Phylogenetic Inference using RevBayes

Substitution Models

1 Overview

This tutorial demonstrates how to set up and perform an analysis for different substitution models. The substitution models used in molecular evolution are continuous time Markov models which are characterized by their substitution 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 rate of substitution from i to j. Given the rate matrix Q, we can compute the transition probabilities by

$$P(t) = \begin{pmatrix} p_{AA}(t) & p_{GA}(t) & p_{CA}(t) & p_{TA}(t) \\ p_{AG}(t) & p_{GG}(t) & p_{CG}(t) & p_{TG}(t) \\ p_{AC}(t) & p_{GC}(t) & p_{CC}(t) & p_{TC}(t) \\ p_{AT}(t) & p_{GT}(t) & p_{CT}(t) & p_{TT}(t) \end{pmatrix} = e^{Qt} = \sum_{j=0}^{\infty} \frac{tQ^j}{j!} \ .$$

The difference between different molecular substitution models is how the rate matrix Q is defined.

In this tutorial you will create a phylogenetic model for the evolution of DNA sequences under a JC, HKY85, GTR, GTR+Gamma and GTR+Gamma+I substitution model. For all these models you will perform an MCMC run to estimate phylogeny and other model parameters.

Requirements

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

- RB_Getting_Started
- RB_Basics_Tutorial

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

2.1 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 branch lengths (and all other parameters) using MCMC, (3) summarizing the MCMC output by computing the maximum a posteriori tree.

The general structure of the model is represented in Figure 1. In the example we will use the simplest

