Phylogenetic Inference using `RevBayes`

Phylogenies and the comparative method

Nicolas Lartillot and Sebastian Höhna

1 Introduction

The subject of the comparative method is the analysis of trait evolution at the macroevolutionary scale. In a comparative context, many different questions can be addressed: tempo and mode of evolution, correlated evolution of multiple quantitative traits, trends and bursts, changes in evolutionary mode correlated with major key innovations in some groups, etc (for a good introduction see Harvey and Pagel 1991).

In order to correctly formalize comparative questions, the underlying phylogeny should always be explicitly accounted for. This point is clearly illustrated, in particular, by the independent contrasts method (Felsenstein 1985; Huelsenbeck and Rannala 2003). Practically speaking, the phylogeny and the divergence times are usually first estimated using a separate phylogenetic reconstruction software. In a second step, this time-calibrated phylogeny is used as an input to the comparative method. Doing this, however, raises a certain number of methodological problems:

- the uncertainty about the phylogeny (and about divergence times) is ignored
- the traits themselves may have something to say about the phylogeny
- the rate of substitution, and more generally the parameters of the substitution process, can also be seen as quantitative traits, amenable to a comparative analysis.

All these points are not easily formalized in the context of the step-wise approach mentioned above. Instead, what all this suggests is that phylogenetic reconstruction, molecular dating and the comparative method should all be considered jointly, in the context of one single overarching probabilistic model.

Thanks to its modular structure, `RevBayes` represents a natural framework for attempting this integration. The aim of the present tutorial is to guide you through a series of examples where this integration is achieved, step by step. It can also be considered as an example of the more general perspective of *integrative modeling*, which can be recruited in many other contexts.

2 Data and files

We provide several data files which we will use in this tutorial. You may want to use your own data instead. In the **`data`** folder, you will find the following files

- **`primates_cytb.nex`**: Alignment of the *cytochrome b* subunit from 23 primates representing 14 of the 16 families (*Indriidae* and *Callitrichidae* are missing).

- **primates_lhtlog.nex**: 2 life-history traits (endocranial volume (ECV), body mass; each for males and females separately) for 23 primate species (taken from the Anage database, De Magalhaes and Costa 2009). The traits have been log-transformed.
- **primates.tree**: A time calibrated phylogeny of the same 23 primates.

3 Univariate Brownian evolution of quantitative traits

As a first preliminary exercise, we wish to reconstruct the evolution of body mass in primates and, in particular, estimate the body mass of their last common ancestor. For this, we will assume that the logarithm of body mass follows a simple univariate Brownian motion along the phylogeny. In a first step, we will ignore phylogenetic uncertainty: thus, we will assume that the Brownian process describing body mass evolution runs along a fixed time-calibrated phylogeny (with fixed divergence times), such as specified in the file **primates.tree**.

→ You may want to take the time to visualize the tree given in **primates.tree** as well as the matrix of quantitative traits specified by the **primates_lhtlog.nex** file, before going into the modeling work described below.

3.1 The model and the priors

A univariate Brownian motion $x(t)$ is parameterized by its starting value at the root of the phylogeny $x(0)$ and a rate parameter σ. This rate parameter tunes the amplitude of the variation per unit of time. Specifically, along a given time interval $(0, T)$, the value of X at time T is normally distributed, with mean $x(0)$ and variance $\sigma^2 T$:

$$x(T) \sim \text{Normal}\left(x(0), \sigma^2 T\right).$$

Concerning σ, we can formalize the idea that we are ignorant about the *scale* (the order of magnitude) of this parameter by using a log-uniform prior:

$$\sigma \sim \frac{1}{\sigma}.$$

Concerning the initial value $x(0)$ of the Brownian process at the root of the phylogeny. Alternatively, you may want to specify a normal distribution as the prior distribution on the root value if you have some prior information.

Finally, the tree topology ψ is, as mentioned above, fixed to some externally given phylogeny. The entire model is now specified: tree ψ, variance σ and Brownian process $x(t)$:

$$\begin{aligned} \sigma &\sim \frac{1}{\sigma}, \\ x(0) &\sim \text{Uniform}, \\ x(t) \mid \Psi, \sigma &\sim \text{Brownian}\left(x(0),\, \psi,\, \sigma\right). \end{aligned}$$

Conditioning the model on empirical data by clamping $x(t)$ at the tips of the phylogeny, we can then run a MCMC to sample from the joint posterior distribution on σ and x. Once this is done, we can obtain posterior means, medians or credible intervals for the value of body mass or other life-history traits for specific ancestors.

