MCMCcoal

Markov Chain Monte Carlo Coalescent Program

Version 1.1, April 2005

Ziheng Yang
Department of Biology
University College London
Gower Street, London WC1E 6BT
Email: z.yang@ucl.ac.uk
Phone: +44 (20) 7679 4379
Fax: +44 (20) 7679 7096

Introduction

The ANSI C program MCMCcoal implements the Bayes Markov chain Monte Carlo (MCMC) algorithm of Yang and Rannala (2003) for estimating species divergence times and population sizes from DNA sequence alignments at multiple loci. Consult the paper for details of the model assumptions and the theory. The program can also be used to estimate θ from a sample from one modern species under the standard neutral coalescent. A simulation program (MCcoal) can be compiled from the same source code, to simulate sequence data sets under the same model; read the section "The simulation program (MCcoal)".

Getting Started

- Compiling the program (already done if you are using MS Windows)
- Execute the program by typing the following command at the command line

  `MCMCcoal`

  The program reads the control file `MCMCcoal.dat` by default.
  If you type

  `MCMCcoal MCMCcoalChenLi4s.dat`

  The program will use the data set of Chen and Li (2001) and duplicate the analysis of Rannala and Yang (2003: table 2 "Posterior (53 loci)").
  If you type

  `MCMCcoal MCMCcoalYu2001.dat`

  The program will use the data set of Yu et al. (2001), which is a sample from the human species, to estimate θ. This should duplicate the result of table 3 (first row) in Rannala and Yang (2003).
  (Note: The output on one line is longer than 80 characters, so you should make the window wider, to 110 characters. You can try to drag the window to make it wider. On MS Windows, you should right-click on the title bar, then choose Properties, and change Layout - Window size.)
- Print out a copy of the control file and read it together with this document.

Compiling the Program

Win32 executables are included, so you can use the .exe executables on windows 95/98/NT/2K/XP.

If you are using UNIX (linux, MAC OSX), you should remove the .exe files and then compile the program. You need to do this only once. You can use gcc or any ANSI C-compatible compiler. The program used some source codes from my PAML package (`paml.h`, `tools.c`, `treesub.c`), and I did not bother to edit those files. Try the following:

```
cc -o MCMCcoal -O3 MCMCcoal.c tools.c -lm
```

The –o flag specifies the name of the output executable file, -O3 optimizes the code, and –lm links to the math library. You might want to change some of flags. For example you might use gcc instead of cc, -fast, -O4 instead of –O3, and you might not need the –lm flag.

The Control File

Analysis of Multiple Species Data

Three example control files are included in the package, for three different runs. `MCMCMCcoalChenLi4s.dat` provides the control file to analyze the Chen & Li (2002) data set, included in the package. This should duplicate the results of table 2 "Posterior (53 loci)" in Rannala and Yang (2003). The control file `MCMCcoalYu2001.dat` is for analyzing a sample from one single species to estimate parameter $\theta = 4N\mu$. This should duplicate table 3 (first row) in Rannala and Yang (2003). This is explained in the next subsection.

The description below uses the third control file `MCMCMCcoal.dat`. This is not for a real data set, but is used below to explain the general features of the program. You can print out the copy included in the package. Also look at (but do not print) the included example sequence data file `NeksTestData.txt` for the data format. In the following, "population size" means $\theta = 4N\mu$, the product of population size and the mutation rate μ, and "speciation time" means τ, the age of species divergence multiplied by the mutation rate μ. See Rannala and Yang (2003) for the notation.

```
NeksTestData.txt
-1
4  H  C  G  O
   3  2  1  1
(((H, C), G), O);

    0                    # use data? 0: prior; 1: posterior
10000   2  100000         # burnin output nsample

0.05  0.01  0.01  0.02  0.8   # finetune for GBtj, GBtip, theta, tau, mix

   2     2     2      2      2     7.4     4    20   # a_gamma
2000  2000  2000   2000   2000    1000  2500  4000   # b_gamma
```

Below are details of the control variables.

