Analyzing BAMM results

Dan Rabosky

November 24, 2013

Contents

1 Getting and using **BAMMtools**

This section contains general instructions on getting BAMMtools . In general, you should simply be able to install the package from the CRAN repository. In R, try the following:

```
> install.packages("BAMMtools")
```

You should also be able to obtain the latest versions of the software via the BAMM github repository. Instructions to follow.

1.1 Following the example code

This document has been written in `Sweave` and `latex`. If you have never followed `R` documentation in this style, there are a few things to be aware of. `R` code is shown in formatted code chunks, like this:

```
> print("hello world")
```

To execute this bit of code, type the commands exactly as they appear after the symbol for the `R` prompt, or >. Occasionally, we will have lines of code that are too long to fit on a single line in this document, like this:

```
> cat("h", "e", "l", "l", "l", "o",
+          "\t", "w", "o", "r", "l", "d", sep="")
```

Here, you are to ignore the + symbol on the second line: it appears solely to tell you (the user) that the line of code you are entering has been split solely to make it fit in this document (and you should thus enter it as a **single** line in your `R` console). In addition, we will occasionally show (in this document) output from entering `R` commands. Here's a simple example where we will call a random number generation function (`rnorm`) and print five random numbers to the `R` console:

```
> rnorm(5)
```

```
[1] -1.2554723 -0.2024789 -1.5818381  0.1426062 -0.6237480
```

The set of 5 numbers that appears after the code snippet (prefaced by `[1]`) is similar to the output that you, the user, should see if you are following the examples.

Finally, this document assumes that you have some knowledge of the directory structure on your computer, such that you can specify appropriate *path* names for `BAMM` input and output files. In general, to avoid confusion, create a new directory in which to explore the examples. Launch `R`, and navigate to this directory. Make sure **ALL** of the files you wish to analyze (example datasets, `BAMM` output files, etc) are located in this directory. Thus, you are always referencing files that are stored within your current working directory. A few quick commands:

```
> getwd()
```

will print your current working directory.

```
> dir()
```

will list all files within your current working directory. If you are having trouble accessing a datafile, and you don't see it with the `dir` function, then (usually) either the file is not where you think it is, or you are not where you think you are.

Finally, comments in `R` are lines of code that begin with the pound symbol, #. Sometimes you may see this within a code snippet, but it is only there for reference:

```
> # This is a comment.
```

2 Notes on the example datasets

The dataset used in the following examples is a time-calibrated phylogeny for whales (cetaceans) from Steeman et al., *Syst. Biol* 58:573-585 (2009).

3 Understanding BAMM output

BAMM generates two primary types of output files.

- **MCMC output files** This file contains standard information associated with the Markov chain Monte Carlo algorithm itself. This includes the log-likelihood of the data, the log prior density of the parameters, and the number of *events* (= distinct macroevolutionary regimes) at a given point in the simulation. This file, by default, will typically be title something like `MCMC_out.txt`, unless you've changed your settings in the control file. If you choose to run BAMM while sampling from the prior only (no likelihoods computed), this file will be generated but will have 0's for the log-likelihoods.

- **Macroevolutionary regime data** The core BAMM results are parameters associated with distinct macroevolutionary rate regimes across a phylogenetic tree. During each stage (or generation) of the MCMC simulation, the Markov chain is characterized by a particular configuration of discrete macroevolutionary rate regimes. Each regime is defined by the occurrence of an *event*, which can also be thought of as a *rate shift* or a change in the fundamental dynamics of the system. Regimes/events are characterized by a set of parameters that define a particular time-varying or time-constant process of speciation, extinction, or trait evolution. All of the data necessary to reconstruct the evolutionary rate configuration across a phylogenetic tree at any point in the MCMC simulation can be reconstructed from knowing the locations of events on the tree and their corresponding rate parameters. The event data, by default, is typically written to a file called `event_data.txt` or equivalent, unless you have changed your default settings. We will explore this further below.

Loading and exploring the MCMC output file:

```
> x <- read.csv("run_full/mcmc_out.txt", header=T);
> head(x)

  generation N_shifts logPrior   logLik eventRate acceptRate
1          1        0 -1.37701 -320.928 1.0000000      0.000
2       1001        0 -2.28461 -278.885 1.8402900      0.439
3       2001        0 -1.30787 -278.157 0.8805340      0.402
4       3001        1 -4.43350 -276.919 1.2191700      0.488
5       4001        1 -1.99887 -275.746 0.0762522      0.394
6       5001        0 -3.20785 -278.123 2.7015000      0.346
```

These parameters are:

- `generation` is the generation of MCMC sampling
- `N_shifts` is the number of non-root macroevolutionary regimes on the tree
- `logPrior` is the log of the prior density of the parameters
- `logLik` is the log-likelihood of the data
- `eventRate` is the rate parameter of the Poisson distribution governing the number of non-root macroevolutionary rate regimes.
- `acceptRate` is the fraction of proposed states that were accepted since the last time the state was written to file.

Now we will look at the primary results file: the `event_data.txt` file. This example is for a speciation-extinction analysis of the *whales* dataset from Steeman *et al.* 2009 (several parameters would be different for a trait evolution analysis, e.g., there would be no speciation *lambda* parameters):

```
> x <- read.csv("run_full/event_data.txt", header=T)
```

And here are the variables (columns) stored in this data table:

```
> colnames(x)

[1] "generation"  "leftchild"   "rightchild"  "abstime"     "lambdainit"
[6] "lambdashift" "muinit"      "mushift"
```

This file contains a set of samples from the posterior. These parameters fully specify the exactly location where each regime begins as well as the evolutionary rate parameters of the regimes. Each rate regime gets its own line, such that a single sample from the posterior in some generation will occupy a number of lines equal to the number of shift regimes. These parameters are:

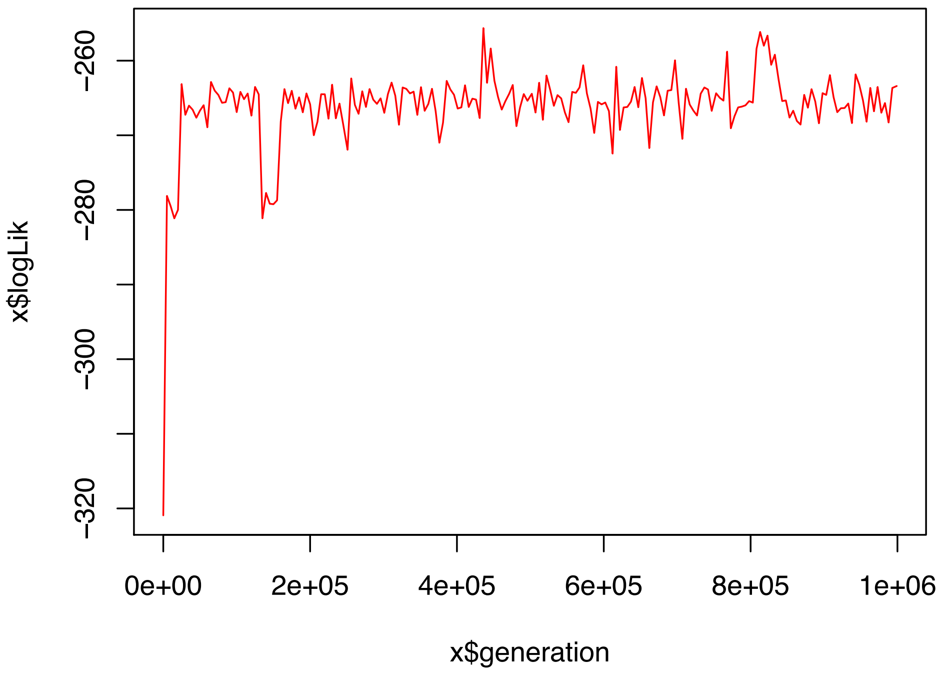
- `generation` The generation from which the sample was drawn
- `leftchild`, `rightchild`, `abstime` These parameters specify the exact location of the start of the regime on the tree. `rightchild` and `leftchild` are random descendants of the right and left branches descended from the node subtending the branch on which the shift event occurs. `abstime` is the exact position in absolute time (0 at the root) when the shift event occurred.
- `lambdainit, lambdashift, muinit, mushift, betainit, betashift` Depending on the specific model, these parameters describe the dynamics of speciation, extinction, or phenotypic evolution through time.

