DistMCMC: Maximum Likelihood Estimation of Pairwise Distance Using Markov Chain Monte Carlo

Miao Lin

1101110335

Abstract

Estimation of the number of nucleotide substitution between DNA sequences is a fundamental question in the field of molecular evolution. The JC69 and K80 are the simplest nucleotide substitution models. Maximum likelihood estimation is a method of estimating parameters under a certain model. Markov chain Monte Carlo can be used to get the probability distribution of parameters by sampling from the equilibrium distribution of a Markov chain that has experienced enough generations. To learn nucleotide substitution models, maximum likelihood estimation, and Markov chain Monte Carlo, I write a Perl program called DistMCMC, which perform maximum likelihood estimation of distance between two DNA sequences using Markov chain Monte Carlo. For learning purpose, only JC69, JC69+Γ, K80 and K80+Γ are implemented.

Introduction

It has been a long time for me to learn molecular evolution, but it's always hard for me to understand such mathematic models and statistic concepts. I always feel confused when perform analysis using phylogenetic software, especially when there a lot of parameters need to be specified. I say I have waited for three years for the course of molecular evolution. During this summer, after having the course by Dr. Xia, a lot of confusedness has become clear in my mind. I think nucleotide substitution models, maximum likelihood estimation and

Markov chain Monte Carlo are three of them. So I write a Perl program called DistMCMC, which arrange the three concepts properly. I have asked a question in class. From JC69 to JC69+Γ or K80, both of the two have added only one parameter, but why JC69+Γ make the variance larger than K80. I run DistMCMC on real data, and compare the results on different nucleotide substitution models, and get the answer of the question I have asked in class.

JC69 is a nucleotide substitution model introduced by Jukes and Cantor in 1969, which assumed equal base frequency and equal substitution rate among different substitution types. Given two aligned DNA sequences, let d be the number of substitution per site, the probability of each aligned nucleotides being different is (Yang, 2006):

$$P_1 = 0.75 - 0.75\exp(-4d/3) \quad (1)$$

And the probability of each aligned nucleotides being same is:

$$P_2 = 0.25 + 0.75\exp(-4d/3) \quad (2)$$

If assume the distribution of substitution rate among nucleotides is gamma distribution, let α be the shape parameter, the probability of each aligned nucleotides being different is (Yang, 2006):

$$P_3 = 0.75 - 0.75(1 + 4d/3\alpha)^{(-\alpha)} \quad (3)$$

And the probability of each aligned nucleotides being same is:

$$P_4 = 0.25 + 0.75(1 + 4d/3\alpha)^{(-\alpha)} \quad (4)$$

K80 model is also called Kimura's two-parameter model, which assumed equal base frequency, and different substitution rate between transition and transversion. Given two aligned DNA sequences, let d be the number of substitution per site, κ be the transition and transversion rate ratio, the probability of each aligned nucleotides being different with transition is (Yang, 2006):

$$P_5 = 0.25 + 0.25\exp[-4d/(\kappa+2)] - 0.5\exp[-2d(\kappa+1)/(\kappa+2)] \quad (5)$$

And the probability of each aligned nucleotides being different with transversion is:

$$P_6 = 0.5 - 0.5\exp[-4d/(\kappa+2)] \quad (6)$$

And the probability of each aligned nucleotides being same is:

$$P_7 = 0.25 + 0.25\exp[-4d/(\kappa+2)] + 0.5\exp[-2d(\kappa+1)/(\kappa+2)] \quad (7)$$

If assume the distribution of substitution rate among nucleotides is gamma distribution, let α be the shape parameter, the probability of each aligned nucleotides being different with transition is (Yang, 2006):

$$P_8 = 0.25 + 0.25[1 + 4d/\alpha(\kappa+2)]^{(-\alpha)} - 0.5[1 + 2d(\kappa+1)/\alpha(\kappa+2)]^{(-\alpha)} \quad (8)$$