`NeksTestData.txt`

sequence data file name.

`-1`

Random number seed to determine the starting point of the MCMC run. If you use –1, the program will pick up a seed at random, and different runs will use different random number seed. If you use a positive integer, the program will use it as the seed, in which case different

runs of the program should produce exactly identical results.

```
4  H  C  G  O
   3  2  1  1
```

There are 4 species in the species tree: H (human), C (chimp), G (gorilla), O (orang). Species names are case-sensitive.
The next line says that there are 3 humans, 2 chimps, 1 gorilla, and 1 orang. These numbers are like upper bounds and serve two purposes. First, they are used to determine which θ parameters are involved in the model and should be estimated. The specification here means that we should estimate θ_H and θ_C, as well as the three θ's for the ancestral populations, but not θ_G and θ_O. (The other parameters are the speciation times τ_{HCGO}, τ_{HCG}, τ_{HC}.) Second, they specify the maximum number of sequences at a locus. The exact rules are as follows: (i) If the number of sequences specified here is 1, that species (such as the gorilla) cannot have more than 1 sequence at any locus in the sequence data file; (ii) If a species never has two or more sequences at any locus, you must use 1 (not 2 or 3) for that species here. (iii) The total number of sequences at any locus in the data file should not exceed the total number specified here (7 in the example). So given the specifications in this example, feasible cases in the data file include $H_2C_3G_0O_0$ and $H_2C_3G_1O_1$, but not $H_0C_0G_2O_1$. Each locus in the data file should have at least 2 sequences.

In case the above is confusing, you can just specify the maximum number, among loci, of sequences for each species.

```
(((H, C), G), O);
```

The species tree is fixed.

```
    0                    # use data? 0: prior; 1: posterior
```

When 0 is used, the MCMC runs without using the sequence data (except for the number of loci and number sequences at each locus); it should thus approximate the prior for parameters. This option can be used to test the algorithm. When 1 is used, the MCMC run uses sequence data to approximate the posterior.

```
10000   2  100000          # burnin output nsample
```

The first number (10000) is the number of generations of the MCMC used as burnin. These are discarded before samples are taken from the chain. After the burnin, samples are taken every 2 generations, , and 100,000 samples are taken. The total number of MCMC generations is burnin + output × nsample. Parameter values sampled from the MCMC run are collected in a plain text file named mcmc.out. Note that this file can get very large if many samples are taken.

```
0.05  0.01  0.002  0.005  1.5   # for GBtj, GBtip, theta, tau, mix
```

These are the five fine-tuning variables ε_1, ε_2, ε_3, ε_4, ε_5 in Rannala & Yang (2003). They are used in the five MCMC proposal steps for (1) changing internal node ages in the gene tree, (2) pruning and re-grafting nodes in the gene tree, (3) updating θs, (4) updating τs using the rubber-band algorithm, and (5) the mixing step. You should run the MCMC for a small number of generations and look at the screen output for the five acceptance proportions (see below). There are no hard rules, but theory says that those proportions should best be in the range (0.1, 0.7). The results are entirely untrustworthy if any of the proportions are 0. So if the recorded proportion is too high (low), you should increase (reduce) the corresponding fine-tuning parameter. Note if there is only one species, step 4 is not used and ε_4 is irrelevant.

```
   2     2     2      2      2     7.4     4    20   # a_gamma
2000  2000  2000   2000   2000    1000  2500  4000   # b_gamma
```

These two lines specify the values of α and β in the gamma priors for parameters θs and τs in the model. The mean of the gamma is α/β, while the variance is α/β^2. Note that α is called the shape parameter of the gamma distribution while β is the scale parameter. If $\alpha \leq 1$, the distribution has an L shape, with the most likely values near 0, while if $\alpha > 1$, the distribution

has a peak in the middle. **I suggest you always use $\alpha > 1$** since it is almost certain that all θ and τ parameters are strictly positive.