3.1 Diagnosing MCMC convergence

After you've finished a BAMM run, the first thing you'll want to explore is whether or not your run has "converged." MCMC is a beautiful yet simple algorithm for simulating complex probability distributions, but this simplicity comes at a price: we may *never* truly know whether our results have converged on the true the posterior probability density. Fortunately, there are a number of simple checks we can perform that will at least allow us to reject particularly egregious cases of non-convergence.

The easiest (and least reliable) way to check for MCMC convergence is to visually inspect the trace of the log-likelihood as a function of the number of generations. Using the analysis of the whaletree.tre dataset, we see:

```
> x <- read.csv("run_full/mcmc_out.txt", header=T)
> x <- x[seq(1, nrow(x), length.out=200), ]

> plot(x$logLik ~ x$generation, type = "l", col="red", lwd=1);
```



Based on this plot, we can see a tendency for the log-likelihood to stabilize around -265 or so, but you also notice a few period spikes to better values (> -260). This is probably a case where we could have ran the analysis out for a number of additional generations.

A better way to test for convergence is to compute the effective sample sizes associated with the log-likelihood and the number of rate regimes. The *effective size* of a parameter is the estimated number of independent samples that exist once you've controlled for the autocorrelation among successive datapoints. In R we can do this using the coda library, after first throwing out the first 10% as burn-in (using x as our dataframe, as above):

```
> library(coda)
```

Now we subset our dataframe to get rid of the first 10% of samples:

```
> burn <- floor(0.10 * nrow(x))
> x_postburn <- x[ burn: nrow(x), ]
```

And now we use the effectiveSize function on the parameters of interest:

```
> effectiveSize(x_postburn$logLik)

    var1
46.59528

> effectiveSize(x_postburn$logPrior)

    var1
140.8208

> effectiveSize(x_postburn$N_shifts)

    var1
95.09171
```

Generally, it would be nice to see these effective sizes above 200 or so.

Finally, a great way to test for convergence is to perform multiple independent BAMM simulations using different starting parameters. You can then test whether the runs are converging on the same distribution. You can compare the marginal rate distributions across all branches in a phylogeny, something we will explore in section ????

4 What is the best macroevolutionary model?

The first question we generally wish to ask is: how much evidence do we have for multiple evolutionary rate regimes within our data? BAMM is largely designed to answer precisely this question (despite the caveats discussed earlier regarding the interpretation of rate shifts). This is fundamentally a question of model selection, and the reversible jump MCMC algorithm in BAMM makes it relatively straightforward to compare models with different numbers of evolutionary rate regimes.

The simplest model in BAMM is what we will refer to as M0: a model of rank 0, or a model with just a single evolutionary rate regime. This is a model where a single set of evolutionary parameters, starting at the root, can account for the entire pattern of diversification (or trait evolution) across the tree. We will refer to a model MX as a model with X distinct non-root evolutionary rate regimes, such that - for example - model M2 involves two distinct shifts in evolutionary dynamics.

Put simply, **if you cannot reject a model M0, then you have no statistical evidence for heterogeneous among-clade diversification dynamics.** Note that, because BAMM allows rates to vary through time within regimes, you may still have very strong evidence for *temporal* rate variation, even if you have no support for clade-specific rate variation.

4.1 How many distinct macroevolutionary regimes?

The first step of an analysis is to explore the distribution of models visited during the simulation of the posterior by the reversible jump MCMC sampler. Here, we will read in the raw MCMC output generated during the BAMM run, discard some fraction as burn-in, and look at the relative frequency of models visited:

```
> x <- read.csv("run_full/mcmc_out.txt", header=T)
> burn <- floor(0.1 * nrow(x))
> x_postburn <- x[burn:nrow(x), ]
```

Now to tabulate the frequency of models:

```
> table(x_postburn$N_shifts)

  0   1   2   3   4   5
 17 582 240  51   9   2
```

Clearly, we visited models M1 and M2 much more frequently than model M0. This strongly suggests heterogeneous rates among cetacean clades. We can directly compute the posterior probability of a model with just a single evolutionary rate regime (M0), like this:

```
> post_prob_M0 <- sum(x_postburn$N_shifts == 0) / nrow(x_postburn)
> post_prob_M0

[1] 0.01886792
```

This suggests a low posterior probability of a model with just a single evolutionary rate regime. For the whale dataset. In fact, we can compute the posterior probabilities of any particular model MX as follows:

```
> table(x_postburn$N_shifts) / nrow(x_postburn)

          0           1           2           3           4           5
0.018867925 0.645948946 0.266370699 0.056603774 0.009988901 0.002219756
```

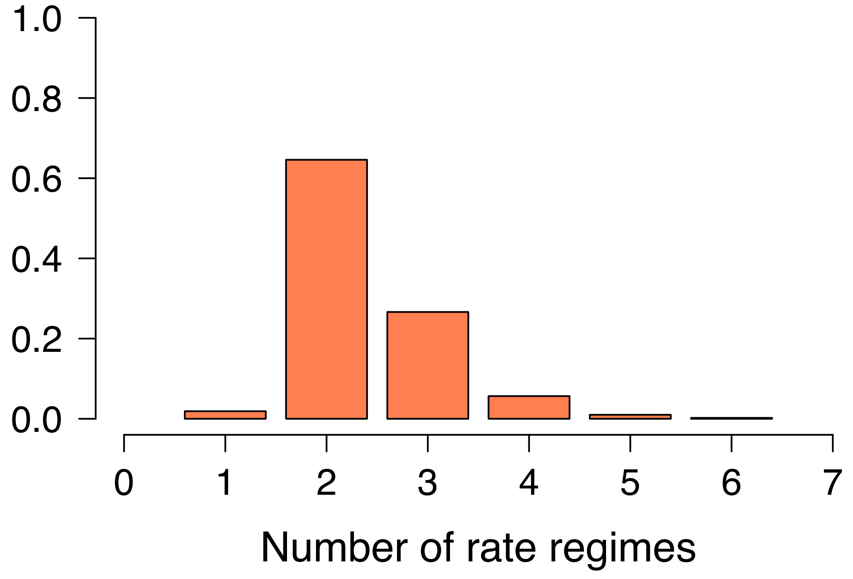
and the ratio of counts or frequencies for any models `Y` and `Z` is equal to the posterior odds ratio favoring one model over another:

```
> counts_1 <- sum(x_postburn$N_shifts == 1)
> counts_0 <- sum(x_postburn$N_shifts == 0)
> post_odds_1v0 <- counts_1 / counts_0
> post_odds_1v0
```

```
[1] 34.23529
```

This is a substantial amount of evidence in favor of `M1` over `M0`. Here we will use a somewhat more involved approach to make a pretty figure, showing the distribution of macroevolutionary rate regimes:

```
> tx <- table(x_postburn$N_shifts) / nrow(x_postburn)
> nn <- as.numeric(names(tx))
> plot.new()
> xwid <- 0.4
> par(mar=c(6,6,1,1))
> plot.window(xlim=c(0, 7), ylim=c(0, 1))
> for (i in 1:length(tx)){
+         xco <- c(i-xwid, i-xwid, i+xwid, i+xwid)
+         yco <- c(0, tx[i], tx[i], 0)
+         polygon(xco, yco, col="coral")
+ }
> axis(1, cex.axis=1.3)
> axis(2, las=1, cex.axis=1.3)
> mtext(side=1, text="Number of rate regimes", cex=1.5, line=3.0)
```



Somewhat confusingly, we are displaying here on the X-axis the **total number** of macroevolutionary rate regimes, including the root process. So, model `M0` is a model with a single macroevolutionary rate regime (or 0 *non-root* rate regimes).

