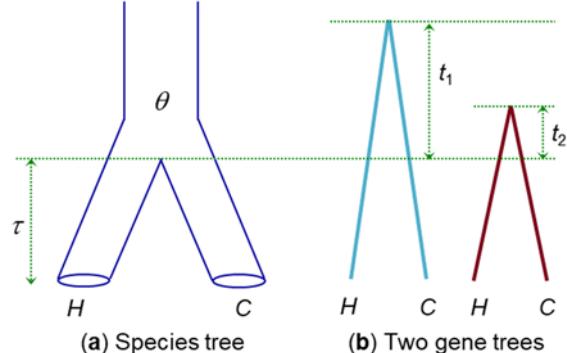


## MSC2s: MCMC implementation of the MSC model for two species

Ziheng Yang, 26 January 2021

This is a Bayesian MCMC program for fitting the multispecies coalescent (MSC) model for two species to the data from the human and the chimp. The dataset consists of sequences from the human and the chimpanzee at  $L$  loci. We use the JC mutation model so that the data can be summarized as the number of sites ( $n_i$ ) and the number of differences ( $x_i$ ) between the two sequences for each locus  $i$ , with  $i = 1, \dots, L$ . In other words, the data  $X = \{x_1, x_2, \dots, x_L\}$ . The MSC model has two parameters: the species divergence time ( $\tau = T\mu$ ) and the population size parameter  $\theta = 4N\mu$  for the H-C common ancestor. Here  $T$  is the divergence time in generations between the two species,  $N$  is the (effective) population size of the ancestral species, and  $\mu$  is the mutation rate per site per generation. Both  $\tau$  and  $\theta$  are measured in the number of mutations per site. As there is only one sequence from each species, population sizes for the two modern species are not estimable and are not parameters in the model.



Given the parameters ( $\tau$  and  $\theta$ ), the coalescent time  $t_i$  in the H-C ancestor at locus  $i$  is an exponential variable with mean  $\theta/2$ , with density

$$f(t_i | \tau, \theta) = \frac{2}{\theta} e^{-\frac{2}{\theta} t_i}, \quad i = 1, 2, \dots, L. \quad (1)$$

Note that one time unit (for both  $\tau$  and  $t_i$ ) is the amount of time taken to accumulate one mutation per site. The sequence distance at locus  $i$  (defined as the expected number of mutations per site) is  $d_i = 2(\tau + t_i)$ . Given  $t_i$  or  $d_i$ , the probability for a difference at any site at the locus is  $\frac{3}{4} - \frac{3}{4} e^{-\frac{4}{3}d_i}$ , given by the JC mutation model, and the probability of observing  $x_i$  differences at  $n_i$  sites at the locus is given by the binomial probability. Thus the likelihood (or the probability of data at all  $L$  loci) is

$$f(X | \tau, \theta, \{t_i\}) = \prod_{i=1}^L f(x_i | \tau, t_i) = \prod_{i=1}^L \binom{n_i}{x_i} \left( \frac{3}{4} - \frac{3}{4} e^{-\frac{8}{3}(\tau+t_i)} \right)^{x_i} \left( \frac{1}{4} + \frac{3}{4} e^{-\frac{8}{3}(\tau+t_i)} \right)^{n_i-x_i}. \quad (2)$$

Note the loci here are loosely-linked short genomic segments sampled from the genome. Each locus is short so that we ignore recombination between sites of the same locus (so that the  $n_i$  sites share the same  $t_i$ ), while different loci are far apart so that their histories ( $t_i$ ) are independent.

The data file [HC.SitesDiffs.txt](#) has  $n_i$  and  $x_i$  from the human and chimpanzee genomes at  $L = 14,663$  loci, from Burgess & Yang (2008 Mol Biol Evol 25:1979-1994, table 1 ‘Neutral’). For example the first locus has  $n_1 = 455$  sites with  $x_1 = 10$  differences. For your assignment, you can use either the first 1000 loci, or examine the impact of the data size.

