Lab 8. Estimating time trees

Due Wednesday, November 30, 2022 at 3 pm

So far, we have focused on estimating *cladograms* (with meaningless branch lengths) or *phylograms* (with branch lengths in units of change per character). Today, we are going to infer *chronograms*, or time trees: phylogenetic trees with branch lengths in units of calendar time. We are going to build on what we've learned in the last two labs, and do it using RevBayes: it's going to be much easier now that you're familiar with the program!

Theory

In Lab 4, we compared the fit of different cladograms to the stratigraphic record, and suggested that we could use this criterion to decide between different most parsimonious trees (which, by definition, are all equally "good" based on parsimony alone). In doing so, we briefly mentioned the fact that there are more sophisticated methods that can leverage stratigraphic age information not just to decide between competing topologies, but to infer them in the first place. It is these methods that we are going to explore today.

By now, we are quite used to phylograms, whose branch lengths confound *rate* and *time*. In a phylogram, the length of a branch expresses the amount of character change that has been accumulated along it, but there is no way to determine whether this amount is due to slow change happening over a long period of time, or rapid change happening over a short period of time. In statistical parlance, we would say that these quantities are *not identifiable* from character data alone. This is a pretty intuitive idea: if I tell you I live two miles from my workplace, and ask you how long you think it takes me to get there, you'll likely find that information insufficient. That's because here, too, distance is the product of rate and time. On the other hand, if I give you an additional piece of data – e.g., that I drive to work during rush hour – you'll be able to come up with a decent estimate, since you can now disentangle the two factors. It works just the same with phylogenies:

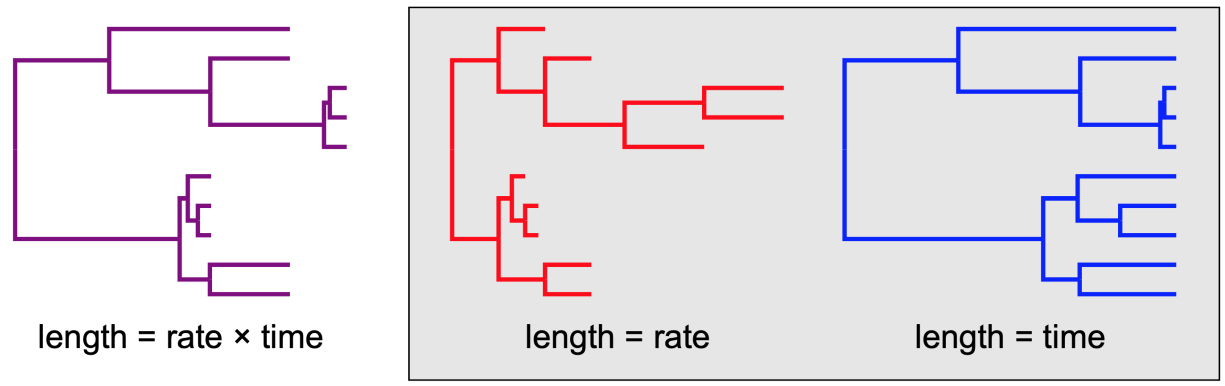


Image credit: Tracy A. Heath, Iowa State University

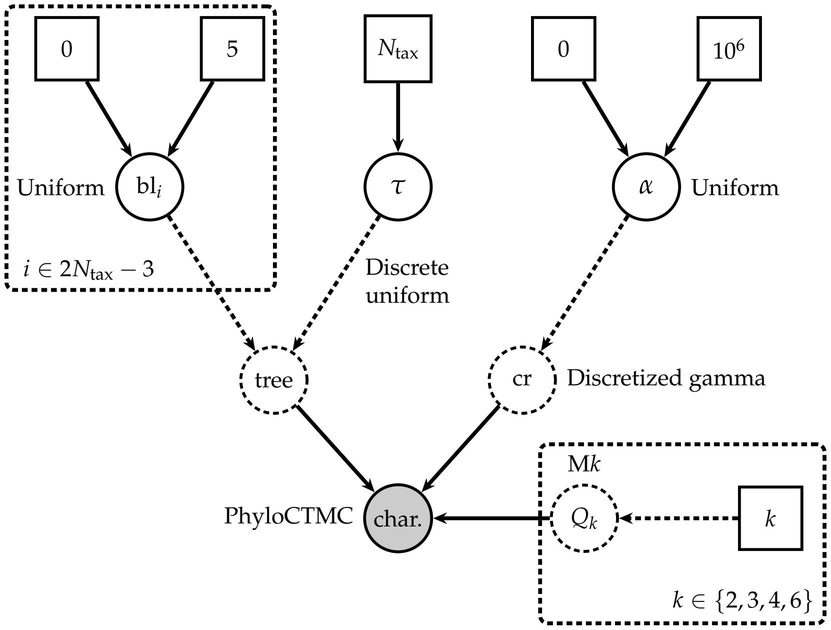
The earliest attempt to tease apart the relative contributions of rate and time to the total amount of character change is known as the *strict clock* hypothesis, and dates back to the early 1960s, when the biochemists Émile Zuckerkandl and Linus Pauling noticed that the number of changes between the hemoglobin proteins of different species of mammals appeared to be proportional to the time elapsed since their last common ancestor, according to paleontological estimates. If that were the case, we would just have to determine the exact constant of proportionality, or the slope of the line when the amount of change is plotted against age. This is exactly what many researchers attempted to do: they would take a fossil they believed to be close in time to a particular divergence, and use it to "calibrate the clock" in order to convert branch lengths from units of substitutions per character into units of time. If, for example, two taxa differed in 5 out of 100 characters, and fossil evidence suggested their evolutionary lineages diverged 20 million years ago, we would conclude that the rate of evolution equals $(5/100)/(2 \times 20) = 0.00125$ changes per character per million years. Once we've obtained this proportionality constant, converting the remaining branches into units of time is a matter of simple cross-multiplication.