And the probability of each aligned nucleotides being different with transversion is:

$$P_9 = 0.5 - 0.5[1 + 4d/\alpha(\kappa+2)]^{(-\alpha)} \quad (9)$$

And the probability of each aligned nucleotides being same is:

$$P_{10} = 0.25 + 0.25[1 + 4d/\alpha(\kappa+2)]^{(-\alpha)} + 0.5[1 + 2d(\kappa+1)/\alpha(\kappa+2)]^{(-\alpha)} \quad (10)$$

Maximum likelihood estimation is a method of estimating parameters under a certain model. Equation (1) ~ (10) have defined models and provide basic elements of following likelihood functions. In addition to taking partial derivative of likelihood functions to get the maximum value, Markov chain Monte Carlo can be used to get the probability distribution of parameters by sampling from the equilibrium distribution of a Markov chain that has experienced enough generations.

In this paper, I will first build the likelihood function of each nucleotide substitution model, and then introduce the Markov chain Monte Carlo used to sample the probability distribution of parameters. Finally applied the program on real data, and try to find the difference of variance under different nucleotide substitution model.

Methods

Likelihood Function

Given two aligned DNA sequences, if the number of total nucleotide pairs is n, the number of segregated nucleotide pairs is x, in which y is transition, and z is transversion, the probability of observing this alignment, which is also being called likelihood, should be:

Under JC69 model:

$$L(x, n, d) = P(x, n \mid d) = P_1\text{^}x * P_2\text{^}(n\text{-}x) \quad (11)$$

Under JC69+Γ model:

$$L(x, n, d, \alpha) = P(x, n \mid d, \alpha) = P_3\text{^}x * P_4\text{^}(n\text{-}x) \quad (12)$$

Under K80 model:

$$L(y, z, n, d, \kappa) = P(y, z, n \mid d, \kappa) = P_5\text{^}y * P_6\text{^}z * P_7\text{^}(n\text{-}y\text{-}z) \quad (13)$$

Under K80+Γ model:

$$L(y, z, n, d, \kappa, \alpha) = P(y, z, n \mid d, \kappa, \alpha) = P_8\text{^}y * P_9\text{^}z * P_{10}\text{^}(n\text{-}y\text{-}z) \quad (14)$$

In these likelihood functions, $P_1 \sim P_{10}$ are from equation (1) ~ (10).

Markov Chain Monte Carlo

To build the Markov chain Monte Carlo, under the JC69 model, I specify a hard interval (a, b) of the only estimating parameter d, and generate a random number $r_1 \sim U(a, b)$, and specify r_1 as the initial state of d, which is assigned as d_0. Then specify a sliding window size of w (assign as step 1), and propose a new state by generating a random number $r_2 \sim U(d_0 - w/2, d_0 + w/2)$. If $r_2 < a$, new state of d, assigned as $d_1 = 2a - r_2$; if $r_2 > b$, assign $d_1 = 2b - r_2$; else, assign $d_1 = r_2$. Then calculate the acceptance ratio η with the following equation, in which the likelihood function is from equation (11):

$$\eta = \min[1, L(x, n, d_0)/L(x, n, d_1)] \quad (15)$$

Then generate another random number $r_3 \sim U(0, 1)$. If $\eta > r_3$, accept the new state d_1, which means assigning $d_0 = d_1$; else keep the old state d_0. Then go back to step 1 and repeat as many generations as enough. For each generation, there is a d saved. After burning in, the

remaining ones are used to calculate mean, median, and 95% highest probability density (95% HPD) interval. The mean or median represents the maximum likelihood estimate of d, and 95% HPD interval can represent the level of variance of d.

When JC69+Γ model is applied, one has to use equation (12) instead of (11) and specify the gamma shape parameter α. When K80 model is applied, another parameter κ is taken part into estimation. For each generation of Markov chain, in addition to sampling d, κ is also being sampled with same algorithm, and equation (13) is applied to calculate likelihood. When K80+Γ model is applied, in contrast to K80 model, just specify the gamma shape parameter α, and use equation (14) for likelihood calculating.

