Phylogenetic Inference Using RevBayes

Sebastian Höhna,[1,2] Michael J. Landis,[3] and Tracy A. Heath[4]

[1]Department of Integrative Biology, University of California, Berkeley, California
[2]Department of Statistics, University of California, Berkeley, California
[3]Department of Ecology & Evolutionary Biology, Yale University, New Haven, Connecticut
[4]Department of Ecology, Evolution and Organismal Biology, Iowa State University, Ames, Iowa

Bayesian phylogenetic inference aims to estimate the evolutionary relationships among different lineages (species, populations, gene families, viral strains, etc.) in a model-based statistical framework that uses the likelihood function for parameter estimates. In recent years, evolutionary models for Bayesian analysis have grown in number and complexity. RevBayes uses a probabilistic-graphical model framework and an interactive scripting language for model specification to accommodate and exploit model diversity and complexity within a single software package. In this unit we describe how to specify standard phylogenetic models and perform Bayesian phylogenetic analyses in RevBayes. The protocols focus on the basic analysis of inferring a phylogeny from single and multiple loci, describe a hypothesis-testing approach, and point to advanced topics. Thus, this unit is a starting point to illustrate the power and potential of Bayesian inference under complex phylogenetic models in RevBayes.

Keywords: Bayesian phylogenetics • Markov chain Monte Carlo • posterior probabilities • probabilistic graphical models • substitution model

How to cite this article:
Höhna, S., Landis, M.J. and Heath, T.A. 2017. Phylogenetic inference using RevBayes. *Curr. Protoc. Bioinform.* 57:6.16.1-6.16.34.
doi: 10.1002/cpbi.22

INTRODUCTION

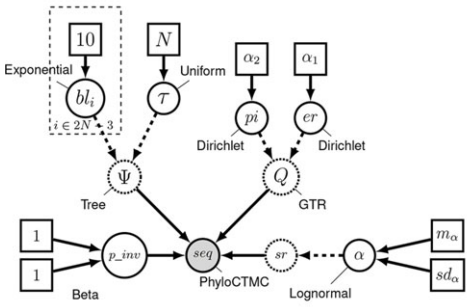
Phylogeny programs, in general, seek to estimate the evolutionary history of a study group from a set of sequences. Several different methodologies exist for estimating phylogenies, including distance-based methods, parsimony, maximum likelihood, and Bayesian inference. Distance-based methods and maximum parsimony estimate a phylogeny based on their respective optimality criteria, e.g., the smallest number of changes required to explain the observed sequence data. Maximum likelihood and Bayesian inference, on the other hand, use stochastic models of evolution to describe the observed data. Bayesian methods are attractive for their ability to directly quantify uncertainty in parameter estimates and because they remain efficient when applied to relatively complex models. However, Bayesian inference requires the specification of prior probabilities that are transformed by the likelihood function into posterior probabilities, which demands special considerations. For a more detailed comparison of different phylogeny inference methods, we refer the reader to more comprehensive reviews, such as Huelsenbeck, Larget, Miller, & Ronquist, (2002); Holder & Lewis (2003); and Yang & Rannala (2012).

Current Protocols in Bioinformatics 6.16.1-6.16.34, March 2017
Published online March 2017 in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpbi.22

Phylogenetic inference in a Bayesian statistical framework aims to estimate the posterior probability distributions of phylogenetic parameters for a given study group. These parameters include the phylogenetic relationships among lineages, the amount of divergence between lineages, rates of substitution, rates of diversification, and many other measures of the tempo and mode of evolution (Huelsenbeck, Ronquist, Nielsen, & Bollback, 2001). Uncertainty in the estimates of these parameters is handled naturally by the Bayesian approach (Huelsenbeck et al., 2002; Holder & Lewis, 2003). The posterior distribution is approximated using numerical algorithms such as Markov chain Monte Carlo (MCMC) sampling (Yang & Rannala, 1997). The popularity of Bayesian phylogenetic methods can be attributed to the straightforward interpretation of the posterior probabilities, the ability to apply complex and mechanistic models of evolution, and their wide availability in a number of software programs, e.g., MrBayes (Ronquist, Teslenko, van der Mark, Ayres, Darling, Höhna, Larget, Liu, Suchard, & Huelsenbeck, 2012), BEAST (Drummond & Rambaut, 2007; Bouckaert, Heled, Kühnert, Vaughan, Wu, Xie, Suchard, Rambaut, & Drummond, 2014), and PhyloBayes (Lartillot, Lepage, & Blanquart, 2009).

The space of described phylogenetic models has expanded rapidly in recent years, giving rise to a wide range of models that vary in their complexity. Simple models of sequence evolution, for example, assume equal base frequencies and equal transition rates between DNA characters (Jukes & Cantor, 1969). Furthermore, simple models assume that all sites in a sequence evolve at the exact same evolutionary rate and via the same process (Yang, 1994). Complex models, however, aim to relax these assumptions by capturing known biological properties, like unequal evolutionary rates at different codon positions (Shapiro, Rambaut, & Drummond, 2006). To keep up to speed with model development, phylogenetic software has also increased in complexity. In a recent paper, we introduced a new approach for phylogenetic model representation—probabilistic graphical models (Höhna, Heath, Boussau, Landis, Ronquist, & Huelsenbeck, 2014)—to enable a flexible and expandable phylogenetic inference platform and a mechanism for visually representing hierarchical models of evolution. Within the probabilistic graphical model paradigm, a statistical model is represented visually as a graph of nodes and edges depicting probability distributions and parameter transformations (see Fig. 6.16.1). These model graphs lay bare the conditional dependence structure of the model. `RevBayes` is a new phylogenetic inference program that harnesses the power of probabilistic graphical models, allowing users to specify any model with any complexity and estimate posterior probabilities in a Bayesian framework (Höhna, Landis, Heath, Boussau, Lartillot, Moore, Huelsenbeck, & Ronquist, 2016). To interface with the `RevBayes` core, we designed a new interactive model-specification language called `Rev` (Höhna et al., 2016), which instantiates a graphical model in computer memory. Both graphical models and the interactive model specification language `Rev` are central parts of `RevBayes` and are similar in philosophy and intention to other probabilistic programming environments, e.g., WinBUGS (Lunn, Thomas, Best, & Spiegelhalter, 2000) and Stan (Carpenter, Gelman, Hoffman, Lee, Goodrich, Betancourt, Brubaker, Guo, Li, & Riddell, in press). The complexity of the language and modeling framework of `RevBayes` may present initial challenges to new users, but it is important to state that once the central concepts are appreciated and understood (this is facilitated by detailed user tutorials and documentation on *http://revbayes.com)*, `RevBayes` provides a powerful platform for conducting fully integrative Bayesian phylogenetic inference. Analysis of biological data under complex models in a Bayesian framework will better capture statistical uncertainty in phylogenetic parameters and enable greater understanding of the processes driving evolution.

In this unit, we provide three related protocols for performing phylogenetic analyses in `RevBayes`. Basic Protocol 1 guides the user through the specification of a phylogenetic model of molecular sequence evolution (i.e., a substitution model and a tree topology




```
for (i in 1:n_branches) {
        bl[i] ~ dnExponential(10.0)
}
topology ~ dnUniformTopology(taxa)
psi := treeAssembly(topology, bl)

alpha1 <- v(1,1,1,1,1,1)
alpha2 <- v(1,1,1,1)
er ~ dnDirichlet( alpha1 )
pi ~ dnDirichlet( alpha2 )
Q := fnGTR(er, pi)

m_alpha <- ln(5.0)
sd_alpha <- 0.587405
alpha ~ dnLognormal( m_alpha, sd_alpha )
sr := fnDiscretizeGamma( alpha, alpha, 4, false )

p_inv ~ dnBeta(1,1)

seq ~ dnPhyloCTMC( tree=psi, Q=Q, pInv=p_invar,
                   siteRates=sr, type="DNA" )
seq.clamp( data )
```

Figure 6.16.1 Graphical model representation of the general time reversible (GTR) substitution model with Gamma distributed rate variation among sites and a proportion of invariant site (GTR + Γ + I). The left side shows the graphical model with the dependencies between the parameters. The right side shows the corresponding `Rev` code to instantiate this model. A brief overview on graphical models is provided in Background Information (Fig. 6.16.5). The stationary frequencies *pi* are drawn from a Dirichlet prior distribution with parameter α_2, and similarly the exchangeability rates *er* are drawn from a Dirichlet prior distribution with parameter α_1. The stationary frequencies *pi* and the exchangeability rates *er* are transformed using the GTR function into the GTR rate matrix. The details of the GTR rate matrix are described under Substitution Model in the Commentary. The probability of a site being invariant `p_inv` is drawn from a Beta(1,1) prior distribution. The rate variation among sites is modeled by a the per-site rates *sr* corresponding to a discretized Gamma distribution with rate and shape parameters α. The parameter α is drawn from a log normal prior distribution. Finally, the tree topology τ is drawn from a uniform topology prior distribution and each branch length bl_i is drawn from an exponential distribution with mean 0.1. The tree variable Ψ is the union of the tree topology with branch lengths. For a more detailed discussion on probabilistic graphical models for phylogenetics, see Höhna et al. (2014) and Höhna et al. (2016). This GTR + Γ + I model is used in Basic Protocol 1 and is explained in more detail step by step.

including branch lengths), including all parameters and MCMC proposals. This procedure is followed by a description of how to apply the MCMC algorithm to approximate the posterior distribution of all model parameters. In Basic Protocol 2, we outline how to construct a model for a multi-locus dataset of protein-coding genes and partition one locus by codon position. Each partition is assumed to be generated from a different substitution model. This second basic protocol extends the core concepts of the first basic protocol to emphasize the generality and flexibility of `RevBayes`. Basic Protocol 3 gives the steps for comparing two substitution models—the one-parameter Jukes-Cantor model (Jukes & Cantor, 1969) and the five-parameter Hasegawa-Kishino-Yano model (Hasegawa, Kishino, & Yano, 1985)—by estimating the marginal likelihood under each model and comparing the support using Bayes factors.

ESTIMATING PHYLOGENY (TOPOLOGY AND BRANCH LENGTHS)

BASIC PROTOCOL 1

Probabilistic graphical models are central to `RevBayes`. The probabilistic graphical model is instantiated in computer memory, variable by variable, by executing a sequence of commands in `Rev`. `Rev` is the programming language used by `RevBayes`. This distinction separates the design of the syntax (i.e., commands) from the implementation of the method. It is of utmost importance to think about a phylogenetic model as a probabilistic graphical model to understand which `Rev` commands are required to build the desired model.

When applying Bayesian methods, the choice of tree model (a prior on the topology and branch lengths) and the substitution model are central to accurate phylogenetic inference. The graphical model in Figure 6.16.1 represents the commonly used general-time reversible substitution model with among-site rate variation (GTR+Γ+I; Tavaré,

1986) with a uniform prior on tree topologies. Additionally, Figure 6.16.1 provides the corresponding `Rev` commands to specify the model parameters and structure. Below, we explain and motivate the steps to build this model and estimate the posterior probabilities of parameters using MCMC.

The MCMC algorithm in `RevBayes` consists of two main components called moves and monitors. Moves are specific tools that update one or several parameters of the model. For example, the sliding-window move updates a single parameter by adding a normally distributed value to the current value of the parameter (for a detailed explanation of common moves used in phylogenetics, see Yang, 2014), while the nearest-neighbor-interchange (NNI) move updates the tree topology by randomly switching neighboring nodes (Höhna, Defoin-Platel, & Drummond, 2008; Höhna & Drummond, 2012). The collection of moves allows the MCMC algorithm to fully explore parameter space. The second main component of the MCMC algorithm are the monitors, which simply define, among other things, the variables to be sampled (i.e., monitored), the format of the output and destination (i.e., stored into a file), and the frequency of sampling. The flexibility of generic monitors enables users in `RevBayes` to store different variables, like the tree topology, in separate files for specific post-processing.

In this unit, we use `RevBayes` interactively, by executing every single line in the terminal. `RevBayes` can also run an analysis from a script file (a text file) by using the `source("my_analysis.Rev")` command. Running `RevBayes` from a script file is preferred, because runs are more easily reproduced, and analyses can run unattended. We provide the examples as scripts in the Supplementary Material and on our Web site: *http://revbayes.com/tutorials.html*.

Necessary Resources

Hardware

Standard workstation (e.g., Macintosh, Windows, Linux, or Unix system) or computer cluster. In principle, `RevBayes` runs on any modern computer architecture. More powerful computers with many CPUs can be helpful for large datasets. The examples described in the protocol as well as small to intermediate datasets run sufficiently well on standard desktop computers.

Software

`RevBayes` is a stand-alone software application that comes with the boost and NCL libraries included (Lewis, 2003). The source code is freely available from *https://github.com/revbayes/revbayes* and can be compiled on any modern platform using `cmake` and a C++ compiler (e.g., GCC 4.2 or newer). Additionally, pre-compiled versions of `RevBayes` for Windows 7 and Mac OS X 10.6 (or higher) are provided (*https://revbayes.com*). The analyses provided in this protocol were written for `RevBayes` version 1.0.2 (commit 67426a2). We recommend using `RevBayes` v1.0.2 or higher for any analyses based on the exercises described below.

Files

`RevBayes` recognizes any standard file format for molecular sequence data files with aligned DNA sequences, e.g., NEXUS, PHYLIP, and FASTA. All data files and analysis scripts are available for download from our Web site: *http://revbayes.com/tutorials.html*. For the analyses outlined in this protocol, we will estimate the phylogeny of 23 primate taxa using an alignment of cytochrome *b* sequences in the file labeled `primates_cytb.nex`.

