Phylogenetic Inference using RevBayes

Relaxed-Clocks & Calibrated Time Trees

1 Exercise: Estimating Time-Calibrated Phylogenies

1.1 Introduction

Central among the questions explored in biology are those that seek to understand the timing and rates of evolutionary processes. Accurate estimates of species divergence times are vital to understanding historical biogeography, estimating diversification rates, and identifying the causes of variation in rates of molecular evolution.

This tutorial will provide a general overview of divergence time estimation and fossil calibration in a Bayesian framework. The exercise will guide you through the steps necessary for estimating phylogenetic relationships and dating species divergences using the program `RevBayes`.

1.2 Getting Started

The various exercises in this tutorial take you through the steps required to perform phylogenetic analyses of the example datasets. In addition, we have provided the output files for every exercise so you can verify your results. (Note that since the MCMC runs you perform will start from different random seeds, the output files resulting from your analyses *will not* be identical to the ones we provide you.)

- Download data and output files from: http://bit.ly/1oplDTb

In this exercise, we will compare among different relaxed clock models and estimate a posterior distribution of calibrated time trees. The dataset we will use is an alignment of 10 caniform sequences, comprising 8 bears, 1 spotted seal, and 1 gray wolf. Additionally, we will use occurrence times from three caniform fossils to calibrate our analysis to absolute time (Table 1).

Table 1: Fossil species used for calibrating divergence times in the caniform tree.

Fossil species	**Age range (My)**	**Citation**
Hesperocyon gregarius	37.2–40	Wang 1994; Wang et al. 1999
Parictis montanus	33.9–37.2	Clark and Guensburg 1972; Krause et al. 2008
Kretzoiarctos beatrix	11.2–11.8	Abella et al. 2011; Abella et al. 2012

The alignment in file **`data/bears_irbp.nex`** contains interphotoreceptor retinoid-binding protein (irbp) sequences for each extant species.

1.3 Creating Rev Files

This tutorial sets up three different relaxed clock models and a calibrated birth-death model. Because of the complexity of the various models, this exercise is best performed by specifying the models and samplers in different `Rev` files. At the beginning of each section, you will be given a suggested name for each component file; these names correspond to the provided `Rev` scripts that reproduce these commands.

1.4 Calibrating the Birth-Death Model

Fortunately, the fossil record for caniforms (and other carnivores) is quite good. We must formulate a birth-death model that accounts for the fossil occurrence times in Table 1. This part of the exercise will involve specifying a birth-death model with clamped stochastic nodes representing the observation times of two fossils descended from internal nodes in our tree: (1) *Paractis montanus*, the oldest fossil in the family Ursidae, a stem fossil bear, and (2) *Kretzoiarctos beatrix*, the fossil Ailuropodinae, a crown fossil bear. Additionally, we will use the canid fossil, *Hesperocyon gregarius*, to offset the age of the root of the tree.

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 Warnock et al. 2012; Heled and Drummond 2012; Heath et al. 2014). 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 (Yang and Rannala 2006).

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).

Create the Rev File

> Open your text editor and create the birth-death model file called `m_BDP_Tree_bears.Rev` in the `RevBayes_scripts` directory.
>
> Enter the `Rev` code provided in this section in the new model file.

Read in the Starting Tree

When calibrating nodes in the birth-death process, it is very helpful to have a starting tree that is consistent with the topology constraints and calibration priors, otherwise, the probability of the model would be 0

