Phylogenetic Inference using `RevBayes`

Total-Evidence Analysis of Fossil and Extant Taxa under the Fossilized Birth-Death Process

Tracy A. Heath, April Wright, and Walker Pett

1 Overview

This tutorial demonstrates how to specify the models used in a Bayesian "total-evidence" phylogenetic analysis of extant and fossil species, combining morphological and molecular data as well as stratigraphic range data from the fossil record (*e.g.,* Ronquist et al. 2012; Zhang et al. 2016; Gavryushkina et al. 2016). We begin by outlining the models used in this analysis in section 2, followed by a detailed exercise in section 3 demonstrating how to apply these models in `RevBayes` (Höhna et al. 2017) and use Markov chain Monte Carlo (MCMC) to estimate the posterior distribution of dated phylogenies for data collected from living and fossil bears (family Ursidae).

1.1 Requirements

1.1.1 Required Software

This tutorial requires that you download and install the latest release of `RevBayes` (Höhna et al. 2017), which is available for Mac OS X, Windows, and Linux operating systems. Directions for downloading and installing the software are available on the program webpage: http://revbayes.com.

The exercise provided also requires additional programs for editing text files and visualizing output. The following are very useful tools for working with `RevBayes`:

- A good text editor – if you do not already have one that you like, we recommend one that has features for syntax coloring, easy navigation between different files, line numbers, etc. A good option is Sublime Text, which is available for Mac OSX, Windows, and Linux.
- Tracer – for visualizing and assessing numerical parameter samples from `RevBayes`
- IcyTree – a web-hosted phylogenetic tree visualization tool that is supported for Firefox or Google Chrome browsers
- FigTree – a tree visualization prgram

1.1.2 Prerequisite Tutorials

In addition to installing the software, this tutorial assumes that you have read and completed the following tutorials:

- Getting started
- `Rev` basics

2 Introduction

The "total-evidence" analysis described in this tutorial uses a graphical model (Höhna et al. 2014) integrating three separate likelihood components or data partitions (Fig. 1): one for molecular data (section 2.3), one for morphological data (section 2.4), and one for fossil stratigraphic range data (section 2.2). In addition, all likelihood components are conditioned on a tree topology with divergence times, which is modeled according to a separate prior component (section 2.1).

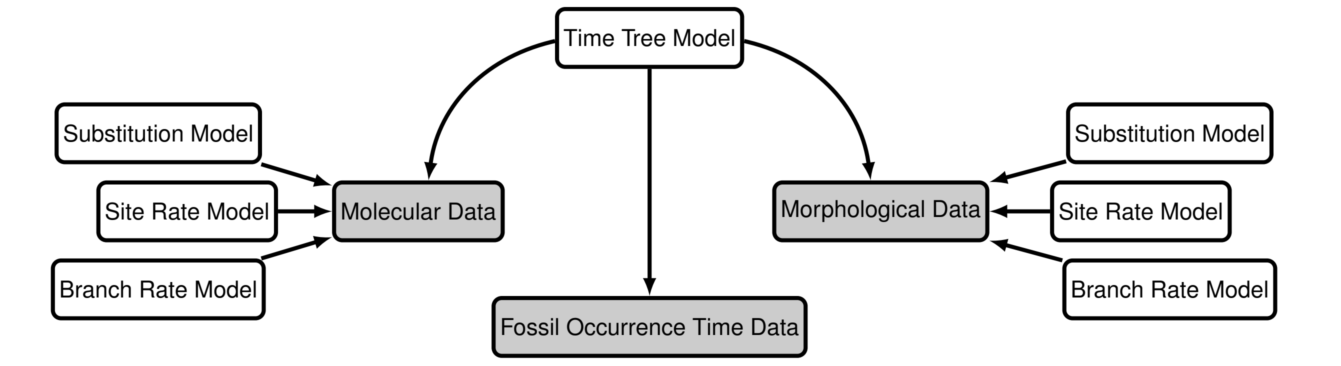


Figure 1: Modular components of the graphical model used in the "total-evidence" analysis described in this tutorial.

In figure 2 we provide an example of the type of tree estimated from a total-evidence analysis. This example shows the complete tree (Fig. 2A) and the sampled or reconstructed tree (Fig. 2B). Importantly, we are interested in estimating the topology, divergence times, and fossil sample times of the *reconstructed tree* (Fig. 2B). We will describe the distinction between these two trees in section 2.1.

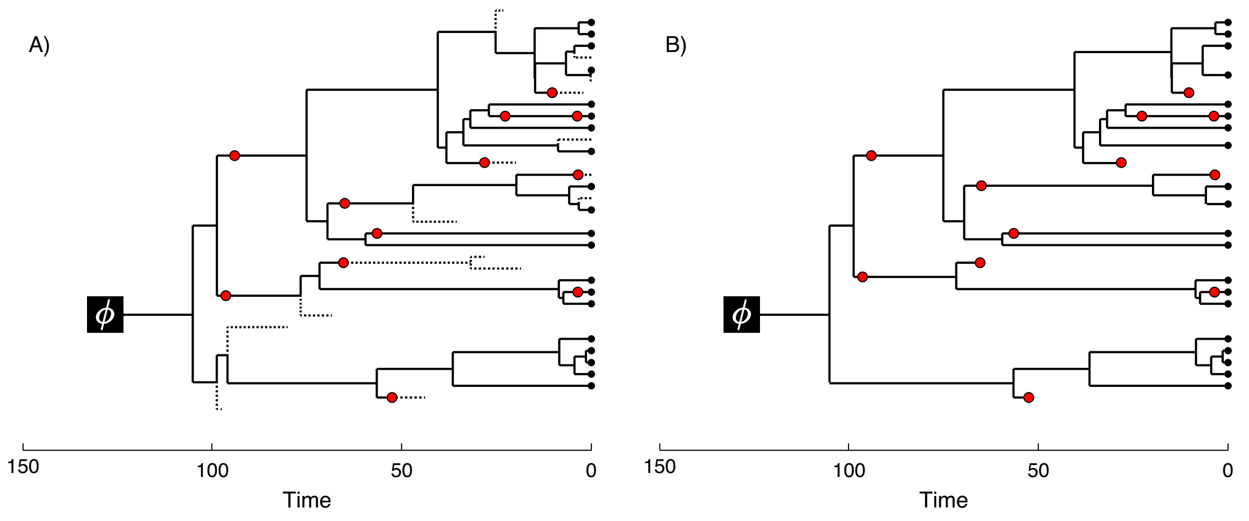


Figure 2: One possible realization of the fossilized birth-death process starting at origin time ϕ, showing fossil sampling events (red circles), and 15 sampled extant taxa (black circles). (A) The complete tree shows all lineages both sampled (solid lines) and unsampled (dotted lines). (B) The reconstructed tree (also called the sampled tree) shows only the sampled lineages.

2.1 Lineage Diversification and Sampling

The joint prior distribution on tree topologies and divergence times of living and extinct species used in this tutorial is described by the *fossilized birth-death* (FBD) process (Stadler 2010; Heath et al. 2014). This model simply treats the fossil observations as part of the process governing the tree topology and branch times (the *Time Tree Model* node in Fig. 1). The fossilized birth-death process provides a model for the distribution of speciation times, tree topology, and distribution of lineage samples before the present (*e.g.,* non-contemporaneous samples like fossils or viruses). This type of tree is shown in figure 2. Importantly, this model can be used *with or without* character data for the historical samples. Thus, it provides a reasonable prior distribution for analyses combining morphological or DNA data for both extant and fossil taxa—*i.e.,* the so-called "total-evidence" approaches described by Ronquist et al. (2012) and extended by Zhang et al. (2016) and Gavryushkina et al. (2016). When matrices of discrete morphological characters for both living and fossil species are unavailable, the fossilized birth-death model imposes a time structure on the tree by *marginalizing* over all possible attachment points for the fossils on the extant tree (Heath et al. 2014), therefore, some prior knowledge of phylogenetic relationships is important, much like for calibration-density approaches.

The FBD model (Fig. 3) describes the probability of the tree and fossils conditional on the birth-death parameters: $f[\mathcal{T} \mid \lambda, \mu, \rho, \psi, \phi]$, where $\mathcal{T}$ denotes the tree topology, divergence times, fossil occurrence times, and the times at which the fossils attach to the tree. The birth-death parameters λ and μ denote the speciation and extinction rates, respectively. The "fossilization rate" or "fossil recovery rate" is denoted ψ and describes the rate at which fossils are sampled along lineages of the complete tree. The sampling probability parameter ρ represents the *probability* that an extant species is sampled, and ϕ represents the time at which the process originated.

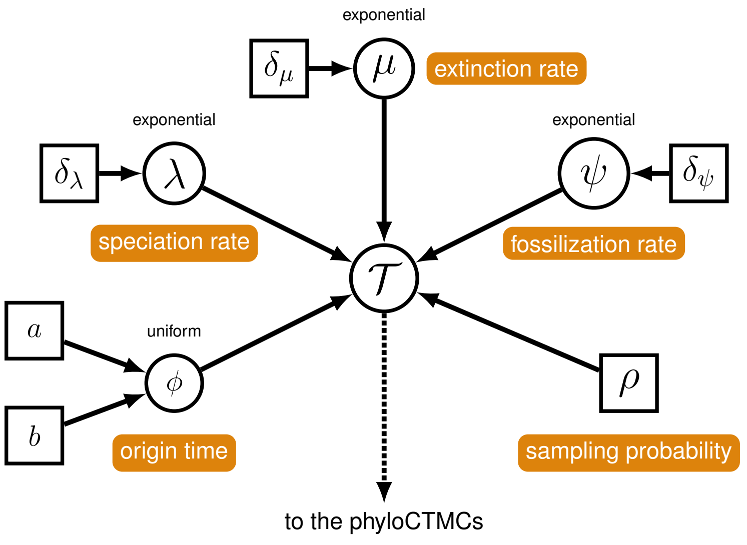


Figure 3: A graphical model of the fossilized birth-death model describing the generation of the time tree (***Time Tree Model*** in Fig. 1) used in this tutorial. The parameters of the fossilized birth-death process are labeled in orange. The speciation, extinction and fossilization rates are stochastic nodes (circles) drawn from exponential distributions, while the origin time is uniformly distributed. The sampling probability is constant node (square) and equal to one. For more information on probabilistic graphical models and their notation, please see Höhna et al. (2014).

