random sample from the prior might not be a good idea, since the starting point for the MCMC might be very far from the region of the highest density of the posterior distribution.

One of the solutions to this problem is to start with the Maximum likelihood estimate (MLE) for the model. We can easily do that by setting the parameter `start = "mle"` in the function `ratematrixMCMC`. This option will make the estimate of the MLE for the model first, then use this estimate as the starting point for the MCMC.

If our data show a pattern in which most of the tree is associated with a single regime (call it the *background regime*) and all other regimes are restricted to isolated, relativelly small clades, there is another possible strategy to improve the convergence of the model. Since the *background regime* is present in the majority of the tree and the fact that we can compute the likelihood for the model by traversing each branch of the tree, we can deduct that most of the likelihood of the model will be driven by this *background regime*. Thus, we could produce a rough guess of where the posterior for the model is in this this vast, multidimentional space by performing a preliminary MCMC analysis with a single regime only.

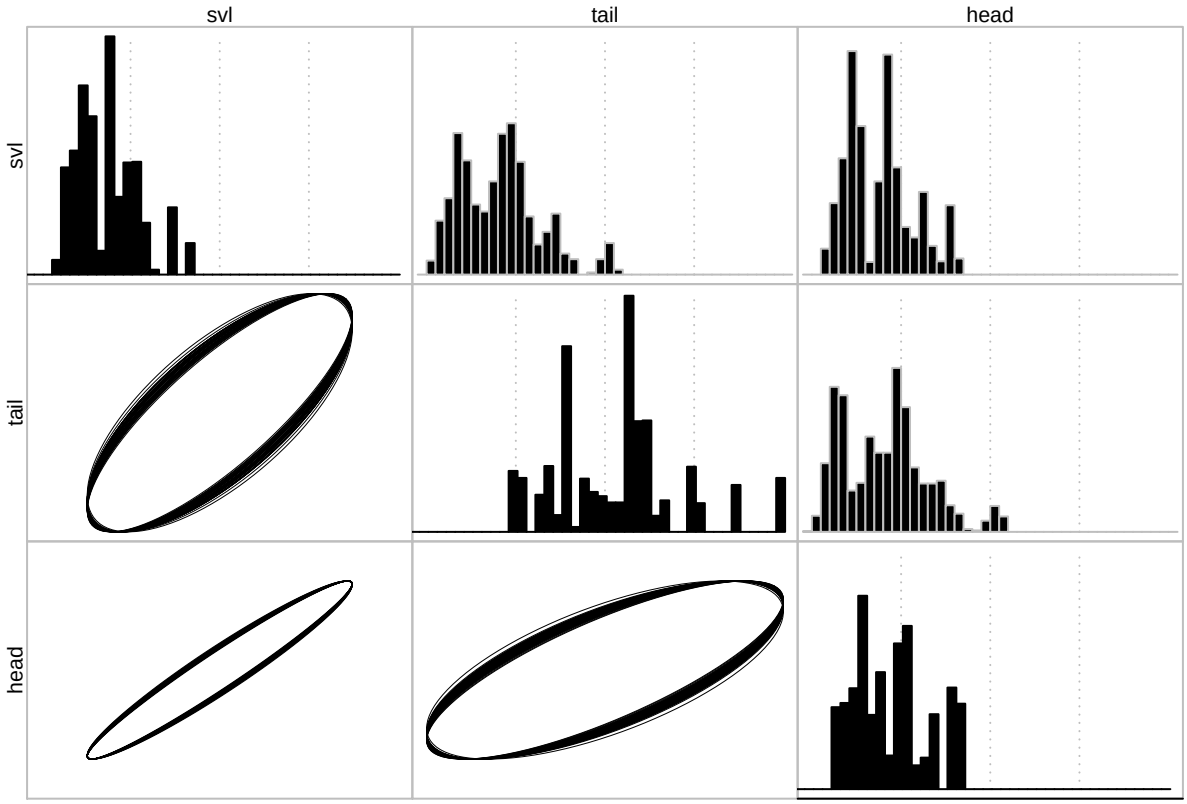
In this example the preliminary MCMC analysis with a single regime will start from the prior distribution and work its way to the posterior distribution of a model that assumes a single evolutionary rate matrix for the whole tree. This posterior will be, roughly, a mean across the evolutionary rate matrices for each p regimes of the full model. Thus, if we take samples from this posterior distribution and use it as a starting point for each of the evolutionary rate matrix regimes, it might be a better initial guess than the default behavior of taking a sample from the prior, specially if the prior is uninformative.