Getting started

1. Start `RevBayes`:

```
rb
```

In Unix systems, open a terminal window and type `rb` *in the command line. You should make sure that the* `RevBayes` *executable is in your path variable so that you can start* `RevBayes` *from any directory. The directory from which you start* `RevBayes` *is important in order to use relative file paths for reading in data. On Windows systems, you can either double click the* `RevBayes` *executable or open the command-line window and type* `rb.exe`*.*

2. Load the data into your workspace:

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

`RevBayes` *reads in standard NEXUS, FASTA, and Phylip formatted files [see UNIT 6.3 (Desper & Gascuel, 2006) and UNIT 6.4 (Wilgenbusch & Swofford, 2003) for descriptions of these file formats]. Here we read in the cytochrome b (cyt-b) sequence data from a file called* `primates_cytb.nex` *and store the data in a variable named* `data`*. The sequence alignment file is stored in the* `data` *directory, which should be within the directory from which you started* `RevBayes`*. Alternatively, you can provide the full path to read in a file from a different directory. If you want to know the current directory in which* `RevBayes` *is running, then you can query this information by typing* `getwd()` *into the console window.*

3. Query necessary information about the taxa from the data:

```
n_species <- data.ntaxa()
n_branches <- 2 * n_species - 3
taxa <- data.taxa()
```

In step 11 of this protocol, we need to know the number of branches in the tree so that we can create the desired number of branch length variables. We obtain the necessary information from the data variable created above using the member method `.ntaxa()` *(which returns the number of taxa) and* `.taxa()` *(which returns a vector of taxa, i.e., the names of the species). To list the member methods a variable provides, use the member method* `.methods()`*, which is available for every variable, e.g.,* `data.methods()`*.*

4. Instantiate helper variables:

```
mvi = 0
mni = 0
```

In `RevBayes`*, you must create the moves and monitors manually and store them in a vector. Moves are algorithms used to propose new parameter values during the MCMC simulation, and monitors are simple functions that print a subset of variables either to the screen or to a file [also see the section on Markov Chain Monte Carlo (MCMC) Simulation, below]. For convenience, we will create two counter variables that tell us how many moves and monitors we have already created. These counter variables can then be used to add a new move or monitor at the end of the corresponding vectors.*

Substitution model

5. Create the stationary frequency parameters:

```
alpha2 <- v(1,1,1,1)
pi ~ dnDirichlet(alpha2)
```

The first parameter of our model is the vector of the stationary frequencies `pi`*. Every parameter in a Bayesian analysis must have a prior distribution. Prior distributions are required because we are interested in the posterior distribution of the parameters, where the posterior distribution is obtained by calculating the product of the prior distribution and the likelihood function. The stationary frequencies are a vector of probabilities that sum to one. Such a vector is also called a 'simplex'. The Dirichlet distribution is a natural*

choice for the prior distribution because the Dirichlet distribution assigns probability densities to a group of parameters that measure proportions and must sum to one. Here, we have specified a four-parameter Dirichlet prior, where each value describes one of the four stationary frequencies of the GTR model. Without strong prior knowledge about the pattern of stationary frequencies, however, we can better reflect our uncertainty by using a vague prior. Notably, all patterns of stationary frequencies have the same probability density under `alpha2 <- v(1,1,1,1)`*. Note, that we could also fix the stationary frequencies to a constant/fixed parameter by declaring a constant variable for* `pi`*;* `pi <- simplex(1,1,1,1)`.

```
moves[++mvi] = mvBetaSimplex(pi, weight=2.0)
moves[++mvi] = mvDirichletSimplex(pi, weight=1.0)
```

We need to select moves that propose new values of the stationary frequencies during the MCMC simulation because the stationary frequencies of the GTR substitution model are parameters that we want to estimate, i.e., stochastic variables. Here we choose two moves: the `mvBetaSimplex`*, which updates only a single element (one of the stationary frequencies), and the* `mvDirichletSimplex`*, which proposes new values drawn from a Dirichlet distribution centered around the current values. Both moves rescale the stationary frequencies after the proposal so that the values always sum to 1. The first move receives a weight of 2.0 and the second move a weight of 1.0, which implies that the first move will be applied on average 2.0 times per MCMC iteration and the second 1.0 times per iteration (also see the section on Markov Chain Monte Carlo (MCMC) below). We add each move to our vector of moves and automatically increment the counter variable* `mvi` *using the pre-increment operator* `++mvi`.

6. Create the exchangeability rate parameters:

```
alpha1 <- v(1,1,1,1,1,1)
er ~ dnDirichlet(alpha1)
```

Next, we create a stochastic variable for the exchangeability parameters of the GTR substitution model. In `RevBayes` *we have adopted the convention that exchangeability rates sum to 1, i.e., the exchangeability rates are normalized. This convention ensures that the model is identifiable and yields branch lengths in expected number of substitutions. Hence, we can apply a Dirichlet prior distribution to the exchangeability rates. Representing our lack of prior information about the exchangeability rates, we use a flat Dirichlet prior distribution:* `alpha1 <- v(1,1,1,1,1,1)`.

```
moves[++mvi] = mvBetaSimplex(er, weight=3.0)
moves[++mvi] = mvDirichletSimplex(er, weight=1.5)
```

We need to specify moves for the exchangeability rates as we did for the stationary frequencies. Since the exchangeability rates are also of type simplex and drawn from a Dirichlet distribution, we can use the same type of moves as before.

7. Combine exchangeability rates and stationary frequencies into the substitution rate matrix:

```
Q:= fnGTR(er,pi)
```

Given the exchangeability rates and stationary frequencies, we can deterministically compute the instantaneous substitution rate matrix. We show the relationship of the parameters and the transformation into the rate matrix in the section titled Phylogenetic Models and Theory in the Commentary, below. In `RevBayes`*, we use the function* `fnGTR` *given the parameters* `er` *and* `pi` *to instantiate the deterministic variable* `Q`*. The dependencies of the substitution rate matrix* `Q` *and its parameters* `er` *and* `pi` *is shown using dashed arrows in Figure 6.16.1.*

8. Model among site rate variation (ASRV):

```
alpha_prior_mean <- ln(1.5)
alpha_prior_sd <- 0.587405
```

```
alpha ~ dnLognormal(alpha_prior_mean, alpha_prior_sd)
sr:= fnDiscretizeGamma(shape=alpha, rate=alpha,
  numCats=4, median=FALSE)
```

To specify the discrete Gamma ASRV model, we need a deterministic node that is a vector of k rates drawn from a Gamma distribution with k rate categories. The `fnDiscretizeGamma` *function returns this deterministic node and takes four arguments: the shape and rate of the Gamma distribution, the number of categories, and whether to use the mean or median of the quantiles. Since we want to discretize a mean-one Gamma distribution, we can pass in* `alpha` *for both the shape and rate.* `alpha` *itself is a stochastic variable drawn from a log normal prior distribution with location parameter* `ln(1.5)` *and scale of* `0.587405`*. Thus, our resulting 95% prior interval ranges from a vector of site rates of [0.029, 0.235, 0.801, 2.936] to [0.491, 0.798, 1.084, 1.628], representing either a 100-fold difference in rates or a 3-fold difference in rates. This specific prior shows an example of a biologically motivated prior. Note that here, by convention, we set k = 4 using the argument* `numCats=4` *(Yang, 1994). Particular to* `RevBayes` *is that the number of rate categories k and the distribution of the rate categories can be exchanged easily, for example, by using a log normal distribution with 6 rate categories instead (*`sr:= fnDiscretizeDistribution(dnLognormal(mean=0, sd=alpha), num_cats=6))`.*

```
moves[++mvi] = mvScale(alpha, lambda=1.0, weight=2.0)
```

The random variable that controls the rate variation is the stochastic node `alpha`*. We apply a simple scale move to this parameter (for details also see Yang, 2014). The scale move proposes new parameter values by multiplying the current value by a factor* $e^{\lambda u}$*, where u is drawn uniformly from (−0.5, 0.5). Here,* λ *is a tuning parameter that controls the range of proposed scaling factors. For example, if* $\lambda = 1.0$*, then the current value is scaled by a factor between* $e^{-0.5}$ *and* $e^{0.5}$*, and if* $\lambda = 2.0$*, then the scaling factor is between* e^{-1} *and* e^{1}*. Tuning parameters can be set manually or tuned automatically to achieve "optimal" performance (Haario, Saksman, & Tamminen, 1999; Roberts & Rosenthal, 2009).*

9. Model probability of invariable sites:

```
p_inv ~ dnBeta(1,1)
```

This adds a stochastic variable for the probability of a site being invariant, `p_inv`*, to the parameter for rate variation among sites. Since* `p_inv` *should be defined as a probability (any number between 0 and 1), we choose a Beta(1,1) distribution as the prior. This flat Beta distribution gives equal probability to any value between 0 and 1.*

```
moves[++mvi] = mvBetaProbability(p_inv, weight=2.0)
```

For the `p_inv` *variable, we apply a Beta-Probability move which is applicable to stochastic variables of type Probability and uses a Beta distribution to propose new values. We could also have used (additionally) a scaling move (*`mvScale`*) or sliding move (*`mvSlide`*) as for any other continuous variable (Yang, 2014).*

Tree topology and branch lengths

In the previous steps, we specified the substitution model with its parameters. The substitution model is necessary to compute the likelihood of a phylogeny given the data. Now, we specify our main variable of interest: the phylogeny. The phylogeny variable comprises the tree topology and the branch lengths. We will first specify the topology and branch length variables independently, then merge the parameters together to create a phylogeny variable.

10. Specify a uniform prior distribution on the tree topology:

```
out_group = clade("Galeopterus_variegatus")
```

```
topology ~ dnUniformTopology(taxa, outgroup=
  out_group)
```

In this protocol, we will estimate an unrooted phylogeny. We first specify a uniform distribution on all topologies that have this set of taxa and our outgroup. Hence, every topology has the same prior probability of one over the number of distinct topologies (for computing the number of topologies, see Felsenstein, 1978). Note that the prior probability can become very small because the number of topologies is growing super-exponentially with the number of taxa, but this is no concern for Bayesian phylogenetics because all topologies are equally probable a priori. For this dataset, the outgroup is a single species: the flying lemur (Galeopterus variegatus). However, `RevBayes` *allows you to specify any number of outgroup species. Other software, such as* `MrBayes`*, implicitly assume that the first entry in the data file is the outgroup species. In* `RevBayes`*, we force users to select the outgroup manually to make all decisions conscious and visible. Since the trees sampled by MCMC are actually all unrooted, they can also be re-rooted after the analysis even if no outgroup is specified (see UNIT 6.1; Page, 2003).*

```
moves[++mvi] = mvNNI(topology, weight=5.0)

moves[++mvi] = mvSPR(topology, weight=3.0)
```

We apply two moves on the tree topology: a nearest neighbor interchange move (`mvNNI`*) and a subtree-prune-regraft move (*`mvSPR`*). These two moves change only the tree topology. The tree topology is often the most difficult parameter to estimate (mix over). Therefore, more specialized moves that propose new topologies (and branch lengths) using more sophisticated methods are available in* `RevBayes`*. For a discussion, see Höhna et al. (2008); Lakner, van der Mark, Huelsenbeck, Larget, & Ronquist (2008); Höhna & Drummond (2012); Yang (2014).*

11. Specify an exponential branch length prior:

```
for (i in 1:n_branches) {
   bl[i] ~ dnExponential(10.0)
   moves[++mvi] = mvScale(bl[i])
}
TL:= sum(bl)
```

Every branch length is represented in this model as its own independent and identically distributed stochastic variable. This is achieved by using a `for` *loop. The* `for` *loop corresponds to the repetition of the variable* `bl` *(shown as a dashed box in Fig. 6.16.1; also see Fig. 6.16.2) around the* `bl` *variable in Figure 6.16.1. In the* `for` *loop, we create each branch length variable drawn from an exponential distribution with rate 10. Remember that an exponential distribution with rate 10 has an expectation of one divided by the rate (1/10). Additionally, we compute the total tree length by summing all branch lengths, which is done in the deterministic variable* `TL`*. The tree length is not a variable of the model itself, but we might want to monitor it to, for example, learn about the posterior distribution of the tree length itself instead of only the single branch lengths. Even though the exponential branch length prior is the most common choice, it has been shown to bias tree inference (e.g., Yang & Rannala, 2005; Brown, Hedtke, Lemmon, & Lemmon, 2010; Rannala, Zhu, & Yang, 2012). Alternatively, we could have specified a prior distribution on the tree length, such as a Gamma prior, and a distribution on the relative branch length using a Dirichlet distribution (Zhang, Rannala, & Yang, 2012). Then, the actual branch length is the product of the relative branch length and the tree prior.*

```
psi:= treeAssembly(topology, bl)
```

Finally, we combine the tree topology variable (`topology`*) and branch lengths (*`bl`*) into a tree variable with branch lengths. The separation between tree topology and branch lengths allows us more flexibility in specifying prior distributions on each.*

