Manual

SubT 1.1

Martin Ryberg[*] and R. Henrik Nilsson
[*]kryberg@utk.edu

Introduction

SubT is a program for estimation of net diversification rates (speciation minus extinction rate) and relative extinction rate (extinction rate divided by speciation rate) based on the method of Bokma (2008). It applies Markov chain Monte Carlo (MCMC) methods in a Bayesian framework to produce an estimate of the posterior probability distribution for the parameters using node height (node age/distance from tip) data. SubT supports the inclusion of taxa for which the node height is unknown by treating them as parameters in the model (Bokma 2008). Such taxa are referred to as substitute taxa (Ryberg and Matheny 2011). SubT reads a text file with the data and specific commands and then executes the MCMC. The output is a tab-separated table of the Markov Chain that can be summarized by a consecutive command or loaded into Excel or R for analysis. The program is available from `http://web.utk.edu/~kryberg/`

Bayesian estimation of posterior distribution

Bayes' theorem states that the posterior probability distribution ($f(\theta|X)$) for the parameters (θ) and the data (X) is:

$$f(\theta \mid X) = \frac{f(X \mid \theta) f(\theta)}{f(X)}$$

where $f(X|\theta)$ is the likelihood, $f(\theta)$ is the prior probability distribution, and $f(X)$ is the marginal likelihood. The marginal likelihood is a normalizing constant that makes the posterior probability distribution integrate to 1. Analytic solutions are rare for the marginal likelihood except for very simple cases. However, the posterior distribution can be estimated using MCMC methods without calculating the marginal likelihood. A Markov chain is a system that transfers from one state to another and where the next state is only dependent on the present state, not any previous states. The Monte Carlo property translates into an essentially random walk between states. SubT uses the Metropolis-Hastings algorithm for the MCMC (Metropolis et al. 1953; Hastings 1970) and a block design for suggesting new parameters in each iteration of the chain.

The MCMC algorithm:

1. Set initial parameter values (θ: net diversification rate, relative extinction rate, and node heights of substitute taxa).
2. Suggest new net diversification rate and relative extinction rate according to a normal distribution centered on their respective present values to form a new parameter vector (θ^*: new net diversification rate, new relative extinction rate, and node heights of substitute taxa).

3. If a random value between 0 and 1 is less than $\frac{f(X \mid \theta^*)}{f(X \mid \theta)} \times \frac{f(\theta^*)}{f(\theta)}$ then accept the suggested state and set θ = θ^* (otherwise keep θ as is).
4. If there are substitute taxa: Suggest new node heights for each taxon where node heights are unknown according to a normal distribution centered on the present value to form a new parameter vector (θ^*: net diversification rate, relative extinction rate, and new node heights for taxa where node heights are unknown).
5. If there are substitute taxa: If a random value between 0 and 1 is less than $\frac{f(X \mid \theta^*)}{f(X \mid \theta)} \times \frac{f(\theta^*)}{f(\theta)}$ accept the suggested state and set θ = θ^* (otherwise keep θ as is).
6. At a regular number of iterations print θ.
7. Go to step 2.

After repeating steps 2-7 infinitely many times, the chain is guaranteed to converge on the posterior distribution. Of course, in reality steps 2-7 cannot be repeated infinitely many times, and the number of iterations needed to reach convergence depends on how efficient the chain is. Methods to judge if the chain has converged on the target (posterior) distribution are treated below. For a more complete introduction to MCMC in Bayesian analysis see, for example, Gamerman and Lopes (2006).

The likelihood
The likelihood of observing the node height data k given the net diversification rate (netdiv) and relative extinction rate (extinction proportion; extprop) is provided by Nee et al. (1994):

$$L(netdiv, extprop) = (s-1)!\,netdiv^{s-2} \exp\left((netdiv)\sum_{i=2}^{s-1} k_{i+1}\right) \times (1-extprop)^{s} \prod_{i=2}^{s} \left(\exp(netdiv)k_i - extprop\right)^{-2}$$

where s is the number of species in the clade.

