Phylogenetic Inference using `RevBayes`

Calibrated Time Trees & Divergence Time Estimation

Sebastian Höhna and Tracy A. Heath

1 Overview

This tutorial demonstrates how to estimate divergence times and a dated phylogeny. The key concepts of this tutorial are the global molecular clock and node- and fossil-calibrations.

In this tutorial you will perform a Bayesian inference to estimate a time-calibrated phylogeny. In the first part we will demonstrate you how to set up a basic model for time-calibrated phylogeny inference. This analysis will use an informative prior distribution on the global molecular clock. In the second part you will use an informative prior distribution on the root age (crown age) to calibrate the phylogeny. In the third and final part you will use node- and fossil-calibrations instead of the informative prior on the root age. All the assumptions will be covered more in detail later in this tutorial.

Requirements

We assume that you have read and hopefully completed the following tutorials:

- RB_Getting_Started
- RB_Basics_Tutorial
- RB_CTMC_Tutorial

Note that the RB_Basics_Tutorial introduces the basic syntax of `Rev` but does not cover any phylogenetic models. You may skip the RB_Basics_Tutorial if you have some familiarity with `R`. We tried to keep this tutorial very basic and introduce all the language concepts on the way. You may only need the RB_Basics_Tutorial for a more in-depth discussion of concepts in `Rev`.

2 Data and files

We provide the data file(s) 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).

Table 1: Node information used for calibrating divergence times in the primate tree.

Clade	Age range (My)	Citation
Simiiformes	43 ± 4.5	
galagids and lorisids	40 ± 3	

3 Example: Divergence time estimation using a strict, global clock and an informative prior of the clock rate

3.1 Getting Started

The first section of this exercise involves: (1) setting up a general time reversible (GTR) substitution model with gamma distributed rate variation among sites for an alignment of the cytochrome b subunit; (2) use an informative prior on the clock rate to date the phylogeny; (3) approximating the posterior probability of the tree topology and node ages (and all other parameters) using MCMC, and; (4) summarizing the MCMC output by computing the maximum *a posteriori* tree.

The general structure of the model is represented in Figure **??**. This figure shows the full model graph.

3.2 Loading the Data

→ Download data and output files (if you don't have them already) from: http://revbayes.github.io/tutorials.html

First load in the sequences using the **readDiscreteCharacterData()** function.

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

Executing these lines initializes the data matrix as the respective **Rev** variables.

Next we will specify some useful variables based on our dataset. The variable **data** has *member functions* that we can use to retrieve information about the dataset. These include the number of species (**n_species**) and the tip labels (**names**). Each of these variables will be necessary for setting up different parts of our model.

```
n_species <- data.ntaxa()
names <- data.names()
```

Additionally, we set up a counter variable for the number of moves that we already added to our analysis. [Recall that moves are algorithms used to propose new parameter values during the MCMC simulation.] This will make it much easier if we extend the model or analysis to include additional moves or to remove some moves.

```
mi = 0
```

You may have noticed that we used the **=** operator to create the move index. This simply means that the variable is not part of the model. You will later see that we use this operator more often, e.g., when we create moves and monitors.

With the data loaded, we can now proceed to specify our substitution model.

3.3 General Time Reversible (GTR) Substitution Model

The GTR model requires that we define and specify a prior on the six exchangeability rates, which we will describe using a flat Dirichlet distribution. As we did previously for the Dirichlet prior on base frequencies, we first define a constant node specifying the vector of concentration-parameter values using the **v()** function:

```
er_prior <- v(1,1,1,1,1,1)
```

This node defines the concentration-parameter values of the Dirichlet prior distribution on the exchangeability rates. Now, we can create a stochastic node for the exchangeability rates using the **dnDirichlet()** function, which takes the vector of concentration-parameter values as an argument and the **~** operator. Together, these create a stochastic node named **er** (θ in Figure **??**):

```
er ~ dnDirichlet(er_prior)
```

For each stochastic node in our model, we must also specify a proposal mechanism if we wish to estimate that parameter. The Dirichlet prior on our parameter **er** creates a *simplex* of values that sum to 1.