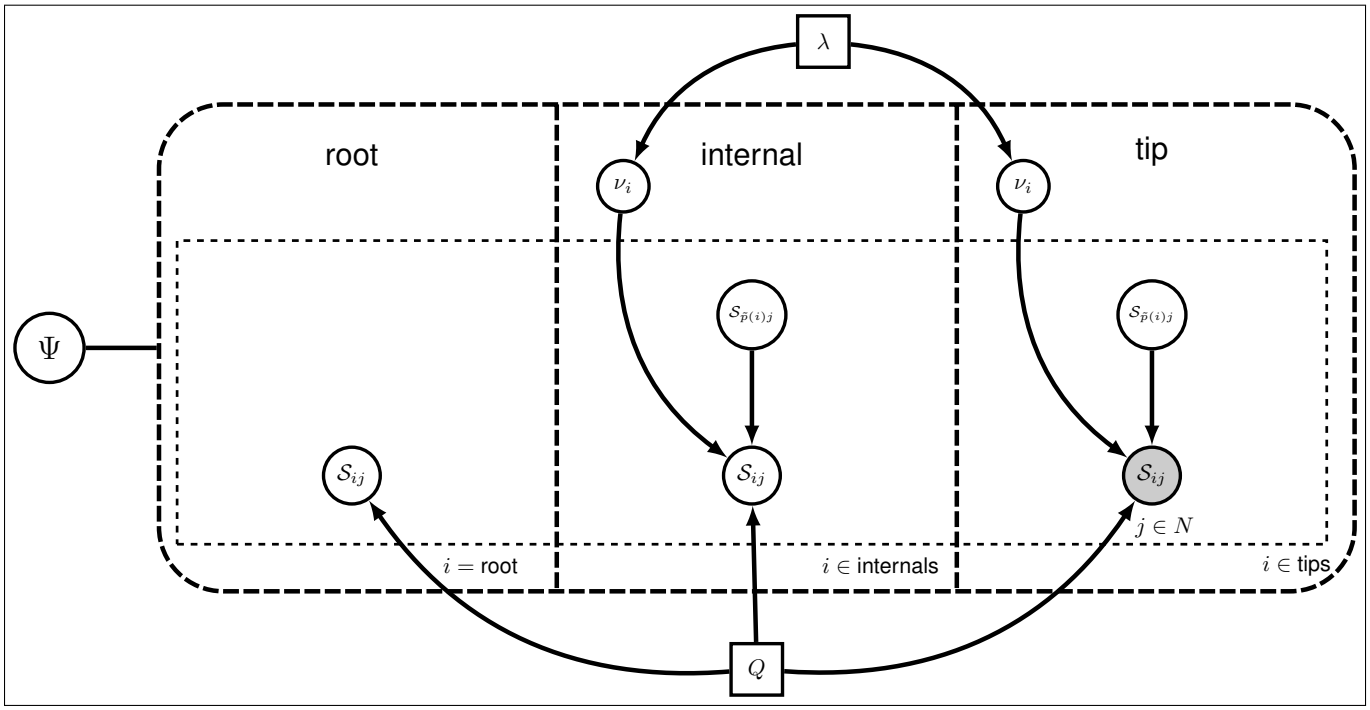

Figure 1: Graphical model representation of a simple phylogenetic model. The graphical model shows the dependencies between the parameters. Here, the rate matrix Q is constant variable because it is fixed and does not depend on any parameters. The only free parameters of this model, the unconstrained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

substitution model, the Jukes-Cantor (Jukes and Cantor 1969). The rate matrix is defined as

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

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

$$P_{JC69} = \begin{pmatrix} \frac{1}{4}+\frac{3}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} \\ \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}+\frac{3}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} \\ \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}+\frac{3}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} \\ \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}-\frac{1}{4}e^{-t\mu} & \frac{1}{4}+\frac{3}{4}e^{-t\mu} \end{pmatrix}.$$

In the later exercises you will be asked to use more complex substitution models. **Don't be scared about the math!** `RevBayes` will take care of all the computations for you. Here we only provide some of the equations for the models in case that you are 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 and you can run these without significant effort using the **`source()`** function in the `RevBayes` console:

```
source("RevBayes_scripts/JukesCantor.Rev")
```

If everything loaded properly, then you should see the program begin running the Markov chain Monte Carlo analysis needed for estimating 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. (It is worth noting, however, that the file **`JukesCantor.Rev`** performs shorter runs with fewer generations for a faster run time.)

Ultimately, this is how you will execute most analyses in `RevBayes` and the full specification of the model and analyses are contained in the sourced files. You could easily run this entire analysis on your own data if you changed the name of the files containing the tutorial's sequences. However, it is important to understand the components of the model to be able to take advantage of the flexibility and richness of `RevBayes`. Furthermore, without inspecting the `Rev` scripts sourced in **`JukesCantor.Rev`**, you may have inadvertently conducted an inappropriate analysis 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.

2.2 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_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 24 taxa and 1141 characters
   ====================================================
   Origination:                  primates_cytb.nex
   Number of taxa:               24
   Number of included taxa:      24
   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 the number of species (**n_species**), the tip labels (**names**), and the number of internal branches (**n_branches**). Each of these variables will be necessary for setting up different parts of our model.

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

Additionally, we set up a counter variable for the number of moves that we already added to our analysis. This will make it much easier if we extend the model or analysis to include additional moves or to remove some moves.

```
mi = 0
```

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.

Now we can proceed with building our Jukes-Cantor model.

2.3 Jukes-Cantor Substitution Model

The substitution model is defined by the rate matrix Q. Since the Jukes-Cantor substitution model does not have any parameter, we can define it as a constant variable. The function **fnJC(n)** will create a rate matrix for character with n states. Since we use DNA data here, we create a 4x4 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.

