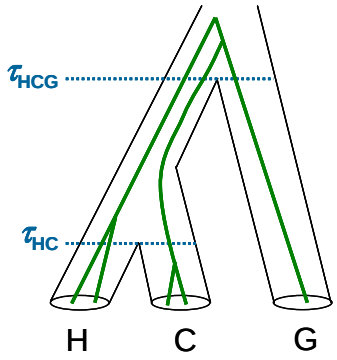
MCMCcoal:

Markov Chain Monte Carlo Coalescent Program

Version 1.2, March 2008



0. Introduction

The ANSI C program MCMCcoal implements the Bayesian Markov chain Monte Carlo (MCMC) algorithm of Rannala and Yang (2003) (RY03, see also Yang, 2002) for estimating species divergence times and population sizes from DNA sequence alignments at multiple loci. The population size parameter is defined as $\theta = 4N\mu$, the product of the effective population size N and the mutation rate μ (per nucleotide site per generation). Also θ is the average proportion of different sites in the sequence between two individuals drawn at random from the population. The species divergence time parameter is τ, which is the time of species divergence multiplied by the mutation rate μ. The analysis accommodates the species (population) tree and ancestral polymorphism (lineage sorting).

The model assumes no recombination within a locus, free recombination between loci, no migration (gene flow) between species, and neutral evolution. The JC69 mutation model (Jukes and Cantor, 1969) is assumed to accommodate multiple hits. The program is supposed to be used for analysis of closely-related species, so that more complex substitution models (such as models of variable mutation rates among sites within locus) are deemed unnecessary.

Consult Rannala and Yang (2003) for details of the model and theory.

A simulation program (MCcoal) can be compiled from the same source code, to simulate sequence data sets under the same model. See the section *The simulation program (MCcoal)* later in this document for details. The small program ds (for descriptive statistics) is included for summarizing MCMC results.

Citing the program

You may say something like "… analysis was conducted using the program MCMCcoal (Rannala and Yang, 2003)".

Rannala, B., and Z. Yang. 2003. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. *Genetics* 164:1645-1656.

1. Getting started

Trial run

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

  ```
  MCMCcoal
  ```

  The program reads the control file `MCMCcoal.ctl` by default. This analyzes the data set of Chen and Li (2001), in the file `ChenLi4s.txt` and duplicate the analysis of RY03 table 2; see table 1.

(Note: The output may be wider than 80 characters, so you should make your window wider. On Windows, you can do this by right-clicking the title bar, and changing Properties - Layout - Window size.)

To read a different control file, type something like the following

```
MCMCcoal MCMCcoalYu2001.ctl
```

- Print the control file `MCMCcoal.ctl` and read the rest of this document. Make sure you read the section on adjusting the finetune parameters.

Compiling the program

Win32 executables are included, so skip this section if you use Windows. 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 one of the following:

```
cc -o MCMCcoal -fast MCMCcoal.c tools.c -lm
cc -o MCMCcoal -O4 MCMCcoal.c tools.c
gcc -o MCMCcoal -O3 MCMCcoal.c tools.c -lm
cl -O2 -Ot MCMCcoal.c tools.c   (MS VC++ 6)
```

The `-o` flag specifies the name of the resulting executable file, `-O3` optimizes the code, and `-lm` flag links to the math library. Note that those compiler flags are case-sensitive. You may want or have to change some of the optimization flags (`-fast`, `-O4`, `-O3`, etc.) and the `-lm` flag is required on some systems but not on others.

2. A bit of theory

The basic theory and algorithm are described in Rannala and Yang (2003). The main purpose of this section is to provide some details of features added since publication of that paper. See also the list of new features in the History section.

Parameters in the model

If there are s species in the species tree, the model will involve the following parameters: $(s - 1)$ species divergence times (τs) and $(s - 1)$ ancestral θs. If any species has at least two sequences at any loci, a θ for that species will have to be considered in the model.

If the data are from one single-species, the only parameter will be θ for that species. In this case, there is no species tree. Thus the program can be used to estimate θ from a random sample of DNA sequences taken from one modern species under the standard coalescent model.

Variable mutation rates among loci

Model of fixed relative rates

Two approaches (models) are used to account for variable mutation rates among loci. (Note that the rate is assumed to be constant across sites in the same locus.) The first is called the model of fixed rates. You get estimates of mutation rates for the loci from external sources and collect them in a file. You then let MCMCcoal read those fixed rates from the file and use them in likelihood calculation in the MCMC algorithm.

This is a strategy I used in analysis of the ape data (Yang, 2002). The data included sequences from human, chimp, gorilla, and orang. First, the distance between orangutan and each of the HCG is calculated to give d_{HO}, d_{CO}, and d_{GO}, so that $d_i = (d_{HO} + d_{CO} + d_{GO})/3$ is the average distance from the orang to the human, chimp or gorilla at locus i. Then $r_i = d_i / \bar{d}$ is considered the relative rate for loci

i, where $\bar{d}$ is the average distance over all loci. These relative rates can be used in the analysis of the human-chimp-gorilla data. The orang is assumed to be sufficiently distant in the analysis of the H-C-G data. See the last two column of table 1 in Yang (2002). This example is included in the package. The relative rates are in the file ChenLidata3sMCMCcoal.rates. Run the program by

```
MCMCcoal MCMCcoalChenLi3sRates.ctl
```

You should scale the rates when preparing the locus-rate file, so that the average rate is one and parameters θs and τs are defined using this average mutation rate. If you scale the rates so that the first locus has rate one (by defining $r_i = d_i/d_1$ in the above example, say), parameters θs and τs will be defined using the mutation rate at the first locus. The MCMCcoal program does not scale the rates.