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

We can use the same type of distribution as a prior on the 4 stationary frequencies ($\pi_A, \pi_C, \pi_G, \pi_T$) since these parameters also represent proportions. Specify a flat Dirichlet prior density on the base frequencies:

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

The node **pi** represents the π node in Figure **??**. Now add the simplex scale move on the stationary frequencies to the moves vector:

```
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)
```

3.4 Setting up the Gamma Model

Create a constant node called **shape_prior** for the rate parameter of the exponential prior on the gamma-shape parameter (this is represented as the constant λ-rate parameter in Figure **??**):

```
shape_prior <- 0.05
```

Then create a stochastic node called **alpha** with an exponential prior (this represents the stochastic node for the α-shape parameter in Figure **??**):

```
alpha ~ dnExponential(shape_prior)
```

The way the ASRV model is implemented involves discretizing the mean-one gamma distribution into a set number of rate categories, k. Thus, we can analytically marginalize over the uncertainty in the rate at each site. The likelihood of each site is averaged over the k rate categories, where the rate multiplier is the mean (or median) of each of the discrete k categories. To specify this, we need a deterministic node that is a vector that will hold the set of k rates drawn from the gamma distribution with k rate categories. The **fnDiscretizeGamma()** function returns this deterministic node and takes three arguments: the shape and rate of the gamma distribution and the number of categories. Since we want to discretize a mean-one gamma distribution, we can pass in **alpha** for both the shape and rate.

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

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

Note that here, by convention, we set $k = 4$. 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)
```

3.5 Tree Prior: Tree Topology and Node Ages

The tree (the topology and node ages) is a stochastic node in our phylogenetic model. In Figure **??**, the tree is denoted Ψ.

We will assume a constant-rate birth-death process as the prior distribution on the tree. This means that all possible labeled, rooted tree topologies have equal probability. The distribution in `RevBayes` is **`dnBDP()`**. For the birth-death process we need a speciation rate and extinction rate parameter. Let us start with those two variables. We use a *gamma* distribution with rate $a = 5$ and shape $b = 1$ for both the **`diversification`** and **`turnover`** variables.

```
a <- 5
b <- 1
diversification ~ dnGamma(shape=a, rate=b)
c <- 5
d <- 1
turnover ~ dnGamma(shape=c,rate=d)
```

Now we can transform the **`diversification`** and **`turnover`** into the **`speciation`** rate and **`extinction`** rate.

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

We also need to specify a prior on the root age (our informed guess is about 75-80 mya). So we use a uniform distribution between 50 and 100.

```
root ~ dnUniform(50.0,100.0)
```

Additionally, we know that we do not have all primate species included in this data set. We only have 23 out of the approximately 270 primate species. Thus, we use a sampling fraction to represent this incomplete taxon sampling.

```
sampling_fraction <- 23 / 270
```

Here we have created our first three stochastic variables. For each one of them we need to create at least one moves that change the stochastic variables. In this case we use sliding window proposals but you could use scaling proposals for the rates too.

```
moves[++mi] = mvSlide(diversification,delta=1,tune=true,weight=1)
moves[++mi] = mvSlide(turnover,delta=1,tune=true,weight=1)
moves[++mi] = mvSlide(root,delta=1,tune=true,weight=1)
```

Next, specify the **tree** stochastic node by passing in the tip labels **names** to the **dnBDP()** distribution:

```
psi ~ dnBDP(lambda=speciation, mu=extinction, rootAge=abs(root), rho=sampling_fraction
    , nTaxa=n_species, names=names )
```

Some types of stochastic nodes can be updated by a number of alternative moves. Different moves may explore parameter space in different ways, and it is possible to use multiple different moves for a given parameter to improve mixing (the efficiency of the MCMC simulation). In the case of our rooted tree, for example, we can use both a nearest-neighbor interchange move without and with changing the node ages (**mvNarrow** and **mvNNI**) and a fixed-nodeheight subtree-prune and regrafting move (**mvFNPR**). We also need moves that change the ages of the internal nodes; which are for example the **mvSubtreeScale** and **mvNodeTimeSlideUniform**. These moves do not have tuning parameters associated with them, thus you only need to pass in the **psi** node and proposal **weight**.

