MCMCTree tutorials

Mario dos Reis and Ziheng Yang

October 7, 2013

MCMCTree performs Bayesian estimation of species divergence times using soft fossil constraints[8] under various molecular clock models[4, 5, 8]. The general theory of molecular dating is given in chapter 7 of Yang[9] and in dos Reis and Yang[3]. The program uses for input a sequence alignment (nucleotide or protein), a phylogenetic tree with fossil calibrations, and a control file (usually called mcmctree.ctl) that contains the instructions for the program. MCMCTree is part of the PAML package[7].

It is assumed that you have some basic knowledge of using the command line in Windows or Unix systems (e.g. Linux and MacOS). You need to download and install the PAML package from

http://abacus.gene.ucl.ac.uk/software/paml.html.

Make sure you have the latest version, currently 4.7 (as of March 2013). There are executables for Windows (*.exe) but Unix users may need to compile the programs. Please follow the instructions in the PAML website to modify your operating system PATH variable. This is necessary so that you can call the programs from the command line without having to type their full folder (path) location.

Tutorial 1: Divergence time of apes

In this tutorial we will analyze a data set of mitochondrial protein-coding genes for 7 ape species. This is the same data set analyzed by Yang and Rannala[8], and is included in the PAML[7] release (examples/DatingSoftBound). File “mtCDNApri123.txt” contains the nucleotide alignment. The alignment is divided into 3 partitions corresponding to the 1st, 2nd and 3rd codon sites. File “mtCDNApri.trees” contains the phylogenetic tree relating the 7 species with the fossil calibrations. The tree file looks like this

```
7 1
((((human, (chimpanzee, bonobo)) ’>.06<.08’, gorilla),
(orangutan, sumatran)) ’>.12<.16’, gibbon);
```

The first line has the number of species (7) and the number of trees (1). Then the tree in Newick format is given. The tree must not have branch lengths. The tree has two fossil calibrations, one for the most recent common ancestor of human-chimp: ’> .06 < .08’ and another for the most recent common ancestor of the greater apes: ’.12 < .16’. The time unit is 100 Million years (Myr). So the first calibration restricts the common ancestor of human-chimp to be between 6-8 Myr ago. Calibration bounds in MCMCTree are soft, and there is a small probability (0.025 by default) that the bounds may be violated. Note that the tree does not have a fossil calibration at the root. MCMCTree always needs a calibration on the root of the tree, and when this calibration is not present in the tree file, it must be specified in the control file (variable RootAge).

The control file “mcmctree.ctl” contains all the necessary instructions to run the MCMCTree program. You can open this file with a text editor (e.g. Notepad or TextEdit). The file should look like this

```
        seed = -1
     seqfile = mtCDNApri123.txt
    treefile = mtCDNApri.trees
     outfile = out
       ndata = 3
     usedata = 1    * 0: no data; 1:seq; 2:approximation; 3:out.BV (in.BV)
       clock = 2    * 1: global clock; 2: independent; and 3: correlated rates
     RootAge = '<1.0'  * safe constraint on root age, used if no fossil for root.
       model = 0    * 0:JC69, 1:K80, 2:F81, 3:F84, 4:HKY85
       alpha = 0    * alpha for gamma rates at sites
       ncatG = 5    * No. categories in discrete gamma
   cleandata = 0    * remove sites with ambiguity data (1:yes, 0:no)?
     BDparas = 1 1 0   * birth, death, sampling
 kappa_gamma = 6 2     * gamma prior for kappa
 alpha_gamma = 1 1     * gamma prior for alpha
 rgene_gamma = 2 2     * gamma prior for rate for genes
sigma2_gamma = 1 10    * gamma prior for sigma^2     (for clock=2 or 3)
    finetune = 1: .1 .1 .1 .1 .1 .1 * auto (0 or 1) : times, rates, etc.
       print = 1
      burnin = 2000
    sampfreq = 2
     nsample = 20000
```

seed: this sets the random seed used by the program. When set to -1 the program will use the computer’s current time to set the seed, so every time you run MCMCTree it will start with a different seed and the results of the MCMC will look different. If you need reproducible results, you can set the seed to an odd number or an even number.

seqfile and *treefile*: these are the names of the sequence alignment file and the tree file respectively.

outfile: once the program completes, it will write a summary of the results to this file.

ndata: the number of partitions in the alignment file. In our example, we have three partitions, corresponding to each one of the three codon positions in the mitochondrial proteins.

usedata: when set to 1, the likelihood function is calculated in the normal way, and the MCMC analysis proceeds as usual. When usedata=0, the likelihood is not calculated (it is set to 1), so only the prior is computed. When usedata=2 and =3, approximate likelihood calculation and ML estimation of branch lengths is performed. We will explain these in the next tutorial.

clock: the clock model to use. Here we will work with the independent rates model (clock=2), where the rates follow a log-normal distribution (that is, the logarithm of the rate is normally distributed).

RootAge: a calibration to use for the root if this is not provided in the tree file. Here we use '$<$ 1.0' or a maximum constraint of 100 Myr for the age of the most recent common ancestor of all apes.

model, *alpha* and *ncatG*: the substitution model to be used. In this example we will use JC69 (it is very quick to compute). Because alpha=0, We do not use a gamma model of rate variation in this example. The example mcmctree.ctl file in the PAML package may have model=4 and alpha=0.5 (HKY+G5), if that is the case, change to the JC69 model instead.

BDparas: parameters controlling the birth-death process. The birth-death process is used to construct the time prior for the nodes in the tree that do not have a fossil calibration. Here we used the default 1 1 0, which generates uniform node age priors.

kappa_gamma and alpha_gamma: gamma priors for the substitution model parameters κ (transition/transversion rate ratio) and α (gamma shape parameter for variable rates among sites).

rgene_gamma: gamma prior for the mean substitution rate. The gamma distribution has mean α/β and variance α/β^2. The first parameter (α) controls the shape of the distribution. Values of $\alpha = 1$ or $= 2$ lead to fairly diffuse priors. It is advisable to set α to one of those two values, and then fix β so that then mean rate is reasonable. In this example we set $\alpha = 2$ and $\beta = 2$ for a mean rate of 1 substitution per 100 Myr. Users of the R program for statistics (www.r-project.org) can easily plot the gamma distribution with

```
> curve(dgamma(x, shape=2, rate=2), from=0, to=10)
```

sigma2_gamma: gamma prior for the rate drift parameter (i.e, the variance of the logarithm of the rate, σ^2). Larger values of σ^2 imply more rate heterogeneity. The prior on σ^2 can have a strong impact on posterior time estimates[4], particularly for short alignments and few loci.

finetune: step sizes for proposals during the MCMC. From version 4.4e, autofinetune has been implemented so setting the step sizes is not as critical as before.

print: if set to 1, the output of the MCMC and a summary of the results will be written to the hard disk (the MCMC is written to the mcmc.out file and the summary to the outfile as set above). You want this. If set to 0, results will be printed to the screen only. You don't want that.

burnin, *sampfreq* and *nsample*: in our example, the program will discard the first 2,000 iterations as burn-in, and then it will sample every 2 iterations until it has gathered 20,000 samples. In total, the MCMC will run for $2,000 + 2 \times 20,000 = 42,000$ iterations. Normally, you should gather between 10,000 to 20,000 samples for a good statistical summary. Large sample sizes (say 100,000) tend to waste a lot of hard drive space providing very little statistical improvement. It may also take a long time for the program to summarize the results. If you need to increase the length of the MCMC (to improve convergence), increase sampfreq but keep nsample at a reasonable value.