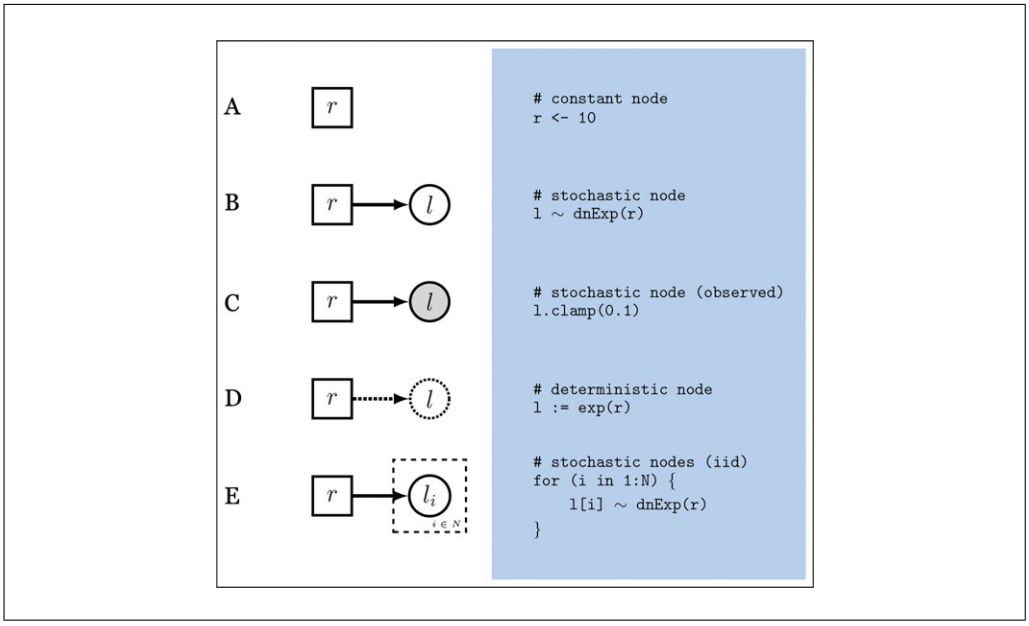


Figure 6.16.2 Graphical model variables and `Rev` code. This figure was modified from Figure 1 of Höhna et al. (2016). Parent-child relationships between model variables are given on the left-hand side. Corresponding `Rev` code is given on the right hand side. (A) The variable `r` is a constant node (`<-`) with the value `10`. (**B**) The variable `l` is a stochastic node (`~`) that is exponentially distributed with rate `r=10`; whenever the value of `r` changes, so does the probability of 1. (**C**) The variable `l` is observed to have the value `0.1` and now contributes to the model likelihood rather than the model prior; the likelihood is $p(x = 0.1 \mid \lambda = 10) = 0.3678794$ under an exponential distribution. (**D**) The variable `l` is a deterministic node (`:=`) whose value always equals $l = e^r$; whenever r changes, the value of l is updated immediately. For $r = 10$, $l = e^{10} = 22026.47$. (**E**) Each element `l [i]` in the vector of variables `l` is independently and identically distributed under an Exponential distribution, `dnExp(r)`.

Putting it all together

12. Model character evolution along the phylogeny:

```
seq ~ dnPhyloCTMC(tree=psi, Q=Q, siteRates=sr,
  pInv=p_inv, type="DNA")
```

We have fully specified all of the parameters of our phylogenetic model—the tree topology with branch lengths, and the substitution model that describes how the sequence data evolved over the tree with branch lengths. Collectively, these parameters comprise a distribution called the 'phylogenetic continuous-time Markov chain', and we use the `dnPhyloCTMC` *distribution to create a stochastic variable* `seq` *for the sequence data. This distribution requires several input arguments (arguments marked with * are optional): (1) the tree with branch lengths* `psi`*; (2) the instantaneous-rate matrix* `Q`*; (*3) the rate categories for the site-specific rates* `sr`*; (*4) the probability of a site being invariant* `p_inv`*; (*5) the clock rate that scales all branch lengths, which is not part of this model; and (6) the type of character data "*`DNA`*".*

13. Attach the data to the sequence variable:

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

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

14. Instantiate a model object:

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

Finally, we wrap the entire model to provide convenient access to the model graph. To do this, we only need to give the `model` *function a single node. With this node, the* `model` *function can find all of the other nodes by following the arrows in the graphical model. The variable* `mymodel` *now holds its own independent copy of the entire model graph in computer memory.*

Performing an MCMC analysis

15. Add monitors to store samples from the MCMC simulation into files:

```
monitors[++mni] = mnModel(filename="output/primates_
  cytb.log", printgen=10, separator=TAB)
monitors[++mni] = mnFile(filename="output/primates_
  cytb.trees", printgen=10, separator=TAB, psi)
monitors[++mni] = mnScreen(printgen=1000, TL)
```

For our MCMC analysis, we need to set up a vector of monitors to record the states of our Markov chain. The monitor functions are all called `mn*`*, where* `*` *is the wildcard representing the monitor type. First, we will initialize the model monitor using the* `mnModel` *function. This creates a new monitor object that will output the states for all simple, numeric model parameters (i.e., not the tree and the rate matrix) when passed into a MCMC function. The* `mnFile` *monitor will record the states for only the parameters passed in as arguments. We use this monitor to specify the output for our sampled trees and branch lengths:* `psi`*. Finally, the screen monitor will report the states of specified variables to the screen with* `mnScreen`*. The screen monitor is just for our convenience, to see what is happening. All monitors have an argument called* `printgen` *that specifies how frequently samples are stored (i.e., thinning of the samples). Thinning is important to reduce file size and maximize the ratio of effective sample size (ESS) to the number of samples taken.*

16. Run an MCMC simulation:

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

mymcmc.burnin(generations=10000,tuningInterval=200)

mymcmc.run(generations=30000)
```

With a fully specified model, a set of monitors, and a set of moves, we can now set up the MCMC algorithm that will sample parameter values in proportion to their posterior probability. The `mcmc` *function will create our MCMC object. We may wish to run the* `.burnin()` *member function. Recall that this function does not specify the number of states that we wish to discard from the MCMC analysis as burnin (i.e., the samples collected before the chain converges to the stationary distribution). Instead, the* `.burnin()` *function specifies a completely separate preliminary MCMC simulation that is used to auto-tune the moves to improve mixing of the MCMC analysis. Additionally, the* `.burnin()` *function will move the parameters towards the stationary distribution and thus fewer, if any, samples from the actual MCMC simulation have to be discarded. When the analysis is complete, you will have the monitored files in your output directory. See the section titled Guidelines for Understanding Results for information on evaluating and summarizing MCMC output.*

PARTITIONED DATA ANALYSIS

BASIC PROTOCOL 2

This is an introduction to partitioned phylogenetic analysis. Partitioned analyses allow for different sets of homologous sites to evolve according to different sets of evolutionary parameters—for example, if two genes with different functions face different selection

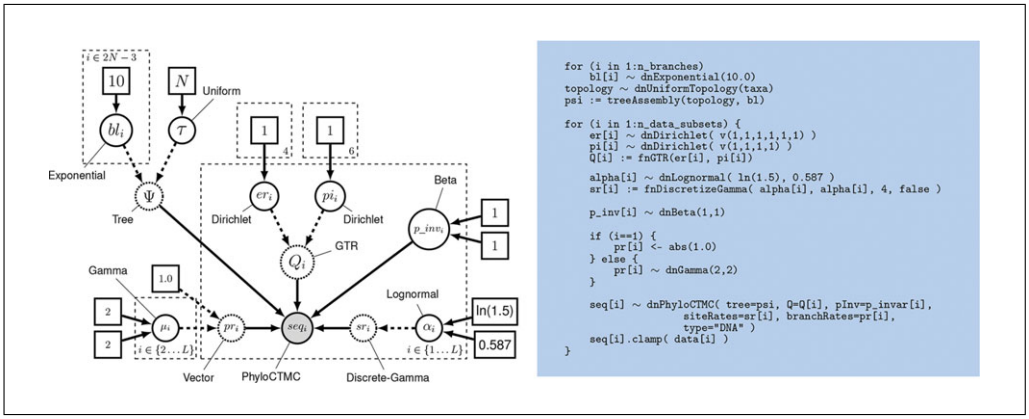


Figure 6.16.3 Graphical model representation of the partitioned general time reversible (GTR) substitution model with Gamma distributed rate variation among sites and a proportion of invariant site (GTR + Γ + I). Note that all substitution process parameters are replicated by the plate, but there is one common tree parameter shared across subsets of characters in the partition. The left side shows the graphical model with the dependencies between the parameters. The right side shows the corresponding `Rev` code to instantiate this model. For a more detailed discussion on probabilistic graphical models for phylogenetics, see Höhna et al. (2014) and Höhna et al. (2016). In contrast to Figure 6.16.1, the model graph shown here instantiates many of its prior distributions directly with unnamed parameter values rather than with named model variables. This leads to more compact code. For example, `pi` is assigned a Dirichlet prior with concentration parameters `v(1,1,1,1)` rather than creating a new `alpha2` variable. In the model graph, the anonymous vector `v(1,1,1,1)` appears as a plate of four constant nodes, each with value 1. Importantly, using named or unnamed variables is a notational difference that does not affect the performance of the model itself.

pressures, and they may evolve according to different processes. Within a single protein-coding gene, the third codon position is expected to have a relatively high rate of substitution when compared with first and second codon positions, owing to the structure of the genetic code (Bull, Huelsenbeck, Cunningham, Swofford, & Waddell, 1993; Brandley, Schmitz, & Reeder, 2005; Brown & Lemmon, 2007).

This protocol describes how to perform a partitioned data analysis using `RevBayes`. A key idea underlying this section is plate notation (Fig. 6.16.3). In a graphical model, a plate represents a set of random variables and their dependencies that are replicated with identical structure and visually represented as a dashed rectangle encompassing the replicated variable nodes. This has a natural correspondence to the `for` loop in programming languages. In both cases, this helps to keep the model concise and easy to modify. For practical purposes, by instantiating the distribution of branch lengths as a "repetition of variables" in Basic Protocol 1, this creates a plate of branch length variables. Plates are common structures in phylogenetic models, but here we focus on their application to multi-locus partitioned analyses.

In this example, we will consider a partition with four subsets of characters: first and second codon positions for cyt-b, third codon positions for cyt-b, first and second codon positions for COX2 (cytochrome *c* oxidase subunit *2),* and third codon positions for COX2. Each character subset will evolve under its own substitution process, each of which being similar to the single-locus model in Basic Protocol 1. Unlike the substitution process parameters, all character subsets in the partition share a single phylogeny parameter.

Necessary Resources

All of the necessary resources for this tutorial are described above in Basic Protocol 1. All data files and analysis scripts are available for download from our Web site, *http://revbayes.com/tutorials.html*. For this protocol, we will use a

second dataset in addition to the cytochrome *b* alignment analyzed in Basic Protocol 1. This dataset contains 23 primate sequences for the gene cytochrome oxidase II in the file called `primates_cox2.nex`.

Getting started

1. Start `RevBayes`:

```
rb
```

In Unix systems, open a terminal window and type `rb` *in the command line. On Windows systems, you can either double click the* `RevBayes` *executable or open the command-line window and type* `rb.exe`*.*

2. Load the two alignments from file:

```
data_cytb <- readDiscreteCharacterData
  ("data/primates_cytb.nex")
data_cox2 <- readDiscreteCharacterData
  ("data/primates_cox2.nex")
```

We read in the sequence data from the files `primates_cytb.nex` *and* `primates_cox2.nex` *and store them in the variables named* `data_cytb` *and* `data_cox2`*. See step 2 of Basic Protocol 1 for more information about these commands.*

3. Divide the data into partitions:

```
data[1] <- data_cox2
data[2] <- data_cox2
data[3] <- data_cytb
data[4] <- data_cytb
```

First, we store two copies of each gene into a vector. Elements `data[1]` *and* `data[2]` *correspond to cyt-b, while* `data[3]` *and* `data[4]` *correspond to cox2.*

```
data[1].setCodonPartition(v(1,2))

data[2].setCodonPartition(3)

data[3].setCodonPartition(v(1,2))

data[4].setCodonPartition(3)
```

Then, we assign partitions to differentiate third codon positions from first and second codon positions for cyt-b and COX2.

4. Record the dimensions of the dataset:

```
n_species <- data_cytb.ntaxa()

n_branches <- 2 * n_species - 3

taxa <- data[1].taxa()

n_data_subsets <- data.size()
```

In the latter part of this protocol, we need to know the number of branches in the tree and the number of data subsets in the partition to design the structure of the model. See step 3 of Basic Protocol 1 for more information about these commands.

5. Instantiate helper variables:

```
mvi = 0

mni = 0
```

In RevBayes *you create the moves and monitors manually and store them in a vector. For convenience, we will create two counter variables that tell us how many moves and monitors we have already created. These counter variables can then be used to add a new move or monitor add the end of vectors. See step 4 of Basic Protocol 1 for more information about these commands.*

Substitution model

6. Declare a `for` loop over data subsets:

```
for (i in 1:n_data_subsets)
```

We use a `for` *loop to assign individual substitution model parameters to each data subset in the partition. The contained code will be executed once for each of the four data subsets. From the graphical modeling perspective, the* `for` *loop behaves like plate representation, where each data subset is drawn from a common model structure. Each iteration of the loop essentially creates an independent copy of the model defined in Basic Protocol 1.*

7. Begin defining the code block for the `for` loop to execute:

```
{
```

All code contained between the open curly brace (`{`*) and the matching closed curly brace (*`}`*) will be executed once for each value of* `i` *between* `1` *and* `n_data_subsets` *times (this* `for` *loop is closed in step 13, below). Here, we will create stationary frequencies, exchangeability rates, rate matrices, among-site rate variation multipliers, and invariable site parameters for each of the four character subsets in our partition. Building upon Basic Protocol 1, we are now specifying the model with vectors of parameters, where each element in the vector is associated with a block of characters. For example,* `er` *will no longer correspond to the simplex of exchangeability rates for the entire analysis, but rather a vector of four simplices of exchangeability rates for the partitioned analysis, accessed by* `er[1]`, `er[2]`, `er[3]`, *and* `er[4]`. *The following code block is indented to emphasize that it will be executed within a loop.*

8. Create the *i*th stationary frequency parameters:

```
pi[i] ~ dnDirichlet(alpha=[1,1,1,1])

moves[++mvi] = mvBetaSimplex(pi[i], weight=2.0)

moves[++mvi] = mvDirichletSimplex(pi[i], weight=2.0)
```

Each data subset has its own set of stationary frequencies, each of which has a flat Dirichlet prior distribution. Two MCMC moves are assigned to update the parameter. `mvBetaSimplex` *updates one simplex value at a time, whereas* `mvDirichletSimplex` *updates all simplex values simultaneously. See step 5 of Basic Protocol 1 for more information about these commands.*

9. Create the *i*th exchangeability rate parameters:

```
er[i] ~ dnDirichlet(alpha=[1,1,1,1,1,1])
moves[++mvi] = mvBetaSimplex(er[i], weight=3.0)
moves[++mvi] = mvDirichletSimplex(er[i], weight=1.5)
```

Each data subset has its own set of exchangeability rates, each of which has a flat Dirichlet prior distribution. See step 6 of Basic Protocol 1 for more information about these commands.

10. Create the *i*th rate matrix from the stationary frequencies and exchangeability rate parameters:

```
Q[i]:= fnGTR(er[i], pi[i])
```

Each character subset evolves according to its own GTR rate matrix. Each rate matrix is a function of exchangeability rates and stationary frequencies associated with that particular character subset in the partition. If the value of `er[1]` *or* `pi[1]` *changes, it will only cause the value of* `Q[1]` *to change; the values of* `Q[2]`, `Q[3]`, *and* `Q[4]` *will remain the same because they are not child nodes of* `er[1]` *or* `pi[1]`. *See step 7 of Basic Protocol 1 for more information about these commands.*

11. Create the *i*th discrete Gamma distribution to model among site rate variation (ASRV):

```
alpha[i] ~ dnLognormal(mean=ln(1.5), sd=0.587405)
sr[i]:= fnDiscretizeGamma(alpha[i], alpha[i], 4,
  false)
moves[++mvi] = mvScale(alpha[i], lambda=1.0,
  weight=2.0)
```

We create a discrete Gamma distribution to model among site rate variation, which takes `alpha[i]` *as a shape and rate parameter.* `alpha[i]` *is log-normally distributed; unlike Basic Protocol 1, we set the* `mean` *and* `sd` *parameters directly rather than creating two constant nodes only to pass those constant nodes as arguments. See step 8 of Basic Protocol 1 for more information about these commands.*

12. Create the *i*th invariable sites parameter:

```
p_inv[i] ~ dnBeta(alpha=1, beta=1)
moves[++mvi] = mvBetaProbability(p_inv[i],
  weight=2.0)
```

The proportion of invariable sites is free to vary across subsets in the partition. Each parameter, `p_inv[i]`, *has a flat distribution,* `dnBeta(alpha=1,beta=1)`, *and an MCMC move to sample posterior values. See step 9 of Basic Protocol 1 for more information about these commands.*

13. Complete the definition of the `for` loop code block:

```
}
```

This completes the `for` *loop that was initialized in step 6 above. The contents of the code block will be executed once for each of the four character subsets.*

Tree topology and branch lengths

14. Create the tree topology variable:

```
out_group = clade("Galeopterus_variegatus")

topology ~ dnUniformTopology(taxa, outgroup=out_group)
```

The model will assume that all sites in the mitochondrial genome share a single gene tree. First, we assign a uniform prior distribution over all possible topologies that could explain the evolutionary relationships shared by `taxa` *with flying lemur set to be the outgroup.*

```
moves[++mvi] = mvNNI(topology, weight=5.0)

moves[++mvi] = mvSPR(topology, weight=3.0)
```

Then, we add two moves to inform MCMC how to explore the space of possible tree topologies: nearest neighbor interchange (`mvNNI`*) and subtree-prune-regraft (*`mvSPR`*). See step 10 of Basic Protocol 1 for more information about these commands.*

15. Create branch length parameters for the tree:

```
for (i in 1:n_branches) {
    bl[i] ~ dnExponential(10.0)
    moves[++mvi] = mvScale(bl[i])
}
sum_br_lens:= sum(bl)
```

Next, we assign a prior distribution over the expected number of substitutions per site per branch. For each branch, `bl[i]`*, we create an exponentially distributed stochastic node and create a scale move (*`mvScale`*) to enable MCMC to mix over the posterior distribution of branch lengths. We also monitor the tree length by creating a deterministic node,* `sum_br_lens`*, whose value always equals* `sum(bl)`*. See step 11 of Basic Protocol 1 for more information about these commands.*

16. Per-subset tree and tree length:

```
psi:= treeAssembly(topology, bl)
```

We instantiate non-clock tree, `psi`*, whose value is determined by the function* `treeAssembly` *by mapping the vector of branch lengths,* `bl` *onto the topology variable,* `topology`*.*

17. Per-subset scaling factor:

```
for (i in 1:n_data_subsets) {
  if (i == 1) {
    part_rates[1] <- 1.0
  } else {
    part_rates[i] ~ dnGamma(2,2)
    moves[++mvi] = mvScale(part_rates[i])
}
TL[i]:= sum_br_lens * part_rates[i]
}
```

We assume that each subset of characters evolves according to its own substitution process, and thus its own substitution rate. The relative difference in rates can be treated as a multiplicative factor. We choose the first subset to have a multiplicative factor of 1 and for the remaining subsets evolve at some rate relative to the first subset. If we choose the prior distribution for the remaining factors to have a mean of one, the expected prior distribution over relative rates will favor equal rates across the partition. If the data support differential substitution rates, however, we expect to see the values of `part_rates` *to deviate from 1.*

To accomplish this, we construct a `for` *loop over the four subsets and assign the constant rate multiplier of* `1.0` *to the first element and the stochastic rate multiplier* `dnGamma(2,2)` *to all other elements. The value* `part_rates[i]` *will later be passed into the phylogenetic substitution process,* `dnPhyloCTMC`*, via the* `branchRates` *argument.*

Putting it all together

18. Model character evolution along the phylogeny:

```
for (i in 1:n_data_subsets) {
```

```
seq[i] ~ dnPhyloCTMC(tree=psi, Q=Q[i],
  branchRates=part_rates[i], siteRates=sr[i],
  pInv=p_inv[i], type="DNA")
seq[i].clamp(data[i])
}
```

Each subset of data in the partitioned analysis evolved according an independent phylogenetic substitution process. When declaring the relationship between the sequence data and their underlying distribution, it is important to recall the model assumptions for this exercise. All data subsets have independent substitution process parameters but share a common phylogeny (topology and branch lengths). Note that `tree=psi` *is the only parameter that does not correspond to an element in a vector (e.g.,* `Q[i]`, `p_inv[i]`). *After creating each* `seq[i]` *variable, we want to condition on the partitioned multiple sequence alignments we input earlier in the protocol. Just as with the single-locus analysis in Basic Protocol 1, we call* `seq[i].clamp(data[i])` *to inform the model that* `seq[i]` *has observed the outcome* `data[i]` *of the evolutionary process defined by* `dnPhyloCTMC(..)` *in the previous line. See steps 12 and 13 of Basic Protocol 1 for more information about these commands.*

19. Instantiate a model object:

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

The full graph of the model parameters is now specified. Calling the `model` *function wraps all the variables in the graph and provides an interface between the graphical model and analysis objects, such as* `Mcmc`. *See step 14 of Basic Protocol 1 for more information about these commands.*

Performing an MCMC analysis

20. Add monitors to store samples from the MCMC simulation into files:

```
monitors[++mni] = mnModel(filename="output/primates_
  partition.log", printgen=10, separator = TAB)

monitors[++mni] = mnFile(filename="output/primates_
  partition.trees", printgen=10, separator = TAB, psi)

monitors[++mni] = mnScreen(printgen=1000, TL)
```

We create three monitors: a model monitor to record the sampled parameter values to file, a file monitor to record the sampled phylogenies to file, and a screen monitor to report the tree length values to the screen. See step 15 of Basic Protocol 1 for more information about these commands.

21. Run a MCMC simulation:

```
mymcmc = mcmc(mymodel, monitors, moves)
mymcmc.burnin(generations=10000,tuningInterval=200)
mymcmc.run(generations=30000)
```

Calling `mcmc(mymodel, monitors, moves)` *creates the* `Mcmc` *analysis object. During burn-in, we tune the efficiency of the MCMC proposals found in* `moves` *for 10000 generations but do not record the MCMC state. After burn-in, we run the MCMC for 30000 generations, updating the state according to the tuned* `moves` *vector and recording the state according to the* `monitors` *vector. See step 16 of Basic Protocol 1 for more information about these commands.*

MODEL COMPARISON USING BAYES FACTORS

For most datasets of molecular sequence alignments, several (possibly many) substitution models of varying complexity are plausible a priori. As a result, we need an objective way to compare different models and quantify the evidence in favor of each one so that we may choose the best model for our data. Choosing the wrong model can have a severe impact on the inferred phylogenetic tree (Posada & Crandall, 2001). In Bayesian statistics, model selection is based on Bayes factors (Jeffreys, 1961; Kass & Raftery, 1995), which provides a method for hypothesis testing and evaluating the support for a given model (for more detailed information on Bayes factors, please see the Bayesian Model Selection section in the Commentary below).

This protocol will describe the steps for comparing two models in `RevBayes`. Specifically, we will investigate the evidence in favor of a Jukes-Cantor (Jukes & Cantor, 1969) substitution model relative to the evidence supporting the Hasegawa-Kishino-Yano (Hasegawa et al., 1985) model of sequence evolution for the primate cytochrome *b* alignment. Computing the Bayes factor requires that one first calculate the marginal likelihood of each candidate model. We demonstrate two approaches, stepping-stone sampling and path sampling, to estimating marginal likelihoods that have been applied in phylogenetics (Lartillot and Philippe, 2006; Fan, Wu, Chen, Kuo, & Lewis, 2011; Xie, Lewis, Fan, Kuo, & Chen, 2011; Baele, Li, Drummond, Suchard, & Lemey, 2013).

Both stepping-stone sampling and path sampling rely on power posteriors to compute the marginal likelihood of a model (Baele et al., 2013). Power posterior analyses are similar to standard MCMC analyses of the posterior distribution, with the difference that for a power posterior distribution the posterior distribution in an MCMC simulation is raised to a power β. All other components of the model and the MCMC algorithm remain unchanged. In practice, one has to run an MCMC simulation for many values of $\beta = [0,1]$, commonly between 30 and 200 different values. Each analysis is considered as a stepping-stone or element of a path from the prior to the posterior. Finally, the marginal likelihood is computed by the stepping-stone and path-sampling formulae, which are both different estimators of the same marginal likelihood using the same power posterior analyses.

Note that in this third protocol, we use simplified models of molecular evolution (e.g., removed the ASRV component of the model) to avoid complexity and to demonstrate the modularity of the graphical-model framework. The key aspect of this protocol is to show a simple, flexible, and generic approach to estimating marginal likelihoods and selecting among any set of models in `RevBayes`.

Necessary Resources

All of the necessary resources for this tutorial are described above in Basic Protocol 1. All data files and analysis scripts are available for download from the `RevBayes` Web site *http://revbayes.com/tutorials. html*.

Getting started

1. Start `RevBayes`:

```
rb
```

See step 1 of Basic Protocol 1 for more information about executing `RevBayes`.

2. Load the sequence data from file:

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

This protocol will describe a simple procedure for comparing the substitution model for one dataset. Therefore, only one alignment is loaded (see step 2 of Basic Protocol 1 for more information about this command).

3. Create the dataset-dimension and helper variables:

```
n_species <- data.ntaxa()

n_branches <- 2 * n_species - 3

taxa <- data.taxa()

n_data_subsets <- data.size()

mvi = 0

mni = 0
```

See steps 3 and 4 of Basic Protocol 1 for more information about these commands.

The Jukes-Cantor model

4. Create the constant node representing the rate matrix under the Jukes-Cantor (JC) substitution model (Jukes and Cantor, 1969):

```
Q <- fnJC(4)
```

The `fnJC` *function creates a Q-matrix where the rate of change between every state is equal. This function takes the number of states, e.g., 4 for nucleotides, as an argument. Importantly, one can use this function to create a rate matrix for characters with k states and equal rates of change between all states. Note that because the rates of change between states are fixed (all equaling 1), this makes the Q-matrix a constant node and no moves are defined for the parameters of the Jukes-Cantor model.*

Tree topology and branch lengths

5. Specify the prior distribution on the tree topology:

```
out_group = clade("Galeopterus_variegatus")
topology ~ dnUniformTopology(taxa, outgroup=out_group)
moves[++mvi] = mvNNI(topology, weight=5.0)
moves[++mvi] = mvSPR(topology, weight=3.0)
```

See step 10 of Basic Protocol 1 for more information about these commands.

6. Define the branch length priors and assemble the tree:

```
for (i in 1:n_branches) {
   bl[i] ~ dnExponential(10.0)
   moves[++mvi] = mvScale(bl[i])
}
TL:= sum(bl)
psi:= treeAssembly(topology, bl)
```

See step 11 of Basic Protocol 1 for more information about these commands.

Putting it all together

7. Specify the model of character evolution along the phylogeny and attach the observed sequence data:

```
seq ~ dnPhyloCTMC(tree=psi, Q=Q, type="DNA")
seq.clamp(data)
```

See steps 12 and 13 of Basic Protocol 1 for more information about these commands.

8. Create the model object:

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

See step 14 of Basic Protocol 1 for more information about these commands.

Compute power posterior distributions

9. Add the monitors to write MCMC samples to file and to the screen:

```
monitors[++mni] = mnModel(filename="output/
  primates_cytb_JC.log", printgen=10,separator=TAB)
monitors[++mni] = mnFile(filename="output/
  primates_cytb_JC.trees", printgen=10,separator=TAB,
  psi)
monitors[++mni] = mnScreen(printgen=1000, TL)
```

This creates files to store the MCMC samples of the model parameters. However, it is important to note that the sampler (specified below) for this analysis is not the same as in the first two protocols. Because we are sampling from power posteriors, the MCMC samples are not valid samples from the true target distribution. Thus, the samples in these files are best for troubleshooting the power-posterior run. See step 15 of Basic Protocol 1 for more information about these commands.

10. Run MCMC under a series of power posteriors:

```
mypowerp = powerPosterior(mymodel, moves, monitors,
  "output/powerp_JC.out", cats=127, sampleFreq=10)
mypowerp.burnin(generations=10000, tuningInterval
  =200)
mypowerp.run(generations=10000)
```

To estimate the marginal likelihood of a given model, we must first run the MCMC under a series of power posteriors. This essentially raises the posterior probability to a power between 1 and 0 in an iterative manner. This method computes a vector of powers from a beta distribution, then executes an MCMC run for each power step while raising the likelihood to that power. In this implementation, the vector of powers starts with 1, sampling the likelihood close to the posterior and incrementally sampling closer and closer to the prior as the power decreases. With the power-posterior samples saved to file, we can use stepping-stone sampling (Xie et al., 2011) or path sampling (Lartillot & Philippe, 2006) to estimate the marginal likelihood under this model (see steps 22 to 25, below).

Evaluate a second model

11. Clear the workspace of the previously defined model (JC model):

```
clear()
```

Unless `RevBayes` *has been restarted, the workspace must be cleared of the previous model.*

12. Re-load the data from file, and instantiate the helper variables:

```
data <- readDiscreteCharacterData("data/primates_
  cytb.nex")
n_species <- data.ntaxa()
n_branches <- 2 * n_species - 3
taxa <- data.taxa()
n_data_subsets <- data.size()
```

```
mvi = 0
mni = 0
```

See steps 2 to 4 of Basic Protocol 1 for more information about these commands.

The Hasegawa-Kishino-Yano (HKY) model

13. Specify a flat Dirichlet prior on the stationary frequencies:

```
sf_prior <- v(1,1,1,1)
sf ~ dnDirichlet(sf_prior)
moves[++mvi] = mvBetaSimplex(sf, weight=3)
```

Like the GTR model specified in step 5 of Basic Protocol 1, the HKY model also assumes that the base frequencies are not equal to one another.

14. Specify log normal prior on the transition-transversion rate ratio:

```
kappa ~ dnLognormal(0, 1)
moves[++mvi] = mvScale(kappa, weight=3)
```

Under the HKY model, transitions—substitutions between two pyrimidines (C ↔ T) or between two purines (A ↔ G)—occur at a different rate compared with transversions—substitutions from a pyrimidine to a purine or vice versa (A ↔ C, A ↔ T, G ↔ C, or G ↔ T). Thus, this model has a parameter called the transition-transversion rate ratio (`kappa`*), which is a measure of the relative rate of transitions to transversions. The commands above specify that* `kappa` *is log-normally distributed with a location parameter (μ) of 0 and a scale parameter (σ) of 1. This corresponds to an expected value of 1.648721, since $\mathbb{E}(kappa) = e^{\mu+\frac{\sigma^2}{2}}$.*

15. Create a deterministic variable for the instantaneous rate matrix:

```
Q:= fnHKY(kappa,sf)
```

Similar to the specification of the GTR model in step 7 of Basic Protocol 1, the instantaneous rate matrix of the HKY model is a deterministic node in the graphical model. This node is created using the `fnHKY` *function computed via the base frequencies and transition-transversion rate ratio.*

Tree topology and branch lengths

16. Specify the uniform prior distribution on the tree topology:

```
out_group = clade("Galeopterus_variegatus")
topology ~ dnUniformTopology(taxa, outgroup=out_group)
moves[++mvi] = mvNNI(topology, weight=5.0)
moves[++mvi] = mvSPR(topology, weight=3.0)
```

See step 10 of Basic Protocol 1 for more information about these commands.

17. Set up the stochastic nodes representing branch lengths and assemble the tree in a deterministic node:

```
for (i in 1:n_branches) {
  bl[i] ~ dnExponential(10.0)
  moves[++mvi] = mvScale(bl[i])
}
TL:= sum(bl)
psi:= treeAssembly(topology, bl)
```

See step 11 of Basic Protocol 1 for more information about these commands.

Putting it all together

18. Specify the model of character evolution along the phylogeny and attach the observed sequence data:

```
seq ~ dnPhyloCTMC(tree=psi, Q=Q, type="DNA")
seq.clamp(data)
```

See steps 12 and 13 of Basic Protocol 1 for more information about these commands.

19. Instantiate the model object:

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

See step 14 of Basic Protocol 1 for more information about this command.

Compute power posterior distributions

20. Create a vector of monitors that output the MCMC samples to file and to screen:

```
monitors[++mni] = mnModel(filename="output/
  primates_cytb_HKY.log", printgen=10, separator=TAB)
monitors[++mni] = mnFile(filename="output/
  primates_cytb_HKY.trees", printgen=10,
  separator=TAB, phylogeny)
monitors[++mni] = mnScreen(printgen=1000, TL)
```

See step 9, above (in this protocol), for more information about these commands.

21. Run MCMC under a series of power posteriors:

```
mypowerp = powerPosterior(mymodel, moves, monitors,
  "output/powerp_HKY.out", cats=127, sampleFreq=10)
mypowerp.burnin(generations=10000,tuningInterval=200)
mypowerp.run(generations=10000)
```

See step 10, above (in this protocol), for more information about these commands.

Estimate marginal likelihoods under the Jukes-Cantor model

22. Use stepping-stone sampling to calculate marginal likelihoods from the output of the `powerPosterior` function:

```
ss_JC = steppingStoneSampler(file="output/
  powerp_JC.out", powerColumnName=likelihoodColumn
  Name="likelihood")
```

The `steppingStoneSampler` *function reads the output file produced by the* `powerPosterior` *function and computes the marginal likelihood using stepping-stone sampling. The command above assigns the sampler to a variable called* `ss_JC` *and reads in the power-posterior file saved under the JC model.*

23. Assign the stepping-stone estimate of the marginal likelihood to a variable in the workspace:

```
ssmlnl_JC = ss_JC.marginal()
```

A stepping-stone sampler object has a member function called `marginal` *that returns the marginal likelihood computed from the power-posterior using the stepping-stone approach (Fan et al., 2011; Xie et al., 2011). In the* `Rev` *code above, the value is assigned to the variable called* `ssmlnl_JC`.

24. Use path sampling to calculate marginal likelihoods from the output of the `powerPosterior` function:

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

Path sampling (also called thermodynamic integration) is an alternative approach to computing the marginal likelihood from a series of power posteriors (Lartillot and Philippe, 2006; Baele, Lemey, Bedford, Rambaut, Suchard, & Alekseyenko, 2012). Like the stepping-stone sampler above, the `pathSampler` *function also reads in the power-posterior output file and can be assigned to a workspace variable.*

25. Assign the path-sampling estimate of the marginal likelihood to a workspace variable:

```
psmlnl_JC = ps_JC.marginal()
```

Similar to the stepping-stone approach, we can assign the marginal likelihood computed by path sampling to a workspace variable.

Estimate marginal likelihoods under the HKY model

26. Use stepping-stone sampling to calculate marginal likelihoods:

```
ss_HKY =
  steppingStoneSampler(file="output/powerp_HKY.out",
  powerColumnName="power",
  likelihoodColumnName="likelihood")
ssmlnl_HKY = ss_HKY.marginal()
```

Like steps 24 and 25 above, performed for the JC model, a stepping-stone sampler is created for the HKY model.

27. Use path sampling to calculate marginal likelihoods:

```
ps_HKY = pathSampler(file="output/powerp_HKY.out",
  powerColumnName="power",
  likelihoodColumnName="likelihood")
psmlnl_HKY = ps_HKY.marginal()
```

Like steps 22 and 23 above, performed for the JC model, a path sampler is created for the HKY model.

Compute Bayes factors

28. Compute the ln-Bayes factor in favor of the JC model using stepping-stone sampling:

```
ssmlnl_JC - ssmlnl_HKY
```

To compute the Bayes factor, simply calculate the difference in marginal likelihoods between the two models under a given type of sampler. This procedure is described further below, and the difference is equal to K, which is defined in Equation 6.16.2 (see Guidelines for Understanding Results). The commands above will print the value of K to the screen, which is the support in favor of the JC model relative to the HKY model (with marginal likelihoods estimated under stepping-stone sampling; Fan et al., 2011; Xie et al., 2011).

29. Compute the ln-Bayes factor in favor of the JC model using path sampling:

```
psmlnl_JC - psmlnl_HKY
```

The commands above will print the value of K to the screen, which is the support in favor of the JC model relative to the HKY model (with marginal likelihoods estimated under path sampling; Lartillot and Philippe, 2006; Baele et al., 2012).

30. Refer to Guidelines for Understanding Results for information about how to interpret ln-Bayes factors.

Evaluate the GTR model

31. Estimate the marginal likelihood under the GTR model and evaluate the support under the GTR relative to the JC and HKY models using Bayes factors.

    The steps outlined in this protocol and the model specification in Basic Protocol 1 provide all the necessary commands needed for one to estimate the marginal likelihood under GTR for this dataset. Then, pair-wise comparisons using Bayes factors will enable model selection among the JC, HKY, and GTR models.

GUIDELINES FOR UNDERSTANDING RESULTS

The goal of a Markov chain Monte Carlo analysis is to generate samples from a target distribution. In a Bayesian phylogenetic analysis, the target distribution is the joint posterior distribution, $\mathbb{P}(\theta|\mathbf{X}, M)$, where θ includes all the estimated model parameters, including the tree topology, branch lengths, exchangeability rates, stationary frequencies, etc. Thus, we use MCMC simulation to approximate the posterior distribution, and to find a range of parameters with high posterior probability density (i.e., the 95% credible interval). In Basic Protocol 1 and Basic Protocol 2, the posterior samples are separated into two files. The `.trees` file contains the sampled posterior distribution of phylogenies (topologies and branch lengths) stored in Newick format. The `.log` file contains a tab-delimited "trace" of each model parameter, with one parameter per column. In both cases, each row corresponds to the MCMC state when sampled by the corresponding `RevBayes` monitor.

Summarizing the Posterior Distribution of Phylogenies

The primary goal of many phylogenetic analyses is to produce a point estimate of the phylogeny, including its topology and branch lengths. We will compute the maximum a posteriori (MAP) phylogeny in `RevBayes`. First, we read in the posterior sample of non-clock phylogenies.

```
treetrace = readTreeTrace("output/primates_
  cytb.trees", treetype="non-clock")
```

Next, we compute the MAP tree using the `mapTree` function.

```
mapTree(treetrace,"output/primates_partition_MAP.
  tre")
```

The function first finds the topology with the highest posterior probability. Given that topology, the `mapTree` then uses the mean posterior branch length distribution to provide a smooth estimate the MAP tree's branch lengths. Finally, `mapTree` converts the MAP tree to a Newick string, annotates it with useful quantities, such as the posterior probabilities of clade support, then saves the Newick string to file. Figure 6.16.4 corresponds to the MAP tree estimated in Basic Protocol 1 when viewed in the tree-visualization program FigTree (*http://tree.bio.ed.ac.uk/software/figtree*).

This analysis of the cytochrome *b* sequence data of primates shows some interesting results. There has been considerable disagreement about the placement tarsiers (genus *Tarsius*) in the primate phylogeny (Yoder, 2003; Chatterjee, Ho, Barnes, & Groves, 2009; Hartig, Churakov, Warren, Brosius, Makałowski, & Schmitz, 2013). In previous studies, most support is given to a *Tarsius*-Haplorhini sister relationship (Haplorhini includes New World monkeys, Old World monkeys, and anthropoid primates), although several molecular studies have found conflicting results. The sequence of our *Tarsius* representative is placed as a sister lineage to all Strepsirrhini (Strepsirrhini includes

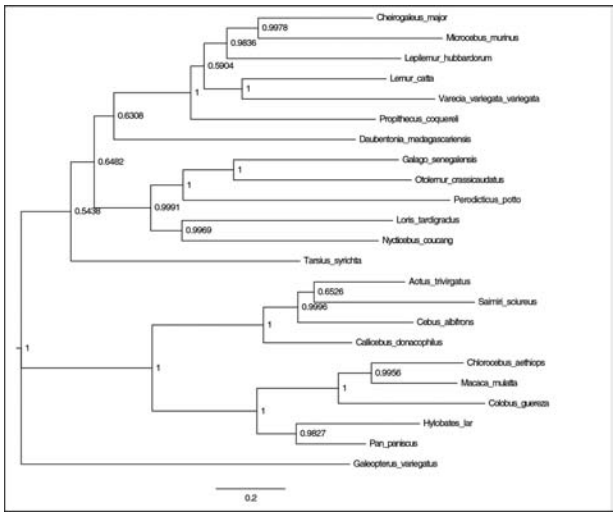


Figure 6.16.4 MAP tree from Basic Protocol 1 viewed in `FigTree`. The branch lengths are in expected number of substitutions and the scale is given at the bottom. Posterior probabilities supporting clades are reported at internal nodes.

lemurs, lorises, and bushbabies). However, the support for this *Tarsius*-Strepsirrhini sister relationship is comparably weak, with a posterior probability of 0.5438, albeit being the most probable evolutionary relationship given our model and data. Importantly, we want to stress that this result is based on a very small dataset intended for this exercise. Nevertheless, this result exemplifies the type of topological questions one can readily answer from our analysis.

There are alternative ways to summarize the posterior distribution of phylogenies then using the MAP. For example, other approaches to combining the posterior distribution into a single point estimate are consensus tree methods (Holder, Sukumaran, & Lewis, 2008; Heled & Bouckaert, 2013). The output of `RevBayes` can be summarized with consensus tree methods implemented in other software, such as DendroPy (Sukumaran & Holder, 2010).

Posterior Estimates for Standard Parameters

The `mnModel` monitor in `RevBayes` will discover all non-constant nodes in the graphical model and save their values to a tab-delimited file, known as a trace file. Each row corresponds to an iteration when the MCMC state was sampled, and each column corresponds to a particular variable in the MCMC state space, such as the shape parameter for among site rate variation.

We will use Tracer (*http://tree.bio.ed.ac.uk/software/tracer/*) to validate the analysis ran correctly.

First, we will review the effective sample size (ESS) for parameter estimates. Parameters with large ESS values generally indicate that the MCMC gathered enough samples to accurately estimate the parameter's marginal posterior density. If the ESS is low for a given parameter (indicated in red), the MCMC may need to be run for more generations or the move responsible for sampling the parameter in question may need to be assigned a greater `weight`.

Second, we will compare the posterior distribution between two independent runs to assess convergence. If the posterior samples do not appear equivalent, then one (or both) MCMC analyses failed to produce valid samples from the same posterior distribution.

Note that even if both posterior samples appear equivalent, it is still possible that neither MCMC reached convergence. As a rule of thumb, it is easier to show that an MCMC run has failed than it is to show it succeeded. The visual test described here is not rigorous, but adequate to rule out gross MCMC failures. More sophisticated tests are available in the R package (R Core Team, 2013) coda (Plummer, Best, Cowles, & Vines, 2006).

Two independent runs may be accomplished by adding the argument `nruns=2` in step 16 of Basic Protocol 1.

