MrBayes 3.2.2 Primer
(with a focus on model selection)

Jeremy M. Brown
Dept. of Biological Sciences
Louisiana State University
www.phyleauxgenetics.org
jembrown@lsu.edu
@jembrown

Bodega Bay Applied Phylogenetics Workshop
3.9.14

From the MrBayes webpage (http://mrbayes.sourceforge.net):

MrBayes is a program for the Bayesian estimation of phylogeny. Bayesian inference of phylogeny is based upon a quantity called the posterior probability distribution of trees, which is the probability of a tree conditioned on the observations. The conditioning is accomplished using Bayes's theorem. The posterior probability distribution of trees is impossible to calculate analytically; instead, MrBayes uses a simulation technique called Markov chain Monte Carlo (or MCMC) to approximate the posterior probabilities of trees.

This tutorial will occasionally assume a Unix-based operating system, but should generally work for PCs. Lines preceded by `$` represent Unix commands.

Installation

MrBayes can be downloaded from the MrBayes website (http://mrbayes.sourceforge.net). Pre-compiled, executable versions of MrBayes for PCs or Macs are available for direct download. Both serial and parallel versions are available. Parallel versions allow a single analysis to take advantage of multiple processors, if they are available.

The source code can also be downloaded and compiled locally for use on Unix systems or for command-line use on other systems. Command-line versions will facilitate batch and cluster-based analyses. Instructions for compilation are given on the website.

Starting MrBayes

If you have downloaded a pre-compiled executable version of MrBayes, simply double-click on the icon to start MrBayes. If you have compiled the source code locally, you can execute MrBayes from the command line (e.g.,

```
$ ./mb
```

in Unix), once you have set your working directory properly. To be able to start MrBayes from anywhere without having to recall where it's installed each time, you can either modify your PATH variable (see here: http://kb.iu.edu/data/acar.html) to include your MrBayes directory or you can move the MrBayes executable to a directory that's already in your PATH (e.g., /usr/local/bin/). The latter can be accomplished (if you have administrator privileges), by typing:

```
$ sudo mv ./mb /usr/local/bin/
```

if you are in the directory where MrBayes is installed.

Once MrBayes has started, it should print out an intro splash screen and then wait for your commands at the command prompt:

```
MrBayes >
```

Getting Help

The single most important MrBayes command is *help*. Simply typing *help* at the MrBayes command prompt,

```
MrBayes > help
```

will provide a list of the possible MrBayes commands and a brief description of their purpose. If you would like more information on a particular command, including a list of its associated parameters and their current settings, use *help <command_name>* where *<command_name>* should be replaced by the name of the command in which you are interested. For example, to get help on the *execute* command, type:

```
MrBayes > help execute
```

Loading and Manipulating Data

The first command that you will need to use in MrBayes is *execute*. This command loads information (either a data matrix, a set of trees, or a set of MrBayes commands) from a file. The file (e.g., chr1_1057.nex) should be in the folder where you were when you started MrBayes. Type:

```
MrBayes > execute chr1_1057.nex
```

The *execute* command can be used identically with a file containing MrBayes commands. If this file contains all the commands needed for an analysis, one could simply open MrBayes and execute this command file at the command prompt. You could also run MrBayes with a command file by specifying the file name (e.g., mb_cmds.nex) on the command line when executing MrBayes. For instance, in Unix:

```
$ mb mb_cmds.nex
```

Once data has been loaded into MrBayes, sites can be excluded from an analysis using the *exclude* command. Sites are designated for exclusion based on the number corresponding to their column in an alignment. For instance, to exclude sites 90-99, type:

```
MrBayes > exclude 90-99
```

If sites need to be re-included, use the *include* command:

```
MrBayes > include 90-99
```

Often, one might want to include/exclude every *n*-th character. For instance, defining a character set that starts with the first character and includes every 3rd character would circumscribe all 1st codon positions. This type of inclusion/exclusion is accomplished by using the notation \3 at the end of the character list. For example, to exclude all 1st codon positions, type:

```
MrBayes > exclude 1-. \3
```

The `"."` symbol is used to mean the last position in our data set. To

exclude all 2nd codon positions, use a very similar command, but start the character list with the 2nd site:

```
MrBayes > exclude 2-. \3
```

The inclusion/exclusion status of all sites in our dataset can be viewed using the *charstat* command.