We are now ready to run the program and look at the results. Open a terminal window (on Windows go to Start > All programs > Accessories > Command prompt) and go to the directory where the tutorial files have been saved. Create a new directory called run01, and copy the tree, alignment and control files into this directory. On my Windows computer, the tutorial files were copied into C:\Users\Mario\Tutorial\run01>. Go into this new directory, and on the command line (terminal window) type

```
C:\Users\Mario\Tutorial\run01> mcmctree mcmctree.ctl
```

The MCMC program will start. It will read the alignment, tree and control files and it will first perform some safety checks. When the MCMC itself starts running, the output on the terminal should look like

```
lnL0 = -40215.47
Starting MCMC (np = 48) . . .
finetune steps (time rate mixing para RatePara ...):  0.1000  0.1000  0.1000  0.1000  0.1000
  paras: 6 times, 3 mu, 3 sigma2 (& rates, kappa, alpha)
 -4% 0.16 0.64 0.33 0.00 0.69  0.178 0.149 0.091 0.067 0.032 - 1.438 -34999.3
Current Pjump:      0.16033  0.64117  0.32800  0.00000  0.69267
Current finetune:   0.10000  0.10000  0.10000  0.10000  0.10000
New     finetune:   0.05050  0.31051  0.11113  0.00100  0.37446
 -2% 0.38 0.19 0.40 0.00 0.37  0.160 0.147 0.092 0.066 0.029 - 1.313 -34989.9
Current Pjump:      0.38033  0.18628  0.39600  0.00000  0.36633
Current finetune:   0.05050  0.31051  0.11113  0.00100  0.37446
New     finetune:   0.06743  0.18359  0.15638  0.00001  0.47671
 -1% 0.27 0.37 0.27 0.00 0.30  0.154 0.147 0.092 0.066 0.029 - 1.280 -34995.0
Current Pjump:      0.27133  0.36956  0.27400  0.00000  0.30433
Current finetune:   0.06743  0.18359  0.15638  0.00001  0.47671
New     finetune:   0.06009  0.23632  0.14090  0.00000  0.48476
  0% 0.29 0.27 0.32 0.00 0.30  0.155 0.145 0.094 0.067 0.029 - 1.332 -34996.7  0:02
Current Pjump:      0.29367  0.26978  0.32200  0.00000  0.29833
Current finetune:   0.06009  0.23632  0.14090  0.00000  0.48476
New     finetune:   0.05862  0.20922  0.15316  0.00000  0.48163
  5% 0.32 0.31 0.25 0.00 0.31  0.158 0.147 0.092 0.066 0.029 - 1.280 -34997.5  0:04
 10% 0.31 0.32 0.27 0.00 0.30  0.158 0.146 0.093 0.067 0.029 - 1.303 -35000.8  0:06
 15% 0.32 0.32 0.28 0.00 0.30  0.157 0.146 0.093 0.067 0.029 - 1.308 -34994.6  0:08
 20% 0.32 0.32 0.28 0.00 0.31  0.157 0.146 0.093 0.067 0.029 - 1.311 -34993.5  0:10
```

The initial likelihood is lnL0 $= -40,215.47$. Our rooted tree of 7 species has $7-1=6$ internal nodes and $7\times 2-2=12$ branches. Therefore we are estimating 6 divergence times; 3 mean mutation rates and 3 rate drift parameters, one for each one of our 3 partitions (codon sites); and $12\times 3=36$ branch rates. In total we are estimating 48 parameters.

Now let's look at the first line of the MCMC proper:

```
-4% 0.16 0.64 0.33 0.00 0.69  0.178 0.149 0.091 0.067 0.032 - 1.438 -34999.3
```

The negative percentage (-4%) indicates that we are in the burn-in stage of the MCMC. The next 5 numbers are the acceptance proportions. They are printed in the order times, rates, mixing, substitution model parameters, and rate parameters. For example, 16% of all proposed times were accepted during this stage of the MCMC (i.e. 84% were rejected), while 64% of the rates proposed were accepted. A good MCMC analysis should have acceptance proportions close to 30% (20-40% being a good range and 15-70% being acceptable). You can see that the program goes through various rounds of finetune improvement until the acceptance proportions get very close to 30%:

```
0% 0.29 0.27 0.32 0.00 0.30  0.155 0.145 0.094 0.067 0.029 - 1.332 -34996.7  0:02
```

The JC69 model has no parameters and so the acceptance proportion is 0%. This is fine, no parameters are being proposed therefore none are being accepted! The next five numbers are the mean divergence times for five nodes. The first number (0.155) is the age of the root. At this stage, the MCMC is estimating the average time for the ancestor of apes to be 15.5 Myr ago. After the dash we see one branch rate, the likelihood ($-34,996.7$) and the time it has taken the MCMC to run up to that point: 2 seconds (0:02). The rest of the output looks like

```
 25% 0.32 0.32 0.29 0.00 0.31  0.157 0.146 0.093 0.067 0.029 - 1.313 -34998.0  0:11
 30% 0.31 0.32 0.30 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.318 -34992.6  0:13
 35% 0.32 0.32 0.30 0.00 0.30  0.156 0.146 0.093 0.067 0.029 - 1.316 -34989.3  0:15
 40% 0.32 0.32 0.29 0.00 0.30  0.156 0.146 0.093 0.067 0.029 - 1.318 -35004.4  0:17
 45% 0.32 0.33 0.29 0.00 0.30  0.156 0.146 0.093 0.067 0.029 - 1.315 -34993.0  0:19
 50% 0.32 0.33 0.29 0.00 0.30  0.156 0.146 0.093 0.067 0.029 - 1.315 -34996.8  0:21
 55% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.316 -34998.6  0:23
 60% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.317 -34992.6  0:25
 65% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.318 -34992.3  0:27
 70% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.318 -34997.5  0:28
 75% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.316 -34994.6  0:30
 80% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.317 -34994.3  0:32
 85% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.318 -34998.6  0:34
 90% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.317 -34997.2  0:36
 95% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.317 -34993.1  0:38
100% 0.32 0.33 0.29 0.00 0.30  0.156 0.145 0.093 0.067 0.029 - 1.318 -34998.9  0:40
```

It is important to look at the values in each column (the acceptance proportions, times and rates). They should be stable throughout the MCMC run. If the acceptance proportions are changing too much (specially at the beginning), it means that the burn-in was not long enough, so you should increase the burnin variable in the control file and run the analysis again. If the age of the root wanders too much, specially if it seems to be wandering in a particular direction (getting older or younger as the MCMC progresses), increase the burning and sampfreq in the control file. Similarly examine the behavior of the other times, the rates and the likelihood and modify the length of the MCMC if necessary.