2.4 Tree Topology and Branch Lengths

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

We will assume that all possible labeled, unrooted tree topologies have equal probability. This is the **dnUniformTopology()** distribution in RevBayes. Specify the **topology** stochastic node by passing in the tip labels **names** to the **dnUniformTopology()** distribution:

```
topology ~ dnUniformTopology(names)
```

For some types of stochastic nodes there are several available moves. Often the different moves explore parameter space in a different way and nothing prevents one from using multiple different moves to improve mixing. For the unrooted tree topology, 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 **weight**

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

The weight specifies how often the move will be applied either on average per iteration or relatively 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

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 branch-length node 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[++mi] = mvScale(br_lens[i])
}
```

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

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

Finally, we can create a branch-length phylogeny by combining the tree topology and branch lengths 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 **phylogeny** variable is a deterministic node:

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

2.5 Putting it All Together

Now that we have initialized virtually all of our model parameters and we can link all of the parts in the stochastic node that will be clamped by the data. The sequence substitution model is a distribution called the *phylogenetic continuous-time Markov chain* and we use the **PhyloCTMC** 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.

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

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

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

When this function is called, RevBayes sets each of the stochastic nodes representing the tip nodes of the tree to the sequence corresponding to that species in the alignment. This essentially tells the program that we have observed data for the sequences at the tips.

Now we can wrap up the whole model to conveniently access 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 molecular phylogeny 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
```

2.6 Perform MCMC Analysis Under the Uniform Model

This section will cover setting up the MCMC sampler and summarizing the posterior distribution of trees.

2.6.1 Specify Monitors

For our MCMC analysis we need to set up a vector of *monitors* to save 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[1] = mnModel(filename="output/primates_cytb_JC_posterior.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[2] = mnFile(filename="output/primates_cytb_JC_posterior.trees",printgen=10,
    separator = TAB, phylogeny)
```

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

```
monitors[3] = mnScreen(printgen=1000, TL)
```

2.7 Initialize and Run MCMC

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 can run the **.burnin()** member function if we wish to pre-run the chain and discard the initial states.

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

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. (`RevBayes` does, however, have a tool for convergence assessment called **`beca`**.)

→ Look at the file called **`output/primates_cytb_JC_posterior.log`** in Tracer.

2.8 Exercise

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

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

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

```
mapTree(treetrace,"output/primates_cytb_JC.tree")
```

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

Table 1: Posterior probabilities of phylogenetic relationship*.

Model	*Lemuroidea*	*Lorisoidea*	*Platyrrhini*	*Catarrhini*	*other*
Jukes-Cantor					
HKY85					
F81					
GTR					
GTR+Γ					
GTR+Γ+I					
Your model 1					
Your model 2					
Your model 3					

*you can edit this table

3 The Hasegawa-Kishino-Yano (HKY) 1985 substitution model

Previously with the Jukes-Cantor substitution model we assumed that all rates are equal. This implies that also the base-frequencies are all equal. These assumptions are very unlikely to be realistic. Instead, we can use the HKY model (Hasegawa et al. 1985) which has unequal equilibrium frequencies π and a different transition-transversion rate κ. The HKY model thus defines the rate matrix Q as

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

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

This means that there are two additional variables in the model. We can define a variable **`pi`** for the base frequency drawn from a flat Dirichlet distribution by

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

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

```
moves[++mi] = mvSimplexElementScale(pi, weight=4.0)
```

The second new variable is the transition-transversion rate κ. κ needs to be a positive real number and a natural choice as the prior distribution is the lognormal distribution:

```
kappa ~ dnLnorm(0.0,1.25)
```

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