Model of random rate variation among loci

The second model is a model of random rate variation among loci. The average rate across all loci is fixed at one, and the rates for all loci are estimated from the data in the MCMC under this constraint. The prior on the locus rates is a transformed Dirichlet distribution, described in Burgess and Yang (2008). Simply every locus rate has the prior mean to be one, and there is a parameter α in the prior that specifies how variable the rates are assumed to be. When there are many loci, the variance of any locus rate is $1/\alpha$. The program calculates the posterior distribution of parameters θs and τs through the MCMC. In theory, this analysis will also lead to a posterior distribution for each locus rate. As there may be many loci in the data, this is not explicitly calculated and the program only calculates the posterior distribution of the rate at the first locus.

The variable-rates model implemented here has some differences from a similar model implemented in the IM program (Hey and Nielsen, 2004). The biggest difference appears to be the parametrization. MCMCcoal defines mutation rate on a per-nucleotide basis, so the prior specifies that the expectation of the mutation rate per site is constant among loci. IM defines mutation rate on a per-locus basis, so its prior specifies that the expectation of the mutation rate per locus is constant among loci. Suppose locus one has 500 sites and locus two has 5000 sites. Then the IM prior assumes that the per-nucleotide rate for locus one is 10 times the per-nucleotide rate at the second locus, while the MCMCcoal prior assumes that the per-nucleotide rates are equal. The two programs also differ in the constraint used to avoid over-parametrization. IM constrains the geometric mean of rates across loci to be one, while MCMCcoal constrains their arithmetic mean to be one. The priors on rates are also different.

Locus-specific heredity multiplier

When data from nuclear and mitochondrial genomes, or from autosomes and X chromosomes, are analyzed together, the effective population sizes are different among loci. Then a heredity multiplier (inheritance scalars, Hey and Nielsen, 2004) may be applied to each locus to reflect the effective different population sizes. MCMCcoal implements two options to deal with this issue. The first is for the user to specify the multiplier in the sequence data file, which will then be used as fixed in the MCMC algorithm.

Genome	Heredity scalar
Nuclear autosome	1
X chromosome	0.75
Y chromosome	0.25
Mitochondrial	0.25

The second option is to estimate the multipliers from the data, using a gamma prior.

(Implementations of those options were relatively simple, but I have not got time to test them carefully.)

Hey and Nielsen (2004) explained nicely that many factors such as natural selection might cause the apparent inheritance scalars to deviate from the expected values based on the mode of inheritance.

The effect of the locus rates and the locus-specific heredity multipliers are different. A locus-specific mutation rate is used to multiply all θs and τs for the locus. A heredity multiplier is used to multiply all θ parameters but not the τs. Because their effects are different (at least when multiple species are analyzed so that at least one τ parameter exists), they are identifiable parameters. Nevertheless, one may expect a strong correlation between them, especially when the species tree is small.

Model of random sequencing errors

Population genetics analysis can be sensitive to sequencing errors, as the natural polymorphism is typically low, and may be comparable to sequencing errors. A model is implemented to deal with random sequencing errors in certain species (Burgess and Yang, 2008). Suppose the model specifies that a certain species may have sequencing errors, at the rate of ε per nucleotide site. Then every terminal branch leading to that species in any gene tree is extended by length ε. The error rate ε is assigned a gamma prior, which applies to all species assumed to have errors. The program estimates a posterior distribution of the error rate for every species assumed to have errors, and of course the posterior differs between species.

One can also specify a fixed error rate for a species.

This model is motivated by analysis of some ape genomic data, in which certain species are understood to have substantial sequencing errors, due to smaller coverage compared with the human genomic sequences (Burgess and Yang, 2008). Nevertheless one should use the model with caution. Below are more notes to further clarify the model, which may serve as warnings.

- The model assumes *random* sequencing errors, meaning that the same species has a fixed per-nucleotide error rate in all loci. For example, if errors are expected to be clustered in a few loci (as they are on a chromosome of poor coverage, say), the model will be unrealistic.
- The model aims to detect and estimate sequencing errors in particular species by relying on the molecular clock (rate constancy across species or lineages). If the molecular clock is wrong, this may be a very unreliable exercise.
- You can't specify the model so that every species has sequencing errors. At least one species should have no errors.