Achieving convergence in an MCMC analysis is a tricky business. Even if the acceptance proportions, the times and rates look stable, convergence of the MCMC is not guaranteed. The only way to check whether convergence has been achieved is by repeating the analysis. Create a new directory and call it run02, and copy the necessary files into this new directory. Run the analysis again and then compare the output of the two analyses. They should be similar (but not identical).

Once the MCMC has finished (it reached 100%) the program will summarize the results and will print the summary to the screen. The program will also generate several output files: out, SeedUsed, mcmc.out and FigTree.tre. The out file contains a summary of the results. Open this file with your favorite text editor (Notepad, TextEdit, etc.). There is a lot of rubbish printed at the beginning of the file which you can usually ignore. Scroll down the file until you see six phylogenetic trees printed to the screen:

```
Species tree for TreeView.  Branch lengths = posterior mean times; 95% CIs = label
((((1_human, (2_chimpanzee, 3_bonobo) 12 ) 11 , 4_gorilla) 10 , (5_orangutan, 6_su ...
((((human: 0.067071, (chimpanzee: 0.028581, bonobo: 0.028581): 0.038490): 0.025788 ...
((((human: 0.067071, (chimpanzee: 0.028581, bonobo: 0.028581) 0.023-0.035: 0.03849 ...
rategram locus 1:
((((human: 0.389222, (chimpanzee: 0.427481, bonobo: 0.394052): 0.391420): 0.442241 ...
rategram locus 2:
((((human: 0.158087, (chimpanzee: 0.132350, bonobo: 0.145101): 0.129604): 0.164731 ...
rategram locus 3:
((((human: 1.969288, (chimpanzee: 1.926311, bonobo: 1.653482): 2.067122): 1.553286 ...
```

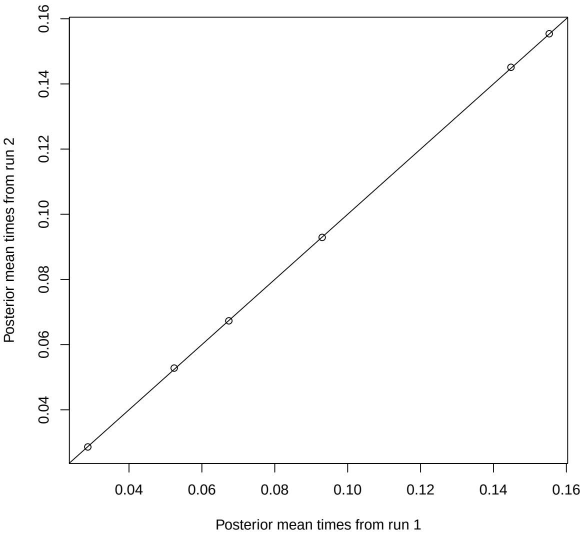
The first tree simply contains node labels. The second tree contains branch lengths in time units. The third tree contains branch lengths in time units, plus credibility intervals for the ages of the nodes. The last three trees have substitution rates instead of branch lengths, each tree representing each locus, in our example, each one of the three codon positions. After the trees, the means and 95% credibility intervals for the 48 estimated parameters are printed to the screen. For example

```
t_n8 0.1560 (0.1315, 0.1805) (Jeffnode 12)
```

is the age (time) for node 8 (the root of the tree). The jeffnode is the node number used by the program Multidivtime, written by Jeff Thorne[6].

To check for convergence you should cut the values for the times from the first out file (from the run01 directory) and paste them into an Excel spreadsheet (or if you're a pro, you can do a unix 'grep t_n' on the out file and load grep's output into R). You should then do the same with the times in the second out file (from the run02 directory). You can now use Excel (or R) to plot the set of times for the first run against the set of times for the second run. The points should fall very closely onto a straight line (the $x = y$ line). If not, convergence was not achieved and you may need to run the MCMC chain for longer (increasing either nsamp, burnin or both). This is one of the most important steps in any MCMC analysis. You must always do this. Below is an example figure created with R that shows how nicely convergence was achieved in my computer.

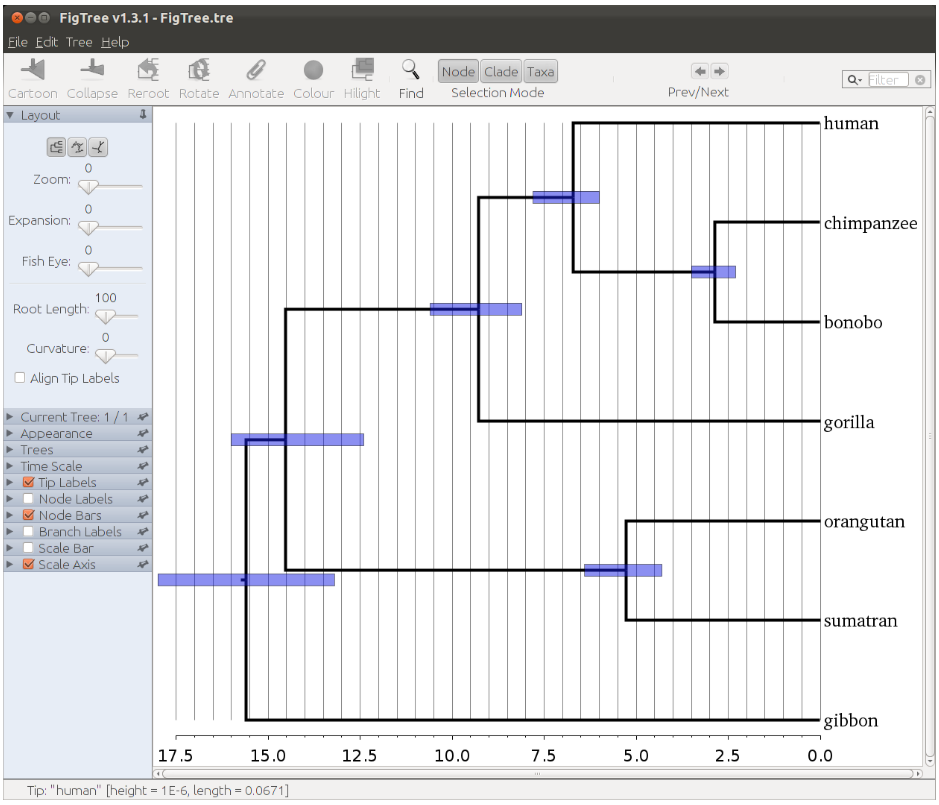
```
> # This is R code
> # Posterior mean times previously uploaded into R
> plot(t1$t, t2$t, xlab="Posterior mean times from run 1",
+ ylab="Posterior mean times from run 2")
> abline(0, 1) # this adds the x = y line.
```



File mcmc.out contains the raw output of the MCMC. In our example, this file has 50 columns and 20,002 lines. The first column is the generation (iteration) number of the sample, the next 48 columns correspond to each of the 48 parameters analyzed, and the last column has the likelihood. The number of lines corresponds to the number of samples taking during the MCMC. The mcmc.out file is suitable for analysis with the Tracer program (http://beast.bio.ed.ac.uk/Tracer).