4.2 Estimating Bayes factors

We can do many things with the posterior probabilities and odds ratios discussed above. One of the drawbacks to these estimates is that they are dependent on the prior probabilities of the model. However, the calculation of **Bayes factors** allows us to explicitly compare the relative support for any two models in a manner that is largely independent of the prior. Consider Bayes theorem:

$$P(M_k|\mathbf{D}) = \frac{P(\mathbf{D}|M_k)P(M_k)}{P(\mathbf{D})} \tag{1}$$

where $P(\mathbf{D})$ is the probability of the data, and M_k is a model with k non-root processes. $P(M_k|\mathbf{D})$ is the posterior probability of model M_k, given the data. With `BAMM`, the probability of the data $P(\mathbf{D})$ is identical across all models. Because $P(\mathbf{D})$ is equivalent across all models, we can immediately derive the following expression for the posterior odds ratio between two models i and j:

$$\frac{P(M_i|\mathbf{D})}{P(M_j|\mathbf{D})} = \frac{P(\mathbf{D}|M_i)}{P(\mathbf{D}|M_j)}\frac{P(M_i)}{P(M_j)} \tag{2}$$

The ratio of $P(M_i|\mathbf{D})$ to $P(M_j|\mathbf{D})$ is the posterior odds ratio for models i and j, and $P(M_i)$ over $P(M_j)$ is the prior odds ratio. This leads to the following relationship:

$$\frac{\text{Posterior odds ratio}}{\text{Prior odds ratio}} = \frac{P(\mathbf{D}|M_i)}{P(\mathbf{D}|M_j)} \tag{3}$$

The term on the right-hand side of this equation is the ratio of **marginal likelihoods** of the data under models M_j and M_i and is also known as the **Bayes factor** for models i and j, or BF_{ij}. In principle, this term is independent of the prior odds on any two models and provides a convenient assessement of the relative evidence favoring model i over model j. For most applications, it can be difficult to compute Bayes factors, due to the difficulty in estimating the marginal likelihood of the data under a given model. However, this is straightforward when using reversible jump MCMC. The *posterior odds ratio* emerges immediately from our analysis, and the *prior odds ratio* can be estimated by simply performing a dummy run of `BAMM` where we sample from the prior without computing any likelihoods. `BAMM` has an option to do this, which is specified in the control file. This parameter is `sampleFromPriorOnly`, and will instruct `BAMM` to run an MCMC chain where the acceptance probabilities of all moves are determined strictly by the ratio of prior probability densities on proposed/accepted states (as well as the ratio of transition kernels between states).

To compute the Bayes factor for models, we assume you have an MCMC output file from a `BAMM` run where you sampled only from the prior. BAMMtools provides a handy function for computing pairwise Bayes factors from reversible jump MCMC output. As arguments, you will need to specify the filenames where the regular (full data) and prior-only MCMC output are stored, as well as the percentage of samples to discard as burn-in. Let's load BAMMtools :

```
> library("BAMMtools")
> source("BAMMtools.R")
```

Now, assume we have files `MCMC_out.txt` for our full `BAMM` results, and file `MCMC_prioronly.txt` for a run where we sample only from the prior. We now compute the Bayes factors using the function `computeBayesFactors`:

```
> source("BAMMtools.R")
> postfile <- "run_full/mcmc_out.txt"
> priorfile <- "run_prioronly/prior1_mcmc_out.txt"
> models <- 0:3
> burnin <- 0.1
> results <- computeBayesFactors(postfile, priorfile,
+          burnin, models)
```

Here, `models` is a vector of the models we wish to consider. In this example, we send the function the vector `0:3`, which corresponds to models M_0, M_1, M_2, and M_3. The `results` variable is a pairwise matrix of Bayes factors for the specified models. By default, they are always computed so that the model with the higher rank is the numerator, e.g., M_2/M_1, not M_1/M_2.

```
> #Printing Bayes factor matrix:
> results
```

```
   0       1         2          3
0 NA 38.7037 38.1106950 20.7127883
1 NA      NA  0.9846782  0.5351630
2 NA      NA         NA  0.5434902
3 NA      NA         NA         NA
```

Row and column names give the corresponding model ranks. In general, Bayes factors greater than 5 imply robust support for one model over another. In this example, we have massive support for both M_1 and M_2 over M_0 ($BF > 30$). However, there is very little support for additional complexity beyond a model with a single shift in dynamics. The Bayes factor evidence favoring M_2 versus M_1 is only 0.98, implying no improvement in fit of a 2-shift model relative to a 1-shift model. M_3 fares even worse against M_1, although M_0 is much worse than all models with at least 1 shift.

What if we try to compute Bayes factors for models that are very rarely sampled? Let's see how many times BAMM sampled a model with 10 rate shifts:

```
> sum(x_postburn$N_shifts == 10)
```

```
[1] 0
```

Clearly, we can't very well compute an accurate Bayes Factor involving a model that is sampled so infrequently, and this is where Bayes Factor estimation is highly inaccurate. By default, computeBayesFactors restricts the set of possible models compared to the model set that explains 99.5% of the total probability of the data. If you specify models outside of this range, you will receive a warning message, and the candidate model set will automatically be truncated:

```
> # Now consider models of rank 0:10
> moremodels <- 0:10
> r2 <- computeBayesFactors(postfile, priorfile, burnin, moremodels)
```

```
*****************************************
You have selected to compute Bayes Factors for models
 that were sampled very infrequently and for which
 the Bayes Factors are likely to be (wildly ) inaccurate.
 Consequently, the maximum rank of the models considered
 will be constrained to <<<  3  >>>
*****************************************
```

```
> r2
```

```
   0       1         2          3
0 NA 38.7037 38.1106950 20.7127883
1 NA      NA  0.9846782  0.5351630
2 NA      NA         NA  0.5434902
3 NA      NA         NA         NA
```

In this example, models M_4 through M_{10} were excluded, as the resulting Bayes factors would have low accuracy.

- **General thoughts on rate shifts** As discussed elsewhere in BAMM documentation, excessive focus on the precise number of macroevolutionary rate regimes can be a distraction. My personal opinion is that it is useful in many cases to assess the evidence favoring a single rate regime versus a more complex mixture of rate regimes. But it can be misleading to focus on (say) whether 5 or 6 regimes are **"the"** set of shifts. The user of BAMM (and all other macroevolutionary analysis methods) must remember that the very notion of a discrete shift is part of a statistical model that we have imposed on the data. There is no reason why the true underlying process of diversification should have occured in this fashion.

5 Where are the rate shifts?

In contrast to other methods for macroevolutionary analysis, BAMM does not estimate a single *best* rate-shift configuration. Rather, the underlying philosophy recognizes that many different combinations of shift events can have roughly similar probabilities of explaining the data.

In the worst case, methods that focus on a single, global "best" shift configuration can be highly misleading. Imagine, for example, phylogeny of two clades (A, B), where there is a true probability of 1.0 that one of two events occured: either a rate increase occured on the branch leading to clade A, with probability 0.51; or a rate decrease occured on the branch leading to clade B, with probability 0.49. A stepwise *AIC* approach to finding the global best model might identify a single rate shift, on the branch leading to A. But this is a very misleading view of our true support for this model, because it is very nearly as likely that nothing happened in the ancestor of this clade. In the BAMM framework, these models (shift on A, or shift on B) would be sampled in proportion to their posterior probability, providing us with a much more accurate measure of relative support for any given scenario.

