Phylogenetic Inference using RevBayes

Substitution Models

Sebastian Höhna, Michael Landis, Brian Moore and Tracy Heath

1 Overview

This tutorial provides the first protocol from our recent publication (Höhna et al. 2017). The second protocol is described in the Partitioned data analysis tutorial and the third protocol is described in the Bayes factor tutorial.

The present tutorial demonstrates how to set up and perform analyses using common nucleotide substitution models. The substitution models used in molecular evolution are continuous time Markov models, which are fully characterized by their instantaneous-rate matrix:

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

where μ_{ij} represents the instantaneous rate of substitution from state i to state j. The diagonal elements μ_i are the rates of *not* changing out of state i, equal to the sum of the elements in the corresponding row. 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 this tutorial you will perform phylogeny inference under common models of DNA sequence evolution: JC, F81, HKY85, GTR, GTR+Gamma and GTR+Gamma+I. For all of these substitution models, you will perform an MCMC analysis to estimate phylogeny and other model parameters. The estimated trees will be unrooted trees with independent branch-length parameters. We will provide comments on how to modify the tutorial if you wish to estimate rooted, clock-like trees. All the assumptions will be covered more in detail later in this tutorial.

1.1 Requirements

We assume that you have read and hopefully completed the following tutorials:

- Getting started

- Rev basics

Note that the Rev basics tutorial introduces the basic syntax of Rev but does not cover any phylogenetic models. You may skip the Rev basics tutorial if you have some familiarity with R. We tried to keep this tutorial very basic and introduce all the language concepts and theory on the way. You may only need the Rev basics tutorial for a more in-depth discussion of concepts in Rev.

2 Data and files

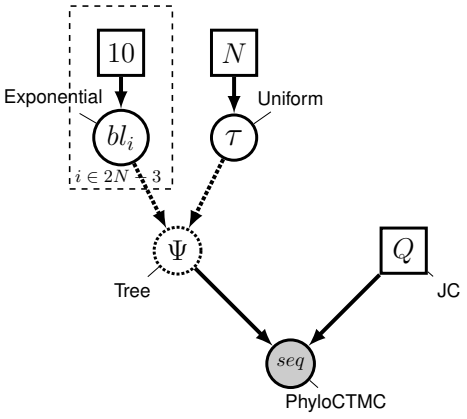
We provide the data file(s) which we will use in this tutorial. You may want to use your own data instead. In the **data** folder, you will find the following files

- **primates_and_galeopterus_cytb.nex**: Alignment of the *cytochrome b* subunit from 23 primates representing 14 of the 16 families (*Indriidae* and *Callitrichidae* are missing). Note that there is one outgroup species included: *Galeopterus variegatus*.

3 Example: Character Evolution under the Jukes-Cantor Substitution Model

3.1 Getting Started

The first section of this exercise involves: (1) setting up a Jukes-Cantor (JC) substitution model for an alignment of the cytochrome b subunit; (2) approximating the posterior probability of the tree topology and node ages (and all other parameters) using MCMC, and; (3) summarizing the MCMC output by computing the maximum *a posteriori* tree.




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


Q <- fnJC(4)

seq ~ dnPhyloCTMC( tree=psi, Q=Q, type="DNA" )

seq.clamp( data )
```

Figure 1: Graphical model representation of a simple phylogenetic model. The graphical model shows the dependencies between the parameters. Here, the rate matrix Q is a constant variable because it is fixed and does not depend on any parameters. The only free parameters of this model, the Jukes-Cantor model, are the tree Ψ including the node ages.

We first consider the simplest substitution model described by Jukes and Cantor (1969). The instantaneous-

rate matrix for the JC substitution model is defined as

$$Q_{JC69} = \begin{pmatrix} * & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & * & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & * & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & * \end{pmatrix} ,$$

which has the advantage that the transition probability matrix can be computed analytically

$$P_{JC69} = \begin{pmatrix} \frac{1}{4} + \frac{3}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} \\ \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} + \frac{3}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} \\ \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} + \frac{3}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} \\ \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} - \frac{1}{4}e^{-rt} & \frac{1}{4} + \frac{3}{4}e^{-rt} \end{pmatrix} ,$$

where t is the branch length in units of time, and r is the rate (clock) for the process. In the later exercises you will be asked to specify more complex substitution models. **Don't worry, you won't have to calculate all of the transition probabilities, because RevBayes will take care of all the computations for you.** Here we only provide some of the equations for the models in case you might be interested in the details. You will be able to complete the exercises without understanding the underlying math.

→ The files for this example analysis are provided for you, which can easily be run using the **source()** function in the RevBayes console:

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

If everything loaded properly, then you should see the program initiate the Markov chain Monte Carlo analysis that estimates the posterior distribution. If you continue to let this run, then you will see it output the states of the Markov chain once the MCMC analysis begins.

Ultimately, this is how you will execute most analyses in RevBayes, with the full specification of the model and analyses contained in the sourced files. You could easily run this entire analysis on your own data by substituting your data file name for that in the model-specification file. However, it is important to understand the components of the model to be able to take full advantage of the flexibility and richness of RevBayes. Furthermore, without inspecting the Rev scripts sourced in **mcmc_JC.Rev**, you may end up inadvertently performing inappropriate analyses on your dataset, which would be a waste of your time and CPU cycles. The next steps will walk you through the full specification of the model and MCMC analyses.

3.2 Loading the Data

→ Download data and output files (if you don't have them already) from:
http://revbayes.github.io/tutorials.html

First load in the sequences using the **readDiscreteCharacterData()** function.

```
data <- readDiscreteCharacterData("data/primates_and_galeopterus_cytb.nex")
```

Executing these lines initializes the data matrix as the respective `Rev` variables. To report the current value of any variable, simply type the variable name and press enter. For the **data** matrix, this provides information about the alignment:

```
data
   DNA character matrix with 23 taxa and 1141 characters
   ====================================================
   Origination:                  primates_and_galeopterus_cytb.nex
   Number of taxa:               23
   Number of included taxa:      23
   Number of characters:         1141
   Number of included characters: 1141
   Datatype:                     DNA
```

Next we will specify some useful variables based on our dataset. The variable **data** has *member functions* that we can use to retrieve information about the dataset. These include, for example, the number of species and the taxa. We will need that taxon information for setting up different parts of our model.

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

Additionally, we set up a counter variable for the number of moves that we already added to our analysis. [Recall that moves are algorithms used to propose new parameter values during the MCMC simulation.] This will make it much easier if we extend the model or analysis to include additional moves or to remove some moves. Similarly, we set up a counter variable for the number of monitors. [Monitors print the values of model parameters to the screen and/or log files during the MCMC analysis].