The prior distributions
A uniform distribution between 0 and 1 is used as a prior for the relative extinction rate. For the net diversification rate, a gamma distribution is assumed. For substitute taxa, a uniform distribution between zero and the age of the oldest known node age for the clade is used as prior. In other words it is assumed that the node height data includes the oldest split of the clade (the crown age of the group).

Installing SubT
SubT is distributed as C++ source code in a zip archive. This means that the archive needs to be unzipped and that the source code must be compiled before the software can be used. To compile the source code you need a compiler. Most Unix-like systems come with the GNU compiler g++, but for MacOS X you need to install the developer tool kit (Xcode, found on the System DVD bundled with your computer) to get g++. There are several free compilers for Windows. For example, MinGW includes a compiler that

works much like g++. The following instructions are for Unix-type systems (Unix, Linux, MacOS X, and BSD); for instructions on how to compile on Windows, please consult the instructions for the specific compiler you use. The first step is to unzip the archive. Many file browsers can unzip files through a simple double click on the mouse, but if you want to do this in a bash environment (such as the Terminal of MacOS X), the easiest way is to be in the same directory (folder; i.e. have that directory/folder as your present working directory) as the SubT1.1.zip file. You can change your directory using the command `cd` followed by the directory you want to go. To see what directory you are in type `pwd`, and to see what files and folders are in the directory type `ls -ltr`. When in the same directory as the file SubT1.1.zip type:

```
unzip SubT1.1.zip
```

After unzipping the archive you may want to enter the directory where the source code file is by typing:

```
cd SubT1.1
```

When in the same directory as the source code, you compile the code by typing:

```
g++ -o SubT SubT1.1.cpp
```

You should now have an executable file named SubT. In the unlikely event that you get any error messages while compiling, it is likely that the source code is not compatible with your version of g++. You can execute SubT by typing:

```
./SubT -h
```

This should print the help text, listing the different commands for the program.

(Under rare circumstances, the binary file that is produced by the compiler is not executable by default:

```
[user SubT1.1] : ./SubT -h
-bash: ./SubT: Permission denied
```

This is changed through

```
chmod 755 SubT
```

)

Getting started

The commands and data for SubT are given in a single text file. It is important to keep in mind that it should be pure text format (.txt) and not any format that includes text formatting (e.g. Microsoft documents (.doc) or rich text format (.rtf)). SubT is case sensitive. That is, it matters if you type NTAXA, Ntaxa, or ntaxa. The first thing the program needs to know is the number of taxa for which you have node height data (i.e. *the number of node heights plus one*). So your file should contain the command `ntaxa` followed by the number of taxa, for example:

```
ntaxa 10
```

This will control how your data is read, so it is important that this number is correct. To make SubT start reading you data and execute the MCMC, give the command `start` followed by your data and then `end`:

```
start
10.88 10.72 10.50 7.36 4.86 2.02 1.38 0.26 0.000000001
end
```

Here, 10.88 correspond to the first split in the tree (i.e. *the crown age of the clade*), 10.72 to the second split, and so on. From version 1.1 SubT can also read trees in the Newick format. Then the command `start_tree` should be used:

```
start_tree
(taxa1:11.18,(taxa2:1.94,(taxa3:0,taxa4:0):1.94):9.25);
```

It is also possible to read trees in Newick format from a file by giving the command `start_file`:

```
start_file tree_file.tree
```

SubT does not check that the trees are ultrametric and will produce results even if they are not (using the distance to the leftmost tip from each node [as seen from the root towards the tips]). The interpretation of results from non-ultrametric trees is up to the user. If there is more than one set of node heights or more than one trees (e.g. from the posterior distribution of a Bayesian phylogenetic analysis), SubT will read the node height data for the first tree and execute an MCMC analysis, then read the node height data from the second tree and execute an MCMC analysis for that set of node heights/tree, and so forth. It is important that all sets of node heights/trees are based on the same number of taxa. The above is all that is needed to read the data and initiate the analysis. All other settings have default values, but you will probably want to set at least the number of generations for the MCMC and the sample frequency. These are given by `ngen` and `sampfreq` respectively, which – like all other parameters – should be given before `start`:

```
ngen 1000000
sampfreq 1000
```

The default model is the Yule (pure birth) model, but you can switch to the birth-death model by giving:

```
model d
```