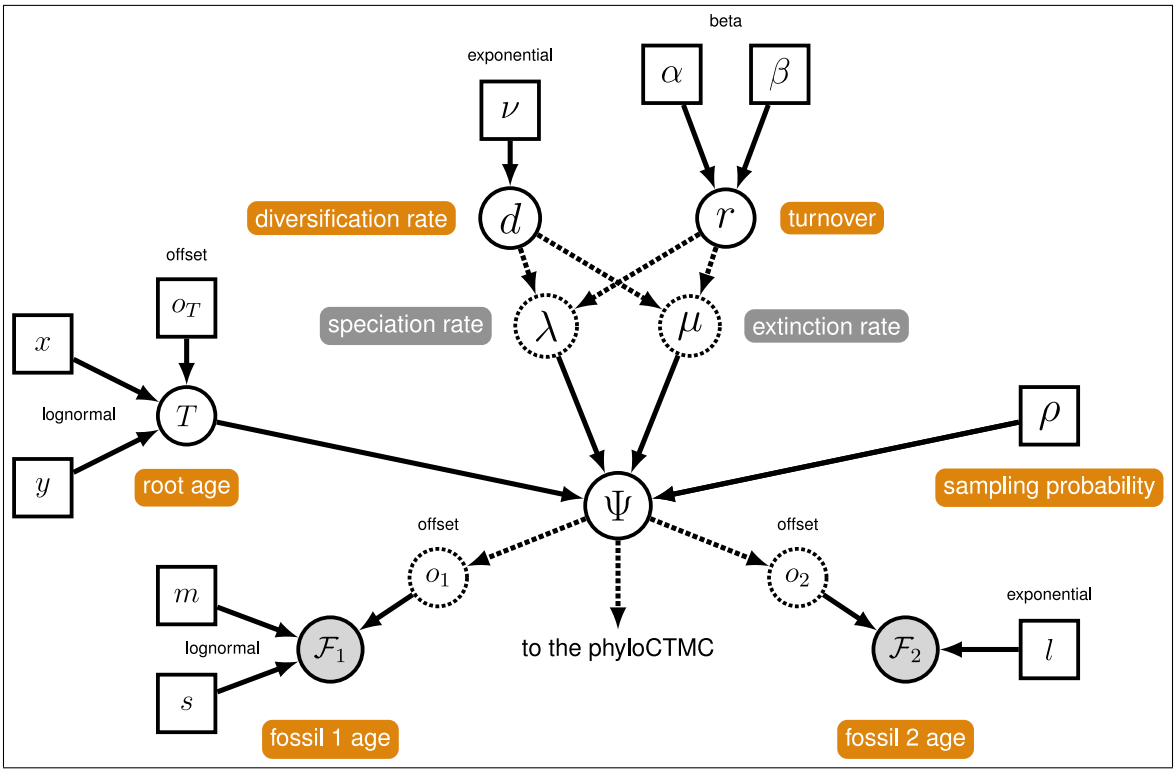

Figure 1: The graphical model representation of the node-calibrated birth-death process in `RevBayes`.

and the MCMC cannot run. For a starting tree we will use the tree estimated by dos Reis et al. (2012).

```
T <- readTrees("data/bears_dosReis.tre")[1]
```

From the tree we can initialize some useful variables.

```
n_taxa <- T.ntips()
names <- T.names()
```

1.4.1 Birth-Death Parameters

We will begin by setting up the model parameters and proposal mechanisms of the birth-death model. Note that we have not initialized the workspace iterator **`mi`** yet. Because of this, if you typed these lines in `RevBayes`, you would get an error. Since this code is intended to be in a sourced `Rev` file, we are assuming that you would initialize **`mi`** before calling **`source("RevBayes_scripts/m_BDP_Tree_bears.Rev")`**.

Diversification

```
diversification ~ dnExponential(10.0)
moves[mi++] = mvScale(diversification,lambda=1.0,tune=true,weight=3.0)
```

Turnover

```
turnover ~ dnBeta(2.0, 2.0)
moves[mi++] = mvSlide(turnover,delta=1.0,tune=true,weight=3.0)
```

Deterministic Nodes for Birth and Death Rates

The birth rate and death rate are deterministic functions of the diversification and turnover. First, create a deterministic node for $1 - r$, which is the denominator for each formula.

```
denom := abs(1.0 - turnover)
```

Now, the rates will both be positive real numbers that are variable transformations of the stochastic variables.

```
birth_rate := diversification / (denom)
death_rate := (turnover * diversification) / (denom)
```

Sampling Probability

Fix the probability of sampling to a known value. Since there are approximately 147 described caniform species, we will create a constant node for this parameter.

```
rho <- 0.068
```

1.4.2 Prior on the Root Node

The fossil *Hesperocyon gregarius* is a fossil descendant of the most-recent common ancestor of all caniformes and has an occurrence time of ∼38 Mya. Thus, we can assume that the probability of the root age being younger than 38 Mya is equal to 0, using this value to offset a prior distribution on the root-age.