```
moves[++mi] = mvScale(kappa, weight=1.0)
```

Finally, we only need to create the HKY 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.

3.1 Exercises

- Copy the file called **JukesCantor.Rev** and modify it with the new parameters so that you have an HKY analysis.
- Run an MCMC to estimate the posterior distribution.
- Are 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.empiricalBaseFrequencies()**.
- Is the rate of transition higher than the rate of transversion? If so, by how much?
- The Felsenstein 1981 substitution model has unequal base frequencies, just as the HKY model, but assumes an equal transition-transversion rate (Felsenstein 1981). Can you set up the F81 model and run an analysis?
- Complete the table of the phylogenetic relationship of Tarsiers.

4 GTR

In the above section we used the HKY substitution model and thus a model with unequal base frequencies and a rate for the difference between transitions and transversion. However, the HKY model may still be too simple in most realistic cases. Here we extend the model to the General Time Reversible (GTR) substitution model (Tavaré 1986). The rate matrix of the GTR mode is defined as

$$Q_{GTR} = \begin{pmatrix} -(x_1+x_2+x_3) & \frac{\pi_1 x_1}{\pi_2} & \frac{\pi_1 x_2}{\pi_3} & \frac{\pi_1 x_3}{\pi_4} \\ x_1 & -(\frac{\pi_1 x_1}{\pi_2}+x_4+x_5) & \frac{\pi_2 x_4}{\pi_3} & \frac{\pi_2 x_5}{\pi_4} \\ x_2 & x_4 & -(\frac{\pi_1 x_2}{\pi_3}+\frac{\pi_2 x_4}{\pi_3}+x_6) & \frac{\pi_3 x_6}{\pi_4} \\ x_3 & x_5 & x_6 & -(\frac{\pi_1 x_3}{\pi_4}+\frac{\pi_2 x_5}{\pi_4}+\frac{\pi_3 x_6}{\pi_4}) \end{pmatrix}.$$

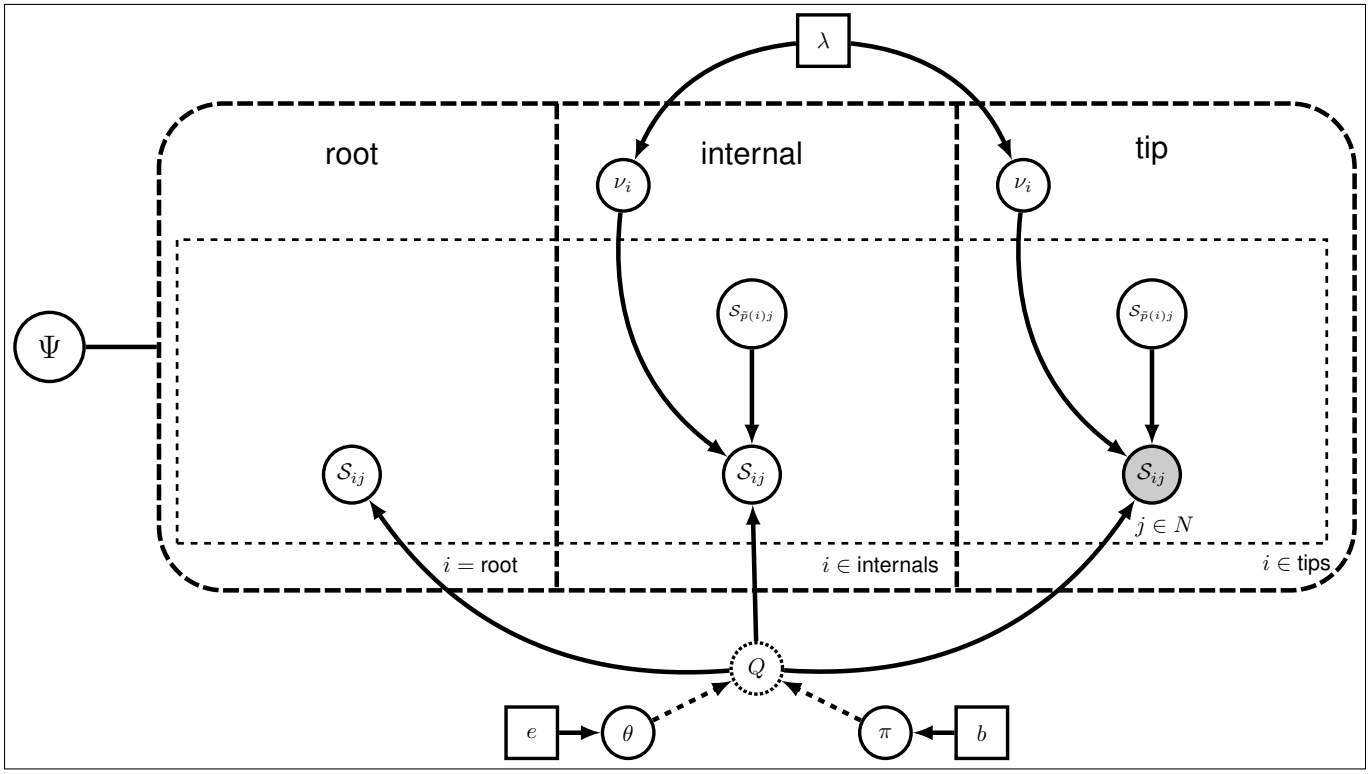

Figure 2: Graphical model representation of the General Time Reversible (GTR) phylogenetic model.

4.1 The Parameters

First, we will define and specify a prior on the exchangeability rates of the GTR model. We will use a flat Dirichlet prior distribution on these six rates. To do this, we must begin by defining a constant node that specifies the vector of concentration values of the Dirichlet prior using the **v()** function:

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

This node defines the parameters of the Dirichlet prior distribution on the exchangeability rates. Thus, we can create a stochastic node for the exchangeability rates using the **dnDirichlet()** function, which takes

a vector of values as an argument and the **~** operator. Together, these create a stochastic node named **er** (θ in Figure 2):

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

The Dirichlet distribution assigns probability densities to grouped parameters: e.g., , those that measure proportions and must sum to 1. Above, we specified a 6-parameter Dirichlet prior on the relative rates of the GTR model, where the placement of each value specified represents one of the 6 relative rates: (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 parameters or concentration parameters and a value is specified for each of the 6 GTR rates. The expectation and variance for each variable are related to the sum of the shape parameters. The prior above is a 'flat' or symmetric Dirichlet since all of the shape parameters are equal (1,1,1,1,1,1), thus we are specifying a model that allows for equal rates of change between nucleotides, such that the expected rate for each is equal to $\frac{1}{6}$ (Zwickl and Holder 2004). Figure 3a shows the probability density of each rate under this model. If we parameterized the Dirichlet distribution such that all of the parameters were equal to 100, this would also specify a prior with an expectation of equal exchangeability rates (Figure 3b). However, by increasing the shape parameters of the Dirichlet distribution, **er_prior <- v(100,100,100,100,100,100)**, would heavily restrict the MCMC from sampling sets of GTR rates in which the values were not equal or very nearly equal (*i.e.*, this is a very *informative* prior). We can consider a different Dirichlet parameterization if we had strong prior belief that transitions and transversions occurred at different rates. In this case, we could specify a more informative 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 the expected rate for transitions would equal $\frac{8}{32}$, and there would be greater prior probability on sets of GTR rates that matched this configuration (Figure 3c). An alternative informative prior would be one where we assumed that each of the 6 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 3d). Without strong prior knowledge about the pattern of relative rates, however, we can better capture our statistical uncertainty with 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 sample that value. The Dirichlet prior on our parameter **er** creates a *simplex* of values that sum to 1.