Apply DistMCMC on Real Data

Cytochrome B DNA sequences of 8 bear species or subspecies were used to test DistMCMC, including sloth bear (*Melursus ursinus*, Genbank: NC_009970), brown bear (*Ursus arctos*, Genbank: NC_003427), polar bear (*Ursus maritimus*, Genbank: NC_003428), sun bear (*Helarctos malayanus*, Genbank: NC_009968), American black bear (*Ursus americanus*, Genbank: NC_003426), Asiatic black bear (*Ursus thibetanus formosanus*, Genbank: NC_009331), Asiatic black bear (*Ursus thibetanus ussuricus*, Genbank: NC_011117) and Asiatic black bear (*Ursus thibetanus thibetanus*, Genbank: NC_011118). For each pair of DNA sequences, JC69, JC69+Γ, K80 and K80+Γ distance were calculated. The hard interval of parameter d was set to (0, 1), and sliding window size was set to 0.1. The hard interval of parameter κ was set to (0, 100), and sliding window size was set to 10. The length of Markov chain was set to 100000, and burn in was set to 10000. Save states of parameters every 100 generations, so for each parameter, there were 1000 samples, which were used to calculate the mean, median and 95% HPD interval. The shape parameter α was set to 0.2, 0.6 and 1.0 to run models with gamma rate.

Result

The mean or median of *d* output from DistMCMC is consisting with results output from PAUP* (Swofford, 2003) on any of the nucleotide substitution models. When gamma rate was arranged, as decreasing of shape parameter α, there was an obvious trend on increasing of estimated *d*, and there was also an obvious trend on increasing of 95% HPD interval of *d* (Table 1). These two trends were observed on both JC69 and K80 models.

Discussion

Because of limited time, I didn't perform hypothesis testing of my results. Actually, every conclusion talked in my results need statistic support, but I only test them by eye. The observed trends on both mean *d* and 95% HPD interval of *d* are consistent with Dr. Xia said in class and are also consistent with several text books (Nei, 2000; Yang, 2006).

Acknowledgement

I thank Dr. Lu's arrangement of the course, and thank Dr. Xia's wonderful lecture.

Reference

Nei, M. and Kumar, S. 2000. *Molecular Evolution and Phylogenetics*. Oxford University Press, New York.

Swofford, D. L. 2003. PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4. Sinauer Associates, Sunderland, Massachusetts.

Yang Z. 2006. *Computational Molecular Evolution*. Oxford University Press, New York.

Table 1. Distance estimated with DistMCMC. Result show only NC_003426 compared with other sequences.