First specify the occurrence-time of the fossil.

```
tHesperocyon <- 38.0
```

We will assume a lognormal prior on the root age that is offset by the observed age of *Hesperocyon gregarius*. We can use the previous analysis by dos Reis et al. (2012) to parameterize the lognormal prior on the root time. The age for the MRCA of the caniformes reported in their study was ∼49 Mya. Therefore, we can specify the mean of our lognormal distribution to equal $49 - 38 = 11$ Mya. Given the expected value of the lognormal (**mean_ra**) and a standard deviation (**stdv_ra**), we can also compute the location parameter of the lognormal (**mu_ra**).

```
mean_ra <- 11.0
stdv_ra <- 0.25
mu_ra <- ln(mean_ra) - ((stdv_ra*stdv_ra) * 0.5)
```

With these parameters we can instantiate the root age stochastic node with the offset value.

```
root_time ~ dnLnorm(mu_ra, stdv_ra, offset=tHesperocyon)
```

1.4.3 Topology Constraints & Time Tree

To create the tree with calibrated nodes, we must constrain the topology such that the calibrated nodes always have the same descendants.

The two non-root nodes we are calibrating in this tree is the MRCA of all living bears:

```
clade_Ursidae <- clade("Ailuropoda_melanoleuca","Tremarctos_ornatus","
   Helarctos_malayanus", "Ursus_americanus","Ursus_thibetanus","Ursus_arctos
   ","Ursus_maritimus","Melursus_ursinus")
```

And the MRCA of all bears and pinnipeds.

```
clade_UrsPinn <- clade("Ailuropoda_melanoleuca","Tremarctos_ornatus","
   Helarctos_malayanus", "Ursus_americanus","Ursus_thibetanus","Ursus_arctos
   ","Ursus_maritimus","Melursus_ursinus", "Phoca_largha")
```

Once we have a set of constraints, we can use the vector function **v()** to bind them in a constant vector.

```
constraints <- v(clade_Ursidae, clade_UrsPinn)
```

Now we have all of the elements needed to specify the time-tree parameter.

```
timetree ~ dnBDP(lambda=birth_rate, mu=death_rate, rho=rho, rootAge=
   root_time, samplingStrategy="uniform", condition="nTaxa", nTaxa=n_taxa,
   names=names,constraints=constraints)
```

1.4.4 Calibrating Constrained Nodes

In order that our tree is consistent with the calibration ages, we must first set the value of the time-tree node to our starting tree.

```
timetree.setValue(T)
```

To begin specifying the calibration density on the MRCA of all ursids, we must first create the deterministic node representing the age of the MRCA. The way in which these densities work requires the offset to be negative. Therefore we are creating two deterministic variables, one positive for monitoring, and one negative for the off-set. We use the **tmrca()** function to create these nodes which require that you provide a clade constraint.

```
tmrca_Ursidae := tmrca(timetree,clade_Ursidae)
n_TMRCA_Ursidae := -(tmrca_Ursidae)
```

Now, we must specify our fossil occurrence time. This is the age for the fossil panda, *Kretzoiarctos beatrix*. Note that we also make this value negative.

```
tKretzoiarctos <- -11.2
```

Create the stochastic node for the age of the crown ursid fossil, using a lognormal distribution.

```
M <- 10
sdv <- 0.25
mu <- ln(M) - ((sdv * sdv) * 0.5)
crown_Ursid_fossil ~ dnLnorm(mu, sdv, offset=n_TMRCA_Ursidae)
```

Now clamp the fossil age stochastic node with the observation time of *Kretzoiarctos beatrix*

```
crown_Ursid_fossil.clamp(tKretzoiarctos)
```