In the example FBD tree shown in figure 2, the diversification process originates at time ϕ, giving rise to $n = 20$ species in the present, with both sampled fossils (red circles) and extant species (black circles).

All of the lineages represented in figure 2A (both solid and dotted lines) show the *complete tree*. This is the tree of all extant *and* extinct lineages generated by the process. The complete tree is distinct from the *reconstructed tree* (Fig. 2B) which is the tree representing only the sampled *extant* lineages. Fossil observations (red circles in figure 2) are recovered over the lifetime of the process along the lineages of the complete tree. If a lineage does not have any descendants sampled in the present, it is lost and cannot be observed, these are the dotted lines in figure 2A. The probability must be conditioned on the origin time of the process ϕ. The origin (ϕ) of a birth-death process is the starting time of the *stem* lineage, thus this conditions on a single lineage giving rise to the tree.

An important characteristic of the FBD model is that it accounts for the probability of sampled ancestor-descendant pairs (Foote 1996). Given that fossils are sampled from lineages in the diversification process, the probability of sampling fossils that are ancestors to taxa sampled at a later date is correlated with the turnover rate ($r = \mu/\lambda$) and the fossil recovery rate (ψ). This feature is important, particularly for datasets with many sampled fossils. In the example (Fig. 2), several of the fossils have sampled descendants. These fossils have solid black lines leading to the present.

2.2 Incorporating Fossil Stratigraphic Range Data

In order to account for uncertainty in the ages of our fossil species, we can incorporate stratigraphic range information from the fossil record. We do this by assuming each fossil can occur with uniform probability anywhere within its observed stratigraphic range. This is somewhat different from the typical approach to node calibration. Here, instead of treating the calibration density as an additional prior distribution on the tree, we treat it as the *likelihood* of our fossil data given the tree parameter. Specifically, we assume the likelihood of a particular fossil observation $\mathcal{F}_i$ is equal to one if the fossil's inferred age on the tree t_i falls within its observed stratigraphic range (a_i, b_i), and zero otherwise:

$$f[\mathcal{F}_i \mid a_i, b_i, t_i] = \begin{cases} 1 & \text{if } a_i < t_i < b_i \\ 0 & \text{otherwise} \end{cases}$$

In other words, we assume the likelihood is equal to one if the inferred age is consistent with the fossil record. We can represent this likelihood in `RevBayes` using any uniform distribution that is proportional to the likelihood, *i.e.,* non-zero when the likelihood is equal to one (Fig. 4).

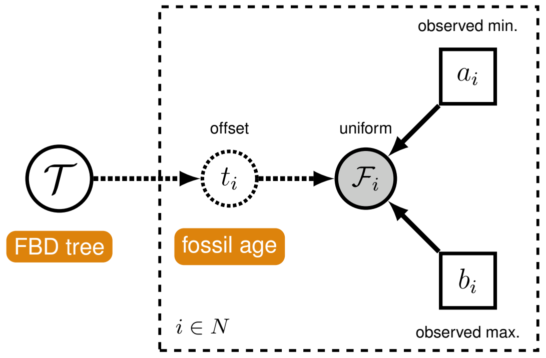


Figure 4: A graphical model of the fossil age likelihood model used in this tutorial. The likelihood of fossil observation $\mathcal{F}_i$ is uniform and non-zero when the inferred fossil age t_i falls within the observed stratigraphic range (a_i, b_i).

2.3 Nucleotide Sequence Evolution

The model component for the molecular data uses a general time-reversible model of nucleotide evolution and gamma-distributed rate heterogeneity across sites. This model of sequence evolution is covered thoroughly in the Substitution Models tutorial.

2.3.1 Lineage-Specific Rates of Sequence Evolution

Rates of nucleotide sequence evolution can vary widely among lineages, and so models that account for this variation by relaxing the assumption of a strict molecular clock can allow for more accurate estimates of substitution rates and divergence times (Drummond et al. 2006). The simplest type of relaxed clock model assumes that lineage-specific substitution rates are independent or "uncorrelated". One example of such an uncorrelated relaxed model is the "Uncorrelated exponential relaxed clock", in which the substitution rate for each lineage is assumed to be independent and identically distributed according to an exponential density (Fig. 5). This is the model that we will use in this tutorial. Another possible uncorrelated relaxed clock model is the uncorrelated lognormal model, described in the Relaxed Clocks tutorial.

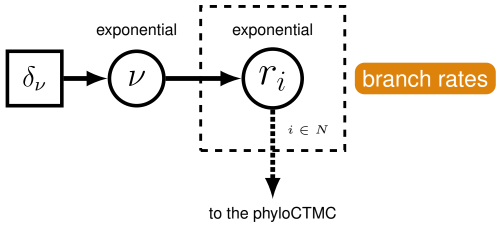


Figure 5: A graphical model of the uncorrelated exponential relaxed clock model. In this model, the clock rate on each branch is independent and identically distributed according to an exponential density with mean drawn from an exponential hyperprior distribution.

2.4 Morphological Character Evolution

For the vast majority of extinct species, fossil morphology is the primary source of phylogenetically informative characters. Therefore, an appropriate model of morphological character evolution is needed to reliably infer the positions of these species in a phylogenetic analysis. The Mk model (Lewis 2001) uses a generalized Jukes-Cantor matrix to allow for the incorporation of morphology into likelihood and Bayesian analyses. One key assumption of this model is that characters exhibit symmetrical change - that a given character is as likely to transition from a one state to another as it is to reverse. In this tutorial we will consider only binary morphological characters, *i.e.,* characters that are coded as being in one of two states, 0 or 1. So for example, the assumption of the Mk model applied to our binary character would mean that a change from a 0 state to a 1 state is as likely as a change from a 1 state to a 0 state. This assumption is equivalent to assuming that the stationary probability of being in a 1 state is equal to 1/2.

In this tutorial, we will relax this assumption by allowing the stationary 1-state probability to take on any value between 0 and 1, drawn from a beta prior distribution. In addition, we will allow each morphological character to have its own substitution matrix with its own stationary frequency (Fig. 6). Then, similar to the way we model site-specific substitution rates using a discretized gamma distribution (described in the Substitution Models tutorial), we will model site-specific stationary frequencies using a discretized beta distribution (Wright et al. 2016).

Because of the way morphological data are collected, we need to exercise caution in how we model the data. Traditionally, phylogenetic trees were built from morphological data using parsimony. Therefore, only parsimony informative characters were collected - that is, characters that are useful for discriminating between phylogenetic hypotheses under the maximum parsimony criterion. This means that many morphological datasets do not contain invariant characters or autapomorphies, as these are not parsimony informative. However, by excluding these low rate of evolution characters, estimates of the branch lengths can be inflated (Felsenstein 1992; Lewis 2001). Therefore, it is important to use models that can condition on the coding bias. `RevBayes` has two ways of doing this: one is used for datasets in which only parsimony informative characters are observed; the other is for datasets in which parsimony informative characters and parsimony uninformative variable characters (such as autapomorphies) are observed.

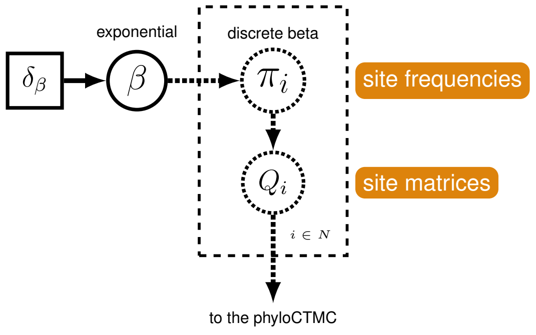


Figure 6: A graphical model of the beta-distributed frequencies across sites model. In this model, stationary frequencies are drawn from a discretized beta distribution and used to generate site-specific asymmetric rate matrices.

2.4.1 The Morphological Clock

Just like with the molecular data (section 2.3.1), our observations of discrete morphological characters are conditional on the rate of change along each branch in the tree. This model component defines the *branch rate model* of the morphological data in the generalized graphical model shown in Fig. 1. The relaxed clock model we described for the molecular data in section 2.3.1 it allows the substitution rate to vary through time and among lineages. For the morphological data, we will instead use a "strict clock" model (Zuckerkandl and Pauling 1962), in which the rate of discrete character change is assumed to be constant throughout the tree. The strict clock is the simplest morphological branch rate model we can construct (graphical model shown in Fig. 7).

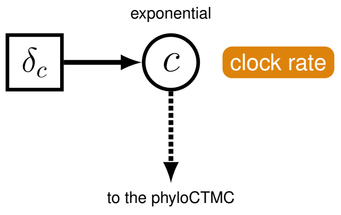


Figure 7: The graphical-model representation of the branch-rate model governing the evolution of morphological characters. This model is consistent with a strict morphological clock, where every branch has the same rate of change (c) and that rate is drawn from an exponential distribution with a rate parameter of δ_c.

3 Exercise: Estimating the Phylogeny and Divergence Times of Fossil and Extant Bears

In this exercise, we will combine different types of data from 22 species of extant and extinct bears to estimate a posterior distribution of calibrated time trees for this group. We have molecular sequence data for ten species, which represent all of the eight living bears and two extinct species sequenced from sub-fossil specimens (*Arctodus simus, Ursus spelaeus*). The sequence alignment provided is a 1,000 bp fragment of the cytochrome-b mitochondrial gene (Krause et al. 2008). The morphological character matrix unites 18 taxa (both fossil and extant) with 62 binary (states `0` or `1`) characters (Abella et al. 2012). For the fossil species, occurrence times are obtained from the literature or fossil databases like the Fossilworks PaleoDB or the Fossil Calibration Database, or from your own paleontological expertise. The 14 fossil species used in this analysis are listed in Table 1 along with the age range for the species and relevant citation. Finally, there are two fossil species (*Parictis montanus, Ursus abstrusus*) for which we do not have morphological character data (or molecular data) and we must use prior information about their phylogenetic relationships to incorporate these taxa in our analysis. This information will be applied using clade constraints.

Table 1: Fossil species used for calibrating divergence times under the FBD model. Modified from Table S.3 in the supplemental appendix of Heath et al. (2014).