The core of all downstream BAMM analyses is the creation of a BAMM-data object. This is a complex datatype that is generated in R from the raw event data and the phylogeny that was analyzed. Each sample from the posterior is a *rate shift configuration* and also includes the evolutionary parameters associated with the shifts. A BAMM-data object takes all of the event data and maps it to the tree, such that one can easily compute branch-specific diversification rates, or rate-through-time curves, and so on.

Here, we will generate a BAMM-data object for the cetacean analysis, using the function getEventData:

```
> library(ape)
> whaletree <- read.tree("whaletree.tre")
> eventfile <- "run_full/event_data.txt"
> bammdata <- getEventData(whaletree, eventfile,
+         burnin=0.25, nsamples=5)
```

```
Reading event datafile:  run_full/event_data.txt
                ...........
Read a total of  200  samples from posterior

Discarded as burnin: GENERATIONS <  245001
Analyzing  5  samples from posterior

Setting recursive sequence on tree...

Done with recursive sequence
```

```
> class(bammdata)
```

```
[1] "phylo"     "bamm-data"
```

In this example, we are only choosing to analyze 5 samples from the posterior (`nsamples = 5`), after discarding the first 25% as burn-in. In a ”for-real” analysis, this should be considerably higher. For large phylogenies, however, this can be extremely time consuming and you may wish to limit yourself to 200 or fewer samples from the posterior.

The `bamm-data` object is complex and includes a number of attributes, which we won't worry much about (see `?getEventData`, for more information). It also inherits all the attributes of class `phylo`, so all methods that apply to class `phylo` objects can also be applied to `BAMM`. One important attribute with `BAMM-data` objects is the `type` of data object:

```
> bammdata$type
```

```
[1] "diversification"
```

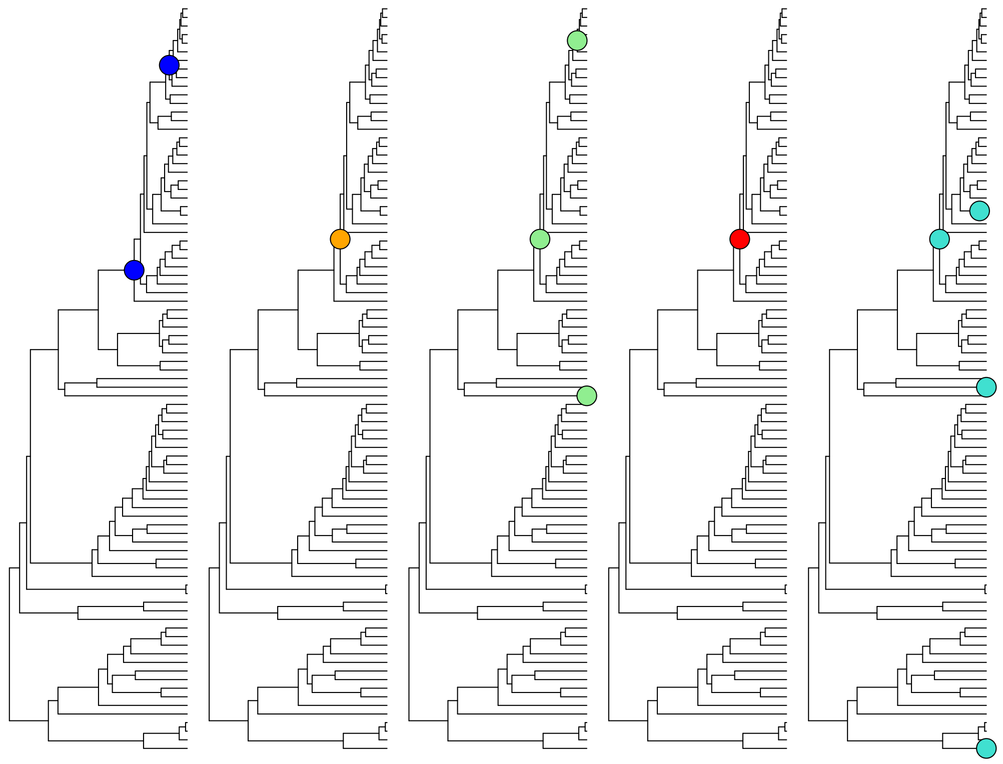
If you have performed a phenotypic evolution analysis, `type` will be `trait`. One important function for working with a `BAMM-data` object is `getShiftNodes`. This function will extract the nodes at which a rate shift occurred for a particular sample from the posterior, specified by the parameter `index`. In our `BAMM-data` object, we only have 5 samples, so we can only access the shift parameters for these 5 samples. To extract the nodes at which shifts occurred for (say) the 2nd sample in the posterior (after discarding burn-in and thinning), you can do:

```
> shifts <- getShiftNodesFromIndex(bammdata, index=2)
> shifts
```

```
[1] 141
```

And `shifts` is now a vector of the `ape` format node numbers (excluding the root) where rate shifts occurred. What if we want to plot these in a single figure? We only have 5 samples from the posterior, so we'll plot all 5 distinct shift configurations in a single plot (with different colors for each):

```
> colorset <- c("blue", "orange", "lightgreen", "red", "turquoise")
> par(mfrow=c(1,5))
> for (i in 1:5){
+         plot.phylo(whaletree, no.margin=T, show.tip.label=F)
+         shifts <- getShiftNodesFromIndex(bammdata, i)
+         nodelabels(pch=21, node=shifts, bg=colorset[i], cex=4)
+ }
```



In this example, we are not plotting the locations of shifts on particular branches, but rather the nodes at the end of the branch on which a shift occurred. You should see here that there is some variation in the number and location of shifts among samples in the posterior. If you see a shift on a tip node, this implies that the shift occurred somewhere along the specified terminal branch. To be clear, the `BAMM-data` object contains all information about precisely where shifts occur on branches, but it requires a little more work to plot these locations.

`BAMM` provides support for three methods of visualizing rate shift configurations:

- **The *marginal shift probability tree***
- **The *cumulative shift probability tree***
- **The *maximum shift credibility tree***

Each of these methods for visualizing shifts is designed to convey different types of information about the distribution of potential shift locations in a phylogenetic tree.

5.1 Marginal shift probability tree

The **marginal shift probability tree** shows the marginal probabilities of rate shifts on all branches in the tree. There are many ways of displaying this information, but the default BAMMtools function to handle this is
`marginalShiftProbsTree`. This function returns a copy of your phylogenetic tree, but where the branch lengths have been drawn proportional to the probability that they contain a rate shift event. Because it is the *marginal* shift probability tree, the shift probabilities for any branch are computed independently of those for all other branches. Formally, each branch length b_i in the marginal shift tree is computed as

$$b_i = \frac{\sum_{k=1}^{S} I_{k,i}}{S} \tag{4}$$

where S is the number of post-burnin samples from the posterior that have been retained for analysis and $I_{k,i}$ is an indicator variable that takes a value of 1 if at least one shift occurs on branch i in sample k (0 otherwise). A value of 1.0 indicates that at least one shift occured on every sample from the posterior that was examined. We will compute the marginal shift tree for the whales dataset, but now we'll use a much larger set of samples from the posterior to compute the branch-specific shift probabilities by recomputing the `BAMM-data` object.

