Chapter 1

Introduction

Understanding phylogenetic relationships between organisms is a prerequisite for almost any evolutionary study, as all contemporary species share a common history through their ancestor. Today, phylogenetics relies on using mathematical methods to infer the past from features of contemporary species.

A phylogenetic tree (also called the evolutionary tree) is a directed graph showing the relationships between a group of contemporary taxa (leaves) and their hypothetical ancestors. The simplest tree consists of two taxa, where the leaves of the tree are labelled by bimolecular DNA sequences.

DNA is the basis of heredity; it consists of very long sequences of four bases (nucleotides): adenine (A), guanine (G), cytosine (C), and thymine (T). In human, the DNA consist of approximately 3.1 billion nucleotides; other species have more or less DNA, depending partly on the number of genes. These DNA sequences undergo changes within any population over the course of many generations, as random mutations arise and become fixed in the population. The construction of the phylogenetic trees is of interest in its own right in evolutionary studies. It is also useful in many other ways, for example in the prediction of gene function (Eisen, 1998).

Phylogenetic methods are based on assumptions (see e.g. Zharkikh, 1994; Swofford et al., 1996; Felsenstein, 2004a, 2004b). Evolutionary models describe the substitution process in sequences of nucleotides through time. A phylogenetic reconstruction method is statistically consistent if phylogenetic estimates converge towards the true tree as more data (longer sequences) are analysed. All phylogenetic methods make assumptions about the process of sequence evolution either implicity (in the case of parsimony methods) or explicitly (in the case of distance and probabilistic methods). In theory, phylogenetic methods are statistically consistent as long as their assumptions are met. However, every method is known to be inconsistent under some conditions (Felsenstein, 2004). When their assumptions are violated, current methods are prone to converge towards an incorrect solution (see e.g. Huelsenbeck, 1995; Swofford, 2001; Kolaczkowski, 2004; Ho and Jermiin, 2004; Jermiin, 2004).

We attempt to provide general methods to analyse this sort of data, which permits partial removal of the restriction of the stationarity and homogeneity assumptions. In particular, models are motivated by the bacterial data analyzed by Gaultier and Guoy (1995) who inferred a phylogeny among five eubacterial species using the small-subunit ribosomal RNA sequences from *Aquafix pyrophilus, Thermotoga maritima, Thermus thermophilus, Deinococcus radiodurans,* and a fifth species chosen from the following genera: *Chlamydia, Spirochaeta, Bacterides, Agrobacterium, Escherichia, Fusobacterium, Clostridium, Bacillus, Micrococcus,* and *Anabaena*. They used a nucleotide substitution model that assumes that $\pi_A = \pi_T$ and $\pi_C = \pi_G$ whereas $\pi_C + \pi_G$ was allowed to vary across the tree; hence, they used a non-stationary and non-homogeneous model to infer their eubacterial phylogeny. Examining the marginal distributions for this bacterial data we can see the evolutionary processes are not stationary. It is clear that the marginal distributions for *Aquifex, Thermus,* and *Thermotoga* are approximately equal as are the marginal distributions for *Bacillus* and *Deinococcus.* This suggested a possible model, where from the root of the tree we permit different Markov processes to operate along different descendant lineages.

Most currently available evolutionary models assume that the evolutionary processes at nucleotide sites are independent and identically distributed, so it is sufficient to describe the evolutionary process at a single site. This is done by specifying a continuous time finite Markov model depending on the time parameter $t \geq 0$. We start Chapter 2 by reviewing Markov models on phylogenetic trees. In the beginning we introduce a single continuous time finite Markov chain then extend it to K matched sequences. We develop programs to calculate the joint distribution of K matched sequences under the new model we suggested for the bacterial data.

In order to study the performance of phylogenetic methods to estimate phylogeny and evolutionary parameters, we need nucleotide sequences generated under controlled conditions. Monte Carlo simulations provide an opportunity to produce such data. Several computer programs have been developed to simulate evolution of nucleotide sequences on a known bifurcating tree (listed in Jermiin et al., 2003) but most of these have restricted attention to stationary and homogeneous cases and in general have not given detailed descriptions of their relation to the Markov processes of substitutions. In Chapter 3, we describe three methods to simulate the evolution of nucleotide sequences on a known bifurcating tree. In the first method, we calculate the joint probabilities and obtain the random numbers of sites containing each arrangement of matched nucleotides using the multinomial distribution; in the second method, we use the Markov process at each site (as in Rambaut and Grassly, 1997), or the equivalent embedded Markov chain and simulated waiting times, to obtain the simulated results at the ends of the tree; and in the third method, we describe a simulation method employed in the program called *Hetero* (Jermiin et al., 2003). We also give generalizations for these methods to be able to simulate K matched sequences.

Most phylogenetic methods assume that the sequences of nucleotides have evolved under stationary, reversible, and homogeneous conditions. When these assumptions are violated by the data, there is an increased probability of errors in the phylogenetic estimates. Methods to examine aligned sequences for violations of phylogenetic assumptions are available, but they are rarely used. In Chapter 4, we describe and compare the available tests for symmetry of K-dimensional contingency tables from homologous sequences, and develop two new tests to evaluate different aspects of the evolutionary processes. For any pair of sequences, we consider a partition of the test for symmetry into a test for marginal symmetry and a test for internal symmetry. The proposed tests can be used to identify appropriate models for estimation of evolutionary relationships under a Markovian model. Simulations under more or less complex evolutionary conditions were done to display the performance of the tests. Finally, the tests were applied to an alignment of small-subunit ribosomal RNA sequences of five species of bacteria to outline the evolutionary processes under which they evolved.

In Chapter 5, we review distance methods without giving comprehensive details. We define the additive and ultrametric distances. The paralinear distance is based on the general Markov model of evolution. It is an additive distance between sequences under general assumptions. We proceed by explaining how to generate trees from the distance matrix using the hierarchical clustering method, noting that it is equivalent to the UPGMA method of Sokal and Michener (1958). Finally we illustrate by examples the different situations where we can or cannot find the exact tree lengths and topology.

In Chapter 6, we discuss estimation using maximum likelihood. First we discuss the estimation method for a two leaf tree under stationary, homogeneous and reversible processes, then extend to K matched sequences allowing for non-stationary and non-homogeneous processes. Then we use the methods of Chapters 4 and 5 to choose the topology of the phylogenetic tree. Throughout the thesis we will restrict attention to cases where we permit non-heterogeneity for each side of the root. We describe the method of estimation under the general Markov model for this topology using the an optimization function, from the statistical package R, to maximize the log likelihood ratio. We proceed by giving different examples examining the accuracy of the method by generating joint distribution functions $F(t)$ with known parameters and from data sets simulated from these joint distributions. We obtain estimates from these data sets to examine the properties of the method of estimation. Then we apply the method of estimation to two sets of real data. Next we consider non-parametric and parametric bootstrap methods to obtain bias and standard deviation for the estimates. We apply these bootstrap methods to the simulated data set examples, for which we consider estimation, to examine the properties under conditions where the models hold. Then we use these bootstrap methods to give information about the estimates of the parameters for the real data sets.

We complete with a discussion, where we conclude that the methods of estimation work well for simulated data where the models are known to hold. However, this is not the case for the real data. Although we get improved estimates relative to the methods in current use, it is clear that more complex models will be required to give an adequate fit for these real data sets. For example, approaches allowing site heterogeneity, as in Pagel and Meade (2004), may be needed.

Note that throughout the thesis, we develop and use functions in S-Plus or R packages, which are provided in the appendices.

Chapter 2

Markov Process for Substitution Models

Phylogenetic methods are based on assumptions (see e.g. Zharkikh, 1994; Swofford et al., 1996; Felsenstein, 2004a, 2004b) and when these are violated by the data, there is an elevated risk of errors in the estimates of phylogeny (for a set of increasingly complex scenarios, see Ho and Jermiin, 2004). Given this risk, there is a need to understand better when phylogenetic methods are likely to produce reliable results and when they are likely to fail (see e.g. Jermiin et al., 2004).

Evolutionary models describe the substitution process in sequences of nucleotides through time. They are used to model this process along the edges of phylogenetic trees. In this chapter we will concentrate on the process operating at the level of nucleotides.

Most currently available evolutionary models assume that the evolutionary processes at nucleotide sites are independent and identically distributed, so it is sufficient to describe the evolutionary process at a single site. This is done by specifying a continuous time finite Markov model depending on the time parameter $t \geq 0$.

2.1. The Markov Process for One Nucleotide Site

In the beginning we will introduce a single continuous time finite Markov chain as follows. Consider a random variable X that takes values in a discrete space 1, 2, 3, 4 but whose values can change in continuous time. The value of X at time t is denoted by $X(t)$. The Markov process is based on the assumptions that all nucleotide sites change independently, all sites evolve by the same Markov process and the conditional probabilities of nucleotide substitutions do not change over time.

We will describe the substitution process $X(t),\ t \geq 0$, by the transition function

$$P_{ij}(t) = P[X(t) = j|X(0) = i], \tag{2.1}$$

where $P_{ij}(t)$ is the probability that the nucleotide labelled i changes to the nucleotide labelled j during the period of t.
We will represent $P_{ij}(t)$ in matrix notation as $P(t)$ and write it as

$$P(t) = \begin{bmatrix} P_{11}(t) & P_{12}(t) & P_{13}(t) & P_{14}(t) \\ P_{21}(t) & P_{22}(t) & R_{23}(t) & P_{24}(t) \\ P_{31}(t) & P_{32}(t) & P_{33}(t) & P_{34}(t) \\ P_{41}(t) & P_{42}(t) & P_{43}(t) & P_{44}(t) \end{bmatrix}.$$

Any row in this matrix corresponds to the state *from* which the transition is made, and any column in the matrix corresponds to the state *to* which the transition is made. Thus, the sum of probabilities in any particular row in the transition matrix must be 1.

For a homogeneous process having a finite number of states, the probabilities $P_{ij}(t)$ can be found by solving a matrix differential equation. To derive this equation we use the short-time approximation

$$P_{ij}(t) = R_{ij}t + o(t) \tag{2.2}$$

for $i \neq j$, where R_{ij} is the transition rate from state i to state j. Equation (2.2) implies the further short-time approximation

$$P_{ii}(t) = 1 - R_i t + o(t), \tag{2.3}$$

where $R_i = \sum_{i \neq j} R_{ij}$.

Now consider the Chapman-Kolmogorov relation (see for example Iosifescu, 1980)

$$P_{ij}(t+h) = P_{ij}(t)P_{jj}(h) + \sum_{k \neq j} P_{ik}(t)P_{kj}(h), \tag{2.4}$$

which says the process must pass through some intermediate state k at time t enroute to state j at time $t+h$. Substituting the approximations (2.2) and (2.3) in (2.4) yields

$$P_{ij}(t+h) = P_{ij}(t)(1 - R_j h) + \sum_{k \neq j} P_{ik}(t) R_{kj} h + o(h). \tag{2.5}$$

Sending h to 0 in the difference quotient

$$\frac{P_{ij}(t+h) - P_{ij}(t)}{h} = -P_{ij}(t)R_j + \sum_{k \neq j} P_{ik}(t) R_{kj} + \frac{o(h)}{h},$$

produces the forward differential equation

$$P^{'}_{ij}(t) = -P_{ij}(t)R_j + \sum_{k \neq j} P_{ik}(t) R_{kj} \tag{2.6}$$

The system of differential equation (2.6) can be summarized in matrix notation by introducing the matrix $P(t)$ and $R = (R_{ij})$. The forward equation in this notation becomes

$$P^{'}(t) = P(t)R \tag{2.7}$$

with

$$P(0) = I,$$

where $P^{'}(t) = (P^{'}_{ij}(t))$ and I is the identity matrix. The solution of the initial value problem (2.7) is furnished by the matrix exponential

$$P(t) = e^{Rt}, \tag{2.8}$$

where

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ R_{21} & R_{22} & R_{23} & R_{24} \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & R_{42} & R_{43} & R_{44} \end{bmatrix}$$

Here R is a time independent rate matrix satisfying

1. $R_{ij} \geq 0, i \neq j$
2. $R_{ii} = -\sum_{i \neq j}^{4} R_{ij} = -R_i$, so $R\mathbf{1} = \mathbf{0}$, where $\mathbf{1} = (1,1,1,1)^T$ and $\mathbf{0} = (0,0,0,0)^T$

3. $\pi^T R = \mathbf{0}^T$, where $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)^T$ is the stationary distribution.

Suppose that a Markov process has transition matrix $P(t)$ and at time t the probability that the process is in state j is equal to π_j, $j = 1,\ 2,\ 3,\ 4$. This implies that at time $u, u \geq t$, the probability that the process is in state j is equal to $\sum_{k=1}^{4} \pi_k P_{kj}(u-t)$. Suppose that for every j these two probabilities are equal, so that

$$\pi_j = \sum_{k=1}^{4} \pi_k P_{kj}(u-t),\ j = 1,\ 2,\ 3,\ 4.$$

Definition 2.1. *A vector $\pi^T = (\pi_1, \pi_2, \pi_3, \pi_4)$ with $0 \leq \pi_i \leq 1$, $i = 1, \cdots, 4$ and $\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$ is called a stationary probability distribution of a Markov process if $\pi^T R = (0, 0, 0, 0)^T$. This is equivalent to requiring that $\pi^T P(t) \equiv \pi$.*

In this case we say that the process is stationary with stationary distribution π. So if $P(X(t) = j) = \pi_j$, $j = 1, \cdots, 4$, for all t, then $X(.)$ is a stationary process. The matrix R can be reparameterised as:

$$R = \begin{bmatrix} R_{11} & \alpha_{12}\pi_2 & \alpha_{13}\pi_3 & \alpha_{14}\pi_4 \\ \alpha_{21}\pi_1 & R_{22} & \alpha_{23}\pi_3 & \alpha_{24}\pi_4 \\ \alpha_{31}\pi_1 & \alpha_{32}\pi_2 & R_{33} & \alpha_{34}\pi_4 \\ \alpha_{41}\pi_1 & \alpha_{42}\pi_2 & \alpha_{43}\pi_3 & R_{44} \end{bmatrix}, \tag{2.9}$$

where $R_{ii} = -\sum_{j \neq i} \alpha_{ij}\pi_j,\ i = 1, \cdots, 4$. We can regard the 12 parameters $\alpha_{12}, \alpha_{13}, \cdots, \alpha_{43}$ as free, in which case π is the left eigenvector of R corresponding to the eigenvalue 0.

Assuming that R is diagonalizable, then we can write R as

$$R = M \Lambda M^{-1}, \tag{2.10}$$

where M is a 4×4 matrix containing the right eigenvectors as columns and Λ is a 4×4 diagonal matrix containing the eigenvalues of R, which are thus functions of $\alpha_{12}, \alpha_{13}, \cdots, \alpha_{43}$. Therefore, the transition probability matrix (2.8) can be expressed as

$$\begin{aligned} P(t) &= e^{Rt} \\ &= \sum_{n=0}^{\infty} \frac{(Rt)^n}{n!} \\ &= \sum_{n=0}^{\infty} \frac{t^n}{n!} M\Lambda^n M^{-1} \\ &= Me^{\Lambda t}M^{-1} \\ &= \sum_{k=1}^{4} e^{\lambda_k t} u_k v_k^T, \end{aligned} \tag{2.11}$$

where $\Lambda = diag(\lambda_1, \cdots, \lambda_4)$, with $\lambda_1 = 0$, $M = (u_1, \cdots, u_4)$ and $M^{-1} = (v_1, \cdots, v_4)^T$, since

$$\begin{aligned} R^n &= M\Lambda M^{-1}M\Lambda M^{-1} \cdots M\Lambda M^{-1} \\ &= M\Lambda^n M^{-1}. \end{aligned}$$

Note that if f_0 is the vector of the initial probabilities, so $f_0^T = (f_{01}, \cdots, f_{04})$, then $P(X(t) = i) = (f_0^T P(t))_i$ and, in particular, if $f_0 = \pi$ then $P(X(t) = i) = (\pi^T P(t))_i = \pi_i$

2.1.1. The General Reversible Case. Some stochastic processes have the property that when the direction of time is reversed the behavior of the process remains the same. In this section we will assume that the model is reversible, where reversibility of the substitution process $X(.)$ intuitively means that the substitution viewed from now into the future is probabilistically identical to its behavior from now back into the past. This means that the direction of evolution does not affect the probabilities of different sites in two sequences. This property is described formally in the following definition.

Definition 2.2. *A stochastic process $X(t)$ is reversible if $(X(t_1), X(t_2), ..., X(t_n))$ has the same distribution as $(X(\tau - t_1), X(\tau - t_2), ..., X(\tau - t_n))$ for all $t_1, t_2, ..., t_n, \tau$.*

Note that reversible processes must be stationary.

Theorem 2.1. *A stationary Markov process is reversible if and only if there exists a collection of positive numbers π_j summing to unity that satisfy the balance equation*

$$\pi_i R_{ij} = \pi_j R_{ji},\ 1 \leq i, j \leq 4. \tag{2.12}$$

Proof: see Kelly (1979).

When such a condition exists, then π is the stationary distribution of the process, that is $\pi^T R = \mathbf{0}$.
Define S as a symmetric matrix such that

$$S = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} \\ S_{21} & S_{22} & S_{23} & S_{24} \\ S_{31} & S_{32} & S_{33} & S_{34} \\ S_{41} & S_{42} & S_{43} & S_{44} \end{bmatrix}, \tag{2.13}$$

where S satisfies

1. $S_{ij} \geq 0, i \neq j$
2. $S_{ii} = -\sum_{j \neq i}^{4} S_{ij}\pi_j/\pi_i$
3. $S\pi = \mathbf{0}$, where $\mathbf{0} = (0,0,0,0)^T$.

Let s be the vector of the off-diagonal elements in the S matrix, that is $s = (S_{12},\ S_{13},\ S_{14},\ S_{23},\ S_{24},\ S_{34})$. Now let

$$R = S\Pi \tag{2.14}$$

where $\Pi = diag(\pi)$, then R satisfies the balance equation listed above, and the Markov process is reversible. Further, we can express R as in (2.9) with $\alpha_{ij} = \alpha_{ji}$ and $S_{ij} = \alpha_{ij}$. Also

$$\Pi^{1/2} R \Pi^{-1/2} = \Pi^{1/2} S \Pi^{1/2}.$$

Let λ be an eigenvalue of R with right eigenvector v, then λ is also an eigenvalue of $\Pi^{1/2} R \Pi^{-1/2}$ with eigenvector $u = \Pi^{1/2} v$.

Returning to equation (2.10), when $R = M\Lambda M^{-1}$. Let $M = \Pi^{-1/2}U$, where U is an orthogonal matrix, that is $U^T = U^{-1}$ and Π is the diagonal matrix of π, so that

$$M^{-1} = U^{-1}\Pi^{1/2} = U^T\Pi^{1/2}.$$

We can rewrite R as

$$R = \Pi^{-1/2}U\Lambda U^T\Pi^{1/2} \tag{2.15}$$

so the transition probability matrix can be expressed as

$$\begin{aligned} P(t) &= \sum_{n=0}^{\infty}\frac{t^n}{n!}\Pi^{-1/2}U\Lambda^n U^T\Pi^{1/2} \\ &= \Pi^{-1/2}Ue^{\Lambda t}U^T\Pi^{1/2}, \end{aligned} \tag{2.16}$$

since

$$\begin{aligned} R^n &= \Pi^{-1/2}U\Lambda U^T\Pi^{1/2}\Pi^{-1/2}U\Lambda U^T\Pi^{1/2} \ ... \ \Pi^{-1/2}U\Lambda U^T\Pi^{1/2} \\ &= \Pi^{-1/2}U\Lambda^n U^T\Pi^{1/2}. \end{aligned}$$

In order to calculate the transition probability function $P(t)$, we prepared an S-Plus function, which depends on the following parameters: π, S and t.