3.2 Programming the model in RevBayes

The problem of continuous trait evolution —just as for discrete trait evolution— along a phylogeny is that we do not know the values of the traits at the internal nodes. That means, that we need to treat the states at the internal nodes as additional parameters of the model. For discrete characters we use the sum-product (a.k.a. pruning) algorithm (Felsenstein 1981) to analytically integrate over all possible states at the internal nodes. For continuous characters (traits) similar methods have been proposed. In RevBayes you have three main ways of specifying this model and running an analysis on it. The three approaches are: (1) phylogenetic independent contrasts using the reduced likelihood (REML), (2) Brownian motion using a phylogenetic covariance matrix, and (3) a full Brownian motion model using data augmentation. Each of these approaches has there advantages and disadvantages as will be explained below. Nevertheless, all approaches give the same results in terms of rate estimation.

3.3 Phylogenetic Independent Contrasts using the reduced likelihood (REML)

The reduced or restricted maximum likelihood (REML) method computes the probability of observing the continuous character at the tips by an analytical solution to integrate over the internal states (Felsenstein 1985). This analytical solution is very fast to compute and thus can be applied to large phylogenies and/or many independent characters. However, the REML method looses the information about the location of the root state and thus you cannot infer which state the root or other internal nodes have.

You do not need to understand the algorithm but we provide a sketch of the idea behind REML to give you some insights. REML compute the values at the internal nodes as the phylogenetic contrasts $x_k = x_i - x_j$ where x_i and x_j are the values of the child nodes in the phylogeny. Then, we can compute the probability of observing the contrast x_k using the probability density of a normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{\nu_i + \delta_i + \nu_j + \delta_j}$ where ν_i and ν_j are the (scaled) branch lengths leading to node i and j respectively. δ is the additional uncertainty that is propagated through the phylogeny and is compute by $\delta_k = ((\nu_i + \delta_i) * (\nu_j + \delta_j))/(\nu_i + \delta_i + \nu_j + \delta_j)$. These computations are done for you in the RevBayes distribution called **dnPhyloBrownianREML**.

In the directory **RevBayes_scripts/** you will find a script called **primatesMass_BM_REML.Rev**. This script implements the univariate Brownian model described above. Instead of re-typing the content of script entirely in the context of an interactive RevBayes session, you can instead run the script directly:

```
source("RevBayes\_scripts/primatesMass_BM_REML.Rev")
```

This script essentially reformulates what has been explained in the last subsection and serves as an example solution for you. For the later section you need to adjust the script.

Let us go through the script step by step in the Rev language. First, load the trait data:

```
contData <- readContinuousCharacterData("data/primates_lhtlog.nex")
```

If you type you will see that the continuous character data matrix contains several characters (columns).

```
contData

   Continuous character matrix with 23 taxa and 11 characters
   ==========================================================
   Origination:                primates_lhtlog.nex
   Number of taxa:             23
   Number of included taxa:    23
   Number of characters:       11
   Number of included characters: 11
   Datatype:                   Continuous
```

Since we only want the body mass (of females) we exclude all but the third character

```
contData.excludeAll()
contData.includeCharacter(3)
```

Next, load the time-tree from file. Remember that we use in this first simple example a fixed tree that we assume is known without uncertainty.

```
treeArray <- readTrees("data/primates.tree")
psi <- treeArray[1]
```

→ You may want to look at this tree before by loading the **primates.tree** in FigTree or any other tree visualization software.

As usual, we start be initializing some useful helper variables. For example, we set up a counter variable for the number of moves that we already added to our analysis. This will make it much easier if we extend the model or analysis to include additional moves or to remove some moves.

```
mi = 0
```

Then, we define the overall rate parameter σ which we assign a (truncated) log-uniform prior. Note that it is more efficient in Bayesian inference to specify a uniform prior and then to transform the parameter which we will use here:

```
logSigma ~ dnUniform(-5,5)
sigma := 10^logSigma
```

Using this approach we have specified a prior probability distribution on **sigma** between 10^{-5} to 10^{5} which should be broad enough to include all reasonable values.