It would be nice if things were this simple, wouldn't it? No such luck, unfortunately. It became clear quite early on that certain lineages evolve much faster than others, which really throws a wrench into the strict clock model. As a result, biologists began to search for ways in which the assumption of a single rate governing the entire tree could be relaxed. We could, for example, choose to assume that different parts of the tree are governed by different clocks, in which case we would just need to estimate the number and location of points at which there is a switch from one clock to another. This is known as the *local clock* model. However, a more popular alternative to the strict clock is represented by the so-called *relaxed clock* models, where each branch can have its own distinct rate. Here, we only make a much looser assumption that these rates are drawn from some common distribution (analogous to how we drew branch lengths from a common distribution in Lab 7). Moreover, the exact shape of this distribution can itself be estimated from the data, just like we estimated the shape of the gamma distribution of rates across characters.

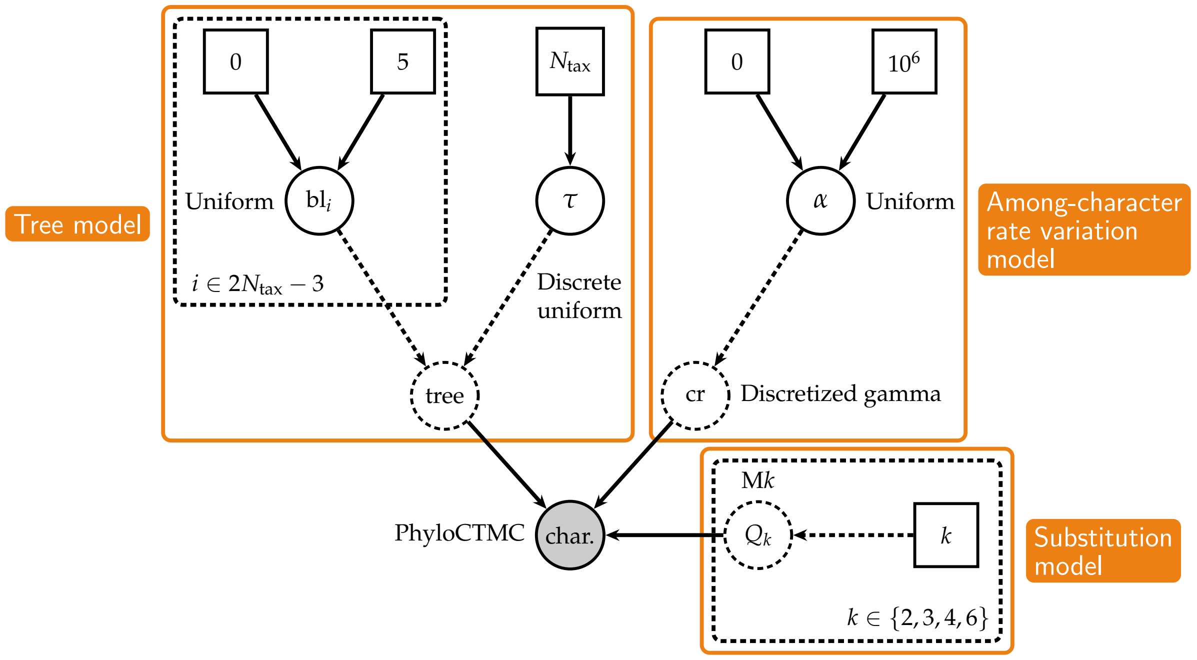
By making these assumptions, and by adding in extra data in the form of taxon ages, we can get around the non-identifiability problem and disentangle rate and time. However, the resulting models are very complex – so much so, in fact, that it's very hard to develop good optimization algorithms for estimating their parameters in a maximum-likelihood framework. Fortunately, we can still rely on the faithful workhorse that is the MCMC. For this reason, relaxed clock analyses are exclusively carried out using Bayesian inference.