Fossil species	**Age range (My)**	**Citation**
Parictis montanus	33.9–37.2	Clark and Guensburg (1972); Krause et al. (2008)
Zaragocyon daamsi	20–22.8	Ginsburg and Morales (1995); Abella et al. (2012)
Ballusia elmensis	13.7–16	Ginsburg and Morales (1998); Abella et al. (2012)
Ursavus primaevus	13.65–15.97	Andrews and Tobien (1977); Abella et al. (2012)
Ursavus brevihinus	15.97–16.9	Heizmann et al. (1980); Abella et al. (2012)
Indarctos vireti	7.75–8.7	Montoya et al. (2001); Abella et al. (2012)
Indarctos arctoides	8.7–9.7	Geraads et al. (2005); Abella et al. (2012)
Indarctos punjabiensis	4.9–9.7	Baryshnikov (2002); Abella et al. (2012)
Ailurarctos lufengensis	5.8–8.2	Jin et al. (2007); Abella et al. (2012)
Agriarctos spp.	4.9–7.75	Abella et al. (2011; 2012)
Kretzoiarctos beatrix	11.2–11.8	Abella et al. (2011; 2012)
Arctodus simus	0.012–2.588	Churcher et al. (1993); Krause et al. (2008)
Ursus abstrusus	1.8–5.3	Bjork (1970); Krause et al. (2008)
Ursus spelaeus	0.027–0.25	Loreille et al. (2001); Krause et al. (2008)

3.1 Tutorial Format

This tutorial follows a specific format for issuing instructions and information

> The boxed instructions guide you to complete tasks that are not part of the `RevBayes` syntax, but rather direct you to create directories or files or similar.

Information describing the commands and instructions will be written in paragraph-form before or after they are issued.

All commands that are specific to `RevBayes` and command-line text, including all `Rev` syntax, are given in **`monotype font`**. Furthermore, blocks of `Rev` code that are needed to build the model, specify the analysis, or execute the run are given in separate shaded boxes. For example, we will instruct you to create a constant node called `a` that is equal to `1.5` using the `<-` operator like this:

```
a <- 1.5
```

It is important to be aware that some PDF viewers may render some characters given as `Rev commands` differently. Thus, if you copy and paste text from this PDF, you may introduce some incorrect characters. Because of this, we recommend that you type the instructions in this tutorial or copy them from the scripts provided.

3.2 Data and Files

On your own computer, create a directory called `RB_TotalEvidenceDating_FBD_Tutorial` (or any name you like).

In this directory download and unzip the archive containing the data files: `data.zip`.

This will create a folder called `data` that contains the files necessary to complete this exercise.

In the `data` folder, you will find the following files:

- `bears_taxa.tsv`: a tab-separated table listing every bear species (both fossil and extant) and their occurrence dates. For extant taxa, the occurrence date is `0.0` (*i.e.,* the present) and for fossil species, the occurrence date is equal to the mean of the age range (the ranges are defined in a separate file).
- `bears_cytb.nex`: an alignment in NEXUS format of 1,000 bp of cytochrome b sequences for 10 bear species. This alignment includes 8 living bears and 2 extinct sub-fossil bears.
- `bears_morphology.nex`: a matrix of 62 discrete, binary (coded `0` or `1`) morphological characters for 18 species of fossil and extant bears.
- `bears_fossil_intervals.tsv`: a tab-separated table containing the age ranges (minimum and maximum in millions of years) for 14 fossil bears.

3.3 Getting Started

Create a new directory (in `RB_TotalEvidenceDating_FBD_Tutorial`) called `scripts`. (If you do not have this folder, please refer to the directions in section 3.2.)

When you execute `RevBayes` in this exercise, you will do so within the main directory you created (`RB_TotalEvidenceDating_FBD_Tutorial`), thus, if you are using a Unix-based operating system, we recommend that you add the `RevBayes` binary to your path.

3.4 Creating Rev Files

For complex models and analyses, it is best to create Rev script files that will contain all of the model parameters, moves, and functions. In this exercise, you will work primarily in your text editor[1] and create a set of modular files that will be easily managed and interchanged. You will write the following files from scratch and save them in the **scripts** directory:

- **mcmc_TEFBD.Rev**: the master Rev file that loads the data, the separate model files, and specifies the monitors and MCMC sampler.
- **model_FBDP_TEFBD.Rev**: specifies the model parameters and moves required for the fossilized birth-death prior on the tree topology, divergence times, fossil occurrence times, and diversification dynamics.
- **model_UExp_TEFBD.Rev**: specifies the components of the uncorrelated exponential model of lineage-specific substitution rate variation.
- **model_GTRG_TEFBD.Rev**: specifies the parameters and moves for the general time-reversible model of sequence evolution with gamma-distributed rates across sites (GTR+Γ).
- **model_Morph_TEFBD.Rev**: specifies the model describing discrete morphological character change (binary characters) under a strict morphological clock.

All of the files that you will create are also provided in the RevBayes tutorial repository[2]. Please refer to these files to verify or troubleshoot your own scripts.

3.5 Start the Master Rev File and Import Data

Open your text editor and create the master Rev file called mcmc_TEFBD.Rev in the **scripts** directory.

Enter the Rev code provided in this section in the new model file.

The file you will begin in this section will be the one you load into RevBayes when you've completed all of the components of the analysis. In this section you will begin the file and write the Rev commands for loading in the taxon list and managing the data matrices. Then, starting in section 3.6, you will move on to writing module files for each of the model components. Once the model files are complete, you will return to editing **mcmc_TEFBD.Rev** and complete the Rev script with the instructions given in section 3.10.

3.5.1 Load Taxon List

Begin the Rev script by loading in the list of taxon names from the **bears_taxa.tsv** file using the **readTaxonData()** function.

```
taxa <- readTaxonData("data/bears_taxa.tsv")
```

This function reads a tab-delimited file and creates a variable called **taxa** that is a list of all of the taxon names relevant to this analysis. This list includes all of the fossil and extant bears.

[1] In section 1.1.1 we offer a recommendation for a text editor.
[2] https://github.com/revbayes/revbayes_tutorial/tree/master/RB_TotalEvidenceDating_FBD_Tutorial/scripts

3.5.2 Load Data Matrices

RevBayes uses the function **readDiscreteCharacterData()** to load a data matrix to the workspace from a formatted file. This function can be used for both molecular sequences and discrete morphological characters.

Load the cytochrome-b sequences from file and assign the data matrix to a variable called **cytb**.

```
cytb <- readDiscreteCharacterData("data/bears_cytb.nex")
```

Next, import the morphological character matrix and assign it to the variable **morpho**.

```
morpho <- readDiscreteCharacterData("data/bears_morphology.nex")
```

3.5.3 Add Missing Taxa

In the descriptions of the files in section 3.2, we mentioned that the two data matrices have different numbers of taxa. Thus, we must add any taxa that are not found in the molecular (**cytb**) partition (*i.e.,* are only found in the fossil data) to that data matrix as missing data, and do the same with the morphological data partition (**morpho**). In order for all the taxa to appear on the same tree, they all need to be part of the same dataset, as opposed to present in separate datasets. This ensures that there is a unified taxon set that contains all of our tips.

```
cytb.addMissingTaxa( taxa )
morpho.addMissingTaxa( taxa )
```

3.5.4 Create Helper Variables

Before we begin writing the Rev scripts for each of the model components, we need to instantiate a couple "helper variables" that will be used by downstream parts of our model specification files. These variables will be used in more than one of the module files so it's best to initialize them in the master file.

Create a new constant node called **n_taxa** that is equal to the number of species in our analysis (22).

```
n_taxa <- taxa.size()
```

Next, create a workspace variable called **mvi**. This variable is an iterator that will build a vector containing all of the MCMC moves used to propose new states for every stochastic node in the model graph. Each time a new move is added to the vector, **mvi** will be incremented by a value of **1**.

```
mvi = 1
```

One important distinction here is that **mvi** is part of the RevBayes workspace and not the hierarchical model. Thus, we use the workspace assignment operator **=** instead of the constant node assignment **<-**.

Save your current working version of **mcmc_TEFBD.Rev** in the **scripts** directory.

We will now move on to the next Rev file and will complete **mcmc_TEFBD.Rev** in section 3.10.

3.6 The Fossilized Birth-Death Process

Open your text editor and create the fossilized birth-death model file called **model_FBDP_TEFBD.Rev** in the **scripts** directory.

Enter the Rev code provided in this section in the new model file.

3.6.1 Speciation and Extinction Rates

Two key parameters of the FBD process are the speciation rate (the rate at which lineages are added to the tree, denoted by λ in Fig. 3) and the extinction rate (the rate at which lineages are removed from the tree, μ in Fig. 3). We'll place exponential priors on both of these values. Each parameter is assumed to be drawn independently from a different exponential distribution with rates δ_λ and δ_μ respectively (see Fig. 3). Here, we will assume that $\delta_\lambda = \delta_\mu = 10$. Note that an exponential distribution with $\delta = 10$ has an expected value (mean) of 1/10.

Create the exponentially distributed stochastic nodes for the **speciation_rate** and **exticntion_rate** using the **~** operator.

```
speciation_rate ~ dnExponential(10)
extinction_rate ~ dnExponential(10)
```

For every stochastic node we declare, we must also specify proposal algorithms (called *moves*) to sample the value of the parameter in proportion to its posterior probability. If a move is not specified for a stochastic node, then it will not be estimated, but fixed to its initial value.

The rate parameters for extinction and speciation are both real, positive numbers (*i.e.,* floating point variables). For both of these nodes, we will use a scaling move (**mvScale()**), which proposes multiplicative changes to a parameter. Many moves also require us to set a *tuning value*, called **lambda** for **mvScale()**, which determine the size of the proposed change. Here, we will use three scale moves for each parameter with different values of lambda. By using multiple moves for a single parameter, we will improve the mixing of the Markov chain.

```
moves[mvi++] = mvScale(speciation_rate, lambda=0.01, weight=1)
moves[mvi++] = mvScale(speciation_rate, lambda=0.1, weight=1)
moves[mvi++] = mvScale(speciation_rate, lambda=1.0, weight=1)

moves[mvi++] = mvScale(extinction_rate, lambda=0.01, weight=1)
moves[mvi++] = mvScale(extinction_rate, lambda=0.1, weight=1)
moves[mvi++] = mvScale(extinction_rate, lambda=1, weight=1)
```

You will also notice that each move has a specified **`weight`**. This option allows you to indicate how many times you would like a given move to be performed at each MCMC cycle. The way that we will run our MCMC for this tutorial will be to execute a *schedule* of moves at each step in our chain instead of just one move per step, as is done in `MrBayes` (Ronquist and Huelsenbeck 2003) or `BEAST` (Drummond et al. 2012; Bouckaert et al. 2014). Here, if we were to run our MCMC with our current vector of 6 moves, then our move schedule would perform 6 moves at each cycle. Within a cycle, an individual move is chosen from the move list in proportion to its weight. Therefore, each move has an equal probability of being executed and will be performed on average one time per MCMC cycle. For more information on moves and how they are performed in `RevBayes`, please refer to the Introduction to MCMC and Substitution Models tutorials.