Next we will create the variable for the age of the MRCA of all bears and pinnipeds.

```
tmrca_UrsidaePinn := tmrca(timetree,clade_UrsPinn)
n_TMRCA_UrsidaePinn := -(tmrca_UrsidaePinn)
```

Set the observed time for the stem fossil bear.

```
tParictis <- -33.9
```

Create the stochastic node using the exponential prior and clamp it with the observation time of the fossil.

```
stem_Ursid_fossil ~ dnExponential(lambda=0.0333, offset=n_TMRCA_UrsidaePinn)
stem_Ursid_fossil.clamp(tParictis)
```

1.4.5 Proposals on the Time Tree

Next, create the vector of moves. These tree moves act on node ages:

```
moves[mi++] = mvNodeTimeSlideUniform(timetree, weight=30.0)
moves[mi++] = mvSlide(root_time, delta=2.0, tune=true, weight=10.0)
moves[mi++] = mvScale(root_time, lambda=2.0, tune=true, weight=10.0)
moves[mi++] = mvTreeScale(tree=timetree, rootAge=root_time, delta=1.0, tune=
   true, weight=3.0)
```

And these change the tree topology. If we wish to keep the topology constant, then we can leave these moves out. Note that for some relaxed clock models (autocorrelated rates, DPP, random-local clock) tree topology moves often induce very long mixing times.

```
moves[mi++] = mvNNI(timetree, weight=8.0)
moves[mi++] = mvNarrow(timetree, weight=8.0)
moves[mi++] = mvFNPR(timetree, weight=8.0)
```

1.5 Specifying Branch-Rate Models

The next sections will walk you through setting up the files specifying different relaxed clock models. You may type this syntax directly into the `RevBayes` console or write them in a text file.

1.5.1 The Global Molecular Clock Model

The global molecular clock assumes that the rate of substitution is constant over the tree and over time. When estimating trees on an absolute time-scale, it is often necessary to parameterize relaxed clock models with two rates, a base rate which effectively scales the tree and a clock rate. Then, the absolute rate applied to the tree is a deterministic node (Fig. 2).

Load Sequence Alignment

Read in the sequences and initialize important variables.

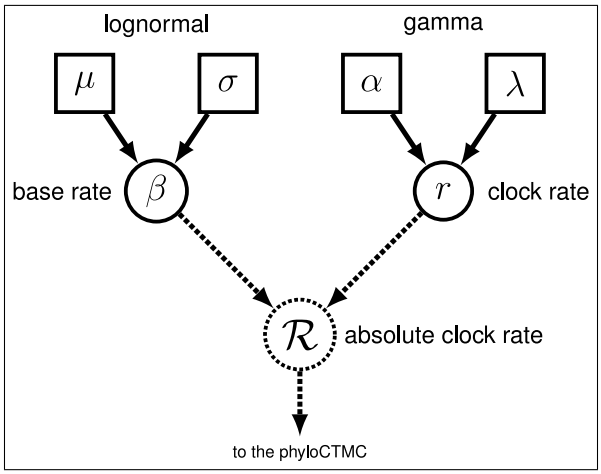

Figure 2: The graphical model representation of the global molecular clock model used in this exercise.

```
D <- readDiscreteCharacterData(file="data/bears_irbp.nex")
n_sites <- D.nchar(1)
mi = 1
```

The Calibrated Time-Tree Model

Load the calibrated tree model from file. Note that this file does not have moves that operate on the tree topology, which is helpful when you plan to estimate the marginal likelihoods and compare different relaxed clock models.

```
source("RevBayes_scripts/m_BDP_bears.Rev")
```

The Clock-Rate

We specify the absolute clock rate by first creating a node for the base rate. This value is set to be drawn from a lognormal prior.

```
br_M <- 5.4E-3
br_s <- 0.05
br_mu <- ln(br_M) - ((br_s * br_s) * 0.5)
base_rate ~ dnLnorm(br_mu, br_s)
moves[mi++] = mvScale(base_rate,lambda=0.25,tune=true,weight=5.0)
```