File SeedUsed contains the random seed used to initialize the MCMC. If you copy the value in the file (in my case it is 949119895) into the seed variable of the control file, you can run the analysis again and get identical results.

File FigTree.tre has a version of the posterior tree in nexus format, suitable for the Fig Tree program of Andrew Rambaut (http://tree.bio.ed.ac.uk/software/figtree/). Note that this is a text file, and at the end of it there are some notes about options you may use for Figtree. This is a screen shot of our ape phylogeny as plotted by Figtree



Zoom: 0

Expansion: 0

Fish Eye: 0

Root Length: 100

Curvature: 0

☐ Align Tip Labels

☑ Tip Labels

☐ Node Labels

☑ Node Bars

☐ Branch Labels

☐ Scale Bar

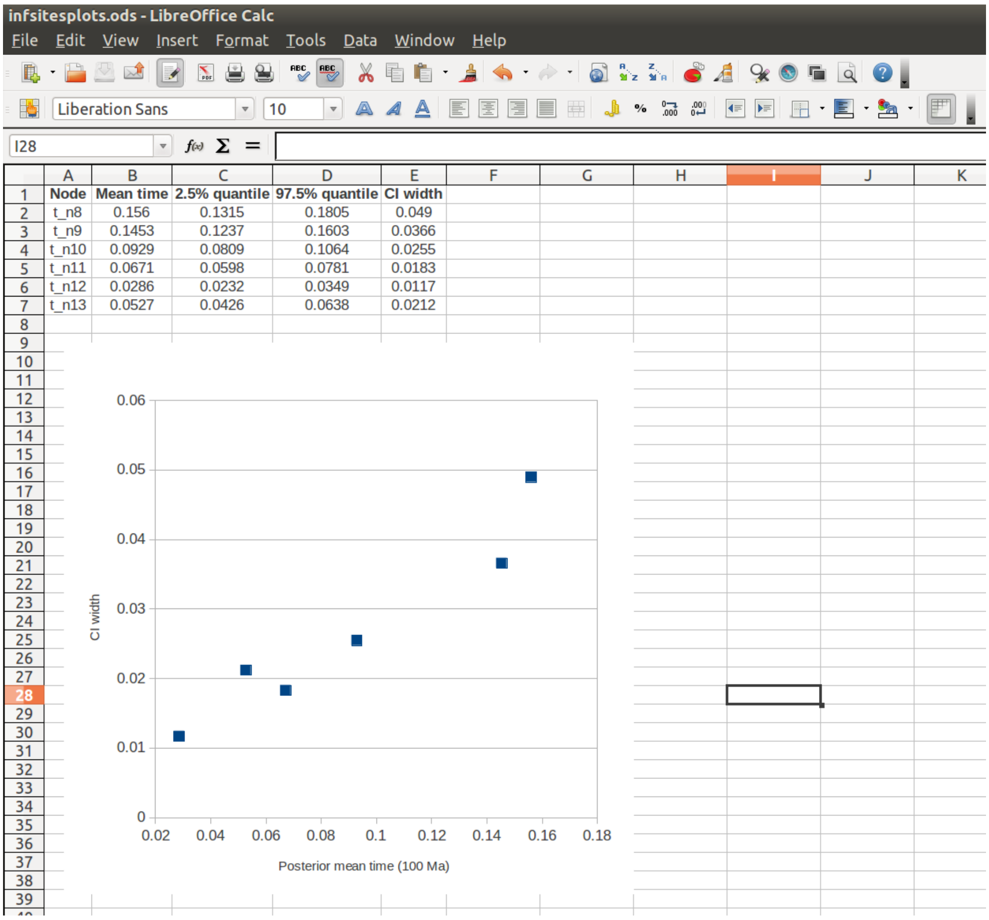
☑ Scale Axis

Tip: "human" [height = 1E-6, length = 0.0671]

As the number of sites and loci tend to infinity, a plot of mean posterior times vs. credibility interval widths will tend to a straight line. It is useful to generate this plot to assess whether collecting additional molecular data would improve the analysis. Open the out file with your favorite text editor and look for the divergence times:

```
t_n8       0.1560 (0.1315, 0.1805)  (Jeffnode 12)
t_n9       0.1453 (0.1237, 0.1603)  (Jeffnode 11)
t_n10      0.0929 (0.0809, 0.1064)  (Jeffnode 10)
t_n11      0.0671 (0.0598, 0.0781)  (Jeffnode  9)
t_n12      0.0286 (0.0232, 0.0349)  (Jeffnode  8)
t_n13      0.0527 (0.0426, 0.0638)  (Jeffnode  7)
```

For example, for the root (node 8), the credibility interval width is 0.1805 – 0.1315 = 0.049. You should be able to copy those lines and paste them into a spreadsheet (Microsoft Excel or LibreOffice Calc, some manual editing will be necessary). Use your spreadsheet to calculate the CI widths for all nodes, and then you can plot them against the mean posterior times. Using LibreOffice Calc, I get:


infsitesplots.ods - LibreOffice Calc
File Edit View Insert Format Tools Data Window Help
Liberation Sans
10
I28
0.06
0.05
0.04
0.03
0.02
0.01
0
CI width
0.02 0.04 0.06 0.08 0.1 0.12 0.14 0.16 0.18
Posterior mean time (100 Ma)

Node	Mean time	2.5% quantile	97.5% quantile	CI width
t_n8	0.156	0.1315	0.1805	0.049
t_n9	0.1453	0.1237	0.1603	0.0366
t_n10	0.0929	0.0809	0.1064	0.0255
t_n11	0.0671	0.0598	0.0781	0.0183
t_n12	0.0286	0.0232	0.0349	0.0117
t_n13	0.0527	0.0426	0.0638	0.0212

In this case I plotted column E (CI width) vs. column B (Mean time). As an exercise, try adding a trend line (passing through the origin) so that you can better gauge the linearity of the data. This is known as an infinite sites plot[5, 8].

Now open the mcmctree.ctl file, and change the usedata variable.

```
      seed = -1
   seqfile = mtCDNApri123.txt
  treefile = mtCDNApri.trees
   outfile = out
     ndata = 3
   usedata = 0    * 0: no data; 1:seq; 2:approximation; 3:out.BV (in.BV)
```

Now repeat the analysis. The program will run the MCMC without using the molecular data, that is, the prior distribution of times will be generated. Open the out file and examine the prior times, compare them to the posterior times obtained above (the values from your run may look slightly different):

```
t_n8       0.5849 (0.1604, 1.0043)  (Jeffnode 12)
t_n9       0.1400 (0.1201, 0.1599)  (Jeffnode 11)
t_n10      0.1052 (0.0684, 0.1481)  (Jeffnode 10)
t_n11      0.0702 (0.0601, 0.0800)  (Jeffnode  9)
t_n12      0.0353 (0.0018, 0.0721)  (Jeffnode  8)
t_n13      0.0696 (0.0033, 0.1418)  (Jeffnode  7)
```

Note that the program uses the fossil calibrations to generate the time prior, and that sometimes the time prior may look very different to the original fossil calibrations. You should always run the program with usedata=0 to check that the time prior is sensible. Using these results, prepare an infinite sites plot. How does it compare to the results with usedata=1?