2.2. The Markov Process for Paired Nucleotides Sites

We now present the theory for the case when we have two lineages. Consider two nucleotide sequences of length n evolving from a common ancestor by independent Markov processes at each site. That is, at each homologous site in the two sequences, we have two Markov processes $X(t)$ and $Y(t)$, $t \geq 0$, operating independently from the same common ancestor, for which $X(0) = Y(0)$.

The substitution process $X(t)$, $t \geq 0$, can be described by the transition function

$$\begin{aligned} P^X(t) &= (P^X_{ij}(t)) \\ &= (P[X(t) = j|X(0) = i]) \\ &= e^{R^X t}, \end{aligned} \tag{2.17}$$

with a corresponding function for $Y(.)$, that is

$$\begin{aligned} P^Y(t) &= (P^Y_{ij}(t)) \\ &= (P[Y(t) = j|Y(0) = i]) \\ &= e^{R^Y t}. \end{aligned} \tag{2.18}$$

Define

$$f_{ij}(t) = P[X(t) = i, Y(t) = j|X(0) = Y(0)], \tag{2.19}$$

where $f_{ij}(t)$ is the joint probability that two sequences have i and j, respectively, at a given homologous site, $i, j = 1, 2, 3, 4$.
Let the probability of the initial frequency for the bases be defined by the vector f_0 such that

$$f_0 = (f_{01}, f_{02}, f_{03}, f_{04})^T,$$

where f_{0k} is the probability of the k^{th} base, that is

$$f_{0k} = P(X(0) = Y(0) = k),$$

where $\sum_{k=1}^4 f_{0k} = 1$. Now since the two processes $X(t)$ and $Y(t)$ behave independently and identically, conditionally on $X(0) = Y(0)$, then

$$f_{ij}(t) = \sum_{k=1}^{4} f_{0k} P^X_{ik}(t) P^Y_{jk}(t) \tag{2.20}$$

Let $F(t) = (f_{ij}(t))$, and since $P(t) = (P_{ij}(t))$, then we can write $f_{ij}(t)$ in matrix notation as

$$F(t) = P^X(t)^T F(0) P^Y(t), \tag{2.21}$$

where

$$F(0) = diag(f_{01}, f_{02}, f_{03}, f_{04}).$$

The product $nF(t)$ is the expected divergence matrix between two sequences of length n and the theoretical equivalent of the observed divergence matrix.

2.2.1. Process for Paired Nucleotides Sites. We will restrict attention here to $R = S\Pi$, with S being a symmetric matrix. In the case of paired nucleotides sites, where the two processes $X(t)$ and $Y(t)$, coming from a common ancestor $X(0) = Y(0)$, are stationary, this will result in reversible processes. However, we will retain this symmetry of S when we consider non-stationary processes for simplicity.

The substitution process $X(t),\ t \geq 0$, can be described by the transition function

$$P^X(t) = \Pi_X^{-1/2} \sum_{k=1}^{4} e^{\lambda_{Xk} t} U_{Xk} U_{Xk}^T \Pi_X^{1/2} \tag{2.22}$$

with a corresponding function for $Y(.)$, that is

$$P^Y(t) = \Pi_Y^{-1/2} \sum_{k=1}^{4} e^{\lambda_{Yk} t} U_{Yk} U_{Yk}^T \Pi_Y^{1/2}, \tag{2.23}$$

where U_X and U_Y correspond to U of Section 2.1.1 for the processes X and Y. So we can write (2.21) as

$$\begin{aligned} F(t) &= P^X(t)^T F(0) P^Y(t) \\ &= \Pi_X^{1/2} U_X e^{\Lambda_X t} U_X^T \Pi_X^{-1/2} F(0) \Pi_Y^{-1/2} U_Y e^{\Lambda_Y t} U_Y^T \Pi_Y^{1/2} \\ &= \Pi_X^{1/2} (\sum_{i=1}^{4} \sum_{j=1}^{4} e^{(\lambda_{Xi} + \lambda_{Yj})t} U_{Xi} U_{Xi}^T \Pi_X^{-1/2} F(0) \Pi_Y^{-1/2} U_{Yj} U_{Yj}^T) \Pi_Y^{1/2} \\ &= \Pi_X^{1/2} (\sum_{i=1}^{4} \sum_{j=1}^{4} e^{(\lambda_{Xi} + \lambda_{Yj})t} U_{Xi} U_{Yj}^T a_{ij}) \Pi_Y^{1/2}, \end{aligned} \tag{2.24}$$

where

$$a_{ij} = U_{Xi}^T \Pi_X^{-1/2} F(0) \Pi_Y^{-1/2} U_{Yj}. \tag{2.25}$$

Note that equations (2.22) and (2.23) will hold for rate matrices and transition matrices even if the process is not stationary.

2.3. Stationary and Homogeneous Processes

Suppose that we have two processes $X(t)$ and $Y(t)$ with transition functions $P^X(t)$ and $P^Y(t)$, then we say that the two processes $X(t)$ and $Y(t)$ are homogeneous if they have the same transition matrices, that is if $P^X(t) = P^Y(t)$. We use homogeneous both for a time homogeneous process, where the rate is constant for all times, and, when it refers to more than one process, for equality of transition matrices throughout a tree. Now for the case of stationary and homogeneous processes, that is when $\Pi_X = \Pi_Y = F(0)$ and $P^X(t) = P^Y(t)$, $F(t)$ will be symmetric, see for example (2.27). $F(t)$ will also be symmetric if $P^X(t) = P^Y(t)$ even if $F(0) \neq \Pi_X = \Pi_Y$.
If we assume in, addition, that the processes are reversible, we can write $P(t)$ instead of $P^X(t)$ and $P^Y(t)$, Π instead of Π_X, Π_Y and $F(0)$, and U_i instead of U_{X_i} and U_{Y_i}. Now we can rewrite $F(t)$ in (2.24) as

$$\begin{aligned} F(t) &= \Pi^{1/2}(\sum_{i=1}^{4}\sum_{j=1}^{4} e^{2\lambda_i t} U_i U_j^T a_{ij})\Pi^{1/2} \\ &= \Pi^{1/2}\sum_{i=1}^{4} e^{2\lambda_i t} U_i U_i^T \Pi^{1/2} \\ &= F(t)^T, \end{aligned}$$

since

$$U_i^T U_j = 0, i \neq j,\ U_i^T U_i = 1,\ \lambda_1 = 0,$$

and

$$\begin{aligned} a_{ij} &= U_i^T \Pi^{-1/2} F(0) \Pi^{-1/2} U_j \\ &= U_i^T \Pi^{-1/2} \Pi \Pi^{-1/2} U_j \\ &= U_i^T U_j. \end{aligned}$$

So $a_{ij} = 0$ if $i \neq j$ and $a_{ii} = 1$ if $i = j$.

Note that if the sequences have evolved under a stationary, homogeneous and reversible process and $nF(t)$ is known, then it is possible

to determine uniquely the nine parameters in π and S (if we take t as some arbitrary value).
Knowing that $F(t)\mathbf{1} = \pi$, we can obtain $\Pi^{-1/2}F(t)\Pi^{-1/2}$, which allows us to get the eigenvalues and eigenvectors, that is

$$\Pi^{-1/2}F(t)\Pi^{-1/2} = \sum_{i=1}^{4} e^{2\lambda_i t} U_i U_i^T. \tag{2.26}$$

So for given t we can obtain

$$\Pi^{1/2}R\Pi^{-1/2} = \sum_{i=1}^{4} \lambda_i U_i U_i^T, \text{ and } S = R\Pi^{-1}.$$

In other cases, the parameters cannot be determined without some restrictions being imposed.

Example 2.1. The calculations presented here were obtained using functions in the S-Plus or R languages. Consider the case of stationarity, with $\Pi = F(0) = diag(0.25, 0.25, 0.25, 0.25)$, where

$$S_X = \begin{bmatrix} -0.6 & 0.2 & 0.2 & 0.2 \\ 0.2 & -0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & -0.6 & 0.2 \\ 0.2 & 0.2 & 0.2 & -0.6 \end{bmatrix}$$

and take $t = 0.5$. Then the transition probability, $P^X(t) = e^{R^X t}$, equals

$$P^X(t) = \begin{bmatrix} 0.9286 & 0.0237 & 0.0237 & 0.0237 \\ 0.0237 & 0.9286 & 0.0237 & 0.0237 \\ 0.0237 & 0.0237 & 0.9286 & 0.0237 \\ 0.0237 & 0.0237 & 0.0237 & 0.9286 \end{bmatrix}$$

Now, if we assume homogeneity ($i.e.P^X(t) = P^Y(t)$ and therefore implying $S_X = S_Y$), then we get a joint distribution function that equals

$$F(t) = \begin{bmatrix} 0.2160 & 0.0113 & 0.0113 & 0.0113 \\ 0.0113 & 0.2160 & 0.0113 & 0.0113 \\ 0.0113 & 0.0113 & 0.2160 & 0.0113 \\ 0.0113 & 0.0113 & 0.0113 & 0.2160 \end{bmatrix} \tag{2.27}$$

Notice that both $P^X(t)$ and $F(t)$ are symmetric matrices since the processes leading to them were symmetric and homogeneous. Finally, if the sequences contain 10000 nucleotides (*i.e.* $n = 10000$), then

$$nF(t) = \begin{bmatrix} 2160.12 & 113.29 & 113.29 & 113.29 \\ 113.29 & 2160.12 & 113.29 & 113.29 \\ 113.29 & 113.29 & 2160.12 & 113.29 \\ 113.29 & 113.29 & 113.29 & 2160.12 \end{bmatrix}$$

is the expected divergence matrix.

2.4. The Markov Process for K Matched Sequences

First, before we describe the K matched sequences, we need to define what we mean by the tree topology: "the evolutionary tree".

2.4.1. The Evolutionary Tree. The evolutionary tree is a directed graph showing the relationship between groups of taxa and their hypothetical common ancestor. The root of the tree is a common ancestor of all the taxa, the other nodes are either the contemporary taxa at the tips of the tree, called external nodes, or speciation events, called internal nodes, from which two new taxa bifurcate. The length of each edge represents the evolutionary time t. For any rooted tree of K matched sequences we will have $2K - 1$ nodes and $2K - 2$ edges.

Now we will represent the case when we have K matched nucleotide sequences having equal length n derived from a common ancestor $X(0)$, with a known tree topology. Each edge has length t_ℓ, where ℓ takes one of the values $\{1, \cdots, K-2\}$ for edges leading to an internal node and $\{-1, \cdots, -K\}$ for edges leading to a leaf, where we number the nodes with positive and negative integers for internal and leaf nodes, respectively. We take the edge length to be proportional to time, where the sum of the edge lengths of the edges from the root to any leaf is taken to be t. At each nucleotide site consider the Markov processes giving rise to $X_1(t), \cdots, X_K(t)$, the values of the processes at the leaf node. We can generalize the paired nucleotide case to this situation by noting that at time $t = 0$ we have $X_1(0) = \cdots = X_K(0)$. If the two edges of the tree starting at this ancestral node are of lengths t_A and t_B, and the taxa split into groups $X_1(t_A) = \cdots = X_m(t_A)$ and $X_{m+1}(t_B) = \cdots = X_K(t_B)$, then the joint probability of the processes at these nodes is

$$F(t_A, t_B) = P^A(t_A)^T F(0) P^B(t_B), \tag{2.28}$$

where P^A is the transition matrix of the Markov process operating along the lineage from the common ancestor to the node corresponding to A, P^B is the transition matrix of the Markov process operating along the lineage from the common ancestor to the node corresponding to B, and t_A and t_B are the edge lengths from the root node to the nodes corresponding to A and B, respectively, as we can see below in Figure 1

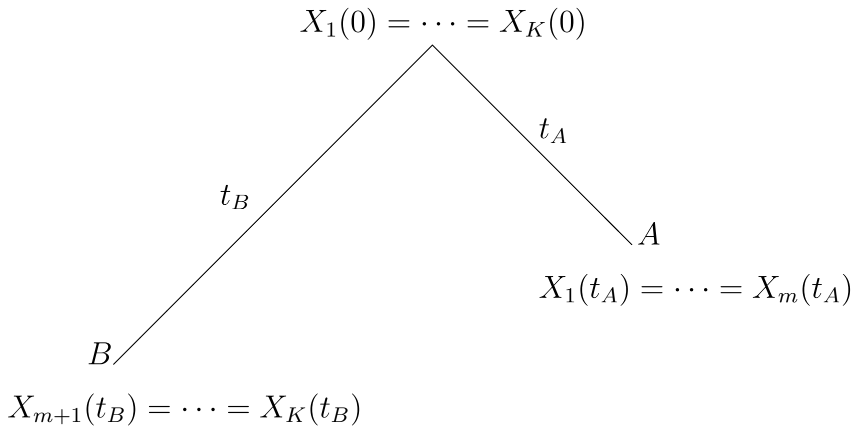

$X_1(0) = \cdots = X_K(0)$
t_A
t_B
A
$X_1(t_A) = \cdots = X_m(t_A)$
B
$X_{m+1}(t_B) = \cdots = X_K(t_B)$

Figure 1. Two Edge Tree

Now for the next step if the new nodes are not external nodes, we repeat the same process for the internal nodes, starting from the node nearest to the root, using in place of $F(0)$ in (2.28) the diagonal matrix of the conditional probabilities at the node corresponding to the nearest node A, given the values taken at the node corresponding to B. Multiplication by the marginal probabilities at the node corresponding to B gives a 4^3 array of joint probabilities of the nodes deriving from A and the node corresponding to B.

At any step, take for example the z-th step, before the process splits at this step, the subtree will have z tips and after this step the subtree will have $(z+1)$ tips.

For the z-th step, we start from the internal node nearest to the root, using in place of $F(0)$ the diagonal matrix of the conditional probabilities at the node corresponding to the z-th step, given the values taken at the nodes corresponding to all other tips. This replaces the conditional probability of this internal node given the value at the $z-1$ other tips by the conditional probability at the two tips deriving from this internal node conditional on the other $z-1$ tips. Multiplication by the joint distribution of the other $z-1$ tips gives a 4^{z+1} array for the joint distribution after the z-th step.

We repeat this process for each new internal node starting always from the internal node nearest to the root and dividing the group of that node into two groups until at the end all groups have just one member. Finally, we get $X_1(t), \cdots, X_K(t)$ at the leaves of the tree, as we can see in Figure 2.

This permits us to generate the entire 4^K array of probabilities

$$\begin{aligned} F_{i_1,\cdots,i_K}(t,\cdots,t) &= P(X_1(t)=i_1,\cdots,X_K(t)=i_K), \\ &\qquad i_j=1,\cdots,4,\ j=1,\cdots,K. \end{aligned}$$

In order to illustrate the way of finding the joint distribution function for a K edge tree we will provide more details in the next section.

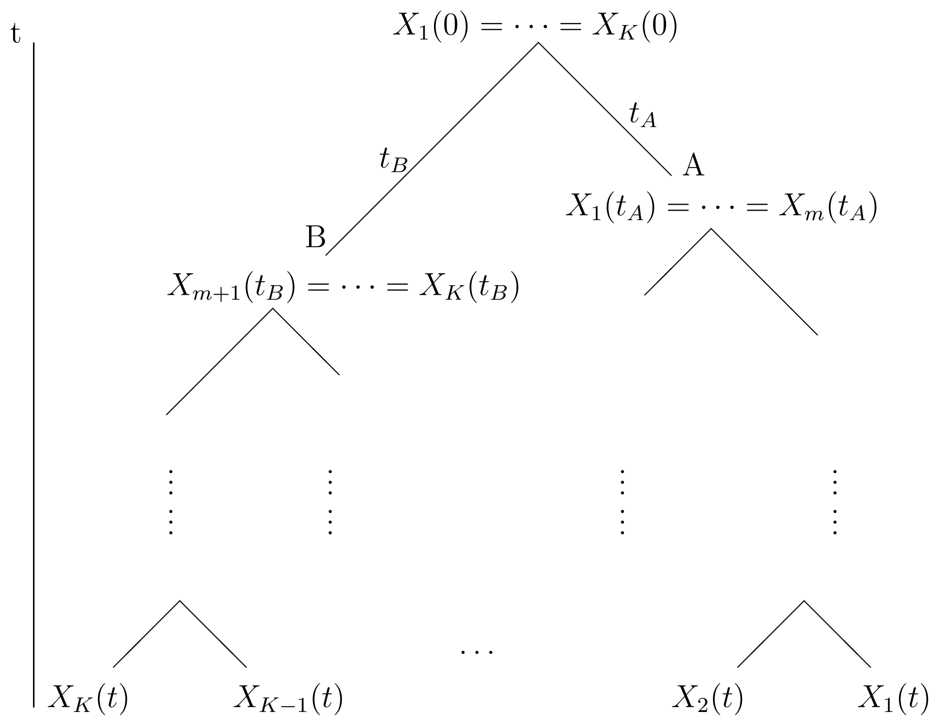

t
$X_1(0) = \cdots = X_K(0)$
t_A
t_B
A
$X_1(t_A) = \cdots = X_m(t_A)$
B
$X_{m+1}(t_B) = \cdots = X_K(t_B)$
$\cdots$
$X_K(t)$
$X_{K-1}(t)$
$X_2(t)$
$X_1(t)$

Figure 2. K-tipped tree

2.5. Three-tipped Phylogenetic Tree

For any tree we will divide the tree into two types of processes X and Y depending on the first bifurcation from the root. Suppose that we have three matched nucleotide sequences, derived from a common ancestor and a known tree. Take, for example, the tree shown in Figure 3. In this figure let X_r denote the nucleotide appearing at a particular site in species r, $r = 1,\ 2, 3$. Let t_1 be the divergence time from the internal node, to the tips 1 and 2, t be the divergence time from the root to any one of the three species (1, 2) and 3, which we will assume known and equal to a constant.

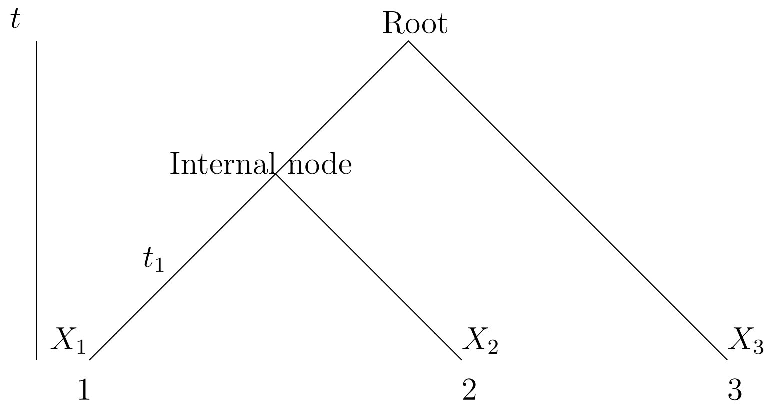


Figure 3. The three-tipped tree

Define

$$\begin{aligned} f_{ijk}(t) &= P[X_1(t) = i, X_2(t) = j, X_3(t) = k \mid X_1(0) = X_2(0) = X_3(0), \\ &\quad X_1(z) = X_2(z),\ 0 \le z \le t - t_1], \end{aligned}$$

where $f_{ijk}(t)$ is the probability that for a given site, the first, second and the third sequences have nucleotides i, j and k, where $i, j, k = 1, \cdots, 4$. Now by conditioning on the ancestral nucleotides at the root and the internal node in Figure 3, we have

$$F_{ijk}(t) = \sum_{l=1}^{4}\sum_{h=1}^{4} P_{li}^X(t_1)P_{lj}^X(t_1)P_{hl}^X(t-t_1)P_{hk}^Y(t)f_h(0),$$

where $f_h(0)$ is the probability of the nucleotides at the root, $i, j, l, h = 1, \cdots, 4$.