The clock-rate parameter is a stochastic node from a gamma distribution.

```
clock_rate ~ dnGamma(2.0,4.0)
moves[mi++] = mvScale(clock_rate,lambda=0.5,tune=true,weight=5.0)
```

The absolute clock rate is the value on which the phylogenetic CTMC model depends. This is a deterministic node and equal to the product of the base rate and clock rate.

```
abs_clock_rt := clock_rate * base_rate
```

The Sequence Model and Tree Plate

Specify the parameters of the GTR model and the moves to operate on them.

```
sf ~ dnDirichlet(v(1,1,1,1))
er ~ dnDirichlet(v(1,1,1,1,1,1))
Q := fnGTR(er,sf)
moves[mi++] = mvSimplexElementScale(er, alpha=10.0, tune=true, weight=3.0)
moves[mi++] = mvSimplexElementScale(sf, alpha=10.0, tune=true, weight=3.0)
```

And instantiate the phyoCTMC.

```
phySeq ~ dnPhyloCTMC(tree=timetree, Q=Q, branchRates=abs_clock_rt, nSites=
   n_sites, type="DNA")
phySeq.clamp(D)
```

Specify the work-space model object.

```
mymodel = model(er)
```

- The global-molecular clock model is specified in the file called **m_GMC_bears.Rev** .

Estimate the Marginal Likelihood

Below is the code to estimate the marginal likelihood. Enter the value for this model in Table 2.

- The strict clock marginal likelihood analysis is specified in the file called **mlnl_GMC_bears.Rev** .

```
source("RevBayes_scripts/m_GMC_bears.Rev")

pow_p = powerPosterior(mymodel, moves, "output/GMC_bears_powp.out", cats=50)
pow_p.burnin(generations=5000,tuningInterval=200)
pow_p.run(generations=1000)

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

### use path sampling to calculate marginal likelihoods
ps = pathSampler(file="output/GMC_bears_powp.out", powerColumnName="power",
   likelihoodColumnName="likelihood")
ps.marginal()
```

1.5.2 The Uncorrelated Lognormal Rates Model

The uncorrelated lognormal model relaxes the assumption of a single-rate molecular clock. Under this model, the rate associated with each branch in the tree is a stochastic node. Each branch-rate variable is drawn from the same lognormal distribution (Fig. 3).

Given that we might not have prior information on the parameters of the lognormal distribution, we can assign hyper priors to these variables. Generally, it is more straightforward to construct a hyperprior on the expectation (i.e., the mean) of a lognormal density rather than the location parameter μ. Here, we will assume that the mean branch rate is exponentially distributed and as is the stochastic node representing the standard deviation. With these two parameters, we can get the location parameter of the lognormal by:

$$\mu = \log(M) - \frac{\sigma^2}{2}.$$

Thus, μ is a deterministic node, which is a function of M and σ.

In Figure 3, we can represent the vector of N branch rates using the plate notation. Additionally, each branch rate is rescaled by the base rate.

First, clear the workspace.

```
clear()
```

Load the Sequence Data and Birth-Death Model

```
D <- readDiscreteCharacterData(file="data/bears_irbp.nex")
n_sites <- D.nchar(1)
mi = 1
```

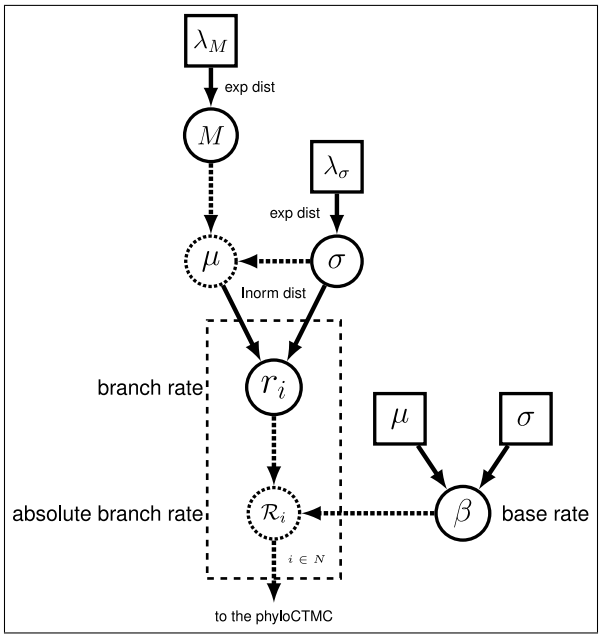

Figure 3: The graphical model representation of the UCLN model used in this exercise.

```
source("RevBayes_scripts/m_BDP_bears.Rev")
```