Now run the program again using model=4 and alpha=0.5 in the mcmctree.ctl file. This is the HKY+G model. How do the times compare with those estimated under the JC69 model? Calculate the time prior again, is it different from the prior under the JC69 model?

Tutorial 2: Divergence time of apes with approximate likelihood calculation

For large alignments, calculation of the likelihood function during the MCMC is computationally expensive, and estimation of divergence times is very slow. Thorne et al.[6] suggested using an approximate method to calculate the likelihood that improves the speed of the MCMC dramatically. The approximate method is explained in detail by dos Reis and Yang[2]. As of MCMCTree v4.5, a modification of Thorne's method has been introduced, and the arcsine-based approximation is now the default[2]. An example of a very large alignment (over 20 million sites) analysed with the approximate method is given in dos Reis et al.[1].

Estimation of divergence times using the approximate method follows two steps. In the first step the branch lengths are estimated by maximum likelihood, together with the gradient and Hessian (i.e. vector of first derivatives and matrix of second derivatives) of the likelihood function at the maximum likelihood estimates. The gradient and Hessian contain information about the curvature of the likelihood surface. In the second step, estimation of divergence times proceeds using MCMC, but using the gradient and Hessian to construct an approximation to the likelihood function by Taylor expansion[2].

Go to the same folder where you ran the previous tutorial. Create a new folder called “Hessian”, and copy the tree, alignment and control files into this folder. Open the control file “mcmctree.ctl” using your favorite text editor. Set the usedata variable to 3:

```
usedata = 3    * 0: no data; 1:seq; 2:approximation; 3:out.BV (in.BV)
```

Now run the MCMCTree program:

```
C:\Users\Mario\Tutorial\Hessian> mcmctree mcmctree.ctl
```

MCMCTree will create three temporary files for each partition in the alignment: tmp1.txt containing the alignment for the first partition, tmp1.tree with the tree for the first alignment, tmp1.ctl a control file for the BASEML program to analyze tmp1.tree and tmp1.txt, and so on. MCMCTree will then call BASEML three times to perform maximum likelihood estimation of branch lengths, gradient and Hessian for the three partitions (for this step to work properly, you need to have set your operating system PATH variable correctly, see above).

Now look for a file called “out.BV”. This file contains the three sets of branch lengths, gradient and Hessian for each partition. The beginning of the file should look like

```
7
(((human: 0.025136, (chimpanzee: 0.013241, bonobo: 0.010461): 0.014365): 0.013406, ...
0.025520 0.013406 0.025136 0.014365 0.013241 0.010461 0.029460 0.041605 0.022252 ...
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 ...
Hessian
-96520.86 -3713.45 -10369.47 -11757.41 -13729.28 -20652.92 -8997.32 -4832.46 ...
-3713.45 -176562.71 -4315.44 -143.38 -13722.26 -9111.27 -21294.71 -7017.87 ...
```

The first line is the number of species (7), then there is the (unrooted) tree with branch lengths, then the vector of $2 \times 7 - 3 = 11$ branch lengths, then the gradient (usually all zero values) and the Hessian matrix ($11 \times 11 = 121$ values). If you scroll down the file, you will see another block with another tree, set of branch lengths, etc. corresponding to the second partition; and further down, a final block corresponding to the third partition. Every time BASEML finishes analyzing a partition, it writes the tree, gradient and Hessian to a file called rst2. MCMCTree collects the rst2 files and joins them together in the larger out.BV file.

Return to the parent directory (in my example, C:\Users\Mario\Tutorial), and create a new folder called “approx01”. Into this folder copy the tree, alignment, control and out.BV files. Go into the new folder, and rename file out.BV as in.BV. Open the mcmctree.ctl file and modify the usedata variable:

```
usedata = 2    * 0: no data; 1:seq; 2:approximation; 3:out.BV (in.BV)
```

and then run the program

```
C:\Users\Mario\Tutorial\approx01> mcmctree mcmctree.ctl
```

MCMCTree will now perform divergence time estimation, but this time using the gradient and Hessian to approximate the likelihood. As with any MCMC, you need to run the analysis again to check for convergence. Create a new folder and call it approx02 and repeat the MCMC step of the analysis (it is not necessary to repeat the first step of estimation of branch lengths, gradient and Hessian). Compare the results obtained with the approximate method with those from the exact method. They should be very similar. Compare the time used by both type of analyses (look for a Time used line in the out file of each analysis).

Note that the approximate method should not be used with the strict clock (clock=1) as the approximation in this case is very poor and the results will be incorrect[2].

Tutorial 3: Changing the time scale

In the independent rates model (clock=2) the rate (r) follows a log-normal distribution

$$f(r \mid \mu, \sigma^2) = \frac{1}{r\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}\left[\log(r/\mu) + \sigma^2/2\right]^2\right\}$$

with mean

$$E(r) = \mu$$

and variance

$$Var(r) = \left(e^{\sigma^2} - 1\right)\mu^2.$$

The distribution is completely specified by μ and σ^2. Parameter σ^2 is the variance of $\log(r)$.

Let's write t for the time. If we change the time scale by a constant factor so that the new time is $t' = kt$ then the substitution rate needs to be re-scaled accordingly so that the new rate is $r' = r/k$. For a constant a, $E(aX) = aE(X)$ and $Var(aX) = a^2 Var(X)$. Therefore the rate r' in the transformed time scale has mean

$$E(r') = E(r/k) = \frac{1}{k}E(r) = \frac{\mu}{k}$$

and variance

$$Var(r') = Var(r/k) = \frac{1}{k^2}Var(r) = \frac{1}{k^2}\left(e^{\sigma^2} - 1\right)\mu^2 = \left(e^{\sigma^2} - 1\right)\left(\frac{\mu}{k}\right)^2.$$

It is easy to see that r' is log-normally distributed with parameters σ^2 and $\mu' = \mu/k$.