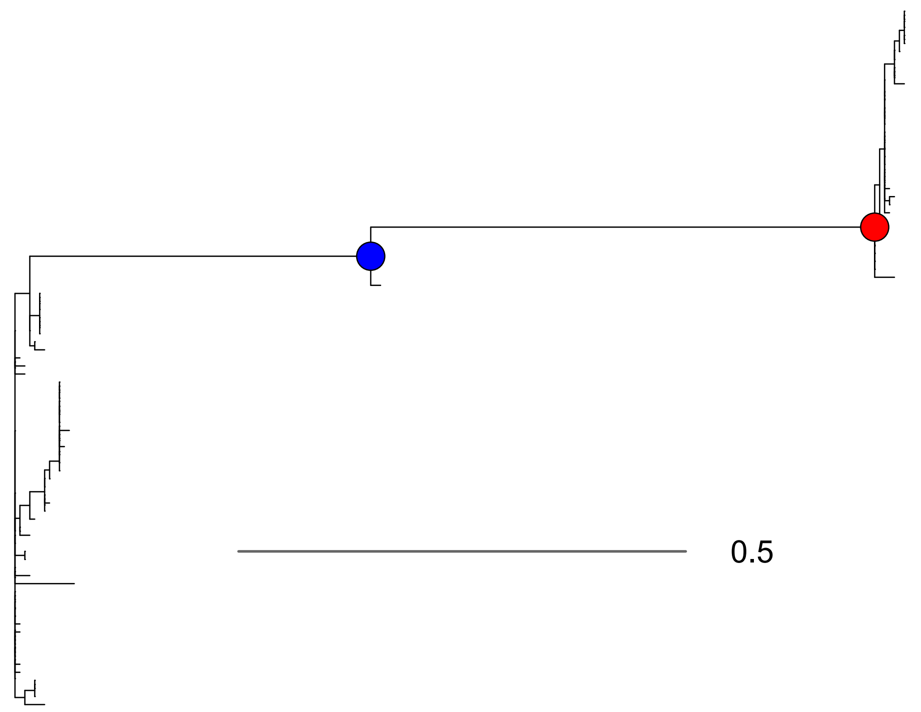
```
> bammdata <- getEventData(whaletree, eventfile, burnin=0.1)
```

And now we compute the `MSP` tree:

```
> msp <- marginalShiftProbsTree(bammdata)
```

And now to plot the tree, with a scale bar to show the probability of a shift on any given branch:

```
> plot(mst, show.tip.label=F, no.margin=T);
> add.scale.bar(length=0.5, x= 0.25, y=20, lcol="gray40",
+           lwd=2, cex=1.5);
> nodelabels(node=141, bg="red", cex=3, pch=21);
> nodelabels(node=140, bg="blue", cex=3, pch=21);
```



The above figures shows the marginal shift probability tree for the cetacean data. Branches are drawn proportional the posterior probability that a rate shift event occurred on the branch. The branch subtending the blue node (shift probability = 0.38) is the Delphinidae (dolphins), and the branch subtending the red node (shift probability 0.56) is the Delphinidae minus the killer whale (*Orcinus orca*).

Note that there are many alternative ways of visualizing the marginal shift probabilities. For example, branches could be colored by the marginal shift probability, and this information could be conveyed on the original time-calibrated phylogeny.

Disadvantages to the marginal shift tree The marginal shift tree suffers from two disadvantages. First, because it displays individual branch-specific shift probabilities, a casual observer might conclude that we have weak evidence for a rate shift, when in fact the evidence for a shift is extremely strong. In the cetacean dataset, no single branch has a shift probability greater than 0.60. However, the probability that a shift occured on at least one of the branches leading to the dolphin clade exceeds 0.97. Visualizing this aspect of shifts is the focus of the next section, the **cumulative shift probability tree**.

A second disadvantage is the the marginal shift tree is exactly that: a *marginal* shift tree. As such, it entirely ignores the covariances between shifts across the tree. In an extreme example, two branches might each have marginal shift probabilities of 0.5, but a reconstruction of their joint marginal density would show that they never occur together (e.g., every sample from the posterior has a shift on branch A or branch B, but never on branches A and B together).

There is a fairly precise analogy to phylogenetics here. If you have performed a Bayesian phylogenetic analysis, you almost certainly cannot draw a single phylogenetic tree that shows all clades that were sampled during the analysis: some clades are incompatible with other clades, and the only way to show them in a single tree is to collapse them to polytomies. For example, consider two topologies $(A, (B, C))$ and $(B, (A, C))$ with equivalent posterior probabilities. It is quite likely that one of these is correct, but the two trees are incompatible. Diversification shifts can be similar, something that we will continue to explore through this documentation. The marginal shift tree is rather like a consensus tree, in which incompatible diversification configurations have been collapsed down to marginal probability estimates on individual branches.

5.2 Cumulative shift probability tree

The **cumulative shift shift probability tree** shows the probability that a given node has evolutionary rate dynamics that are decoupled from the root process. For a given node v to be decoupled from the "background" evolutionary dynamic, a rate shift must occur somewhere on the path between v and the root node. Branches with a cumulative shift probability of 1.0 imply that every sample in the posterior shows at least one rate shift between the focal branch and the root of the tree, leading to evolutionary dynamics that are decoupled from the background process. In the case of the cetacean data (see above), we expect the cumulative shift probability for all the branches in the dolphin clade to be quite high: even though we have relatively low confidence of the precise branch on which a shift may have occurred, we have high confidence that a shift occurred on one of the ancestral branches leading to the clade. Formally, the cumulative shift probability for branch b_i is computed as:

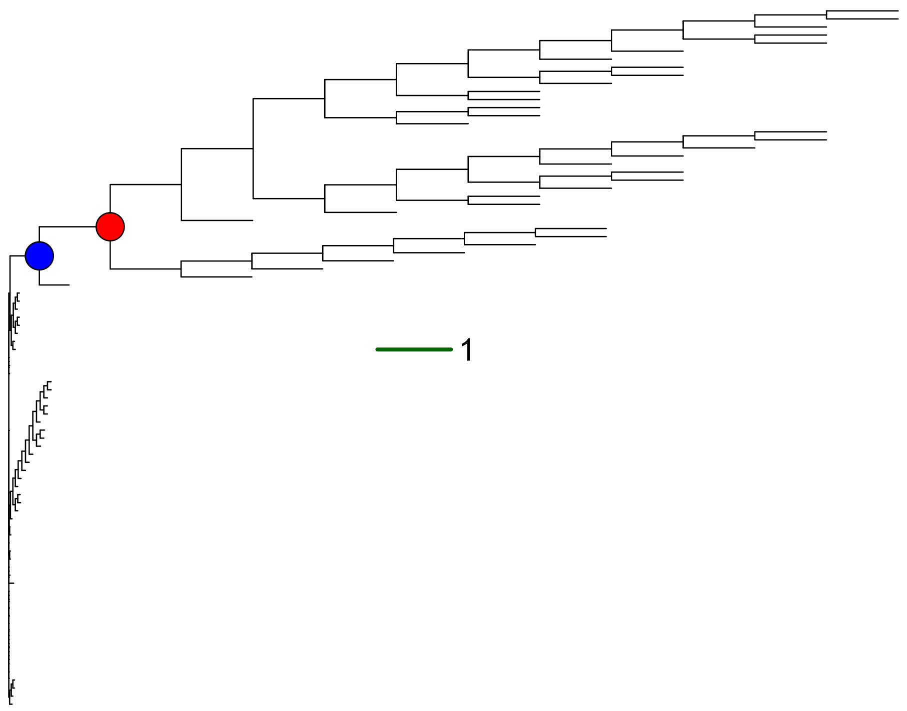
$$b_i = \frac{\sum_{k=1}^{S} \Phi_{k,i}}{N} \tag{5}$$

where $\Phi_{k,i}$ is an indicator variable that takes a value of 1 if a shift occurs somewhere on the path between branch i and the root of the tree (0 otherwise). The cumulative shift probability on a particular branch b_i might thus be extremely high even if shifts are unlikely to have occurred on branch b_i itself. Here we will compute the marginal shift probability tree using the `cumulativeShiftProbsTree` function from BAMMtools .

```
> cst <-cumulativeShiftProbsTree(bammdata)
```

And to plot (and adding a scale bar to allow interpretation of branch probabilities):

```
> quartz.options(height=6, width=8, dpi=72);
> # The above line is operating-system specific
> #        You need another graphics command on windows,
> #        or just ignore the line altogether
> plot(cst, show.tip.label=F, no.margin=T);
> nodelabels(node=141, bg="red", cex=3, pch=21);
> nodelabels(node=140, bg="blue", cex=3, pch=21);
> add.scale.bar(length=1.0, x= 5, y=45, lcol="darkgreen",
+          lwd=3, cex=1.5);
```



In this tree, we have identified exactly the same nodes as in the marginal shift probability tree (blue and red). However, branch lengths in this tree clearly indicate that evolutionary rate dynamics for branches *downstream* of the MRCA of the dolphins (blue node) are decoupled from the background rate dynamics. Branches with low comulative shift probabilities (e.g, all the lineages in the lower half of the figure) have evolutionary rate dynamics that are mostly identical to the background rate.