For n species, the number of θ parameters is $c + (n - 1)$, where c is the number of species for which more than one sequence is available at some loci in the data. In the example above, $c = 2$ (for H and C). The number of speciation time parameters τs is $(n - 1)$, corresponding to the $(n - 1)$ ancestral nodes. There should be $c + (n - 1) \times 2$ values on each of the two lines.

Parameters θs are before parameters τs, but the exact order of θs and τs is determined by the program rather than by the user. So what you should do is to count the total number of parameters, and supply the right number of α and β values on those two lines. You then start the program and stop it (Ctrl-C), and use the screen output to figure out the order of the parameters. In the example, the following screen output suggests that the parameters are in the order θ_H, θ_C, θ_{HC}, θ_{HCG}, θ_{HCGO}, $(\tau_{HCGO} - \tau_{HCG})$, $(\tau_{HCG} - \tau_{HC})$, and τ_{HC}. Note that the speciation times are defined as gaps.

```
Gamma prior: mean +- SE (95% CI) for theta's and tau's
theta_H                     0.00100 +- 0.00071 (0.00012, 0.00279)
theta_C                     0.00100 +- 0.00071 (0.00012, 0.00279)
theta_HC                    0.00100 +- 0.00071 (0.00012, 0.00279)
theta_HCG                   0.00100 +- 0.00071 (0.00012, 0.00279)
theta_HCGO                  0.00100 +- 0.00071 (0.00012, 0.00279)
tau_HCGO - tau_HCG          0.00740 +- 0.00272 (0.00307, 0.01361)  tau_HCGO  0.01400
tau_HCG - tau_HC            0.00160 +- 0.00080 (0.00044, 0.00351)  tau_HCG   0.00660
tau_HC                      0.00500 +- 0.00112 (0.00305, 0.00742)  tau_HC    0.00500
```

Analysis of Data from One Species

Run the analysis by

```
MCMCcoal MCMCcoalYu2001.dat
```

The control file `MCMCcoalYu2001.dat` is for analyzing a sample from one single species to estimate parameter $\theta = 4N\mu$ under the standard neutral coalescent. This should duplicate table 3 (first row) in Rannala and Yang (2003).

The control file is copied below. I think everything is explained above. Here are a few more notes. There is only one species so there is no tree. Multiple loci can be used in the data file, but one locus will work fine. The proposal step for changing tau is not used in the MCMC for analyzing data of one species. As a result, the fourth acceptance ratio printed on the monitor is always 0. You still need all the five numbers in the control file, and the program reads the fourth number and ignores it.

There is only one parameter in the model: θ. Note that the mutation rate is assumed to be the same in the program so that only one θ is involved in the model.

The printout on the monitor will include the current posterior mean of θ, and the current posterior means of μt_{MRCA} for the loci. If you have many loci, only the first few μt_{MRCA} are printed in the monitor.

If you have g loci, the output file mcmc.out will list $g + 2$ columns. The first is θ, followed by the μt_{MRCA} for the g loci. The μt_{MRCA} are not parameters, but sometimes people are interested in them, so they are included in the output and summarized as well. The last column is the log likelihood, which you should ignore.

```
yu2001.txt
-1
1 H
  100                    # max # of sequences

    1                    # use data? 0: prior; 1: posterior
2000   2  2000           # burnin output nsample

0.8  0.0001  0.0005  0.01  1   # finetune for GBtj, GBtip, theta, tau, mix

   2  # a_gamma
2000  # b_gamma
```

Sequence Data File

I include an example data file `NeksTestData.txt`, which works with the default control file `MCMCcoal.dat`. The other data file, `ChenLiData4s.nuc`, has the 53-loci data of Chen and Li (2001). You can use those files as examples to assemble your data in the right format. The format is basically phylip/paml format but you need to provide more information. The beginning of `NeksTestData.txt` is as follows.

```
   3

0 2 1 1
       4      50
C1          ?CCAGGCGG? TTACGCGCTA TTAGGCCGTT AGAATGCACG CGAGTCTCCT
C2          YCCAGGCGGT TTACGC---- TTAGGCCGTT AGAATGCACG CGAGTCTCCT
G           YCCAGGCGGT TTACGCGCTA TTAGGCCGTT AGAATGCACG CGAGTCTCCT
O           CCCAGGAGGT TTTCGCGCTG TTAAGGCGCT AGAATGCACG TGCCTTTCTT
```

The first number means 3 loci.