```
moves[++mi] = mvNarrow(psi, weight=5.0)
moves[++mi] = mvNNI(psi, weight=1.0)
moves[++mi] = mvFNPR(psi, weight=3.0)
moves[++mi] = mvSubtreeScale(psi, weight=3.0)
moves[++mi] = mvNodeTimeSlideUniform(psi, weight=15.0)
```

The weight specifies how often the move will be applied either on average per iteration or relative to all other moves. Have a look at the MCMC tutorial for more details about moves and MCMC strategies: http://revbayes.github.io/tutorials.html

3.6 The Global Molecular Clock Model

The global molecular clock assumes that the rate of substitution is constant over the tree and over time (Fig. **??**).

Here we use an informative prior on the clock rate. In several studies it has been suggested that the rate of mitochondrial evolution is about 0.01 (=1%) per million years per site. Thus, we will specify a narrow prior distribution centered around 0.01.

```
clock_M <- 0.01
clock_s <- 1
clock_mu <- log(clock_M) - ((clock_s * clock_s) * 0.5)

clockRate ~ dnLognormal(clock_mu, clock_s)
```

As usual, we need a move on the stochastic variable, the **clockRate**. We use a scaling move because this is a rate parameter.

```
moves[++mi] = mvScale(clockRate,lambda=1,tune=true,weight=1)
```

3.7 Putting it All Together

We have fully specified all of the parameters of our phylogenetic model—the tree topology with branch lengths, and the substitution model that describes how the sequence data evolved over the tree with branch lengths. Collectively, these parameters comprise a distribution called the *phylogenetic continuous-time Markov chain*, and we use the **PhyloCTMC** constructor function to create this node. This distribution requires several input arguments: (1) the **tree** with branch lengths; (2) the instantaneous-rate matrix **Q**; (3) the clock rate, and; (4) the **type** of character data.

Build the random variable for the character data (sequence alignment).

```
# the sequence evolution model
seq ~ dnPhyloCTMC(tree=psi, Q=Q, branchRates=clockRate, siteRates=gamma_rates, type="
    DNA")
```

Once the **PhyloCTMC** model has been created, we can attach our sequence data to the tip nodes in the tree.

```
seq.clamp(data)
```

[Note that although we assume that our sequence data are random variables—they are realizations of our phylogenetic model—for the purposes of inference, we assume that the sequence data are "clamped".] When this function is called, **RevBayes** sets each of the stochastic nodes representing the tips of the tree to the corresponding nucleotide sequence in the alignment. This essentially tells the program that we have observed data for the sequences at the tips.

Finally, we wrap the entire model to provide convenient access to the DAG. To do this, we only need to give the **model()** function a single node. With this node, the **model()** function can find all of the other nodes by following the arrows in the graphical model:

```
mymodel = model(Q)
```

3.8 Performing an MCMC Analysis Under the Global Clock Model

In this section, will describe how to set up the MCMC sampler and summarize the resulting posterior distribution of trees.

3.8.1 Specifying Monitors

For our MCMC analysis, we need to set up a vector of *monitors* to record the states of our Markov chain. The monitor functions are all called **mn***, where * is the wildcard representing the monitor type. First, we will initialize the model monitor using the **mnModel** function. This creates a new monitor variable that will output the states for all model parameters when passed into a MCMC function.

```
monitors[1] = mnModel(filename="output/primates_cytb_global_clock.log",printgen=10,
    separator = TAB)
```

The **mnFile** monitor will record the states for only the parameters passed in as arguments. We use this monitor to specify the output for our sampled trees and branch lengths.