```
moves[++mi] <- mvSimplexElementScale(er, weight=3)
```

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:

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

The node **pi** represents the π node in Figure 2. Now add the simplex scale move on the stationary frequencies to the moves vector:

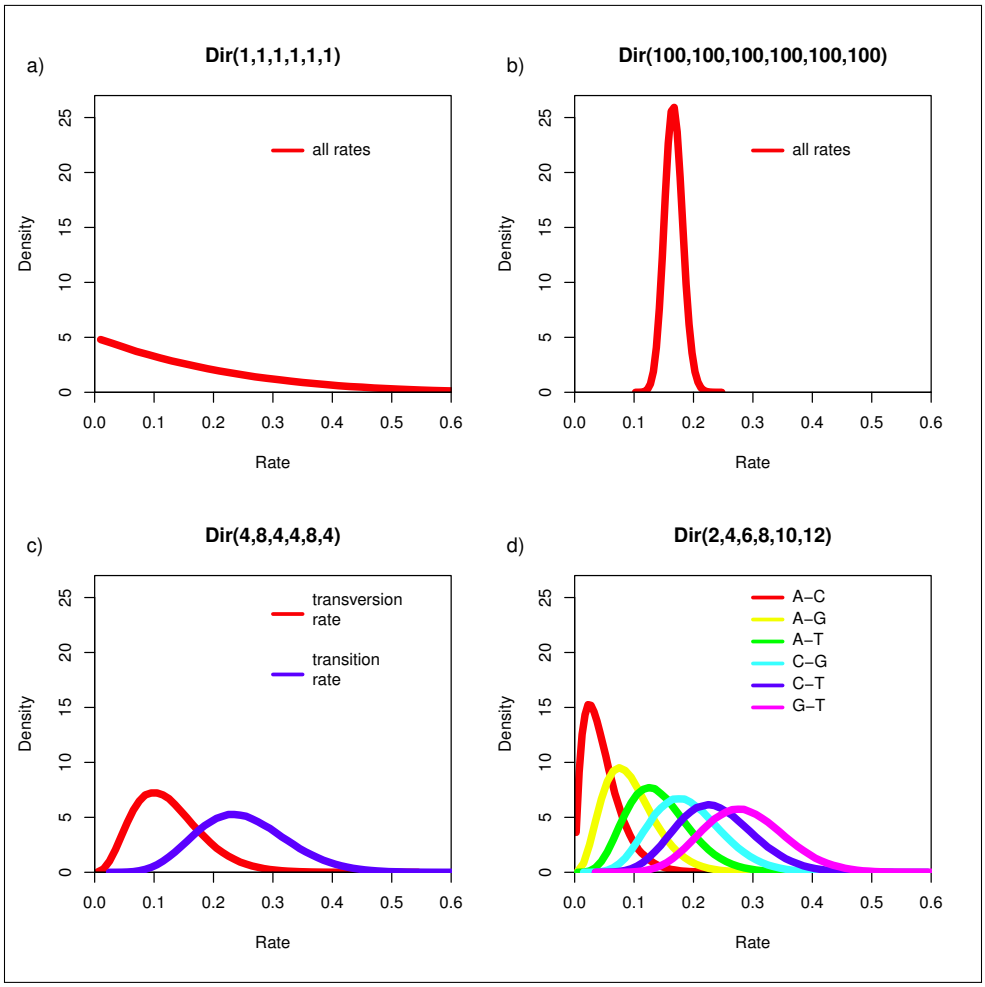

Figure 3: Four different examples of Dirichlet priors on exchangeability rates.