The Base Clock Rate

As in the clock model above, we create a lognormally distributed stochastic node, representing the base rate.

```
br_M <- 5.4E-3
br_s <- 0.05
br_mu <- ln(br_M) - ((br_s * br_s) * 0.5)
base_rate ~ dnLnorm(br_mu, br_s)
moves[mi++] = mvScale(base_rate,lambda=0.25,tune=true,weight=5.0)
```

Independent Branch Rates

Before we can set up the variable of the branch-rate model, we must know how many branches exist in the tree.

```
n_branches <- 2 * n_taxa - 2
```

We will start with the mean of the lognormal distribution, M in Figure 3.

```
ucln_mean ~ dnExponential(2.0)
```

And the exponentially distributed node representing the standard deviation. We will also create a deterministic node, which is the variance, σ^2.

```
ucln_sigma ~ dnExponential(3.0)
ucln_var := ucln_sigma * ucln_sigma
```

Now we can declare the function that gives us the μ parameter of the lognormal distribution on branch rates.

```
ucln_mu := ln(ucln_mean) - (ucln_var * 0.5)
```

The only stochastic nodes we need to operate on for this part of the model are the lognormal mean (M or **ucln_mean**) and the standard deviation (σ or **ucln_sigma**).

```
moves[mi++] = mvScale(ucln_mean, lambda=1.0, tune=true, weight=4.0)
moves[mi++] = mvScale(ucln_sigma, lambda=0.5, tune=true, weight=4.0)
```

With our nodes representing the μ and σ of the lognormal distribution, we can create the vector of stochastic nodes for each of the branch rates using a **for** loop. Within this loop, we also add the move for each branch-rate stochastic node to our moves vector.

```
for(i in 1:n_branches){
   branch_rates[i] ~ dnLnorm(ucln_mu, ucln_sigma)
   moves[mi++] = mvScale(branch_rates[i], lambda=1, tune=true, weight=2.)
}
```

Because we are dealing with semi-identifiable parameters, it often helps to apply a range of moves to the variables representing the branch rates and branch times. This will help to improve the mixing of our MCMC. Here we will add 2 additional types of moves that act on vectors.

```
moves[mi++] = mvVectorScale(branch_rates,lambda=1.0,tune=true,weight=2.0)
moves[mi++] = mvVectorSingleElementScale(branch_rates,lambda=30.0,tune=true,
   weight=1.0)
```

We can combine the base rate and branch rates in a vector of deterministic nodes.

```
branch_subrates := branch_rates * base_rate
```

The mean of the branch rates is a convenient deterministic node to monitor, particularly in the screen output when conducting MCMC.

```
mean_rt := mean(branch_rates)
```

The Sequence Model and Tree Plate

Now, specify the stationary frequencies and exchangeability rates of the GTR matrix.

```
sf ~ dnDirichlet(v(1,1,1,1))
er ~ dnDirichlet(v(1,1,1,1,1,1))
Q := fnGTR(er,sf)
moves[mi++] = mvSimplexElementScale(er, alpha=10.0, tune=true, weight=3.0)
moves[mi++] = mvSimplexElementScale(sf, alpha=10.0, tune=true, weight=3.0)
```

Now, we can put the whole model together in the phylogenetic CTMC and clamp that node with our sequence data.

```
phySeq ~ dnPhyloCTMC(tree=timetree, Q=Q, branchRates=branch_subrates, nSites
   =n_sites, type="DNA")
attach the observed sequence data
phySeq.clamp(D)
```

```
mymodel = model(er)
```

- The UCLN model is specified in the file called **m_UCLN_bears.Rev** .