Which would give a text file looking something like this:

```
ntaxa 10
ngen 1000000
sampfreq 1000
model d

start
10.88 10.72 10.50 7.36 4.86 2.02 1.38 0.26 0.000000001
end
```

If this file is saved as `infile.txt` it can be processed by typing the below while in the same directory as SubT and with the file `infile.txt` in that directory

```
./SubT infile.txt > outfile.txt
```

This will store the output in the file `outfile.txt`. Longer runs can take some time to complete. It is possible to output the results to file and still monitor the progress:

```
./SubT infile.txt > outfile.txt &
tail -f outfile.txt
```

This uses the Unix command `tail` in its file mode, which allows viewing of files as rows are appended to them. `Ctrl-C` stops `tail`, but it will not stop the underlying SubT run. Both the program and the infile can be called from any directory using their respective full paths. For a complete account of possible settings see *List of commands* below.

The output

SubT will produce text-formatted output. The program starts by printing the command it was called with, followed by the settings used for the run. When the settings have been printed, the program initiates the MCMC. The Markov chain is printed as a tab-separated table with the headings:

Tree_no specifies what set of node heights the MCMC output refers to;
Generation specifies what iteration the MCMC is at for that specific set of node heights;
Log_Lh gives the log likelihood;
Prior_P gives the log prior probability;
Post_P gives the log posterior probability (Log_Lh plus Prior_P);
Netdiv gives the net diversification rate;
Rel_ext_rate gives the relative extinction rate;
Acceptance_rate gives the number of proposals of changes in the net diversification rate and relative extinction rate that have been accepted for the MCMC on that specific set of node heights;

Acceptance_rate_nodes gives the number of proposals of changes in the node heights for the substitute taxa that has been accepted for the MCMC on that specific set of node heights.

The first row printed for each MCMC on separate node heights represents the starting values of that chain (Generation 0). SubT will summarize the output if given the switch `-s` followed by the file name, and the burn-in (see below) can be given by the `-b` switch, for example:

```
./SubT -s outfile.txt -b 1000
```

The summary statistics will be averaged over all trees for which there are MCMCs in the file. The given statistics include mean, median, and 95% credibility intervals for net diversification rate and relative extinction rate. If preferred, the output can be read into spreadsheet programs where the appropriate summary statistics can be calculated. If the rows before the heading of the Markov chain are deleted (in a text editor), the output can be read into R for analysis.

Convergence of the chain and effective sampling size

The first iterations of the Markov chain will be dependent on the starting point and it will take some time for the chain to start sampling from the posterior distribution as unbiased by the starting point. The first iterations must therefore be excluded as a burn-in period. There are several techniques to try to determine if the chain has converged on the posterior distribution. Many of these are implemented in the R package CODA (Plummer et al. 2006). One of the more basic ways is to plot the (log)likelihood scores for each sampling point of the chain and see when they start to fluctuate around some mean instead of reflecting a general increasing trend (Figure 1)

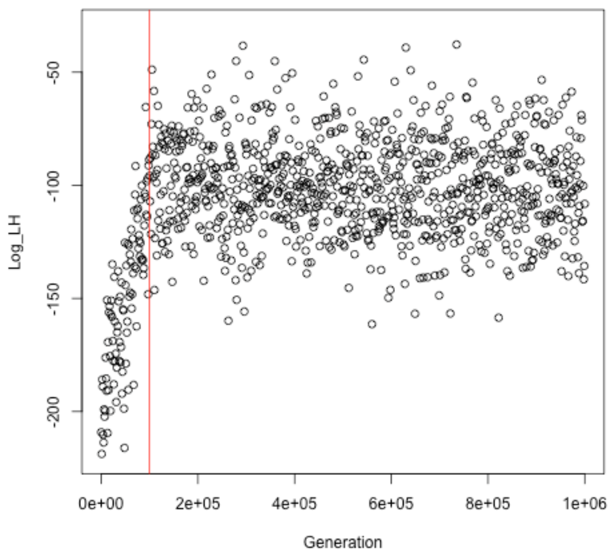

Figure 1. Log likelihood values for an MCMC run. The red line marks the point where the chain levels off and, presumably, has reached the posterior distribution.