```
monitors[2] = mnFile(filename="output/primates_cytb_global_clock.trees",printgen=10,
    separator = TAB, psi)
```

Finally, create a screen monitor that will report the states of specified variables to the screen with **mnScreen**:

```
monitors[3] = mnScreen(printgen=1000, clockRate)
```

3.8.2 Initializing and Running the MCMC Simulation

With a fully specified model, a set of monitors, and a set of moves, we can now set up the MCMC algorithm that will sample parameter values in proportion to their posterior probability. The **mcmc()** function will create our MCMC object:

```
mymcmc = mcmc(mymodel, monitors, moves)
```

We may wish to run the **.burnin()** member function. Recall that this function **does not** specify the number of states that we wish to discard from the MCMC analysis as burnin (i.e., the samples collected before the chain converges to the stationary distribution). Instead, the **.burnin()** function specifies a *completely separate* preliminary MCMC analysis that is used to tune the scale of the moves to improve mixing of the MCMC analysis.

```
mymcmc.burnin(generations=10000,tuningInterval=1000)
```

Now, run the MCMC:

```
mymcmc.run(generations=30000)
```

When the analysis is complete, you will have the monitored files in your output directory.

→ Look at the file called **output/primates_cytb_global_clock.log** in Tracer.

3.9 Exercise 1

We are interested in the divergence time estimate between *New World Monkeys* and *Old World Monkeys*, which is the infraorder *Simiiformes*. We have provided the Table 2 of the primates species and their relationships for you convenience (if you forgot which primate species belongs to either of the two groups). To obtain an estimate of the divergence time we read in the tree trace and build the annotated maximum

Table 2: Primate species and famaly relationships.

Species	**Family**	**Parvorder**	**Suborder**
Alouatta palliata	Atelidae	Platyrrhini (NWM)	Haplorrhini
Aotus trivirgatus	Aotidae	Platyrrhini (NWM)	Haplorrhini
Callicebus donacophilus	Pitheciidae	Platyrrhini (NWM)	Haplorrhini
Cebus albifrons	Cebidae	Platyrrhini (NWM)	Haplorrhini
Cheirogaleus major	Cheirogaleidae	Lemuroidea	Strepsirrhini
Chlorocebus aethiops	Cercopithecoidea	Catarrhini	Haplorrhini
Colobus guereza	Cercopithecoidea	Catarrhini	Haplorrhini
Daubentonia madagascariensis	Daubentoniidae	Lemuroidea	Strepsirrhini
Galago senegalensis	Galagidae	Lorisidae	Strepsirrhini
Hylobates lar	Hylobatidea	Catarrhini	Haplorrhini
Lemur catta	Lemuridae	Lemuroidea	Strepsirrhini
Lepilemur hubbardorum	Lepilemuridae	Lemuroidea	Strepsirrhini
Loris tardigradus	Lorisidae	Lorisidae	Strepsirrhini
Macaca mulatta	Cercopithecoidea	Catarrhini	Haplorrhini
Microcebus murinus	Cheirogaleidae	Lemuroidea	Strepsirrhini
Nycticebus coucang	Lorisidae	Lorisidae	Strepsirrhini
Otolemur crassicaudatus	Galagidae	Lorisidae	Strepsirrhini
Pan paniscus	Hominoidea	Catarrhini	Haplorrhini
Perodicticus potto	Lorisidae	Lorisidae	Strepsirrhini
Propithecus coquereli	Indriidae	Lemuroidea	Strepsirrhini
Saimiri sciureus	Cebidae	Platyrrhini (NWM)	Haplorrhini
Tarsius syrichta	Tarsiidae		Haplorrhini
Varecia variegata variegata	Lemuridae	Lemuroidea	Strepsirrhini

a posteriori tree.

```
treetrace = readTreeTrace("output/primates_cytb_global_clock.trees",
                          treetype="clock")
mapTree(treetrace,"output/primates_cytb_global_clock.tree")
```

Fill in the following table as you go through the tutorial.

Table 3: Estimated divergence times of the infraorder *Simiiformes**.

	Divergence time estimates	
Clock Model	*Mean Estimate*	*Credible interval*
3.6 Global molecular clock		
?? Root calibration (M_1)		
?? Node calibration (M_2)		

*you can edit this table

→ Look at the file called **`output/primates_cytb_global_clock.tree`** in `FigTree`.

- Add a deterministic node monitoring the age of the *Simiiformes*:

```
clade_simiiformes = clade("Cebus_albifrons", "Macaca_mulatta")
age_simiiformes := tmrca(psi, clade_simiiformes)
```

- Experiment with different prior distributions on the clock rate, e.g., a uniform distribution between 0.005 and 0.02.

4 Root calibration

In the previous section we calibrated the phylogeny using an informative prior on the clock rate. Unfortunately, we don't always have a good estimate of the clock rate. A very common alternative approach is to use an informative prior on the root age. For example, another analysis might have dated the crown age of your group of interested.

In our reference publication, **?** used a normal distribution with mean of 90.0 MYA with stdev = 6.0 as the prior distribution on the root age.

