# Phylogenetic Inference using RevBayes

*Simple Birth-Death Process Models*

Tracy A. Heath

## 1 Exercise: Estimating Speciation & Extinction Rates

### 1.1 Introduction

Models of speciation and extinction are fundamental to any phylogenetic analysis of macroevolutionary processes. A prior describing the distribution of speciation events over time is critical to estimating phylogenies with branch lengths proportional to time. Moreover, stochastic branching models allow for inference of speciation and extinction rates. These inferences allow us to investigate key questions in evolutionary biology.

This tutorial describes how to specify simple branching-process models in `RevBayes`; two variants of the constant-rate birth-death process (??????). The probabilistic graphical model is given for each component of this exercise. After each model is specified, you will estimate the marginal likelihood of the model and evaluate the relative support using Bayes factors. Finally, you will estimate speciation and extinction rates using Markov chain Monte Carlo (MCMC) under the model supported by the data.

### 1.2 The "Observed" Data

- Download data and `Rev` files from: http://bit.ly/1tEDwTg

The tree in this exercise contains a subset of the taxa resulting from the divergence-time analysis of 274 placental mammal species by ?. The tree comprises all living bear species (8 taxa) plus two outgroups—the gray wolf and spotted seal. Thus, the root of this tree represents the most-recent common ancestor of all living members of the suborder Caniformia (Fig. 1). The phylogenetic relationships and speciation times are the median estimates reported by ?. In this exercise, this time tree is treated as an "observation", and we are estimating parameters of the birth-death model without accounting for uncertainty in the time tree.

- Open the tree **`data/bears_dosReis.nex`** in FigTree.

## 2 Specifying Constant-Rate Birth-Death Models

Before evaluating the relative support for different models, we must first specify them in the `Rev` language. In this section, we will walk through specifying three different variants of the birth-death process model and estimating the marginal likelihood under each one.

### 2.1 Pure-Birth Model

The simplest branching model is the *pure-birth process* described by ?. Under this model, we assume at any instant in time, every lineage has the same speciation rate $\lambda$. In its simplest form, the speciation rate

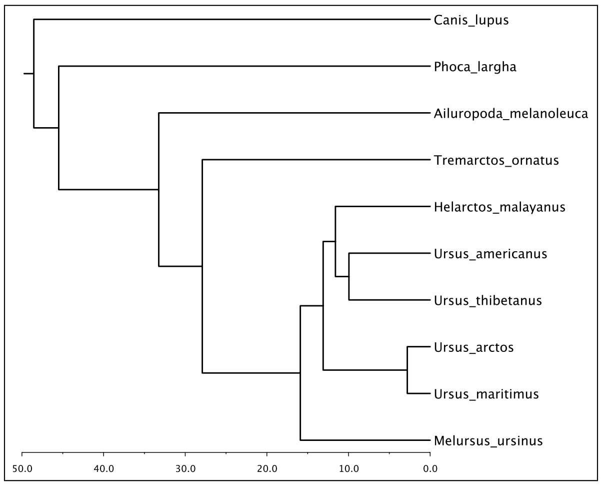

Figure 1: The relationships and median speciation times for bears and two outgroups estimated in the analysis by **?**.

remains constant over time. As a result, the waiting time between speciation events is exponential, where the rate of the exponential distribution is the product of the number of extant lineages ($n$) at that time and the speciation rate: $n\lambda$ (**???**). The pure-birth branching model does not allow for lineage extinction (this is similar to population-level coalescent models). However, the model depends on a second parameter $\rho$ which is the probability of sampling a species in the present time as well as the time of the start of the process, whether that is the origin time or root age. Therefore, the probabilistic graphical model of the pure-birth process is quite simple, where the observed time tree topology and node ages are conditional on the speciation rate, sampling probability, and root age (Fig. 2).