We can rewrite $F_{ijk}(t)$ in another way, that is at the beginning we define the probability function

$$\begin{aligned} F_{lk}(t-t_1,t) &= \sum_{h=1}^{4} P_{hl}^X(t-t_1)P_{hk}^Y(t)f_h(0) \\ &= \left[(P^X)^T(t-t_1)F(0)P(t)^Y\right]_{lk}, \end{aligned} \tag{2.29}$$

where $F_{lk}(t-t_1,t)$ is the probability that for a given site the first sequence has nucleotide l at the internal node after a period of time $t-t_1$ and the second sequence has nucleotide k after a period of time t. So in order to calculate the joint distribution function for the three species 1, 2 and 3 in Figure 3 we find the conditional probability for species 1 and 2 given $F_{lk}(t-t_1,t)$ at the internal node.
Note that

$$F_{lk}(t-t_1,t) = F_{l|k}(t-t_1|t)f_k(t),$$

where

$$f_k(t) = \sum_{l=1}^{4} F_{lk}(t-t_1,t).$$

So

$$\begin{aligned} F_{ijk}(t) &= \left[\sum_{l=1}^{4} P_{li}^X(t_1)P_{lj}^X(t_1)F_{l|k}(t-t_1|t)\right] f_k(t) \\ &= \left[(P^X)^T(t_1)D_kP^X(t_1)\right]_{ij} \\ &= F_{ijk}(t), \end{aligned} \tag{2.30}$$

where $D_k = diag(F_{1k}(t-t_1,t), \cdots, F_{4k}(t-t_1,t))$. From the joint distribution function $F_{ijk}(t)$, we can find the marginal distribution at each tip, as such the marginal distribution at node 3 is given by

$$F_{..k}(t) = \sum_{i=1}^{4}\sum_{j=1}^{4} F_{ijk}(t)$$

and the marginal distribution at node 2 is given by

$$F_{.j.}(t) = \sum_{i=1}^{4}\sum_{k=1}^{4} F_{ijk}(t)$$

finally the marginal distribution at node 1 is given by

$$F_{i..}(t) = \sum_{j=1}^{4}\sum_{k=1}^{4} F_{ijk}(t)$$

Define the joint distribution function for the two tips 1 and 2 as

$$F_{ij.}(t) = \sum_{k=1}^{4} F_{ijk}(t)$$

with a corresponding definition for all the other pairs.

Example 2.2. Consider the tree in Figure 3. Assume that the processes are stationary and homogeneous by taking

$$\Pi = \Pi_X = \Pi_Y = F(0) \text{ and } R^X = R^Y.$$

Let $\Pi = diag(0.25, 0.25, 0.25, 0.25)$. Define the S matrix as

$$S = \begin{bmatrix} -1.2 & 0.4 & 0.4 & 0.4 \\ 0.4 & -1.2 & 0.4 & 0.4 \\ 0.4 & 0.4 & -1.2 & 0.4 \\ 0.4 & 0.4 & 0.4 & -1.2 \end{bmatrix}, \tag{2.31}$$

and the s-vector $= (0.4, 0.4, 0.4, 0.4, 0.4, 0.4)$. Let $t = 1$ and $t_1 = 0.6$. Now using (2.30) we can calculate the 4^3 array, which represents the joint distribution function $F_{ijk}(1)$ for the whole tree. Depending on the joint distribution function $F_{ijk}(1)$, we can calculate the marginal distribution for all the tips, that is

$$F_{i..}(1) = F_{.i.}(1) = F_{..i}(1) = (0.25, 0.25, 0.25, 0.25).$$

We can also see that $F_{i.k}(1)$ is symmetric and equal to $F_{.ik}(1)$ but not equal to $F_{ik.}(1)$. We obtain exactly the same result of symmetry and equality of the tips marginal distributions, for the case of stationarity and non-homogeneity, such that $\Pi_X = \Pi_Y = F(0)$ and $R^X = \rho R^Y$, where $\rho > 1$. From this example we can see that, depending on the marginal distributions of the tips, we can not tell that the assumption of homogeneity is justified, even though all the tips have the same marginal distribution. Also we cannot tell that the stationarity assumption is justified or not, as we will see in the next example.

Example 2.3. Consider the same tree topology given in Example 2.2. Assume that the process is not stationary by taking

$\Pi_X = \Pi_Y = diag(0.2, 0.2, 0.2, 0.4)$ and $F(0) = diag(0.25, 0.25, 0.25, 0.25)$.

Assume also the process is homogeneous by taking $R^X = R^Y$. Let $t = 1$, $t_1 = 0.6$ and the s-vector be as in Example (2.2), where the S matrix can be calculated using (2.13). By finding $F_{ijk}(1)$ for the whole tree and the marginal distribution for the three edges, we can see that

$$F_{i..}(1) = F_{.i.}(1) = F_{..i}(1) = (0.2335, 0.2335, 0.2335, 0.2995).$$

Also we can see that $F_{i.k}(1)$ is symmetric and equal to $F_{.ik}(1)$ but not equal to $F_{ik.}(1)$.

In general, for any tree of K matched sequences, if all the marginal distributions are equal to each other, we cannot say that we have a homogeneous or stationary processes.

Example 2.4. Consider that we have the same tree topology and the same assumption of non-stationarity as the previous example, but in this example let

$$\Pi_X = diag(0.2, 0.2, 0.2, 0.4),$$

$$\Pi_Y = diag(0.3, 0.3, 0.3, 0.1)$$

and

$$F(0) = diag(0.25, 0.25, 0.25, 0.25).$$

Take $t = 1$, $t_1 = 0.6$ and the s-vector as in as in Example (2.2). Now since $\Pi_X \neq \Pi_Y$, this will result in a non-homogeneous process.

By finding the marginal distributions for each tip depending on the joint distribution $F_{ijk}(1)$, we can see that

$$F_{i..}(1) = F_{.i.}(1) = (0.2335, 0.2335, 0.2335, 0.2995)$$

and

$$F_{..i}(1) = (0.2665, 0.2665, 0.2665, 0.2005).$$

Also we can see that $F_{i.k}(1)$ is symmetric and equal to $F_{.ik}(1)$ but not equal to $F_{ik.}(1)$. These results show that, if the marginal distributions for a tree are not the same we can say that the process is not stationary, but we cannot tell any thing about the homogeneity assumptions, since we can get the same result of non equality for the marginal distribution at the tips, in the case of non-stationarity, where $\Pi_X = \Pi_Y \neq F(0)$ and homogeneous processes.

Example 2.5. Consider the same tree topology and the same assumption of non-stationarity by taking $\Pi_X, \Pi_Y, F(0)$, t, t_1 and the s-vector as in Example 2.3, but in this case let $R^X = \rho R^Y$, where $\rho = 3$. We can see that

$$F_{i..}(1) = F_{.j.}(1) = (0.2335, 0.2335, 0.2335, 0.2995)$$

and

$$F_{..k}(1) = (0.2151, 0.2151, 0.2151, 0.3548).$$

Also, depending on the joint distribution function, we can see that $F_{i.k}(1)$ is symmetric and equal to $F_{.jk}(1)$.
Knowing that we have the same edge length $t = 1$ from the root to any tips, and $\Pi_X = \Pi_Y = \Pi$, we can see that $F_{..k}(1)$ is going faster toward the stationary distribution $\pi = (0.2, 0.2, 0.2, 0.4)^T$ than both $F_{i..}(1)$ and $F_{.j.}(1)$. The reason for this is, the rate matrix for the edge corresponding to the site k is multiplied by a factor $\rho = 3$.

In the last two examples we have shown that for the case of non-stationarity and non-homogeneity, we cannot tell whether the non-homogeneity came from different rate matrices $R^X \neq R^Y$, or from different stationary probability for each side of the tree, $\Pi_X \neq \Pi_Y$, by just examining the marginal probability of the tips.

2.6. Programs

In this section we will describe the programs that we developed in order to calculate the joint distribution function for any rooted tree of K matched sequences. We used S-Plus or R Packages to write these programs.

In order to understand how the gn and gn2 programs work, we need to describe what we mean by the Merge Matrix and the vector of heights, which will be used in our programs.

The Merge Matrix. The merge matrix is a $(K-1) \times 2$ matrix, where K is the number of sequences (taxa) and therefore also the number of tips in the tree. The merge matrix contains negative and positive numbers. The positive numbers represent the rows of the merge matrix and also represent the internal nodes. The negative numbers represent the external nodes.

The merge matrix gives the order of bifurcation in the phylogenetic tree, that is the first row in the matrix represents the last two nodes in the tree splitting from an internal node, and so on for the other rows, till we reach the last row in the matrix, which represents the two nodes splitting from the root of the tree.

Example 2.6. We will give here an example to show how we can derive the tree topology from the merge matrix. Let the merge matrix be

$$merge = \begin{bmatrix} -1 & -2 \\ 1 & -3 \\ -4 & -5 \\ 3 & 2 \end{bmatrix}. \tag{2.32}$$

From the dimension of the merge matrix 4×2, we know that we have a five taxa tree. Also, we can see that we have five terminal edges since in the merge matrix the smallest negative number is -5, which represents the fifth edge, see Figure 4(D).

At the beginning we start from the last row in the merge matrix, that is the root of the tree which splits into two new nodes 3 and 2, as we can see in Figure 4(A). Since both of the new nodes are positive, implying that we have two internal nodes, the process will take the largest number from them, number 3, and go to the row in the merge matrix corresponding to this number, then the process will split node 3 into two new nodes -4 and -5 see Figure 4(B). As we can see the last two nodes are negatives, which means that the we have external nodes and the process will stop in this direction. The process then will take the next largest number in row 4, which is number two, then node 2 will split into two new nodes 1 and -3 given in row 2, as we can see in Figure 4(C), then the process will take node 1 since it is the largest positive number we still have to split this node into two new nodes -1 and -2 as we can see in Figure 4(D). Now since all the nodes are negative the process will stop.

The Vector of Heights. The vector of heights is a vector of size $(K-1)$ giving the lengths from all internal nodes (including the root) to any leaf, where length is the sum of the edge lengths for edges connecting the internal node to its descendent leaves.

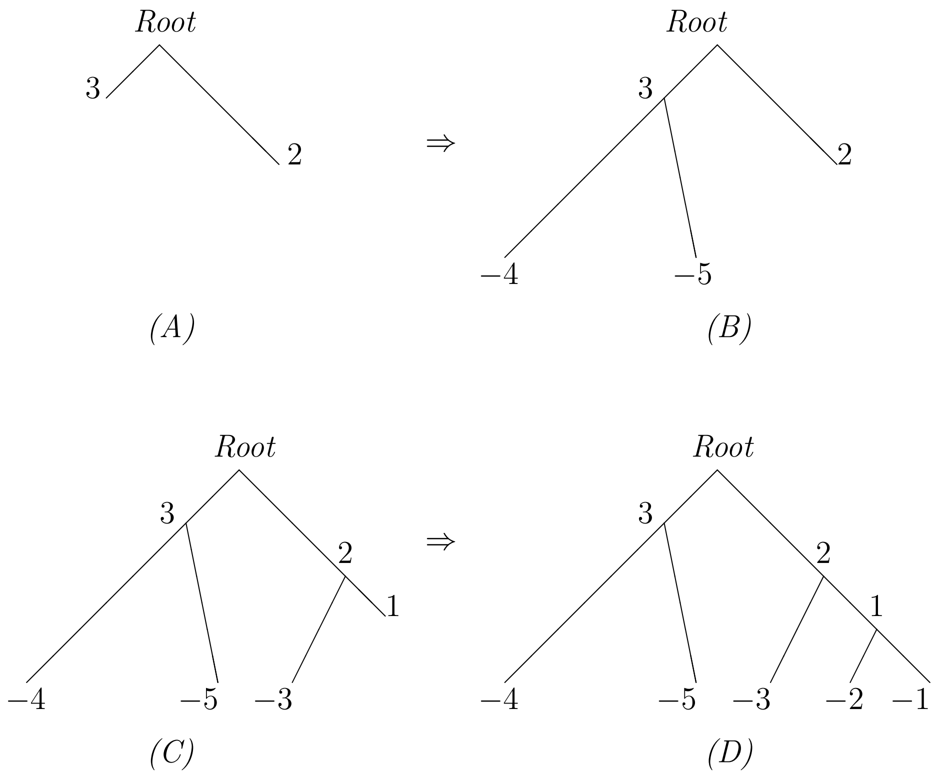


Figure 4. Rooted five tipped tree

2.6.1. The gn Program. We will show here how the gn program we developed works, in order to calculate the joint distribution function

$$\begin{aligned} F_{i_1,\cdots,i_K}(t,\cdots,t) &= P(X_1(t)=i_1,\cdots,X_K(t)=i_K), \\ & i_j=1,\cdots,4,\ j=1,\cdots,K. \end{aligned} \tag{2.33}$$

for any tree of K matched sequences. The gn program uses a tree description based on a matrix describing the tree topology. An example of this matrix is given in (2.32) with a corresponding 5 taxa tree given in Figure 4(D). The gn program works on a tree where the length from the root to any external node is fixed.

The gn program depends on the following parameters $\pi_X,\ \pi_Y,\ f_0$, ρ, the s-vector and a vector of heights depending on the size of the tree. The program recursively uses (2.28) at each internal node, commencing with the root node and proceeding from the bottom of the merge matrix using edge lengths calculated inside the program from the differences in the lengths between the nodes at each step and the next two nodes which bifurcate from that node, where the lengths of the external nodes will be given (as a vector of heights). It is possible to allow different transition matrices on each edge, but this would be impractical for large values of K. For any tree, our program will divide the tree into two parts depending on the first bifurcation at the root. That is the program restricts attention to two transition matrices $P^X(t)$ and $P^Y(t)$, with properties of reversible processes; we use rate matrices $R^X = S_X\Pi_X$ and $R_Y = S_Y\Pi_Y$, where S_X and S_Y are symmetric matrices depending on Π_X, Π_Y and the s-vector. We use transitions based on $P^X(t)$ for all edges associated with the node given by the first element of the last row of the matrix describing the topology (those on the right of the root node in the Figure 4(D)) and $P^Y(t)$ is used for the edges on the other side of the tree. Further, if $\pi_X \neq \pi_Y$, the initial distribution at the root node can differ from these. In this case the processes are not reversible, since they are not stationary, but we restrict attention to transition matrices of the form just described.

2.6.2. The gn2 Program. We will describe here a generalization of the gn program developed in order to calculate the joint distribution function (2.33) for any tree of K matched sequences. The gn2 program allows us to have different rates for each edge, which allows us to have different lengths from the root to the external nodes, the rates here represent the lengths of edges. Note that we can not distinguish between the time and the length of the edge.

This program can be applied to any type of bifurcating tree, it requires the set of parameters $\pi_X,\ \pi_Y,\ f_0$, the s-vector, a tree description based on a matrix describing the tree and a matrix of instantaneous rates for each edge. The following two matrices correspond to those necessary in order to generate sequences on a rooted, 5-leafed tree in Figure 5. The first matrix describes the tree with the nodes labelled and the second matrix gives the rates or lengths of the eight edges in the tree.

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ 2 & -3 \\ 3 & 1 \end{bmatrix}, \ rate = \begin{bmatrix} .2 & .2 \\ .5 & .5 \\ .3 & .8 \\ .2 & 1.8 \end{bmatrix} \equiv \begin{bmatrix} \rho_{-1} & \rho_{-2} \\ \rho_{-4} & \rho_{-5} \\ \rho_{2} & \rho_{-3} \\ \rho_{3} & \rho_{1} \end{bmatrix}$$

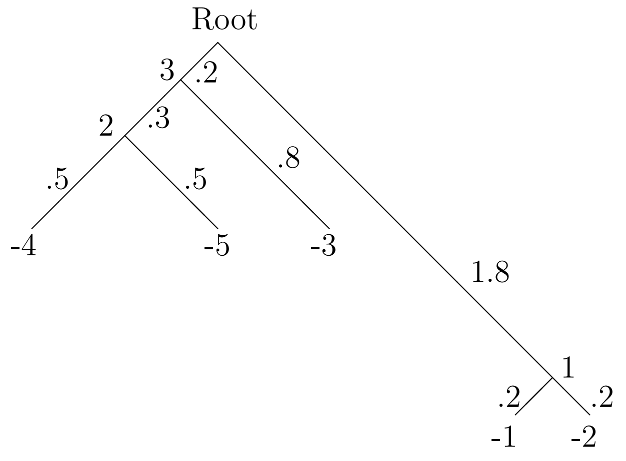


Figure 5. Rooted five-leaf tree

In order to reduce the number of parameters we use the two stationary distributions, π_X for all edges to the left of the root node and π_Y for all other edges. Further, if $\pi_X \neq \pi_Y$, the initial distribution at the root node can differ from these. The program proceeds recursively at each internal node commencing with the root node, and proceeding from the bottom of the first matrix using the rate associated with each edge from the second matrix. In the case illustrated by the matrices and the tree in the Figure 5, the program starts at the root giving the joint distribution for the nodes 3 and 1. Then it goes to node 3 giving the joint distribution at the nodes 1, 2 and -3, and so on.

Note that the two programs, gn and gn2, start at the root node and proceed to the end nodes via each bifurcation at an internal node. This is necessary since in general we may not have stationary, homogeneous or reversible processes. This differs from the Felsenstein (1981) method, which proceeds by pruning pairs of end nodes and so applies only to stationary processes. The method of generating the entire probability distribution here is only feasible for relatively small values of K, say $K \leq 15$, since it needs memory for 4^K probabilities.

2.7. Evolutionary Models

All phylogenetic methods make assumptions, whether implicit or explicit, about the substitution models that operate at the DNA or protein levels (see for example Felsenstein, 1973, Goldman, 1990, Penny, 1992). Models are abstractions or simplifications of hypothesis of the real world, but they are intended to include the most important features and omit irrelevant detail.