Other programs

A brief comparison between MCMCcoal and other similar programs may be useful. IM, for isolation and migration (Hey and Nielsen, 2004), is a Bayesian MCMC program that deals with two species but allows for gene flow. MCMCcoal can work with one species or on a species tree of several species but assumes no gene flow. Both programs assume a species (population) phylogeny with species divergence times estimated as parameters in the model. For IM, the species tree is a tree of two species. MIGRATE (Beerli and Felsenstein, 1999; Beerli and Felsenstein, 2001) and GENETREE (Bahlo and Griffiths, 2000) are two maximum likelihood programs that deal with multiple species (populations), but assume an equilibrium migration model, which is equivalent to assuming that the populations diverged from each other an infinitely long time ago and have since been exchanging migrates. GENETREE also assumes the infinite-sites mutation model.

3. File formats

Example data sets

Two data sets are included in the package, to duplicate the results of RY03. The default control file `MCMCcoal.ctl` and the data file `ChenLi4s.txt` are for duplicating results in the column "Posterior (53 loci)" in table 2 of RY03. However, I have since changed the definition of speciation time

parameters, so the results generated by the program now are given in table 1. R&Y03 used $\tau_{HCGO} - \tau_{HCG}$, $\tau_{HCG} - \tau_{HC}$, and τ_{HC} as parameters but the program now uses τ_{HCGO}, τ_{HCG}, and τ_{HC} in both prior specification and output. In other words, the time gaps are replaced by node ages. The node ages are constrained as $\tau_{HCGO} > \tau_{HCG} > \tau_{HC}$. Thus the gamma priors for them are not independent as they may seem to be in the control file. Instead the joint distribution of τ_{HCGO}, τ_{HCG}, and τ_{HC} used by the program is a truncated version of the distribution.

You should compare your runs with results in table 1, which are close to those in table 2 in RY03. The priors used in the two analyses are not exactly the same.

TABLE 1 Prior and posterior distributions of parameters in the Bayesian analysis of the 53 loci of Chen and Li (2001, compare with RY03, table 2)

		Prior	Posterior (53 loci)
Parameter	(α, β)	Mean (95% interval)	Mean (95% interval)
θ_{HC}	(2, 2000)	0.001 (0.00012, 0.00279)	0.00188 (0.00054, 0.00383)
θ_{HCG}	(2, 2000)	0.001 (0.00012, 0.00279)	0.00348 (0.00219, 0.00499)
θ_{HCGO}	(2, 2000)	0.001 (0.00012, 0.00279)	0.00180 (0.00023, 0.00429)
τ_{HCGO}	(14, 1000)	0.01415 (0.00824, 0.02221)	0.01409 (0.01246, 0.01553)
τ_{HCG}	(19.8, 3000)	0.00690 (0.00463, 0.00981)	0.00595 (0.00519, 0.00679)
τ_{HC}	(20, 4000)	0.00474 (0.00300, 0.00673)	0.00484 (0.00403, 0.00559)

NOTE .— Both τ and θ are measured as the expected number of mutations per site.

The two files `MCMCcoalYu2001.ctl` and `yu2001.txt` can be used to duplicate the first row in table 3 of RY03. Run the program as follows.

```
MCMCcoal MCMCcoalYu2001.ctl
```

Control file

Analysis of multiple species data

```
ChenLi4s.txt
-1
4  H  C  G  O
   1  1  1  1
(((H, C), G), O);

    1                # use data? 0: prior; 1: posterior
5000   2  100000        # burnin output nsample

0.05 0.0018 0.3 0.0004 0.06 1.2 0.8 # finetune for GBtj, GBtip, theta, tau, mix, locusrate, seqerr

   2     2     2      14  19.8    20   # a_gamma
2000  2000  2000    1000  3000  4000   # b_gamma

1 4 4   # heredity  (0: No, 1: estimate, 2: from file)     & a_gamma b_gamma
1 2.0   # locusrate (0: No variation, 1: estimate, 2: from file) & a_Dirichlet (if 1)
0 0 0 1: 0.002 1 # sequencing errors: gamma(a, b) prior
```

The above is a copy of the control file `MCMCcoal.ctl`. Here are details of the control variables.

`ChenLi4s.txt`

Sequence data file name. You can look at the file, but don't print.

`-1`

Random number seed to determine the starting point of the MCMC run. 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. This is useful for debugging the program. If you use –1, the program will pick up a

seed at random based on the clock time, and different runs will use different random number seeds. If the calculation is healthy, those different runs should produce highly similar results. It is recommended that you run the same analysis twice using different random number seeds and confirm that the results are stable.

```
4  H  C  G  O
   1  1  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 specifies the maximum number of sequences among loci for each species. These numbers serve two purposes. First, they are used to determine which θ parameters are involved in the model and should be estimated. Second, they specify the maximum number of sequences for all species at a locus. The model always use a θ and a τ (age) for every interior node on the species tree. The model also uses a θ for each extant species if and only if that species has more than one sequence at some locus. If the example here, the parameters are the three ancestral θs and three node ages (τs).