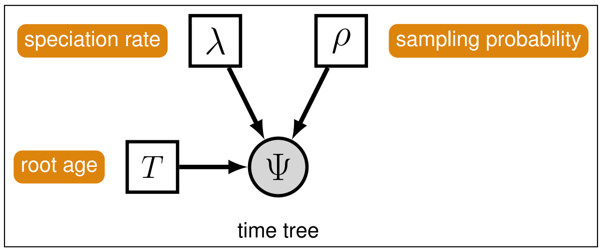

Figure 2: The graphical model representation of the pure-birth (Yule) process.

We can add hierarchical structure to this model and account for uncertainty in the value of the speciation rate by placing a hyperprior on $\lambda$ (Fig. 3). The graphical models in Figures 2 and 3 demonstrate the simplicity of the Yule model. Ultimately, the pure birth model is just a special case of the birth-death process, where the extinction rate (typically denoted $\mu$) is a constant node with the value 0.

For this exercise, we will specify a Yule model, such that the speciation rate is a stochastic node, drawn from an exponential distribution as in Figure 3. In a Bayesian framework, we are interested in estimating the posterior probability of $\lambda$ given that we observe a time tree.

$$\mathbb{P}(\lambda \mid \Psi) = \frac{\mathbb{P}(\Psi \mid \lambda)\mathbb{P}(\lambda \mid \nu)}{\mathbb{P}(\Psi)} \tag{1}$$

In this example, we have a phylogeny of all living bears plus two outgroup species, the gray wolf and spotted seal. We are treating the time tree $\Psi$ as an observation, thus clamping the model with an observed value.

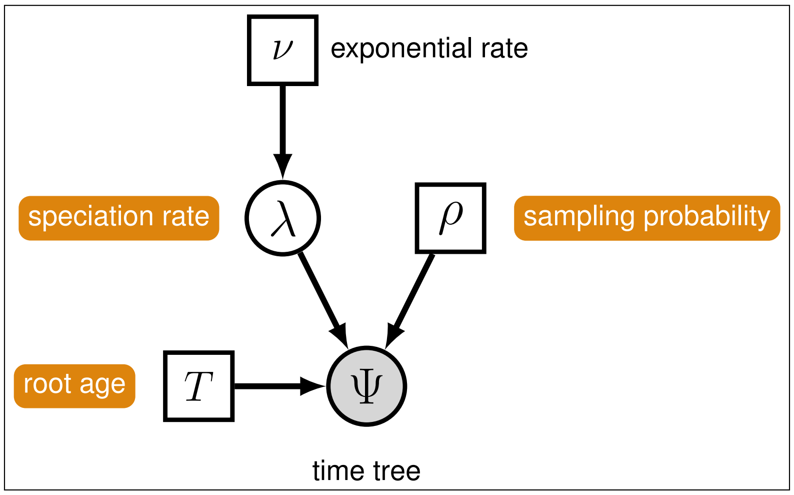

Figure 3: The graphical model representation of the pure-birth (Yule) process, where the speciation rate is treated as a random variable drawn from an exponential distribution with rate parameter $\nu$.

The time tree we are conditioning the process on is taken from the analysis by **?** and shown in Figure 1. Furthermore, there are approximately 147 described caniform species, so we will fix the parameter $\rho$ to 10/147.

- The full Yule-model specification is in the file called `m_Yule_bears.Rev`.

### 2.1.1 Read the tree

Begin by reading in the observed tree from Figure 1.

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

From this tree, we can get some helpful variables:

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

Additionally, we can initialize an iterator variable for our vector of moves:

```
mi = 1
```

### 2.1.2 Birth rate

The model we are specifying only has three nodes (Fig. 3). We can specify the birth rate $\lambda$, the rate-parameter $\nu$ of the exponential hyperprior on $\lambda$, and the conditional dependency of the two parameters all in one line of `Rev` code.

```
birth_rate ~ dnExponential(0.1)
```