Since the rate of trait evolution **logSigma** is a stochastic variable and we want to estimate it, we need to add a sliding move on it. Remember that the sliding move proposes new values drawn from a window with width **delta** and is centered around the current values; thus it slides through the parameter space together with the current parameter value.

```
moves[++mi] = mvSlide(logSigma, delta=1.0, tune=true, weight=2.0)
```

Next, define a random variable from the univariate Brownian-Phylo-REML process, which we will call **logmass**. We need to provide the tree variable **psi**, some branch-specific rate multiplier parameter which we simply set to 1, the shared rate for this site **sigma** and the number of sites (number of continuous traits) that we use.

```
logmass ~ dnPhyloBrownianREML(psi, branchRates=1.0, siteRates=sigma, nSites=1)
```

Now, condition the Brownian model on empirically observed values for body mass in the extant taxa.

```
logmass.clamp( contData )
```

The model is now entirely specified and we can create a model object containing the entire model graph by providing it with only one of our model variables, e.g., **sigma**.

```
mymodel = model(sigma)
```

To see what it happing during the MCMC let us make a screen monitor that tracks the rate **sigma**.

```
monitors[1] = mnScreen(printgen=10, sigma)
```

Additionally, we'll use a file monitor that does the same thing, but directly stores the values into a file.

```
monitors[2] = mnFile(filename="output/primates_mass_REML.log", printgen=10, separator = TAB,
    sigma)
```

We can finally create a mcmc, and run it for a good 100 000 cycles after we did a burnin phase of 10 000 iterations:

```
mymcmc = mcmc(mymodel, monitors, moves)
mymcmc.burnin(generations=10000,tuningInterval=500)
mymcmc.run(100000)
```

Exercises

- Run the model.
- using **Tracer**, visualize the posterior distribution on the rate parameter **sigma**
- calculate the 95% credible interval for the rate of evolution of the log of body mass (σ)

3.4 Phylogenetic covariance matrix

The second method that we will use creates a phylogenetic covariance matrix. The phylogenetic covariance matrix method integrates over the states at the internal nodes as well but uses instead a multivariate normal distribution. The key advantage is that this method provides information about the root state since it models the root state as an additional parameter of the model. The disadvantage is that it is very computationally intensive. That means, that the phylogenetic covariance matrix approach may take long for very large data sets (at least in its current implementation).

→ Copy the file **primatesMass_BM_REML.Rev**, name it for example **primatesMass_BM_Cov.Rev** and start editing it.

In the previous example, the REML approach, we did not specify a parameter for the state at the root. In this exercise, we need this additional parameter. Let us use a uniform prior distribution on the logarithm of the root mass. A uniform prior between -100 and 100 should be diffuse enough. Just image how big or small an individual needs to be if it has body mass smaller than exp(-100) or larger than exp(100).

```
rootlogmass ~ dnUniform(-100,100)
```

Next, we'll specify a sliding move that proposes new values for the **rootlogmass** randomly drawn from a window centered around the current value.

```
moves[++mi] = mvSlide(rootlogmass,delta=10,tune=true,weight=2)
```

Finally, we need to substitute the **dnPhyloBrownianREML** by **dnPhyloBrownianMVN** to use the phylogenetic covariance matrix approach. Again, we provide the tree variable **psi**, some branch-specific rate multiplier parameter which we simply set to 1, the shared rate for this site **sigma** and the number of sites (number of continuous traits) that we use.

```
logmass ~ dnPhyloBrownianMVN(psi, branchRates=1.0, siteRates=sigma, rootStates=rootlogmass,
    nSites=1)
```

This will automatically connect the parameters of the model together.

Additionally, we now have the extra parameter **rootlogmass** which we want to monitor. Thus we need to replace the file monitor and use instead.

```
monitors[2] = mnFile(filename="output/primates_mass_Cov.log", printgen=10, separator = TAB,
    sigma, rootlogmass)
```

→ Don't forget to change the output file names in the monitors, otherwise your old analyses files will be overwritten.