2.7.1. The Jukes-Cantor Model. The simplest (and earliest) model of nucleotide substitution is the Jukes-Cantor model (Jukes, 1969). The Jukes-Cantor model set the S matrix defined in (2.13) as

$$S = \begin{bmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{bmatrix},$$

where α is a positive constant called the evolutionary rate. For the Jukes-Cantor model, given that $UU^T = I$ and $\sum_{i=2}^{4} u_i u_i^T = I - \frac{1}{4}J$, where $J = \mathbf{1}\mathbf{1}^T$, $u_1 = \frac{1}{2}\mathbf{1}$ and $\pi = \frac{1}{4}\mathbf{1}$, the rate matrix R defined in (2.14) and (2.15) can be written as

$$R = \alpha\left(\frac{1}{4}J - I\right), \tag{2.34}$$

so the instantaneous transition rates R_{ij} and R_{ii} are defined by $R_{ij} = \alpha/4$ for all $i \neq j$ and $R_{ii} = -3\alpha/4$.
Now for the transition probability matrix $P(t)$ defined in (2.16), since $\lambda_1 = 0$, $\lambda_2 = \lambda_3 = \lambda_4 = \alpha$, we can rewrite the transition matrix $P(t)$ as

$$\begin{aligned} P(t) &= u_1 u_1^T + e^{-\alpha t} \sum_{i=2}^{4} u_i u_i^T \\ &= \frac{1}{4}J + e^{-\alpha t}(I - \frac{1}{4}J). \end{aligned} \tag{2.35}$$

That is

$$P_{ii}(t) = \frac{1}{4} + \frac{3}{4}e^{-t\alpha},$$

$$P_{ij}(t) = \frac{1}{4} - \frac{1}{4}e^{-t\alpha}, i \neq j.$$

From the last two equations we can see that as $t \to \infty$, both $P_{ii}(t)$ and $P_{ij}(t)$ approach $\frac{1}{4}$.

The Jukes-Cantor model assumes stationarity with stationary probability distribution equal to $\pi = (1/4, 1/4, 1/4, 1/4)^T$. It assumes also that the model is reversible.

2.7.2. The Kimura Models. The Kimura model is a generalization of the Jukes-Cantor model. The highly symmetric assumptions implicit in the Jukes-Cantor model are not realistic. A *transversion* is the replacement of ($A \to C, C \to A, A \to T, T \to A, C \to G, G \to C, G \to T, T \to G$) and *transition* is the replacement of ($A \to G, G \to A, C \to T, T \to C$). Kimura (1980) proposed a two-parameter model to allow for differences in the rates of transition and transversion. In the Kimura model the matrix S is set as

$$S = \begin{bmatrix} -\alpha - 2\beta & \alpha & \beta & \beta \\ \alpha & -\alpha - 2\beta & \beta & \beta \\ \beta & \beta & -\alpha - 2\beta & \alpha \\ \beta & \beta & \alpha & -\alpha - 2\beta \end{bmatrix}.$$

The Kimura model turns into the Jukes-Cantor model when $\beta = \alpha$. Its stationary probability distribution is unique (and is identical to that of the Jukes-Cantor model); also the model is reversible.

A generalization for this model is given by the Kimura 3ST model. The rate matrix for this model is of the form

$$S = \begin{bmatrix} -\alpha - \beta - \gamma & \alpha & \beta & \gamma \\ \alpha & -\alpha - \beta - \gamma & \gamma & \beta \\ \beta & \gamma & -\alpha - \beta - \gamma & \alpha \\ \gamma & \beta & \alpha & -\alpha - \beta - \gamma \end{bmatrix}.$$

2.7.3. The Felsenstein Model. The Felsenstein (1981) model is also a generalization of the Jukes-Cantor model. The rate matrix R is set as

$$R = \begin{bmatrix} -\alpha + \alpha\pi_1 & \alpha\pi_2 & \alpha\pi_3 & \alpha\pi_4 \\ \alpha\pi_1 & -\alpha + \alpha\pi_2 & \alpha\pi_3 & \alpha\pi_4 \\ \alpha\pi_1 & \alpha\pi_2 & -\alpha + \alpha\pi_3 & \alpha\pi_4 \\ \alpha\pi_1 & \alpha\pi_2 & \alpha\pi_3 & -\alpha + \alpha\pi_4 \end{bmatrix}.$$

where α is the model parameter and $(\pi_1, \pi_2, \pi_3, \pi_4)$ is the stationary distribution.

This model also generalized the Jukes-Cantor model, to which it will be reduced if $\pi_1 = \pi_2 = \pi_3 = \pi_4 = 0.25$.

2.7.4. The Hasegawa-Kishino-Yaho Model. The Hasegawa (1985) model (HKY) depends on five parameters and assumes that the rate matrix R is given by

$$R = \begin{bmatrix} -\beta\pi_2 - \alpha\varphi_1 & \beta\pi_2 & \alpha\pi_3 & \alpha\pi_4 \\ \beta\pi_1 & -\beta\pi_1 - \alpha\varphi_1 & \alpha\pi_3 & \alpha\pi_4 \\ \alpha\pi_1 & \alpha\pi_2 & -\beta\pi_4 - \alpha\varphi_2 & \beta\pi_4 \\ \alpha\pi_1 & \alpha\pi_2 & \beta\pi_3 & -\beta\pi_3 - \alpha\varphi_2 \end{bmatrix},$$

where $\varphi_1 = \pi_3 + \pi_4,\ \varphi_2 = \pi_1 + \pi_2$. The HKY model is an amalgam of Kimura's (1980) two parameters model and Felsenstein's (1981) model, since if $\beta = \alpha$ this will reduce this model to Felsenstein model and if $\pi_1 = \pi_2 = \pi_3 = \pi_4 = 0.25$ it will reduce it to Kimura model.

2.7.5. The General Time-Reversible Markov Model. The general time-reversible Markov model (Lanave, 1984) assumes that the process is stationary, with a rate matrix R given by

$$R = \begin{bmatrix} R_{11} & S_{12}\pi_2 & S_{13}\pi_3 & S_{14}\pi_4 \\ S_{21}\pi_1 & R_{22} & S_{23}\pi_3 & S_{24}\pi_4 \\ S_{31}\pi_1 & S_{32}\pi_2 & R_{33} & S_{34}\pi_4 \\ S_{41}\pi_1 & S_{42}\pi_2 & S_{43}\pi_3 & R_{44} \end{bmatrix}, \tag{2.36}$$

where $R_{ii} = -\sum_{y \neq i} S_{iy}\pi_y,\ i = 1, \cdots, 4$. A necessary condition for the Markov process with rate matrix (2.36) to be reversible is that $S_{ij} = S_{ji},\ i \neq j$. If the initial probabilities are equal π, then the process is stationary, and this is also a sufficient condition for reversibility.

For a two leaf tree the general time-reversible Markov model is a nine free parameter model, which are six parameters for the rate matrix, $s = (S_{12},\ S_{13},\ S_{14},\ S_{23},\ S_{24},\ S_{34})$ and three parameters for π, since $\sum_{i=1}^{4} \pi_i = 1$.

2.7.6. The General Markov Model for our Programs. In this subsection we will describe the general Markov model for which we develop the gn and gn2 programs.

As we mentioned while describing the way the two programs work, for any tree with K matched sequences, our programs will divide the tree into two parts depending on the first bifurcation from the tree root. Our model will have $16 + (K - 2)$ parameters for the gn program, which are six parameters for the s-vector, three parameters for π_X, three parameters for π_Y, three parameters for f_0, one ρ and $K - 2$ heights, note that the height from the root to any external node is equal to one. For the gn2 program will have $15 + (2K - 3)$ parameters, which are six parameters for the s-vector, three parameters for π_X, three parameters for π_Y, three parameters for f_0 and $2K - 3$ for the rate matrix. Note that these lengths need a different standardization and we make the sum of all rates equal to a fixed value since multiplying the rate matrix by a constant scale and dividing the elements of the s-vector by the same constant will produce the same distribution function.

Chapter 3

Simulation the Evolution of Matched Nucleotide Sequences on a Phylogeny

3.1. Introduction

In order to study the performance of phylogenetic methods to estimate phylogeny and evolutionary parameters, we need nucleotide sequences generated under controlled conditions. Monte Carlo simulations provide an opportunity to produce such data. The scientist chooses: (*i*) a rooted, bifurcating tree with labelled tips and known edge lengths; (*ii*) a Markov model for each edge that describes the instantaneous rates of change from one nucleotide to one of the other nucleotides; and (*iii*) an ancestral sequence from which the descendant sequences evolve. The simulated data can then be used, for example, to evaluate the performance of the phylogenetic methods (Gaut and Lewis, 1995) or the reliability of phylogenetic results (Rambaut and Grassly, 1997).

Several computer programs have been developed to simulate evolution of nucleotide sequences on a known bifurcating tree (listed in Jermiin et al., 2003) but most of these have restricted attention to stationary and homogeneous cases and in general have have not given detailed descriptions of their relation to the Markov processes of substitutions. Indeed, there are many studies of molecular evolution that have used Monte Carlo simulations without describing in sufficient detail the statistical and computational aspects of the software that was developed to do the simulations (see e.g. Van Den Bussche et al., 1998; Conant and Lewis, 2001). The fact that the computational implementation of the Markov process was not described in detail means that these and many other simulation studies cannot be replicated. In some cases the simulation programs are no longer available.

In this chapter we describe three methods to simulate the evolution of nucleotide sequences on a known bifurcating tree. In the first method, we calculate the joint probabilities and obtain the random numbers of sites containing each arrangement of matched nucleotides using the multinomial distribution; in the second method, we use the Markov process at each site (as in Rambaut and Grassly, 1997), or the equivalent embedded Markov chain and simulated waiting times, to obtain the simulated results at the ends of the tree; and in the third method, we describe a simulation method employed in the program called *Hetero* (Jermiin et al., 2003), and we show that the third method produces results that approximate those from the other two methods. The methods cater for different computational situations, and therefore require different statistical approaches.

3.2. Generating Random DNA Samples from a Multinomial Distribution

A multinomial distribution with parameters n, $p_1, p_2, ..., p_r$, where n is a positive integer and $0 \leq p_i \leq 1$, $\sum_{i=1}^{r} p_i = 1$, results from n independent trials, each of which can result in an outcome of one of r types, with probabilities $p_1, p_2, ..., p_r$. Let N_i be the total number of outcomes of type i in n trials, then the multinomial distribution is

$$P(N_1 = n_1, N_2 = n_2, ..., N_r = n_r) = \begin{pmatrix} & n & \\ n_1 & ... & n_r \end{pmatrix} p_1^{n_1}\, p_2^{n_2} \, ... \, p_r^{n_r},$$

provided that $n_1 + n_2 + ... + n_r = n$, where

$$\begin{pmatrix} & n & \\ n_1 & ... & n_r \end{pmatrix} = \frac{n!}{n_1!\, n_2!\, ...\, n_r!}.$$

The binomial distribution is a special case of multinomial distribution with $r = 2$. Further, for $k < r$ and given $N_1 + \cdots + N_k = m$, the conditional distribution of N_{k+1} is binomial with parameters $n - m$ and

$p_{k+1}/(1-(p_1+\cdots+p_k))$.
For the case of a two edge tree $r = 16$. Now knowing the joint distribution function and the sample size, we will generate the observed divergence matrix

$$N = \begin{bmatrix} N_{11} & N_{12} & N_{13} & N_{14} \\ N_{21} & N_{22} & N_{23} & N_{24} \\ N_{31} & N_{32} & N_{33} & N_{34} \\ N_{41} & N_{42} & N_{43} & N_{44} \end{bmatrix},$$

where N_{ij} is the number of times we observe $X_k(t) = i, Y_k(t) = j, 1 \leq k \leq n$, $i, j = 1, \cdots, 4$. Given that all nucleotide sites are independent and identically distributed random samples, N_{ij} will be distributed as a joint multinomial distribution with parameters n (the sequence length) and

$$F(t) = \begin{bmatrix} f_{11}(t) & f_{12}(t) & f_{13}(t) & f_{14}(t) \\ f_{21}(t) & f_{22}(t) & f_{23}(t) & f_{24}(t) \\ f_{31}(t) & f_{32}(t) & f_{33}(t) & f_{34}(t) \\ f_{41}(t) & f_{42}(t) & f_{43}(t) & f_{44}(t) \end{bmatrix},$$

where $F(t)$ is the joint probability distribution for the two sequences $X(.)$ and $Y(.)$. Here we take $p_1 = f_{11}, p_2 = f_{12}, \cdots, f_{44} = p_{16}$, and write N_{11} for N_1, and so forth. Now in order to generate the multinominal distribution of N_{ij}, first generate N_{11} random sample from binomial distribution with parameter n and $f_{11}(t)$; given N_{11}, conditionally N_{12} is a random sample from binomial distribution with parameters $n - N_{11}$ and $f_{12}(t)/(1-f_{11}(t))$; given N_{11} and N_{12}, conditionally N_{13} is a random sample from the binomial distribution with parameters $n-(\ N_{11}+N_{12})$ and $f_{13}(t)/(1-(f_{11}(t)+f_{12}(t)))$; and so forth. The last two terms are, given N_{11}, N_{12} . . . N_{42}, conditionally N_{43} is a random sample from the binomial distribution with parameters $n - N_{11} - N_{12} - \cdots - N_{42}$ and $f_{43}(t)/(1 - f_{11}(t) - \cdots - f_{42}(t))$, and given N_{11}, N_{12} . . . N_{43}, conditionally N_{44} is a random sample from binomial distribution with parameters $n - N_{11} - \cdots N_{43}$ and $(f_{44}(t)/(1 - f_{11}(t) - \cdots - f_{43}(t))) = 1$.

We shall now give a numerical example in which we generate the observed divergence matrix N from the binomial distribution, and then find the matched sequences from the observed divergence matrix.

Example 3.1. For a two edge tree, consider the case of stationarity but not homogeneity processes. That is let

$$\Pi_Y = F(0) = \Pi_X \text{ and } S_X \neq S_Y.$$

Taking $\Pi_X = diag(0.25, 0.25, 0.25, 0.25)$,

$$S_X = \begin{bmatrix} -0.6 & 0.2 & 0.2 & 0.2 \\ 0.2 & -0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & -0.6 & 0.2 \\ 0.2 & 0.2 & 0.2 & -0.6 \end{bmatrix} \tag{3.1}$$

and

$$S_Y = \begin{bmatrix} -0.7 & 0.2 & 0.2 & 0.3 \\ 0.2 & -0.9 & 0.3 & 0.4 \\ 0.2 & 0.3 & -0.9 & 0.4 \\ 0.3 & 0.4 & 0.4 & -1.1 \end{bmatrix}.$$

Taking the same time for the two edges t=0.5, we can calculate the theoretical joint distribution function $F(t)$.

For a sequence containing 10000 nucleotides, that is $n = 10000$, the expected divergence matrix is given by

$$nF(t) = \begin{bmatrix} 2135 & 114 & 114 & 138 \\ 114 & 2086 & 139 & 162 \\ 114 & 139 & 2086 & 162 \\ 138 & 162 & 162 & 2037 \end{bmatrix} \tag{3.2}$$

Now depending on the joint distribution function $F(t)$ and a sample size $n = 10000$, we can simulate an observed divergence matrix, an example being

$$N = \begin{bmatrix} 2189 & 120 & 126 & 143 \\ 120 & 2105 & 142 & 151 \\ 117 & 150 & 2013 & 152 \\ 152 & 163 & 139 & 2018 \end{bmatrix}.$$

which is similar to the expected divergence matrix (3.2).

In the previous example we showed how to generate random nucleotide sequences from a multinomial distribution. We can apply this method to any tree of K matched sequences. That is, depending on the joint distribution function $F(t)$ of K matched sequences, we can generate a 4^K observed divergence array using the same multinomial process described for two matched sequences.

From the 4^K array we can find the K matched sequences. For instance, in the previous example, where $K = 2$, we obtained two matched sequences with 2189 sites having the value A in both sequences, 120 sites having the value A for the first sequences and G for the second sequence, and so on; that is

$$\begin{array}{lcccc} & 2189 & 120 & \cdots & 2018 \\ \textit{Sequence } 1 & \overbrace{AAA\cdots A} & \overbrace{AAA\cdots A} & \cdots\cdots & \overbrace{TTT\cdots T} \\ \textit{Sequence } 2 & AAA\cdots A & GGG\cdots G & \cdots\cdots & TTT\cdots T \end{array}$$

We have developed an S-Plus function to calculate the divergence array N for any set of K matched sequences. This function depends on the joint distribution function $F(t)$ for the tree and on the sample size n. We also we developed another function, which changes any 4^K divergence array N to K matched sequences.

3.3. Generating Random DNA Samples From The Process At Each Site

Consider the processes at the first site, $X(t)$ and $Y(t)$: at time $t = 0$, the processes satisfy $X(0) = Y(0)$, where $P(X(0) = i) = f_{0i}$. After t time units, the processes are in states $X(t)$ and $Y(t)$, where $P(X(t) = j|X(0) = i) = P^X_{ij}(t)$ and $P(Y(t) = j|Y(0) = i) = P^Y_{ij}(t)$. Repeating this construction independently n times leads to two sequences of n independent states corresponding to $X(t)$ and $Y(t)$. From these two matching sequences, we can find the observed divergence matrix N such that N_{ij} is the number of times that $X(t) = i$ and $Y(t) = j$. This method of simulation has been proposed earlier by (Rambaut and Grassly, 1997).

Alternatively, we could use embedded Markov chains, $\tilde{X}(m), m = 0, 1, \cdots$, which as well only involves the rate matrix(see for example section 8.3 of Iosifescu, 1980). At time $T_0 = 0$ the process is in state $\tilde{X}(0) = X(0)$; the process stays in state $\tilde{X}(0)$ for a positive time, but at the end of some random time, T_1, it jumps to a new state $\tilde{X}(1)$ with probability Q_{ij} where,

$$\begin{aligned} Q_{ij} &= P[\tilde{X}(m+1) = j|\tilde{X}(m) = i] \\ &= \begin{cases} \frac{R_{ij}}{-R_{ii}}, & \text{if } i \neq j, \\ 0, & \text{if } i = j. \end{cases} \end{aligned}$$

The process stays in this state for some time until, at the end at some time, T_2, it jumps to another state $\tilde{X}(2)$, and so on. We continue this process until we reach a fixed time t such that $T_{M-1} \leq t < T_M$, then take $X(t) = \tilde{X}(M-1)$. Let the random variable

$$W_M = T_{M+1} - T_M$$

be the positive time that the process stays in the state $\tilde{X}(M)$, called the "sojourn time" in the state. Conditional on $\tilde{X}(M) = i$, W_M is distributed as an exponential variable with parameter R_i, that is

$$P(W_M > t|\tilde{X}(M) = i) = e^{-R_i t},$$

where $R_i = -R_{ii}$ is the absolute value of the diagonal of the rate matrix. Taking $\tilde{Y}(0) = Y(0) = X(0)$ and proceeding as above independently we get $Y(t)$. Repeating the construction independently n times leads to sequences of n independent states corresponding to $X(t)$ and $Y(t)$. From these two matching sequences we can find the observed divergence matrix N such that N_{ij} is the number of times that $X(t) = i$ and $Y(t) = j$. We would not generate simulated data using this embedded chain but we describe it since there are some similarities to the generation method of the next section.

Example 3.2. Consider Example 3.1 in Section 3.2. First we generate a sequence of 10000 nucleotides taken as the common ancestor. Conditional on this sequence, we generate two independent sequences using the method of Rambaut and Grassly (1997) described previously. This led to

$$N = \begin{bmatrix} 2234 & 110 & 122 & 136 \\ 102 & 2090 & 153 & 150 \\ 126 & 129 & 1994 & 149 \\ 143 & 181 & 167 & 2014 \end{bmatrix}$$

which is again similar to the expected divergence matrix (3.2). The matrix is the realization of the same processes.