Here, the stochastic node called **birth_rate** represents $\lambda$ and the **0.1** is the constant node $\nu$, given the value 0.1. Note that this value leads to an expected value for $\lambda$ of 10:

$$\mathbb{E}[\lambda] = \nu^{-1} = 10$$

To estimate the value of $\lambda$, we assign a proposal mechanism to operate on this node. In RevBayes these MCMC sampling algorithms are called *moves*. We need to create a vector of moves and we can do this by using vector indexing and our pre-initialized iterator **mi**. We will use a scaling move on $\lambda$ called **mvScale**.

```
moves[mi++] = mvScale(birth_rate,lambda=1,tune=true,weight=3)
```

### 2.1.3 Sampling probability

Our prior belief is that we have sampled 10 out of 147 living caniform species. To account for this we can set the sampling parameter as a constant node with a value of 0.068

```
rho <- 0.068
```

### 2.1.4 Root age

Any stochastic branching process must be conditioned on a time that represents the start of the process. Typically, this parameter is the *origin time* and it is assumed that the process started with *one* lineage. Thus, the origin of a birth-death process is the node that is *ancestral* to the root node of the tree. For macroevolutionary data, particularly without any sampled fossils, it is difficult to use the origin time. To accommodate this, we can condition on the age of the root by assuming the process started with *two* lineages that both originate at the time of the root.

We can get the value for the root from the **?** tree.

```
root_time <- treeHeight(T)
```

### 2.1.5 The time tree

Now we have all of the parameters we need to specify the full pure-birth model. We can initialize the stochastic node representing the time tree. Note that we set the **mu** parameter to the constant value **0.0**.

```
timetree ~ dnBDP(lambda=birth_rate, mu=0.0, rho=rho, rootAge=root_time,
    samplingStrategy="uniform", condition="nTaxa", nTaxa=n_taxa, names=names)
```

If you refer back to Equation 1 and Figure 3, the time tree $\Psi$ is the variable we observe, i.e., the data. We can set this in the Rev language by using the **clamp()** function.

```
timetree.clamp(T)
```

Here we are fixing the value of the time tree to our observed tree from **?**. If we did not clamp this node, and ran MCMC, we would simply simulate time trees under the model.

Finally, we can create a workspace object of our whole model using the **model()** function. Workspace objects are initialized using the **=** operator. This distinguishes the objects used by the program to run the MCMC analysis from the distinct nodes of our graphical model. The model workspace objects makes it easy to work with the model in the Rev language and creates a wrapper around our model DAG. Because our model is a directed, acyclic graph (DAG), we only need to give the model wrapper function a single node and it does the work to find all the other nodes through their connections.

```
mymodel = model(birth_rate)
```

The **model()** function traversed all of the connections and found all of the nodes we specified. We can now visualize our graphical model using the **.graph()** member method of the model object. This function writes a file in the DOT graph-description language. The contents of this file describes the nodes and edges of the model DAG and can be read by an interpreter program called Graphviz. First create the model graph file using the **.graph()** method. Set the flag for extra output (good for development debugging) to false: **verbose=false**. And specify a named RBG color for the background (for this graph, we like **"honeydew2"**).

```
mymodel.graph("output/m_Yule_bears_GM.dot", verbose=false, bg="honeydew2")
```

Open the **output/m_Yule_bears_GM.dot** file in the Graphviz program or paste the contents in an online viewer:

- http://graphviz-dev.appspot.com/
- http://stamm-wilbrandt.de/GraphvizFiddle/

Your graph should look like the one depicted in Figure 4. Compare this figure to the model in Figure 3. You should notice that there are two extra nodes in the Figure 4. The constant node with the value **0** connected to the **birth_rate** stochastic node represents the **offset** variable of the exponential distribution which is given the default value of 0 when no offset is provided. Additionally, there is a nameless constant node with the value of **0** pointing into the clamped **timetree** stochastic node. This constant node represents the death rate, **mu**, which we set to **0** when we initialized **timetree** using the **dnBDP()** constructor function. Viewing the model graph is helpful for identifying any problems prior to running MCMC.