Exercises

- Run the analysis.
- Using **Tracer**, visualize the posterior distribution on the rate parameter **sigma** and the **rootlogmass**
- How does the posterior distribution of **sigma** looks compared with the first analysis?
- Calculate the 95% credible interval for the rate of evolution of the log of body mass (σ) and the **rootlogmass**

3.5 Data augmentation

The third method to we will use is a data augmentation method. The data augmentation method uses explicitly the states at the internal nodes. We will specify the full model in Rev. That means that we will create a random variable for each node of the tree using a for loop. Each trait is then simply assign a normal distribution, as we described above in the model description. The advantage of this method is that you estimate the values at the internal nodes directly and that you have full control about modifying any part of the model. The disadvantage is that the MCMC algorithm will be harder if you want to jointly estimate the phylogeny, although the likelihood computation is very fast. The problem is the mixing of

the MCMC because proposing new trees involves proposing new good values for the states at the internal nodes.

→ Copy the file **primatesMass_BM_REML.Rev**, name it for example **primatesMass_BM_DA.Rev** and start editing it.

]As in the previous example we will use a uniform prior distribution on the logarithm of the root mass.

```
rootlogmass ~ dnUniform(-100,100)
```

Again, we'll specify a sliding move that proposes new values for the **rootlogmass** randomly drawn from a window centered around the current value.

```
moves[++mi] = mvSlide(rootlogmass,delta=10,tune=true,weight=2)
```

In order to create the random variables for the internal states we need to know the number of nodes and the number of tips. We will store these as some helper variables.

```
numNodes = psi.nnodes()
numTips = psi.ntips()
```

Now we are ready to specify the Brownian motion model for each branch. That is, we simply specify a new normal distributed random variable for each node with mean being equal to the value of the parent variable and the standard deviation being equal to the product of the square root of the branch length and our rate parameter **sigma**. We store all the variables in the vector **logmass**. Then we are able to access the value at the parent node using the index of the parent node, which we can obtain from the tree using the function **psi.parent(i)**. Similarly, since the variance depends on the branch length we retrieve the branch length of node with index **i** using the function **psi.branchLength(i)**.

First we need to copy (create a reference to) the **rootlogmass**

```
logmass[numNodes] := rootlogmass
```

Let us start by creating the random variables for the internal nodes. Remember that the variance is equal to **sigma**-squared times the branch length, and we need to compute the square root of it to obtain the standard deviation.

```
# univariate Brownian process along the tree
# parameterized by sigma
for (i in (numNodes-1):(numTips+1) ) {
  logmass[i] ~ dnNormal( logmass[psi.parent(i)], sd=sigma*sqrt(psi.branchLength(i)) )
  # moves on the Brownian process
  moves[++mi] = mvSlide( logmass[i], delta=10, tune=true ,weight=2)
}
```

You may have noticed that we specified in the loop a move for each internal **logmass**. This is because we want to use the MCMC algorithm to integrate over the uncertainty in the states.

Next, we repeat the same loop but now for the tip nodes. Instead of applying a move to each tip node we will clamp the nodes. The nodes will be clamped with the data that we read in before.

```
for (i in numTips:1 ) {
  logmass[i] ~ dnNormal( logmass[psi.parent(i)], sd=sigma*sqrt(psi.branchLength(i)) )

  # condition Brownian model on quantitative trait data (second column of the dataset)
  logmass[i].clamp(contData.getTaxon(psi.nodeName(i))[1])
}
```

Here we do not need a **dnPhyloBrownianREML** or **dnPhyloBrownianMVN** distribution anymore because we explicitly instantiated the model. Note that the approach we have taken using the loop is the tree-plate approach described in (Höhna et al. 2014)

Since we have several additional parameters —the states at the internal nodes— we will use a model monitor to write to file instead.

```
monitors[2] = mnModel(filename="output/primates_mass_DA.log", printgen=10, separator = TAB)
```

→ Don't forget to change the output file names in the monitors, otherwise your old analyses files will be overwritten.

Exercises

- Run the analysis.
- Using **Tracer**, visualize the posterior distribution on the rate parameter **sigma** and the **rootlogmass** and the internal states.
- How does the posterior distribution of **sigma** looks compared with the first and second analysis?
- Calculate the 95% credible interval for the rate of evolution of the log of body mass (σ) and the **rootlogmass**. Have they changed?