Estimate the Marginal Likelihood

Below is the code to estimate the marginal likelihood. Enter the value for this model in Table 2.

- The UCLN marginal likelihood analysis is specified in the file called **mlnl_UCLN_bears.Rev** .

```
source("RevBayes_scripts/m_GMC_bears.Rev")

pow_p = powerPosterior(mymodel, moves, "output/UCLN_bears_powp.out", cats
   =50)
```

```
pow_p.burnin(generations=5000,tuningInterval=200)
pow_p.run(generations=1000)

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

### use path sampling to calculate marginal likelihoods
ps = pathSampler(file="output/GMC_bears_powp.out", powerColumnName="power",
   likelihoodColumnName="likelihood")
ps.marginal()
```

1.6 Compute Bayes Factors and Select Model

Now that we have estimates of the marginal likelihood under each of our different models, we can evaluate their relative plausibility using Bayes factors. Use Table 2 to summarize the marginal log-likelihoods estimated using the stepping-stone and path-sampling methods.

Table 2: Estimated marginal likelihoods for different partition configurations*.

	Marginal lnL estimates	
Partition	*Stepping-stone*	*Path sampling*
Global molecular clock (M_0)		
Uncorrelated lognormal (M_1)		

*you can edit this table

Phylogenetics software programs log-transform the likelihood to avoid underflow, because multiplying likelihoods results in numbers that are too small to be held in computer memory. Thus, we must calculate the ln-Bayes factor (we will denote this value $\mathcal{K}$):

$$\mathcal{K} = \ln[BF(M_0, M_1)] = \ln[\mathbb{P}(\mathbf{X} \mid M_0)] - \ln[\mathbb{P}(\mathbf{X} \mid M_1)], \tag{1}$$

where $\ln[\mathbb{P}(\mathbf{X} \mid M_0)]$ is the *marginal lnL* estimate for model M_0. The value resulting from equation 1 can be converted to a raw Bayes factor by simply taking the exponent of $\mathcal{K}$

$$BF(M_0, M_1) = e^{\mathcal{K}}. \tag{2}$$

Alternatively, you can interpret the strength of evidence in favor of M_0 using the $\mathcal{K}$ and skip equation 2. In this case, we evaluate the $\mathcal{K}$ in favor of model M_0 against model M_1 so that:

if $\mathcal{K} > 1$, then model M_0 wins
if $\mathcal{K} < -1$, then model M_1 wins.

Thus, values of $\mathcal{K}$ around 0 indicate ambiguous support.

Table 3: Bayes factor calculation*.

	ln-Bayes Factor ($\mathcal{K}$)	
Model comparison	*Stepping-stone*	*Path sampling*
M_0, M_1		
Supported model?		

*you can edit this table

Using the values you entered in Table 2 and equation 1, calculate the ln-Bayes factors (using $\mathcal{K}$) for the different model comparisons. Enter your answers in Table 3 using the stepping-stone and the path-sampling estimates of the marginal log likelihoods.

1.7 Estimate the Topology and Branch Times

Below the code is provided to run the MCMC analysis under the UCLN model while estimating the topology and branching times.

```
### Load the sequence alignment
D <- readDiscreteCharacterData(file="data/bears_irbp.nex")

### get helpful variables from the data
n_taxa <- D.ntaxa()
n_sites <- D.nchar(1)
names <- D.names()

### initialize an iterator for the moves vector
mi = 1

### set up the birth-death model from file
### this file includes tree topology moves
source("RevBayes_scripts/m_BDP_Tree_bears.Rev")

#####################################
############ UCLN model ############
#####################################

source("RevBayes_scripts/m_UCLN_bears.Rev")
```