Our objective is to estimate  $\tau$  and  $\theta$ . We assign an exponential prior with mean  $\mu_\tau = 0.005$  on  $\tau$  and another exponential prior with mean  $\mu_\theta = 0.001$  on  $\theta$ .

$$f(\tau) = \frac{1}{\mu_\tau} e^{-\frac{1}{\mu_\tau} \tau}, \quad f(\theta) = \frac{1}{\mu_\theta} e^{-\frac{1}{\mu_\theta} \theta}. \quad (3)$$

The unnormalised posterior (or the prior times the likelihood) is the product of equations 1, 2, and 3. We take the logarithm to get the log posterior as

$$\log f(\tau, \theta, \{t_i\} | X) = C - \frac{1}{\mu_\tau} \tau - \frac{1}{\mu_\theta} \theta + \sum_{i=1}^L \left[ \log \frac{2}{\theta} - \frac{2}{\theta} t_i + x_i \log \left( \frac{3}{4} - \frac{3}{4} e^{-\frac{8}{3}(\tau+t_i)} \right) + (n_i - x_i) \log \left( \frac{1}{4} + \frac{3}{4} e^{-\frac{8}{3}(\tau+t_i)} \right) \right], \quad (4)$$

where  $C$  is a constant. Note that the binomial coefficient from eq. 2 and  $\log \frac{1}{\mu_\tau \mu_\theta}$  from eq. 3 are absorbed into the constant  $C$ , which is then ignored in the MCMC algorithm.

The MCMC algorithm samples from the posterior specified by eq. 4. We use three sliding-window proposals with reflection to change  $\tau$ ,  $\theta$ , and  $t_i$ , with window sizes  $w_\tau$ ,  $w_\theta$ , and  $w_t$ . The algorithm may be as follows.

1. Initialise window sizes  $w_\tau = 0.0002$ ,  $w_\theta = 0.0005$ ,  $w_t = 0.02$  (say). Initialise parameters:  $\tau = 0.004$ ,  $\theta = 0.004$ , and  $t_i = 0.001$ , for  $i = 1, \dots, L$ .
2. Loop through the following steps.
  - a. Change  $\tau$ , using a sliding window of size  $w_\tau$ .  
Set  $\tau^* = \tau + (u - 0.5)w_\tau$  where  $u \sim U(0, 1)$ . If  $\tau^* < 0$ , set  $\tau^* = -\tau^*$ . Accept the proposed value with probability equal to the ratio of the unnormalized posterior. If accepted, set  $\tau = \tau^*$ .
  - b. Change  $\theta$ , using a sliding window of size  $w_\theta$ .
  - c. Loop through and change each of the  $L$  coalescent times ( $t_i$ ), using a sliding window of size  $w_t$ .
  - d. Save or print  $\tau$  and  $\theta$ .

### Notes about the algorithm

- i. Calculate the unnormalized posterior (prior and likelihood) on the log scale to avoid overflows and underflows.
- ii. Make obvious savings. For example, `log` and `exp` inside loops are expensive. Store the log unnormalized posterior for the current state  $(\tau, \theta, \{t_i\})$  in a variable (`lnpost`, say) to avoid repeated calculations.
- iii. Record the acceptance proportions for the three moves in the algorithm (2a, 2b, 2c). Try to adjust the window size so that the acceptance rate is close to 40% or within 20-80%. Note that we are using one window size  $w_t$  for the  $L$  coalescent times ( $t_i$ ).
- iv. *Bayesian marginalisation.* Note that the state of the Markov chain has  $(2 + L)$  dimensions. There are two parameters in the MSC model  $(\tau, \theta)$ , but the  $L$  coalescent times  $\{t_i\}$  are changing in the MCMC algorithm as well. Whether or not we print out  $t_i$ , the full sample has  $(2 + L)$  variables, and we summarize the samples for  $\tau$  and  $\theta$  only while ignoring the  $t_i$ . This way we generate the so-called marginal posterior distribution of  $\tau$  and  $\theta$ , with the  $t_i$  averaged over. This is called Bayesian marginalisation, and accommodates the uncertainties in the coalescent times.