```
mvi = 1
mni = 1
```

You may have noticed that we used the **=** operator to create the move index. This simply means that the variable is not part of the model. You will later see that we use this operator more often, *e.g.,* when we create moves and monitors.

With the data loaded, we can now proceed to specify our Jukes-Cantor substitution model.

3.3 Jukes-Cantor Substitution Model

A given substitution model is defined by its corresponding instantaneous-rate matrix, Q. The Jukes-Cantor substitution model does not have any free parameters (as the substitution rates are all assumed to be equal, and there is a separate parameter that scales their overall magnitude), so we can define it as a constant variable. The function **fnJC(n)** will create an instantaneous-rate matrix for a character with n states. Since we use DNA data here, we create a 4x4 instantaneous-rate matrix:

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

You can see the rates of the Q matrix by typing

```
Q
   [ [ -1.0000, 0.3333, 0.3333, 0.3333 ] ,
     0.3333, -1.0000, 0.3333, 0.3333 ] ,
     0.3333, 0.3333, -1.0000, 0.3333 ] ,
     0.3333, 0.3333, 0.3333, -1.0000 ] ]
```

As you can see, all substitution rates are equal.

3.4 Tree Topology and Branch Lengths

The tree topology and branch lengths are stochastic nodes in our phylogenetic model. In Figure 1, the tree topology is denoted Ψ and the length of the branch leading to node i is bl_i.

We will assume that all possible labeled, unrooted tree topologies have equal probability. This is the **dnUniformTopology()** distribution in **RevBayes**. Note that in **RevBayes** it is advisable to specify the outgroup for your study system if you use an unrooted tree prior, whereas other software, *e.g.,* **MrBayes** uses the first taxon in the data matrix file as the outgroup. Specify the **topology** stochastic node by passing in the tip labels **names** to the **dnUniformTopology()** distribution:

```
out_group = clade("Galeopterus_variegatus")
topology ~ dnUniformTopology(taxa, outgroup=out_group)
```

Some types of stochastic nodes can be updated by a number of alternative moves. Different moves may explore parameter space in different ways, and it is possible to use multiple different moves for a given parameter to improve mixing (the efficiency of the MCMC simulation). In the case of our unrooted tree topology, for example, we can use both a nearest-neighbor interchange move (**mvNNI**) and a subtree-prune and regrafting move (**mvSPR**). These moves do not have tuning parameters associated with them, thus you only need to pass in the **topology** node and proposal **weight**.

```
moves[mvi++] = mvNNI(topology, weight=1.0)
moves[mvi++] = mvSPR(topology, weight=1.0)
```

The weight specifies how often the move will be applied either on average per iteration or relative to all other moves. Have a look at the MCMC Diagnosis tutorial for more details about moves and MCMC strategies (found on the RevBayes Tutorials Website).

Next we have to create a stochastic node for each of the $2N-3$ branches in our tree (where $N =$ **n_species**). We can do this using a **for** loop — this is a plate in our graphical model. In this loop, we can create each of the branch-length nodes and assign each move. Copy this entire block of Rev code into the console:

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

It is convenient for monitoring purposes to add the tree length as deterministic variable. The tree length is simply the sum of all branch lengths. Accordingly, the tree length can be computed using the **sum()** function, which calculates the sum of any vector of values.

```
TL := sum(br_lens)
```

Alternative branch-length priors
Some studies, *e.g.,* Brown et al. (2010); Rannala et al. (2012), have criticized the exponential prior distribution for branch lengths because it induces a gamma-dsitributed tree-length and the mean of this gamma distribution grows with the number of taxa. For example, we can use instead a specific gamma prior distribution (or any other distribution defined on a positive real variable) for the tree length, and then use a Dirichlet prior distribution to break the tree length into the corresponding branch lengths (Zhang et al. 2012).

```
# specify a prior distribution on the tree length with your desired mean
TL ~ dnGamma(2,4)
moves[mvi++] = mvScale(TL)

# now create a random variable for the relative branch lengths
rel_branch_lengths ~ dnDirichlet( rep(1.0,n_branches) )
moves[mvi++] = mvBetaSimplex(rel_branch_lengths, weight=n_branches)
moves[mvi++] = mvDirichletSimplex(rel_branch_lengths, weight=n_branches/10.0)

# finally, transform the relative branch lengths into actual branch lengths
br_lens := rel_branch_lengths * TL
```

Finally, we can create a *phylogram* (a phylogeny in which the branch lengths are proportional to the expected number of substitutions/site) by combining the tree topology and branch lengths. We do this using the **treeAssembly()** function, which applies the value of the i^{th} member of the **br_lens** vector to the branch leading to the i^{th} node in **topology**. Thus, the **psi** variable is a deterministic node:

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

Alternative Analysis
Prior on Time-Trees: Tree Topology and Node Ages
Alternatively, you may want to specify a prior on time-trees. Here we will briefly indicate how to specify such an prior which will lead to inference of time trees.

The tree (the topology and node ages) is a stochastic node in our phylogenetic model. For simplicity, we will assume a uniform prior on both topologies and node ages. The distribution in **RevBayes** is **dnUniformTimeTree()**.

→ Fore more information on tree priors, such as birth-death processes, please read the RB_DiversificationRate_Tutorial.

First, we need to specify the age of the tree:

```
root_age <- 10.0
```

Here we simply assumed that the tree is 10.0 time units old. We could also specify a prior on the root age if we have fossil calibrations (see Divergence Time and Calibration Tutorial). Next, we specify the **tree** stochastic variable by passing in the taxon information **taxa** to the **dnUniformTimeTree()** distribution:

```
psi ~ dnUniformTimeTree(rootAge=root_age, taxa=taxa)
```

Some types of stochastic nodes can be updated by a number of alternative moves. Different moves may explore parameter space in different ways, and it is possible to use multiple different moves for a given parameter to improve mixing (the efficiency of the MCMC simulation). In the case of our rooted tree, for example, we can use both a nearest-neighbor interchange move without and with changing the node ages (**mvNarrow** and **mvNNI**) and a fixed-nodeheight subtree-prune and regrafting move (**mvFNPR**) and its Metropolized-Gibbs variant (**mvGPR**) (Höhna et al. 2008; Höhna and Drummond 2012). We also need moves that change the ages of the internal nodes, for example, **mvSubtreeScale** and **mvNodeTimeSlideUniform**. These moves do not have tuning parameters associated with them, thus you only need to pass in the **psi** node and proposal **weight**.