Models and submodels

We spent two weeks drinking from the firehose when it came to learning the ins and outs of RevBayes, and now we finally get to reap the benefits. First of all, instead of having to introduce yet another piece of software, we can just stick with what we know. Even better, we can recycle a good chunk of the model we so laboriously put together in Lab 7. This is because phylogenetic models are modular – they consist of smaller components, or "submodels", that can be freely replaced while the rest remains intact. Let's take a detailed look at the model from Lab 7 one more time to demonstrate this:



We see that the model itself has a tree-like shape: there are three separate branches that don't really communicate with each other, except that all of them ultimately enter the PhyloCTMC distribution from which we are drawing our character data. These are our submodels: let's highlight them and label them to make it clearer:



You may have also noticed this modular nature of our model in the corresponding Rev script (`Tedford_phylo.Rev`), which used subheadings for the individual submodels:

```
################################
# Topology and branch lengths #
################################

⋮

#####################################
# Rate variation across characters #
#####################################

⋮

#########################################################
# Mk substitution models, partitioned by number of states #
#########################################################
```

The whole trick behind switching from inferring phylograms to inferring chronograms consists in (1) replacing the tree submodel and (2) adding a new submodel corresponding to our relaxed clock. We can keep everything else as-is, including the substitution and among-character rate variation models, as well as the MCMC machinery under the hood.

The Fossilized Birth-Death model

What follows has been adapted from a BEAST 2 tutorial developed by Prof. Tracy Heath's lab at the Iowa State University. You can access the full version from `https://taming-the-beast.org/tutorials/FBD-tutorial/FBD-tutorial.pdf`. BEAST 2 is a program similar to RevBayes, but unlike the latter, it can only infer chronograms, not phylograms.

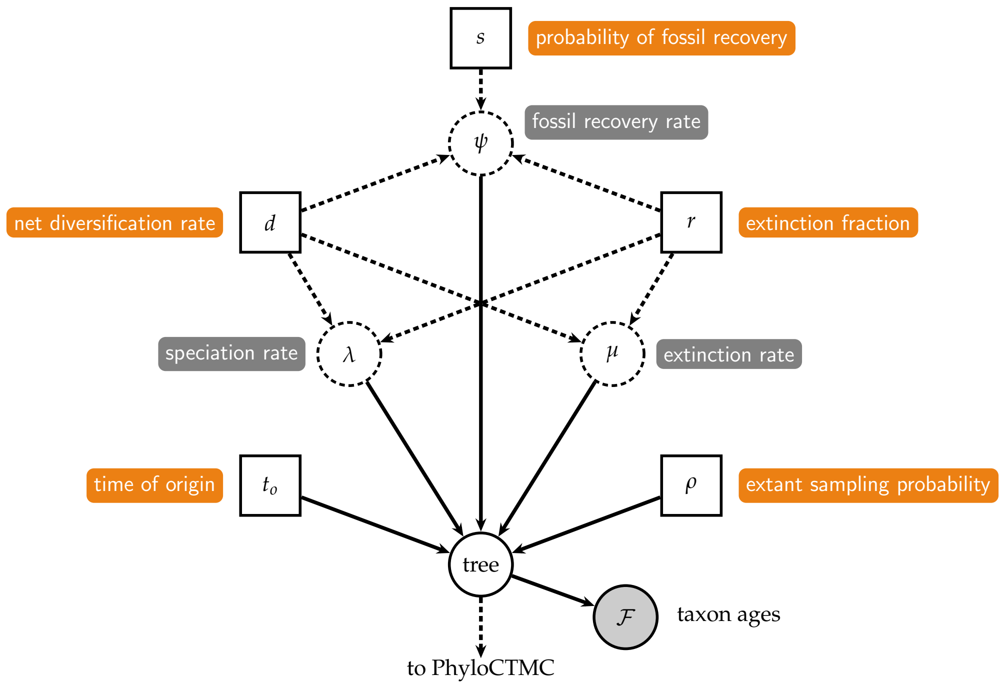
The new tree model we are going to be using is called the Fossilized Birth-Death (FBD) model. It was described by Tracy Heath and colleagues in 2014, building upon earlier work by Tanja Stadler. As its name suggests, the FBD model considers a phylogenetic tree to be the product of three processes: the "birth" of new lineages, or speciation; the "death" of lineages, or extinction; and the fossilization process, by which we come to learn about lineages that did not survive to the present. It further assumes that the waiting times between the successive occurrences of the same kind of event (e.g., between two speciations, two extinctions, etc.) are exponentially distributed, making short waiting times more likely than long waiting times. In total, the model has five parameters:

Parameter	Description
λ	Speciation rate
μ	Extinction rate
ψ	Fossil recovery rate
ρ	Probability of sampling extant species
t_o	Time of origin

Instead of dealing with the three rates λ, μ, ψ directly, however, it is often more convenient to *reparameterize* the FBD model. The concept of reparameterization is analogous to the idea of different coordinate systems. When it comes to locating a point on a two-dimensional plane, most of us would probably use two orthogonal axes, x and y. However, we could also employ, say, the polar coordinate system, which relies on radial distance r from a reference point and angle ϕ relative to a reference direction. Both descriptions are valid, both have the same expressive power, and both are equally complex in that they require two coordinates: (x, y) or (r, ϕ). The same is true of reparameterization. The reason why we do this with the FBD model is that two of the three new parameters range from 0 to 1, which makes it easier to place uninformative priors on them:

Reparameterization	Description
$d = \lambda - \mu$	Net diversification rate
$r = \frac{\mu}{\lambda}$	Extinction fraction
$s = \frac{\psi}{\mu+\psi}$	Probability a species leaves behind a fossil before going extinct

The remaining two parameters stay the same, yielding the following model:



And that's before we even placed priors on the free parameters – you can tell that things are starting to get a bit complicated!