### 2.1.6 Estimating the marginal likelihood of the model

With a fully specified model, we can set up the **powerPosterior()** analysis to create a file of 'powers' and likelihoods from which we can estimate the marginal likelihood using stepping-stone or path sampling.

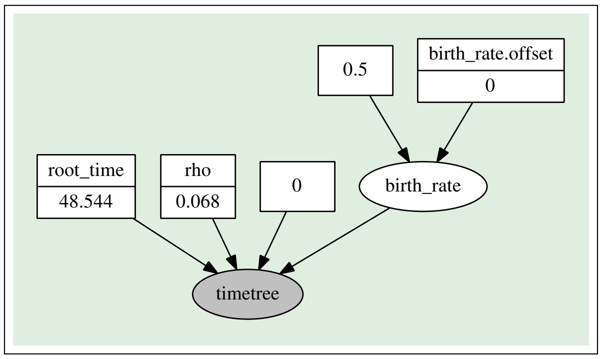

Figure 4: The graphical model representation of the pure-birth (Yule) process generated using the DOT language and the Graphviz program.

This method computes a vector of powers from a beta distribution, then executes an MCMC run for each power step while raising the likelihood to that power. In this implementation, the vector of powers starts with 1, sampling the likelihood close to the posterior and incrementally sampling closer and closer to the prior as the power decreases.

- The `Rev` file for performing this analysis: `mlnl_Yule_bears.Rev`.

First, we create the variable containing the power posterior. This requires us to provide a model and vector of moves, as well as an output file name. The **`cats`** argument sets the number of power steps.

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

We can start the power posterior by first burning in the chain and and discarding the first 10000 states.

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

Now execute the run with the **`.run()`** function:

```
pow_p.run(generations=1000)
```

Once the power posteriors have been saved to file, create a stepping stone sampler. This function can read any file of power posteriors and compute the marginal likelihood using stepping-stone sampling.

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

Compute the marginal likelihood under stepping-stone sampling using the member function **`marginal()`** of the **`ss`** variable and record the value in Table 1.

```
ss.marginal()
```

Path sampling is an alternative to stepping-stone sampling and also takes the same power posteriors as input.

```
ps = pathSampler(file="output/Yule_bears_powp.out", powerColumnName="power",
    likelihoodColumnName="likelihood")
```

Compute the marginal likelihood under stepping-stone sampling using the member function **`marginal()`** of the **`ps`** variable and record the value in Table 1.

```
ps.marginal()
```

## 3 Birth-Death Process

The pure-birth model does not account for extinction, thus it assumes that every lineage at the start of the process will have sampled descendants at time 0. This assumption is fairly unrealistic for most phylogenetic datasets on a macroevolutionary time scale since the fossil record provides evidence of extinct lineages. **?** described a more general branching process model to account for lineage extinction called the *birth-death process*. Under this model, at any instant in time, every lineage has the same rate of speciation $\lambda$ and the same rate of extinction $\mu$. This is the *constant-rate* birth-death process, which considers the rates constant over time and over the tree (**?**). Importantly, this model assumes that all of the extant descendants of the process have been sampled at time 0.

**?** derived the probability of time trees under an extension of the birth-death model that accounts for incomplete sampling of the tips (Fig. 5). Under this model, the parameter $\rho$ accounts for the probability of sampling in the present time, and because it is a probability, this parameter can only take values between 0 and 1.

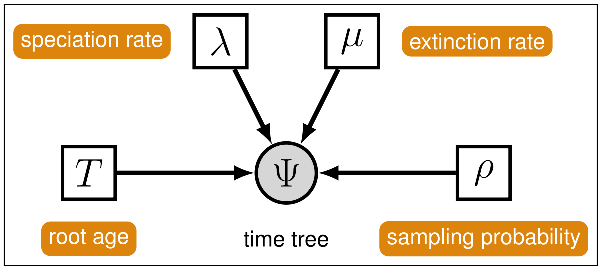

Figure 5: The graphical model representation of the birth-death process with uniform sampling and conditioned on the root age.