To show how to do this I will perform a short MCMC analysis with a small dataset. However, the procedure would be the same with a larger dataset.

```
anoles.single.regime <- mergeSimmap(phy = anoles.phy, drop.regimes = TRUE)
handle <- ratematrixMCMC(data=anoles$data[,1:3], phy=anoles.single.regime, gen=500000
                         , dir=tempdir() )
mcmc <- readMCMC(handle, thin = 200)
```

```
plotRatematrix(mcmc)
```

```
## Plotting a single regime.
```



It is clear from this plot that the MCMC chain is not yet converged. One should run it for more generations. But this might be good enough for the present tutorial.

Now we can get two random samples from this distribution and use it to set the initial state for the second MCMC run comprising the three regimes (`island`, `mainland`, and `mainland.2`).

Check the structure of the `mcmc` object:

```
mcmc
```

```
##
## Posterior distribution with single regime
## Number of traits:  3
## Number of posterior samples:  1875
##
## Use 'plotRatematrix' and 'plotRootValue' to plot the distribution.
## Use 'checkConvergence' to verify convergence.
## Use 'mergePosterior' to merge two or more posterior chains.
## Check 'names' for more details.
```

```
names( mcmc )
```

```
## [1] "root"   "matrix"
```

```
class( mcmc$root )
```

```
## [1] "matrix"
```

```
dim( mcmc$root )
```

```
## [1] 1875    3
```

Here `mcmc$root` is a matrix with the root value for each trait in the columns and each row is a sample from the posterior distribution.

```
class( mcmc$matrix )
```

```
## [1] "list"
```

```
length( mcmc$matrix )
```

```
## [1] 1875
```

```
mcmc$matrix[[1]]
```

```
##              [,1]        [,2]        [,3]
## [1,] 0.002103129 0.002181801 0.002048703
## [2,] 0.002181801 0.003811547 0.002083268
## [3,] 0.002048703 0.002083268 0.002130959
```

Here `mcmc$matrix` is a list object with number of elements equal to the number of rows of `mcmc$root`. Each element of the list is a evolutionary rate matrix. A variance-covariance matrix that describes the evolutionary pattern among the three traits on the branches of the phylogeny under multivariate Brownian-motion.

Now we get one sample for the root values and a number of random samples equal to the rate regimes:

```
id <- sample(x = 1:length(mcmc$matrix), size = 2)
( root <- as.numeric( mcmc$root[id[1],] ) ) ## Just one sample.
```

```
## [1] 4.148555 4.857826 2.936310
```

```
( R <- mcmc$matrix[id] ) ## Two samples in our case.
```

```
## [[1]]
##              [,1]        [,2]        [,3]
## [1,] 0.002103129 0.002198135 0.002025421
## [2,] 0.002198135 0.003811547 0.002065124
## [3,] 0.002025421 0.002065124 0.002130959
##
## [[2]]
##              [,1]        [,2]        [,3]
## [1,] 0.001985567 0.002219143 0.002041288
## [2,] 0.002219143 0.004475686 0.002278400
## [3,] 0.002041288 0.002278400 0.002229516
```

We are almost there. Note that the samples we took here are in a different format from the start point for the `ratematrixMCMC` function described above. We need to transform the variance-covariance matrices into correlation matrices and variance vectors.

Transform the matrices using the package `corp.cor`. Then compose the starting point object.

```
library( corpcor )
corr <- lapply(R, function(x) decompose.cov(x)[[1]] )
var <- lapply(R, function(x) decompose.cov(x)[[2]] )
( start <- list(root = root, matrix = corr, sd = var) )
```

```
## $root
## [1] 4.148555 4.857826 2.936310
##
## $matrix
## $matrix[[1]]
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7763736 0.9567419
## [2,] 0.7763736 1.0000000 0.7246160
## [3,] 0.9567419 0.7246160 1.0000000
##
## $matrix[[2]]
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7444117 0.9701898
## [2,] 0.7444117 1.0000000 0.7212650
## [3,] 0.9701898 0.7212650 1.0000000
##
##
## $sd
## $sd[[1]]
## [1] 0.002103129 0.003811547 0.002130959
##
## $sd[[2]]
## [1] 0.001985567 0.004475686 0.002229516
```

Now we check whether it works:

```
undebug( ratematrixMCMC )
( custom.start.handle <- ratematrixMCMC(data=anoles$data[,1:3], phy=anoles.phy, start=start
                                        , gen=1000, dir=tempdir()) )
```

```
##
## MCMC chain with multiple regimes
## Number of traits:  3
## Number of species:  125
## Number of regimes:  2
## Number of generations:  1000
## Output files:  ratematrixMCMC.28754.*
## Files directory: Same as analysis directory ('.')
##
## Use 'readMCMC' to load the MCMC chain.
## Check 'names' for more details.
```

Great!