Sequence Pair	Model	d_mean	d_median	d_95HPD_lower	d_95HPD_upper	95HPD_upper-lower
NC_003426 NC_003427	JC69	0.121779095	0.122335533	0.101403037	0.144491786	0.043088749
NC_003426 NC_003427	JC69+G a=1	0.131619389	0.131894491	0.104165372	0.155996301	0.051830929
NC_003426 NC_003427	JC69+G a=0.6	0.140525672	0.139418419	0.112867081	0.168057979	0.055190898
NC_003426 NC_003427	JC69+G a=0.2	0.18868148	0.188131457	0.140643147	0.235334916	0.094691769
NC_003426 NC_003427	K80	0.125906076	0.125544147	0.103037854	0.149574246	0.046536392
NC_003426 NC_003427	K80+G a=1	0.144636641	0.144028142	0.116761877	0.174235496	0.057473619
NC_003426 NC_003427	K80+G a=0.6	0.157696394	0.157247583	0.122647092	0.192800005	0.070152913
NC_003426 NC_003427	K80+G a=0.2	0.246946744	0.243486352	0.174767968	0.326332556	0.151564588
NC_003426 NC_003428	JC69	0.11466155	0.114413361	0.097512979	0.135950324	0.038437345
NC_003426 NC_003428	JC69+G a=1	0.124209907	0.123489075	0.099807591	0.149981623	0.050174032
NC_003426 NC_003428	JC69+G a=0.6	0.130956527	0.130636399	0.104193365	0.15510275	0.050909384
NC_003426 NC_003428	JC69+G a=0.2	0.173944338	0.173269248	0.130441876	0.215643119	0.085201244
NC_003426 NC_003428	K80	0.119346508	0.118380683	0.095614138	0.139590645	0.043976508
NC_003426 NC_003428	K80+G a=1	0.133816873	0.133565721	0.107981778	0.161143271	0.053161493
NC_003426 NC_003428	K80+G a=0.6	0.144882708	0.144323113	0.114188663	0.180463439	0.066274776
NC_003426 NC_003428	K80+G a=0.2	0.223725467	0.220238558	0.149650773	0.290675006	0.141024233
NC_003426 NC_009331	JC69	0.093693592	0.0931432	0.07541727	0.114349584	0.038932314
NC_003426 NC_009331	JC69+G a=1	0.099118716	0.097839957	0.0764107	0.118201847	0.041791146
NC_003426 NC_009331	JC69+G a=0.6	0.104547949	0.104267303	0.081513852	0.12959173	0.048077878
NC_003426 NC_009331	JC69+G a=0.2	0.130878637	0.129652762	0.099702721	0.171104904	0.071402183
NC_003426 NC_009331	K80	0.096316535	0.095555842	0.077497687	0.116533024	0.039035337
NC_003426 NC_009331	K80+G a=1	0.106750798	0.106793493	0.08283534	0.129789275	0.046953935
NC_003426 NC_009331	K80+G a=0.6	0.112645283	0.112083264	0.088267181	0.138746266	0.050479085
NC_003426 NC_009331	K80+G a=0.2	0.159061346	0.157207016	0.111143243	0.205033407	0.093890164
NC_003426 NC_009968	JC69	0.105126824	0.104279713	0.087620301	0.125548034	0.037927733
NC_003426 NC_009968	JC69+G a=1	0.113134053	0.11304212	0.092789789	0.135241627	0.042451838
NC_003426 NC_009968	JC69+G a=0.6	0.118711976	0.118414241	0.095994461	0.145037583	0.049043122
NC_003426 NC_009968	JC69+G a=0.2	0.153846068	0.152926728	0.116873875	0.194953768	0.078079893