Example 3.3. Consider exactly the same case as the previous example. Although we would not in practice use the embedded method, we illustrate it here, conditional on the same common ancestor sequence, by generating two independent sequences using the embedded method. This leads to

$$N = \begin{bmatrix} 2146 & 103 & 103 & 122 \\ 113 & 2053 & 159 & 171 \\ 118 & 135 & 2087 & 173 \\ 156 & 129 & 176 & 2056 \end{bmatrix}$$

which is also similar to the expected divergence matrix (3.2) and it is the realization of the same processes.

3.3.1. Generalization of the Rambaut and Grassly Method. We shall show here the generalization of the Rambaut and Grassly (1997) method in order to be able to simulate K matched sequences.

Suppose there are K matched nucleotide sequences of length n derived from a common ancestor with a known tree topology. Each edge has length t_ℓ, where ℓ takes one of the values $\{1, \cdots, K-2\}$ for edges leading to an internal node and $\{-1, \cdots, -K\}$ for edges leading to a leaf, where we number the nodes with positive and negative integers for internal and leaf nodes, respectively. At time $t = 0$, take $X_1(0) = \cdots = X_K(0)$, generating $X_1(0)$ using

$$P(X_1(0) = i) = f_{0i}, \ i = 1, \cdots, 4.$$

If the two edges of the tree starting at this ancestral node are of lengths t_A and t_B, and the taxa split into groups $X_1(t_A) = \cdots = X_m(t_A)$ and $X_{m+1}(t_B) = \cdots = X_K(t_B)$, then we generate these using

$$P[X_1(t_A) = j | X_1(0) = i] = P^A_{ij}(t_A) \tag{3.3}$$

and

$$P[X_{m+1}(t_B) = j | X_{m+1}(0) = i] = P^B_{ij}(t_B) \tag{3.4}$$

$i, j = 1, \cdots, 4$, where P^A and P^B denote the transition matrices of the Markov processes operating along the two lineages descending from the common ancestor and t_A and t_B are the edge lengths from the root node. This process of generation is repeated at each of the descendant internal nodes; for example, at the second step, if $t_A < t_B$, depending on the sequence on the node corresponding to A the process generates two sequences. This is continued until we get K matched values $X_1(t), \cdots, X_K(t)$ at the tree leaves. Repeating this construction independently n times gives K matched sequences. From these sequences, we find the 4^K divergence array N.

We developed an S-Plus function to generate the divergence array N for any K matched sequence tree, this function requires a tree description based on a matrix describing the tree, a set of parameters θ and the sequence length n. We use here the same technique we used for the gn program, that is, we use the two rate matrices, $R^X = S\Pi_X$ for all edges to the left of the root node and $R^Y = S\Pi_Y$ for all other edges. So $P^A(t) = P^B(t)$ in (3.3) and (3.4) is equal to either $P^X(t)$ or $P^Y(t)$ for all internal nodes except the root node. The set of parameters required for this algorithm is: π_X, π_Y, f_0, six variables for the symmetric S matrix, ρ and a vector of heights depending on the size of the tree. This means that this function allows us to generate the observed divergence matrix N in this general case.

3.4. Generating Random DNA Samples Using An Approximation Method

In this method, which has been implemented in a program called *Hetero* (Jermiin et al., 2003), we use only the rate matrices. Consider a two leaf tree first. Generate independent and identically distributed random variables

$$X_{01},\ X_{02},\ \ldots\ , X_{0n}$$

taking values $1, \cdots, 4$ with probability

$$P(X_{0k} = i) = f_{0i},\ k = 1,\ 2, \cdots, n\ ,\ i = 1, \cdots, 4\ .$$

Note that according to this notation X_{mn} is the random variable corresponding to the the n-th nucleotide in the m-th step in the Markov process – accordingly, X_{07} represents the nucleotide at the seventh position in the ancestral sequence. Let $Y_{0k} = X_{0k}$, this means that we have two sequences exactly the same, that is

$$\{X_{01}, X_{02},\ \ldots\ , X_{0n}\} = \{Y_{01}, Y_{02},\ \ldots\ , Y_{0n}\}.$$

Now from the first sequence $X_{01},\ X_{02},\ \ldots\ , X_{0n}$, select randomly an element W from $1, \cdots, n$, with probability $1/n$, then change the element in the W-th position X_{0W} to X_{1W} with transition probability

$$P(X_{1W} = j | X_{0W} = i, W) = \begin{cases} 1 + R^X_{ii}, & \text{if } i = j, \\ R^X_{ij}, & \text{if } i \neq j. \end{cases}$$

Now consider any element from the sequence, take for example $X_{01} = i$, so the transition probability of changing X_{01} to X_{11} is

$$P(X_{11} = j | X_{01} = i) = \begin{cases} (1 - \frac{1}{n}) + \frac{1}{n}(1 + R^X_{ii}), & \text{if } i = j, \\ \frac{1}{n} R^X_{ij}, & \text{if } i \neq j. \end{cases}$$

After the first step we get the sequence

$$X_{11},\ X_{12},\ \ldots, X_{1n}.$$

If we randomly choose any element from this new sequence, the probability that this element is equal to j is

$$f^X_{1j} = \sum_{i \neq j} (\frac{1}{n} f_{0i} R^X_{ij}) + f_{0j}(1 + \frac{R^X_{jj}}{n}).$$

In matrix notation this is

$$(f^X_1)^T = f^T_0 (I + \frac{1}{n} R^X).$$

For the second step we just repeat the first step depending on the new sequence

$$X_{11},\ X_{12},\ \ldots, X_{1n}.$$

This implies that after the second step we have a new sequence $X_{21},\ X_{22}, \cdots, X_{2n}$, with probability of selecting any element from this sequence and this element is equal to j is

$$\begin{aligned} (f^X_2)^T &= (f^X_1)^T (I + \frac{1}{n} R^X) \\ &= (f_0)^T (I + \frac{1}{n} R^X)^2. \end{aligned}$$

We continue this process, and after m repetitions of the process, we get the sequence

$$X_{m1},\ X_{m2},\ \ldots, X_{mn}.$$

Let $m = nt$, so we have nt steps of size $1/n$, then

$$(f_{nt}^X)^T = (f_0^X)^T(I + \frac{1}{n}R^X)^{nt},$$

but

$$\begin{aligned} (f_{nt}^X)^T &= f_0^T(I + \frac{1}{n}R^X)^{nt} \\ &= f_0^T e^{tR^X + O(\frac{1}{n})} \\ &= f_0^T P^X(t)(1 + O(\frac{1}{n})), \end{aligned}$$

that is

$$(f_{nt}^X)^T \approx (f_0)^T P^X(t).$$

In the same way we apply this procedure for the second sequence $Y_{01},\ Y_{02},\ Y_{03},\ \ldots, Y_{0n}$, such that for $m = nt$ we get

$$\begin{aligned} (f_{nt}^Y)^T &= (f_0)^T(I + \frac{1}{n}R^Y)^{nt} \\ &\approx (f_0)^T P^Y(t) \end{aligned}$$

From the two sequences

$$X_{nt1},\ X_{nt2},\ \ldots, X_{ntn},\ \text{and}\ Y_{nt1},\ Y_{nt2},\ \ldots, Y_{ntn}$$

we can simulate an observed divergence matrix N.
Suppose that at the same time we start from the two sequences

$$\{X_{01}, X_{02}, \cdots, X_{0n}\} = \{Y_{01}, Y_{02},\ \ldots, Y_{0n}\}.$$

Select randomly two elements, W and L from $1, \cdots, n$, each with probability $1/n$, then change the elements in the W-th and L-th positions, X_{0W} to X_{1W} and Y_{0L} to Y_{1L} with transition probability

$$P(X_{1W} = j, Y_{1L} = j' | X_{0W} = i, Y_{0L} = i', W, L)$$

$$= \begin{cases} (1 + R_{ii}^X)(1 + R_{i'i'}^Y), & \text{if } i = j, i' = j' \\ (1 + R_{ii}^X)R_{i'j'}^Y, & \text{if } i = j, i' \neq j' \\ (1 + R_{i'i'}^Y)R_{ij}^X, & \text{if } i \neq j, i' = j' \\ R_{ij}^X R_{i'j'}^Y, & \text{if } i \neq j, i' \neq j' \end{cases}$$

Now from the first sequence take for example $X_{01} = i$ and from the second sequence take for example $Y_{01} = i'$, then the transition probability of changing X_{01} to X_{11} and Y_{01} to Y_{11} is

$P(X_{11} = j, Y_{11} = j' | X_{01} = i, Y_{01} = i')$

$$= \begin{cases} (1 + \frac{1}{n}R^X_{ii})(1 + \frac{1}{n}R^Y_{i'i'}), & \text{if } i = j, i' = j' \\ (1 + \frac{1}{n}R^X_{ii})\frac{1}{n}R^Y_{i'j'}, & \text{if } i = j, i' \neq j' \\ (1 + \frac{1}{n}R^Y_{i'i'})\frac{1}{n}R^X_{ij}, & \text{if } i \neq j, i' = j' \\ (\frac{1}{n})^2 R^X_{ij} R^Y_{i'j'}, & \text{if } i \neq j, i' \neq j' \end{cases}$$

After the first step, we get the sequences

$$X_{11}, X_{12}, \ \dots\ , X_{1n} \text{ and } Y_{11}, Y_{12}, \ \dots\ , Y_{1n}.$$

If we randomly choose any two elements one from the first sequence and the other from the second sequence, then the probability that these elements are j and j', respectively, is

$$f_{1jj'} = [(I + \frac{1}{n}R^X)' F(0)(I + \frac{1}{n}R^Y)]_{jj'} = (F_1)_{jj'}$$

For the second step we just repeat the first step depending on the new sequences, so we will get another two sequences

$$X_{21}, X_{22}, \ \dots\ , X_{2n} \text{ and } Y_{21}, Y_{22}, \ \dots\ , Y_{2n},$$

with probability of selecting any two elements from the first sequence and the other from the second sequence and these elements are j and j' respectively is

$$\begin{aligned} f_{2jj'} &= [(I + \frac{1}{n}R^X)' F_1 (I + \frac{1}{n}R^Y)]_{jj'} \\ &= [((I + \frac{1}{n}R^X)')^2 F(0)(I + \frac{1}{n}R^Y)^2]_{jj'}. \end{aligned}$$

After m steps of repeating this process, we get the sequences

$$X_{m1}, X_{m2}, \cdots, X_{mn}; Y_{m1}, Y_{m2}, \cdots, Y_{mn}.$$

If $m = nt$ then the probability of selecting any two elements from the first sequence and the other from the second sequence and these elements are j and j', respectively, is

$$f_{ntjj'} = [((I + \frac{1}{n}R^X)')^{nt} F(0)(I + \frac{1}{n}R^Y)^{nt}]_{jj'}$$

that is

$$(f_{ntjj'}) = ((I + \frac{1}{n}R^X)')^{nt} F(0)(I + \frac{1}{n}R^Y)^{nt}.$$

Now

$$\begin{aligned}(I + \frac{1}{n}R^X)^{nt} &= e^{tR^X + O(\frac{1}{n})} \\ &= P^X(t)(1 + O(\frac{1}{n})) \\ &\simeq P^X(t),\end{aligned}$$

so

$$\begin{aligned}(f_{ntjj'}) &\simeq P^X(t)^T F(0) P^Y(t) \\ &= F(t).\end{aligned}$$

Example 3.4. Consider the same set of parameters described in Example 3.1 and a sequence of 10000 nucleotides, using the previous method of generation, the observed divergence matrix is

$$N = \begin{bmatrix} 2119 & 115 & 132 & 141 \\ 119 & 2063 & 125 & 155 \\ 118 & 125 & 2101 & 150 \\ 148 & 183 & 134 & 2072 \end{bmatrix}$$

which is again similar to the expected divergence matrix (3.2). The matrix is an approximate realization of the same processes.

3.4.1. Generalization for the Approximation Method. We will show here the generalization of the previous approximation method in order to be able to simulate K matched sequences.

Suppose there are K matched nucleotide sequences of length n derived from a common ancestor with a known tree topology. Define the edge length as in the generalization of Rambaut and Grassly (1997) method. At the beginning, at time $t = 0$, the process generates a sequence of length n, $X_{01}, \cdots, X_{0n}$, from the numbers $1, \cdots, 4$, with probability

$$P(X_{0k} = i) = f_{0i},\ i = 1, \cdots, 4,\ k = 1, \cdots, n.$$

Let $Y_{0k} = X_{0k},\ k = 1, \cdots, n$. If the two edges of the tree starting from the ancestral node are of lengths t_A and t_B, then the process generates two sequences each of length n, such that, for the edge corresponding to node A, from the ancestor sequence $X_{01}, \cdots, X_{0n}$, we randomly select an element W from $1, \cdots, n$ with probability $1/n$, then change the element in the W-th position X_{0W} to X_{1W} with probability

$$P(X_{1W} = j | X_{0W} = i, W) = \begin{cases} 1 + R^A_{ii}, & \text{if } i = j, \\ R^A_{ij}, & \text{if } i \neq j, \end{cases}$$

where R^A denotes the rate matrix of the Markov processes operating along the lineage descending from the common ancestor to the node corresponding to A. This means that we will get the sequence $X_{11}, \cdots, X_{1n}$. We repeat this process $m = nt_A$ times, where t_A represents the length from the ancestral node to the node corresponding to A. This repetition will give us the sequence $X_{m1}, \cdots, X_{mn}$ for the node corresponding to A.

Simultaneously, for the edge corresponding to node B, from the ancestor sequence $Y_{01}, \cdots, Y_{0n}$, we randomly select an element L from $1, \cdots, n$ with probability $1/n$, then change the element in the L-th position Y_{0L} to Y_{1L} with probability

$$P(X_{1L} = j | X_{0L} = i, L) = \begin{cases} 1 + R^B_{ii}, & \text{if } i = j, \\ R^B_{ij}, & \text{if } i \neq j, \end{cases}$$

where R^B denotes the rate matrix of the Markov processes operating along the lineage descending from the common ancestor to the node corresponding to B. This means that we will get the sequence $Y_{11}, \cdots, Y_{1n}$. We repeat this process $u = nt_B$ times, where t_B represents the length from the ancestral node to the node corresponding to A. This repetition will give us the sequence $X_{u1}, \cdots, X_{un}$ for the node corresponding to B.

This process of sequence generation is repeated at each of the descendant internal nodes; for example, at the second step, if $t_A < t_B$, depending on the sequence on the node corresponding to A the process generates two sequences. This is continued until we get K matched sequences at the tree leaves. From these K matched sequences, we find the 4^K divergence array N.

We developed an S-Plus function to generate the divergence array N for any set of K matched sequences generalizing the program called *Hetero* (Jermiin et al., 2003). The function requires the same set of parameters that the generalization of the Rambaut and Grassly (1997) method required. We also use the same technique with two rate matrices, $R^X = S_X \Pi_X$ for all edges to the left of the root node and $R^Y = S_Y \Pi_Y$ for all other edges.

3.5. Examples based on Five Matched Sequences

We will use the Multinomial method described in Section 3.2 to generate the divergence array N, for larger trees (five leaf tree), knowing that the other two methods will give the same result.

In the following example we discuss the case of five matched sequences evolving on a tree under non-stationary and non-homogeneous conditions. The purpose of this example to generate a set of data similar to the real data we will analyze in the next chapter.

Example 3.5. Consider the case of five leaf tree, where the topology of the tree is described by the following merge matrix, corresponding to Figure 4(D),

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ 1 & -3 \\ 3 & 2 \end{bmatrix}. \tag{3.5}$$

Assume that the process is neither stationary nor homogeneous, by taking

$$\pi = \pi_X = \pi_Y = (0.2, 0.2, 0.2, 0.4)^T,$$

$f_0 = (0.25, 0.25, 0.25, 0.25)^T$ and the s-vector $= (0.2, 0.2, 0.2, 0.2, 0.2, 0.2)$. Let $R^X = \rho R^Y$, $\rho = 3$, and take the vector of heights to be $(0.1, 0.6, 0.8, 1)$.

Now using the gn program described in Chapter 2, we can compute the theoretical joint distribution function $F(t)$ for the tree. For a sample size $n = 10000$ and depending on the theoretical joint distribution function, using the multinomial method we can generate the expected 4^5 divergence array N.

From the joint distribution function $F(t)$, the marginal distribution for the nodes $-1, -2$ and -3 are

$$F_{i....}(1) = F_{.i...}(1) = F_{..i..}(1) = (0.2409, 0.2409, 0.2409, 0.2772)$$

and for the nodes -4 and -5 are

$$F_{...i.}(1) = F_{....i}(1) = (0.2274, 0.2274, 0.2274, 0.3177).$$

The reason for this difference between the marginal distributions is because the edges $-1, -2$ and -3 are derived from the same process, which is different from the process from which the edges -4 and -5 are derived. By comparison, using the multinomial method we can simulate an observed joint divergence array N, from which the estimated marginal distribution at each end node obtained by summing $N/10000$ over other dimensions for $-1, -2$ and -3 are

$$(0.2392, 0.2386, 0.2434, 0.2788),$$

$$(0.2400, 0.2380, 0.2440, 0.2780),$$

$$(0.2401, 0.2417, 0.2408, 0.2774),$$

and for -4 and -5 are

$$(0.2286, 0.2261, 0.2280, 0.3173),$$

$$(0.2267, 0.2229, 0.2329, 0.3175),$$

Notice that these estimated marginals are very close to the marginals derived from the theoretical joint distribution $F(t)$. Notice also that the marginal distributions for the edges -4 and -5 are closer to the stationary distribution, π, than the marginal distributions for the edges $-1, -2$ and -3, since the rate matrix for the edges -4 and -5 is multiplied by a factor $\rho = 3$.

We can obtain any 4×4 observed divergence matrix for any pair such as $(-1, -2)$ by summing N over the other three dimensions.

Example 3.6. Consider the same tree as the previous example and assume non-stationary and non-homogeneous processes, but in this case let

$$\pi_X^T = (0.2, 0.2, 0.2, 0.4),$$

$$\pi_Y^T = (0.3, 0.3, 0.3, 0.1)$$

and

$$f_0^T = (0.25, 0.25, 0.25, 0.25).$$

Let $R^X = R^Y$, and take the s-vector and the height vector as in the previous example. Now depending on the theoretical joint distribution function $F(t)$ and a sample of size $n = 10000$, using the multinomial method we can generate the expected 4^5 divergent array N. The estimated marginal distribution at each end node obtained by summing $N/10000$ over other dimensions for $-1, -2$ and -3 are

$$(0.2472, 0.2356, 0.2403, 0.2769),$$

$$(0.2471, 0.2409, 0.2343, 0.2777),$$

$$(0.2434, 0.2449, 0.2359, 0.2758),$$

which are very close to the marginal distributions derived from the joint distribution $F(t)$ for the same edges

$$F_{i....}(1) = F_{.i...}(1) = F_{..i..}(1) = (0.2409, 0.2409, 0.2409, 0.2772),$$

and the same holds for node -4 and -5, in which case the estimated marginals are

$$(0.2594, 0.2580, 0.2540, 0.2286),$$

$$(0.2577, 0.2586, 0.2619, 0.2218),$$

which are very close to the marginal distribution derived from the joint distribution $F(t)$ for edge -4 and -5

$$F_{...i.}(1) = F_{....i}(1) = (0.2591, 0.2591, 0.2591, 0.2228).$$

3.6. Discussion

We have described three methods to simulate evolution of nucleotide sequences on a phylogenetic tree. The methods cater for different mathematical and computational limitations and requirements.