In addition to the speciation (λ) and extinction (μ) rates, we may also be interested in inferring diversification ($\lambda - \mu$) and turnover (μ/λ). Since these parameters can be expressed as a deterministic transformation of the speciation and extinction rates, we can monitor their values by creating two deterministic nodes using the **`:=`** operator.

```
diversification := speciation_rate - extinction_rate
turnover := extinction_rate/speciation_rate
```

3.6.2 Probability of Sampling Extant Taxa

All extant bears are represented in this dataset. Therefore, we will fix the probability of sampling an extant lineage (ρ in Fig. 3) to 1. The parameter **`rho`** will be specified as a constant node using the **`<-`** operator.

```
rho <- 1.0
```

Because ρ is a constant node, we do not have to assign a move to this parameter.

3.6.3 The Fossil Sampling Rate

Since our data set includes serially sampled lineages, we must also account for the rate of sampling back in time. This is the fossil sampling (or recovery) rate (ψ in Fig. 3), which we will instantiate as a stochastic node (named **`psi`**). As with the speciation and extinction rates (Sect. 3.6.1), we will use an exponential prior on this parameter and use scale moves to sample values from the posterior distribution.

```
psi ~ dnExponential(10)
moves[mvi++] = mvScale(psi, lambda=0.01, weight=1)
moves[mvi++] = mvScale(psi, lambda=0.1, weight=1)
moves[mvi++] = mvScale(psi, lambda=1, weight=1)
```

3.6.4 The Origin Time

We will condition the FBD process on the origin time (ϕ in Fig. 3) of bears, and we will specify a uniform distribution on the origin age. For this parameter, we will use a sliding window move (**`mvSlide`**). A sliding window samples a parameter uniformly within an interval (defined by the half-width **`delta`**). Sliding window moves can be tricky for small values, as the window may overlap zero. However, for parameters such as the origin age, there is little risk of this being an issue.

```
origin_time ~ dnUnif(37.0, 55.0)
moves[mvi++] = mvSlide(origin_time, delta=0.01, weight=5.0)
moves[mvi++] = mvSlide(origin_time, delta=0.1, weight=5.0)
moves[mvi++] = mvSlide(origin_time, delta=1, weight=5.0)
```

Note that we specified a higher move **`weight`** for each of the proposals operating on **`origin_time`** than we did for the three previous stochastic nodes. This means that or move schedule will propose five times as many updates to **`origin_time`** than it will to **`speciation_rate`**, **`extinction_rate`**, or **`psi`**.

3.6.5 The FBD Distribution Object

All the parameters of the FBD process have now been specified. The next step is to use these parameters to define the FBD tree prior distribution, which we will call **`fbd_dist`**.

```
fbd_dist = dnFBDP(origin=origin_time, lambda=speciation_rate, mu=extinction_rate, psi=
    psi, rho=rho, taxa=taxa)
```

3.6.6 Clade Constraints

Note that we created the distribution as a workspace variable using the workspace assignment operator **`=`**. This is because we still need to include a topology constraint in our final specification of the tree prior. Specifically, we do not have any morphological or molecular data for the fossil species *Ursus abstrusus*. Therefore, in order to use the age of this fossil as an observation, we need to specify to which clade it belongs. In this case, *Ursus abstrusus* belongs to the subfamily Ursinae, so we define a clade for the total group Ursinae including *Ursus abstrusus*.

```
clade_ursinae = clade("Melursus_ursinus", "Ursus_arctos", "Ursus_maritimus",
                      "Helarctos_malayanus", "Ursus_americanus", "Ursus_thibetanus",
                      "Ursus_abstrusus", "Ursus_spelaeus")
```

Then we can specify the final constrained tree prior distribution by creating a vector of constraints, and providing it along with the workspace FBD distribution to the constrained topology distribution. Here we use the stochastic assignment operator ~ to create a stochastic node for our constrained, FBD-tree variable (called **fbd_tree**).

```
constraints = v(clade_ursinae)
fbd_tree ~ dnConstrainedTopology(fbd_dist, constraints=constraints)
```

It is important to recognize that we do not know if *Ursus abstrusus* is a *crown* or *stem* Ursinae. Because of this, we defined this clade constraint so that it constrained the *total group* Ursinae and this uncertainty is taken into account. As a result, our MCMC will marginalize over both stem and crown positions of *U. abstrusus* and sample the phylogeny in proportion to its posterior probability, conditional on our model and data.

3.6.7 Moves on the Tree Topology and Node Ages

Next, in order to sample from the posterior distribution of trees, we need to specify moves that propose changes to the topology (**mvFNPR**) and node times (**mvNodeTimeSlideUniform**). Included with these moves is a proposal that will collapse or expand a fossil branch (**mvCollapseExpandFossilBranch**). This will change a fossil that is a sampled ancestor (see Fig. 2 and Sect. 2.1) so that it is on its own branch and vice versa. In addition, when conditioning on the origin time, we also need to explicitly sample the root age (**mvRootTimeSlideUniform**).

```
moves[mvi++] = mvFNPR(fbd_tree, weight=15.0)
moves[mvi++] = mvCollapseExpandFossilBranch(fbd_tree, origin_time, weight=6.0)
moves[mvi++] = mvNodeTimeSlideUniform(fbd_tree, weight=40.0)
moves[mvi++] = mvRootTimeSlideUniform(fbd_tree, origin_time, weight=5.0)
```

3.6.8 Sampling Fossil Occurrence Ages

Next, we need to account for uncertainty in the age estimates of our fossils using the observed minimum and maximum stratigraphic ages provided in the file **bears_fossil_intervals.tsv**. First, we read this file into a matrix called **intervals**.

```
intervals = readDataDelimitedFile(file="data/bears_fossil_intervals.tsv", header=true)
```

Next, we loop over this matrix and for each fossil observation create a uniform random variable representing the likelihood. Remember, we can represent the fossil likelihood using any uniform distribution that is non-zero when the likelihood is equal to one (Sect. 2.2).

For example, if t_i is the inferred fossil age and (a_i, b_i) is the observed stratigraphic interval, we know the likelihood is equal to one when $a_i < t_i < b_i$, or equivalently $t_i - b_i < 0 < t_i - a_i$. So let's represent the likelihood using a uniform random variable uniformly distributed in $(t_i - b_i, t_i - a_i)$ and clamped at zero.

```
for(i in 1:intervals())
{
    taxon = intervals[i][1]
    a_i = intervals[i][2]
    b_i = intervals[i][3]

    t[i] := tmrca(fbd_tree, clade(taxon))

    fossil[i] ~ dnUniform(t[i] - b_i, t[i] - a_i)
    fossil[i].clamp( 0 )
}
```

Finally, we add a move that samples the ages of the fossil nodes on the tree.

```
moves[mvi++] = mvFossilTimeSlideUniform(fbd_tree, origin_time, weight=5.0)
```

3.6.9 Monitoring Parameters of Interest using Deterministic Nodes

There are additional parameters that may be of particular interest to us that are not directly inferred as part of this graphical model. As with the diversification and turnover nodes specified in section 3.6.1, we can create deterministic nodes to sample the posterior distributions of these parameters.

Create a deterministic node called **num_samp_anc** that will compute the number of sampled ancestors in our **fbd_tree**.

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

We are also interested in the age of the most-recent-common ancestor (MRCA) of all living bears. To monitor the age of this node in our MCMC sample, we must use the **clade()** function to identify the node. Importantly, since we did not include this clade in our constraints that defined **fbd_tree**, this clade will not be constrained to be monophyletic. Once this clade is defined, we can instantiate a determinisstic node called **age_extant** with the **tmrca()** function that will record the age of the MRCA of all living bears.

```
clade_extant = clade("Ailuropoda_melanoleuca","Tremarctos_ornatus","Melursus_ursinus",
                  "Ursus_arctos","Ursus_maritimus","Helarctos_malayanus",
                  "Ursus_americanus","Ursus_thibetanus")
age_extant := tmrca(fbd_tree, clade_extant)
```

In the same way we monitored the MRCA of the extant bears, we can also create a deterministic node to monitor the age of a fossil taxon that we are particularly interested in recording. We will monitor the marginal distribution of the age of *Kretzoiarctos beatrix*, which is between 11.2–11.8 My.

```
age_Kretzoiarctos_beatrix := tmrca(fbd_tree, clade("Kretzoiarctos_beatrix"))
```

You have completed the FBD model file. Save **model_FBD_TEFBD.Rev** in the **scripts** directory.

We will now move on to the next model file.

3.7 The Uncorrelated Exponential Relaxed-Clock Model

Open your text editor and create the lineage-specific branch-rate model file called **model_UExp_TEFBD.Rev** in the **scripts** directory.

Enter the `Rev` code provided in this section in the new model file.

For our hierarchical, uncorrelated exponential relaxed clock model, we first define the mean branch rate as an exponential random variable. Then, we specify scale proposal moves on the mean rate parameter.

```
branch_rates_mean ~ dnExponential(10.0)
moves[mvi++] = mvScale(branch_rates_mean, lambda=0.01, weight=1.0)
moves[mvi++] = mvScale(branch_rates_mean, lambda=0.1, weight=1.0)
moves[mvi++] = mvScale(branch_rates_mean, lambda=1.0, weight=1.0)
```

Before creating a rate parameter for each branch, we need to get the number of branches in the tree. For rooted trees with n taxa, the number of branches is $2n - 2$.