Ultimately, it is difficult to formulate prior densities on rate parameters, particularly when our uncertainty in the values of the speciation and extinction rates is quite large. Furthermore, without sampling the process back in time, it is difficult to estimate extinction. Thus, we can re-parameterize the birth-death process to account for these issues. In this parameterization, we use the net diversification rate $d$ and the turnover rate $r$ (also called relative extinction rate) instead of the $\lambda$, $\mu$ parameters.

$$\begin{aligned} d &= \lambda - \mu && \text{Net diversification rate} \\ r &= \mu/\lambda && \text{Turnover} \end{aligned}$$

Importantly, we can recover $\lambda$ and $\mu$ via:

$$\lambda = \frac{d}{1-r}, \quad \mu = \frac{rd}{1-r}. \tag{2}$$

Thus, $\lambda$ and $\mu$ are deterministic nodes, transformed from $d$ and $r$. By using the diversification and turnover parameters, we now have another variable, $r$ that can only take values between 0 and 1. This is because, under the constant-rate birth-death process, $\mu$ can never be greater than $\lambda$ (Fig. 6). Note that if $\mu = 0$, then $d = \lambda$ in this parameterization.

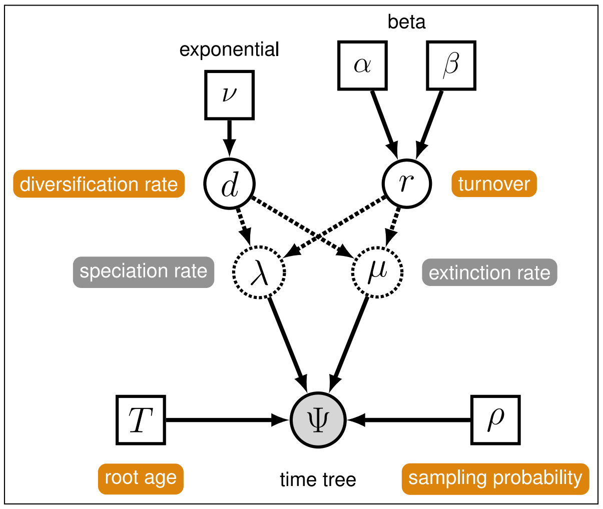

Figure 6: The graphical model representation of the birth-death process with uniform sampling parameterized using the diversification and turnover.

In this model, we will specify an exponential prior density on $d$ and a beta prior on $r$. There are approximately 147 described caniform species, so we will fix the parameter $\rho$ to 10/147.

- The full birth-death, fixed sampling model specification is in the file called `m_BD_bears.Rev`.

## 3.1 Clear the workspace and read the tree

It is best to remove all of the previous model variables created in the previous section.

```
clear()
```

Now read in the observed tree from Figure 1.

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

Initialize the useful variables:

```
n_taxa <- T.ntips()
names <- T.names()
mi = 1
```

## 3.2 Diversification and turnover

The diversification and turnover are the parameters which we will treat as stochastic nodes in our model. We will assume an exponential prior on **`diversification`** and assign it a scale move.

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

The **`turnover`** parameter can only take values between 0 and 1, thus we will assume a beta prior on this parameter and sample from the posterior distribution using a scale move.

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

## 3.3 Birth rate and death rate

The birth and death rates are both deterministic nodes. Refer to Equation 2. Note that both the birth rate and death rate are functions of $d$ and $r$.

Because our variable transformations use the **`-`** operator, we must additionally use the **`abs()`** function to ensure that the rates are of type **`RealPos`**, which is required by the birth-death process model.

```
birth_rate := abs(diversification / (1.0 - turnover))
```

```
death_rate := abs(turnover * diversification / (1.0 - turnover))
```

## 3.4 The sampling probability

If we assume that the 147 described caniform species represent all of the living caniforms on Earth, then it is quite reasonable to fix the parameter $\rho$ to a known value.

```
rho <- 0.068
```

### 3.5 Root age

Get the value for the root from the **?** tree.

```
root_time <- treeHeight(T)
```

### 3.6 The time tree

Initialize the stochastic node representing the time tree.