When changing the time scale, we need to change the rate prior accordingly. If μ has a gamma prior

$$f(\mu) = \mathrm{Gamma}(\mu \mid \alpha, \beta),$$

then μ' must have the equivalent gamma prior

$$f(\mu') = \mathrm{Gamma}(\mu' \mid \alpha, k\beta).$$

However, note that σ^2 is unchanged during the scale transformation, therefore the prior on σ^2 must remain unchanged. The birth and death rates (but not the sampling fraction) in the birth-death process also need to be changed. Because these are rates, they must be divided by k. For example, in the ape phylogeny above, if we change the time scale from 100 Myr to 1 Myr (i.e. $t' = 100t$ and $r' = r/100$), the tree with rescaled fossil calibrations would look like

```
7 1
((((human, (chimpanzee, bonobo)) '>6<8', gorilla),
(orangutan, sumatran)) '>12<16', gibbon);
```

and the RootAge, BDparams and rgene_gamma parameters in the control file would need to be modified accordingly (compare with the file on page 2)

```
        seed = -1
     seqfile = mtCDNApri123.txt
    treefile = mtCDNApri.trees
     outfile = out
       ndata = 3
     usedata = 1    * 0: no data; 1:seq; 2:approximation; 3:out.BV (in.BV)
       clock = 2    * 1: global clock; 2: independent; and 3: correlated rates
     RootAge = '<100.0'  * safe constraint on root age, used if no fossil for root.
       model = 0    * 0:JC69, 1:K80, 2:F81, 3:F84, 4:HKY85
       alpha = 0    * alpha for gamma rates at sites
       ncatG = 5    * No. categories in discrete gamma
   cleandata = 0    * remove sites with ambiguity data (1:yes, 0:no)?
     BDparas = .01 .01 0   * birth rate, death rate, sampling
 kappa_gamma = 6 2      * gamma prior for kappa
 alpha_gamma = 1 1      * gamma prior for alpha
 rgene_gamma = 2 200     * gamma prior for rate for genes
sigma2_gamma = 1 10    * gamma prior for sigma^2     (for clock=2 or 3)
    finetune = 1: .1 .1 .1 .1 .1 .1 * auto (0 or 1) : times, rates, etc.
       print = 1
      burnin = 2000
    sampfreq = 2
     nsample = 20000
```

Without auto-finetune, the finetune parameters should also be modified to achieve better mixing. Running the analysis with the new time scale leads to the same results as before, with all posterior times (rates) multiplied (divided) by constant k.

In the correlated rates model (clock=3), the rate follows a log-normal distribution with parameters μ and $t\sigma^2$. Note that the variance of $\log(r)$ is now a function of the time t. With the change of time scale this variance becomes $t'\sigma'^2$, where $\sigma'^2 = \sigma^2/k$. If the prior on σ^2 is $\text{Gamma}(\sigma^2 \mid \alpha, \beta)$, then for σ'^2 it is $\text{Gamma}(\sigma'^2 \mid \alpha, k\beta)$. For example, if the time scale is 100Myr and we are using correlated rates, the control file would look like

```
        seed = -1
     seqfile = mtCDNApri123.txt
    treefile = mtCDNApri.trees
     outfile = out
       ndata = 3
     usedata = 1    * 0: no data; 1:seq; 2:approximation; 3:out.BV (in.BV)
       clock = 3    * 1: global clock; 2: independent; and 3: correlated rates
     RootAge = '<1.0'  * safe constraint on root age, used if no fossil for root.
       model = 0    * 0:JC69, 1:K80, 2:F81, 3:F84, 4:HKY85
       alpha = 0    * alpha for gamma rates at sites
       ncatG = 5    * No. categories in discrete gamma
   cleandata = 0    * remove sites with ambiguity data (1:yes, 0:no)?
     BDparas = 1 1 0   * birth, death, sampling
 kappa_gamma = 6 2      * gamma prior for kappa
 alpha_gamma = 1 1      * gamma prior for alpha
 rgene_gamma = 2 2     * gamma prior for rate for genes
sigma2_gamma = 1 10    * gamma prior for sigma^2     (for clock=2 or 3)
    finetune = 1: .1 .1 .1 .1 .1 .1 * auto (0 or 1) : times, rates, etc.
       print = 1
      burnin = 2000
    sampfreq = 2
     nsample = 20000
```

and with a time scale of 1 Myr the control file would be

```
        seed = -1
     seqfile = mtCDNApri123.txt
    treefile = mtCDNApri.trees
     outfile = out
       ndata = 3
     usedata = 1    * 0: no data; 1:seq; 2:approximation; 3:out.BV (in.BV)
       clock = 3    * 1: global clock; 2: independent; and 3: correlated rates
     RootAge = '<100.0'  * safe constraint on root age, used if no fossil for root.
       model = 0    * 0:JC69, 1:K80, 2:F81, 3:F84, 4:HKY85
       alpha = 0    * alpha for gamma rates at sites
       ncatG = 5    * No. categories in discrete gamma
   cleandata = 0    * remove sites with ambiguity data (1:yes, 0:no)?
     BDparas = .01 .01 0   * birth, death, sampling
 kappa_gamma = 6 2      * gamma prior for kappa
 alpha_gamma = 1 1      * gamma prior for alpha
 rgene_gamma = 2 200     * gamma prior for rate for genes
sigma2_gamma = 1 1000    * gamma prior for sigma^2     (for clock=2 or 3)
    finetune = 1: .1 .1 .1 .1 .1 .1 * auto (0 or 1) : times, rates, etc.
       print = 1
      burnin = 2000
    sampfreq = 2
     nsample = 20000
```

As an exercise, repeat the analysis using a time scale of 100 Myr and clock=3 and compare the results with those from clock=2 (from tutorial 1). Then repeat changing the time scale to 1 Myr.

Tutorial 4: Approximate likelihood with protein data

If the input alignment are amino acid sequences, a few extra steps are necessary if we want to use the approximate method. We will work with the “abglobin.aa” alignment file in the “examples” directory. This file contains globin sequences for five mammals. Create a new directory called “mcmctree-globin”, and copy the “abglobin.aa” file into it. Using your favorite text editor (Notepad, TextEdit, etc.), create the tree file and call it “abglobin.tree”:

```
  5  1
((((rabbit, rat), human), goat-cow), marsupial)'B(1.7,1.9)';
```

We use a time unit of 100 My and so we calibrate the marsupial/placental divergence to be between 170-190 Ma. Copy the primates “mcmctree.ctl” file (the same tutorial 2 above) into the “mcmctree-globin” directory. Open the file in your favourite text editor and edit it:

```
       seed = -1
    seqfile = abglobin.aa
   treefile = abglobin.tree
    outfile = out
      ndata = 1
    seqtype = 2    * 0: nucleotides; 1:codons; 2:AAs
    usedata = 3    * 0: no data; 1:seq like; 2:normal approximation; 3:out.BV (in.BV)
      clock = 2    * 1: global clock; 2: independent rates; 3: correlated rates
    RootAge = '<1.0'  * safe constraint on root age, used if no fossil for root.
      model = 0    * 0:JC69, 1:K80, 2:F81, 3:F84, 4:HKY85
      alpha = 0    * alpha for gamma rates at sites
      ncatG = 5    * No. categories in discrete gamma
  cleandata = 0    * remove sites with ambiguity data (1:yes, 0:no)?
    BDparas = 1 1 0.1   * birth, death, sampling
kappa_gamma = 6 2      * gamma prior for kappa
alpha_gamma = 1 1      * gamma prior for alpha
rgene_gamma = 2 2     * gamma prior for rate for genes
sigma2_gamma = 1 10    * gamma prior for sigma^2     (for clock=2 or 3)
    finetune = 1: .1 .1 .1 .1 .1 .1 * auto (0 or 1): times, rates, mixing, paras, RateParas, FossilErr
       print = 1
      burnin = 2000
    sampfreq = 2
     nsample = 20000
```