```
(((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 of 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 iterations of the MCMC used as burnin. 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`. The size of this file is about $(8 \times p + 12) \times$ `nsample` bytes, where p is the number of θ and τ parameters. For the example data with $p = 6$ parameters, 60MB is needed for 10^6 samples.

```
0.05 0.01 0.005 0.005 0.5 1.2 0.8 # finetune for GBtj, GBtip, theta, tau, mix, locusrate, seqerr
```

These are fine-tuning variables for proposals in the MCMC. The first five were ε_1, ε_2, ε_3, ε_4, ε_5, described in the paper RY03, and are for proposals that (1) change internal node ages in the gene tree, (2) prune and re-graft nodes in the gene tree, (3) update θs, (4) update τs using the rubber-band algorithm, and (5) implements the mixing step. A few other steps were added later. The sixth finetune variable is for the step that updates locus-specific mutation rates or locus-specific heredity multipliers. The seventh is for a step that estimates sequencing errors in one or a few species. The model here assumes a species-specific proportion of sequencing errors that occur at random in sequences from that species. At least one species should be assumed to have no sequencing errors.

You should run the MCMC for a small number of generations and look at the screen output for the five acceptance proportions (see Section 4. *Running the program – Adjusting the finetune parameters*). The acceptance proportions should ideally be in the interval (0.15, 0.7) (see Yang, 2006, page 169). The results are not trustable if any of the proportions is near 0 or 1. If a proportion is too high (low), you increase (decrease) the corresponding fine-tuning parameter. Note if there is only one species in the tree, step 4 is not used and the value ε_4 has no effect.

```
   2     2     2      14  19.8    20   # a_gamma
2000  2000  2000    1000  3000  4000   # b_gamma
```

These two lines specify the values of α and β in the gamma priors for parameters θ's and τs in the model. The distribution has mean α/β and variance α/β^2, where α is called the shape parameter and β the scale parameter. If $\alpha \le 1$, 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, $c = 0$. 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.

The exact order of θs and τs is determined by the program. You should count the total number of parameters, and supply the right number of α and β values on those two lines. 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 parameters are in the order θ_{HC}, θ_{HCG}, θ_{HCGO}, τ_{HCGO}, τ_{HCG}, and τ_{HC}, as can be seen from the screen output:

```
3 theta parameters (populations) in the order:  7 (HC)  6 (HCG)  5 (HCGO)
3 species divergence times in the order:  5 (HCGO)  6 (HCG)  7 (HC)
Gamma prior: mean +- SE (95% CI) for theta's and tau's
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                  0.01400 +- 0.00374 (0.00765, 0.02223)
tau_HCG                   0.00660 +- 0.00148 (0.00402, 0.00981)
tau_HC                    0.00500 +- 0.00112 (0.00305, 0.00742)
```

```
1 4 4   # heredity  (0: No, 1: estimate, 2: from file)     & a_gamma b_gamma (if 1)
```

This line specifies the treatment of the locus-specific heredity multiplier. There are three options: 0 means no difference among loci (for example, all loci are autosomal); 1 means estimating the multipliers from the data with a gamma prior (the next two numbers are α and β of the gamma distribution); and 2 means reading the multipliers from the sequence data file.

```
1 2.0   # locusrate (0: No variation, 1: estimate, 2: from file) & a_Dirichlet (if 1)
```

This line concerns the locus-specific mutation rate. Three values (0, 1, 2) are possible for the first variable on the line: 0 means the same rate for all loci, 1 means the *random-rates model*, and 2 means that the fixed-rates model. The random-rates model assumes that the proportions of the rates among loci follow a Dirichlet distribution with parameter α (see the last page in this document), with $\alpha \to \infty$ meaning all loci having the same rate and a small α meaning highly variable rates. Thus with this option, you have to specify the parameter α for the prior. (Here $\alpha = 2.0$ in the example.) The fixed-rates model uses the relative mutation rates in an external file in the likelihood calculation. With this option, the line should look like the following, with the name of the rate file specified.

```
2 ChenLi3sMCMCcoal.rates      # locusrate
```

With two species on the tree, one should not try to estimate both locus mutation rates and heredity multipliers. It is probably a bad idea to try to estimate both even on large species trees.

```
0 0 1 0: 0.002 1   # sequencing errors: gamma(a, b) prior
```

This option specifies a model of random sequence errors. "0 0 1 0" means that the first species have no errors while the third (gorilla) has random sequencing errors, which occur as random variables from the prior gamma(0.002, 1), with the mean equal to 0.2%. The program will calculate a posterior distribution of the error rate for gorilla from the data.

Instead of the number 1 in the above line to mean that the error rate will be estimated from the data, you can specify a number such as 0.002, so that the line will be `"0 0 0.002 0: "` (say). This then means that the error rate in gorilla will be fixed at 0.2% in the MCMC calculation. This value must be between 0 and 1 and should be small.

Analysis of data from one species

```
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
```

The above is a copy of `MCMCcoalYu2001.ctl`, for analyzing a sample 61 human sequences of Yu et al. (2001) to estimate the single parameter $\theta = 4N\mu$ under the standard neutral coalescent. The mutation rate is assumed to be the same among loci. This should duplicate table 3 (first row) in RY03.