It should be noted that not all cases are as clear as in Figure 1 and that the likelihood scores can start to increase again after a period of deceptive stability around the same mean. After the burn-in has been determined and removed, you need to make sure that you have enough samples left for a sufficient representation of the posterior distribution. The samples of the Markov chain are not independent since each iteration depends on the previous one. Your samples do not therefore fully qualify as independent samples. It is possible to calculate how many independent samples your chain represents (effective sample size), and CODA has a function to do so. How many samples fewer than you have in your chain this will be depends on how well your chain was mixing (something that can be manipulated by the `windowidth` and `nodewindow` settings, see below) and how often you sampled the chain. Too high sample frequency will give large output files that may require a lot of memory/disk space and may be slow to work with, but may not increase the effective number of samples that much. How many samples you need depends on what precision you need in your estimation of the posterior distribution. For small simple datasets sampling every 1,000 generations of a one-million-generation run will often be enough, but for larger and more complex datasets more than ten million generations – with every 10,000 generations sampled - may be needed. Depending on the number of substitute taxa and the distribution of your samples the parameter space will be more or less complex. In complex cases the chain may get stuck in local optima, and it is therefore advisable to run and compare several MCMCs. However if you have no - or only a few - substitute taxa, the parameter space will be relatively easy and it is not likely that the chain will get stuck at a local optimum.

Comparing models using Bayes factors

In Bayesian statistics different hypotheses (eg. H_1 and H_2) can be compared by posterior odds (Kass and Raftery 1995). If the prior distributions $f(H_2)=1-f(H_1)$ then Bayes' theorem (see above) gives us

$$f(H_{1 and 2} \mid X) = \frac{f(X \mid H_{1 and 2}) f(H_{1 and 2})}{f(X \mid H_1) f(H_1) + f(X \mid H_2) f(H_2)},$$

such that

$$\frac{f(H_1 \mid X)}{f(H_2 \mid X)} = \frac{f(X \mid H_1)}{f(X \mid H_2)} \frac{f(H_1)}{f(H_2)},$$

where $\frac{f(X \mid H_1)}{f(X \mid H_2)}$ = Bayes factors, such that the posterior odds = Bayes factors × prior odds. If the prior probability for each hypothesis is the same, the posterior odds depend only on the Bayes factors. Bayes factors can be estimated from the harmonic mean of the log likelihood from the MCMC. A difference of more than two in two times the difference in harmonic mean of the log likelihood scores (2 * Δ harmonic mean of Log Lh) is taken as *positive* support, more than six as *strong* support, and more than ten as *very strong* support (for the model with highest mean log likelihood score). The harmonic mean estimate of BFs is not fully robust so it is recommended that several separate MCMCs are performed for each model to validate the results.

List of commands
SubT offers a number of options to add substitute taxa, to change model, and to tweak the chain to be more efficient. The default values are set in a way that assumes the ages are given in million years (Ma) and that net diversification rates are in the order of 0.1 speciations/species/Ma. If your data deviates from this you should consider setting your own prior and window width. Appropriate starting values for net diversification rates and substitute taxa node heights are calculated under the Yule model.

end will stop reading the data and no more MCMC analyses will be performed after this command. Example:

```
end
```

meanGamma gives the mean value for the net diversification rate prior. Appropriate values may be inferred from closely related groups with similar age and number of species. Example:

```
meanGamma 0.2
```

model defines which of the two models will be used: Yule (relative extinction rate = 0), or Birth-Death (default). To apply the Yule model give:

```
model y
```