There are then 3 blocks for the 3 loci. Each block begins by telling the program how many sequences are in the data for each species. So the first locus has 0 Humans, 2 Chimps, 1 Gorilla, and 1 Orang. The order of the species is fixed according to the control file MCMCcoal.dat.

The rest of the block should be the same as a standard paml/phylip sequence data file; see the PAML manual if you need more details. The locus has 4 (= 0 + 2 + 1 + 1) sequences, and 50 nucleotides in each sequence. The 4 sequences have to be in the right order; that is, they are 2 chimps, followed by 1 gorilla, and followed by 1 orang. Similarly, the second locus has 3 humans, 0 chimps, 1 gorilla, and 1 orang, while the third locus has 1 human, 2 chimp, 1 gorilla, and 1 orang (see the file NeksTestData.txt). Different numbers of sequences can be used at different loci. For the program to run, each locus should have at least 2 sequences.

By default, alignment gaps in the sequence data are treated as ambiguity characters, which seems a valid choice, since in closely related species, alignment gaps are rare, where gaps at the ends of the alignments do represent undetermined nucleotides. This is the case for the Chen & Li data. It is not possible to tell the program to remove ambiguity characters at some loci while keeping them at others. Details of likelihood calculation concerning ambiguity characters are in Yang (2000). Dealing with ambiguity characters slows down the program. So if you would rather have ambiguity characters at some loci removed, you should do that yourself before assembling the alignments into one data file. You can use baseml to do that. Otherwise if you want ambiguity characters removed at every locus, you can change `com.cleandata = 0` into `com.cleandata = 1` at the beginning of the routine `main()` in the file `MCMCcoal.c` and recompile.

Running the program

For MCMC runs, parameters θ and τ included in the species tree, if any, are ignored. The starting values of the parameters θ and τ are generated as random numbers, and the starting gene trees and coalescent times are generated from their priors given those values of parameters.

You should do a short run of the program to understand the order of the parameters assumed by the program and to get estimates of the acceptance proportions of the five proposal steps. You can run the program for some seconds or even minutes and then stop it (use Ctrl-C, for example). You use the order of the parameters to specify parameters α and β of the prior gamma distributions, and you use the acceptance proportions to adjust the five fine-tuning parameters in the control file. Note that the optimum fine-tuning parameters might depend on the prior, and on whether you are approximating the prior or the posterior. As a guide, you should try to bring the acceptance proportions into the range (0.1, 0.5). Although the exact values should not matter much to the MCMC run, you should be aware that an acceptance proportion of near 0 may mean lack of convergence.

Read the section on Screen Outputs below for details.

When the MCMC run is finished at long last, the program will read and process the output in `mcmc.out`. It uses kernel density estimation method to smooth the histogram. It also can estimate bi-variate densities. So it asks you how many 2-D densities you want. (This is a bit awkward when the program asks a question after running for a day.) In general the MCMC run should take much longer than summarizing the results.

A separate program ds (for descriptive statistics) can read and summarize output in `mcmc.out`. So you can finish the MCMC run and summarize results later on. If you use ds to process output from other MCMC programs, look at `mcmc.out` for the input data format.

Monitoring Convergence

You can read some notes in Rannala and Yang (2003) or in any books on MCMC. Here are a few tips:

- Run the program at least twice using different random number seeds, and confirm that the results are similar between runs.
- Plot the output in mcmc.out against generation (iteration) and examine the pattern.
- Examine the serial correlation coefficient in the output and confirm that they all go down to 0 with the increase of the lag length.