```
moves[++mi] <- mvSimplexElementScale(pi, weight=2)
```

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

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

4.2 Execises

- Use your previous analysis file, either the **`JukesCantor.Rev`** or **`HKY.Rev`** to create a GTR analysis in a new file called **`GTR.Rev`**. Adopt the old analysis to use now the GTR model.
- Run an MCMC to estimate the posterior distribution.

- Complete the table of the phylogenetic relationship of Tarsiers.

5 Gamma-Distributed Among Site Rate Variation

We will also assume that the substitution rates vary among sites according to a 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, $\alpha = \beta$. Therefore, we need only consider the single shape parameter, α (**?**). The degree of among-site substitution rate variation (ASRV) is inversely proportional to the value of the shape parameter—as the value of α-shape parameter increases, the gamma distribution increasingly resembles a normal distribution with decreasing variance, which corresponds to decreasing levels of ASRV (Figure 4). If $\alpha = 1$, then the gamma distribution collapses to an exponential distribution with a rate parameter equal to β. By contrast, when the value of the α-shape parameter is < 1, the gamma distribution assumes a concave distribution that places most of the prior density on low rates but allows some prior mass on sites with very high rates, which corresponds to high levels of ASRV (Figure 4).

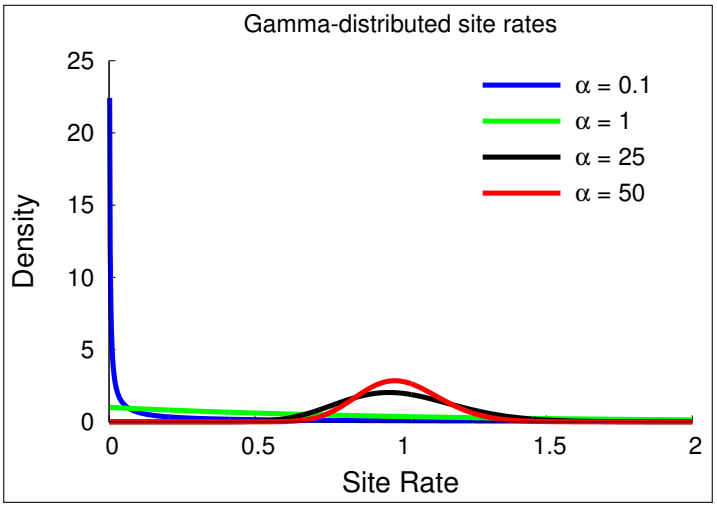

Figure 4: The probability density of mean-one gamma-distributed rates under different shape parameters.

Alternatively, we might not have good prior knowledge about the variance in site rates, thus we can place an uninformative, or diffuse prior on the shape parameter. For this analysis, we will use an exponential distribution with a rate parameter, **shape_prior**, equal to `0.05`. Under an exponential prior, we are placing non-zero probability on values of α ranging from 0 to ∞. The rate parameter, often denoted λ, of an exponential distribution controls both the mean and variance of this prior such that the expected (or mean) value of α is: $\mathbb{E}[\alpha] = \frac{1}{\lambda}$. Thus, if we set $\lambda = 0.05$, then $\mathbb{E}[\alpha] = 20$.

5.1 Setting up the model in RevBayes

Create a constant node called **shape_prior** for the rate parameter of the exponential prior on the gamma-shape parameter