```
moves[mvi++] = mvNarrow(psi, weight=5.0)
moves[mvi++] = mvNNI(psi, weight=1.0)
moves[mvi++] = mvFNPR(psi, weight=3.0)
moves[mvi++] = mvGPR(psi, weight=3.0)
moves[mvi++] = mvSubtreeScale(psi, weight=3.0)
moves[mvi++] = mvNodeTimeSlideUniform(psi, weight=15.0)
```

The weight specifies how often the move will be applied either on average per iteration or relative to all other moves. Have a look at the MCMC tutorial for more details about moves and MCMC strategies: http://revbayes.github.io/tutorials.html

Molecular clock
Additionally, in the case of time-calibrated trees, we need to add a molecular clock rate parameter. For example, we know from empirical estimates that the molecular clock rate is about 0.01 (=1%) per million years per site. Nevertheless, we can estimate it here because we fixed the root age. We use a uniform prior on the log-transform clock rate. This specifies our lack of prior knowledge on the magnitude of the clock rate.

```
log_clock_rate ~ dnUniform(-6,1)
moves[mvi++] = mvSlide(log_clock_rate, weight=2.0)
clock_rate := 10^log_clock_rate
```

→ Instead, you could also fix the clock rate and estimate the root age. Fore more information on molecular clocks please read the RB_DivergenceTime_Tutorial.

3.5 Putting it All Together

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** constructor function to create this node. This distribution requires several input arguments: (1) the **tree** with branch lengths; (2) the instantaneous-rate matrix **Q**; (3) the **type** of character data.

Build the random variable for the character data (sequence alignment).

```
# the sequence evolution model
seq ~ dnPhyloCTMC(tree=psi, Q=Q, type="DNA")
```

Once the **PhyloCTMC** model has been created, we can attach our sequence data to the tip nodes in the tree.

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

[Note that 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.

Finally, we wrap the entire model to provide convenient access to the DAG. 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:

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

Now we have specified a simple phylogenetic analysis—each parameter of the model will be estimated from every site in our alignment. If we inspect the contents of **`mymodel`** we can review all of the nodes in the DAG:

```
mymodel
```

3.6 Performing an MCMC Analysis Under the Jukes-Cantor Model

In this section, will describe how to set up the MCMC sampler and summarize the resulting posterior distribution of trees.

3.6.1 Specifying Monitors

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 variable that will output the states for all model parameters when passed into a MCMC function.

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

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.

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

Finally, create a screen monitor that will report the states of specified variables to the screen with **mnScreen**:

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

This monitor mostly helps us to see the progress of the MCMC run.

3.6.2 Initializing and Running the MCMC Simulation

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:

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

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 analysis that is used to tune the scale of the moves to improve mixing of the MCMC analysis.

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

Now, run the MCMC:

```
mymcmc.run(generations=30000)
```

When the analysis is complete, you will have the monitored files in your output directory.

Methods for visualizing the marginal densities of parameter values are not currently available in RevBayes itself. Thus, it is important to use programs like Tracer (Rambaut and Drummond 2011) to evaluate mixing and non-convergence.

→ Look at the file called **output/primates_cytb_JC.log** in Tracer. There you see the posterior distribution of the continuous parameters, *e.g.,* the tree length variable **TL**.

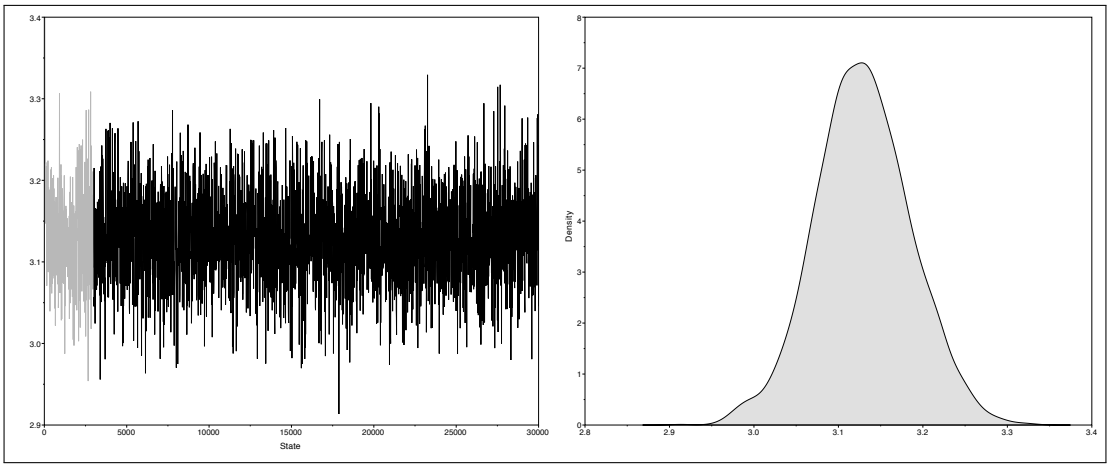


Figure 2: Left: Trace of tree-length samples for one MCMC run. The caterpillar-like look is a good sign. You will also see that the effective sample size is comparably large, *i.e.,* much larger than 200. Right: Posterior distribution of the tree length of the primate phylogeny under a Jukes-Cantor substitution model.

3.7 Exercise 1

We are interested in the phylogenetic relationship of the Tarsiers. Therefore, we need to summarize the trees sampled from the posterior distribution. `RevBayes` can summarize the sampled trees by reading in the tree-trace file:

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

The **`mapTree()`** function will summarize the tree samples and write the maximum *a posteriori* tree to file:

```
map_tree = mapTree(treetrace,"output/primates_cytb_JC_MAP.tree")
```

→ Look at the file called **`output/primates_cytb_JC_MAP.tree`** in `FigTree`. We show it in Figure 3.

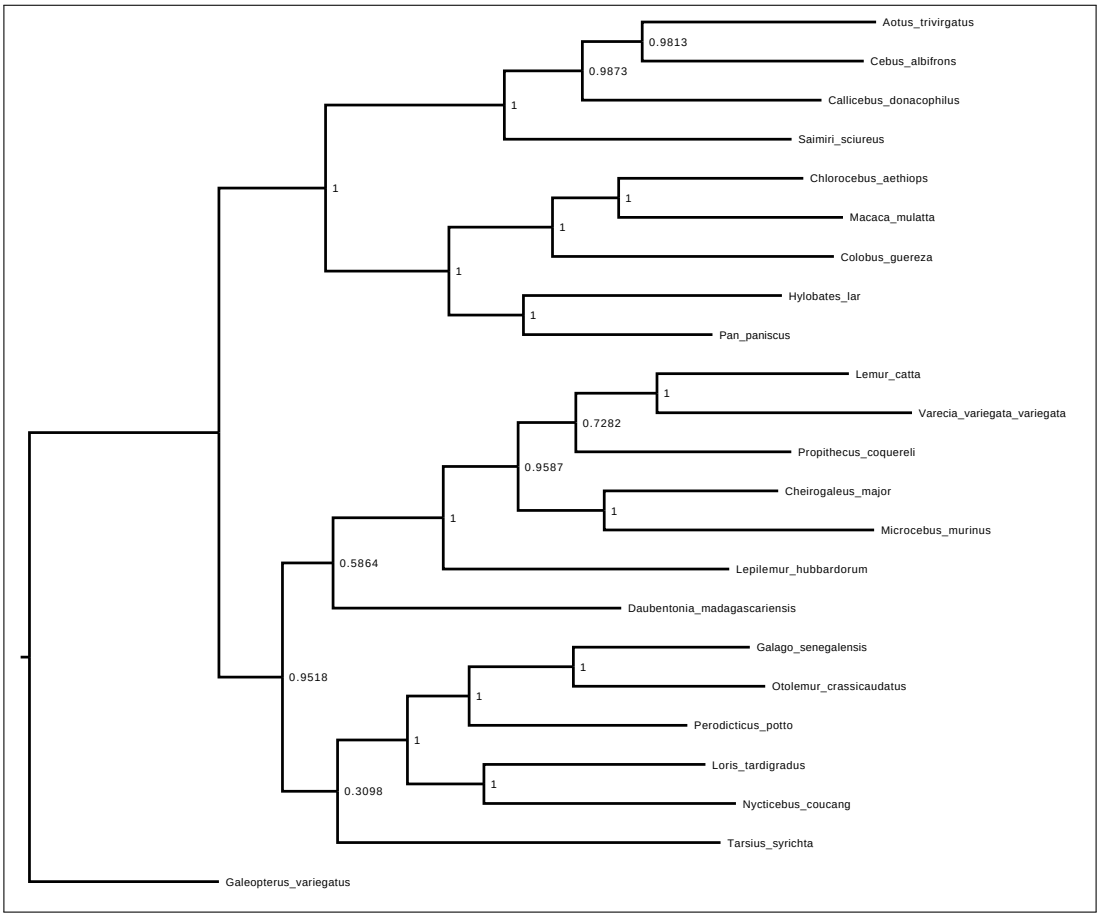


Figure 3: Maximum a posteriori estimate of the primate phylogeny under a Jukes-Cantor substitution model. The numbers at the nodes show the posterior probabilities for the clades. We have rooted the tree at the outgroup *Galeopterus_variegatus*

→ Fill in the following table as you go through the tutorial.

Note, you can query the posterior probability of a clade being monophyletic using the following command:

```
Lemuroidea <- clade("Cheirogaleus_major",
                    "Daubentonia_madagascariensis",
                    "Lemur_catta",
                    "Lepilemur_hubbardorum",
                    "Microcebus_murinus",
                    "Propithecus_coquereli",
                    "Varecia_variegata_variegata")