In addition to producing the simulated sequences, the first method also uses a theoretical joint probability distribution produced by the method in section 2.4. The method is relatively easy to implement, and computational time is essentially independent of the number of sequences K, and their length n. The method may, however, use too much memory when K is very large, because the method requires storage of at least 4^K elements (i.e. the joint probability distribution).

The second method produces the simulated sequences using Kn steps, in each of which only a small number of computations is required.

The method is relatively easy to implement, and requires memory for at least Kn elements. A version of the method was implemented in Seq-Gen (Rambaut and Grassly, 1997).

The third method also requires memory for at least Kn elements but may require more computational time than the other methods, especially when the edge lengths are long, the rates of change are low, and n is large. However, the method is simpler to implement because exponentiation of the rate matrices is not needed. Moreover, the method allows for additional features to be considered, including continued monitoring of properties of the sequence data (e.g. the number of times that each site has changed across the tree, and the change in nucleotide content over time). The method was implemented in *Hetero* (Jermiin et al., 2003).

It is possible to extend the simulation methods to processes with different rates at nucleotide sites. All methods could be easily extended to generate sequences under a codon model with only three fixed different rates. However, only the second and third methods can be extended to give a different randomly chosen rate at each site by choosing a rate for each site from the assumed distribution of rates, as has been proposed (Rambaut and Grassly, 1997).

CHAPTER 4

Tests of Homogeneity

4.1. Introduction

Most phylogenetic methods assume that the sequences of nucleotides have evolved under stationary, reversible, and homogeneous conditions. When these assumptions are violated by the data, there is an increased probability of errors in the phylogenetic estimates. Methods to examine aligned sequences are available, but they are rarely used, either because they are unknown or because they are poorly understood.

We describe and compare the available tests for symmetry of k-dimensional contingency tables from homologous sequences, and develop two new tests to evaluate different aspects of the evolutionary processes. For any pair of sequences, we consider a partition of the test for symmetry into a test for marginal symmetry and a test for internal symmetry. The proposed tests can be used to identify appropriate models for estimation of evolutionary relationships under a Markovian model. Simulations under more or less complex evolutionary conditions were done to display the performance of the tests. Finally, the tests were applied to an alignment of small-subunit ribosomal RNA sequences of five species of bacteria with the aim to outline the evolutionary processes under which they evolved.

Suppose we have k matched observations of n independently and identically distributed variables taking values in r categories. An example of such data would be an alignment of $k = 5$ sequences of $n = 2000$ nucleotides (implying that $r = 4$) or amino acids (implying that $r = 20$) – other examples are discussed in, for instance, Agresti (1990, Chapters 10 and 11). Data of this nature can be summarized in k-dimensional tables with r^k categories. Hypotheses of interest concern symmetry in these tables. In the particular cases of homologous nucleotide or amino acid sequences, tests of symmetry or marginal symmetry can be used to consider goodness of fit of the Markov models used to describe evolutionary processes. The importance of using these infrequently-used tests prior to phylogenetic analysis of aligned sequence data has long been common knowledge (Tavaré, 1986; Lanave and Pesole, 1993; Rzhetsky and Nei, 1995; Waddell and Steel, 1997; Waddell *et al.*, 1999) but has not yet been accommodated by the wider scientific community. The importance of employing these tests was recently emphasized by Jermiin *et al.* (2004).

In the simple case where $K = 2$, matched pairs tests can be used to test for symmetry and marginal symmetry. The null hypothesis for symmetry is

$$H_{0B} : f_{ij} = f_{ji}, i \neq j, i, j = 1, \dots, r,$$

where f_{ij} is the probability that a randomly chosen variable belongs to the ij-th category (Bowker, 1948), and the null hypothesis for marginal symmetry is

$$H_{0S} : f_{i.} = f_{.i}, i = 1, \dots, r,$$

where $f_{i.}$ is the sum of f_{ij} over j (Stuart, 1955). The two hypotheses are obviously the same in a 2×2 contingency table - in general, however, symmetry implies marginal symmetry, whereas the opposite is not so. We will show that Bowker's (1948) chi-squared test statistic for symmetry can be partitioned into two independent components, one component being Stuart's (1955) chi-squared test statistic for marginal symmetry, and the other component being a chi-squared test statistic for internal symmetry. This partition was formally proposed by O'Neill (1975).

In the more complex cases where $K > 2$, a test of marginal symmetry has been formulated for analyses of nucleotide sequences (Rzhetsky and Nei, 1995), but their test was not described for more general terms. Denote by $f_{i_1,\cdots,i_K}$ the probability of an observation belonging to the

i_j-th category of the j-th variable, $j = 1, \cdots, K,\ i_j = 1, \ldots, r$. Write

$$f^j_{i_j} = \sum_{\ell=1, \ell \neq j}^{K} \sum_{i_\ell=1}^{r} f_{i_1, \cdots, i_K}.$$

Clearly, f^j_i are the marginal probabilities of the j-th variable. We will use similar notation for an observed table. For instance, $n_{i_1, \cdots, i_K}$ represents the observed frequency or count in the $i_1, \cdots, i_K$-th cell of a r^K table, and n^j_i represents the total number of observed counts in the i-th category of the j-th dimension. We will derive a combined test for marginal symmetry of all sequences, essentially equivalent to that proposed by Rzhetsky and Nei (1995), and relate this test to tests for all pairs.

Finally, we consider a Markov model for evolution and discuss the use of these tests in deciding on appropriate topologies for a set of data assumed to be generated under the model. We obtain results by simulation that illustrates the use of the tests and we apply the tests to bacterial data that have been discussed previously, for example, in Galtier and Guoy (1995).

4.2. Bowker's Test for Symmetry

Consider an $r \times r$ contingency table with the ij-th cell containing the frequency n_{ij}. We will derive an orthogonal decomposition of the test statistic of Bowker (1948) for testing symmetry in terms of that of Stuart's (1955) test for marginal symmetry, which we will show later.

The test statistic of Bowker (1948) for symmetry is given by

$$S^2_B = \sum_{i<j} \frac{(n_{ij} - n_{ji})^2}{n_{ij} + n_{ji}},$$

or alternatively,

$$S^2_B = m^T B^{-1} m,$$

where

$$m^T = (n_{12} - n_{21}, \ldots, n_{1r} - n_{r1}, n_{23} - n_{32}, \ldots, n_{2r} - n_{r2}, \ldots, n_{r-1,r} - n_{r,r-1}),$$

and B is a diagonal matrix with elements $n_{12}+n_{21}, \ldots, n_{1r}+n_{r1}, n_{23}+n_{32}, \ldots, n_{2r}+n_{r2}, \ldots, n_{r-1,r}+n_{r,r-1}$.

4.3. Stuart's Test for Marginal Symmetry

Consider an $r \times r$ table with the ij-th cell frequency n_{ij}. Let

$$d = (n_{1.} - n_{.1}, \ldots, n_{r-1.} - n_{.r-1})^T$$

and V be the $(r-1) \times (r-1)$ matrix with the elements

$$v_{ij} = \begin{cases} n_{i.} + n_{.i} - 2n_{ii}, & i = j \\ -(n_{ij} + n_{ji}), & i \neq j. \end{cases}$$

Here V is the estimated covariance matrix of d under the assumption of marginal symmetry. The test statistic of Stuart (1955) for marginal symmetry is given by

$$S_S^2 = d^T V^{-1} d.$$

To derive an alternative expression of S_S^2 in terms of m, which was defined in the previous section, notice that d can be written as

$$d = Cm, \tag{4.1}$$

where C is a $(r-1) \times \frac{r(r-1)}{2}$ matrix, uniquely defined by equation (4.1), with the following form for the case $r = 4$,

$$C = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 \end{pmatrix}$$

As a result, Stuart's test statistic is expressed as

$$S_S^2 = m^T C^T V^{-1} C m.$$

A test statistic for marginal symmetry closely related to Stuart's test statistic was presented by Bhapkar (1966) as

$$S_{SB}^2 = d^T G^{-1} d,$$

where G is simply the estimated covariance matrix of d,

$$G = V - dd^T/n.$$

Noting that

$$G^{-1} = V^{-1} + \frac{V^{-1}dd^TV^{-1}}{(1 - d^TV^{-1}d/n)},$$

it can be seen, as was shown by Ireland et al. (1969), that

$$S_{SB}^2 = S_S^2/(1 - S_S^2/n).$$

4.4. Test for Internal Symmetry

Following O'Neill's (1975) formal proposal, we will now develop a test for internal symmetry that depends on Bowker's (1948) and Stuart's (1955) tests.

Note that, conditional on the elements of B, the elements of $B^{-1/2}m$ are asymptotically independent standard normal variables, under the assumption of symmetry, implying that this is also the unconditional distribution. Accordingly,

$$S_S^2 = m^TB^{-1/2}B^{1/2}C^TV^{-1}CB^{1/2}B^{-1/2}m$$

is distributed asymptotically as χ_{r-1}^2, where $B^{1/2}C^TV^{-1}CB^{1/2}$ is a projection matrix of rank $r-1$, as can be seen directly by verifying that $V = CBC^T$.
Consequently

$$\begin{aligned} S_I^2 &= S_B^2 - S_S^2 \\ &= m^TB^{-1/2}(I - B^{1/2}C^TV^{-1}CB^{1/2})B^{-1/2}m, \end{aligned}$$

which leads immediately to the following theorem.

Theorem 4.1. *Under the hypothesis of symmetry, H_{0B}, S_S^2 and S_I^2 are asymptotically distributed as independent chi-square variables with $r-1$ and $(r-1)(r-2)/2$ degrees of freedom, respectively. In addition S_S^2 is asymptotically distributed as a chi-square variable with $r-1$ degrees of freedom, under the null hypothesis of marginal symmetry H_{0S}.*

It is worth noting that the statistic for testing internal symmetry is

$$S_I^2 = m^T B^{-1} Q^T (Q B^{-1} Q^T)^{-1} Q B^{-1} m,$$

where $CQ^T = 0$. Accordingly, we must consider contrasts $QB^{-1}m$ to help interpret internal symmetry. In the case $r = 4$ we could take

$$Q = \begin{pmatrix} 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 1 \end{pmatrix}$$

We develop a function using the S-Plus and R packages to calculate the p-values of the previous three tests for all the pairs in a K edge tree.

4.5. Tests with Several Matched Observations

The simplest extension to K matched observations is to obtain tests for all pairs of observations as in the last three sections. Of course, as K increases this leads to problems of multiple comparisons, so we need to interpret p-values with some care. This simple approach enables us, however, to find observations that match on the basis of symmetry, marginal symmetry and internal symmetry. The p-values can be set out in a two-way table for all pairs, giving a useful method of grouping the observations, even though there are multiple comparison problems. This will be illustrated for nucleotide sequences later.

We may also wish to have an overall test for marginal symmetry. The null hypothesis is

$$H_{0S} : f_i^j = f_i, \; i = 1, \cdots, r, \; j = 1, \cdots, K.$$

Such a test was proposed by Rzhetsky and Nei (1995) for the analysis of nucleotide sequences. We will derive an equivalent test here and relate it to the tests for pairs given in the previous sections.

Consider the case when $K = 3$, which will have obvious extensions to any K. Let

$$\eta^T = (\eta_1^T, \eta_2^T, \eta_3^T),$$

where $\eta_j^T = (n_{j1}, \cdots, n_{jr})$ is the number of sites in the j-th sequence for which the variable takes a value $1, \cdots, r$. We can then write the expectation and covariance matrix of η as

$$\begin{aligned}\frac{1}{n}E(\eta) &= \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \\ &= f\end{aligned}$$

and

$$\begin{aligned}\frac{1}{n}Var(\eta) &= \begin{pmatrix} \ddot{D}_1 & F_{12} & F_{13} \\ F_{21} & \ddot{D}_2 & F_{23} \\ F_{31} & F_{32} & \ddot{D}_3 \end{pmatrix} - ff^T \\ &= V - ff^T,\end{aligned}$$

where

$$f_j^T = (f_1^j, ..., f_r^j),\ F_{jj'} = \sum^{*} \sum_{i_\ell=1}^{r} f_{i_1,\cdots,i_K},\ \text{and } \ddot{D}_j = diag(f_j),$$

where $\sum^{*}$ denotes summation over $\ell \neq j$ or j'.

Let

$$L = \begin{pmatrix} I_r & -I_r & 0_{rr} \\ I_r & 0_{rr} & -I_r \end{pmatrix}$$

and

$$H = \begin{pmatrix} I_{r-1} & \mathbf{0}_{r-1,1} & \mathbf{0}_{r-1,r-1} & \mathbf{0}_{r-1,1} \\ \mathbf{0}_{r-1,r-1} & \mathbf{0}_{r-1,1} & I_{r-1} & \mathbf{0}_{r-1,1} \end{pmatrix}$$

where I_h, is an $h \times h$ unit matrix and $\mathbf{0}_{h,K}$ denotes a $h \times K$ matrix of zeros. Now put $d = HL\eta$; multiplication by L compares sequences 1 and 2 and sequences 1 and 3, while multiplication by H selects the first $r-1$ values, thus giving exactly $2(r-1)$ contrasts which have a covariance matrix of full rank. This generalizes d of (4.1).

We consider the hypothesis

$$H_{0S} : f_1 = f_2 = f_3.$$

Under H_{0S},

$$\frac{1}{n} E(HL\eta) = E(d) = \mathbf{0}$$

and

$$\frac{1}{n} Var(HL\eta) = \frac{1}{n} Var(d) = HLVL^T H^T.$$

V can be estimated by $\hat{V}$, which can be obtained by replacing f_j by $\hat{f}_j = \eta_j/n$ and $F_{jj'}$ by $\hat{F}_{jj'} = N_{jj'}/n$, where $N_{jj'}$ is the observed $r \times r$ matrix of observations for each pair of sequences j and j'. Then, to test H_{0S} we can use the statistic

$$T_s = d^T (HL\hat{V}L^T H^T)^{-1} d/n$$

Under H_{0S}, this is asymptotically distributed as a $\chi^2_{(r-1)(K-1)}$ variate. Equivalently, we can use the computationally simpler,

$$T_s = \eta^T L^T (L\hat{V}L^T + J_{2,r})^{-1} L\eta$$

where

$$J_{2,r} = \begin{pmatrix} J_r & \mathbf{0} \\ \mathbf{0} & J_r \end{pmatrix}$$

for $J_r = 1_r 1_r^T$ and 1_r is a vector of length r with all elements 1.

The method developed here is described in general terms and for a general purpose, but can be used to analyse molecular sequence data. The method differs slightly from that of Rzhetsky and Nei (1995) by estimating the covariance matrix under H_{0S} instead of estimating it under the general model. For the case $K = 2$, Rzhetsky and Nei's (1995) test statistic is that of Bhapkar (1969), while the test statistic considered here is just that of Stuart (1955).

We have developed a function using the S-Plus and R packages to calculate the p-value of the previous test for any tree with K matched nucleotide sequences.

We now finish this chapter by giving some examples that show how these tests are working and why they are useful for the estimating procedure.

4.6. Analysis of Simulated Nucleotide Sequences

Example 4.1. Consider two matched sequences generated under the model $P(t) = e^{Rt}$ and (2.28) with the same time-independent rate matrix

$$R = \begin{pmatrix} -0.6 & 0.2 & 0.2 & 0.2 \\ 0.2 & -0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & -0.6 & 0.2 \\ 0.2 & 0.2 & 0.2 & -0.6 \end{pmatrix},$$

which implies that $\pi^T = (0.25, 0.25, 0.25, 0.25)$ is the stationary distribution of the process, but with $f_0 = (0.2, 0.2, 0.2, 0.4)^T$, and with $t_1 = t_2 = 1$. If we simulate the evolution of two nucleotide sequences of length 1000 using any of the methods in the previous chapter, then we can obtain the observed divergence matrix N and apply the tests. Doing so 1000 times, we obtained p-values for all tests that are uniformly distributed on $(0, 1)$, as expected, illustrating that in the case of homogeneity (i.e., $R^A = R^B$) the tests do not indicate the lack of stationarity, which was obviously present in this simulated data.

Example 4.2. Consider the simplest case of non-homogeneity. If $R_2 = \rho R_1$, $t_1 = t_2 = 1$, but $f_0 \neq \pi$, as in the previous example, then we might expect the test for marginal symmetry to indicate lack of symmetry and the test for internal symmetry to give no evidence of an effect. This indeed occurred: with a simulation taking parameters as in the first example, with $R_A = R$ and $R_B = \rho R$, giving uniform p-values for the test for internal symmetry but having 60% and 90% of p-values less than 0.05 in the test for marginal symmetry, for ρ equal to 3 and 5, respectively. This shows that the test for marginal symmetry can detect lack of stationarity when sequences have evolved under non-homogeneous conditions e.g., when $(R_2 = \rho R_1)$.

Example 4.3. Consider a model under which the test for marginal symmetry is not significant but the test for internal symmetry shows significant differences. For simplicity we will consider the general time-reversible Markov model for which ΠR is symmetric, and again we will take $t_1 = t_2 = 1$. Consider the spectral decomposition

$$\Pi^{1/2} R \Pi^{-1/2} = \sum_{j=1}^{4} \lambda_j u_j u_j^T = U \Lambda U^T$$

where $\Lambda = diag(\lambda_1, \cdots, \lambda_4)$. By taking unequal values of $\lambda_1, \cdots, \lambda_4$ and different u_j for A and B (i.e., the two sequences), while keeping stationarity ($f_0 = \pi_A = \pi_B$), we achieve a suitable model. We took

$$f_0 = (0.25, 0.25, 0.25, 0.25)^T,$$

$$\lambda_1 = 0, \lambda_2 = 5, \lambda_3 = 3, \lambda_4 = 2,$$

$$u_{A1}^T = (0.5, 0.5, 0.5, 0.5),$$

$$u_{A2}^T = (1, -1, 0, 0)/\sqrt{2},$$

$$u_{A3}^T = (1, 1, -2, 0)/\sqrt{6},$$

$$u_{A4}^T = (1, 1, 1, -3)/\sqrt{12},$$

and

$$u_{B1}^T = (0.5, 0.5, 0.5, 0.5),$$

$$u_{B2}^T = (1, 0, 0, -1)/\sqrt{2},$$

$$u_{B3}^T = (1, 0, -2, 1)/\sqrt{6},$$

$$u_{B4}^T = (1, -3, 1, 1)/\sqrt{12}.$$

Using this and so obtaining $F(t)$ from (2.28), and then getting 1000 simulations of matched sequences of length 1000, we obtained p-values for the test for marginal symmetry, which were uniform, as expected; on the other hand, we obtained 14% of p-values less than 0.05 for the test for internal symmetry. We then increased λ_2 to 10, 15 and 20, and obtained 55%, 68%, and 78% p-values less than 0.05, respectively. This illustrates that the test for internal symmetry measures divergence from symmetry in addition to that which might be due to marginal symmetry.

In the previous three examples we apply our test on a two edge tree. Now we will discuss different models for larger trees.