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

Then, use a for loop to define a rate for each branch. The branch rates are independent and identically exponentially distributed with mean equal to the mean branch rate parameter we specified above. For each rate parameter we also create scale proposal moves.

```
for(i in 1:n_branches){
    branch_rates[i] ~ dnExp(1/branch_rates_mean)
    moves[mvi++] = mvScale(branch_rates[i], lambda=1.0, weight=1.0)
    moves[mvi++] = mvScale(branch_rates[i], lambda=0.1, weight=1.0)
    moves[mvi++] = mvScale(branch_rates[i], lambda=0.01, weight=1.0)
}
```

Lastly, we use a vector scale move to propose changes to all branch rates simultaneously. This way we can sample the total branch rate independently of each individual rate, which can improve mixing.

```
moves[mvi++] = mvVectorScale(branch_rates, lambda=0.01, weight=4.0)
moves[mvi++] = mvVectorScale(branch_rates, lambda=0.1, weight=4.0)
moves[mvi++] = mvVectorScale(branch_rates, lambda=1.0, weight=4.0)
```

You have completed the FBD model file. Save **`model_UExp_TEFBD.Rev`** in the **`scripts`** directory.

We will now move on to the next model file.

3.8 The General Time-Reversible + Gamma Model of Nucleotide Sequence Evolution

Open your text editor and create the molecular substitution model file called **`model_GTRG_TEFBD.Rev`** in the **`scripts`** directory.

Enter the `Rev` code provided in this section in the new model file.

For our nucleotide sequence evolution model, we need to define a general time-reversible (GTR) instantaneous-rate matrix (*i.e.*, Q-matrix). A nucleotide GTR matrix is defined by a set of 4 stationary frequencies, and 6 exchangeability rates. We create stochastic nodes for these variables, each drawn from a uniform Dirichlet prior distribution.

```
sf_hp <- v(1,1,1,1)
sf ~ dnDirichlet(sf_hp)

er_hp <- v(1,1,1,1,1,1)
er ~ dnDirichlet(er_hp)
```

We need special moves to propose changes to a Dirichlet random variable, also known as a simplex (a vector constrained sum to one). Here, we use a **`mvSimplexElementScale`** move, which scales a single element of a simplex and then renormalizes the vector to sum to one. The tuning parameter **`alpha`** specifies how conservative the proposal should be, with larger values of **`alpha`** leading to proposals closer to the current value.

```
moves[mvi++] = mvSimplexElementScale(er, alpha=10.0, weight=5.0)
moves[mvi++] = mvSimplexElementScale(sf, alpha=10.0, weight=5.0)
```

Then we can define a deterministic node for our GTR Q-matrix using the special GTR matrix function (**`fnGTR`**).

```
Q_cytb := fnGTR(er,sf)
```

Next, in order to model gamma-distributed rates across, we create an exponential parameter α for the shape of the gamma distribution, along with scale proposals.

```
alpha_cytb ~ dnExponential( 1.0 )
moves[mvi++] = mvScale(alpha_cytb, lambda=0.01, weight=1.0)
moves[mvi++] = mvScale(alpha_cytb, lambda=0.1, weight=1.0)
moves[mvi++] = mvScale(alpha_cytb, lambda=1, weight=1.0)
```

Then we create a Gamma(α, α) distribution, discretized into 4 rate categories using the **`fnDiscretizeGamma`** function. Here, **`rates_cytb`** is a deterministic vector of rates computed as the mean of each category.

```
rates_cytb := fnDiscretizeGamma( alpha_cytb, alpha_cytb, 4 )
```

Finally, we can create the phylogenetic continuous time markov chain (PhyloCTMC) distribution for our sequence data, including the gamma-distributed site rate categories, as well as the branch rates defined as part of our exponential relaxed clock. We set the value of this distribution equal to our observed data and identify it as a static part of the likelihood using the **`clamp`** method.

```
physeq ~ dnPhyloCTMC(tree=fbd_tree, Q=Q_cytb, siteRates=rates_cytb, branchRates=
    branch_rates, type="DNA")
phySeq.clamp(cytb)
```

You have completed the FBD model file. Save **`model_GTRG_TEFBD.Rev`** in the **`scripts`** directory.

We will now move on to the next model file.

3.9 Modeling the Evolution of Binary Morphological Characters

Open your text editor and create the morphological character model file called `model_Morph_TEFBD.Rev` in the **`scripts`** directory.

Enter the `Rev` code provided in this section in the new model file.

As stated in the introduction, we will use a prior on state frequencies to relax the key assumption of the Mk model. Because we are working with binary data, we can use a discretized beta distribution to describe the variation in stationary state frequencies across characters. The beta distribution has two parameters, α and β which describe its shape. For simplicity, we will assume that $\alpha = \beta$. The below code draws a value for β from an exponential distribution and places a move on it.

```
beta_hp ~ dnExponential( 1.0 )

moves[mvi++] = mvScale(beta_hp, lambda=1, weight=1.0 )
moves[mvi++] = mvScale(beta_hp, lambda=0.1, weight=1.0 )
moves[mvi++] = mvScale(beta_hp, lambda=0.01, weight=1.0 )
```

Next, we'll create a vector containing four different site stationary state frequencies. This is similar to allowing gamma-distributed rate variation across sites. We will then use these stationary frequencies to generate a set of new substitution matrices which do not enforce the assumptions of the same transition rates at every site and equal forward and backwards transition rates.

```
beta_cats := fnDiscretizeBeta( beta_hp, beta_hp, 4)
for(i in 1:beta_cats.size())
{
    Q_morpho[i] := fnFreeBinary(v(1-beta_cats[i], beta_cats[i]))
}
```

As in the molecular data partition, we will allow gamma-distributed rate heterogeneity among sites.

```
alpha_morpho ~ dnExponential( 1.0 )
rates_morpho := fnDiscretizeGamma( alpha_morpho, alpha_morpho, 4 )

moves[mvi++] = mvScale(alpha_morpho, lambda=0.01, weight=1.0)
moves[mvi++] = mvScale(alpha_morpho, lambda=0.1, weight=1.0)
moves[mvi++] = mvScale(alpha_morpho, lambda=1, weight=1.0)
```

Each data partition has to have a clock rate. For simplicity, we will assume a strict clock rate drawn from an exponential distribution.

```
clock_morpho ~ dnExponential(1.0)

moves[mvi++] = mvScale(clock_morpho, lambda=0.01, weight=4.0)
moves[mvi++] = mvScale(clock_morpho, lambda=0.1, weight=4.0)
moves[mvi++] = mvScale(clock_morpho, lambda=1, weight=4.0)
```

As in our molecular data partition, we now combine our data and our model. There are some unique aspects to doing this for morphology.

You will notice that we have an option called **coding**. This option allows us to condition on biases in the way the morphological data were collected (ascertainment bias). The option **coding=variable** specifies that we should correct for coding only variable characters (discussed in 2.4).

We use the flag **siteMatrices=true** to indicate that we are providing multiple Q matrices generated as a function of our state frequency variation model.

```
phyMorpho ~ dnPhyloCTMC(tree=fbd_tree, siteRates=rates_morpho, branchRates=
    clock_morpho, Q=Q_morpho, type="Standard", coding="variable", siteMatrices=true)
phyMorpho.clamp(morpho)
```

You have completed the FBD model file. Save **model_Morph_TEFBD.Rev** in the **scripts** directory.

We will now move on to the next model file.

3.10 Complete MCMC File

Return to the master Rev file you created in section 3.5 called **mcmc_TEFBD.Rev** in the **scripts** directory.

Enter the Rev code provided in this section in this file.

3.10.1 Source Model Scripts

RevBayes uses the **source()** function to load commands from Rev files into the workspace. Use this function to load in the model scripts we have written in the text editor and saved in the **scripts** directory.

```
source("scripts/model_FBDP_TEFBD.Rev")

source("scripts/model_UExp_TEFBD.Rev")

source("scripts/model_GTRG_TEFBD.Rev")

source("scripts/model_Morph_TEFBD.Rev")
```

3.10.2 Create Model Object

We can now create our workspace model variable with our fully specified model DAG. We will do this with the **model()** function and provide a single node in the graph (**sf**).

```
mymodel = model(sf)
```

The object **`mymodel`** is a wrapper around the entire model graph and allows us to pass the model to various functions that are specific to our MCMC analysis.

3.10.3 Specify Monitors and Output Filenames

The next important step for our master `Rev` file is to specify the monitors and output file names. For this, we create a vector called **`monitors`** that will each sample and record or output our MCMC.

First, we will specify a workspace variable to iterate over the **`monitors`** vector.

```
mni = 1
```

The first monitor we will create will monitor every named random variable in our model graph. This will include every stochastic and deterministic node using the **`mnModel`** monitor. The only parameter that is not included in the **`mnModel`** is the tree topology. Therefore, the parameters in the file written by this monitor are all numerical parameters written to a tab-separated text file that can be opened by accessory programs for evaluating such parameters. We will also name the output file for this monitor and indicate that we wish to sample our MCMC every 10 cycles.

```
monitors[mni++] = mnModel(filename="output/bears.log", printgen=10)
```

The **`mnFile`** monitor writes any parameter we specify to file. Thus, if we only cared about the speciation rate and nothing else (this is not a typical or recommended attitude for an analysis this complex) we wouldn't use the **`mnModel`** monitor above and just use the **`mnFile`** monitor to write a smaller and simpler output file. Since the tree topology is not included in the **`mnModel`** monitor (because it is not numerical), we will use **`mnFile`** to write the tree to file by specifying our **`fbd_tree`** variable in the arguments.

```
monitors[mni++] = mnFile(filename="output/bears.trees", printgen=10, fbd_tree)
```

The last monitor we will add to our analysis will print information to the screen. Like with **`mnFile`** we must tell **`mnScreen`** which parameters we'd like to see updated on the screen. We will choose the age of the MCRCA of living bears (**`age_extant`**), the number of sampled ancestors (**`num_samp_anc`**), and the origin time (**`origin_time`**).

```
monitors[mni++] = mnScreen(printgen=10, age_extant, num_samp_anc, origin_time)
```

3.10.4 Set up the MCMC

Once we have set up our model, moves, and monitors, we can now create the workspace variable that defines our MCMC run. We do this using the **`mcmc()`** function that simply takes the three main analysis components as arguments.

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

The MCMC object that we named **`mymcmc`** has a member method called **`.run()`**. This will execute our analysis and we will set the chain length to **`10000`** cycles using the **`generations`** option.

```
mymcmc.run(generations=10000)
```

Once our Markov chain has terminated, we will want `RevBayes` to close. Tell the program to quit using the **`q()`** function.

```
q()
```

You made it! Save all of your files.

3.11 Execute the MCMC Analysis

With all the parameters specified and all analysis components in place, you are now ready to run your analysis. The `Rev` scripts you just created will all be used by `RevBayes` and loaded in the appropriate order.