5.3 Maximum shift credibility tree

In Bayesian phylogenetic analyses, researchers often report the *maximum clade credibility (MCC) tree* rather than some overall consensus tree. The reason for this is that the the MCC tree is a tree topology that is actually part of the posterior distribution simulated via MCMC. When we construct consensus trees from posterior distributions of topologies, it is possible that the resulting consensus topology may look somewhat unlike any of the individual topologies from the posterior. The MCC tree is the topology that was sampled during simulation of the posterior but which maximizes some *a posteriori* optimality criterion, such as the product of clade probabilities.

In BAMM, we are able to compute a parallel tree, which we refer to as the *maximum shift credibility (MSC) tree*. This is the diversification shift configuration that, under the assumptions of the model, we believe is most likely to reflect the true underlying process of diversification. The salient point is that the shift configuration identified as the MSC configuration was indeed sampled during simulation of the posterior. To compute the MSC tree, we find the shift configuration(s) that were sampled by MCMC and that maximimizes the product of branch-specific marginal probabilities. Let p_i denote the marginal probability of a rate shift on the $i'th$ branch. The shift credibility score C for the $k'th$ sample from the posterior is computed as:

$$C = \prod_{i=1}^{N} p_i^{I_{i,k}} (1 - p_i)^{1 - I_{i,k}} \tag{6}$$

where $I_{i,k}$ is an indicator variable taking a value of 1 if a shift occurs on branch i for sample k, and 0 if no shift occurs in the sample.

We will now compute the MSC tree for the cetacean analysis:

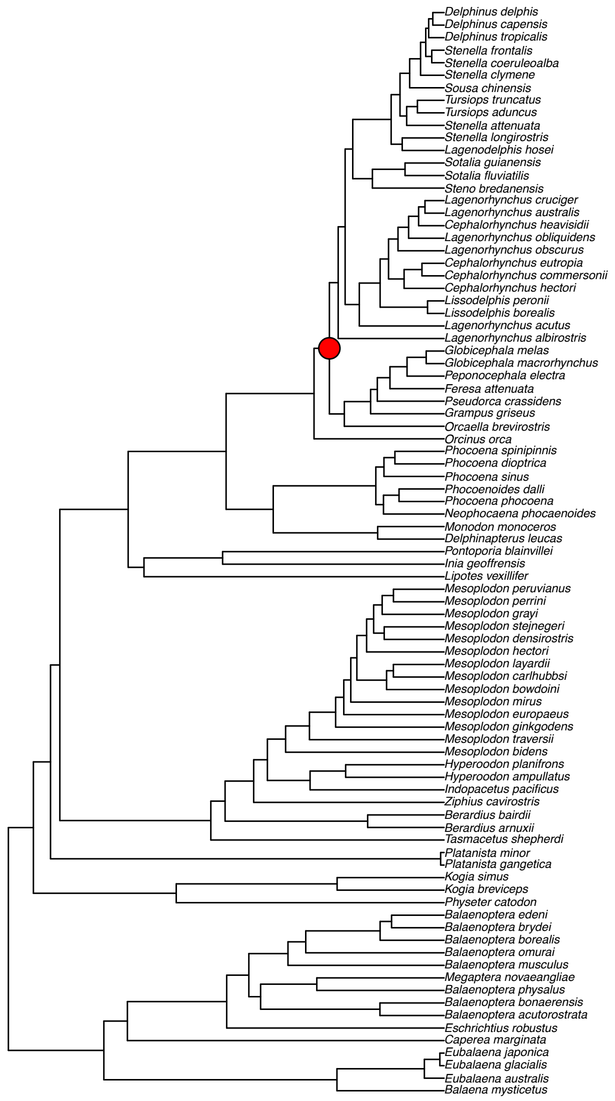
```
> msctree <- maximumShiftCredibilityTree(bammdata)
```

The function `maximumShiftCredibilityTree` returns a vector of `scores`, with the log-probability of each sample in the posterior. Because the best shift configuration may have been sampled many times, it also returns a a vector of indices `bestconfigs` giving the set of indices to samples in the posterior with identical (maximum) shift credibility scores. Finally, the value `sampleindex` gives a single sample from the posterior that can be taken as a representative MSC configuration. Looking at the results for our toy analysis with the cetaceans:

```
> msctree

$bestconfigs
$bestconfigs[[1]]
[1] 2 4



$scores
[1] -5.944031 -1.785148 -4.557737 -1.785148 -5.944031

$optimalityType
[1] "product"

$sampleindex
[1] 2
```

And we can now extract the shift configuration from `msctree$sampleindex` and plot:

```
> msc_sample <- msctree$sampleindex
> shiftnodes <- getShiftNodesFromIndex(bammdata, msc_sample)
> plot.phylo(whaletree, type = "p", cex=0.5)
> nodelabels(node=shiftnodes, pch=21, bg="red", cex=2)
```



In this example, the MSC tree shows a shift in the common ancestor of the dolphin clade, excluding the killer whale *Orcinus orca*.

6 Computing clade-specific rates

Users will often want to estimate clade-specific rates of speciation, extinction, or trait evolution. Because BAMM allows rates to vary through time, it is not necessarily true that a mean clade rate is particularly meaningful: how do you compare rates for two clades if both have undergone a strong temporal deceleration in the rate of speciation, for example? Leaving this question aside for the moment, BAMM provides tools to easily estimate mean rates for individual clades. Formally, the mean speciation (or other) rate for a clade z is computed as

$$\hat{\lambda}_z = \frac{1}{T_z}\sum_{i=1}^{N_z}\int_{t_{i,0}}^{t_{i,1}} \lambda(t)dt \qquad (7)$$

where T_z is the total sum of branch lengths in clade z, N_z is the number of branches in clade z, $t_{i,0}$ is the start time of the $i'th$ branch, and $t_{i,1}$ is the end time of the $i'th$ branch.

For starters, let's compute the mean speciation rate across all whales. This is easy, with our BAMM-data object that we've already computed:

```
> rates_all <- getCladeRates(bammdata)
```

which, for a speciation-extinction analysis, includes the mean rate estimate for each sample in the posterior (for both speciation, λ, and extinction, μ):

```
> names(rates_all)

[1] "lambda" "mu"

> mean(rates_all$lambda)

[1] 0.1384465

> mean(rates_all$mu)

[1] 0.07043847
```

We could use the actual vectors of rates to plot a histogram of the posterior density of rates (we will do this below).