Example 4.4. Consider the case of non-stationarity and homogeneity for a tree of five matched sequences. Consider a model for a tree corresponding to the "merge" matrix

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ -3 & 1 \\ 2 & 3 \end{bmatrix}. \tag{4.2}$$

We use the heights $(0.1, 0.5, 0.8, .1)$, corresponding to the internal nodes represented by the rows of the previous merge matrix. Take

$$\pi = \pi_X = \pi_Y = (0.1, 0.1, 0.1, 0.7)^T,$$

$f_0^T = (0.25, 0.25, 0.25, 0.25)$ and the s-vector $= (0.2, 0.2, 0.2, 0.2, 0.2, 0.2)$. Let $R^X = R^Y = S\Pi$, where S is calculated using (2.13). Now, using any of the simulated methods described in Chapter 3, take, for example, the multinomial method, we generate simulated data set of size $n = 1000$ with 1000 replications from this model. We apply the overall test using the statistic T_s in section 4.5 as well as the matched pairs tests of homogeneity on all the 10 pairs of sequences. We can see that all p-values were uniformly distributed, as seen in Table 4.1, which shows the percentage of p-values from Stuart's test less than 0.05.

The other two matched pairs tests gave similar uniformly distributed results whereas the overall test T_s gave a mean and standard deviation of 11.9 and 4.7, respectively, compared to the mean and standard deviation of a χ^2_{12} variate of 12 and 4.9. These results imply that the test of marginal symmetry is unable to detect that the sequences have

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	4.2			
Seq 3	5.1	4.4		
Seq 4	5.1	4.9	4.5	
Seq 5	4.4	5.3	4.8	5.2

TABLE 4.1. Percentage of p-values of Stuart's (1955) test less than 0.05 - based on simulated data

evolved under non-stationary conditions, when the evolutionary processes otherwise are homogeneous. We obtained the same test results for the case of stationary and homogeneous processes.

Example 4.5. In this example consider the case of non-stationarity and non-homogeneity for the same five edge tree described in equation (4.2), define π and f_0 as in the previous example but in this case let $R^X = S\Pi$, $R^Y = \rho S\Pi$, and $\rho = 2$ where the S matrix and the height vector are defined as in the previous example. Simulating data set of size $n = 1000$ with 1000 replications under these variables, we obtained the result in Table 4.2, which shows the percentage of p-values from the test for marginal symmetry that were less than 0.05. The values of T_s for this case led to a mean and standard deviation of 41 and 11, respectively.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	5.2			
Seq 3	5.4	4.9		
Seq 4	94.6	95.5	95.6	
Seq 5	94.8	94.6	96.2	5.5

TABLE 4.2. Percentage of p-values of Stuart's (1955) test less than 0.05 - based on simulated data

From Table 4.2, we can easily see that the processes leading to nodes $-1, -2$ and -3 are homogeneous and the processes leading to nodes -4 and -5 are homogeneous.

Example 4.6. Consider the case of non-stationarity and non-homogeneity for the tree corresponding to the merge matrix (4.2). Now, assume that the non-homogeneity came from having different π for each side of the tree root, that is take

$$\pi_X^T = (.2, .2, .2, .4),\ \pi_Y^T = (.3, .3, .3, .1),\ f_0^T = (0.25, 0.25, 0.25, 0.25).$$

and s-vector $= (0.2, 0.2, 0.2, 0.2, 0.2, 0.2)$. Simulating 1000 times with a sample size $n = 1000$, we obtained the result in Table 4.3. The values of T_s for this case led to a mean and standard deviation of 47 and 12, respectively.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	4.7			
Seq 3	5.4	5.2		
Seq 4	98.5	97.5	96.9	
Seq 5	97.1	98.6	99.0	5.0

Table 4.3. Percentage of p-values of Stuart's (1955) test less than 0.05 - based on simulated data

Again, the results (Table 4.3) show that the processes leading to nodes $-1, -2$ and -3 are homogeneous and the processes leading to nodes -4 and -5 are homogeneous.

From the last two examples, we can see that the paired test for marginal symmetry cannot distinguish between the case when the lack of homogeneity came from first having different rate matrices for each side of the tree, that is $S_X \neq S_Y$, second having different π for each side of the tree, that is $\pi_X \neq \pi_Y$ or from both.

4.7. Analysis of Real Nucleotide Sequences

Molecular evolution of real data, that is nucleotide sequences from a set of genomes, may or may not have evolved under stationary and homogeneous conditions. We test two data sets by applying the three matched pairs tests of homogeneity and the overall test of marginal symmetry.

Example 4.7. Gaultier and Guoy (1995) inferred a phylogeny among five eubacterial species using the small-subunit ribosomal RNA sequences from *Aquafix pyrophilus, Thermotoga maritima, Thermus thermophilus, Deinococcus radiodurans,* and a fifth species chosen from the following genera: *Chlamydia, Spirochaeta, Bacterides, Agrobacterium, Escherichia, Fusobacterium, Clostridium, Bacillus, Micrococcus,* and *Anabaena.* They used a nucleotide substitution model that assumes that $\pi_A = \pi_T$ and $\pi_C = \pi_G$ whereas $\pi_C + \pi_G$ was allowed to vary across the tree; hence, they used a non-stationary and non-homogeneous model to infer their eubacterial phylogeny.

To illustrate the use of the matched pairs test of symmetry, we have used essentially the same data, except that the fifth species was represented by *Bacillus subtilis.* The alignment consists of 1238 matched sites from five species. The overall test for marginal symmetry based on T_s from Section 4.5 was applied giving an observed value 108.6 comparing to χ^2_{12}, indicating a significantly large deviation from marginal symmetry ($p \leq 0.0001$). More information was obtained by using the pair-wise tests of symmetry, marginal symmetry and internal symmetry, which gave the p-values shown in Table 4.4.

It is clear that all divergence matrices for the set *Aquifex, Thermus, and Thermotoga* show symmetry, as does the divergence matrix for *Bacillus* and *Deinococcus*, but all divergence matrices for pairs between these sets are highly asymmetric. Further there is no indication of

	Test	A	B	D	Ts
B	Bowker	0.000			
	Stuart	0.000			
	Int Sym	0.295			
D	Bowker	0.000	0.995		
	Stuart	0.000	0.946		
	Int Sym	0.754	0.958		
Ts	Bowker	0.509	0.000	0.000	
	Stuart	0.731	0.000	0.000	
	Int Sym	0.263	0.544	0.863	
Ta	Bowker	0.132	0.000	0.000	0.415
	Stuart	0.325	0.000	0.000	0.267
	Int Sym	0.095	0.417	0.297	0.546

TABLE 4.4. Tests of bacterial data - A: *Aquifex*, B: *Bacillus*, D: *Deinococcus*, Ts: *Thermus*, Ta: *Thermotoga*

differences in internal symmetry. The simplest model for which this outcome for the tests would be expected must satisfy

- lack of stationary;
- all terminal edges to *Aquifex, Thermus and Thermotoga* have the same rate matrix R_1 whereas the terminal edges to *Bacillus* and *Deinococcus* have the same rate matrix R_2;
- $R_1 \neq R_2$.

We can present the third condition here in a simpler form if we take $R_1 = S_1\Pi_1$ and $R_2 = \rho S_2\Pi_2$, where either $\rho \neq 1$ or $\Pi_1 \neq \Pi_2$, and Π_1 and Π_2 are diagonal matrices with the stationary distributions of R_1 and R_2, respectively. Consideration of this simple form using the same s-vector (off-diagonal elements in S_1 and S_2) for both R_1 and R_2 has support because the test for internal symmetry was not significant, although such an assumption is not strictly justified. We might also take S_1 and S_2 to be symmetric, making the process on each edge reversible under stationarity, although the tests do not give information on this.

Example 4.8. We will give here simulated data having the same structure as the bacterial data and apply our test on it. Consider the case

	Test	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	Bowker	0.1154			
	Stuart	0.1093			
	Int Sym	0.2429			
Seq 3	Bowker	0.7658	0.8958		
	Stuart	0.6704	0.9901		
	Int Sym	0.6186	0.5456		
Seq 4	Bowker	0.0000	0.0000	0.0000	
	Stuart	0.0000	0.0000	0.0000	
	Int Sym	0.4169	0.0978	0.6611	
Seq 5	Bowker	0.0000	0.0000	0.0000	0.2800
	Stuart	0.0000	0.0000	0.0000	0.4488
	Int Sym	0.9677	0.5725	0.7915	0.1859

TABLE 4.5. Tests of simulated sequences.

of non-stationarity and non-homogeneity for a five edge tree described by the merge matrix (4.2), by taking

$$\pi_X^T = (0.1, 0.1, 0.1, 0.7),\ \pi_Y^T = (0.3, 0.3, 0.3, 0.1),$$

$f_0^T = (0.25, 0.25, 0.25, 0.25)$ and the s-vector $= (0.2, 0.2, 0.2, 0.2, 0.2, 0.2)$. Let $R^X = S_X\Pi_X$ and $R^Y = \rho S_Y\Pi_Y$, where $\rho = 3$, S_X and S_Y are calculated using (2.13). Now simulate data of size $n = 1000$ and applying our tests on it gave us the result in the Table 4.5.
From Table 4.5, we can easily see that all the divergence matrices for the external nodes set 1, 2 and 3 show symmetry, as does the divergence matrices for the external nodes set 4 and 5, but the divergence matrices for pairs between these sets are highly asymmetric. Also there is no

indication of differences in internal symmetry. So all the results give us what we expected, since we generate the data under non-stationary and non-homogeneous processes. Note that these results do not address the question of whether the bacterial phylogeny informed by Galtier Gouy (1995) is right. It merely shows that we can generate data that have similar properties to the real data.

Example 4.9. We consider an alignment of 1206 nucleotides from the mitochondrially-encoded NADH dehydrogenase submit 5 genes of: *Human*, *Chimpanzee*, *Bonobo*, *Gorilla*, *Orangutan*, *Gibbon* and *Macaque*. The overall test for marginal symmetry T_s was applied giving a non-significant observed value 25.35 comparing to χ^2_{18}. Also, the matched pairs tests of homogeneity were applied and gave the p-values shown in Table 4.6.

Its clear that none of the three tests indicate a lack of symmetry, which indicates that the model for this data may satisfy stationarity and homogeneity assumptions.

4.8. Discussion

The tests presented in this chapter assume that the individual observations are independently and identically distributed. It is possible to weaken this condition in two ways. First, it is not necessary to assume that results from the sequences are independent, but instead we can assume that, conditional on the values at the root, the processes on the two branches evolve independently. If the individual observations do not evolve independently, then the effective sample size will be smaller, and the tests will not be appropriate, since the test statistics will not then have the given asymptotic distributions. Second, we may consider a model in which certain sites are invariant, in the sense that the value of the nucleotide taken at the root must remain unchanged. In this case we simply change values of $n_{1,\cdots,1}, \cdots, n_{4,\cdots,4}$, which does not affect any of the test statistics S_B^2, S_S^2 , S_I^2 or T_s, considered here, although it does make asymptotically negligible changes to the statistics

	Test	Hu	Bo	Ch	Go	Or	Gi
Bo	Bowker	0.065					
	Stuart	0.101					
	Int Sym	0.131					
Ch	Bowker	0.523	0.251				
	Stuart	0.722	0.306				
	Int Sym	0.280	0.239				
Go	Bowker	0.313	0.148	0.779			
	Stuart	0.417	0.108	0.488			
	Int Sym	0.236	0.332	0.849			
Or	Bowker	0.362	0.020	0.142	0.102		
	Stuart	0.256	0.022	0.070	0.039		
	Int Sym	0.471	0.142	0.465	0.531		
Gi	Bowker	0.188	0.159	0.583	0.103	0.344	
	Stuart	0.420	0.200	0.324	0.041	0.264	
	Int Sym	0.115	0.201	0.747	0.520	0.427	
Ma	Bowker	0.327	0.513	0.640	0.295	0.244	0.725
	Stuart	0.326	0.274	0.412	0.105	0.096	0.682
	Int Sym	0.325	0.716	0.706	0.767	0.669	0.544

TABLE 4.6. Tests of hominoid data - Hu: *Human*, Bo: *Bonobo*, Ch: *Chimpanzee*, Go: *Gorilla*, Or: *Orangutan*, Gi: *Gibbon*, Ma: *Macaque*

of Bhapkar (1966) and Rzhetsky and Nei (1995). More generally, if the site evolved independently under the same stationary condition but under different homogeneous models (i.e. rate heterogeneity across sites), then the joint distribution would be a mixture of the joint distributions at each site, but would retain the symmetries of the probabilities at each site. So, the tests here would retain their properties of testing for symmetry. We note that consistency with the hypotheses of symmetry does not imply stationarity and homogeneity, but only that the data is consistent with such hypotheses. For example, if the stationary distributions of different sites differ and at each site the substitution process is stationary, then the hypotheses of symmetry will still hold, so the tests have no power to detect such differences. The problems of lack of independence of evolution and rate heterogeneity across sites may be mitigated by partitioning according to codon position, so it is recommended that sequence data be partitioned into appropriate bins before conducting the tests.

Chapter 5

Phylogenetic Distance

5.1. Introduction

There are several methods for constructing phylogenetic trees. In some of these methods a distance matrix is used to construct a phylogenetic tree by joining the sequences having the smallest distance between them. For every pair of sequences in the multiple sequence alignment, distance matrix methods calculate an estimate of the evolutionary distances separating the sequences. The evolutionary distances here are the product of time and the rate of evolution (see Felsenstein, 1996). The evolutionary tree chosen is then the one that makes the best prediction given these pairwise distances based on some criterion.

We will not give a comprehensive view of the distance methods in phylogenetics but will instead give some definitions and discuss the paralinear distance and hierarchical clustering as we will use them. Other methods will only be reviewed as they relate to our technique.

Definition 5.1. *Let X be a set and let D be a real function from pairs of elements of X. We say that D is a distance function on X if*

1: $D_{uv} > 0$ *for all* $u, v \in X, u \neq v$,
2: $D_{uu} = 0$ *for all* $u \in X$,
3: $D_{uv} = D_{vu}$ *for all* $u, v \in X$,
4: *The triangle inequality holds:*
$D_{uv} \leq D_{uw} + D_{wv}$ *for all* $u, v, w \in X$.

We would like distances to have the additivity and ultrametric properties.

Definition 5.2. *A distance matrix D is additive if there exists a tree with positive edge weights such that $D_{rs} = \sum_{e \in h_{rs}} w(e)$, where h_{rs} denotes the path between leaves r and s in the tree, and $w(e)$ is the weight of edge e in the path h_{rs}.*

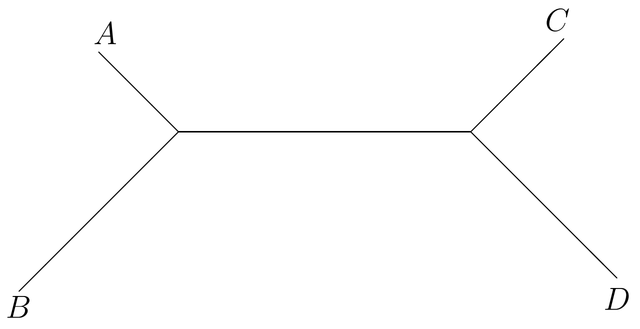


Figure 6. Additive tree

The evolutionary distance between each pair of sequences would be equal to the sum of the lengths of each edge lying on the path between the members of each pair. Mathematically, additive distances satisfy the four point condition for any four taxa A, B, C and D,

$$D_{AB} + D_{CD} \leq max(D_{AC} + D_{BD}, D_{AD} + D_{BC}),$$

where D_{rs} is the distance between taxa r and s, and "max" is the maximum value function (see Figure 6).

Ultrametric distances are more constrained than tree-additive distances. Mathematically, the ultrametric distances satisfy the inequality that for any three taxa A, B and C,

$$D_{AC} \leq max(D_{AB},\ D_{BC}). \tag{5.1}$$

For this inequality to be true, two of the tree pairwise distances must be equal and at least as large as the third (see Figure 7). Phylogenetically, ultrametric distances will precisely fit a tree so that the distance between any two taxa is equal to the sum of the branches joining them, and the tree can be rooted so that all of the taxa are equidistant from the root (see Hillis, 1996). The rooted trees that we will discuss assume every common ancestor is equidistant from all its descendants. Hence, if an ultrametric distance measure between species is given, then there is a unique rooted tree joining these species that gives these distances.

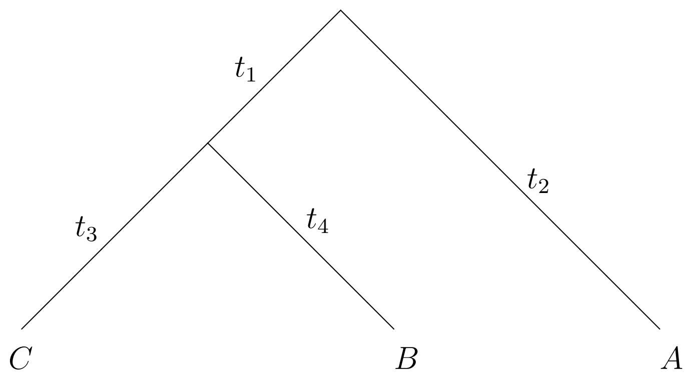


Figure 7. Additive and ultrametric tree

The tree in Figure 7 satisfies the additive properties:

$$D_{AB} = t_1 + t_2 + t_4$$
$$D_{AC} = t_1 + t_2 + t_3$$
$$D_{BC} = t_3 + t_4,$$

and the ultrametric properties:

$$t_3 = t_4$$
$$t_2 = t_1 + t_3 = t_1 + t_4,$$

where D_{rs} is the distance between taxa r and s.

5.2. Measuring Distances from Sequence Data

Distance methods attempt to estimate the mean number of changes per site between two sequences since their divergence. Counting the number of differences may not fully approximate the true amount of divergence, especially when multiple substitutions have occurred at the same site. Hence, we are interested in the case of a distance function on a finite set $X = (X_1, X_2, ..., X_K)$ of nucleotide sequences for which we would like to build a phylogenetic tree.

Many distance methods have been created with the aim of correcting the observed distances by estimating the actual amount of substitutions that have occurred through time. One of the first substitution models used in the estimation of evolutionary distance is the one of Jukes and Cantor (1969). The estimated distance becomes

$$D_{rs} = -\frac{3}{4}\, ln \left(1 - \frac{4}{3} G_{rs}\right), r, s = 1, \cdots, K$$

where G_{rs} is the proportion of pairs of sites in sequences r and s, which have different nucleotides, (see, for example Tavaré, 1986).

Lanave et al. (1984) developed the general time-reversible Markov model, which considers six conditional rates of change (a symmetric S matrix) as well as non-uniform nucleotide content. This is the most general model of the time-reversible Markov models of nucleotide substitution.

The paralinear distance is an even more general measure of the distance between two sequences. It was designed to deal with variable nucleotide frequencies in each pair-wise sequence comparison and is based on the general Markov model of evolution discussed in Chapter 2. The paralinear distance is an additive distance between sequences under very general assumptions, is easy to calculate, and uses data efficiently. Lake (1994), defined the paralinear distance, $\hat{D}_{rs}$, between two sequences r and s as:

$$\hat{D}_{rs} = -log_e \frac{det(N_{rs})}{(det(N_r))^{1/2}(det(N_s))^{1/2}},\ r, s = 1, \cdots, K \qquad (5.2)$$

where $\hat{D}_{rs}$ is the estimated distance between the two sequences r and s, log is the natural logarithm function, det is the determinant of a matrix, N_{rs} is the observed matrix for the sequences r and s described in Chapter 3, N_r and N_s are the diagonal matrices, estimates of the nucleotide compositions of sequences r and s, respectively, and can be constructed from the N_{rs} matrix. For each pair of sequences, r and s, a 4×4 matrix N_{rs} is constructed, which contains the observed frequencies for every possible pair of nucleotides. It is basically a summary of the relative frequencies of bases between two taxa. An equivalent general distance was developed by Steel (1993) and Lockhart et al (1994).