3.6 Brief intermediate summary

In the above exercises you have analyzed the log-transformed body mass using a Brownian motion (BM) along a phylogeny. We assumed the phylogeny to be known without error and were primarily interested in the rate how fast the body mass evolves. The exercises showed you three different ways how to approach the BM model, each having its advantages and disadvantages. You will need to decide for your analysis which approach is most appropriate. Our intention was mainly to show you some of the flexibility in `RevBayes` how to specify the same model in many different ways, exposing sometimes more and sometimes less of the internal model graph structure. As an extra exercise you could start thinking about how to extend the basic BM.

4 Multiple independently evolving traits

The Brownian motion can easily be applied to multiple traits. The exactly same procedure and model will be applied as above and thus we will skip the repetitive description. The interesting questions that you can ask about multiple traits is —amongst others— if rates between traits vary.

As an example we use now the first four characters of the data matrix, which are: 1) female log endocranial volume (ECV), 2) male log ECV, 3) female log body mass and 4) male log body mass.

```
contData <- readContinuousCharacterData("data/primates_lhtlog.nex")
contData.excludeCharacter(5:11)
```

In the first simple example we assume that all site rates are equal, thus we use a rate multiplier we we call **`perSiteRates`**.

```
perSiteRates <- [1,1,1,1]
```

Furthermore, we have now four characters and hence we will estimate a root state for each one of them. We use the same prior probability for every character, just as above for the single character example.

```
for (i in 1:4) {
   rootlogmass[i] ~ dnUniform(-100,100)
   moves[++mi] = mvSlide(rootlogmass[i],delta=10,tune=true,weight=2)
}
```

Finally, we bring together all the parameter to specify our Brownian motion model along the tree. Note that we specify here the site rates as the product of the global rate **sigma** time the sites specific rates (which are, however all equal in this first exercie).

```
logmass ~ dnPhyloBrownianMVN(psi, branchRates=1.0, siteRates=sigma*perSiteRates, rootStates=
    rootlogmass, nSites=4)
```

After this you need to create the model using the **model** function, create your monitors, create the MCMC and finaly run the MCMC.

Exercises

- Run the analysis.
- Using **Tracer**, visualize the posterior distribution on the rate parameter **sigma** and the four **rootlogmass** values.
- Compare the results to the previous single trait analyses?

4.1 Independent site rates

```
perSiteRates ~ dnDirichlet([1,1,1,1])
moves[++mi] = mvSimplexElementScale(perSiteRates,alpha=10,tune=true,weight=4)
```

Exercises

- Run the analysis.
- Using **Tracer**, visualize the posterior distribution on the per site rate parameter **perSiteRates**.
- How are the estimate per site rates compared to each other?

4.2 Partially shared site rates

We hope that your results in the previous analysis showed that the rates for females and males for both the ECV and the body mass are very similar. This motivates us to specify an explicit model where the rates between ECV and body mass evolution are different but shared between females and males. We will model the difference using the **siteRateDiff** parameter which is drawn from a Beta(1,1) distribution, which is essentially a uniform distribution between 0 and 1. We still choose the Beta distribution because it is easier to specify more informative priors.

```
siteRateDiff ~ dnBeta(1,1)
```

Next, we specify a sliding move on the **siteRateDiff**.

```
moves[++mi] = mvSlide(siteRateDiff,delta=10,tune=true,weight=2)
```

To obtain the rates we use **siteRateDiff** as the rate for the ECV evolution in females and males and 1-**siteRateDiff** as the rate for the body mass evolution in females and males. Note, however, that we need a small workaround here. Rev is a strictly typed language, which you may have noticed. The per site rates need to be positive real number (negative rates obviously don't make sense). However, the difference between two number is not guaranteed to be positive, only if we know that the first term is larger than the second. This is exactly the case for 1-**siteRateDiff** but RevBayes doesn't know this so we need to tell it by using the absolute value **abs** function.

```
perSiteRates[1] := siteRateDiff
perSiteRates[2] := siteRateDiff
perSiteRates[3] := abs(1.0 - siteRateDiff) # specify the type here
perSiteRates[4] := abs(1.0 - siteRateDiff)
```

You could have specified many other prior distribution on the relative rate (or ratio) such as a uniform prior distribution and then use logit or hyperbolic tangent transformation. The only important factor is that the rates are somehow normalized because we use the global, shared rate **sigma**. Using the beta distribution makes the rates automatically normalized because the two different rates sum to one. Note that in the previous all four rates summed to 1.0 and here the two different rates sum to 1.0. This has the effect that we should estimate the global rate **sigma** to be half of what you estimated previously. You may want to check this once you ran the analysis and loaded the output into Tracer.