NC_003426 NC_009968	K80	0.108874224	0.108621671	0.088417497	0.13081555	0.042398053
NC_003426 NC_009968	K80+G a=1	0.121505483	0.120748098	0.095999437	0.147050147	0.05105071
NC_003426 NC_009968	K80+G a=0.6	0.131221818	0.131131695	0.100592241	0.162739432	0.062147191
NC_003426 NC_009968	K80+G a=0.2	0.194420589	0.192012442	0.135841622	0.252852701	0.117011079
NC_003426 NC_009970	JC69	0.118007425	0.117695143	0.097602027	0.140453863	0.042851836
NC_003426 NC_009970	JC69+G a=1	0.128542532	0.128159728	0.103755807	0.152699061	0.048943254
NC_003426 NC_009970	JC69+G a=0.6	0.135479762	0.134711415	0.108487033	0.164891583	0.056404551
NC_003426 NC_009970	JC69+G a=0.2	0.180799177	0.179524809	0.139328385	0.22894759	0.089619205
NC_003426 NC_009970	K80	0.121983151	0.121678164	0.099397179	0.144623213	0.045226035
NC_003426 NC_009970	K80+G a=1	0.13835114	0.137531291	0.110335329	0.167316857	0.056981528
NC_003426 NC_009970	K80+G a=0.6	0.150893591	0.149758007	0.11817273	0.188331703	0.070158973
NC_003426 NC_009970	K80+G a=0.2	0.235351523	0.231202154	0.164062328	0.309389009	0.145326681
NC_003426 NC_011117	JC69	0.096476777	0.096216259	0.07961697	0.117154384	0.037537414
NC_003426 NC_011117	JC69+G a=1	0.102185617	0.10143042	0.082591691	0.125450286	0.042858594
NC_003426 NC_011117	JC69+G a=0.6	0.107309788	0.106831566	0.086481923	0.13028535	0.043803427
NC_003426 NC_011117	JC69+G a=0.2	0.137947257	0.136386182	0.102357663	0.174465277	0.072107614
NC_003426 NC_011117	K80	0.099964715	0.099589394	0.08062685	0.120513052	0.039886202
NC_003426 NC_011117	K80+G a=1	0.109496374	0.109157228	0.087125772	0.13305262	0.045926848
NC_003426 NC_011117	K80+G a=0.6	0.117181924	0.117293557	0.092420902	0.146522902	0.054102
NC_003426 NC_011117	K80+G a=0.2	0.168130694	0.165053967	0.119332825	0.221189479	0.101856654
NC_003426 NC_011118	JC69	0.100628687	0.100229396	0.080977265	0.119524015	0.03854675
NC_003426 NC_011118	JC69+G a=1	0.107832024	0.107402249	0.085100617	0.130870259	0.045769642
NC_003426 NC_011118	JC69+G a=0.6	0.112118393	0.111021845	0.089236343	0.138138474	0.04890213
NC_003426 NC_011118	JC69+G a=0.2	0.143333125	0.142133561	0.110517451	0.184295221	0.073777771
NC_003426 NC_011118	K80	0.10403723	0.103701677	0.084268062	0.125447325	0.041179264
NC_003426 NC_011118	K80+G a=1	0.115450674	0.114980417	0.091259429	0.140350629	0.0490912
NC_003426 NC_011118	K80+G a=0.6	0.124024021	0.123762792	0.095293713	0.152896586	0.057602872
NC_003426 NC_011118	K80+G a=0.2	0.181950913	0.180217637	0.125868117	0.232103101	0.106234984