Begin by running the `RevBayes` executable.

```
./rb
```

Provided that you started `RevBayes` from the correct directory (**`RB_TotalEvidenceDating_FBD_Tutorial`**), you can then use the **`source()`** function to feed `RevBayes` your master script file (**`mcmc_TEFBD.Rev`**).

```
source("scripts/mcmc_TEFBD.Rev")
```

This will execute the analysis and you should see the following output (though not the exact same values):

```
   Processing file "scripts/mcmc_TEFBD.Rev"
   Successfully read one character matrix from file 'data/bears_cytb.nex'
   Successfully read one character matrix from file 'data/bears_morphology.nex'
   Processing file "scripts/model_FBDP_TEFBD.Rev"
   Processing of file "scripts/model_FBDP_TEFBD.Rev" completed
   Processing file "scripts/model_UExp_TEFBD.Rev"
   Processing of file "scripts/model_UExp_TEFBD.Rev" completed
   Processing file "scripts/model_GTRG_TEFBD.Rev"
   Processing of file "scripts/model_GTRG_TEFBD.Rev" completed
   Processing file "scripts/model_Morph_TEFBD.Rev"
   Processing of file "scripts/model_Morph_TEFBD.Rev" completed

   Running MCMC simulation
   This simulation runs 1 independent replicate.
   The simulator uses 163 different moves in a random move schedule with 267 moves per iteration

Iter        |      Posterior   |     Likelihood   |          Prior   |     age_extant   |    num_samp_anc   |     origin_time   |      elapsed   |        ETA   |
--------------------------------------------------------------------------------------------------------------------------------------------------------------
0           |       -8174.01   |        -8053.8   |       -120.209   |        34.8641   |               0   |         44.4332   |     00:00:00   |     --:--:--   |
10          |       -4654.95   |        -4611.2   |       -43.7495   |        4.32618   |               7   |         45.4494   |     00:00:01   |     --:--:--   |
20          |       -4294.05   |       -4266.91   |       -27.1443   |        4.58804   |               7   |         46.5636   |     00:00:01   |     00:08:19   |
30          |       -4267.35   |       -4233.41   |         -33.94   |         6.8467   |               6   |         45.9177   |     00:00:02   |     00:11:04   |
40          |       -4226.63   |       -4188.32   |       -38.3037   |        6.40484   |               8   |         44.3696   |     00:00:02   |     00:08:18   |
...
```

When the analysis is complete, `RevBayes` will quit and you will have a new directory called **`output`** that will contain all of the files you specified with the monitors (Sect. 3.10.3).

3.12 Evaluate and Summarize Your Results

3.12.1 Evaluate MCMC

In this section, we will evaluate the *mixing* and *convergence* of our MCMC simulation using the program Tracer. We can also summarize the marginal distributions for particular parameters we're interested in. Tracer (Rambaut and Drummond 2011) is a tool for visualizing parameters sampled by MCMC. This program is limited to numerical parameters, however, and cannot be used to summarize or analyze MCMC samples of the tree topology (this will be discussed further in section 3.12.2).

Open Tracer and import the **`bears.log`** file in the ***File→Import New Trace File***. Or click the $\boxed{+}$ button on the left-hand side of the screen to add your log file (see Fig. 10).

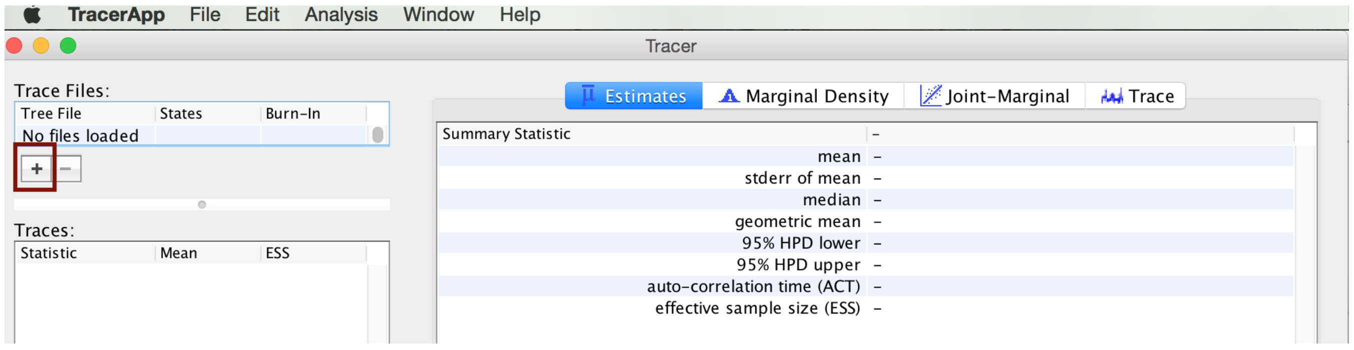
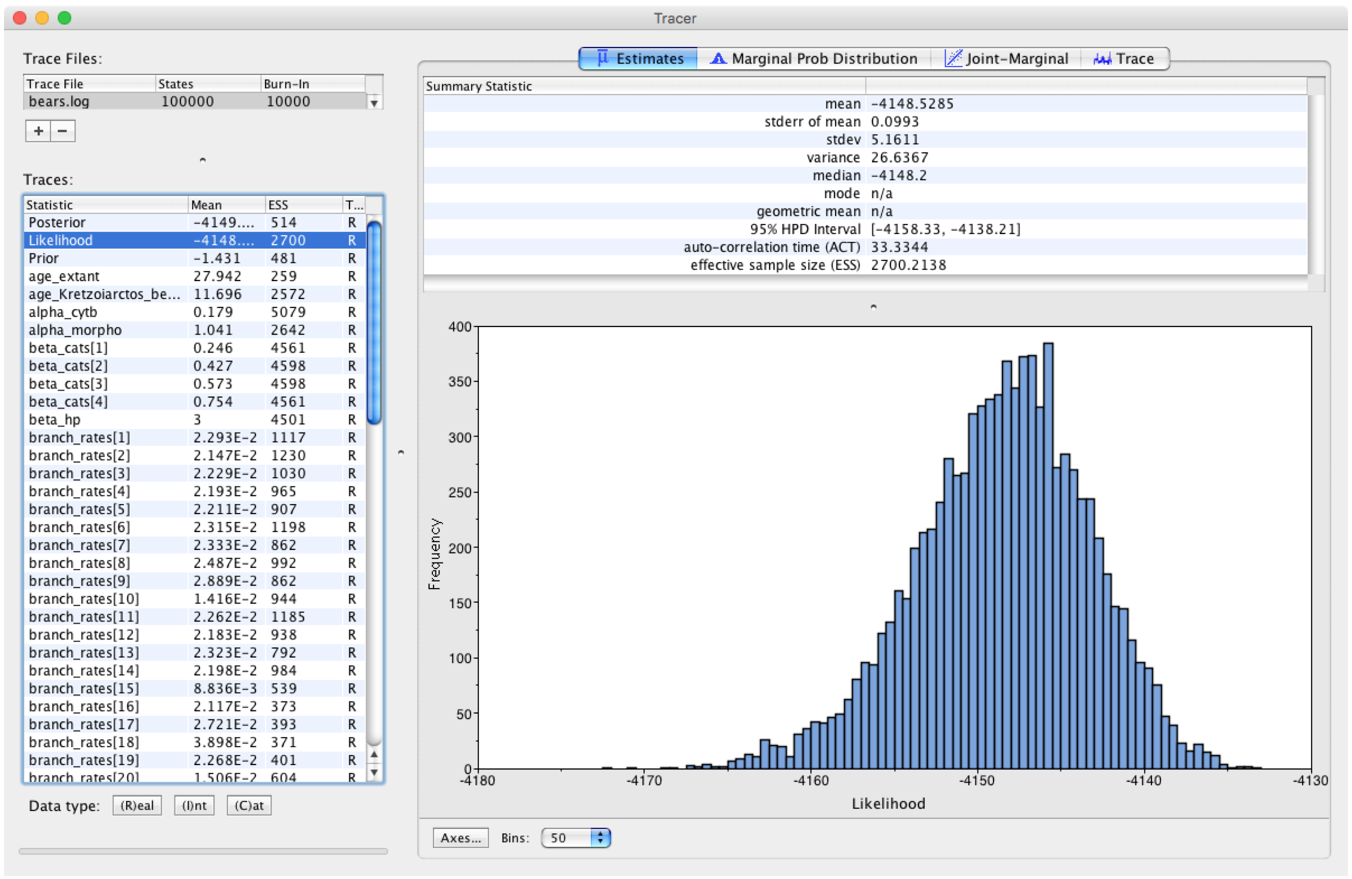


Figure 8: The Tracer window. To add data, click on the "+" sign, highlighted in red above.

Immediately upon loading your file, you will see the list of ***Trace Files*** on the left-hand side (you can load multiple files). The bottom left section, called ***Traces***, provides a list of every parameter in the log file, along with the mean and the Effective Sample Size (ESS) for the posterior sample of that parameter. The ESS statistic provides a measure of the number of independent draws in our sample for a given parameter. This quantity will typically be much smaller than the number of generations of the chain. In Tracer, poor values for the ESS will be colored red. You will likely see a lot of red values because the MCMC runs in this exercise are too short to effectively sample the posterior distributions of most parameters. A much longer analysis is provided in the `output` directory.

The inspection window for your selected parameter is the ***Estimates*** window, which shows a histogram and summary statistics of the values sampled by the Markov chain. Figure 10 shows the marginal distribution of the ***Likelihood*** statistic for the analysis provided in the `output` directory.