The remaining functions are the same as in the previous example.

Exercises

- Run the analysis.
- Using **Tracer**, visualize the posterior distribution on the per site rate parameter **perSiteRates**.
- Does the value of equal rate (**siteRateDiff**=0.5) fall into the 95% credible interval?
- If the rate does not fall into the 95% credible interval, should we reject the hypothesis that the underlying rates are equal?

4.3 Model selection

Our previous analysis explored the parameters of the rate of evolution. You will have seen that the rate of evolution was smaller for the ECV compared with the rate of evolution of body size. However, we did not perform a statistical test to reject the hypothesis if the underlying rates between the different characters are significantly different. Therefore, we will perform a marginal likelihood estimation for all three model: 1) the equal rates, 2) the independent rates, and 3) the shared rates.

→ You need to adopt the three scripts of your previous analysis.

Previously you created an **mymcmc** object using the **mcmc** function. Now, you need to replace this object with the power posterior MCMC object.

```
pow_p = powerPosterior(mymodel, moves, monitors, "output/pow_p_BM_equal.out", cats=100,
    sampleFreq=10)
```

Next, you need to run the burnin and afterwards run the power posterior sampler. This will create 101 output files (100 stepping stones and one for the posterior) logging the values into the monitors that you have specified before (we recommend you not to use a screen monitor). You could look into each of these files to check that you obtain sufficiently many samples for each stone.

```
pow_p.burnin(generations=10000,tuningInterval=250)
pow_p.run(generations=10000)
```

Use stepping-stone sampling to calculate marginal likelihoods.

```
ss = steppingStoneSampler(file="output/pow_p_MultiBM_equal.out", powerColumnName="power",
    likelihoodColumnName="likelihood")
ss.marginal()
```

`RevBayes` will print to the screen the estimated marginal likelihood. Write down this value and repeat the exercise for the other two models.

Exercises

- Run the three different marginal likelihood estimations and write down the marginal likelihood.
- Compute the log-Bayes factors by taking the difference of the alternative model (independent rates & shared rates) and the null hypothesis (equal rates).
- Is any of the log-Bayes factors larger than 1.16 for substantial support of the alternative hypothesis (or 2.3 for strong support)?

5 Estimating the phylogeny from continuous character data

In the motivation we argued that the phylogeny should be estimated together with the parameters of the evolutionary process. Only for simplicity we used fixed trees in the previous exercises. Instead of using a tree fixed to a pre-defined value, the tree should now be moved during the MCMC. Implementing this joint model in `RevBayes` is just a matter of adding the following features to the model defined in the previous section (after duplicating the script). You may want to use the REML approach for computational efficient (i.e., good MCMC mixing) although the phylogenetic covariance matrix and the data augmentation are generally possible too.

We need to get some useful variables from the data so that we will be able to specify the tree prior below. These variables are the number of tips, the number of nodes and the names of the species which we all can query from the continuous character data object.

```
numTips = contData.ntaxa()
names = contData.names()
numNodes = numTips * 2 - 1
```

Instead of having a fixed tree as in the previous, we should now define a *random* tree. We use a birth death prior with prior distributions on the **diversification** rate and **turnover** rate. The **diversification** rate corresponds to the rate of growth of the tree and the **turnover** rate corresponds to the rate how quickly a species is replaced by a new one. This parametrization has the advantage that we can use meaningful prior distributions from the fossil record.

```
diversification ~ dnLognormal(0,1)
turnover ~ dnGamma(4,4)
```

Then, we compute deterministically the **speciation** and **extinction** rate from the **diversification** rate and **turnover** rate.

```
speciation := diversification + turnover
extinction := turnover
```