Before we begin translating this model into Rev, we should notice that a new clamped variable appeared in the graph. Previously, only the characters (drawn from PhyloCTMC) were clamped. That makes sense in light of what we said above: estimating chronograms requires additional data in the form of taxon ages. However, this raises a question of how we can load such data into RevBayes. It turns out there are two things RevBayes absolutely insists on:

1. Your data must consist of three columns, called (in this order): `taxon`, `min_age`, and `max_age`;
2. The entries in your `taxon` column must perfectly match the taxon names in your Nexus file.

I like to keep my age data in a tab-separated file. When I open the file I assembled for the Tedford et al. dataset in a text editor (e.g., TextEdit on macOS, Notepad on Windows, Gedit on Linux), its first few lines look as follows:

```
taxon   min_age max_age
Hesperocyoninae 15.0    40.0
Borophaginae    2.0     34.0
Leptocyon_mollis        29.0    30.0
Leptocyon_douglassi     25.7    30.6
```

For your convenience, I uploaded an example file with the Tedford et al. age data to Canvas (`Tedford_ages.tsv`).

This is how to load a file like this into RevBayes:

```
taxa <- readTaxonData("/Users/David/Tedford_ages.tsv")
```

Next, let's place priors on the three rate parameters of the FBD model. The extinction fraction and the fossil recovery probability range from 0 to 1, so in the absence of any meaningful prior information, the uniform distribution $U(0, 1)$ is the natural choice. The net diversification rate, on the other hand, ranges from 0 to infinity. We'll just use an exponential prior distribution with a rate of 10, corresponding to a mean of $1/10 = 0.1$ species per million years. (Remember, this is the rate of species build-up, i.e., excess of speciation over extinction.)

```
diversification ~ dnExponential(10)
ext_frac ~ dnUniform(0, 1)
fossil_prob ~ dnUniform(0, 1)
```

One trick that can improve parameter space exploration for very complex models is to use multiple moves of the same kind, but with different tuning values. That way, some will be bolder and favor long leaps, while others will be more conservative and keep close to the current value:

```
# Specify a scale move on the diversification parameter
moves.append( mvScale(diversification, lambda=0.01, weight=5) )
moves.append( mvScale(diversification, lambda=0.1,  weight=3) )
moves.append( mvScale(diversification, lambda=1,    weight=1) )

# Specify a sliding-window move on the ext_frac parameter
moves.append( mvSlide(ext_frac, delta=0.01, weight=5) )
moves.append( mvSlide(ext_frac, delta=0.1,  weight=3) )
moves.append( mvSlide(ext_frac, delta=1,    weight=1) )

# Specify a sliding-window move on the fossil_prob parameter
moves.append( mvSlide(fossil_prob, delta=0.01, weight=5) )
moves.append( mvSlide(fossil_prob, delta=0.1,  weight=3) )
moves.append( mvSlide(fossil_prob, delta=1,    weight=1) )
```

Next, reparameterize. If it's not clear to you what's happening below, go back to Table 2 and think about how you could recover λ, μ, ψ from d, r, s:

```
speciation := diversification / abs(1.0 - ext_frac)
extinction := (ext_frac * diversification) / abs(1.0 - ext_frac)
recovery := (fossil_prob / abs(1.0 - fossil_prob)) * extinction
```

An interesting feature of the FBD model is that it is not identifiable when all of its parameters are left free, just like rate and time were not identifiable when we were trying to estimate them from character data alone. To avoid this problem, we can pick one parameter and set it to a fixed value. Fortunately, there is a natural choice for the parameter to be fixed: it would be the extant sampling probability ρ. As long as we have a good idea of how many extant species there are in our group of interest, we can treat this parameter as being known without error. For example, Wikipedia says there are 34 extant species of canids. The Tedford et al. (2009) matrix includes 7 of them, so we can do the following:

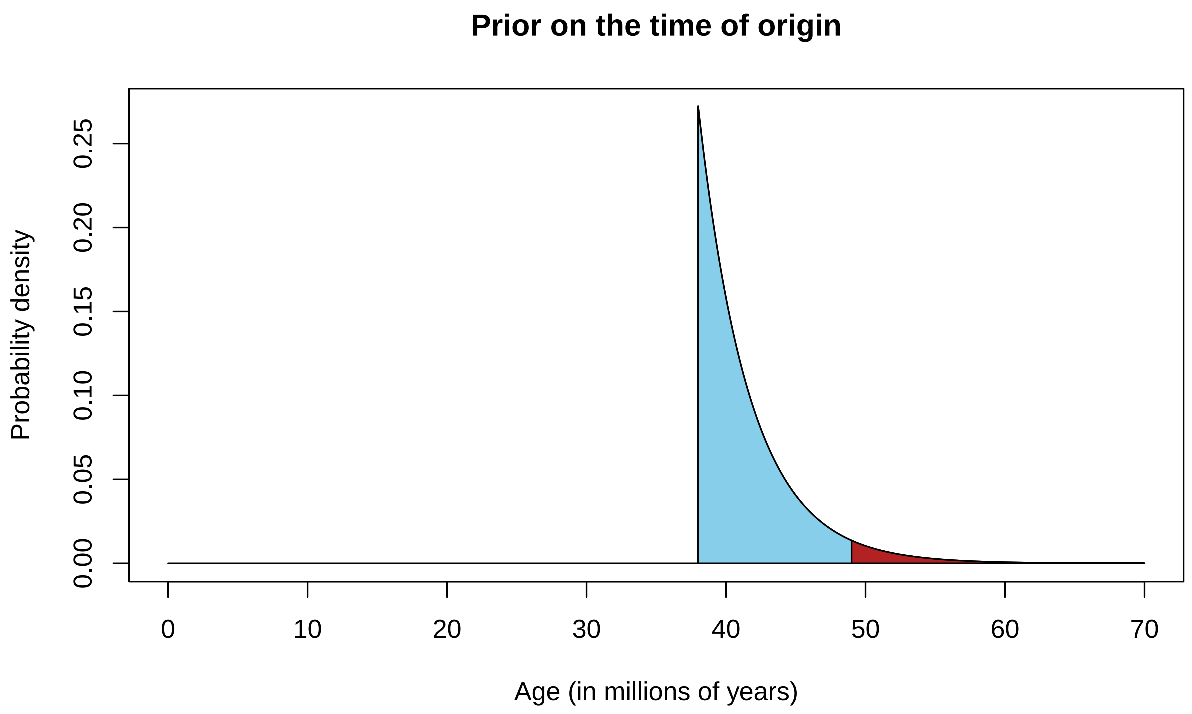
```
rho <- 7/34
```

The only remaining parameter we have to place a prior on is the time of origin of our time tree. Here, it actually bears reflecting on our prior beliefs in a little more detail. We know that the origin of a clade is likely to somewhat predate its first appearance in the fossil record. We can then think of the problem in terms of the expected waiting time between these two events (i.e., origin vs. first appearance). If the fossil record of our group is very good, the expected waiting time will be quite short. If, on the other hand, our fossil record is sparse, the waiting time could be pretty long. The canid fossil record is very good in comparison to most other vertebrates, so we would expect the former. But how short exactly? Fortunately, this happens to be one of the cases where we do, in fact, have some prior information we can use. According to a 2012 study by Mario dos Reis and colleagues, the family Canidae diverged from its sister group approximately 49 million years ago. The earliest canid fossil, *Hesperocyon*, is about 38 million years old. We might therefore want to place, say, 95% probability on origin times that lie between these two values, followed by a long, low-probability tail for ages older than 49 million years.

A good choice would be an *offset exponential* distribution, which looks and behaves just like a regular exponential distribution, except that its support is $[L, \infty)$ rather than $[0, \infty)$, with L being some arbitrary positive number. Here, we set it to 38, because the canid clade cannot be younger than its oldest known fossil. All that's left to do is determine what the mean of our exponential has to be in order for its 95th percentile to equal 49. We can do this using the quantile function `qexp()`, which finds a time T such that the probability $P(t_o \leq T)$ equals a specified value (here, 0.95):

```
mean_prior_age <- (49 - 38)/qexp(0.95)
# The lambda parameter of an exponential function is the inverse of the mean:
origin_time ~ dnExponential(lambda=1/mean_prior_age, offset=38)
```

This is what our prior looks like:



Note that 95% of the total probability mass is contained in the region highlighted in blue, while the red tail extends all the way to positive infinity.

Let's not forget that the time of origin, too, is a parameter we want to estimate, so we have to place a move on it:

```
# Specify a sliding-window move on the origin_time parameter
moves.append( mvSlide(origin_time, delta=1, weight=10) )
```

Now we just put everything together using the `dnFBDP()` function. (You could also use its longer but more descriptive alias, `dnFossilizedBirthDeath()`.)

```
tree ~ dnFBDP(origin=origin_time, lambda=speciation, mu=extinction,
              psi=recovery, rho=rho, taxa=taxa)
```

We have to use a different set of moves now that we are inferring a time tree (chronogram) rather than a branch-length tree (phylogram):

```
# Specify moves on topology; allow some taxa to be ancestors of others
moves.append( mvFNPR(tree, weight=15) )
moves.append( mvCollapseExpandFossilBranch(tree, origin_time, weight=6) )

# Specify moves on the node ages, including the root age
moves.append( mvNodeTimeSlideUniform(tree, weight=40) )
moves.append( mvRootTimeSlideUniform(tree, origin_time, weight=5) )
```

This highlights another cool feature of the FBD model. So far, we have restricted ourselves to inferring sister-group relationships, but we had nothing to say about ancestor-descendant relationships. In a PAUP* cladogram or even a RevBayes phylogram, you'll never see one taxon being directly ancestral to another, even if that was in fact their true relationship. The best we could have done in such a situation was to place the ancestor in a sister-group relationship with the clade formed by its descendants, and make its branch very, very short to indicate that it did not undergo any changes aside from those that were also shared by its descendants. With FBD time trees, however, there are no such limitations, and we can infer ancestor-descendant relationships to our heart's content. Let's ask RevBayes how many ancestors it found among the 39 canid taxa in our sample:

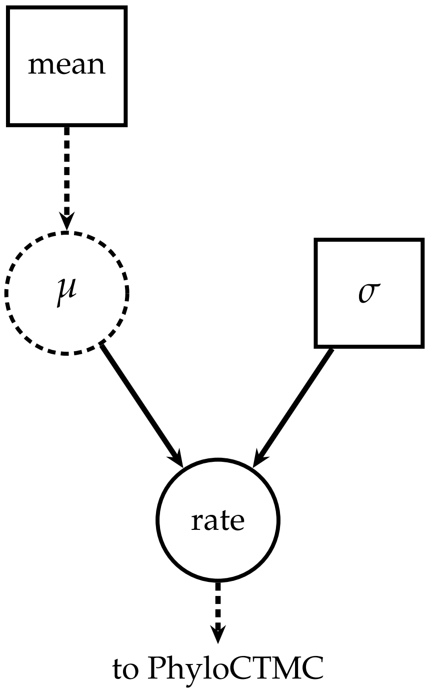
```
num_samp_anc := tree.numSampledAncestors()
```

Strict clock

Compared to the FBD model, setting up a strict clock in RevBayes is a breeze. We just need to specify a prior on our *clock rate*, that is, the "base" rate of character change, which gets further scaled up and down by our among-character rate variation model. We will use a lognormal distribution for this purpose. This is the distribution followed by a stochastic variable whose logarithm is normally distributed. The lognormal distribution has support on $[0, \infty)$ and is fully specified by two parameters, namely the log-scale mean μ and the log-scale standard deviation σ. We back-derive the log-scale mean from the actual mean, which we simply set equal to a value that seems reasonable to us: 0.005 changes per character per million years. Our choice of σ probably seems completely arbitrary, but there is a reason behind it: this way, 95% prior probability is going to span exactly two orders of magnitude.

```
sigma <- 1.17481
clock_mean <- 0.005
mu := ln(clock_mean) - (0.5 * sigma^2)
clock_rate ~ dnLognormal(mean=mu, sd=sigma)

# Specify a scale move on the clock rate parameter
moves.append( mvScale(clock_rate, lambda=1, weight=10) )
```

This is what our strict clock model looks like:



Our specification of PhyloCTMC also has to change to account for the fact that we are now adding branch rates as a new submodel of our overall model:

```
characters[j] ~ dnPhyloCTMC(tree=tree, Q=Q[j], type="Standard",
                            branchRates=clock_rate, siteRates=char_rates,
                            coding="variable")
```

RevBayes is smart enough to automatically apply the `clock_rate` value to every single branch in our tree. Interestingly, here we encounter another subtle difference between cladograms or phylograms on the one hand, and chronograms on the other. Our cladograms and phylograms were *unrooted*, and as a result, had $2N - 3$ branches for every N taxa. However, the FBD model (or indeed, any other tree model that allows us to infer time trees) induces a prior probability distribution on *rooted* trees, which have an extra root branch, and thus $2N - 2$ branches in total.

For your convenience, all of the above code has been packaged into a single Rev script available from Canvas (`Tedford_FBD_strictclock.Rev`).

---

1) Run a strict-clock time tree analysis on your pasta dataset. If you want, you can use the `Tedford_FBD_strictclock.Rev` script as a template. Think about which parts of the code are generally applicable, and which you'll need to tailor to your own data. You might want to increase the number of MCMC generations compared to the 20,000 we used in Lab 7. Why do you think that is?

2) Did your parameters reach stationarity? Which, if any, don't look good?

3) We specified priors on the clock rate, net diversification rate, extinction fraction, and extant sampling probability. What do their posteriors look like? (Tip: in Tracer, click on the "Estimates" tab to get their mean and 95% credibility interval.) How do the posteriors compare to the priors we used?

4) Generate the MCC and/or MAP summary tree for your analysis, visualize it, and include the resulting image in your answer sheet. How does it compare to your trees from the previous weeks? Briefly describe its topology, divergence times, and posterior probabilities.

5) Does your summary tree contain any ancestor-descendant relationships? If so, what are they? What is the posterior mean for the number of sampled ancestors in your tree?

Uncorrelated lognormal relaxed clock

Trying our hand at time tree estimation using the strict clock was fun, but as we've learned above, relaxed clock models is where it's really at. Let's set up an *uncorrelated lognormal* (UCLN) model, which is one of the most commonly used relaxed clocks. The word "lognormal" indicates that the rates of character change for each branch will be drawn from a lognormal distribution. The word "uncorrelated" means that each branch draws its rate independently from the same distribution. There are also *autocorrelated* models, which assume that the rate of any given branch tends to stay close to the rate of its parent branch, and that it takes time for large rate differences to evolve along the tree. In uncorrelated models, this isn't necessarily true: very slow and very fast branches can directly adjoin each other, because they all draw their rates independently. There is an ongoing debate about the relative merits of uncorrelated and autocorrelated clock models.