Outputs

Output Files

The main output is in a file named `out`. The program also output the MCMC states (current values of parameters θ and τ sampled from the MCMC) in another file named `mcmc.out`. Do not use those names for your files as otherwise they will be overwritten. The main output file has information from processing sequence alignments at each locus, as well as summary information generated when the program summarizes the results in `mcmc.out`. The program also uses kernel density estimation methods to smooth the histogram so that you can plot the marginal posterior density for each variable. It can also use kernel smoothing to estimate 2-D densities, but you need many samples (say 100,000) to produce a smooth density.

Screen Outputs

Pay attention to screen outputs to make sure that the specifications in the control file and the format of the sequence data file are correct. Most of the screen output is self-explanatory. The program prints out the tree topology, as well as a population-population table, which you can ignore. It then defines the θ and τ parameters, as explained in the section "The Control File".

The program then reads and processes the sequence data file.

It then starts the Markov chain. The initial values for parameters θs and τs are random numbers, and the starting gene trees and coalescent times at loci are sampled from the prior given θ and τ. The program prints out the initial θs and τs (gaps or inter-arrival times in the species tree), as well as the initial log likelihood lnL0. During the MCMC, the program continuously updates a line of result like the following. (You might have to resize the window width so that the line will fit.)

```
-3.9%  0.25 0.38 0.46 0.54 0.16  0.00102 0.00288 0.00124 0.00399 0.00198 0.03057 0.00338 0.01050 -1560.31
```

The first number is the percentage of the run achieved; a negative number means burnin. This is followed by five numbers, which are the recorded acceptance proportions of the five proposal steps. You use those proportions to adjust the fine-tuning parameters ε_1, ε_2, ε_3, ε_4, ε_5 so that the acceptance proportions lie in the interval (0.1, 0.5). If a proportion is too low, you should decrease the corresponding ε value in the control file, and if the proportion is too high, you should increase the corresponding ε value. Next are the posterior means of the parameters from the MCMC run. In the example, the 8 parameters are in the order θ_H, θ_C, θ_{HC}, θ_{HCG}, θ_{HCGO}, ($\tau_{HCGO} - \tau_{HCG}$), ($\tau_{HCG} - \tau_{HC}$), and τ_{HC}. If no sequence data are used, those means should approach the expected means α/β. For the example data set included, the posterior means of

the eight parameters should approach

```
0.00093 0.00288 0.00133 0.00416 0.00174  0.03193 0.00316 0.01019
```

The last number on the line is the current log likelihood.

The program will then read and process the file mcmc.out to calculate min, max, median, percentiles, and histogram information. This may take quite some time if many samples (say >1M) are collected in the file.

Disk Space and Memory Requirements

The output file `mcmc.out` can become big. For every sample you take, the program prints out the θ and τ parameters as well as the log likelihood. The disk space required is about $(8 \times p + 12) \times$ `nsample` bytes, where p is the number of θ and τ parameters and `nsample` is the number of samples specified in the control file `MCMCcoal.dat`. For the example data, $p = 8$, so the output takes 76MB disk space to take 1,000,000 samples or 760MB of space for 10M samples. For the Chen & Li data of four species, $p = 6$, 60MB is needed for 1M samples and 600MB for 10M samples.

The program does not keep those sampled parameter values in memory, and does not use much memory during the MCMC. When the MCMC run is finished, the program summarizes results by reading the file `mcmc.out`. It does this multiple times to save memory. The big chunk of memory is required when it reads in all the sampled values for one parameter, which is $8 \times$ `nsample` bytes (that is, 8MB for 1M samples or 80MB for 10M samples).

The small program ds can also calculate such descriptive statistics. Run it with

```
ds mcmc.out
```

This does exactly the same calculation as from within `MCMCcoal`. If you want to use it to process output from other programs, look at mcmc.out to see the format of the input file for ds.

To get accurate estimation of the 2-D density, you need many samples, say >100,000.