Now, what if we just want the rates for the dolphin clade? We can send getMeanRates the argument node, which will specify that rates be computed only for the clade descended from the target node. If you don't know the node number, you can get these by just plotting your original tree and then calling ape's nodelabels function. Or you directly find the MRCA of spanning taxa, like this:

```
> tax1 <- "Orcinus_orca"
> tax2 <- "Delphinus_delphis"
> tipnode1 <- which(whaletree$tip.label == tax1)
> tipnode2 <- which(whaletree$tip.label == tax2)
> dolphin_node <- getMRCA(whaletree, c(tipnode1, tipnode2))
```

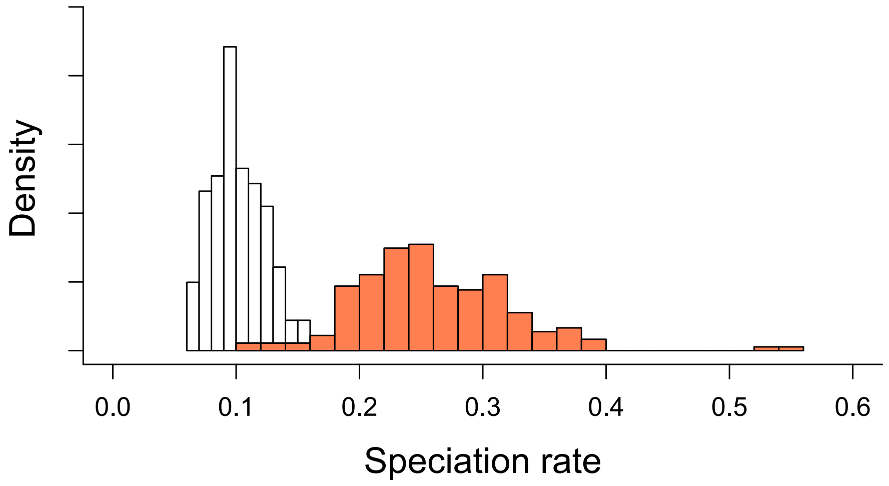
Now we use `getCladeRates`:

```
> rates_dolphins <- getCladeRates(bammdata, node= dolphin_node)
> mean(rates_dolphins$lambda)

[1] 0.2579278
```

This rate is much higher than that which we estimated for the whole tree. Of course, the whole tree rate *included* the dolphins, so we really should be comparing the dolphin rate to the background rate. This can also be justifed on the grounds that we've already uncovered strong evidence that evolutionary rates in the dolphins are decoupled from the background rate (see preceding sections). We will *exclude* the dolphin clade from the analysis and compute the background rate of speciation. We'll do this using the same function, but we'll pass the argument `nodetype`, which will let us exclude all the descendants of a particular node:

```
> rates_background <- getCladeRates(bammdata,
+          node= dolphin_node, nodetype="exclude")
> mean(rates_background$lambda)

[1] 0.1094651
```

We won't show the code for plotting the frequency distributions of rates here, but this is straightforward (see online help for `BAMM` for examples):



Red is the posterior distribution of speciation rates for the dolphin clade, and white is the posterior distribution of rates for the background lineages (e.g., excluding the dolphins).

7 Estimating rates on individual branches

It is straightforward to use your `BAMM-data` object to compute marginal posterior densities of evolutionary rates on every branch of a phylogenetic tree.

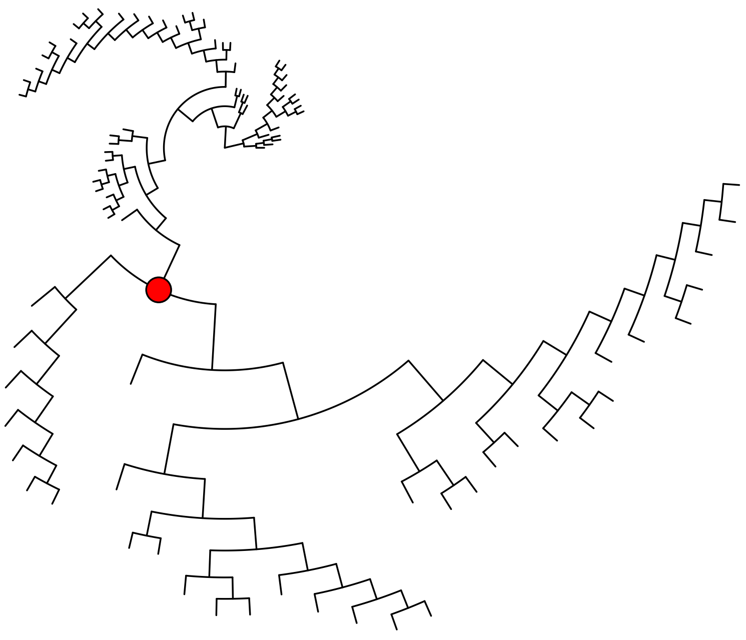
```
> marginal_rates <- getMarginalBranchRateMatrix(bammdata)
```

This gives us (for speciation-extinction analyses) a list with two matrices: `lambda_branch_matrix`, and `mu_branch_matrix`. These matrices contain the marginal rate estimates for each branch (rows), for each sample in our posterior (columns). The matrix rows are ordered to correspond exactly to the order of the `edge.length` component of our phylogeny. Thus, we can compute the mean of this matrix, then swap these vectors in for the `edge.length` attribute of our phylogeny, and generate a plottable "rate-tree":

```
> lam_rates <- rowMeans(marginal_rates$lambda_branch_matrix)
> lambda_tree <- whaletree
> lambda_tree$edge.length <- lam_rates
```

Now we'll plot the tree, with branch lengths scaled by reconstructed rate. Here we are doing this for speciation rates, but it is easy enough to do the same for rates of extinction or trait evolution. We'll also add the reconstructed shift node from the maximum shift credibility tree with a red circle:

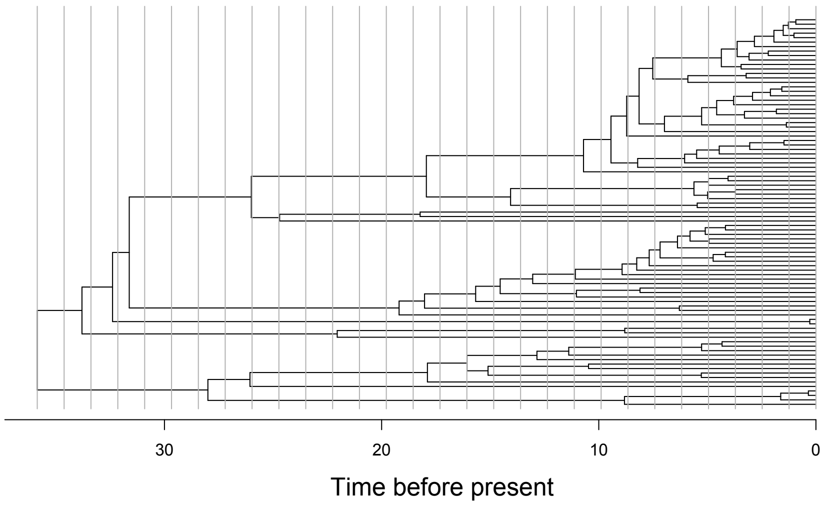
```
> plot.phylo(lambda_tree, type="f", show.tip.label=F);
```



We can visualize branch rates any number of ways. An obvious alternative is to plot the time-calibrated phylogeny but to then color individuals branches by their estimated rate (e.g., the mean of the branch-specific marginal rate distribution).