Appendix 1. Perl Script of DistMCMC

```
#!/usr/bin/perl
use strict;
use warnings;

die("
Program: DistMCMC
DistMCMC does maximum likelihood estimation of pairwise distance using Markov chain Monte Carlo algorithm.
Only JC69, JC69+G, K80 and K80+G are available now.\n
Version: 1.0
Release: Aug. 3, 2014
Auther: Woody\n
Usage: $0 <in.fasta> <out.txt>
Please edit the Perl script directly if you want to make changes of parameters.
\n") if (@ARGV<2);

my $NoItera = 100000; # Number of iterations
my $Burnin = 10000; # Burn in
my $Savfreq = 100; # Frequency of saving states after burn in
my $Prifreq = 50000; # Frequency of printing states on screen
my $dSlidWin = 0.1; # Sliding window size of distance
my $kSlidWin = 10; # Sliding window size of kappa
my @dRange = (0, 1); # Range of distance
my @kRange = (0, 100); # Range of kappa
my $alpha = 0.5; # Gamma shape parameter

my $i = shift;
my $o = shift;
open my $i1, "<", "$i";
open O, ">", "$o";

my $nchar;
my %dna = %{&readfasta($i1)};
close $i1;

my %nseg; # Hash of pairwise number of segregation sites
foreach my $s1 (sort keys %dna) {
        foreach my $s2 (sort keys %dna) {
                if ($nseg{"$s1 $s2"} or $nseg{"$s2 $s1"}) {
                        next;
                } elsif ($s1 eq $s2) {
                        next;
                } else {
                        $nseg{"$s1 $s2"} = &calnseg($dna{$s1}, $dna{$s2});
                }
        }
}

print O "#Pair\tDistance\tDistance_mean\tDistance_median\tDistance_%95_HPD_lower\tDistance_%95_HPD_upper\t";
print O "Kappa_mean\tKappa_median\tKappa_%95_HPD_lower\tKappa_%95_HPD_upper\n";

foreach (sort keys %nseg) {
        print "\n$_";
        print "\nJC69 MCMC simulating...\nNo. Iteration\tDistance\n";
        my @a = &JC69MCMC(@{$nseg{$_}});
```

```
        print O $_, "\tJC69\t", join("\t", @a), "\n";
        print "\nJC69+G MCMC simulating...\nNo. Iteration\tDistance\n";
        @a = &JC69gammaMCMC(@{$nseg{$_}});
        print O $_, "\tJC69+G\t", join("\t", @a), "\n";
        print "\nK80 MCMC simulating...\nNo. Iteration\tDistance\tKappa\n";
        @a = &K80MCMC(@{$nseg{$_}});
        print O $_, "\tK80\t", join("\t", @a), "\n";
        print "\nK80+G MCMC simulating...\nNo. Iteration\tDistance\tKappa\n";
        @a = &K80gammaMCMC(@{$nseg{$_}});
        print O $_, "\tK80+G\t", join("\t", @a), "\n";
}
close O;

sub readfasta { # Read fasta file
        my $in = $_[0];
        my %fa;
        while (<$in>) {
                $/ = ">";
                my $seq = <$in>;
                chomp $seq;
                $seq =~ s/\s//g;
                $/ = "\n";
                $nchar = length $seq unless $nchar;
                s/>//;
                /^(\S+)\s/;
                $fa{$1} = $seq;
        }
        return \%fa;
}

sub calnseg {
        my ($s1, $s2) = @_;
        my $total;
        my $ti = 0;
        my $tv = 0;
        foreach my $i (0 .. $nchar-1) {
                next unless (substr($s1, $i, 1) =~ /[ATCG]/) and (substr($s2, $i, 1) =~ /[ATCG]/);
                ++$total;
                if (substr($s1, $i, 1) eq "A") {
                        if (substr($s2, $i, 1) eq "G") {
                                ++$ti;
                        } elsif (substr($s2, $i, 1) eq "A") {
                                next;
                        } else {
                                ++$tv;
                        }
                } elsif (substr($s1, $i, 1) eq "G") {
                        if (substr($s2, $i, 1) eq "G") {
                                next;
                        } elsif (substr($s2, $i, 1) eq "A") {
                                ++$ti;
                        } else {
                                ++$tv;
                        }
                } elsif (substr($s1, $i, 1) eq "C") {
                        if (substr($s2, $i, 1) eq "T") {
                                ++$ti;
                        } elsif (substr($s2, $i, 1) eq "C") {
                                next;
```

```
                        } else {