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

Since we are computing the likelihood on the **?** tree, we can consider this time tree as an observation and clamp the stochastic node.

```
timetree.clamp(T)
```

Now set the workspace model variable.

```
mymodel = model(diversification)
```

```
mymodel.graph("output/m_BD_bears_GM.dot", bg="LightSteelBlue2")
```

### 3.7 Estimating the marginal likelihood of the model

- The `Rev` file for performing this analysis: `mlnl_BD_bears.Rev`.

```
pow_p = powerPosterior(mymodel, moves, "output/BD_bears_powp.out", cats=50)
pow_p.burnin(generations=10000,tuningInterval=1000)
pow_p.run(generations=1000)
```

Compute the marginal likelihood under stepping-stone sampling using the member function **`marginal()`** of the **`ss`** variable and record the value in Table 1.

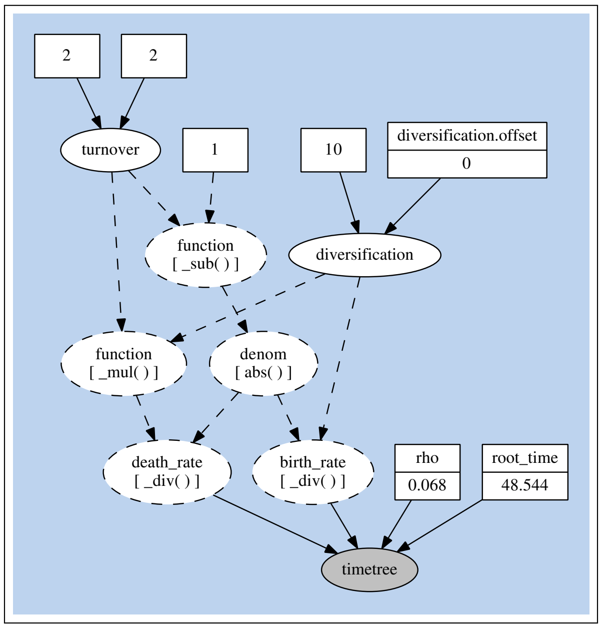

Figure 7: The graphical model representation of the birth-death process.

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

Compute the marginal likelihood under stepping-stone sampling using the member function **`marginal()`** of the **`ps`** variable and record the value in Table 1.

```
ps = pathSampler(file="output/BD_bears_powp.out", powerColumnName="power", likelihoodColumnName
    ="likelihood")
ps.marginal()
```

## 4 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 1 to summarize the marginal log-likelihoods estimated using the stepping-stone and path-sampling methods.

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 $\mathcal{K}$):

$$\mathcal{K} = \ln[BF(M_0, M_1)] = \ln[\mathbb{P}(\mathbf{X} \mid M_0)] - \ln[\mathbb{P}(\mathbf{X} \mid M_1)], \tag{3}$$

where $\ln[\mathbb{P}(\mathbf{X} \mid M_0)]$ is the *marginal lnL* estimate for model $M_0$. The value resulting from equation 3 can be converted to a raw Bayes factor by simply taking the exponent of $\mathcal{K}$

$$BF(M_0, M_1) = e^{\mathcal{K}}. \tag{4}$$

Alternatively, you can interpret the strength of evidence in favor of $M_0$ using the $\mathcal{K}$ and skip equation 4. In this case, we evaluate the $\mathcal{K}$ in favor of model $M_0$ against model $M_1$ so that:

$$\begin{aligned}&\text{if } \mathcal{K} > 1, \text{ then model } M_0 \text{ wins}\\&\text{if } \mathcal{K} < -1, \text{ then model } M_1 \text{ wins.}\end{aligned}$$

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

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

Table 1: Marginal likelihoods and Bayes factors*.

| **Estimate** | *Stepping-stone* | *Path sampling* |
|---|---|---|
| 2.1 Marginal likelihood Yule ($M_0$) | | |
| 3 Marginal likelihood birth-death ($M_1$) | | |
| Eq. 3: $BF(M_0, M_1)$ | | |
| Supported model? | | |

*you can edit this table

Do these data support a model without extinction ($\mu = 0$)?