Specifying a Model of Sequence Evolution

Basic Model Manipulations

Most model specification is done using the *lset* command. The most common model specifications performed in *lset* involve the number of substitution types and the model of rate variation across sites. For instance, the number of unique substitution rates can be specified using the *nst* parameter. Possible values for *nst* are 1 (as in the Jukes-Cantor model), 2 (as in the HKY model), and 6 (as in the GTR model). Possible models of rate variation specified with the *rates* parameter include a proportion of invariable sites (I), gamma-distributed rates across sites (G), or a combination of the two (I+G). For example, you could specify a GTR+G model by typing:

```
MrBayes > lset nst=6 rates=gamma
```

In the newest version of MrBayes (3.2.2), one can also avoid having to specify only one scheme of substitution types by allowing MrBayes to move across different schemes as part of its MCMC sampling. This procedure is known as reversible jump MCMC (RJ-MCMC). To set up reversible jumping, use the following command:

```
MrBayes > lset nst=mixed rates=gamma
```

Reversible jumping is not currently set up for different models of rate variation across sites, so you will still need to specify +I, +G, or +I+G. While I will not provide detailed instructions here on analyses with doublet and codon models in MrBayes, these models can also be specified using the *lset* command. See the MrBayes manual for more information.
By default, MrBayes allows the stationary frequencies of the four

nucleotides to be estimated from the data. To fix these frequencies at particular values (often 0.25 for all four nucleotides), use the *prset* command. MrBayes considers the fixation of base frequencies to be a prior setting, rather than a setting of the likelihood model (these distinctions are somewhat arbitrary). For instance, to fix all frequencies to be equal, type:

```
MrBayes > prset statefreqpr=fixed(equal)
```

Partitioning

It is now widely recognized that the evolutionary process has not been homogeneous across sites. One approach to accommodating heterogeneity in the evolutionary process across sites is to divide sites into distinct subsets (a process called partitioning) and model the evolution of each subset using an independent Markov model of nucleotide substitution. The first step in performing a partitioned analysis is to define the distinct subsets of your data. These definitions are made using the *charset* command. The syntax for specifying sites with *charset* is the same as with the *include* and *exclude* commands. For instance, to specify a subset corresponding to the first codon position, type:

```
MrBayes > charset cod.pos.1 = 1-. \3
```

Subsets corresponding to codon positions 2 and 3 could be similarly specified as:

```
MrBayes > charset cod.pos.2 = 2-. \3
MrBayes > charset cod.pos.3 = 3-. \3
```

Note that each site must be assigned to one and only one subset to perform a partitioned analysis.

Once all of the appropriate character sets have been defined, they need to be explicitly combined into a partition. Appropriately, this is done with a command called *partition*. For instance, to specify a partitioning scheme with 3 subsets corresponding to the three codon positions, type

```
MrBayes > partition cod.pos = 3:cod.pos.1,cod.pos.2,cod.pos.3
```

The number immediately preceding the ":" gives the number of subsets in the partioning scheme. After defining a partitioning scheme, you need to tell MrBayes that you actually want to use that scheme. This is done with the *set* command. So, to use the "`cod.pos`" partitioning scheme, type

```
MrBayes > set partition=cod.pos
```

Now that the partitioning scheme is all set, you need to define models of evolution for each subset. As above, this is done with the *lset* command. The one additional consideration is that you need to tell MrBayes to which subsets you want any particular *lset* call to apply. To apply one *lset* command to all subsets within a partitioning scheme, type

```
MrBayes > lset applyto=(all) nst=6 rates=gamma
```

If you want to apply different *lset* calls to different partitions, use the numbers corresponding to the subset (defined by the order in which the subsets are listed in the partition definition). So, to apply an HKY model to codon positions 1 & 2 and a GTR+G model to codon position 3, type

```
MrBayes > lset applyto=(1,2) nst=2
MrBayes > lset applyto=(3) nst=6 rates=gamma
```

After defining the appropriate models, you need to explicitly tell MrBayes that parameters (some or all) need to be estimated independently (or "`unlinked`") for each partition. The "`unlinking`" of parameters across parititions is accomplished with the *unlink* command

```
MrBayes > unlink revmat=(all) tratio=(all) statefreq=(all) shape=(all)
```

Each parameter (or related set of parameters) needs to be unlinked individually. "`revmat`" corresponds to the relative rates of substitution, "`tratio`" corresponds to the transition/transversion rate ratio, "`statefreq`" corresponds to the stationary nucleotide frequencies, and "`shape`" corresponds to the alpha shape parameter of the gamma distribution that models rate variation across sites. You may also need to unlink the proportion of invariable sites across partitions if they're included in your models. To check that you've unlinked parameters appropriately

across partitions, use the *showmodel* command.