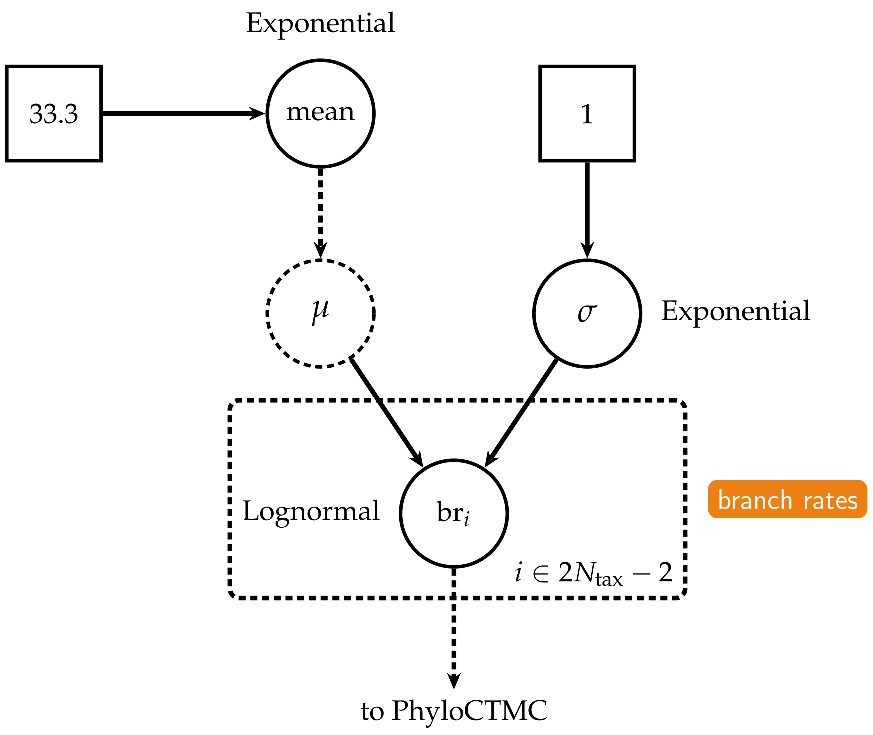
We already know how to parameterize a lognormal distribution, so we just need to come up with appropriate priors on μ and σ:

```
# The exponential prior below ensures that the mean of the prior on the
# mean clock rate will be equal to 0.03, six times greater than before.
ucln_mean ~ dnExponential(33.3)

# The log-scale standard deviation of our branch rate distribution will
# also be exponentially distributed:
ucln_sigma ~ dnExponential(1.0)

ucln_mu := ln(ucln_mean) - (0.5 * ucln_sigma^2)
moves.append( mvScale(ucln_mean, lambda=1.0, weight=4))
moves.append( mvScale(ucln_sigma, lambda=0.5, weight=4))
```

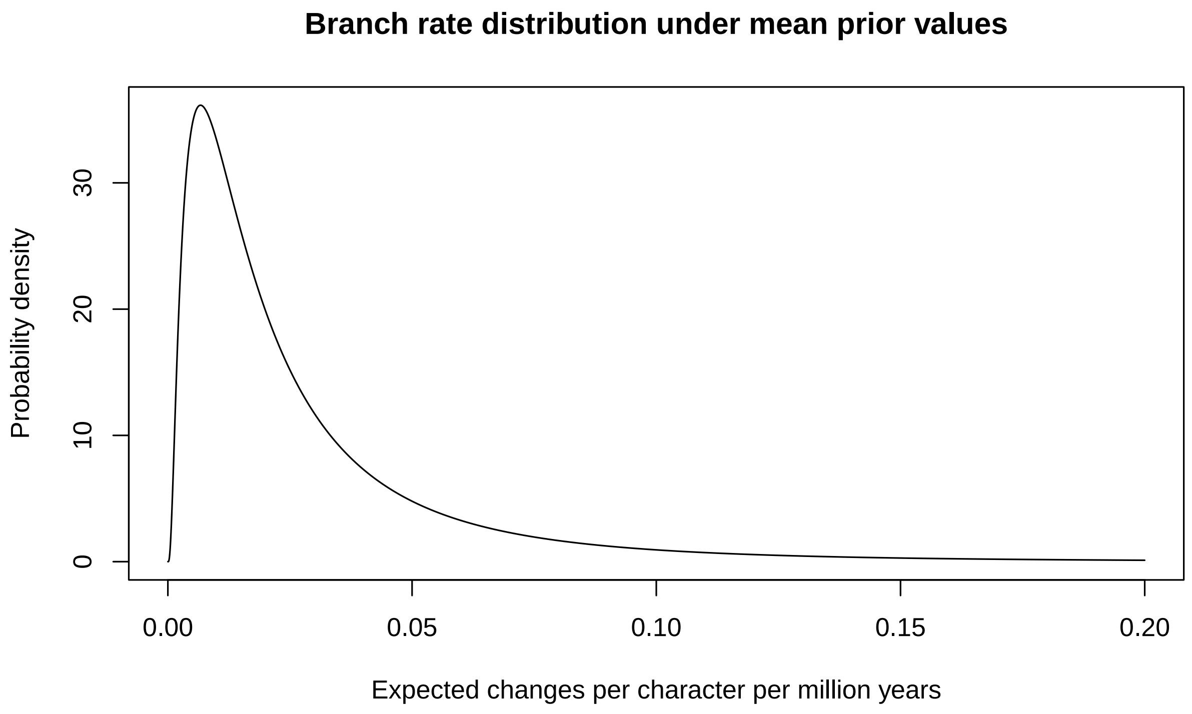
We also know that each branch rate is going to represent an independent draw from the distribution we have just defined. A `for` loop is an appropriate construct for this:

```
for (i in 1:num_branches) {
   branch_rates[i] ~ dnLognormal(ucln_mu, ucln_sigma)
   moves.append( mvScale(branch_rates[i], lambda=1, weight=2) )
}
```