Set up the MCMC

```
### workspace model wrapper ###
```

```
mymodel = model(er)

monitors[1] = mnFile(filename="output/TimeTree_bears_mcmc.log", printgen=10,
   diversification, turnover, birth_rate, death_rate, root_time,
   tmrca_Ursidae, tmrca_UrsidaePinn,er, sf,mean_rt, ucln_mean, ucln_sigma,
   branch_rates, base_rate)
monitors[2] = mnFile(filename="output/TimeTree_bears_mcmc.trees", printgen
   =10, timetree)
monitors[3] = mnScreen(printgen=10, mean_rt, ucln_mean, root_time, base_rate
   )

### workspace mcmc ###
mymcmc = mcmc(mymodel, monitors, moves)

### pre-burnin to tune the proposals ###
mymcmc.burnin(generations=3000,tuningInterval=100)

### run the MCMC ###
mymcmc.run(generations=50000)

### display proposal acceptance rates and tuning ###
mymcmc.operatorSummary()
```

Summarize the tree

```
### summarize the trees ###
tt = readTreeTrace("output/TimeTree_bears_mcmc.trees", "clock")
tt.summarize()

### write MAP tree to file
mapTree(tt, "output/TimeTree_bears_mcmc_MAP.tre")

## quit ##
q()
```

Useful Links

- RevBayes: https://github.com/revbayes/code
- TreePar: http://cran.r-project.org/web/packages/TreePar/index.html
- Tree Thinkers: http://treethinkers.org

Questions about this tutorial can be directed to:

- Tracy Heath (email: tracyh@berkeley.edu)
- Tanja Stadler (email: tanja.stadler@bsse.ethz.ch)
- Sebastian Höhna (email: sebastian.hoehna@gmail.com)

Version dated: November 10, 2014

Relavent References

Abella, J, P Montoya, and J Morales. 2011. Una nueva especie de *Agriarctos* (Ailuropodinae, Ursidae, Carnivora) en la localidad de Nombrevilla 2 (Zaragoza, España). *Estudios Geológicos* 67: 187–191.

Abella, J, DM Alba, JM Robles, A Valenciano, C Rotgers, R Carmona, P Montoya, and J Morales. 2012. *Kretzoiarctos* gen. nov., the oldest member of the giant panda clade. *PLoS One* 17: e48985.

Clark, J and TE Guensburg. 1972. *Arctoid Genetic Characters as Related to the Genus* Parictis. Vol. 1150. Field Museum of Natural History, Chicago, Ill.

dos Reis, M, J Inoue, M Hasegawa, R Asher, P Donoghue, and Z Yang. 2012. Phylogenomic datasets provide both precision and accuracy in estimating the timescale of placental mammal phylogeny. *Proceedings of the Royal Society B: Biological Sciences* 279: 3491–3500.

Heath, TA, JP Huelsenbeck, and T Stadler. 2014. The fossilized birth-death process for coherent calibration of divergence-time estimates. *Proceedings of the National Academy of Sciences, USA* 111: E2957–E2966.

Heled, J and AJ Drummond. 2012. Calibrated tree priors for relaxed phylogenetics and divergence time estimation. *Systematic Biology* 61: 138–149.

Krause, J, T Unger, A Noçon, AS Malaspinas, SO Kolokotronis, M Stiller, L Soibelzon, H Spriggs, PH Dear, AW Briggs, et al. 2008. Mitochondrial genomes reveal an explosive radiation of extinct and extant bears near the Miocene-Pliocene boundary. *BMC Evolutionary Biology* 8: 220.

Wang, X. 1994. *Phylogenetic Systematics of the Hesperocyoninae (Carnivora, Canidae). Bulletin of the AMNH.* Vol. 221.

Wang, X, RH Tedford, and BE Taylor. 1999. *Phylogenetic Systematics of the Borophaginae (Carnivora, Canidae). Bulletin of the AMNH.* Vol. 243.

Warnock, RCM, Z Yang, and PCJ Donoghue. 2012. Exploring the uncertainty in the calibration of the molecular clock. *Biology Letters* 8: 156–159.

Yang, Z and B Rannala. 2006. Bayesian estimation of species divergence times under a molecular clock using multiple fossil calibrations with soft bounds. *Molecular Biology and Evolution* 23: 212–226.