## 5 Estimate Speciation and Extinction Rates

After comparing the marginal likelihoods using Bayes factors, you will discover which model is best supported by the data. With this model, we can now estimate posterior probability of the the global rate of speciation (and extinction if the birth-death model is used) given our observed tree.

$$\mathbb{P}(\lambda, \mu \mid \Psi) = \frac{\mathbb{P}(\Psi \mid \lambda, \mu)\mathbb{P}(d \mid \nu)\mathbb{P}(r \mid \alpha, \beta)}{\mathbb{P}(\Psi)} \tag{5}$$

- The `Rev` file for performing this analysis: `mcmc_BD_bears.Rev` or `mcmc_Yule_bears.Rev`.

### 5.1 Clear the workspace and load the preferred model

It is best to remove all of the previous model variables created in the previous section.

```
clear()
```

Now read in the observed tree from Figure 1.

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

Initialize the useful variables:

```
n_taxa <- T.ntips()
names <- T.names()
mi = 1
```

Source the model file of your favorite model ($*$ = **`Yule`** or **`BD`**).

```
source("RevBayes_scripts/m_*_bears.Rev")
```

```
mymodel = model(birth_rate)
```

### 5.2 Set up parameter monitors

```
monitors[1] = mnFile(filename="output/BDR_mcmc_bears.log",printgen=10, diversification
    , birth_rate, turnover, death_rate)
monitors[2] = mnScreen(printgen=1000, birth_rate)
```

Note that if your preferred model was the Yule process, then you would only have the **`birth_rate`** parameter listed in the **`mnFile`** monitor.

### 5.3 Run MCMC

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

```
mymcmc.burnin(generations=10000,tuningInterval=1000)
mymcmc.run(generations=50000)
```

```
mymcmc.operatorSummary()
```

- Visualize the MCMC samples of the birth rate and death rate parameters in Tracer.

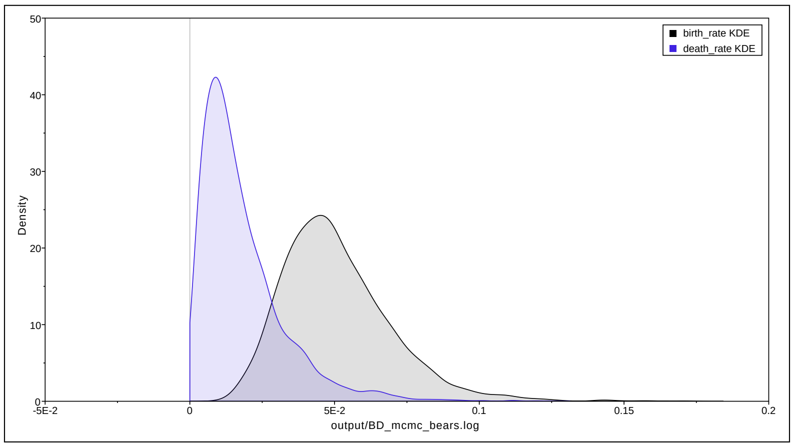

Figure 8: The marginal densities of **`birth_rate`** and **`death_rate`** estimated under the birth-death model in `RevBayes`.

## Useful Links

- RevBayes documentation and project information: RevBayes.com
- RevBayes source: https://github.com/revbayes/revbayes
- 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: trayc7@gmail.com)
- Tanja Stadler (email: tanja.stadler@bsse.ethz.ch)
- Sebastian Höhna (email: sebastian.hoehna@gmail.com)

## References

Version dated: January 30, 2015