## Correct results for L = 1000 loci

```
tau: 0.00401 (0.00374, 0.00430)
theta: 0.00399 (0.00339, 0.00457)
correlation is -0.76.
Window sizes are in the order of .001 .001 .02, for tau, theta and ti, giving
acceptance rates of 0.229 0.388 0.340.
```

## Correct results for L = 14663 loci

```
tau: 0.00417 (0.00409, 0.00425)
theta: 0.00438 (0.00421, 0.00454)
correlation is -0.75.
Window sizes are in the order of .0002 .0005 .02, for tau, theta and ti, giving
acceptance rates of 0.314 0.229 0.360.
```

## Running times, using L = 14663 loci

*Ziheng's PC, Lenovo ThinkCentre M920q, 8x2 cores (September 2019)*

```
nthreads = 1      50% 0.318 0.230 0.360  0.004170 0.004357  75056.723 -491196.376 4:30
nthreads = 2 (update_times always parallelised)
parallel_tau = 0: 50% 0.319 0.228 0.360  0.004170 0.004359  74906.964 -491155.548 4:16
parallel_tau = 1: 50% 0.319 0.230 0.360  0.004169 0.004359  75482.224 -491260.660 3:55

nthreads = 4 (update_times always parallelised)
parallel_tau = 0: 50% 0.322 0.230 0.360  0.004168 0.004362  75536.125 -491164.427 3:37
parallel_tau = 1: 50% 0.320 0.230 0.360  0.004168 0.004362  75242.309 -491161.954 3:00

nthreads = 8 (update_times always parallelised)
parallel_tau = 0: 50% 0.319 0.229 0.360  0.004170 0.004358  75027.297 -491167.238 3:44
parallel_tau = 1: 50% 0.319 0.229 0.360  0.004170 0.004358  75027.297 -491167.238 2:53

parallel_tau = 0: 50% 0.317 0.230 0.360  0.004169 0.004361  75317.208 -491234.136 3:40
parallel_tau = 1: 50% 0.317 0.230 0.360  0.004169 0.004361  75317.208 -491234.136 2:51

nthreads = 16 (update_times always parallelised)
parallel_tau = 0: 50% 0.321 0.227 0.360  0.004168 0.004362  75436.789 -491190.106 3:26
parallel_tau = 1: 50% 0.315 0.229 0.360  0.004170 0.004356  75494.294 -491239.376 2:32
```

*potto, 4 x CPUs, 144 cores/threads in total (threads may be on different CPUs)*

```
nthreads = 1      50% 0.317 0.230 0.360  0.004170 0.004357  75371.930 -491242.354 2:04
nthreads = 2 (update_times always parallelised)
parallel_tau = 0: 50% 0.319 0.230 0.360  0.004169 0.004358  75586.582 -491314.367 5:42
parallel_tau = 1: 50% 0.317 0.229 0.360  0.004168 0.004362  74967.362 -491152.087 4:26

nthreads = 4 (update_times always parallelised)
parallel_tau = 0: 50% 0.318 0.229 0.360  0.004167 0.004364  75051.328 -491094.642 3:50
parallel_tau = 1: 50% 0.318 0.230 0.360  0.004170 0.004357  75020.196 -491175.123 3:42

nthreads = 8 (update_times always parallelised)
parallel_tau = 0: 50% 0.319 0.231 0.360  0.004169 0.004359  74995.066 -491195.157 3:31
parallel_tau = 1: 50% 0.319 0.231 0.360  0.004169 0.004359  74995.066 -491195.157 3:09
```

*potto, threads pinned*

```
nthreads = 2
parallel_tau = 1: 50% 0.320 0.233 0.360  0.004169 0.004360  74959.965 -491098.645 1:34 taskset 0x03
nthreads = 4
parallel_tau = 1: 50% 0.317 0.229 0.360  0.004172 0.004353  75267.593 -491136.170 1:16 taskset 0x0f
nthreads = 8 (update_times always parallelised)
parallel_tau = 0: 50% 0.319 0.228 0.360  0.004169 0.004360  75209.647 -491135.632 2:24
parallel_tau = 1: 50% 0.319 0.228 0.360  0.004169 0.004360  75209.647 -491135.632 1:19
parallel_tau = 0: 50% 0.319 0.229 0.360  0.004169 0.004359  74990.799 -491150.952 1:41
parallel_tau = 1: 50% 0.319 0.231 0.360  0.004169 0.004359  75691.755 -491363.952 1:17
parallel_tau = 1: 50% 0.317 0.230 0.360  0.004168 0.004361  75362.033 -491138.097 1:14 taskset 0xff
1 + 8 threads: the 1 thread does theta move, and is not on the same CPU. Also someone is running 4 serial
jobs (which are moving around) and one 4-threads job (threads 108-111 on CPU2&3).
```