The control file is self-explanatory. There is no species tree as there is only one species. Multiple loci can be used in the data file, but one locus will work fine. The proposal step for changing τs is not used, so that the fourth acceptance ratio corresponding the fine-tuning parameter ε_4 is always 0. The program reads all of the five fine-tuning variables (ε_1, ε_2, ε_3, ε_4, ε_5) so they must all be there but ignores ε_4.

The printout on the monitor will include the posterior mean of θ, and posterior means of μt_{MRCA} for the loci, calculated up to that point in the MCMC run. If you have many loci, only the first few μt_{MRCA} are printed on the monitor.

The output file `mcmc.out` will list $g + 2$ columns for g loci: θ, μt_{MRCA} for the g loci, and the log likelihood. The μt_{MRCA} are not parameters, but are sometimes of interest as well.

Sequence data file format

The file `ChenLi4s.nuc` has the 53-loci data of Chen and Li (2001), which you can use to prepare your data files in the right format and to duplicate results in RY03 (see table 1). The data are in the phylip/paml format, with some more information at the beginning of each locus.

```
     53

     1  1  1  1
    4     483

   H1-2609             ------CTGA TTACCCAACA ATTCTCAGTT TCAGCCACTC AGAAGAGATC AAAACTAGTT TCCCTCACTC TCAATTCAAA ATTTGTGGAA
AAAAATATTA TTAGTTAAAC TTGTATCTAG TGACCACTCT TGTCCAACCA AGTATGACAT AGAGACTGGG TTAAACCAAC TGCTTGGTTT GAGGGTAATT CTTGGAAAAA
GAAGGCTATT GTAATGTTGA CAAAATAGCA ACAGCAGTCC ACTGCAACAT CACAACCAAC TTTACTGACA GAAGATCTTG TTCCTGCTAT AAAATTCATC TTTATCTTCT
GGACTTTAGT AAATCTGCAA AGGTATAGAA ACTATACCTT TAAAAATGTC CACAAAACAA TATGAAAAAT GGCAAGTCAA TCTCCAATGA TGAAATCGTG TATCAAAATC
AGGCTACATG ATTGTCACCC TGGCCATATA TGTAAAGAAA GTAGACAACA AATAAAAACT CAA
   Ch1-2609              ATGCGCATGA TTACCCAACA ATTCTCAGTT TCAGCCACTC AGAAGAGATC AAAACTAGTT TCCCTCACTC TCAATTCAAA ATTTGTGGAA
AAAAATATTA CTAGTTAAAC TTGTATCTAG TGACCACTCT TGTCCAACCA AGTATGACAT AGAGACTGGG TTAAACCAAC TGCTTGGTTT GAGGGTAATT GTTGGAAAAA
GAAGGCTATT GTAATGTTGA CAAAATAGCA ACAGCGGTCC ACTGCAACAT CACAACCAAC CTTACTGACA GAAGATCTTG TTCCTGCTAT AAAATTCATC TTTATCTTCT
GGACTTTAGT AAATCTGCAA AGGTATAGAA ACTATACCTT TAAAAATGTC CACGAAACAA TATGAAAAAT GGCAAGTCAA TCTCCAATGA TGAAATTGTG TATCAAAATC
AGGCTACATG ATTGTCACCC TGGCCATATA TGTAAAGAAA GTAGACAACA AATAAAAA-- ---
   G1-2609               ATGCGCATGA TTACCCAACA ATTCTCAGTT TCAGCCACTC AGAAGAGATC AAAACTAGTT TCCCTCACTC TCAATTCAAA ATTTGTGGAA
AAAAATATTA TTAGTTAAAC TTGTATCTAG TGACCACTCT TGTCCAACCA AGTATGACAT AGAGACTGGG TTAAACCAAC TGCTTGGTTT GAGGGTAATT CTTGGAAAAA
GAAGGCTATT GTAATGTTGA CAAAATAGCA AAAGCAGTCC ACTGCAACAG CACAACCAAC CTTACTGACA GAAGATCTTG TACCTGCTAT AAAATTCATC TTTATCTTCT
GGACTTTAGT AAATCTGCAA AGGTATAGAA ACTATACCTT TAAAAATGTC CACGAAACAA TATGAAAAAT GGCAAGTCAA TCTCCAATGA TGAAATCGTG TATCAAAATC
AGGCTACATG ATTGTCACCC TGGCCATATA TGTAAAGAAA GTAGACAACA AATAAAAACT CA-
   O1-2609               -----CATGA TTACCCAACA ATTCTCAGTT TCAGCCACTC AGAAGAGATC AAAACTAGTT TCCCTCACTC TCAATTCAAA ATTTGTGGAA
AAAAACATTA TTAGTTAAAC TTGTATCTAG TGACCACTCT TGTCCAACCA AGTATGACAC AGAGACTGGG TTAAACCAAC TGCTTGGTTT GGGAGTAATT CTTGGAAAAA
GAGGGCTATT GTAATGTTGA CAAAATACCA AAAGCAGTCC ACTGCAACAT CACAACCAAC CTTACTGACA GAAGATCTTG TTCCTGCTAT AAAATTCATC TTTATCTTCT
GGACTTTAGT AAATCTGCAA AGGTATAGAA ACTATACCTT TAAAAATATC CATGAAACAA TATGAAAAAT GGCAAGTCAA TCTCCAATGA TGAAATCGTG TATGAAAATC
AGGCTACGTG ATTGTCACCC TGGCCATATA TGTAAAGAAA GTAGACAACA AAT------- ???
   ...
```