Now go to the command and run the program

```
C:\Users\Mario\Tutorial\mcmctree-globin> mcmctree mcmctree.ctl
```

MCMCTree will generate the tmp1.ctl, tmp1.tree and tmp1.txt files and will call CODEML to generate the Hessian matrix for the protein data. However, MCMCTree will use the simplest protein model (Poisson and no gamma rates) which is not very useful for real data analysis. Delete the “out.BV” and “rst” files that were generated. Copy file “wag.dat” from the “dat” directory into “mcmctree-globin”. Open “tmp1.ctl” with your favorite text editor and edit it

```
seqfile = tmp1.txt
treefile = tmp1.trees
outfile = tmp1.out
noisy = 3
seqtype = 2
model = 2  * 2: Empirical
aaRatefile = wag.dat
fix_alpha = 0
alpha = .5
ncatG = 4
Small_Diff = 0.1e-6
getSE = 2
method = 1
```

This new control file will run with an empirical rate matrix (WAG) and with gamma rates among sites. Now you can call CODEML

```
C:\Users\Mario\Tutorial\mcmctree-globin> codeml tmp1.ctl
```

to generate the appropriate Hessian matrix using WAG+Gamma. Rename file “rst2” as “in.BV” and now you have a nice Hessian matrix calculated using WAG+Gamma. Now you can edit “mcmctree.ctl” and set

```
usedata = 2
```

and then run MCMCTree with the approximate method. The steps are the same as for the last part of tutorial 2.

You can also use a similar procedure if you want to run codon models. Furthermore, if you have several partitions, say RNA genes and some protein sequences, you can run BASEML on the RNA data with a nucleotide substitution model, and CODEML on the proteins. Then you can join together the two “rst2” files for each data type into one larger “in.BV” file, and you can then run MCMCTree on a combined nucleotide/protein data set. More details about this, perhaps, in a future tutorial.

Tutorial 5: MCMC estimation of times with infinitely many sites

This tutorial assumes that you have a fairly good understanding of Bayesian statistics, the theory of divergence time estimation, and that you are pretty competent using PAML, code compilers, etc. You should have read the infinite-sites theory in [8, 5, 3]. The “Infinitesites” program estimates divergence times assuming infinitely long sequence alignments. Windows users should have a copy of the program in the bin folder in the PAML distribution. Users of Unix-type operating systems (Mac, Linux, etc.) need to compile the program.

There are two types of analysis that can be carried out: (1) Estimation under the clock[8, 3], and (2) Estimation under relaxed clocks[5].

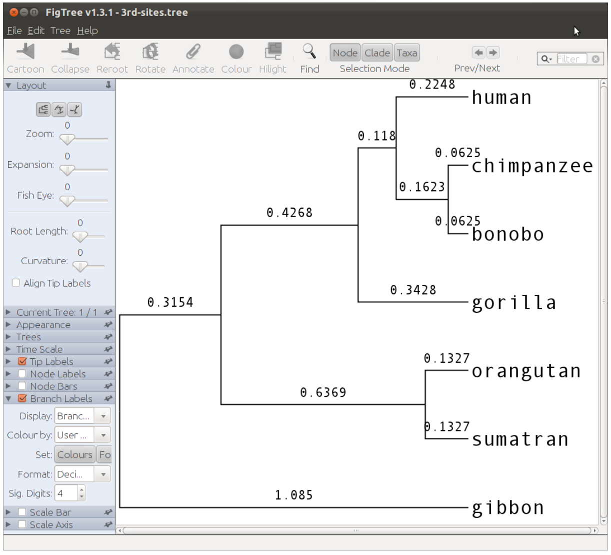
(1) Estimation under the clock: First you need to prepare a special file that contains all the distances (in substitutions per site) from the tips to the internal nodes in the rooted tree. The distances are usually calculated by maximum likelihood from a species phylogeny under the clock, using BASEML or CODEML. For example, for the primate phylogeny of tutorial 1, I used BASEML to estimate the branch lengths under the clock for the third codon sites, using the HKY+G5 substitution model. The tree with branch lengths from the "mlb" file is

```
((((human: 0.224767, (chimpanzee: 0.062487, bonobo: 0.062487): 0.162279): 0.118038,
gorilla: 0.342804): 0.426831, (orangutan: 0.132715, sumatran: 0.132715): 0.636921):
0.315408, gibbon: 1.085044);
```

The tree with node numbers is

```
((((1_human, (2_chimpanzee, 3_bonobo) 12 ) 11 , 4_gorilla) 10 , (5_orangutan,
6_sumatran) 13 ) 9 , 7_gibbon) 8 ;
```

and the tree plotted with FigTree is



Create a folder called "inf", and inside prepare a file called "FixedDsClock1.txt" that has the distances:

```
7
1.085 0.7696 0.3428 0.2248 0.0625 0.1327
```

For a tree with s species under the clock, there will be $s-1$ distances, one for each internal node in the tree. The first line in the file is the number of species in the phylogeny (in this case 7), then the 6 distances are provided, starting with the distance from the tips to the root (node 8), which is 1.085, and then the distances to the internal nodes (nodes 9 to 13 in that order). For example, the distance from orang to node 9 is $0.1327 + 0.6369 = 0.7696$. If you use BASEML or CODEML with clock=1, the distances are provided in the right order in the output file (usually mlb or mlc) below the line of the log-likelihood (lnL). Copy the tree, alignment and control files from tutorial 1 into the “inf” folder. From the alignment file (“mtCDNApri123.txt”) delete the first two alignments (1st and 2nd sites). Open the control file and edit it:

```
seed = -1
seqfile = mtCDNApri123.txt
treefile = mtCDNApri.trees
outfile = out
ndata = 1
seqtype = 0  * 0: nucleotides; 1:codons; 2:AAs
usedata = 1  * 0: no data; 1:seq like; 2:normal approximation; 3:out.BV (in.BV)
clock = 1    * 1: global clock; 2: independent rates; 3: correlated rates
```

We can now run the program:

```
C:\Users\Mario\Tutorial\inf> Infinitesites
```

The program assumes that the distances are perfect MLEs (variance zero) estimated from an infinitely long sequence alignment, and will use these, together with the prior on the rate and times to calculate the posterior of the root age (t_8). See the appropriate equations in [8] for details on how this is done. Note that the substitution model variables (kappa, alpha, etc.) in the mcmctree.ctl file have no effect, since the model used is the one specified in BASEML to estimate the branch lengths. At the end of the run, the program outputs the posterior summary to the screen:

```
mean (95% CI) CI-width for times
Node 8:  0.230088 ( 0.221717, 0.246060) 0.024343
Node 9:  0.163204 ( 0.157266, 0.174532) 0.017267
Node 10: 0.072695 ( 0.070050, 0.077741) 0.007691
Node 11: 0.047672 ( 0.045937, 0.050981) 0.005044
Node 12: 0.013254 ( 0.012772, 0.014174) 0.001402
Node 13: 0.028141 ( 0.027117, 0.030094) 0.002977
mean & 95% CI for rates
gene 1: 4.715580 ( 4.409499, 4.893628)
```

Note that the posterior distribution is one-dimensional. If we know the distribution of the root age, we know the distribution of all the other ages. The posterior means of times are proportional: $t_9/t_8 = d_9/d_8$. If we plot the mean times vs. the CI widths, the points will form a perfectly straight line.