```
mymcmc = mcmc(mymodel, monitors, moves, nruns=2)
```

Third, assuming that the posterior sample appears valid, we will compare it to the prior distribution. The posterior distribution is proportional to the prior distribution times the likelihood function. This means that posterior distribution and prior distribution differ when the likelihood function is informative—i.e., when the parameter estimates are informed by data. We can easily sample from the prior distribution under any model by setting the `underPrior=true` flag in the `mcmc.run` method in `RevBayes`.

To record an estimate of the joint prior distribution under the model, first change the names of the output files in step 15 of Basic Protocol 1:

```
monitors[++mni] = mnModel(filename="output/primates_
  cytb.prior.log", printgen=10, separator = TAB)
monitors[++mni]
mnFile(filename="output/primates_cytb.prior.trees",
  printgen=10, separator = TAB, psi)
```

Then add the `underPrior=true` argument to the following commands in step 16 of Basic Protocol 1:

```
mymcmc.burnin(generations=10000,tuningInterval=200,
  underPrior=true)
mymcmc.run(generations=30000, underPrior=true)
```

Posterior estimates that look very different from prior estimates tend to be strong results. However, when posterior and prior estimates appear to be similar, it often means the data are not informative enough to pull the posterior distribution away from the prior. If the prior and posterior looked identical, one would expect to get a similar parameter estimate even if no data were used! This may motivate collecting more data, re-designing the model to ensure all parameters are identifiable, or considering alternative priors.

Viewing the two independent posterior estimates and the prior estimate in Tracer, it is likely that the `alpha` shape parameter for among-site rate variation was adequately sampled, was estimated with valid samples from the posterior, and does not exhibit strong prior sensitivity (Fig. 6.16.5).

Interpreting Marginal Likelihoods and Bayes Factors

In Basic Protocol 3, we shifted our focus from parameter estimation to model selection. The models' relative fit to the datasets is determined using Bayes factors which are computed by the ratio of marginal likelihoods. Phylogenetic programs log-transform the likelihood values to avoid underflow—multiplying likelihoods (numbers < 1) generates numbers that are too small to be held in computer memory. Accordingly, we need to

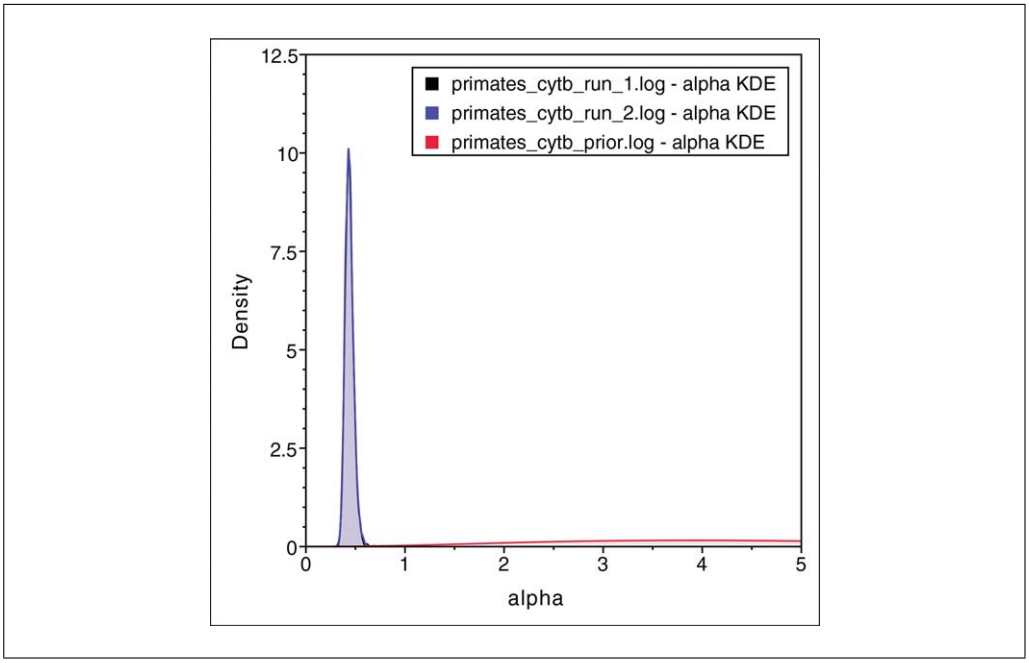


Figure 6.16.5 Posterior and prior estimates for from Basic Protocol 1 viewed in Tracer. The highlight parameter, `alpha`, controls the shape of the discrete Gamma distribution used to model among site rate variation. The peaked black and blue densities correspond to independent posterior densities. The flat red density is the prior density. The posterior estimate of `alpha` appears to be supported by a strong signal in the data and to be estimated consistently between the replicates.

account for this log-transformation in our calculation of the Bayes factor (we will denote this value $\mathcal{K}$):

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

Equation 6.16.1

where $\ln[\mathbb{P}(X|M_0)]$ is the marginal lnL estimate for model M_0. The value resulting from Equation 6.16.1 can be converted to a raw Bayes factor by simply taking the exponent of $\mathcal{K}$ (for more on the raw Bayes factor see Bayesian Model Selection in the Commentary section below)

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

Equation 6.16.2

Alternatively, you can directly interpret the strength of evidence in favor of M_0 in log space by comparing the values of $\mathcal{K}$ to the appropriate scale (Table 6.16.1, second column). In this case, we evaluate $\mathcal{K}$ in favor of model M_0 against model M_1 so that:

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

Thus, values of $\mathcal{K}$ around 0 indicate that there is no preference for either model. Variations on the Bayes factor and further background are provided in the Commentary section below.

Table 6.16.1 The Scale for Interpreting Bayes Factors by Harold Jeffreys (1961)

Strength of evidence	$BF(M0,M1)$	$log(BF(M0,M1))$	$log_{10}(BF(M0,M1))$
Negative (supports M_1)	<1	<0	<0
Barely worth mentioning	1 to 3.2	0 to 1.16	0 to 0.5
Substantial	3.2 to 10	1.16 to 2.3	0.5 to 1
Strong	10 to 100	2.3 to 4.6	1 to 2
Decisive	>100	>4.6	>2

For a detailed description of Bayes factors see Kass and Raftery (1995)

In Basic Protocol 3, you will find that the values of K computed in steps 28 and 29 indicate that the data support the HKY model of substitution. It is important to note, however, that these two models are just a subset of the possible substitution models available for describing the evolution of nucleotide data. `RevBayes` provides a straightforward approach to comparing models (as outlined above), allowing users to evaluate the range of models selected for their analysis.

COMMENTARY

Background Information

Bayesian inference

Bayesian statistics is used to estimate the posterior probability distribution of the parameters of interest given the data following Bayes' theorem:

$$\mathbb{P}(\theta|\mathbf{X}) = \frac{\mathbb{P}[\mathbf{X}|\theta] \times \mathbb{P}(\theta)}{\mathbb{P}(\mathbf{X})},$$

Equation 6.16.3

where the parameters include, among others, the tree, branch lengths, and substitution model parameters. In Bayesian statistics, all parameters are considered as random variables. Accordingly, every parameter is associated with a prior probability distribution. The prior probability distribution represents the probabilities of the parameters before seeing the data. These probabilities are updated by means of the likelihood function when observing data. The result is the posterior probability of the parameters after seeing the data.

The main focus of a Bayesian statistical analysis is the mean, mode, or maximum a posteriori (MAP) parameter estimate. For numerical parameters, such as branch lengths, the mean estimate from the posterior distribution is often the quantity of interest. For discrete or categorical parameters, such as the phylogeny, the MAP estimate is preferred. The MAP estimate of the posterior distribution of phylogenies is interpreted as the most probable tree to have generated the observed data. The posterior probability corresponds to the probability of the tree being the true tree (assuming the underlying model to be the true model). Bayesian inference has two additional strengths: (1) Bayesian analyses naturally integrate over nuisance parameters (i.e., accounts for uncertainty in the substitution rates if the phylogeny is parameter of interest); and (2) credible intervals in Bayesian analyses directly provide a measure of uncertainty in the estimated parameters and are straightforward in their interpretation.

In most cases, such as phylogenetic inference, computing the posterior probabilities analytically is not possible. Instead, we resort to numerical integration procedures such as Markov chain Monte Carlo (MCMC) to approximate the posterior probabilities (Rannala and Yang, 1996; Yang and Rannala, 1997; Mau, Newton, & Larget, 1999; Li, Pearl, & Doss, 2000).

Probabilistic graphical models

Graphical models symbolically depict probabilistic models as graphs (Koller & Friedman, 2009). This is accomplished by representing all model variables as nodes and the relationships between variables as edges. The models in Figures 6.16.1 and 6.16.3 provide a visual representation of the structure of a statistical model, including the prior probability distributions (Höhna et al., 2014). In `RevBayes`, models are specified using directed acyclic graphs (DAGs): graphs where all edges denote parent-child relationships between nodes, and, when following a path of

consecutive directed edges, the graph is devoid of any loops or cycles. In a probabilistic model, a child node is a variable whose value or probability depends on the value of its immediate parent node(s). There is a natural relationship between adding and connecting variables in a graphical model, and doing the same in a programming language. Figure 6.16.2 provides a glossary for the basic components of a graphical model and how they are instantiated in `Rev`. The parameters associated with prior distributions are shown as solid circles (Höhna et al., 2014). These parameters contribute to the joint prior probability density. Arrows between the parameters show the dependency between them. The variables associated with data are shown in shaded solid circles and contribute to the joint likelihood.

When designing a new model, graphical models provide valuable clues for how one might simplify the model by reducing the number of variables (nodes) or dependencies (edges). Moreover, due to the correspondence between asserting parent-child relationships in the graphical model and in `Rev` code, drawing a graphical model before programming it yields code that is easier to read, maintain, and modify. From the computational side, graphical models enable efficient computation of model probabilities with MCMC, simulation under generative models, and marginalization algorithms. Höhna et al., (2014) provide a more detailed discussion on the use of graphical models in phylogenetics.

Markov chain Monte Carlo (MCMC) simulation

The MCMC algorithm simulates samples of parameter values from the posterior distribution. The samples provide information about the posterior mean and credible interval of a parameter estimate. In `RevBayes`, we have implemented a variant of the Metropolis-Hastings algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953; Hastings, 1970). Specifically, our implementation works as follows:

1. Select the next move from the vector of provided moves.
2. Apply the selected move which possibly modifies one or several parameter values.
3. Repeat steps 1 and 2 k times.
4. Sample (i.e., store to a file) the parameter values if the current iteration is divisible by the thinning interval `(printgen)`.