The first number means 53 loci. This is then followed by 53 blocks for the 53 loci. Each block begins with the numbers of sequences for each species. In the example "`1 1 1 1`" means that the first locus has 1 human, 1 chimp, 1 gorilla, and 1 orang. The order of the species is fixed according to the control file `MCMCcoal.ctl` (the line "`4 H C G O`" in the example), in this case, H, C, G, and O. All 53 loci in this file has 1 sequence from each of the 4 species, but the program allows variable numbers of sequences at different loci; for example "`0 1 0 1`" would mean 1 chimp and 1 orang at the locus with human and gorilla missing. Every locus must have at least 2 sequences. The program ignores the sequence names in the alignment and relies solely on the numbers of sequences for the species specified here to attribute correctly which sequence is from which species.

If some loci are from the X or Y chromosomes while the others are autosomes, one should use an heredity multiplier for the locus (and also choose option 2 for the multiplier in the control file). This is specified using a pair of parentheses in the format

"`1 1 1 1 (0.75)`"

For this locus, all θ parameters are multiplied by 0.75. If the multiplier is 1, there is no need for specifying it, and simply use "`1 1 1 1`".

The rest of the block is the same as a standard paml/phylip sequence data file; see the PAML manual if you need more details. The locus has 4 sequences, each of 483 nucleotides. The order of sequences is fixed according to the order of species in the file `MCMCcoal.ctl`, and the sequence names in the alignment are read but ignored.

By default, alignment gaps in the sequence data are treated as ambiguity characters (question marks) and nucleotide sites in the alignment containing gaps or ambiguity characters (such as YRWN? etc.) are used in the likelihood calculation (see Felsenstein, 2004, p. 255; Yang, 2006, pp. 107-108 for details). In closely related species alignment gaps are rare. Furthermore gaps at the ends of the alignment are often undetermined nucleotides. Thus this way of treating gaps may be valid. To remove all ambiguity characters 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. The program does not allow you to remove ambiguity characters at some loci while keeping them at others. You will have to edit the data file yourself, perhaps with the help of BASEML in PAML.

4. Running the program

Pay attention to screen outputs

Make the window wider, with 100 or 120 columns before you run the program. (On Windows, you right click the window title bar and choose Properties – Layout and change Window Size Width.) Pay attention to screen outputs, especially at the start of the run, to make sure that the sequence data are correctly read by the program, that the specifications in the control file are correct, and that the acceptance proportions are reasonable. You can use Ctrl-C to stop the run if you need to change the prior or the fine-tuning parameters.

Here is the steps taken by the program. It first prints out the species tree, as well as a population-population table, which you can ignore. It then defines the θ and τ parameters, as explained before.

The program then reads and processes the sequence data file.

It then generates initial values for parameters θs and τs by using the gamma priors and sample the initial gene trees and coalescent times at loci by sampling from the prior. The program prints out the initial θs and τs, as well as the initial log likelihood lnL0, and then starts the MCMC. During the MCMC, it prints out after the percentage progress indicator the acceptance proportions for the proposals used in the MCMC as well as the posterior means of parameters in the model calculated using the samples taken to that point.

```
Starting MCMC... Initial parameters (gene trees generated from the prior):
   0.00096   0.00095   0.00103   0.01271   0.00689   0.00497
lnL0 =  -42932.575
  0%  0.37 0.39 0.32 0.37 0.36  0.00166 0.00362 0.00166  0.01421 0.00589 0.00489 -42843.32   0:07
  5%  0.37 0.41 0.31 0.37 0.37  0.00182 0.00350 0.00201  0.01397 0.00595 0.00486 -42837.38   0:43
```

```
10%  0.37 0.41 0.31 0.37 0.37  0.00180 0.00353 0.00190  0.01404 0.00592 0.00486 -42845.68   1:26
```

Here the percentage indicates the progress of the run, with a negative number meaning burnin. This is followed by five or seven numbers, which are the acceptance proportions for the proposal steps. You use them to adjust the fine-tuning parameters in the control file so that the acceptance proportions lie in the interval (0.15, 0.7). See the next section for details.

After the acceptance proportions are the posterior means of parameters. In the example above, the 6 parameters are θ_{HC}, θ_{HCG}, θ_{HCGO}, τ_{HCGO}, τ_{HCG}, τ_{HC}. The last number before the time used is the current log likelihood.

The program will then read and process the file `mcmc.out` to calculate the mean, min, max, median, and percentiles, and histogram information. This may take quite some time if many samples (say, 1M) are collected in the file. The included small program ds can also read the `mcmc.out` file to generate those summary statistics. (use `ds mcmc.out`).

The output is in the file named `out.txt`.

Adjusting finetune parameters

You have to adjust the finetune parameters in the control file to achieve reasonable acceptance proportions for all proposals used in the MCMC. Otherwise the results may be meaningless.