The above assumes one locus. To use more than one locus, the branch lengths (node ages) must be proportional across loci. The FixedDsClock1.txt file then has one additional line for each additional locus, with the age of the root (distance) on it.

(2) Estimation with relaxed clocks: When the clock is relaxed the situation becomes more complicated. Having infinitely many sites is not enough to achieve a one-dimensional posterior distribution of the node ages. We need infinitely many loci as well [5]. Infinite sites will estimate divergence times using a finite number of loci, but each loci with an infinite number of sites. Infinitesites needs a list of trees (one per locus) with the MLE of branch lengths for each tree. In theory, the tree should be unrooted with branch lengths estimated without the clock. However, Infinitesites currently assumes the tree is rooted and the branch lengths are estimated without the clock. The program will then sum up the two branch lengths around the root. I used BASEML to estimate the branch lengths on the rooted tree with no clock, HKY+G5 model, for each one of the three alignments (codon positions) in the primate data of tutorial 1. You should use the tree file with fossil calibrations as the user tree in BASEML. Create a folder called “inf-loci” and copy the tree with fossil calibrations, the mcmctree.ctl file and the alignment into this folder. Now prepare a text file called “FixedDsClock23.txt” and copy the ML trees from BASEML:

```
7
((((human: 0.029043, (chimpanzee: 0.014557, bonobo: 0.010908): 0.016729): 0.015344,
gorilla: 0.033888): 0.033816, (orangutan: 0.026872, sumatran: 0.022437): 0.069648):
0.073309, gibbon: 0.024637);
((((human: 0.012463, (chimpanzee: 0.002782, bonobo: 0.003835): 0.003331): 0.004490,
gorilla: 0.014278): 0.006308, (orangutan: 0.010818, sumatran: 0.008845): 0.030551):
0.004363, gibbon: 0.029246);
((((human: 0.270862, (chimpanzee: 0.066698, bonobo: 0.056883): 0.124104): 0.139082,
gorilla: 0.310797): 0.391342, (orangutan: 0.152555, sumatran: 0.114176): 0.696518):
0.017607, gibbon: 1.394718);
```

Now you can run the program

```
C:\Users\Mario\Tutorial\inf-loci> Infinitesites
```

The output should look like:

```
Posterior mean (95% Equal-tail CI) (95% HPD CI) HPD-CI-width
t_n8      0.1942 (0.1610, 0.2364) (0.1602, 0.2306) 0.0704 (Jeffnode 12)
t_n9      0.1557 (0.1438, 0.1623) (0.1456, 0.1633) 0.0178 (Jeffnode 11)
t_n10     0.0921 (0.0827, 0.1039) (0.0820, 0.1031) 0.0210 (Jeffnode 10)
t_n11     0.0622 (0.0588, 0.0695) (0.0582, 0.0679) 0.0097 (Jeffnode 9)
t_n12     0.0242 (0.0185, 0.0302) (0.0185, 0.0302) 0.0117 (Jeffnode 8)
t_n13     0.0453 (0.0348, 0.0563) (0.0347, 0.0561) 0.0214 (Jeffnode 7)
r_left_L1 0.4834 (0.2852, 0.7186) (0.2584, 0.6878) 0.4294
r_left_L2 0.1727 (0.0795, 0.3591) (0.0604, 0.3111) 0.2507
r_left_L3 4.3565 (1.7766, 8.8623) (1.5023, 8.1197) 6.6173
mu_L1     0.5126 (0.4376, 0.6053) (0.4334, 0.5976) 0.1642
mu_L2     0.1736 (0.1366, 0.2302) (0.1338, 0.2213) 0.0875
mu_L3     3.9529 (3.1518, 4.8803) (3.1001, 4.7768) 1.6767
```

```
sigma2_L1 0.0542 (0.0167, 0.1446) (0.0107, 0.1194) 0.1087
sigma2_L2 0.1460 (0.0670, 0.3218) (0.0512, 0.2736) 0.2224
sigma2_L3 0.1476 (0.0604, 0.3035) (0.0552, 0.2760) 0.2208
```

In this case, because we only have three loci, the time posterior is not one-dimensional as in the clock case. However, if you could increase the number of loci and make it really large, the time posterior will become one-dimensional and a plot of mean times vs. CI-widths will approach a straight line.

Note that Infinitesites reads the sequence alignment file even though the sequences are ignored. Thus if you have nadata=3 in the control file, you should have at least three sequence alignments in the sequence file (the sequences are read and ignored by the program) and three trees with branch lengths in FixedDsClock23.txt.

For comments and questions about this tutorial please e-mail:
mariodosreis@gmail.com.
Dep. Genetics, Evolution and Environment, University College London, London, UK, WC1E 6BT.

References

[1] M. dos Reis, J. Inoue, M. Hasegawa, R. J. Asher, P. C. Donoghue, and Z. Yang. Phylogenomic datasets provide both precision and accuracy in estimating the timescale of placental mammal phylogeny. *Proc Biol Sci*, 279(1742):3491–500, 2012.

[2] M. dos Reis and Z. Yang. Approximate likelihood calculation on a phylogeny for Bayesian estimation of divergence times. *Mol Biol Evol*, 28(7):2161–72, 2011.

[3] Mario dos Reis and Ziheng Yang. The unbearable uncertainty of bayesian divergence time estimation. *Journal of Systematics and Evolution*, 51(1):30–43, 2013.

[4] J. Inoue, P. C. Donoghue, and Z. Yang. The impact of the representation of fossil calibrations on Bayesian estimation of species divergence times. *Syst Biol*, 59(1):74–89, 2010.

[5] B. Rannala and Z. Yang. Inferring speciation times under an episodic molecular clock. *Syst Biol*, 56(3):453–66, 2007.

[6] J. L. Thorne, H. Kishino, and I. S. Painter. Estimating the rate of evolution of the rate of molecular evolution. *Mol Biol Evol*, 15(12):1647–57, 1998.

[7] Z. Yang. PAML 4: phylogenetic analysis by maximum likelihood. *Mol Biol Evol*, 24(8):1586–91, 2007.

[8] Z. Yang and B. Rannala. Bayesian estimation of species divergence times under a molecular clock using multiple fossil calibrations with soft bounds. *Mol Biol Evol*, 23(1):212–26, 2006.

[9] Ziheng Yang. *Computational Molecular Evolution*. Oxford University Press, Oxford, 2006.