Trace Files:

Trace File	States	Burn-In
bears.log	100000	10000

Traces:

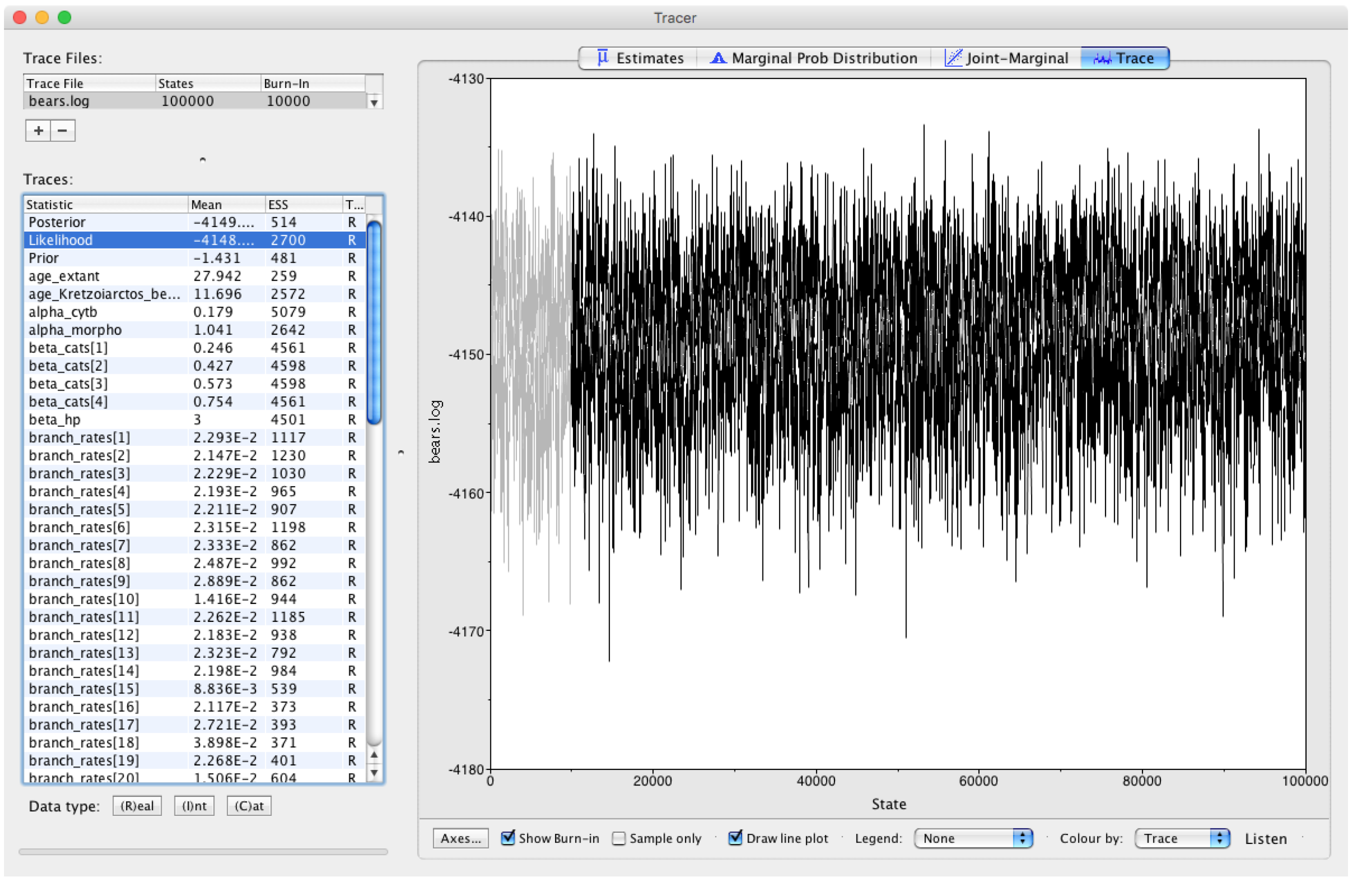
Statistic	Mean	ESS	T...
Posterior	-4149....	514	R
Likelihood	-4148....	2700	R
Prior	-1.431	481	R
age_extant	27.942	259	R
age_Kretzoiarctos_be...	11.696	2572	R
alpha_cytb	0.179	5079	R
alpha_morpho	1.041	2642	R
beta_cats[1]	0.246	4561	R
beta_cats[2]	0.427	4598	R
beta_cats[3]	0.573	4598	R
beta_cats[4]	0.754	4561	R
beta_hp	3	4501	R
branch_rates[1]	2.293E-2	1117	R
branch_rates[2]	2.147E-2	1230	R
branch_rates[3]	2.229E-2	1030	R
branch_rates[4]	2.193E-2	965	R
branch_rates[5]	2.211E-2	907	R
branch_rates[6]	2.315E-2	1198	R
branch_rates[7]	2.333E-2	862	R
branch_rates[8]	2.487E-2	992	R
branch_rates[9]	2.889E-2	862	R
branch_rates[10]	1.416E-2	944	R
branch_rates[11]	2.262E-2	1185	R
branch_rates[12]	2.183E-2	938	R
branch_rates[13]	2.323E-2	792	R
branch_rates[14]	2.198E-2	984	R
branch_rates[15]	8.836E-3	539	R
branch_rates[16]	2.117E-2	373	R
branch_rates[17]	2.721E-2	393	R
branch_rates[18]	3.898E-2	371	R
branch_rates[19]	2.268E-2	401	R
branch_rates[20]	1.506E-2	604	R

Summary Statistic	
mean	-4148.5285
stderr of mean	0.0993
stdev	5.1611
variance	26.6367
median	-4148.2
mode	n/a
geometric mean	n/a
95% HPD Interval	[-4158.33, -4138.21]
auto-correlation time (ACT)	33.3344
effective sample size (ESS)	2700.2138

Figure 9: The Estimates window. The left-hand window provides mean and ESS of the chain. The right-hand window visualizes the distribution of samples.

Are there any parameters that have really low ESS? Why do you think that might be?

Next, we can click over to the Trace window. This window shows us the samples for a given parameter at each iteration of the MCMC. The left side of the chain has a shaded portion that has been excluded as "burn-in". Samples taken near the beginning of chain are often discarded or "burned" because the MCMC may not immediately begin sampling from the target posterior distribution, particularly if the starting condition of the chain is far from the region of highest likelihood.



Trace Files:

Trace File	States	Burn-In
bears.log	100000	10000

Traces:

Statistic	Mean	ESS	T...
Posterior	-4149....	514	R
Likelihood	-4148....	2700	R
Prior	-1.431	481	R
age_extant	27.942	259	R
age_Kretzoiarctos_be...	11.696	2572	R
alpha_cytb	0.179	5079	R
alpha_morpho	1.041	2642	R
beta_cats[1]	0.246	4561	R
beta_cats[2]	0.427	4598	R
beta_cats[3]	0.573	4598	R
beta_cats[4]	0.754	4561	R
beta_hp	3	4501	R
branch_rates[1]	2.293E-2	1117	R
branch_rates[2]	2.147E-2	1230	R
branch_rates[3]	2.229E-2	1030	R
branch_rates[4]	2.193E-2	965	R
branch_rates[5]	2.211E-2	907	R
branch_rates[6]	2.315E-2	1198	R
branch_rates[7]	2.333E-2	862	R
branch_rates[8]	2.487E-2	992	R
branch_rates[9]	2.889E-2	862	R
branch_rates[10]	1.416E-2	944	R
branch_rates[11]	2.262E-2	1185	R
branch_rates[12]	2.183E-2	938	R
branch_rates[13]	2.323E-2	792	R
branch_rates[14]	2.198E-2	984	R
branch_rates[15]	8.836E-3	539	R
branch_rates[16]	2.117E-2	373	R
branch_rates[17]	2.721E-2	393	R
branch_rates[18]	3.898E-2	371	R
branch_rates[19]	2.268E-2	401	R
branch_rates[20]	1.506E-2	604	R

Figure 10: The trace window. The vertical axis is the sampled value of highlighted parameter (in this case likelihood score) and the horizontal axis is the iteration of the MCMC.

The trace window allows us to evaluate how well our chain is sampling parameter space. The output in figure 10 shows fairly good *mixing*–there is no consistent pattern or trend in the samples, nor are there long intervals where the statistic does not change. The presence of a trend or large leaps in a parameter value might indicate that your MCMC is not mixing well. You can read more about MCMC tuning and improving mixing in the tutorials Intro to MCMC and MCMC Algorithms.

Are there any parameters in your log files that show trends or large leaps? What steps might you take to solve these issues?

3.12.2 Summarize Tree

In addition to evaluating the performance and sampling of an MCMC run using numerical parameters, it is also important to inspect the sampled topology and tree parameters. This is a difficult endeavor, however. One tool for evaluating convergence and mixing of the tree samples is RWTY (Warren et al. 2016). In this tutorial, we will only summarize the sampled trees, but we encourage you to consider approaches for assessing the performance of the MCMC with respect to the tree topology.

Ultimately, we are interested in summarizing the sampled trees and branch times, given that our MCMC has sampled all of the important parameters in proportion to their posterior probabilities. `RevBayes` includes some functions for summarizing the tree topology and other tree parameters.

We will complete this part of the tutorial using `RevBayes` interactively. Start up `RevBayes` at the command line. You should do this from within the **`RB_TotalEvidenceDating_FBD_Tutorial`** directory.

```
./rb
```

Read in the MCMC sample of trees from file.

```
trace = readTreeTrace("output/bears.trees")
```

The first thing we need to do is remove taxa for which we did not have any molecular or morphological data. The phylogenetic placement of these taxa is based only on their occurrence times and any clade constraints we applied (section 3.6.6). Because no data are available to resolve their relationships to other lineages, we will treat their placement as *nuisance parameters* and remove them from the summary tree.

We must read our complete taxon list into `RevBayes` so that we can easily access the taxon names of the lineages we'd like to prune.

```
taxa <- readTaxonData("data/bears_taxa.tsv")
```

We will remove two fossil taxa, *Parictis montanus* and *Ursus abstrusus*, from every tree in the trace file before summarizing the samples. These two species have the indices **`17`** and **`20`** in our list of taxa. You can view the list by typing its variable name **`taxa`**:

```
taxa
  [ Ailuropoda_melanoleuca, Helarctos_malayanus, Melursus_ursinus, Tremarctos_ornatus,
  Ursus_americanus, Ursus_arctos, Ursus_maritimus, Ursus_thibetanus, Agriarctos_spp,
  Ailurarctos_lufengensis, Arctodus_simus, Ballusia_elmensis, Indarctos_arctoides,
  Indarctos_punjabiensis, Indarctos_vireti, Kretzoiarctos_beatrix, Parictis_montanus,
  Ursavus_brevirhinus, Ursavus_primaevus, Ursus_abstrusus, Ursus_spelaeus,
    Zaragocyon_daamsi]
```

If you use the size function of the **`trace`** variable by typing the command **`trace.size()`**, you will see that our MCMC sampled 1001 trees. We will first disregard the first 51 trees in our trace by setting the setting the starting value of the iterator we'll use to **`52`**. This burn-in value will be stored in the variable called **`burn`**.

```
burn = 52
```

Use the **fnPruneTree()** function to prune taxon **17** and taxon **20** from each tree using a **for** loop. We'll store each new tree in the vector called **trees**. However, since that vector will be a different length than **trace** we then need another iterator to index **trees**. Call the **trees** index **ti** and increment its value in after it is used using the **++** operator.

```
ti = 1
for(i in burn:trace.size())
{
    trees[ti++] = fnPruneTree(trace.getTree(i), pruneTaxa=v(taxa[17],taxa[20]))
}
```

Now if you view the size of the **trees** vector (use **trees.size()**), you will see that this list of trees has **950** elements.