Technical Notes

Control Variables in the Source File

Occasionally you may want to change some variables at the beginning of the routine `MCMC()` in the source file `MCMCcoal.c`, which control the MCMC run. The default are as follows.

```
mcmc.print=1;
mcmc.moveinnode=1;    /* moves internal nodes in the gene tree as well */
mcmc.speciation[0]=0; /* =0 for gaps; 1 for ages for the prior */
mcmc.speciation[1]=0; /* =0 for gaps; 1 for ages for the output */
mcmc.mixtheta_change=0;
```

`mcmc.print.` If you do not want the MCMC output in the file `mcmc.out`, you can change print to 0. This way, the means of the parameters will still be outputted on the screen, but the program will not write results in the file `mcmc.out`.

`mcmc.moveinnode.` If you want the proposal step updating gene tree topologies to move the tips only but not the internal nodes, you can have `moveinnode = 0`. The default prunes and re-grafts both tips and internal nodes in the gene tree and seems worthwhile if the gene tree is large with many sequences.

`mcmc.speciation[0] & mcmc.speciation[1].` Those two variables specify whether gaps (inter-arrival times) or ages are used to specify the speciation times in the prior and the output. The default uses gaps for both. If you want to use ages for both, you should have `speciation[0]=1` and `speciation[1]=1`. If you want the prior to be specified using gaps in MCMCcoal.dat, but the output (both to the screen and the mcmc.out file) to be in ages, you can have `speciation[0]=0` and `speciation[1]=1`. You cannot specify the prior using ages while having output using gaps; that is, `speciation[0]=1` and `speciation[0]=0` are not permitted.

The MCMC algorithm always operates using the node ages.

mcmc.mixtheta_change = 0. This variable affects the updating of the θ parameters during the mixing step. When it is 0, the algorithm will multiply all parameters including θs, τs, and coalescent times in all gene trees by a constant c around 1. If this variable is 1, the algorithm may multiply some θs by c, dividing some by c, while leaving others unchanged. The decision is based on the correlations between the θ parameter and the τs collected during the MCMC run. The strategy is implemented to overcome the strong correlation between some speciation times (τs) and some θs.

Prior for Speciation Times

Node ages in the species tree are probably more natural to use than gaps. However, node ages are constrained. For example, in the species tree (((H, C), G), O), we have $\tau_{HC} \le \tau_{HCG} \le \tau_{HCGO}$. Thus the gamma priors for the ages cannot be independent. When you specify "independent" gamma priors for node ages in the control file MCMCcoal.dat, the prior used by the program is really the joint gamma density conditioned on the constraint $\tau_{HC} \le \tau_{HCG} \le \tau_{HCGO}$, and the marginal expectations of the node ages are not the same as α/β from the gamma distribution. You can get the expectations of the conditioned distribution by running the MCMC without data, or by using another program such as Mathematica. Perhaps I should use Monte Carlo simulation to calculate them more quickly. Note that the only difficulty with using node ages is that the prior used by the program will be conditioned on the constraint, and it may take quite some trial and error for the prior to match your expectations. For example, if we have complied MCMCcoal to use node ages (`speciation[0]=1` and `speciation[1]=1`; see above) and use the following control file MCMCcoal.dat. The gamma distributions for the speciation times (node ages) τ_{HCGO}, τ_{HCG}, and τ_{HC} have means 0.0014, 0.0066, and 0.0050, which would correspond to divergence times of 14MY, 6.6MY, and 5MY, respectively. However, because of the constraints, the means used by the program for the constrained prior are 0.0143, 0.0072, and 0.0041.

```
ChenLiData4s.nuc
-1
4  H  C  G  O
   1  1  1  1
(((H, C), G), O);
    0                    # use data? 0: prior; 1: posterior
10000   2  100000         # burnin output nsample
0.05  0.01  0.002  0.005  0.5   # finetune for GBtj, GBtip, theta, tau, mix

   2     2     2      14    6.6     5   # a_gamma
2000  2000  2000    1000   1000  1000   # b_gamma
```

Using gaps (inter-arrival times) in the species tree essentially removes the non-independence and makes it easier to specify the prior. So by default, priors for speciation times are specified as priors for the gaps. If the node is not a tip, the prior is specified for the gap between the age of the node and the age of its first (left) daughter node. For the species tree (((H, C), G), O), the speciation time parameters are thus ($\tau_{HCGO} - \tau_{HCG}$), ($\tau_{HCG} - \tau_{HC}$), and τ_{HC}.