5. Repeat steps 1 to 4 until the maximum number of iterations has been reached.

There are several important details about this algorithm. First, the next move is either selected *randomly* or *sequentially.* If the moves are selected randomly, then a move is selected randomly in proportion to its weight. Random selection of moves has, in general, the best performance, although sequential moves are useful for consecutive updates of dependent or correlated parameters. Second, we apply k moves per iteration, where k is equal to the sum of weights of all moves. This has the advantage that models with many parameters still require the same number of iterations but perform more moves per iteration (similar to the MCMC procedure in PhyloBayes; Lartillot et al., 2009). Nevertheless, we provide the option to run `RevBayes` with a single, randomly chosen move per iteration with the option `moveschedule=single`. Third, the composition of moves is unrestricted to the type of algorithm used (e.g., Gibbs sampling or Metropolis-Hasting sampling). In principle, all moves follow the following algorithm:

1. Propose a new parameter value θ' drawn from the proposal distribution.
2. Compute the acceptance probability

$$\alpha = \frac{\mathbb{P}(\mathbf{X}|\theta')}{\mathbb{P}(\mathbf{X}|\theta)} \times \frac{\mathbb{P}(\theta')}{\mathbb{P}(\theta)} \times \frac{\mathbb{P}(\theta|\theta')}{\mathbb{P}(\theta'|\theta)}$$

   where $\mathbb{P}(\theta'|\theta)$ is the forward proposal probability (i.e., proposing θ' given the current value θ) and $\mathbb{P}(\theta|\theta')$ is the backward proposal probability (i.e., proposing θ given the current value would be θ').
3. Set the parameter value to θ' with probability α and keep the current parameter value θ with probability $1 - \alpha$.

These moves include, for example, the sliding-window move, which proposes a new parameter value θ' drawn from a normal distribution with mean θ (the current value of the parameter) and a standard deviation of δ ($\theta' \sim$Normal (θ, δ)). δ is a tuning parameter of the sliding-window move. If δ is large, then the new proposed parameter values are more different than if δ is small. Good values of δ result in 20% to 50% accepted moves. Moves that have tuning parameters are auto-tuned during the burn-in phase of the MCMC [see the `.burnin()` function; Haario et al., 1999).

Furthermore, `RevBayes` implements a Metropolis-Coupled MCMC (MC3) sampler (Altekar, Dwarkadas, Huelsenbeck, & Ronquist, 2004) which is used by the `mcmcmc`

function instead of the `mcmc` function (see step 16 of Basic Protocol 1). The MC3 sampler employs one cold and several heated chains. The heated chains can explore the parameter space more easily, and thus are considered to improve mixing and convergence of the analysis. However, MC3 comes with a higher computational burden because it requires more CPU resources to run several chains.

Bayesian model selection

Given two models, M_0 and M_1, the Bayes-factor comparison assessing the relative fit of each model to the data, $BF(M_0, M_1)$, is:

$$BF(M_0, M_1) = \frac{\text{posterior odds}}{\text{prior odds}}.$$

The posterior odds is the posterior probability of M_0 given the data, $\mathbf{X}$, divided by the posterior odds of M_1 given the data:

$$\text{posterior odds} = \frac{\mathbb{P}(M_0|\mathbf{X})}{\mathbb{P}(M_1|\mathbf{X})},$$

and the prior odds is the prior probability of M_0 divided by the prior probability of M_1:

$$\text{prior odds} = \frac{\mathbb{P}(M_0)}{\mathbb{P}(M_1)}.$$

Thus, the Bayes factor measures the degree to which the data alter our belief regarding the support for M_0 relative to M_1 (Lavine & Schervish, 1999):

$$BF(M_0,\ M_1) = \frac{\mathbb{P}(M_0|\mathbf{X},\ \theta_0)}{\mathbb{P}(M_1|\mathbf{X}, \theta_1)} \div \frac{\mathbb{P}(M_0)}{\mathbb{P}(M_1)}.$$

Equation 6.16.4

Note that interpreting Bayes factors involves some subjectivity That is, it is up to *you* to decide the degree of your belief in M_0 relative to M_1. Despite the absence of an absolutely objective model-selection threshold, we can refer to the scale (outlined by Jeffreys, 1961) that provides a "rule-of-thumb" for interpreting these measures (Table 6.16.1).

Unfortunately, it is generally not possible to directly calculate the posterior odds to prior odds ratio. However, we can further define the posterior odds ratio as:

$$\frac{\mathbb{P}(M_0|\mathbf{X})}{\mathbb{P}(M_1|\mathbf{X})} = \frac{\mathbb{P}(M_0)}{\mathbb{P}(M_1)} \frac{\mathbb{P}(\mathbf{X}|M_0)}{\mathbb{P}(\mathbf{X}|M_1)},$$

where $P(\mathbf{X} \mid M_i)$ is the *marginal likelihood* of the data (this may be familiar to you as the denominator of Bayes Theorem—Equation 6.16.3—which is variously referred to as the *model evidence* or *integrated likelihood*). Formally, the marginal likelihood is the probability of the observed data ($\mathbf{X}$) under a given model (M_i) that is averaged over all possible values of the parameters of the model (θ_i) with respect to the prior density on θ_i:

$$\mathbb{P}(\mathbf{X}|M_i) = \int \mathbb{P}(\mathbf{X}|\theta_i)\mathbb{P}(\theta_i)\, d\theta.$$

Equation 6.16.5

This makes it clear that more complex (parameter-rich) models are penalized by virtue of the associated prior: each additional parameter entails integration of the likelihood over the corresponding prior density.

If you refer back to Equation 6.16.4, you can see that, with very little algebra, the ratio of marginal likelihoods is equal to the Bayes factor:

$$\begin{aligned} BF(M_0,\ M_1) &= \frac{\mathbb{P}(\mathbf{X}|M_0)}{\mathbb{P}(\mathbf{X}|M_1)} \\ &= \frac{\mathbb{P}(M_0|\mathbf{X},\ \theta_0)}{\mathbb{P}(M_1|\mathbf{X},\ \theta_1)} \div \frac{\mathbb{P}(M_0)}{\mathbb{P}(M_1)}. \end{aligned}$$

Equation 6.16.6

Therefore, we can perform a Bayes factor comparison of two models by calculating the marginal likelihood for each one. Alas, exact solutions for calculating marginal likelihoods are not known for phylogenetic models (see Equation 6.16.5); thus, we must resort to numerical integration methods to estimate or approximate these values. In Basic Protocol 3, we estimated the marginal likelihood for each partition scheme using both the stepping-stone (Fan et al., 2011; Xie et al., 2011) and path sampling estimators (Lartillot and Philippe, 2006; Baele et al., 2012).

Phylogenetic Models and Theory

In the previous section we explained Bayesian inference and MCMC simulation to estimate posterior probabilities. At the heart of a Bayesian analysis lies the statistical model and likelihood function. In phylogenetics, the model for character evolution is a continuous time Markov model, and the likelihood function is computed using the Felsenstein's pruning algorithm (Felsenstein, 1981). Here we briefly outline the main elements of the theory behind phylogenetic substitution models and refer the readers to the cited primary literature.

Substitution model

The substitution models used in molecular evolution are continuous time Markov models, which are each fully characterized by their instantaneous-rate matrix:

$$Q_{GTR} = \begin{pmatrix} \cdot & er_{AC} \times pi_C & er_{AG} \times pi_G & r_{AT} \times pi_T \\ er_{AC} \times pi_A & \cdot & er_{CG} \times pi_G & er_{CT} \times pi_T \\ er_{AC} \times pi_A & er_{CG} \times pi_C & \cdot & er_{GT} \times pi_T \\ er_{AC} \times pi_A & er_{CT} \times pi_C & er_{GT} \times pi_G & \cdot \end{pmatrix}.$$

$$Q = \begin{pmatrix} -\mu_A & \mu_{GA} & \mu_{CA} & \mu_{TA} \\ \mu_{AG} & -\mu_G & \mu_{CG} & \mu_{TG} \\ \mu_{AC} & \mu_{GC} & -\mu_C & \mu_{TC} \\ \mu_{AT} & \mu_{GT} & \mu_{CT} & -\mu_T \end{pmatrix},$$

where μ_{ij} represents the instantaneous rate of substitution from state *i* to state *j*. Given the instantaneous-rate matrix, *Q,* we can compute the corresponding transition probabilities for a branch of length *t, P(t),* by exponentiating the rate matrix:

$$P(t) = \begin{pmatrix} p_{AA}(t) & p_{GA}(t) & p_{CA}(t) & p_{TA}(t) \\ p_{AG}(t) & p_{GG}(t) & p_{CG}(t) & p_{TG}(t) \\ p_{AC}(t) & p_{GC}(t) & p_{CC}(t) & p_{TC}(t) \\ p_{AT}(t) & p_{GT}(t) & p_{CT}(t) & p_{TT}(t) \end{pmatrix}$$

$$= e^{Qt} = \sum_{j=0}^{\infty} \frac{tQ^j}{j!}.$$

Each specific substitution model has a uniquely defined instantaneous-rate matrix, *Q*. In `RevBayes`, we provide all standard substitution models: the Jukes-Cantor substitution model (`fnJC`; Jukes & Cantor, 1969), the Kimura 2-parameter model (`fnK80`; Kimura, 1980), the Kimura 3-parameter model (`fnK81`; Kimura, 1981), the Felsenstein, 1981 model with unequal stationary frequencies (`fnF81`; Felsenstein, 1981), the Hasegawa-Kishino-Yano substitution model with unequal stationary frequencies and different transition-transversion rate (`fnHKY`; Hasegawa et al., 1985), the Tamura-Nei substitution model(`fnTrN`; Tamura & Nei, 1993), the transversion model with unequal stationary frequencies and variable transversion rates (`fnTVM`), the transition model with unequal stationary frequencies, variable transition rates, and two transversion rates (`fnTIM`), and the general time reversible substitution model (`fnGTR`; Tavaré, 1986). For example, the instantaneous-rate matrix for the general-time reversible (GTR) substitution model is computed by:

Additionally, we provide a general symmetric substitution model [`fnFreeSymmetricRateMatrix`] and a completely unrestricted substitution model [`fnFreeK`]. Any of these substitution models can be used interchangeably in the continuous time Markov process (`dnPhyloCTMC`) to compute the probability of observing the sequence data.

Critical Parameters and Troubleshooting

Model specification in `RevBayes` depends on variables being created and associated with one another. This is achieved by issuing `Rev` commands with the correct syntax and order. It is common to make minor mistakes when initially designing a model through the interactive console or executing new scripts. When `RevBayes` encounters a mistake, it emits an error message to help the user find and repair the issue. For example, an error reporting:

```
Missing Variable: Variable
  num_species does not exist
Error: Problem processing line
  18 in file
"analysis_files/RevBayes_
  scripts/mcmc_G TR_Gamma
  _Inv.Rev"
```

indicates that `RevBayes` could not execute line 18 of the corresponding script because the variable num_species was not found in the workspace. By examining line 17 of the reported file, we might see the variable is in fact named `n_species`, not `num_species`.

An equally important—but subtler—problem is determining whether the correct object, function, or distribution is behaving as the user intends. Like most programming languages, `RevBayes` will

obediently execute the code exactly as it appears, and it is the user's duty to ensure the code is correct. There are several options if a RevBayes command is not behaving as expected. Help files are available within the console via the ? command, e.g., ?dnNormal will print help files for the normal distribution. The same help files are also available online (*https://cdn.rawgit.com/revbayes/revbayes/master/help_html/index.html*). The command in question might be used in an example published in the online tutorials (*http://revbayes.com/tutorials.html*), where it may be seen in action. Additional help is available online by posting your question to the RevBayes forums (*https://groups.google.com/forum/#!forum/revbayes-users*).

Diagnosing possible MCMC convergence and prior sensitivity issues are briefly discussed in Guidelines for Understanding Results, above.

Suggestions for Further Analysis

The set of protocols outlined here are just a small subset of the analyses possible in RevBayes. The graphical model framework implemented in RevBayes provides a flexible toolkit for assembling a wide range of statistical models (Höhna et al., 2014; Höhna et al., 2016). For example, the analyses described above focus only on unrooted trees with branch lengths in units of the number of substitutions per site. Studies of evolutionary biology are often motivated by questions seeking to understand lineage relationships on an absolute time scale, where the branch lengths are in units of years or millions of years. These analyses, often called 'divergence-time estimation' or 'relaxed-clock analyses', are also possible in RevBayes. To construct the graphical model, one would replace the distributions describing the tree topology [e.g., topology ~ dnUniformTopology()] and branch lengths [e.g., bl[i] ~ dnExponential(10.0)] and the deterministic function uniting them [e.g., psi:= treeAssembly(topology, bl)] with a model that generates rooted time trees (e.g., the birth-death process; Kendall, 1948; Nee, May, & Harvey, 1994) and a model that describes how substitution rates vary across the tree (e.g., autocorrelated, log-normal rates; Thorne, Kishino, & Painter, 1998). For robust estimates of absolute branching times with data from the fossil record, one can replace the general birth-death process (Kendall, 1948) with the 'fossilized birth-death' process (Stadler, 2010; Heath, Huelsenbeck, & Stadler, 2014). Tutorials describing these types of analyses are provided in the Divergence Time Estimation section of the RevBayes tutorial Web page (*http://revbayes.com/tutorials.html*). The flexibility afforded by probabilistic graphical models makes for a vast array of possible model combinations, and therefore the possible questions and analyses are tremendous.

The design of RevBayes enables researchers to conduct statistical phylogenetic analyses under increasingly complex—and biologically realistic—models of evolution in a fully integrative Bayesian framework. The RevBayes documentation includes several tutorials describing the procedures for performing a number of different phylogenetic analyses. These tutorials can be found on the RevBayes Web page (*http://revbayes.com/tutorials.html*) and are applicable to evolutionary biology studies investigating historical biogeography, phylogeography, continuous and discrete traits, evolution of gene trees within species trees, diversification of lineages in the fossil record, variation in rates of molecular evolution across sites, evolution of continuous traits, and many others. Moreover, the engineering, philosophy, and open-source license of RevBayes and the Rev language empower researchers to develop and implement phylogenetic models and methods that have not yet been discovered or described. Thus, flexible and open software for phylogenetic analysis will lead to many exciting advances in our understanding of evolutionary biology.

Acknowledgements

We thank William Pearson for the invitation to write this protocol. RevBayes, the Rev language, and the software documentation are developed collaboratively by many contributors (*https://github.com/revbayes/revbayes/graphs/contributors*). We especially thank the other members of the core development team: Bastien Boussau, John Huelsenbeck, Nicolas Lartillot, and Fredrik Ronquist. Additionally, we thank Brian Moore for contributions to RevBayes tutorials from which the presented protocols are derived. Generous funding from various sources supported this work: S.H. was funded by the Miller Institute for Basic Research in Science; M.J.L. was supported by the Yale Institute of Biospheric Studies under the Gaylord Donnelley Postdoctoral Environmental Fellowship; and T.A.H. was funded by research grants

from the U.S. National Science Foundation (DEB-1556853 and DEB-1556615).

Literature Cited

R Core Team. (2013). R: A Language and Environment for Statistical Computing. Vienna: R Foundation for Statistical Computing. Available at *http://www.R-project.org/*.

Altekar, G., Dwarkadas, S., Huelsenbeck, J. P., & Ronquist, F. (2004). Parallel metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics*, 20, 407–415. doi: 10.1093/bioinformatics/btg427.

Baele, G., Li, W., Drummond, A., Suchard, M., & Lemey, P. (2013). Accurate model selection of relaxed molecular clocks in Bayesian phylogenetics. *Molecular Biology and Evolution*, 30, 239–243. doi: 10.1093/molbev/mss243.

Baele, G., Lemey, P., Bedford, T., Rambaut, A., Suchard, M., & Alekseyenko, A. (2012). Improving the accuracy of demographic and molecular clock model comparison while accommodating phylogenetic uncertainty. *Molecular Biology and Evolution*, 29, 2157–2167. doi: 10.1093/molbev/mss084.

Bouckaert, R., Heled, J., Kühnert, D., Vaughan, T., Wu, C.-H., Xie, D., Suchard, M. A., Rambaut, A., & Drummond, A. J. (2014). BEAST 2: A software platform for Bayesian evolutionary analysis. *PLoS Computational Biology*, 10, e1003537. doi: 10.1371/journal.pcbi.1003537.

Brandley, M. C., Schmitz, A., & Reeder, T. W. (2005). Partitioned bayesian analyses, partition choice, and the phylogenetic relationships of scincid lizards. *Systematic Biology*, 54, 373–390. doi: 10.1080/10635150590946808.

Brown, J. M., & Lemmon, A. R. (2007). The importance of data partitioning and the utility of Bayes factors in Bayesian phylogenetics. *Systematic Biology*, 56, 643–655. doi: 10.1080/10635150701546249.

Brown, J. M., Hedtke, S. M., Lemmon, A. R., & Lemmon, E. M. (2010). When trees grow too long: Investigating the causes of highly inaccurate Bayesian branch-length estimates. *Systematic Biology*, 59, 145–161. doi: 10.1093/sysbio/syp081.

Bull, J., Huelsenbeck, J. P., Cunningham, C. W., Swofford, D. L., & Waddell, P. J. (1993). Partitioning and combining data in phylogenetic analysis. *Systematic Biology*, 42, 384–397. doi: 10.1093/sysbio/42.3.384.

Carpenter, B., Gelman, A., Hoffman, M., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M. A., Guo, J., Li, P., & Riddell, A. (in press). Stan: A probabilistic programming language. *Journal of Statistical Software*.

Chatterjee, H. J., Ho, S. Y., Barnes, I., & Groves, C. (2009). Estimating the phylogeny and divergence times of primates using a supermatrix approach. *BMC Evolutionary Biology*, 9, 1. doi: 10.1186/1471-2148-9-259.

Desper, R. & Gascuel, O. (2006). Getting a tree fast: Neighbor joining, FastME, and distance-based methods. *Current Protocols in Bioinformatics*, 15, 6.3:6.3.1–6.3.28. doi: 10.1002/0471250953.bi0603s15.

Drummond, A., & Rambaut, A. (2007). BEAST: Bayesian evolutionary analysis sampling trees. *BMC Evolutionary Biology*, 7, 214. doi: 10.1186/1471-2148-7-214.

Fan, Y., Wu, R., Chen, M.-H., Kuo, L., & Lewis, P. O. (2011). Choosing among partition models in Bayesian phylogenetics. *Molecular Biology and Evolution*, 28, 523–532. doi: 10.1093/molbev/msq224.

Felsenstein, J. (1978). The number of evolutionary trees. *Systematic Zoology*, 27, 27–33. doi: 10.2307/2412810.

Felsenstein, J. (1981). Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17, 368–376. doi: 10.1007/BF01734359.

Haario, H., Saksman, E., & Tamminen, J. (1999). Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14, 375–396. doi: 10.1007/s001800050022.

Hartig, G., Churakov, G., Warren, W. C., Brosius, J., Makał owski, W., & Schmitz, J. (2013). Retrophylogenomics place tarsiers on the evolutionary branch of anthropoids. *Scientific Reports*, 3, 1756. doi: 10.1038/srep01756.

Hasegawa, M., Kishino, H., & Yano, T. (1985). Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evolution*, 22, 160–174. doi: 10.1007/BF02101694.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97–109. doi: 10.1093/biomet/57.1.97.

Heath, T. A., Huelsenbeck, J. P., & Stadler, T. (2014). The fossilized birth-death process for coherent calibration of divergence-time estimates. *Proceedings of the National Academy of Sciences*, 111, E2957-E2966. doi: 10.1073/pnas.1319091111.

Heled, J. & Bouckaert, R. R. (2013). Looking for trees in the forest: Summary tree from posterior samples. *BMC Evolutionary Biology*, 13, 1. doi: 10.1186/1471-2148-13-1.

Höhna, S., & Drummond, A. J. (2012). Guided tree topology proposals for Bayesian phylogenetic inference. *Systematic Biology*, 61, 1–11. doi: 10.1093/sysbio/syr074.

Höhna, S., Defoin-Platel, M., & Drummond, A. (2008). Clock-constrained tree proposal operators in Bayesian phylogenetic inference. Pages 1-7 in 8th IEEE International Conference on BioInformatics and BioEngineering Athens, Greece. doi: 10.1109/BIBE.2008.4696663.

Höhna, S., Heath, T. A., Boussau, B., Landis, M. J., Ronquist, F., and Huelsenbeck, J. P. (2014).

Probabilistic graphical model representation in phylogenetics. *Systematic Biology*, 63, 753–771. doi: 10.1093/sysbio/syu039.

Höhna, S., Landis, M. J., Heath, T. A., Boussau, B., Lartillot, N., Moore, B. R., Huelsenbeck, J. P., & Ronquist, F. (2016). RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Systematic Biology*, 65, 726–736. doi: 10.1093/sysbio/syw021.

Holder, M. & Lewis, P. (2003). Phylogeny estimation: Traditional and Bayesian approaches. *Nature Reviews Genetics*, 4, 275. doi: 10.1038/nrg1044.

Holder, M. T., Sukumaran, J., and Lewis, P. O. (2008). A justification for reporting the majority-rule consensus tree in Bayesian phylogenetics. *Systematic Biology*, 57, 814–821. doi: 10.1080/10635150802422308.

Huelsenbeck, J., Ronquist, F., Nielsen, R., & Bollback, J. (2001). Bayesian inference of phylogeny and its impact on evolutionary biology. *Science*, 294, 2310–2314. doi: 10.1126/science.1065889.

Huelsenbeck, J., Larget, B., Miller, R., & Ronquist, F. (2002). Potential applications and pitfalls of Bayesian inference of phylogeny. *Systematic Biology*, 51, 673–688. doi: 10.1080/10635150290102366.

Jeffreys, H. (1961). The Theory of Probability. Oxford: Oxford University Press.

Jukes, T., & Cantor, C. (1969). Evolution of protein molecules. *Mammalian Protein Metabolism*, 3, 21–132. doi: 10.1016/B978-1-4832-3211-9.50009-7.

Kass, R., & Raftery, A. (1995). Bayes factors. *Journal of the American Statistical Association*, 90, 773-795. doi: 10.1080/01621459.1995.10476572.

Kendall, D.G. (1948). On the generalized "birth-and-death" process. *The Annals of Mathematical Statistics*, 19, 1–15. doi: 10.1214/aoms/1177730285.

Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16, 111–120. doi: 10.1007/BF01731581.

Kimura, M. (1981). Estimation of evolutionary distances between homologous nucleotide sequences. *Proceedings of the National Academy of Sciences*, 78, 454–458. doi: 10.1073/pnas.78.1.454.

Koller, D., & Friedman, N. (2009). Probabilistic Graphical Models: Principles and Techniques. Cambridge, MA: MIT Press,.

Lakner, C., van der Mark, P., Huelsenbeck, J. P., Larget, B., & Ronquist, F. (2008). Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics. *Systematic Biology*, 57, 86–103. doi: 10.1080/10635150801886156.

Lartillot, N., & Philippe, H. (2006). Computing Bayes factors using thermodynamic integration. *Systematic Biology*, 55, 195. doi: 10.1080/10635150500433722.

Lartillot, N., Lepage, T., & Blanquart, S. (2009). PhyloBayes 3: A Bayesian software package for phylo-genetic reconstruction and molecular dating. *Bioinformatics*, 25, 2286. doi: 10.1093/bioinformatics/btp368.

Lavine, M., & Schervish, M. J. (1999). Bayes factors: What they are and what they are not. *The American Statistician*, 53, 119–122. doi: 10.2307/2685729.

Lewis, P. O. (2003). NCL: A C++ class library for interpreting data files in NEXUS format. *Bioinformatics*, 19, 2330–2331. doi: 10.1093/bioinformatics/btg319.

Li, S., Pearl, D. K., & Doss, H. (2000). Phylogenetic tree construction using Markov chain Monte Carlo. *Journal of the American Statistical Association*, 95, 493–508. doi: 10.1080/01621459.2000.10474227.

Lunn, D. J., Thomas, A., Best, N., & Spiegelhalter, D. (2000). WinBUGS — a Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10, 325–337. doi: 10.1023/A:1008929526011.

Mau, B., Newton, M., & Larget, B. (1999). Bayesian phylogenetic inference via Markov chain Monte Carlo methods. *Biometrics*, 55, 1–12. doi: 10.1111/j.0006-341X.1999.00001.x.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21, 1087–1092. doi: 10.1063/1.1699114.

Nee, S., May, R. M., & Harvey, P. H. (1994). The Reconstructed Evolutionary Process. *Philosophical Transactions: Biological Sciences*, 344, 305–311. doi: 10.1098/rstb.1994.0068.

Page, R.D. 2003. Introduction to inferring evolutionary relationships. *Current Protocols in Bioinformatics 00*, 6.1:6.1.1–6.1.13. doi: 10.1002/0471250953.bi0601s00.

Plummer, M., Best, N., Cowles, K., & Vines, K. (2006). CODA: Convergence diagnosis and output analysis for MCMC. *R News*, 6, 7–11.

Posada, D. & Crandall, K. A. (2001). Selecting the best-fit model of nucleotide substitution. *Systematic Biology*, 50, 580–601. doi: 10.1080/106351501750435121.

Rannala, B., & Yang, Z. (1996). Probability distribution of molecular evolutionary trees: A new method of phylogenetic inference. *Journal of Molecular Evolution*, 43, 304–311. doi: 10.1007/BF02338839.

Rannala, B., Zhu, T., & Yang, Z. (2012). Tail paradox, partial identifiability, and influential priors in Bayesian branch length inference. *Molecular Biology and Evolution*, 29, 325–335. doi: 10.1093/molbev/msr210.

Roberts, G. O., & Rosenthal, J. S. (2009). Examples of adaptive MCMC. *Journal of Computational and Graphical Statistics*, 18, 349–367. doi: 10.1198/jcgs.2009.06134.

Ronquist, F., Teslenko, M., van der Mark, P., Ayres, D. L., Darling, A., Höhna, S., Larget, B., Liu, L., Suchard, M. A., & Huelsenbeck, J. P. (2012). Mrbayes 3.2: Efficient Bayesian phylogenetic inference and model choice across a large model space. *Systematic Biology*, 61, 539–542. doi: 10.1093/sysbio/sys029.

Shapiro, B., Rambaut, A., & Drummond, A. (2006). Choosing appropriate substitution models for the phylogenetic analysis of protein-coding sequences. *Molecular Biology and Evolution*, 23, 7. doi: 10.1093/molbev/msj021.

Stadler, T. (2010). Sampling-through-time in birth-death trees. *Journal of Theoretical Biology*, 267, 396–404. doi: 10.1016/j.jtbi.2010.09.010.

Sukumaran, J., & Holder, M. T. (2010). Dendropy: A Python library for phylogenetic computing. *Bioinformatics*, 26, 1569–1571. doi: 10.1093/bioinformatics/btq228.

Tamura, K., & Nei, M. (1993). Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Molecular Biology and Evolution*, 10, 512–526.

Tavaré, S. (1986). Some probabilistic and statistical problems in the analysis of DNA sequences. In: Some Mathematical Questions in Biology—DNA Sequence Analysis, Miura RM (Ed.), American Mathematical Society, Providence (RI), 17, 57–86.

Thorne, J., Kishino, H., & Painter, I. S. (1998). Estimating the rate of evolution of the rate of molecular evolution. *Molecular Biology and Evolution*, 15, 1647–1657. doi: 10.1093/oxfordjournals.molbev.a025892.

Wilgenbusch, J.C. & Swofford, D. (2003). Inferring evolutionary trees with PAUP*. *Current Protocols in Bioinformatics 00*, 6.4:6.4.1–6.4.28.

Xie, W., Lewis, P., Fan, Y., Kuo, L., & Chen, M. (2011). Improving marginal likelihood estimation for Bayesian phylogenetic model selection. *Systematic Biology*, 60, 150–160. doi: 10.1093/sysbio/syq085.

Yang, Z. (1994). Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *Journal of Molecular Evolution*, 39, 306–314. doi: 10.1007/BF00160154.

Yang, Z. (2014). Molecular Evolution: A Statistical Approach. Oxford: Oxford University Press.

Yang, Z., & Rannala, B. (1997). Bayesian phylogenetic inference using DNA sequences: A Markov Chain Monte Carlo method. *Molecular Biology and Evolution*, 14, 717–724. doi: 10.1093/oxfordjournals.molbev.a025811.

Yang, Z., & Rannala, B. (2005). Branch-length prior influences Bayesian posterior probability of phylogeny. *Systematic Biology*, 54, 455. doi: 10.1080/10635150590945313.

Yang, Z., & Rannala, B. (2012). Molecular phylogenetics: Principles and practice. *Nature Reviews Genetics*, 13, 303–314. doi: 10.1038/nrg3186.

Yoder, A. D. (2003). The phylogenetic position of genus Tarsius: Whose side are you on. In Tarsiers: Past, present, and future (pp. 161-175). New Brunswick, NJ: Rutgers University Press.

Zhang, C., Rannala, B., & Yang, Z. (2012). Robustness of compound Dirichlet priors for Bayesian inference of branch lengths. *Systematic Biology*, 61, 779–784. doi: 10.1093/sysbio/sys030.