Then we will use the **mccTree** function to return a maximum clade credibility (MCC) tree. The MCC tree is the tree with the maximum product of the posterior clade probabilities. When considering trees with sampled ancestors, we refer to the maximum sampled ancestor clade credibility (MSACC) tree (Gavryushkina et al. 2016).

```
trace_pruned = treeTrace(trees)
mccTree(trace_pruned, "output/bears.mcc.tre" )
```

When there are sampled ancestors present in the tree, visualizing the tree can be fairly difficult in traditional tree viewers. We will make use of a browser-based tree viewer called IcyTree, created by Tim Vaughan. IcyTree has many unique options for visualizing phylogenetic trees and can produce publication-quality vector files (*i.e.*, svg). Additionally, it correctly represents sampled ancestors on the tree as a node with only one descendant.

Navigate to `http://tgvaughan.github.io/icytree` and open the file **output/bears.mcc.tre** in IcyTree.

Try to replicate Fig. 11. Hint: Look under style, at ***Mark Singletons***. Why might a node with a sampled ancestor be referred to as a singleton?

Try clicking on a branch. What are the fields telling you? Are the ages of the fossil tips the same as what was shown in Tracer?

References

Abella, J., D. M. Alba, J. M. 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.

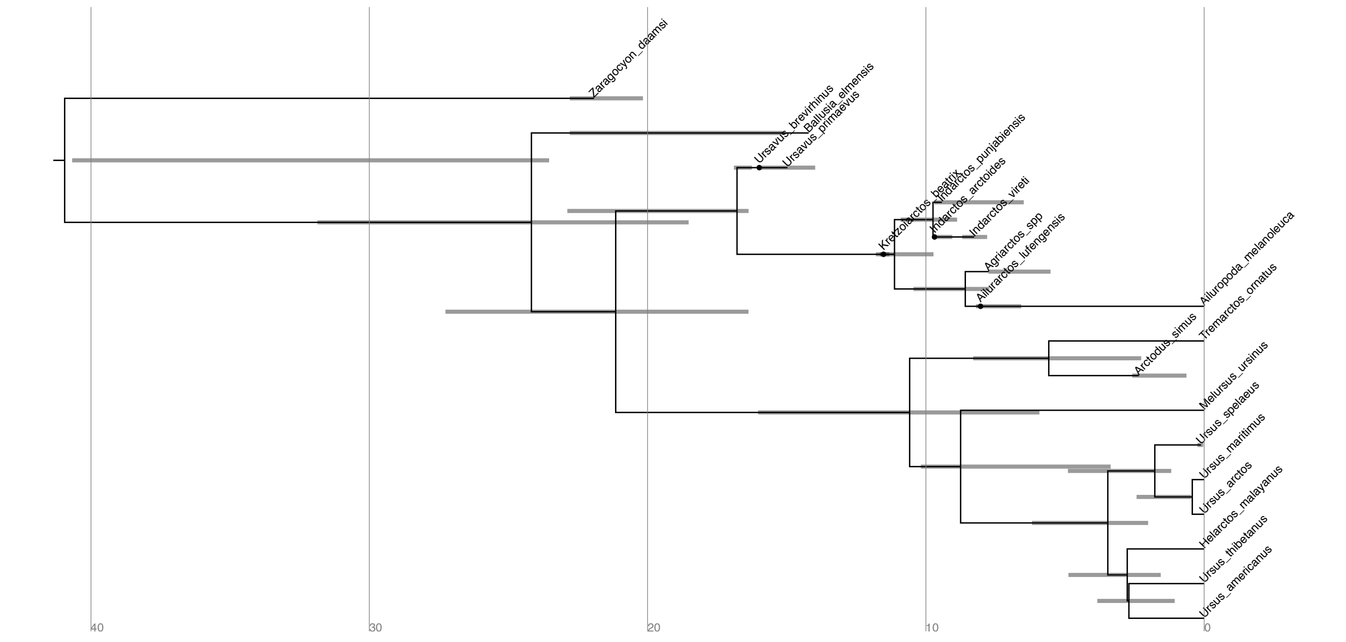


Figure 11: Maximum sampled ancestor clade credibility (MSACC) tree of bear species used in this tutorial. Numbers above fossil nodes indicate the posterior probability of being a sampled ancestor.

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.

Andrews, P. and H. Tobien. 1977. New Miocene locality in turkey with evidence on the origin of *Ramapithecus* and *Sivapithecus*. Nature 268:699.

Baryshnikov, G. F. 2002. Late Miocene *Indarctos punjabiensis atticus* (Carnivora, Ursidae) in Ukraine with survey of *Indarctos* records from the former USSR. Russian J. Theriol 1:83–89.

Bjork, P. R. 1970. The Carnivora of the Hagerman local fauna (late Pliocene) of Southwestern idaho. Transactions of the American Philosophical Society 60:3–54.

Bouckaert, R., J. Heled, D. Kühnert, T. Vaughan, C.-H. Wu, D. Xie, M. A. Suchard, A. Rambaut, and A. J. Drummond. 2014. BEAST 2: a software platform for Bayesian evolutionary analysis. PLoS computational biology 10:e1003537.

Churcher, C., A. Morgan, and L. Carter. 1993. *Arctodus simus* from the Alaskan Arctic slope. Canadian Journal of Earth Sciences 30:1007–1013.

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

Drummond, A., S. Ho, M. Phillips, and A. Rambaut. 2006. Relaxed Phylogenetics and Dating with Confidence. PLoS Biology 4:e88.

Drummond, A., M. Suchard, D. Xie, and A. Rambaut. 2012. Bayesian phylogenetics with beauti and the beast 1.7. Molecular Biology and Evolution 29:1969–1973.

Felsenstein, J. 1992. Phylogenies from Restriction Sites: A Maximum-Likelihood Approach. Evolution 46:159–173.

Foote, M. 1996. On the probability of ancestors in the fossil record. Paleobiology 22:141–151.

Gavryushkina, A., T. A. Heath, D. T. Ksepka, T. Stadler, D. Welch, and A. J. Drummond. 2016. Bayesian total-evidence dating reveals the recent crown radiation of penguins. Systematic Biology .

Geraads, D., T. Kaya, S. Mayda, et al. 2005. Late miocene large mammals from Yulafli, Thrace region, Turkey, and their biogeographic implications. Acta Palaeontologica Polonica 50:523–544.

Ginsburg, L. and J. Morales. 1995. *Zaragocyon daamsi n. gen. sp. nov.*, ursidae primitif du Miocène inférieur d'Espagne. Comptes Rendus de l'Académie des Sciences. Série 2. Sciences de la Terre et des Planètes 321:811–815.

Ginsburg, L. and J. Morales. 1998. Les Hemicyoninae (Ursidae, Carnivora, Mammalia) et les formes apparentées du Miocène inférieur et moyen d'Europe occidentale. Pages 71–123 *in* Annales de Paléontologie vol. 84 Elsevier.

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

Heizmann, E., L. Ginsburg, and C. Bulot. 1980. *Prosansanosmilus peregrinus*, ein neuer machairodontider Felidae aus dem Miozän Deutschlands und Frankreichs. Stuttgarter Beitr. Naturk. B 58:1–27.

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

Höhna, S., M. J. Landis, and T. A. Heath. 2017. Phylogenetic Inference using RevBayes. Current Protocols in Bioinformatics .

Jin, C., R. L. Ciochon, W. Dong, R. M. Hunt, J. Liu, M. Jaeger, and Q. Zhu. 2007. The first skull of the earliest giant panda. Proceedings of the National Academy of Sciences 104:10932–10937.

Krause, J., T. Unger, A. Noçon, A.-S. Malaspinas, S.-O. Kolokotronis, M. Stiller, L. Soibelzon, H. Spriggs, P. H. Dear, A. W. 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.

Lewis, P. O. 2001. A Likelihood Approach to Estimating Phylogeny from Discrete Morphological Character Data. Systematic Biology 50:913–925.

Loreille, O., L. Orlando, M. Patou-Mathis, M. Philippe, P. Taberlet, and C. Hänni. 2001. Ancient DNA analysis reveals divergence of the cave bear, *Ursus spelaeus*, and brown bear, *Ursus arctos*, lineages. Current Biology 11:200–203.

Montoya, P., L. Alcalá, and J. Morales. 2001. *Indarctos* (Ursidae, Mammalia) from the Spanish Turolian (Upper Miocene). Scripta Geologica 122:123–151.

Rambaut, A. and A. J. Drummond. 2011. Tracer v1.5. http://tree.bio.ed.ac.uk/software/tracer/.

Ronquist, F. and J. Huelsenbeck. 2003. MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19:1572–1574.

Ronquist, F., S. Klopfstein, L. Vilhelmsen, S. Schulmeister, D. L. Murray, and A. P. Rasnitsyn. 2012. A total-evidence approach to dating with fossils, applied to the early radiation of the Hymenoptera. Systematic Biology 61:973–999.

Stadler, T. 2010. Sampling-through-time in birth-death trees. Journal of Theoretical Biology 267:396–404.

Warren, D., A. Geneva, D. Swofford, and R. Lanfear. 2016. rwty: R we there yet. A package for visualizing MCMC convergence in phylogenetics .

Wright, A. M., G. T. Lloyd, and D. M. Hillis. 2016. Modeling character change heterogeneity in phylogenetic analyses of morphology through the use of priors. Systematic Biology 65:602–611.

Zhang, C., T. Stadler, S. Klopfstein, T. A. Heath, and F. Ronquist. 2016. Total-evidence dating under the fossilized birth-death process. Systematic Biology 65:228–249.

Zuckerkandl, E. and L. Pauling. 1962. Molecular disease, evolution, and genetic heterogeneity. Pages 189–225 *in* Horizons in Biochemistry (M. Kasha and B. Pullman, eds.) Academic Press, New York.

Version dated: January 5, 2017