Instead, you could also use directly the speciation and extinction rates, e.g., endowed with some diffuse exponential prior.

```
# rescaling moves on speciation and extinction rates
moves[++mi] = mvScale(diversification, lambda=1, tune=true, weight=3.0)
moves[++mi] = mvScale(turnover, lambda=1, tune=true, weight=3.0)
```

The phylogeny that we used are obviously not a complete sample of all the species and you should take the incomplete sampling into account. We will simply use an empirical estimate of the fraction of species which we included in this study. For more information about incomplete taxon sampling see Höhna et al. (2011) and Höhna (2014).

```
sampling_fraction <- 23 / 270  # 23 out of the ~ 270 primate species
```

Now we are able to specify of tree variable **psi** which is drawn from a constant rate birth-death process. We will condition the age of the tree to be 75 million years old which is approximately the crown age of primates, although this estimate is still debated. We only condition here on the crown age for simplicity because we do not use any other fossil calibration.

```
psi ~ dnBDP(lambda=speciation, mu=extinction, rho=sampling_fraction, rootAge=75, nTaxa=numTips,
     names=names)
```

Note that, here, we do not have included any fossil information: we are merely doing *relative* dating.

The first moves on the tree which we specify are moves that change the node ages. The first move randomly picks a subtree and rescales it, and the second move randomly pick a node and uniformly proposes a new node age between its parent age and oldest child's age.

```
moves[++mi] = mvSubtreeScale(psi, weight=5.0)
moves[++mi] = mvNodeTimeSlideUniform(psi, weight=10.0)
```

We also need moves on the tree topology to estimate the phylogeny. The two moves which you use are the nearest-neighbor interchange (NNI) and the fixed-nodeheight-prune-and-regraft (FNPR) (Höhna and Drummond 2012).

```
moves[++mi] = mvNNI(psi, weight=5.0)
moves[++mi] = mvFNPR(psi, weight=5.0)
```

We are essentially done now. We only need to add a new monitor for the tree so that we can monitor and build the maximum a posteriori tree later.

```
monitors[3] = mnFile(filename="output/primates_mass_multiBM_tree.trees", printgen=100,
    separator = TAB, psi)
```

→ A short analysis of 50 000 iterations will be sufficient.

Exercises

- Run the analysis.
- Create the maximum a posteriori (MAP) tree **readTreeTrace** and **mapTree**.
- Look at the estimated tree and compare it to the tree you used before.

- How does the posterior distribution of **sigma** looks compared with the first and second analysis?
- Compute the posterior probabilities of each tree using the **treetrace.summarize()** command.
- What is the posterior probability of the best tree?

6 Including information about the tree from molecular data

Starting from the model implemented in the last section, we now want to account for phylogenetic uncertainty using information contained in molecular data. As first pointed out by Huelsenbeck and Rannala (2003), this can easily be done in a Bayesian framework, through the use of a joint model combining sequence data and quantitative traits. Specifically:

- two data sets are loaded: one for sequence data and one for quantitative traits
- a tree is defined under birth-death process
- a Brownian model is defined over the tree (just as described in the previous section)
- the Brownian model is conditioned on the quantitative trait data
- a substitution model is defined over the same tree (GTR+Gamma)
- the substitution model is conditioned on the molecular sequence data.

6.1 Programming the model in RevBayes

First, we create a substitution model, just like what you probably did in previous tutorial (e.g., RB_CTMC_Tutorial). In a first step, we will use a GTR+Gamma model. Start by loading the sequence data matrix specified in **data/primates_cytb.nex**.

```
seqData <- readDiscreteCharacterData("data/primates_cytb.nex")
```

We can use a flat Dirichlet prior density on the exchangeability rates **er** and the the base frequencies **pi**.

```
er_prior <- v(1,1,1,1,1,1)
er ~ dnDirichlet(er_prior)
pi_prior <- v(1,1,1,1)
pi ~ dnDirichlet(pi_prior)
```

Now add the simplex scale move one each the exchangeability rates **er** and the stationary frequencies **pi** to the moves vector:

```
moves[++mi] = mvSimplexElementScale(er)
moves[++mi] = mvSimplexElementScale(pi)
```