8 Plotting rate-through-time curves

We can also compute rate-through-time trajectories, in order to visualize temporal changes in the dynamics of speciation, extinction, and trait evolution through time. BAMMtools provides several handy functions for extracting rate-through-time trajectories. The function `getRateThroughTimeMatrix` generates a matrix of mean rates through time. Mean rates are computed by draping an imaginary grid over the phylogeny, as follows:



and the evolutionary rates are computed for each point where a vertical line (the timepoints) intersect a branch. The mean of this vector of rates is the mean rate for the time point. `getRateThroughTimeMatrix` will compute rates in this fashion for every sample in the posterior (as included in the `BAMM-data` object), and return an S by k matrix, where S is the number of samples in the posterior and k is the number of timepoints (which can be specified by the user). As for `getCladeRates`, we can compute rate-through-time matrices for individual clades, and we can exclude clades. To exclude clades and compute background rates, we simply specify `nodetype = exclude` as an argument to the function. Here's the full rate-through-time matrix:

```
> rtt.all <- getRateThroughTimeMatrix(bammdata)
```

which has the following attributes:

```
> class(rtt.all)

[1] "bamm-ratematrix"

> names(rtt.all)

[1] "lambda" "mu"     "times"  "type"
```

The `times` component of the result is a vector of timepoints at which the rates were computed. We can sweep out the mean rates at our timepoints like this:

```
> lamvec <- colMeans(rtt.all$lambda)
```

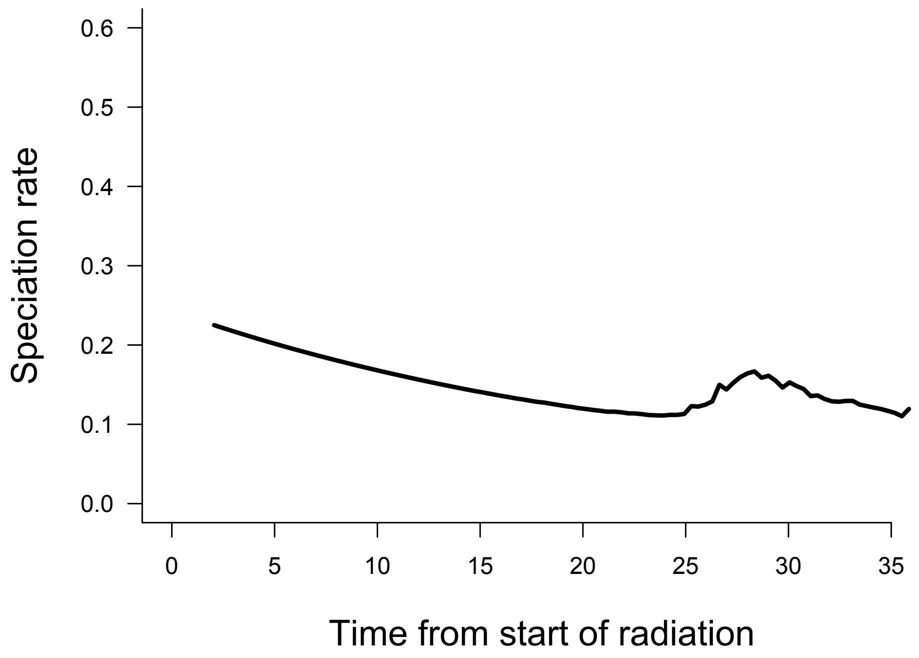
And we could also sweep out other summary statistics. For example, if you wanted the 5% and 95% quantiles on the distribution of estimated rates at any point in time, you can do:

```
> lam_q90 <- apply(rtt.all$lambda, MARGIN=2,
+          quantile, c(0.05, 0.95))
```

Here we'll plot the mean speciation rate through time curve for the entire cetacean radiation. We'll compute the mean rates at each time point, then set up a plot highly-customizable plot window, then add our rate-through- time curve:

```
> lam_rates.all <- colMeans(rtt.all$lambda)
> plot.new()
> par(mar=c(6, 6, 1,1))
> plot.window(xlim=c(0, 36), ylim=c(0, 0.6))
> lines(x=rtt.all$times, y=lam_rates.all, lwd=3, col="black")
> axis(1, at=seq(-5, 35, by=5));
> axis(2, at=seq(-0.1, 0.7, by=0.1), las=1)
> mtext(side=1, text="Time from start of radiation",
+          line =3.5, cex=1.5)
> mtext(side=2, text="Speciation rate", line =3.5, cex=1.5)
```

And the resulting plot for all cetaceans:



And of course, we could do exactly the same for speciation or for trait evolution. In this case, you can clearly see the rise in speciation rates at approximately 27 million years after the radiation start, driven by the burst of speciation in the dolphin clade.

Given that we already have strong evidence for diversification rate heterogeneity across this radiation, it's a little inaccurate to show a single rate-through-time curve for the group. It makes much more sense to show a rate for the dolphin clade separate from the background rate. Now, recall that we already identified the MRCA node for the dolphin clade (variable = `dolphin_node`). Now we'll compute rate-through-time matrices for the dolphins and non-dolphins:

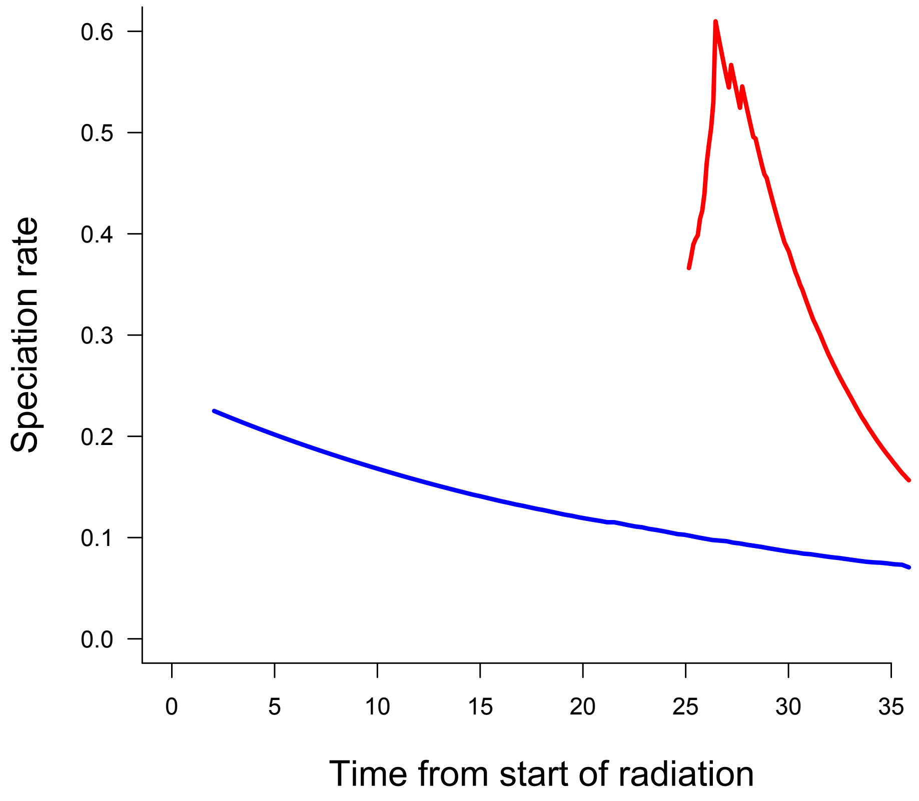
```
> rtt.dolphin <- getRateThroughTimeMatrix(bammdata,
+                              node=dolphin_node)
> rtt.background <- getRateThroughTimeMatrix(bammdata,
+                              node=dolphin_node, nodetype="exclude")
```

Now to sweep out the mean speciation rates from each of these objects:

```
> lam_rates.dolphin <- colMeans(rtt.dolphin$lambda)
> lam_rates.background <- colMeans(rtt.background$lambda)
```

And finally we'll plot everything together: dolphins in red, background in blue:

```
> plot.new()
> par(mar=c(6, 6, 1,1))
> plot.window(xlim=c(0, 36), ylim=c(0, 0.6))
> lines(x=rtt.dolphin$times, y=lam_rates.dolphin,
+         lwd=3, col="red")
> lines(x=rtt.background$times,
+         y=lam_rates.background, lwd=3, col="blue")
> axis(1, at=seq(-5, 35, by=5));
> axis(2, at=seq(-0.1, 0.7, by=0.1), las=1)
> mtext(side=1, text="Time from start of radiation",
+         line =3.5, cex=1.5)
> mtext(side=2, text="Speciation rate", line =3.5, cex=1.5)
```



This portrays a much different picture of rate-through-time dynamics. There is no evidence for anything interesting going on in the background lineages, other than a gradual temporal slowdown in speciation. But the origin of the dolphin clade is associated with a massive spike in speciation, followed by a pronounced deceleration.