```
root ~ dnNormal(90.0,6.0)
```

```
rootAge := abs(root)
```

Otherwise we will use the same priors and parameters for the tree prior, e.g., a constant-rate birth-death process.

```
a <- 5
b <- 1
diversification ~ dnGamma(shape=a, rate=b)
c <- 5
d <- 1
turnover ~ dnGamma(shape=c,rate=d)

speciation := diversification + turnover
extinction := turnover

sampling_fraction <- 23 / 270

moves[++mi] = mvSlide(diversification,delta=1,tune=true,weight=1)
moves[++mi] = mvSlide(turnover,delta=1,tune=true,weight=1)
moves[++mi] = mvSlide(root,delta=1,tune=true,weight=1)
```

Next, specify the **tree** stochastic node by passing in the tip labels **names** to the **dnBDP()** distribution:

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

moves[++mi] = mvNarrow(psi, weight=5.0)
moves[++mi] = mvNNI(psi, weight=1.0)
moves[++mi] = mvFNPR(psi, weight=3.0)
moves[++mi] = mvSubtreeScale(psi, weight=3.0)
moves[++mi] = mvNodeTimeSlideUniform(psi, weight=15.0)
```

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

4.1 Exercise 2

- Run the analysis and fill in the table with the age estimates.
- Experiment again with different prior choices. What impact do you see of your choices on the prior distribution?

5 Node calibration

This part of the exercise will involve specifying a birth-death model with clamped stochastic nodes representing the prior information on internal nodes in our tree: (1) *Semiiformes*, the split between *New World Monkeys* and *OldWorld Monkeys*, and (2) *galagids and lorisids*, the split between *galagids* and *lorisids*.

In `RevBayes`, calibrated internal nodes are treated differently than in many other programs for estimating species divergence times (e.g., BEAST). This is because the graphical model structure used in `RevBayes` does not allow a stochastic node to be assigned more than one prior distribution. By contrast, the common approach to applying calibration densities as used in other dating softwares leads to incoherence in the calibration prior (for detailed explainations of this see **???**). More explicitly, common calibration approaches assume that the age of a calibrated node is modeled by the tree-wide diversification process (e.g., birth-death model) *and* a parametric density parameterized by the occurrence time of a fossil (or other external prior information). This can induce a calibration prior density that is not consistent with the birth-death process or the parametric prior distribution. Thus, approaches that condition the birth-death process on the calibrated nodes are more statistically coherent (**?**).

In `RevBayes`, calibration densities are applied in a different way, treating fossil observation times like data. The graphical model in Figure 1 illustrates how calibrated nodes are specified in the directed acyclic graph (DAG). Here, the age of the calibration node (i.e., the internal node specified as the MRCA of the fossil and a set of living species) is a deterministic node—e.g., denoted o_1 for fossil $\mathcal{F}_1$—and acts as an offset on the stochastic node representing the age of the fossil specimen. The fossil age, $\mathcal{F}_i$, is specified as a stochastic node and clamped to its *observed* age in the fossil record. The node $\mathcal{F}_i$ is modeled using a distribution that describes the waiting time from the speciation event to the appearance of the observed fossil. Thus, if the MCMC samples any state of Ψ for which the age of $\mathcal{F}_i$ has a probability of 0, then that state will always be rejected, effectively calibrating the birth-death process without applying multiple prior densities to any calibrated node (Fig. 1).

The root age is treated differently, however. Here, we condition the birth-death process on the speciation time of the root, thus this variable is not part of the time-tree parameter. The root age can thus be given any parametric distribution over positive real numbers (Fig. 1).