It is also possible to allow the overall tree length to be estimated independently for each subset, through the use of independent "`scaling`" parameters. By default, these scaling parameters are linked across partitions. Unlike the unlinking of other parameters, the scaling parameters are unlinked using *prset* by typing

```
MrBayes > prset ratepr=variable
```

Note that the use of rate multipliers across partitions still assumes that the relative branch lengths in the tree are identical across partitions. If, instead, one wants to model different relative branch lengths across partitions, *every* branch length can be estimated independently across partitions. The unlinking of every branch length across partitions is done with *unlink*

```
MrBayes > unlink brlens=(all)
```

Unlink commands can be reversed using the *link* command, with identical syntax.

Specifying Priors on Model Parameters

Priors for nearly all parameters (i.e., relative rates of substitution, base frequencies, branch lengths, etc) can be set using the *prset* command. The available prior distributions vary by parameter type. Lots of information on possible distributions can be gained from *help prset*. As one example, a more informative prior around equal base frequencies (but not fixed at equal frequencies) can be set by increasing the values of the Dirichlet parameters

```
MrBayes > prset applyto=(all) statefreqpr=Dirichlet(100,100,100,100)
```

One particularly important prior to consider is the branch-length prior. This prior is important because this single distribution is applied to *all* the branch lengths in the tree, which can be a very large number of independent parameters. This is a particularly good prior to try testing for prior sensitivity, as the default prior setting in MrBayes has been shown to cause severe problems with branch-length estimation (Marshall 2010, Brown et al.

2010). In other words, run multiple analyses with different branch-length priors and look for changes in the resulting posteriors. Also note that the parameter specified for the exponential prior on branch lengths is the *rate* of the exponential distribution, which is the inverse of its mean. To specify an exponential prior that is pushed up more tightly around small branch lengths, you should specify a *larger* rate parameter. For instance,

```
MrBayes > prset brlenspr=Unconstrained:Exp(100)
```

- Exponential prior on branch lengths with a mean of 0.01.

```
MrBayes > prset brlenspr=Unconstrained:Exp(10)
```

- Exponential prior on branch lengths with a mean of 0.1. (Default)

```
MrBayes > prset brlenspr=Unconstrained:Exp(1)
```

- Exponential prior on branch lengths with a mean of 1.

It has also been shown that the branch-length prior specification can affect the inferred posterior probabilities of the topology (Yang and Rannala 2005). Therefore, when performing prior sensitivity analyses with different branch-length priors, it might be important to monitor sensitivity to inferred branch lengths and topologies.

Manipulating Markov chain Monte Carlo (MCMC) Settings

There are a huge number of potential settings pertaining to the Markov chain Monte Carlo (MCMC) analysis that can be manipulated in MrBayes. Most of these changes are made using the *mcmcp* and *mcmc* commands. These commands differ only in whether or not they begin the Markov chain when the command is issued. *mcmcp* just sets the parameter values *without* starting the analysis. *mcmc* will begin the analysis after setting the values of parameters. Some of the MCMC parameters that can be set using these commands include:

ngen - The number of MCMC generations MrBayes will run before pausing
nruns - The number of independent MCMC analyses run by MrBayes
nchains - The number of Metropolis-coupled chains for each independent MCMC analysis

samplefreq - The frequency with which MrBayes writes output to files
printfreq - The frequency with which MrBayes prints output to the screen
filename - The root name for output file names
temp - The degree of heating for Metropolis-coupled chains

Many other MCMC options can be set with these commands. The full list of these options can be obtained by typing *help mcmcp* or *help mcmc*.

The other command useful in tweaking the details of the MCMC analysis is the *propset* command. *propset* can alter the relative frequency of proposals to different parameters, as well as the distributions from which proposals are drawn. Proposal distributions differ according to the parameter of interest. The final page of the MrBayes manual (http://mrbayes.sourceforge.net/manual.php) lists the proposal distributions for each parameter type and their associated tuning parameters.

Finally, a **'word of caution**'. Certain changes to proposals can make it virtually impossible for the MCMC analysis to mix properly across the posterior distribution and give biased results. Using rigorous checks of convergence should allow you to detect cases of very poor mixing.