treetrace.cladeProbability( Lemuroidea )
```

Table 1: Posterior probabilities of primate phylogenetic relationships*.

Model	**Lemuroidea**	**Lorisoidea**	**Platyrrhini**	**Catarrhini**
Jukes-Cantor				
HKY85				
F81				
GTR				
GTR+Γ				
GTR+Γ+I				

*you can edit this table

Table 2: Primate species and famaly relationships.

Species	Family	Parvorder	Suborder
Aotus trivirgatus	Aotidae	Platyrrhini (NWM)	Haplorrhini
Callicebus donacophilus	Pitheciidae	Platyrrhini (NWM)	Haplorrhini
Cebus albifrons	Cebidae	Platyrrhini (NWM)	Haplorrhini
Cheirogaleus major	Cheirogaleidae	Lemuroidea	Strepsirrhini
Chlorocebus aethiops	Cercopithecoidea	Catarrhini	Haplorrhini
Colobus guereza	Cercopithecoidea	Catarrhini	Haplorrhini
Daubentonia madagascariensis	Daubentoniidae	Lemuroidea	Strepsirrhini
Galago senegalensis	Galagidae	Lorisidae	Strepsirrhini
Hylobates lar	Hylobatidea	Catarrhini	Haplorrhini
Lemur catta	Lemuridae	Lemuroidea	Strepsirrhini
Lepilemur hubbardorum	Lepilemuridae	Lemuroidea	Strepsirrhini
Loris tardigradus	Lorisidae	Lorisidae	Strepsirrhini
Macaca mulatta	Cercopithecoidea	Catarrhini	Haplorrhini
Microcebus murinus	Cheirogaleidae	Lemuroidea	Strepsirrhini
Nycticebus coucang	Lorisidae	Lorisidae	Strepsirrhini
Otolemur crassicaudatus	Galagidae	Lorisidae	Strepsirrhini
Pan paniscus	Hominoidea	Catarrhini	Haplorrhini
Perodicticus potto	Lorisidae	Lorisidae	Strepsirrhini
Propithecus coquereli	Indriidae	Lemuroidea	Strepsirrhini
Saimiri sciureus	Cebidae	Platyrrhini (NWM)	Haplorrhini
Tarsius syrichta	Tarsiidae		Haplorrhini
Varecia variegata variegata	Lemuridae	Lemuroidea	Strepsirrhini

4 The Hasegawa-Kishino-Yano (HKY) 1985 Substitution Model

The Jukes-Cantor model assumes that all substitution rates are equal, which also implies that the stationary frequencies of the four nucleotide bases are equal. These assumptions are not very biologically reasonable, so we might wish to consider a more realistic substitution model that relaxes some of these assumptions. For example, we might allow stationary frequencies, π, to be unequal, and allow rates of transition and transversion substitutions to differ, κ. This corresponds to the substitution model proposed by Hasegawa et al. (1985; HKY), which is specified with the following instantaneous-rate matrix:

$$Q_{HKY} = \begin{pmatrix} \cdot & \pi_C & \kappa\pi_G & \pi_T \\ \pi_A & \cdot & \pi_C & \kappa\pi_T \\ \kappa\pi_A & \pi_C & \cdot & \pi_T \\ \pi_A & \kappa\pi_C & \pi_G & \cdot \end{pmatrix}.$$

[The diagonal $\cdot$ entries are equal to the negative sum of the elements in the corresponding row.]

→ Use the file **mcmc_JC.Rev** as a starting point for the HKY analysis.

Note that we are adding two new variables to our model. We can define a variable **pi** for the stationary frequencies that are drawn from a flat Dirichlet distribution by

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

Since **pi** is a stochastic variable, we need to specify a move to propose updates to it. A good move on variables drawn from a Dirichlet distribution is the **mvBetaSimplex**. This move randomly takes an element from the simplex, proposes a new value for it drawn from a Beta distribution, and then rescales all values of the simplex to sum to 1 again.

```
moves[mvi++] = mvBetaSimplex(pi, weight=2)
moves[mvi++] = mvDirichletSimplex(pi, weight=1)
```

The second new variable is κ, which specifies the ratio of transition-transversion rates. The κ parameter must be a positive-real number and a natural choice as the prior distribution is the lognormal distribution:

```
kappa ~ dnLognormal(0.0, 1.0)
```

Again, we need to specify a move for this new stochastic variable. A simple scaling move should do the job.

```
moves[mvi++] = mvScale(kappa)
```

Finally, we need to create the HKY instantaneous-rate matrix using the **fnHKY** function:

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

This should be all for the HKY model.

→ Don't forget to change the output file names, otherwise your old analyses files will be overwritten.

4.1 Exercise 2

- With figure 1 as your guide, draw the probabilistic graphical model of the HKY model.
- Copy the file called **mcmc_JC.Rev** and modify it by including the necessary parameters to specify the HKY substitution model.
- Run an MCMC analysis to estimate the posterior distribution under the HKY substitution model.

- Are the resulting estimates of the base frequencies equal? If not, how much do they differ? Are the estimated base frequencies similar to the empirical base frequencies? The empirical base frequencies are the frequencies of the characters in the alignment, which can be computed with `RevBayes` by `data.getEmpiricalBaseFrequencies()`.
- Is the inferred rate of transition substitutions higher than the rate of transversion substitutions? If so, by how much?
- Like the HKY model, the Felsenstein 1981 (F81) substitution model has unequal stationary frequencies, but it assumes equal transition-transversion rates (Felsenstein 1981). Can you set up the F81 model and run an analysis?
- Complete the Table 1 by reporting the posterior probabilities of phylogenetic relationships.

5 The General Time-Reversible (GTR) Substitution Model

The HKY substitution model can accommodate unequal base frequencies and different rates of transition and transversion substitutions. Despite these extensions, the HKY model may still be too simplistic for many real datasets. Here, we extend the HKY model to specify the General Time Reversible (GTR) substitution model (Tavaré 1986), which allows all six exchangeability rates to differ (Figure 4).

The instantaneous-rate matrix for the GTR substitution model is:

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

where the six exchangeability parameters, r_{ij}, specify the relative rates of change between states i and j.

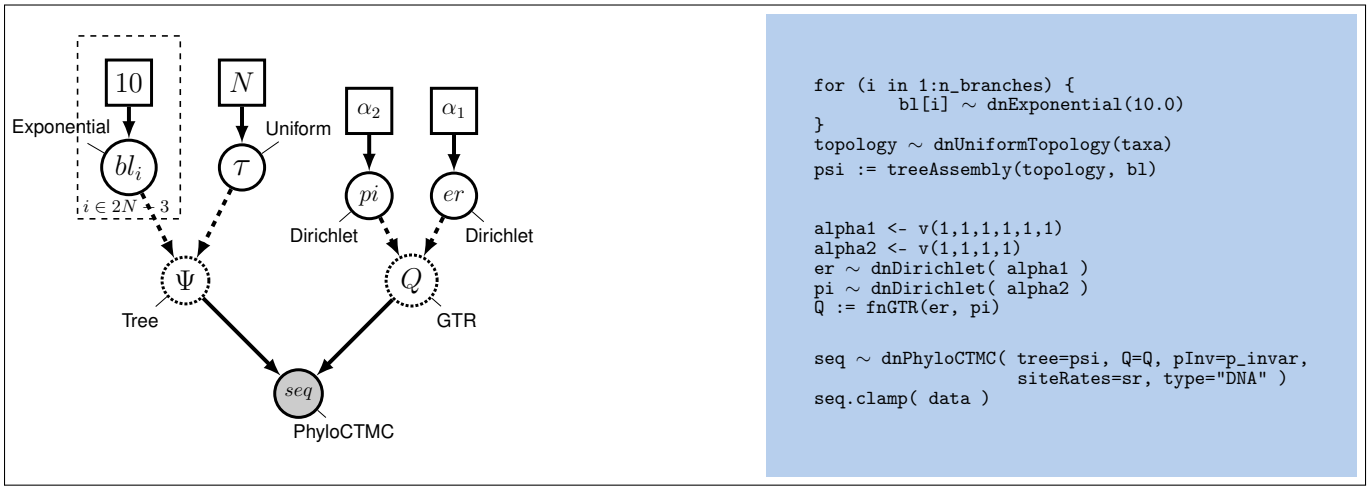


Figure 4: Graphical model representation of the General Time Reversible (GTR) phylogenetic model.

The GTR model requires that we define and specify a prior on the six exchangeability rates, which we will describe using a flat Dirichlet distribution. As we did previously for the Dirichlet prior on base frequencies,

we first define a constant node specifying the vector of concentration-parameter values using the **v()** function:

```
er_prior <- v(1,1,1,1,1,1)
```

This node defines the concentration-parameter values of the Dirichlet prior distribution on the exchangeability rates. Now, we can create a stochastic node for the exchangeability rates using the **dnDirichlet()** function, which takes the vector of concentration-parameter values as an argument and the **~** operator. Together, these create a stochastic node named **er** (θ in Figure 4):

```
er ~ dnDirichlet(er_prior)
```

The Dirichlet distribution assigns probability densities to a group of parameters: e.g., those that measure proportions and must sum to 1. Here, we have specified a six-parameter Dirichlet prior, where each value describes one of the six relative rates of the GTR model: (1) $A \leftrightarrows C$; (2) $A \leftrightarrows G$; (3) $A \leftrightarrows T$; (4) $C \leftrightarrows G$; (5) $C \leftrightarrows T$; (6) $G \leftrightarrows T$. The input parameters of a Dirichlet distribution are called shape (or concentration) parameters. The expectation and variance for each variable are related to the sum of the shape parameters. The prior we specified above is a 'flat' or symmetric Dirichlet distribution; all of the shape parameters are equal (1,1,1,1,1,1). This describes a model that allows for equal rates of change between nucleotides, such that the expected rate for each is equal to $\frac{1}{6}$ (Figure 5a). We might also parameterize the Dirichlet distribution such that all of the shape parameters were equal to 100, which would also specify a prior with an expectation of equal exchangeability rates (Figure 5b). However, by increasing the values of the shape parameters, **er_prior <- v(100,100,100,100,100,100)**, the Dirichlet distribution will more strongly favor equal exchangeability rates; (*i.e.*, a relatively *informative* prior). Alternatively, we might consider an asymmetric Dirichlet parameterization that could reflect a strong prior belief that transition and transversion substitutions occur at different rates. For example, we might specify the prior density **er_prior <- v(4,8,4,4,8,4)**. Under this model, the expected rate for transversions would be $\frac{4}{32}$ and that for transitions would be $\frac{8}{32}$, and there would be greater prior probability on sets of GTR rates that matched this configuration (Figure 5c). Yet another aymmetric prior could specify that each of the six GTR rates had a different value conforming to a Dirichlet(2,4,6,8,10,12). This would lead to a different prior probability density for each rate parameter (Figure 5d). Without strong prior knowledge about the pattern of relative rates, however, we can better reflect our uncertainty by using a vague prior on the GTR rates. Notably, all patterns of relative rates have the same probability density under **er_prior <- v(1,1,1,1,1,1)**.

For each stochastic node in our model, we must also specify a proposal mechanism if we wish to estimate that parameter. The Dirichlet prior on our parameter **er** creates a *simplex* of values that sum to 1.

```
moves[mvi++] = mvBetaSimplex(er, weight=3)
moves[mvi++] = mvDirichletSimplex(er, weight=1)
```

We can use the same type of distribution as a prior on the 4 stationary frequencies ($\pi_A, \pi_C, \pi_G, \pi_T$) since these parameters also represent proportions. Specify a flat Dirichlet prior density on the base frequencies:

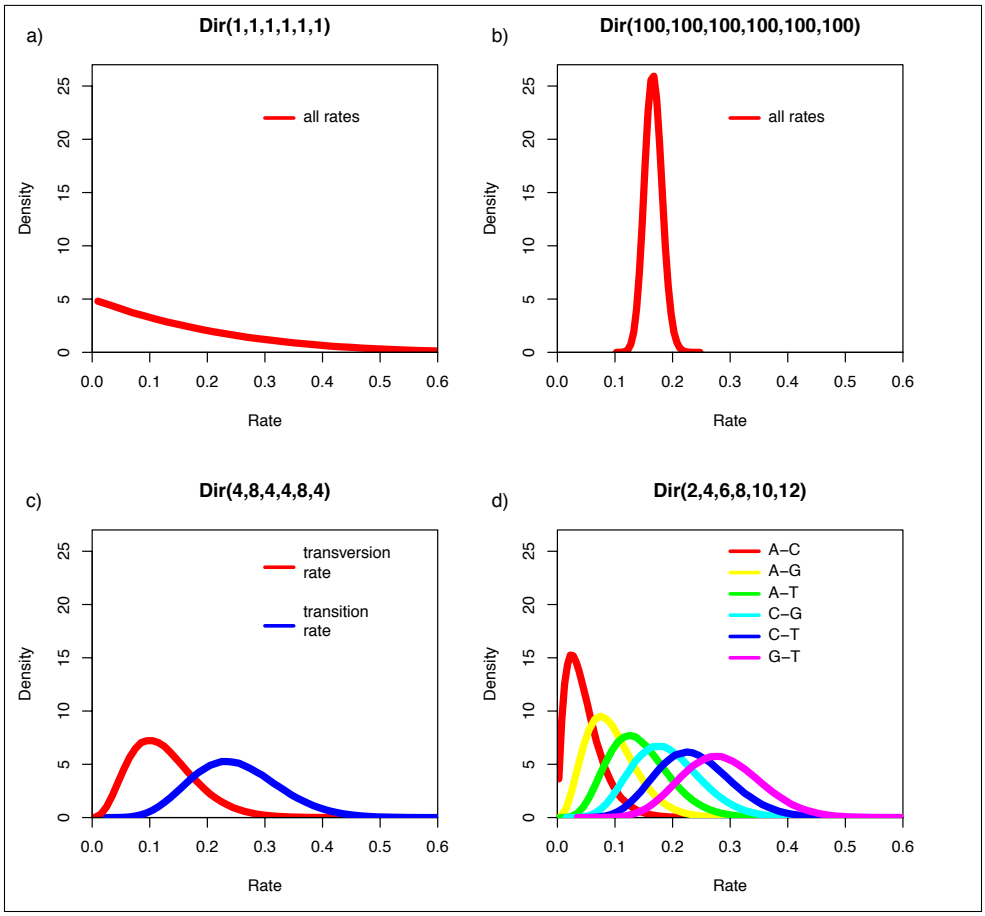


Figure 5: Four different examples of Dirichlet priors on exchangeability rates.

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

The node **pi** represents the π node in Figure 4. Now add the simplex scale move on the stationary frequencies to the moves vector:

```
moves[mvi++] = mvBetaSimplex(pi, weight=2)
moves[mvi++] = mvDirichletSimplex(pi, weight=1)
```

We can finish setting up this part of the model by creating a deterministic node for the GTR instantaneous-rate matrix **Q**. The **fnGTR()** function takes a set of exchangeability rates and a set of base frequencies to compute the instantaneous-rate matrix used when calculating the likelihood of our model.

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

5.1 Exercise 3

- Use one of your previous analysis files—either the `mcmc_JC.Rev` or `HKY.Rev`—to specify a GTR analysis in a new file called `mcmc_GTR.Rev`. Adapt the old analysis to be performed under the GTR substitution model.
- Run an MCMC analysis to estimate the posterior distribution.
- Complete the table of the phylogenetic relationship of primates.

6 The Discrete Gamma Model of Among Site Rate Variation

Members of the GTR family of substitution models assume that rates are homogeneous across sites, an assumption that is often violated by real data. We can accommodate variation in substitution rate among sites (ASRV) by adopting the discrete-gamma model (Yang 1994). This model assumes that the substitution rate at each site is a random variable that is described by a discretized gamma distribution, which has two parameters: the shape parameter, α, and the rate parameter, β. In order that we can interpret the branch lengths as the expected number of substitutions per site, this model assumes that the mean site rate is equal to 1. The mean of the gamma is equal to α/β, so a mean-one gamma is specified by setting the two parameters to be equal, $\alpha = \beta$. This means that we can fully describe the gamma distribution with the single shape parameter, α. The degree of among-site substitution rate variation is inversely proportional to the value of the α-shape parameter. As the value of the α-shape increases, the gamma distribution increasingly resembles a normal distribution with decreasing variance, which therefore corresponds to decreasing levels of ASRV (Figure 6). By contrast, when the value of the α-shape parameter is < 1, the gamma distribution assumes a concave distribution that concentrates most of the prior density on low rates, but retains some prior mass on sites with very high rates, which therefore corresponds to high levels of ASRV (Figure 6). Note that, when $\alpha = 1$, the gamma distribution collapses to an exponential distribution with a rate parameter equal to β.

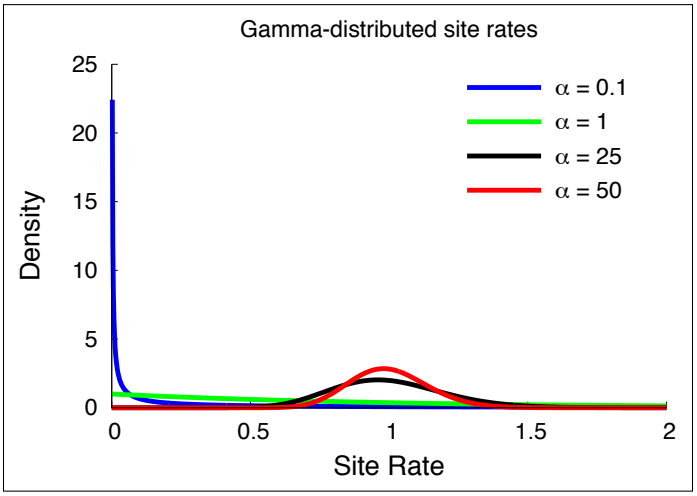
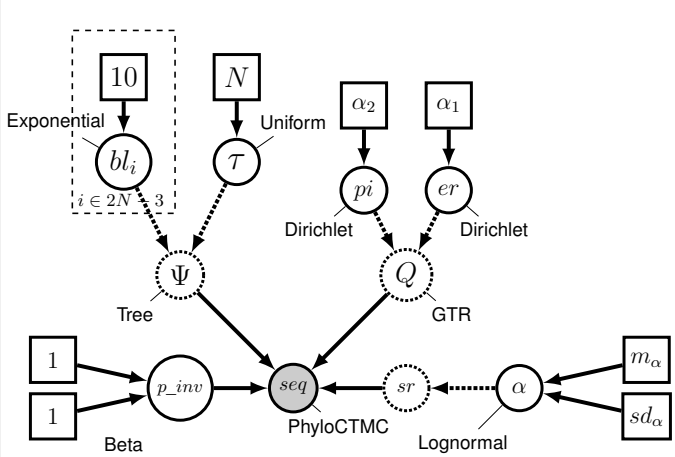


Figure 6: The probability density of mean-one gamma-distributed rates for different values of the α-shape parameter.

We typically lack prior knowledge regarding the degree of ASRV for a given alignment. Accordingly, rather than specifying a precise value of α, we can instead estimate the value of the α-shape parameter from the data. This requires that we specify a diffuse (relatively 'uninformative') prior on the α-shape parameter.

For this analysis, we will use a lognormal distribution with a mean parameter, **alpha_prior_mean**, equal to **5.0**, and standard deviation, **alpha_prior_sd**, equal to 0.587405 (thus, 95% of the prior density spans exactly one order of magnitude).

This approach for accommodating ASRV is another example of a hierarchical model (Figure 7). That is, variation in substitution rates across sites is addressed by applying a site-specific rate multiplier to each of the j sites, r_j. These rate-multipliers are drawn from a discrete, mean-one gamma distribution; the shape of this prior distribution (and the corresponding degree of ASRV) is governed by the α-shape parameter. The α-shape parameter, in turn, is treated as a lognormal distributed random variable. Finally, the shape of the lognormal prior is governed by the mean and standard deviation parameters, which are set to fixed values.