First, we specify the calibration for the split between *New World Monkeys* and *OldWorld Monkeys*

```
clade_simiiformes = clade("Cebus_albifrons", "Macaca_mulatta")
age_simiiformes := tmrca(psi, clade_simiiformes)
obs_age_simiiformes ~ dnNormal(age_simiiformes,4.5)
obs_age_simiiformes.clamp(43)
```

Next, we specify the calibration for the split between *galagids* and *lorisids*

```
clade_galago_loris = clade("Loris_tardigradus", "Galago_senegalensis")
age_galago_loris := tmrca(psi, clade_galago_loris)
obs_age_galago_loris ~ dnNormal(age_simiiformes,3)
obs_age_galago_loris.clamp(40)
```

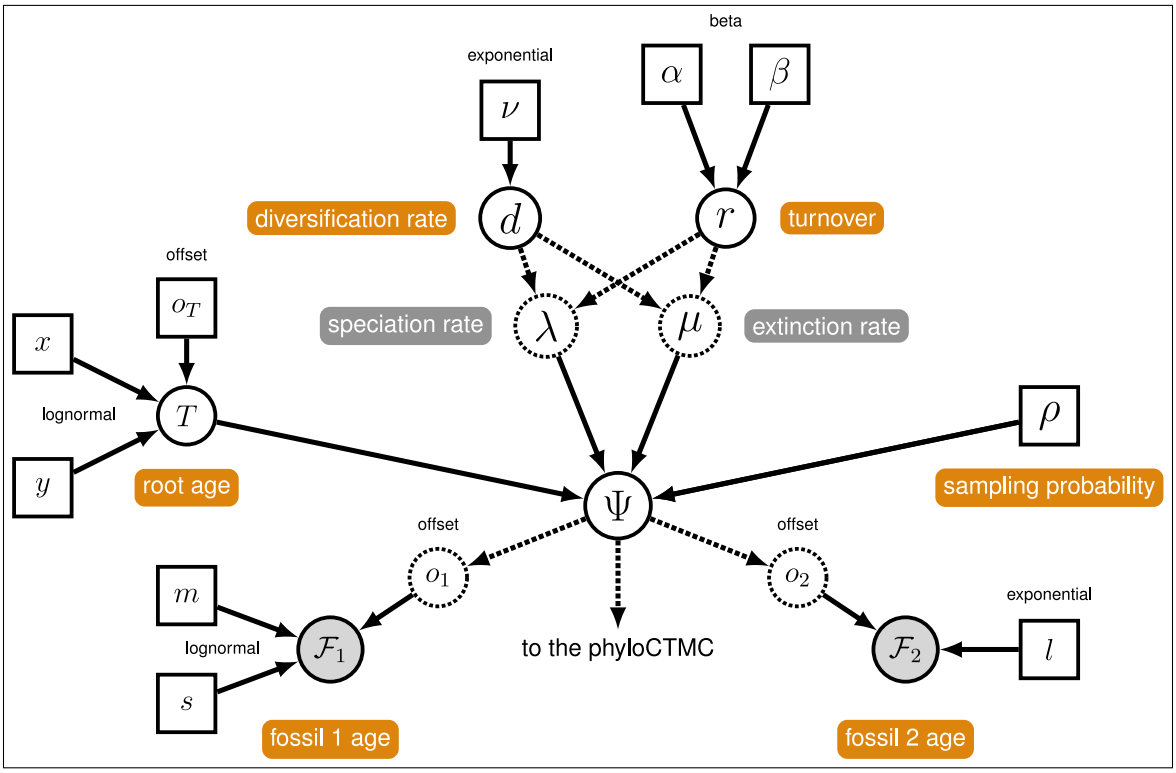

Figure 1: The graphical model representation of the node-calibrated birth-death process in `RevBayes`.

5.1 Exercise 3

- Make the nexessary modifications and runs the analysis.
- Fill in the table.
- Compare the divergence time estimates with the calibration prior.
- Try the **`dnSoftBoundUniformNormal`** instead of a normal distribution as a prior calibration density.

References

Version dated: March 25, 2015