Summarizing Output

After you have run an MCMC analysis in MrBayes and made sure that your runs have converged (a topic not explicitly covered in this tutorial – but see the software Tracer), you can summarize the estimated posterior distributions of both the parameter values and trees using the *sump* (parameters) and *sumt* (trees) commands. You can also exclude those samples biased by the starting point of an MCMC analysis, a process known as discarding the burn-in. Removing burn-in when summarizing the posterior distributions is done by specifying a value for the *burn-in* parameter with *sump* and *sumt*

```
MrBayes > sump burnin=1000
```

- Summarizing posterior samples of parameter values, discarding 1,000 *samples* (NOT generations)

```
MrBayes > sumt burnin=1000
```

- Summarizing posterior samples of trees, discarding 1,000 *samples*

Several other parameters can be specified when performing posterior summaries. A full list can be obtained by typing *help sump* or *help sumt*.

The output of the *sump* command will include means, medians, variances, and 95% credible intervals for all scalar parameters. It will also include an estimate of the marginal likelihood using the harmonic mean estimator.

The output of the *sumt* command includes estimated posterior probabilities of branches (.parts file), estimated posterior probabilities of trees (.trprobs file), and a majority-rule consensus tree calculated from the posterior tree samples (.con file).

Other software can help you summarize posterior distributions of trees and parameter values, as well as diagnose convergence. See Tracer, AWTY, TreeSetViz, and TreeScaper.

Estimating Marginal Likelihoods Using Stepping-stone Sampling

The marginal likelihood of a model is an important quantity that allows comparison of fit across different models, accounting for uncertainty in the values of model parameters. The simplest way to estimate the marginal likelihood of a model based on the output of MCMC sampling of the posterior is to calculate the harmonic mean of their likelihood scores. However, the harmonic mean can have some highly undesirable statistical properties, including bias and extremely high variance. This leads to low repeatability across replicates for estimates that are inaccurate. A much more accurate and repeatable approach was recently developed, also based on importance sampling, and is known as stepping-stone sampling (Xie et al., 2011; Fan et al., 2011). In this approach, an MCMC chain is still used, but it samples a series of "power posterior distributions" that move progressively between the posterior and the prior (or vice versa). A theoretical treatment of this approach is beyond the scope of this tutorial, but I encourage any user interested in estimating marginal likelihoods to read the papers describing and characterizing this approach (Xie et al., 2011; Fan et al., 2011).

To estimate marginal likelihoods via stepping-stone in MrBayes, one first needs to specify appropriate settings. This can be done using the *mcmcp*

and *ssp* commands. Relevant parameters include the total number of generations sampled (set with *mcmcp*), the burn-in before the first step of the sampling (set with *ssp*), the burn-in for each sampling step (set with *mcmcp*), and the number of different power posterior distributions sampled (set with *ssp*).

After everything is set up, start the stepping-stone sampling process using the simple *ss* command:

```
MrBayes > ss
```

Once sampling is completed across all power posterior distributions, MrBayes will provide an estimate of the marginal likelihood. It is always good practice to estimate these values several times with independent runs to ensure their repeatability.

The *sumss* command can also be used to check the quality of the stepping stone sampling. It provides step-specific views of likelihood profiles, allowing one to look for 'temperature lag'. Temperature lag occurs when not enough samples are removed at the beginning of a new step, causing the first few recorded samples to be influenced by the previous step. This can produce a biased estimate of the marginal likelihood.

NOTE: Because MrBayes is sampling many different distributions that vary between the posterior and prior when stepping-stone is run, do NOT use the trees and parameter files from a stepping-stone run to estimate posterior probabilities and credible intervals.

Analyzing Phylogenomic Datasets

There is no built-in way for MrBayes to explicitly handle the analysis of very large, genome-scale datasets. One option is to simply put all of your data in one Nexus file and provide character set definitions for each locus. By linking or unlinking various things across loci, you can vary the amount of independence given to each gene. The simplest concatenated analysis would apply the same sequence evolution model, priors, branch lengths, and topology to each gene. One could then allow separate models for each locus, separate overall rates of evolution, entirely separate branch lengths, or eventually entirely distinct topologies. The downside of this approach is that you are often constraining the computational resources available to analyze the overall dataset (although the parallel version of

MrBayes on a high-performance computing cluster may alleviate some of this).

Another approach is to separate the data from each locus into individual files. You can then set up separate command files tailored to each locus. If you have access to high-performance computing resources (even through publicly available web portals), you can run each of these jobs in parallel. This approach, obviously, does not share any information across loci. It is equivalent to the case where all aspects of the analysis are unlinked across loci.