We can finish setting up this part of the model by creating a deterministic node for the GTR instantaneous-rate matrix **Q**. The **fnGTR()** function takes a set of exchangeability rates and a set of base frequencies to compute the instantaneous-rate matrix used when calculating the likelihood of our model.

```
Q := fnGTR(er,pi)
```

The next part of the substitution process is the rate variation among sites. We will model this using the commonly applied 4 discrete gamma categories which only have a single parameter **alpha**. Let us specify the rate of **alpha** to 0.05 (thus the mean will be 20.0).

```
alpha_prior <- 0.05
```

Then create a stochastic node called **alpha** with an exponential prior:

```
alpha ~ dnExponential(alpha_prior)
```

Initialize the **gamma_rates** deterministic node vector using the **fnDiscretizeGamma()** function with **4** bins:

```
gamma_rates := fnDiscretizeGamma( alpha, alpha, 4 )
```

The random variable that controls the rate variation is the stochastic node **alpha**. We will apply a simple scale move to this parameter.

```
moves[++mi] = mvScale(alpha, weight=2.0)
```

This finishes the substitution process part of the model.

Then next part of the model is the clock model. Here we need a clock model because we work on a time tree. We use our exponentiated uniform distribution to specify a flat prior distribution which is scale invariant.

```
logClockRate ~ dnUniform(-5,5)
clockRate := 10^logClockRate
```

We will apply a sliding window move to the **logClockRate**.

```
moves[++mi] = mvSlide(logClockRate, delta=0.1, tune=true, weight=2.0)
```

Remember that you need to call the **PhyloCTMC** constructor to include the new site-rate parameter:

```
seq ~ dnPhyloCTMC(tree=psi, Q=Q, siteRates=gamma_rates, branchRates=clockRate, type="
    DNA")
```

Finally we need to attach the molecular sequence data to our model.

```
seq.clamp(seqData)
```

You may have wondered how the continuous trait model and the molecular sequence model are combined. This happened automatically when you used the same tree parameter **psi** for both models. Since both models share now the same parameter, they will be used for a joint analyses. The model function thus will construct the joint model graph for the continuous trait model as well as the molecular sequence model.

After you ran the MCMC analyses, read in the tree trace. We will assume that you used a tree monitor and called the file *output/primates_joint.trees*. Change the name if you used a different one.

```
treetrace = readTreeTrace("output/primates_joint.trees", "clock")
```

Then build the maximum a posteriori tree.

```
map = mapTree( file="primates_joint.tree", treetrace )
```

Write this model and make sure that it runs when you give it to `RevBayes`.

Exercises

- Run the analysis.
- Using **Tracer**, visualize the posterior distribution on the rate parameter **sigma** and compare it to the previous analyses.
- Look at the MAP tree which you estimated now.

References

De Magalhaes, J. and J. Costa. 2009. A database of vertebrate longevity records and their relation to other life-history traits. Journal of evolutionary biology 22:1770–1774.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. Journal of Molecular Evolution 17:368–376.

Felsenstein, J. 1985. Phylogenies and the comparative method. American Naturalist Pages 1–15.

Harvey, P. H. and M. D. Pagel. 1991. The comparative method in evolutionary biology vol. 239. Oxford university press Oxford.

Höhna, S. 2014. Likelihood Inference of Non-Constant Diversification Rates with Incomplete Taxon Sampling. PLoS One 9:e84184.

Höhna, S. and A. J. Drummond. 2012. Guided Tree Topology Proposals for Bayesian Phylogenetic Inference. Systematic Biology 61:1–11.

Höhna, S., T. A. Heath, B. Boussau, M. J. Landis, F. Ronquist, and J. P. Huelsenbeck. 2014. Probabilistic Graphical Model Representation in Phylogenetics. Systematic Biology 63:753–771.

Höhna, S., T. Stadler, F. Ronquist, and T. Britton. 2011. Inferring speciation and extinction rates under different species sampling schemes. Molecular Biology and Evolution 28:2577–2589.

Huelsenbeck, J. P. and B. Rannala. 2003. Detecting correlation between characters in a comparative analysis with uncertain phylogeny. Evolution 57:1237–1247.

Version dated: February 4, 2015