```
alpha_prior <- 0.05
```

Then create a stochastic node called **shape** to represent the α node in Figure **??**, with an exponential density as a prior:

```
alpha ~ dnExponential(alpha_prior)
```

The way the ASRV model is implemented involves discretizing the mean-one gamma distribution into a set number of rate categories. Thus, we can analytically marginalize over the uncertainty in the rate at each site. To do this, we need a deterministic node that is a vector of rates calculated from the gamma distribution and the number of 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 **shape** 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 )
```

The random variable that controls the rate variation is the stochastic node **shape**. We will apply a simple scale move to this parameter.

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

Do not forget that you need to the the **PhyloCTMC** distribution that it has now a new parameter:

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

5.2 Execises

Adopt the GTR analysis to use now the GTR+Gamma model. Run an MCMC 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.
- What is the estimated tree length? Is the estimate different to the previous analysis? What could cause this?
- Complete the table of the phylogenetic relationship of Tarsiers.

6 Modeling invariant sites

The final extension to our model is a substitution model where each site has the probability **pinvar** to be invariant. This means, that with probability $1 - pinvar$ each site is able to evolve.

First, let us 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 site.

So let us specify the model in RevBayes. We need a prior probability for the probability of a site being invariant. A Beta distribution is the common choice for parameters representing probabilities.

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

The **Beta(1,1)** distribution is a flat prior distribution which specifies equal probability to each value between 0 and 1.

Then, as usual, we add a move to change this stochastic variable. A simple sliding window move works here.

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

6.1 Exercises

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

Questions about this tutorial can be directed to:

- Tracy Heath (email: tracyh@berkeley.edu)
- Michael Landis (email: mlandis@berkeley.edu)
- Sebastian Höhna (email: sebastian.hoehna@gmail.com)
- Brian R. Moore (email: brianmoore@ucdavis.edu)

References

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.

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/.

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

Zwickl, D. J. and M. T. Holder. 2004. Model parameterization, prior distributions, and the general time-reversible model in bayesian phylogenetics. Systematic Biology 53:877–888.

Version dated: January 20, 2015