```
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 7: Graphical model representation of the General Time Reversible (GTR) + Gamma phylogenetic model with invariable sites.

6.1 Setting up the Gamma Model in RevBayes

Create a constant node called **alpha_prior_mean** for the mean parameter and a constant node called **alpha_prior_sd** for the standard deviation of the lognormal prior on the gamma-shape parameter (this is represented as the constant m_α and sd_α parameters in Figure 7):

```
alpha_prior_mean <- ln(5.0)
alpha_prior_sd <- 0.587405
```

Then create a stochastic node called **alpha** with a lognormal prior (this represents the stochastic node for the α-shape parameter in Figure 7):

```
alpha ~ dnLognormal( alpha_prior_mean, alpha_prior_sd )
```

The way the ASRV model is implemented involves discretizing the mean-one gamma distribution into a set number of rate categories, k. Thus, we can analytically marginalize over the uncertainty in the rate at

each site. The likelihood of each site is averaged over the k rate categories, where the rate multiplier is the mean (or median) of each of the discrete k categories. To specify this, we need a deterministic node that is a vector that will hold the set of k rates drawn from the gamma distribution with k rate categories. The **fnDiscretizeGamma()** function returns this deterministic node and takes three arguments: the shape and rate of the gamma distribution and the number of categories. Since we want to discretize a mean-one gamma distribution, we can pass in **alpha** for both the shape and rate.

Initialize the **gamma_rates** deterministic node vector using the **fnDiscretizeGamma()** function with **4** bins:

```
gamma_rates := fnDiscretizeGamma( alpha, alpha, 4 )
```

Note that here, by convention, we set $k = 4$. The random variable that controls the rate variation is the stochastic node **alpha**. We will apply a simple scale move to this parameter.

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

Remember that you need to call the **PhyloCTMC** constructor to include the new site-rate parameter:

```
seq ~ dnPhyloCTMC(tree=psi, Q=Q, siteRates=gamma_rates, type="DNA")
```

6.2 Exercise 4

Modify the previous GTR analysis to specify the GTR+Gamma model. Run an MCMC simulation to estimate the posterior distribution.

- Is there an impact on the estimated phylogeny compared with the previous analyses? Look at the MAP tree and the posterior probabilities of the clades.
- Complete the table of the phylogenetic relationship of primates.

7 Modeling Invariable Sites

All of the substitution models described so far assume that the sequence data are potentially variable. That is, we assume that the sequence data are random variables; specifically, we assume that they are realizations of the specified **PhyloCTMC** distribution. However, some sites may not be free to vary—when the substitution rate of a site is zero, it is said to be *invariable*. Invariable sites are often confused with *invariant* sites—when each species exhibits the same state, it is said to be invariant. The concepts are related but distinct. If a site is truly invariable, it will necessarily give rise to an invariant site pattern, as such sites will always have a zero substitution rate. However, an invariant site pattern may be achieved via multiple substitutions that happen to end in the same state for every species.

Here we describe an extension to our phylogenetic model to accommodate invariable sites. Under the invariable-sites model (Hasegawa et al. 1985), each site is invariable with probability **pinvar**, and variable with probability 1−**pinvar**.

First, let's have a look at the data and see how many invariant sites we have:

```
data.getNumInvariantSites()
```

There seem to be a substantial number of invariant sites.

Now let's specify the invariable-sites model in RevBayes. We need to specify the prior probability that a site is invariable. A Beta distribution is a common choice for parameters representing probabilities.

```
pinvar ~ dnBeta(1,1)
```

The **Beta(1,1)** distribution is a flat prior distribution that specifies equal probability for all values between 0 and 1.

Then, as usual, we add a move to change this stochastic variable; we'll used a simple sliding window move.

```
moves[mvi++] = mvSlide(pinvar)
```

Finally, you need to call the **PhyloCTMC** constructor to include the new**pinvar** parameter:

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

7.1 Exercise 5

- Extend the GTR model to account for invariable sites and run an analysis.
- What is the estimated probability of invariable sites and how does it relate to the ratio of invariant sites to the total number of sites?

- Extend the GTR+Γ model to account for invariable sites and run an analysis.
- What is the estimated probability of invariable sites now?
- Complete the table of the phylogenetic relationship of primates.

References

Brown, J. M., S. M. Hedtke, A. R. Lemmon, and E. M. Lemmon. 2010. When trees grow too long: Investigating the causes of highly inaccurate Bayesian branch-length estimates. Systematic Biology 59:145–161.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of Molecular Evolution 17:368–376.

Hasegawa, M., H. Kishino, and T. Yano. 1985. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. Journal of Molecular Evolution 22:160–174.

Höhna, S., M. Defoin-Platel, and A. Drummond. 2008. Clock-constrained tree proposal operators in Bayesian phylogenetic inference. Pages 1–7 *in* 8th IEEE International Conference on BioInformatics and BioEngineering, 2008. BIBE 2008.

Höhna, S. and A. J. Drummond. 2012. Guided tree topology proposals for Bayesian phylogenetic inference. Systematic Biology 61:1–11.

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

Jukes, T. and C. Cantor. 1969. Evolution of protein molecules. Mammalian Protein Metabolism 3:21–132.

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

Rannala, B., T. Zhu, and Z. Yang. 2012. Tail paradox, partial identifiability, and influential priors in Bayesian branch length inference. Molecular Biology and Evolution 29:325–335.

Tavaré, S. 1986. Some probabilistic and statistical problems in the analysis of DNA sequences. Some Mathematical Questions in Biology: DNA Sequence Analysis 17:57–86.

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

Zhang, C., B. Rannala, and Z. Yang. 2012. Robustness of compound Dirichlet priors for Bayesian inference of branch lengths. Systematic Biology 61:779–784.

Version dated: September 23, 2017