There is a line like the following in the control file (default `MCMCcoal.ctl`):

```
0.05  0.0018  0.3  0.0004  0.06 1.5 2.5  # finetune for GBtj, GBtip, theta, tau, mix, locusrat, seqerr
```

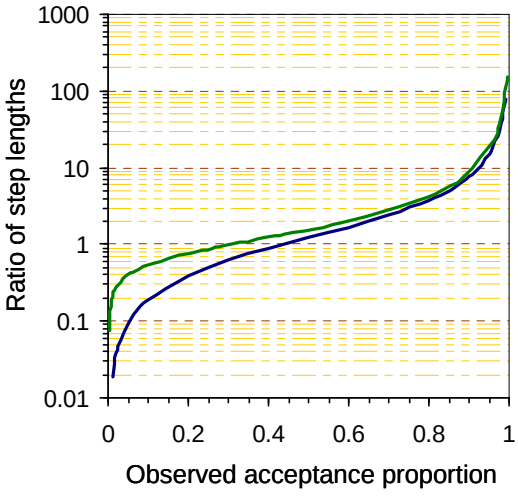
These are the step lengths used in the proposals in the MCMC algorithm. The program does not adjust them automatically and you have to adjust them yourself, by looking at the screen output, like the following.

```
0%  0.37 0.02 0.32 0.37 0.36  0.00166 0.00362 0.00166  0.01421 0.00589 0.00489 -42843.32   0:07
5%  0.37 0.01 0.31 0.37 0.37  0.00182 0.00350 0.00201  0.01397 0.00595 0.00486 -42837.38   0:43
```

There are seven finetune parameters in the control file. They are in fixed order and always read by the program even if the concerned proposal is not used. If the model assumes the same rate for all loci and does not use heredity multipliers, the six proposal step is not used. If the model assumes no sequencing errors, the last step is not used. The acceptance proportions for the first five proposals are always printed out o the screen, but those for the sixth and seventh are printed out only if the concerned proposal is used in the model. In the example above, only the first five proposals are used in the algorithm and only their acceptance proportions are printed out on the monitor.

The optimal acceptance proportions are around 0.3, and you should try to make them fall in the interval (0.15, 0.7). If the acceptance proportion is too small (say, <0.10)), you decrease the corresponding finetune parameter. If the acceptance proportion is too large (say, >0.80), you increase the corresponding finetune parameter. In the above example, the second acceptance proportion, at 0.01 or 0.02, is too small, so you should terminate the program (Ctrl-C) and modify the control file to decrease the corresponding finetune parameter (0.0018 above). Then run the program again (use the up ↑ and down ↓ arrow keys to retrieve past commands). Repeat this process a few times until every acceptance proportion is neither too small nor too large.

In case you are wondering how much you should change the finetune parameter, the graph on the right can be used as a rough guide. The *x*-axis shows the observed



acceptance proportion, and the *y*-axis shows the factor by which you should multiply the corresponding finetine parameter (step length). The green and blue curves are for two different types of proposals. For example, if the observed acceptance proportion is 0.2, you should multiply the finetune parameter by 0.4 (blue curve) or 0.8 (green curve).

Rather than trying to work out the details of the MCMC proposal and deciding which curve you should use, it is easy to monitor the acceptance proportion and adjust the finetune parameter accordingly.

One should not confuse the effects of the finetune parameters and of the priors. Simply, the prior affects the results of an analysis while the finetune parameters affect only how fast one can obtain the results. The finetune parameters affect the efficiency of the MCMC. If they are poorly chosen, it will take longer for the chain to converge and to mix. Mathematically the results should be correct and the same as long as one runs the chain for a very (infinitely) long time, but in practice the algorithm may be so inefficient that the results generated by the program may be totally unreliable. In contrast, changing the prior affects the results of the analysis.

Monitoring convergence

Slow convergence and poor mixing are two common problems with any MCMC algorithm. You may monitor the runs to make sure that the results are not obviously wrong. You can read any books on MCMC about diagnostic tools, but beware that none of them is fool-proof. See section 5.5 in Yang (2006) for example. Here are a few tips:

- Run the program at least twice using different random number seeds, and confirm that the results are stable between runs.
- Make sure that the acceptance proportions are neither too high nor too low.
- Plot the output in `mcmc.out` against 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.

5. Notes

The gamma prior

This program has mostly used the gamma distribution to specify the prior for parameters, partly because all parameters involved in the model are strictly positive. The gamma(α, β) distribution, with shape parameter α and scale parameter β, has mean $m = \alpha/\beta$ and variance $s^2 = \alpha/\beta^2$. It may be easier to think of mean m and standard deviation s in the prior and get $\alpha = (m/s)^2$ and $\beta = m/s^2$. Scaling (for example, due to mutation rate) changes β but not α. For example, suppose the time of divergence between humans and chimps is about 5MY, and we may represent that information by using $m = 5$ and $s = 1$ or $\alpha = (m/s)^2 = 25$ and $\beta = m/s^2 = 1$. If the mutation rate is 10^{-9} mutations per site per year, we expect 5MY of divergence corresponds to 0.005 mutations per site for sequence divergence. We adjust β to get the mean correct, giving $\beta = 5000$. The prior used for τ_{HC} in the program then has $\alpha = 25$ and $\beta = 5000$.