                                ++$tv;
                        }
                } elsif (substr($s1, $i, 1) eq "T") {
                        if (substr($s2, $i, 1) eq "T") {
                                next;
                        } elsif (substr($s2, $i, 1) eq "C") {
                                ++$ti;
                        } else {
                                ++$tv;
                        }
                }
        }
        return [$ti, $tv, $total];
}


sub NewSta { # Propose new state
        my ($OldStat, $SlidWin, $LoBound, $UpBound) = @_;
        my $r1 = rand;
        my $TemStat = $OldStat + ($r1-0.5)*$SlidWin; # Temp state
        my $NewStat; # New state
        if ($TemStat < $LoBound) {
                $NewStat = 2*$LoBound - $TemStat;
        } elsif ($TemStat > $UpBound) {
                $NewStat = 2*$UpBound - $TemStat;
        } else {
                $NewStat = $TemStat;
        }
        return $NewStat;
}

sub statis { # Calculate mean, median and 95% HPD of numbers
        my @d_sort = sort {$a <=> $b} @_;
        my $num_d = @_; # Number of samples in @_
        my $num_mid = int($num_d/2); # Middle number of @d_sort;
        my $d_mid = $d_sort[$num_mid];
        my $sum;
        foreach (@_) {
                $sum += $_;
        }
        my $d_mean = $sum/@_;
        my $num95_d = 0.95*$num_d; # %95 HPD number of samples
        my $intHPD95 = $d_sort[$num95_d-1] - $d_sort[0]; # %95 HPD interval
        my $intHPD95_l = $d_sort[0]; # Left boundary of %95 HPD interval
        my $intHPD95_r = $d_sort[$num95_d-1]; # Right boundary of %95 HPD interval
        for (my $i = 1; $num95_d+$i <= $num_d; ++$i) {
                my $int95 = $d_sort[$num95_d+$i-1] - $d_sort[$i]; # %95 interval
                if ($int95 < $intHPD95) {
                        $intHPD95 = $int95;
                        $intHPD95_l = $d_sort[$i];
                        $intHPD95_r = $d_sort[$num95_d+$i-1];
                }
        }
        return ($d_mean, $d_mid, $intHPD95_l, $intHPD95_r);
}

sub JC69MCMC {
        my ($ti, $tv, $n) = @_;
```

```
	my $d = ($dRange[1]-$dRange[0])*(rand) + $dRange[0]; # Random initial state
	my @d; # An array of saved states
	foreach (1 .. $NoItera) {
		my $d_old = $d; # Old state
		my $d_new = &NewSta($d_old, $dSlidWin, @dRange);
		my $logOld = ($ti+$tv)*log(0.75-0.75*exp(-4/3*$d_old)) + ($n-$ti-$tv)*log(0.25+0.75*exp(-4/3*$d_old)); # Log likelihood of old state
		my $logNew = ($ti+$tv)*log(0.75-0.75*exp(-4/3*$d_new)) + ($n-$ti-$tv)*log(0.25+0.75*exp(-4/3*$d_new)); # Log likelihood of new state
		my $accRat; # Acceptance ratio
		if ($logNew > $logOld) {
			$accRat = 1;
		} else {
			$accRat = exp($logNew-$logOld);
		}
		my $r2 = rand;
		if ($r2 < $accRat) {
			$d = $d_new;
		}
		my $pri = $_/$Prifreq;
		print "$_\t$d\n" if $pri == int($pri);
		if ($_ > $Burnin) {
			my $sav = $_/$Savfreq;
			push @d, $d if $sav == int($sav);
		}
	}
	my @d_stat = &statis(@d);
	return (@d_stat);
}

sub K80MCMC {
	my ($ti, $tv, $n) = @_;
	my $d = ($dRange[1]-$dRange[0])*(rand) + $dRange[0]; # Random initial state of distance
	my $k = ($kRange[1]-$kRange[0])*(rand) + $kRange[0]; # Random initial state of kappa
	my @d; # An array of saved states of distance
	my @k; # An array of saved states of kappa
	foreach (1 .. $NoItera) {
		my $d_old = $d; # Old state of distance
		my $d_new = &NewSta($d_old, $dSlidWin, @dRange);
		my $k_old = $k; # Old state of kappa
		my $k_new = &NewSta($k_old, $kSlidWin, @kRange);
		my $old1 = exp(-4*$d_old/($k_old+2));
		my $old2 = exp(-2*$d_old*($k_old+1)/($k_old+2));
		my $logOld = $ti*log(0.25+0.25*$old1-0.5*$old2)+$tv*log(0.5-0.5*$old1)+($n-$ti-$tv)*log(0.25+0.25*$old1+0.5*$old2); # Log likelihood of old state
		my $new1 = exp(-4*$d_new/($k_new+2));
		my $new2 = exp(-2*$d_new*($k_new+1)/($k_new+2));
		my $logNew = $ti*log(0.25+0.25*$new1-0.5*$new2)+$tv*log(0.5-0.5*$new1)+($n-$ti-$tv)*log(0.25+0.25*$new1+0.5*$new2); # Log likelihood of new state
		my $accRat; # Acceptance ratio
		if ($logNew > $logOld) {
			$accRat = 1;
		} else {