This is what our UCLN model looks like:



And this is what our lognormal distribution of branch rates would look like if its mean and standard deviation were both equal to the expected values of their respective priors:



Note, however, that the actual distribution may be very different, because both parameters are in fact estimated from the data, which will presumably cause them to deviate from the prior.

Finally, we just need to change our call to the `dnPhyloCTMC()` function one more time:

```
characters[j] ~ dnPhyloCTMC(tree=tree, Q=Q[j], type="Standard",
                            siteRates=char_rates, coding="variable"
                            branchRates=branch_rates)
```

---

6) Repeat your analysis using the uncorrelated lognormal relaxed clock instead of the strict clock. Did the change have any impact on the topology of your tree or on your divergence time estimates?

7) Describe the posterior distributions you obtained for `ucln_mean` and `ucln_sigma`. What does this tell you about the branch rates for your pasta dataset? Is there a lot of variation, or are they tightly concentrated about a particular value? Based on your answer, do you think it was important to switch from the strict clock to a relaxed clock for your dataset?

You may have noticed there is one ingredient that's missing from today's analyses, compared to the one we ran last week: nowhere in our script did we specify an outgroup. This is because we are working with rooted trees, and estimating the position of the root is part of the inference process. That's a bit of a mixed blessing: on the one hand, we can actually *test* whether our outgroup forms the sister group to all other taxa in our tree, instead of just assuming it. On the other hand, our datasets are so small that they might not contain enough information to reliably infer the position of the root all on their own.

---

8) Was your analysis able to identify Orzo as the sister group of all other pastas on its own? If not, where exactly did Orzo end up in your tree?

We've established that we don't *need* an outgroup when estimating time trees – but *could* we still specify one if we really wanted to? Of course! We can use *topological constraints*, which tell RevBayes to only explore those trees that satisfy certain conditions. In particular, clade constraints require that specified taxa always form a monophyletic group to the exclusion of all the other taxa. For example, this is how we could enforce a sister-group relationship between the gray wolf (*Canis lupus*) and the dire wolf (*Canis dirus*):

```
tree_distribution = dnFBDP(origin=origin_time, lambda=speciation, rho=rho,
                           mu=extinction, psi=recovery, taxa=taxa)
clade_constraints[1] = clade("Canis_dirus", "Canis_lupus")
tree ~ dnConstrainedTopology(treeDistribution=tree_distribution,
                             constraints=clade_constraints)
```

What we really want is to enforce a clade that contains all taxa other than Hesperocyoninae, our former outgroup. While we could list all the 38 ingroup taxa by their names, one by one, just like we did above, that would be a pretty laborious and error-prone process. Instead, we can do the following:

```
# Create a temporary copy of the "taxa" vector
my_taxa <- taxa

# Remove Hesperocyoninae from the copy
my_taxa.erase(taxon("Hesperocyoninae"))

# Set up the corresponding clade constraint
my_constraints[1] = clade(my_taxa)
tree ~ dnConstrainedTopology(treeDistribution=tree_distribution,
                             constraints=my_constraints)
```

---

9) Re-run your analysis under a topological constraint that enforces the monophyly of all non-Orzo pastas. Did the results change in any way?

However, maybe it's not such a good idea to include outgroups in our analysis in the first place, regardless of whether they are treated as such. Above, we've seen that the parameters of the FBD model include the fossil recovery rate (ψ) and extant sampling probability (ρ). Unfortunately, these are pretty sensitive to the fact that when researchers assemble their character matrices, they never sample the outgroup as densely as the ingroup. For example, if we use a single derived representative of a diverse clade as our outgroup, this outgroup will end up sitting at the end of a branch of very long duration. This implies a long period during which the outgroup lineage failed to leave behind any fossils, which will in turn lead the analysis to underestimate the fossil recovery rate. Only by sampling the outgroup clade more densely could we break up that long branch and get a more accurate estimate of ψ. Similarly, the inclusion of an outgroup makes it hard to meaningfully specify ρ. If my analysis includes 90% of the extant species belonging to the ingroup clade, but only 0.1% of the extant species belonging to the outgroup clade, what value should I use? Often, these problems are most easily avoided by removing the outgroup altogether.

---

10) Re-run your analysis without Orzo. Did this change the relationships among the remaining taxa? Are the posterior estimates of your `fossil_prob` parameter any different now?