To apply the Birth-Death model give:

```
model d
```

ngen gives the number of generations of the MCMC. Example:

```
ngen 100000000
```

nodelog gives the file name to store the estimated node heights *for the substitute taxa*. If no file name is given, the node heights will not be stored. The node height for each generation that is sampled will be stored as a space-separated vector. The generations will be separated by linebreaks.

```
nodelog node.log
```

nodewindow controls how different the proposed state for the node heights of the substitute taxa is from the present state in each generation of the Markov chain. It does this by multiplying the proposed change with the given value divided by the total number of species (`ntaxa` plus `nsubstitute`). The unmodified change is drawn from a standard normal distribution multiplied by the age of the oldest node height. Example:

```
nodewindow 0.5
```

nsubstitute gives the number of substitute taxa, i.e. taxa in the clade for which there are no data on their node height.

```
nsubstitute 5
```

ntaxa gives the number of species the node height/tree data is based on. Example:

```
ntaxa 10
```

sampfreq gives the frequency with which to sample the MCMC. A dense sampling will give large output files and potentially highly autocorrelated sample points, whereas a diluted sampling may give too few sampling points and an overly coarse estimation of the posterior probability. The optimal number will depend on how efficient the chain is. Example:

```
sampfreq 10000
```

shapeGamma gives the shape of the gamma distribution for the net diversification rate prior. It only accepts integer values (Erlang distribution). If the shape is set to 1 it is in fact an exponential distribution. Example:

```
shapeGamma 1
```

start will tell SubT to start reading the data and execute the MCMC. You may have several sets of data as long as they have `ntaxa` -1 node heights each. Example:

```
start
10.88 10.72 10.50 7.36 4.86 2.02 1.38 0.26 0.000000001
11.14 10.45 5.30 4.28 4.19 3.27 1.97 9.35 0.000000001
17.97 15.45 9.29 5.29 5.17 3.18 1.54 1.50 0.000000004
29.89 11.30 10.07 7.96 2.39 2.14 1.79 0.86 0.000000002
20.87 18.07 11.82 10.58 1.98 1.48 1.23 0.65 0.0000001
```

start_tree will tell SubT to start reading the data in form of trees and execute the MCMC. You may give several trees as long as they all have `ntaxa` taxa. Example:

```
start_tree
(((taxa3:0,taxa4:0):1.43,taxa2:1.43):9.37,taxa1:10.80);
((taxa2:5.21,(taxa3:0,taxa4:0):5.21):1.18,taxa1:6.39);
(taxa1:3.14,(taxa2:1.86,(taxa3:0,taxa4:0):1.86):1.28);
```

start_file will tell SubT to start reading trees form a tree file and execute the MCMC for each tree. The file may contain several trees as long as they all have `ntaxa` taxa. Example:

```
start_file tree_file.tree
```

windowidth (or windowwidth) controls how different the proposed state of net diversification rate and relative extinction rate is from the present state in each generation of the Markov chain by multiplying the proposed change with the given value. The unmodified change is drawn from a standard normal distribution. Example:

```
windowidth 0.1
```

Miscellaneous

If you use a spreadsheet program to summarize the MCMC, note that both Excel and OpenOffice Calc sometimes insist on trying to interpret numbers as dates or otherwise reformat them in unexpected ways; solutions are found at http://www.webhostingtalk.com/showthread.php?t=601434 (Excel) and http://user.services.openoffice.org/en/forum/viewtopic.php?f=9&t=32979&start=0 (OpenOffice Calc).

SubT will ignore lines given within square brackets `[such as this text]` to give the user the opportunity to enter comments in the infile. The square brackets must be balanced for the program to run.

SubT does not read trees from NEXUS files, but if you use square brackets to exclude everything but the trees (especially characters with special meaning in trees like "(", ")", ";", and ":") it will usually work.

References

Bokma F. 2008. Bayesian estimation of speciation and extinction probabilities from (in)compleate phylogenies. Evolution 62:2441-2445.

Gamerman D, Lopes HF. 2006. Markov chain Monte Carlo – Stochastic simulations for Bayesian Inference. Chapman & Hall/CRC, Boca Raton.

Hastings, W. 1970. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57:97–109.

Kass RE, Raftery AE. 1995. Bayes factors. Journal of the American Statistical Association 90:773-795.

Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E. 1953. Equations of state calculations by fast computing machines. The Journal of Chemical Physics 21:1087–1092.

Plummer M, Best N, Cowls K, Vines K. 2006. CODA: Convergence Diagnosis and Output Analysis for MCMC. R news 7-11.

Ryberg M, Matheny PB. Dealing with incomplete taxon sampling and diversification of a large clade of mushroom-forming fungi. Evolution. In press. doi: 10.1111/j.1558-5646.2011.01251.x.