			$accRat = exp($logNew-$logOld);
		}
		my $r2 = rand;
		if ($r2 < $accRat) {
			$d = $d_new;
			$k = $k_new;
		}
		my $pri = $_/$Prifreq;
		print "$_\t$d\t$k\n" if $pri == int($pri);
		if ($_ > $Burnin) {
```

```
                        my $sav = $_/$Savfreq;
                        if ($sav == int($sav)) {
                                push @d, $d;
                                push @k, $k;
                        }
                }
        }
        my @d_stat = &statis(@d);
        my @k_stat = &statis(@k);
        return (@d_stat, @k_stat);
}

sub JC69gammaMCMC {
        my ($ti, $tv, $n) = @_;
        my $d = ($dRange[1]-$dRange[0])*(rand) + $dRange[0]; # Random initial state
        my @d; # An array of saved states
        foreach (1 .. $NoItera) {
                my $d_old = $d; # Old state
                my $d_new = &NewSta($d_old, $dSlidWin, @dRange);
                my $old1 = 1/(1+4*$d_old/3/$alpha)**$alpha;
                my $logOld = ($ti+$tv)*log(0.75-0.75*$old1) + ($n-$ti-$tv)*log(0.25+0.75*$old1); # Log likelihood of old state
                my $new1 = 1/(1+4*$d_new/3/$alpha)**$alpha;
                my $logNew = ($ti+$tv)*log(0.75-0.75*$new1) + ($n-$ti-$tv)*log(0.25+0.75*$new1); # Log likelihood of new state
                my $accRat; # Acceptance ratio
                if ($logNew > $logOld) {
                        $accRat = 1;
                } else {
                        $accRat = exp($logNew-$logOld);
                }
                my $r2 = rand;
                if ($r2 < $accRat) {
                        $d = $d_new;
                }
                my $pri = $_/$Prifreq;
                print "$_\t$d\n" if $pri == int($pri);
                if ($_ > $Burnin) {
                        my $sav = $_/$Savfreq;
                        push @d, $d if $sav == int($sav);
                }
        }
        my @d_stat = &statis(@d);
        return (@d_stat);
}

sub K80gammaMCMC {
        my ($ti, $tv, $n) = @_;
        my $d = ($dRange[1]-$dRange[0])*(rand) + $dRange[0]; # Random initial state of distance
        my $k = ($kRange[1]-$kRange[0])*(rand) + $kRange[0]; # Random initial state of kappa
        my @d; # An array of saved states of distance
        my @k; # An array of saved states of kappa
        foreach (1 .. $NoItera) {
                my $d_old = $d; # Old state of distance
                my $d_new = &NewSta($d_old, $dSlidWin, @dRange);
                my $k_old = $k; # Old state of kappa
                my $k_new = &NewSta($k_old, $kSlidWin, @kRange);
                my $old1 = 1/(1+4*$d_old/($k_old+2)/$alpha)**$alpha;
                my $old2 = 1/(1+2*$d_old*($k_old+1)/($k_old+2)/$alpha)**$alpha;
                my $logOld = $ti*log(0.25+0.25*$old1-0.5*$old2)+$tv*log(0.5-0.5*$old1)+($n-$ti-$tv)*log(0.25+0.25*$old1+0.5*$old2); # Log likelihood of old state
                my $new1 = 1/(1+4*$d_new/($k_new+2)/$alpha)**$alpha;
```

```
                my $new2 = 1/(1+2*$d_new*($k_new+1)/($k_new+2)/$alpha)**$alpha;
                my $logNew = $ti*log(0.25+0.25*$new1-0.5*$new2)+$tv*log(0.5-0.5*$new1)+($n-$ti-$tv)*log(0.25+0.25*$new1+0.5*$new2); # Log likelihood of new state
                my $accRat; # Acceptance ratio
                if ($logNew > $logOld) {
                        $accRat = 1;
                } else {
                        $accRat = exp($logNew-$logOld);
                }
                my $r2 = rand;
                if ($r2 < $accRat) {
                        $d = $d_new;
                        $k = $k_new;
                }
                my $pri = $_/$Prifreq;
                print "$_\t$d\t$k\n" if $pri == int($pri);
                if ($_ > $Burnin) {
                        my $sav = $_/$Savfreq;
                        if ($sav == int($sav)) {
                                push @d, $d;
                                push @k, $k;
                        }
                }
        }
        my @d_stat = &statis(@d);
        my @k_stat = &statis(@k);
        return (@d_stat, @k_stat);
}
```