If not much information is available about the parameter, one may want to specify a diffuse prior. One can fix m first, and then adjust s or α, with a large s or small α meaning a less informative prior. In a normal distribution, the 95% interval is given roughly as $m \pm 2s$. This rule does not apply well to the gamma distribution if α is small (say, $\alpha < 1$), but may still be used as a guide. I suggest that $s = m$ ($\alpha = 1$) is very diffuse and $s = m/2$ ($\alpha = 4$) is quite diffuse. If you don't have much info and want to use diffuse priors for θ and τ, you may use $\alpha = 1.5$ or $\alpha = 2$, and then choose β to get the mean roughly right. Try to avoid very small α (such as 0.1 or 0.01), as they may cause numerical problems.

6. History

Version 1.2 July 2007

- Added two models of mutation rate variation among loci: one using fixed locus rates estimated externally, and another of random variable rates across loci (with a transformed Dirichlet prior).
- Added two options for locus-specific heredity multipliers (inheritance scalars). The first is for the user to specify the multipliers in the sequence data file, which will be used in the likelihood calculation in the MCMC. The second is to let the program estimate the multipliers, with a gamma prior (option 2 has not been tested.)
- The definition of species divergence times is changed. In version 1.1 and earlier, the divergence times may be defined as time gaps (say, $\tau_{HCGO} - \tau_{HCG}$, $\tau_{HCG} - \tau_{HC}$, and τ_{HC}) or as ages of nodes in the species tree (say, τ_{HCGO}, τ_{HCG}, τ_{HC}). This definition affects the prior specification and the output. In version 1.2, the parameters are always defined as node ages, in both prior specification and output.
- Added a model of random sequencing errors. This assumes that some species have sequencing errors, with a constant per-nucleotide error rate applied to all sequences for that species.

Version 1.1 April 2005
Version 1.0 September 2002

7. The simulation program (MCcoal)

A windows executable MCcoal.exe is included in the package. For unix systems, delete or comment out (using the pair /* and */ to bracket) the following line

```
#define MetropolisHastings       1
```

and compile the simulation program as follows

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

The program takes input from the control file `MCcoal.ctl`. An example is shown below. This is very similar to the control file `MCMCcoal.ctl` for MCMCcoal. The only difference is that the file name is for the file to be created, and the species tree is used to specify the parameters for the simulation. The prefix ':' in the tree specifies the node ages (parameters τ). This symbol is conventionally used to specify branch lengths, and the usage here is nonstandard. The prefix '#' specifies the population size parameters θs. So the tree in the following specifies the following parameters: $\theta_H = 0.001$, $\theta_C = 0.001$, $\theta_{HC} = 0.001$, $\theta_{HCG} = 0.001$, $\theta_{HCGO} = 0.001$, $\tau_{HCGO} = 0.014$, $\tau_{HCG} = 0.0066$, and $\tau_{HC} = 0.005$.

```
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;
```

The program then asks you for the number of loci and the sequence length. It then generates the sequences and also writes the true coalescent trees into the file `out.trees`. Delete the file if you don't need the trees.

8. References

Bahlo, M., and R. C. Griffiths. 2000. Inference from gene trees in a subdivided population. Theor. Popul. Biol. 57:79-95.

Beerli, P., and J. Felsenstein. 1999. Maximum-likelihood estimation of migration rates and effective population numbers in two populations using a coalescent approach. Genetics 152:763-73.

Beerli, P., and J. Felsenstein. 2001. Maximum likelihood estimation of a migration matrix and effective population sizes in *n* subpopulations by using a coalescent approach. Proc. Natl. Acad. Sci. U.S.A. 98:4563-4568.

Burgess, R., and Z. Yang. 2008. Estimation of hominoid ancestral population sizes under Bayesian coalescent models incorporating mutation rate variation and sequencing errors. Mol. Biol. Evol. 25:1979-1994.

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. Am. J. Hum. Genet. 68:444-456.

Felsenstein, J. 2004. *Inferring Phylogenies*. Sinauer Associates, Sunderland, Massachusetts.

Hey, J., and R. Nielsen. 2004. Multilocus methods for estimating population sizes, migration rates and divergence time, with applications to the divergence of *Drosophila pseudoobscura* and *D. persimilis*. Genetics 167:747-760.

Jukes, T. H., and C. R. Cantor. 1969. Evolution of protein molecules. Pages 21-123 *in* Mammalian Protein Metabolism (H. N. Munro, ed.) Academic Press, New York.

Rannala, B., and Z. Yang. 2003. Bayes estimation of species divergence times and ancestral population sizes using DNA sequences from multiple loci. Genetics 164:1645-1656.

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

Yang, Z. 2006. *Computational Molecular Evolution*. Oxford University Press, Oxford, England.
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. Mol. Biol. Evol. 18:214-222.