BUCKy uses the output of independent analyses across different genes to try and estimate the degree of topological concordance across the genome.

Exercises

Using your newfound MrBayes prowess, see if you set up analyses to answer the following questions (or verify that things are working like they should). You may wish to work with a partner or two, and have each person run a different analysis involved in the comparisons below. Use the 100 loci provided from a recent study (Faircloth et al., 2012) attempting to infer the placement of turtles using the ultraconserved element (UCE) approach to sequencing. Try answering these questions with one locus (or a handful of loci) at first and see what you can scale up.

1. Let's start things off easy. Set up a Jukes-Cantor model (all substitution types equally probably and all base frequencies equal). Run the MCMC. Look at the header of your .p files to make sure your model was set up correctly. Check for topological convergence and (if you have Tracer), check for convergence of scalar values. Pick an appropriate burn-in and use *sump* to summarize your run. Record the log harmonic mean of the likelihood values (an estimate of the log marginal likelihood – NOT the same as the harmonic mean of the log-likelihood values) for each independent run (by default, 2).

2. Set up a GTR+G model (all substitution types free to vary, all base frequencies allowed to be different, and a gamma distribution describing the variation in rates of evolution across sites). Run the

MCMC. Do the same checks as with JC. Again, record the harmonic mean estimates of the marginal likelihood.

3. Let's pretend that this is a protein-coding gene. Set up a partitioned analysis by codon position. Do the same as for JC and GTR+G.

4. Manually calculate the Bayes factors between each pair of these three models using the harmonic mean estimates.

$$\log\left(\frac{marg.like.M_1}{marg.like.M_2}\right) = \log(marg.like.M_1) - \log(marg.like\ M_2)$$

Which model is preferred? How strongly?

5. Now, run stepping-stone sampling under each of three models and record marginal likelihood estimates. How do these values compare to the harmonic mean estimates? Calculate log(BF)s using the stepping-stone estimates. How do these compare to those estimated from the marginal likelihoods?

6. Now, set up a reversible jump MCMC (RJ-MCMC) that samples across different sub-models of the GTR class. Run this MCMC analysis. Use *sump* to summarize the results. Which models had the highest posterior probability?

7. If you have access to PAUP* or have Biopython installed and can use the script uploaded in the course copy folder, combine some of the genes into one dataset. If you can't do this on your own, or don't want to, look at the file "firstFive.nex". This contains a concatenation of all the data from the first five loci. Set up one analysis that unlinks models (let's say, GTR+G) across loci. Estimate the marginal likelihood of this model using stepping-stone. Now unlink rate multipliers across loci and repeat. Now unlink all branch lengths. Finally, unlink topology. Which of these models is preferred? What does this suggest about variation across loci?

8. Either by eye using the consensus topologies (perhaps in FigTree), or using TreeSetViz or TreeScaper to visualize the full posterior

distribution, compare the topologies inferred from each gene. How much do they vary? Are any branches strongly supported and conflicting?

9. Try increasing or decreasing the mean of the exponential branch-length prior and check to see how much your inferred tree length changes. Do your results (either tree length or topology) change?

10. Try performing a Bayes factor based test for the presence/absence of a branch placing turtles as sister to archosaurs (birds and crocodilians). Note that (as recently pointed out by Bergsten et al. in a forthcoming Syst. Biol. paper), to get meaningful results from tests of topology using Bayes factors, you have to provide the proper context. In other words, you need to constrain those parts of the tree for which you have strong evidence. Below are all the uncontroversial, 'backbone' constraints for the amniote taxa in the Crawford et al. data. After applying these, try running a stepping stone that positively and negatively constrains the position of turtles as sister to archosaurs (this is the relationship suggested by Crawford et al. based on analyses of the entire dataset). How much support does each gene provide for this relationship?

```
constraint birdsPos hard = gallus_gallus zebra_finch;
constraint crocsPos hard = alligator_mississippiensis crocodylus_porosus;
constraint archsPos partial = alligator_mississippiensis crocodylus_porosus gallus_gallus zebra_finch :
     homo_sapiens sphenodon_tuatara pantherophis_guttata anolis_carolinensis;
constraint lepsPos hard = sphenodon_tuatara pantherophis_guttata anolis_carolinensis;
constraint squamsPos hard = pantherophis_guttata anolis_carolinensis;
constraint turtsPos hard = chrysemys_picta pelomedusa_subrufa;
```