If the species tree is ((A, B), (C, D)), the following time parameters are used: ($\tau_{ABCD} - \tau_{AB}$), τ_{AB}, and τ_{CD}. In this case there is a constraint, which is τ_{ABCD}, calculated as ($\tau_{ABCD} - \tau_{AB}$) + τ_{AB}, should be > τ_{CD}. Thus the gamma priors are not really independent even with the use of the gaps for parameters. As mentioned above, the issue only concern the interpretation of the priors for speciation times. To reduce the problem, try to use the smaller gap as parameter. So if you expect $\tau_{AB} < \tau_{CD}$, you should use the tree topology ((C, D), (A, B)), so that the parameters used will be ($\tau_{ABCD} - \tau_{CD}$), τ_{CD}, and τ_{AB}.

Incompatibility with future versions.

I hope to add some other components such as population growth and migration into the program. The current user interface is awkward and will almost certainly change.

The Simulation Program (MCcoal)

I did not spend much time on the interface for the simulation program, and you might have to change some variables in the program to simulate data.

To compile the simulation program, you should delete or comment out (using the pair /* and */ to bracket) the following line

```
#define MetropolisHastings       1
```

and then compile the program and generate the output into an executable with a different file name such as MCcoal.

```
cc -o MCcoal -O3 MCMCcoal.c tools.c –lm
```

The program takes input from the control file `MCcoal.dat`. This is shorter than the control file MCMCcoal.dat for the MCMC program, and uses the first few lines only. Note that the file name in the control file now is used to hold simulated sequence data set.

As explained early in this document, the following variables specify the sequence data file name to be generated, the random number seed, the number of species, species names, and the number of sequences in each species. In the tree, the prefix ':' specifies the node ages (parameters τ). Note that this symbol is conventionally used to specify branch lengths, and the usage here is nonstandard. Also the prefix '#' specifies the population size parameters θs. So the tree in the following specifies the following parameters:
θ_H = 0.001, θ_C = 0.001, θ_{HC} = 0.001, θ_{HCG} = 0.001, θ_{HCGO} = 0.001, τ_{HCGO} = 0.014, τ_{HCG} = 0.0066, and τ_{HC} = 0.005.

```
SimulatedData.txt
1234567
4  H  C  G  O
   3  2  1  1
(((H #0.001, C #0.001) : 0.005 #.001, G) : 0.0066 #.001, O) :.014 #.001;
```

You may have to change some other variables in the source code `MCMCcoal.c`. Search for the following line at the beginning of the routine `SimulateData()`: `nr` is the number of replicate data sets or the number of loci, and `ls` is the sequence length (number of nucleotides).

```
nr=20; com.ls=1000;
```

References

Chen, F.-C., and W.-H. Li, 2001 Genomic divergences between humans and other Hominoids and the effective population size of the common ancestor of humans and chimpanzees. *American Journal of Human Genetics* **68**: 444-456.

Rannala, B., and Z. Yang, 2003 Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. *Genetics* submitted.

Yang, Z. 2002. Likelihood and Bayes estimation of ancestral population sizes in Hominoids using data from multiple loci. *Genetics* **162**: 1811-1823

Yu, N., Z. Zhao, Y. X. Fu, N. Sambuughin, M. Ramsay, T. Jenkins, E. Leskinen, L. Patthy, L. B. Jorde, T. Kuromori, and W. H. Li. 2001. Global patterns of human DNA sequence variation in a 10-kb region on chromosome 1. *Molecular Biology and Evolution* **18**:214-222.