From the theoretical joint distribution function $F(t)$ described in Chapter 2, the paralinear distance associated with sequences of infinite lengths can be written as

$$D_{rs} = -log(det(F_r^{-1/2} F_{ij} F_s^{-1/2})),\ r, s = 1, \cdots, K. \tag{5.3}$$

Now let

$$P_{rs} = F_r^{-1} F_{rs},\ r, s = 1, \cdots, K$$

where P_{rs} is the transition matrix from sequence r to sequence s, and

$$P_{sr} = (F_{rs} F_s^{-1})^T,\ r, s = 1, \cdots, K$$

where P_{sr} is the transition matrix from sequence s to sequence r, then the paralinear distance between the two sequences r and s can be written as

$$D_{rs} = -\frac{1}{2}(log(|det(P_{rs})|) + log(|det(P_{sr})|)),\ r, s = 1, \cdots, K. \tag{5.4}$$

We can define distances between all nodes in a similar way, where we have Markov processes generating the joint distribution as in Chapter 2. We have written a function to calculate the paralinear distance between K matched sequences in the R package. The function depends on the joint distribution function of all pairs of the K matched sequences. We will use only the paralinear distance throughout the remainder of this chapter.

5.3. Generating Trees from Distances

Sokal and Michener (1958) introduced the Unweighted Pair Group Method with Arithmetic Average (UPGMA) method, which assumes that the sequences diverged at a constant rate and the distance between them met the additive property. In the UPGMA method, phylogenetic trees are constructed by combining the two closest external nodes and recalculating new distance measures for the new grouping. If distances are ultrametric, then the UPGMA method will produce an ultrametric tree.

Saitou and Nei (1987) developed a neighbor-joining method, whereby pairs of external nodes are identified as neighbours so that the tree with minimum total length is obtained. If the distances are additive, then this method will produce the correct unrooted tree.

The hierarchical clustering method depends on the distance matrix for a set of taxa. In the statistical package R, the hierarchical clustering function, `hclust`, " $\cdots$ performs a hierarchical cluster analysis using a set of dissimilarities for the n objects being clustered. Initially, each object is assigned to its own cluster and then the algorithm proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster".

We will use the average method "ave", where the distance between clusters is the average of the distances between the points in one cluster and the points in the other cluster. The result of the hierarchical clustering function `hclust` gives a merge matrix describing the tree topology and a vector of heights describing the lengths between each of the external nodes and the first node connecting it to another node. In the R package the hierarchical clustering function `hclust` is given as

$$\texttt{hclust}(as.dist(D),\ method = \text{``}ave\text{''}),$$

where $as.dist(D)$ gives the lower triangle of the distance matrix D and *"ave"* is the method of connection between the taxa.

Note that this method is equivalent to the UPGMA method of Sokal and Michener (1958).

5.4. Analysis Based on the Joint Distribution Function and Simulated Nucleotide Sequences

In this section, we give examples of how the paralinear distance and the hierarchical clustering function work under different assumptions. We will illustrate by examples that when we have stationary and homogeneous processes over the tree then the hierarchical clustering methods produce a correct tree topology and edge lengths. In other cases we may get an incorrect tree topology or incorrect edge lengths. The examples depend on the joint distribution functions $F(t)$ calculated using the gn program, except for Example 5.7, where we use the gn2 program. Both gn and gn2 programs were described in Chapter 2.

Example 5.1. Consider the case of stationary and homogeneous processes for a tree with five matched sequences, described by the merge matrix (4.2). Let

$$\pi = f_0 = (0.25,\ 0.25,\ 0.25,\ 0.25)^T$$

and $R^X = R^Y = S\Pi$, where the S matrix is defined as in equation (3.1). Take the height vector to be (0.1, 0.5, 0.8, 1.0). From the joint distribution function array $F(t)$ for this model, we can find the distance matrix D for the five matched sequences using the paralinear distance (5.3), which gives the result shown in Table 5.1.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.12			
Seq 3	0.96	0.96		
Seq 4	1.20	1.20	1.20	
Seq 5	1.20	1.20	1.20	0.6

Table 5.1. Paralinear distance matrix for $F(t)$ under stationary and homogeneous assumptions

Dividing the distance matrix by half the largest distance between the external nodes, forces the total distance from the root to any external node to be 1. The largest distance between the external nodes and the root in the case of stationary, homogeneous and reversible processes is equal to $\Re = -\sum_{i=1}^{4} R_{ii}\pi_i$, where the matrix R was defined in Chapter 2. Now, dividing the distance matrix in Table 5.1 by $\Re = 0.6$ gives the exact distances between the five leaves, see Table 5.2.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.2			
Seq 3	1.6	1.6		
Seq 4	2.0	2.0	2.0	
Seq 5	2.0	2.0	2.0	1

TABLE 5.2. Paralinear distance matrix for $F(t)/.6$ under stationary and homogeneous assumptions

Applying the `hclust` function on the distance matrix, gives the exact shape of tree described by the merge matrix shown below

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ -3 & 1 \\ 2 & 3 \end{bmatrix},$$

and a vector of heights,

$$heights = (0.12,\ 0.60,\ 0.96,\ 1.20).$$

Forcing the largest height of the tree to be 1, by dividing the vector of heights by the largest height, 1.2, gives the following vector of heights,

$$heights = (0.1,\ 0.5,\ 0.8,\ 1.0),$$

which is the exact vector of heights we used to calculate the joint distribution function. From the distance matrix in Table 5.2 and the merge matrix we get from the `hclust` function, we can find the phylogenetic tree, as we can see in Figure 8.

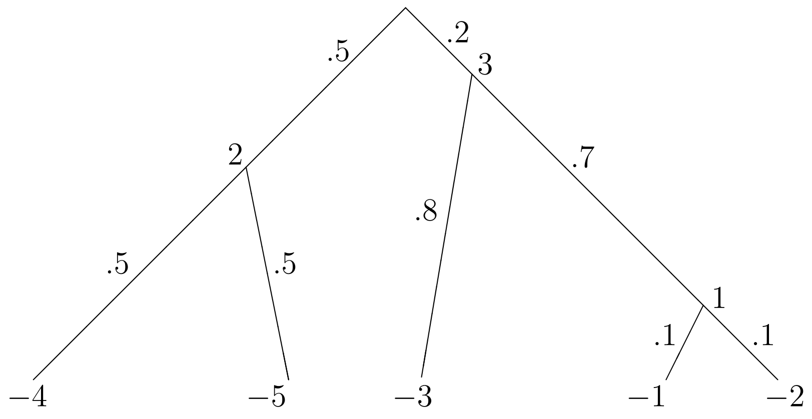


Figure 8. Ultrametric tree

We can see that the tree satisfies the properties of an additive and ultrametric tree, since the distance from the root to any external tips is equal to one and also it satisfies the equations:

$$D_{-1-4} = D_{-1-5} = D_{-2-4} = D_{-2-5} = D_{-3-4} = D_{-3-5} = 2$$

and

$$D_{-1-3} = D_{-2-3} = 1.6,$$

so we can say, that in the case of stationary and homogeneous processes, the distances gave the exact topology of the tree and the exact distances, except for a multiplier.

Example 5.2. Consider the same case of stationary and homogeneous processes for a tree with five matched sequences, as described in the previous example. Depending on the joint distribution function calculated in the previous example, we simulate a data set of size $n = 1000$ and find the observed divergence array N using any of the methods described in Chapter 3. Finding the distance matrix depending on the observed array N, gives the result shown in Table 5.3.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.08			
Seq 3	0.88	0.89		
Seq 4	1.06	1.09	1.13	
Seq 5	1.14	1.16	1.22	0.57

TABLE 5.3. Paralinear distance matrix for simulated data under stationary and homogeneous assumptions

Applying the `hclust` function on this distance matrix, gives the tree topology we used to generate the joint distribution function from where this data was simulated, and the following vector of heights,

$$heights = (0.08,\ 0.57,\ 0.89,\ 1.13).$$

Forcing the largest height of the tree to be 1, by dividing the vector of heights by the largest height, 1.13, gives the following vector of heights,

$$heights = (0.07,\ 0.50,\ 0.78,\ 1.00),$$

which is very close to the vector of heights used to calculate the joint distribution function.

Remark: In the rest of this chapter we will look only at the exact joint distributions rather than simulated ones. Knowing that the simulated data give us approximately the same result, as we can see in Example 5.2.

Example 5.3. Consider the case of stationary and non-homogeneous processes for the same tree described in the previous example. Let

$$\pi = f_0 = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,$$

$R^X = S\Pi$ and $R^Y = \rho S\Pi$, where the S matrix is defined as in equation (3.1). Take $\rho = 3$ and the heights vector to be (0.1, 0.5, 0.8, 1.0). Finding the distance matrix for the joint distribution function under this model gives the result in Table 5.4.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.36			
Seq 3	2.88	2.88		
Seq 4	2.40	2.40	2.40	
Seq 5	2.40	2.40	2.40	0.60

TABLE 5.4. Paralinear distance matrix for $F(t)$ under stationary and non-homogeneous assumptions

Applying the `hclust` function on this distance matrix, gives the merge matrix

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ 1 & 2 \\ -3 & 3 \end{bmatrix}, \tag{5.5}$$

and a vector of $heights = (0.36,\ 0.60,\ 2.40,\ 2.64)$. We can see that the distance matrix and the `hclust` function in this case (of stationarity and non-homogeneity, where non-homogeneity comes from different rate matrices for each side of the root) did not give the correct tree topology or edge lengths, (Figure 9).

We can see that the we do not get the same heights as the one we used to generate the data even when we divide the height vector by the maximum height.

Using the correct tree topology from which we generated the joint distribution, and the height vector from the `hclust` function after we divide it by the maximum height, we can find an ultrametric rooted tree, (Figure 10). In this example the true tree is not ultrametric. However, the `hclust` function gives an ultrametric tree fitting the data as closely as possible. We also have fitted an ultrametric tree using the heights from the `hclust` output and the true topology as given by the true merge matrix.

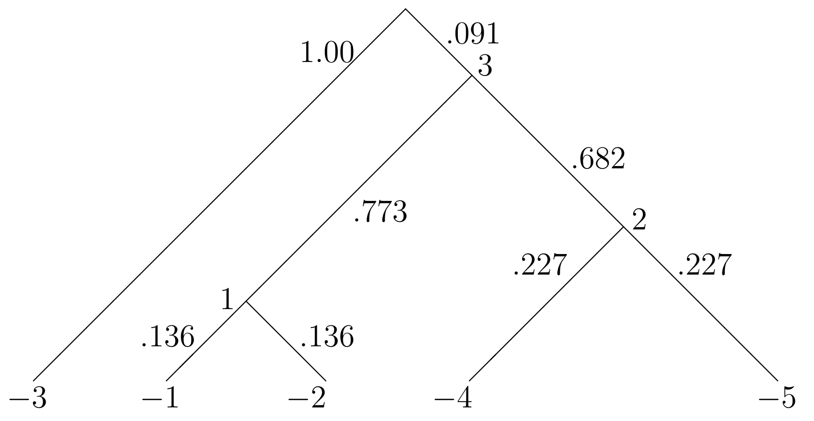


Figure 9. Ultrametric tree from the merge matrix (5.5) and the vector of heights after dividing it by 2.64

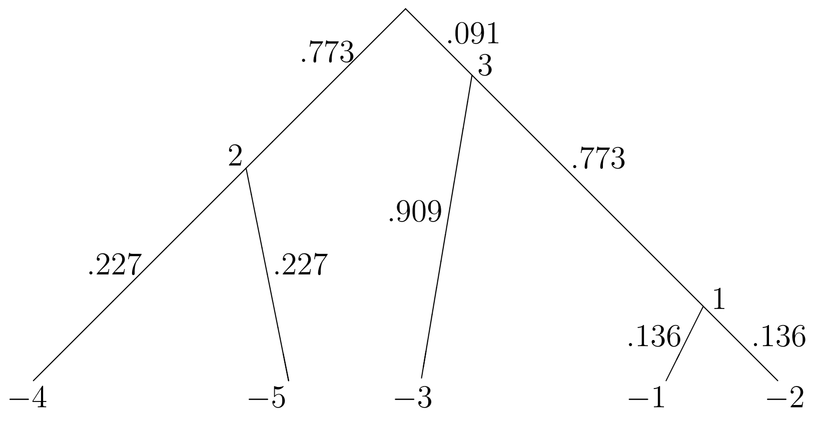


Figure 10. Ultrametric tree

We cannot find an ultrametric rooted tree depending on the merge matrix from the `hclust` function and the distance matrix in Table 5.4, which has distances corresponding to the original distances. However, we can find an additive rooted tree as we can see in Figure 11. Note that ε can be chosen arbitrarily between 0 and 1.44.

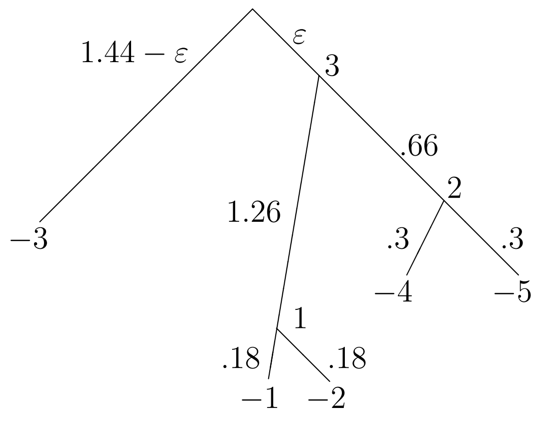


Figure 11. Additive tree

Example 5.4. Consider the case of non-stationarity and non-homogeneity for a five edge tree described in Example 5.1. Let the non-homogeneity come from having a different stationary distributions for each side of the root, that is

$$\pi_X = f_0 = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,\ \pi_Y = (0.1,\ 0.1,\ 0.1,\ 0.7)^T.$$

and the s-vector $= (0.2, 0.2, 0.2, 0.2, 0.2, 0.2)$. Take the heights vector as defined in Example 5.1, $R^X = S_X\Pi_X$ and $R^Y = S_Y\Pi_Y$, where S_X and S_Y are calculated using (2.13). Finding the distance matrix from the joint distribution of this model, gives the result shown in Table 5.5.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.11			
Seq 3	0.90	0.90		
Seq 4	1.17	1.17	1.17	
Seq 5	1.17	1.17	1.17	0.60

Table 5.5. Paralinear distance matrix for $F(t)$ under non-stationary and non-homogeneous case (1)

Applying the `hclust` function on the distance matrix in Table 5.5 gives the merge matrix

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ -3 & 1 \\ 2 & 3 \end{bmatrix}, \tag{5.6}$$

and a vector of *heights* = (0.11, 0.60, 0.90, 1.17). Note that this height vector and the distance matrix in Table 5.5 gives the same edge lengths. The tree describing by the merge matrix (5.6) and the distance matrix in Table 5.5 (after dividing it by the largest distance (1.17)) is shown in Figure 12.

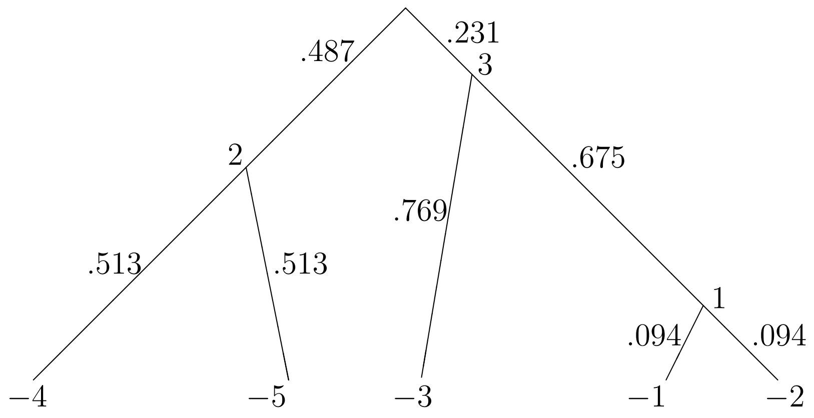


Figure 12. Ultrametric tree from the merge matrix (5.6) and the vector of heights after dividing it by 1.17

Also, notice that the distance matrix in Table 5.5 contains additive and ultrametric distances but they did not give the heights we used to calculate the joint distribution from even if we divide them by the largest distance between the root and the external nodes, see Figure 12.

We can see that the `hclust` function under this case (of non-stationary and non-homogeneity, where the non-homogeneity came from different stationary distributions for each side of the root) gives the exact tree topology but this is not always the case. We can see this by taking

$$\pi_X = f_0 = (0.05,\ 0.05,\ 0.05,\ 0.85)^T \text{ and } \pi_Y = (0.3,\ 0.3,\ 0.3,\ 0.1)^T.$$

Finding the distance matrix from the joint distribution under this model, gives the result shown in Table 5.6.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.24			
Seq 3	2.22	2.22		
Seq 4	2.08	2.08	2.08	
Seq 5	2.08	2.08	2.08	0.60

TABLE 5.6. Paralinear distance matrix for $F(t)$ under non-stationary and non-homogeneous case (2)

Applying the `hclust` function on the distance matrix in Table 5.6 gives the merge matrix

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ 1 & 2 \\ -3 & 3 \end{bmatrix}, \tag{5.7}$$

and a vector of *heights* = (0.24, 0.60, 2.08, 2.15). Note that this height vector and the distance matrix in Table 5.6 gives different edge lengths. Using the the merge matrix (5.7) and the previous vector of heights after dividing it by the maximum length, we can find an ultrametric phylogenetic tree (Figure 13).

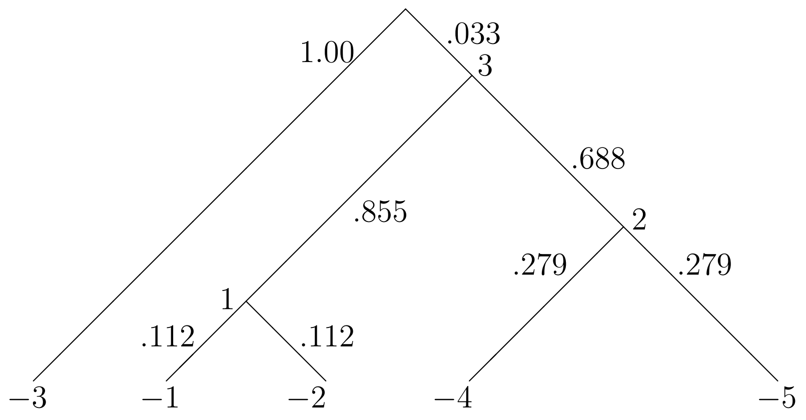


Figure 13. Ultrametric tree from the merge matrix (5.7) and the vector of heights after dividing it by 2.15

Example 5.5. Consider the case of non-stationary and homogeneous processes for the same tree topology as in the previous example. Let

$$\pi_X = \pi_Y = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,\ f_0 = (0.1,\ 0.1,\ 0.1,\ 0.7)^T,$$

and $R^X = R^Y = S\Pi_X$, define the S matrix and the vector of heights as in Example 5.1. Finding the distance matrix from the joint distribution of this model, gives the result shown in Table 5.7.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.17			
Seq 3	1.41	1.41		
Seq 4	1.80	1.80	1.80	
Seq 5	1.80	1.80	1.80	0.86

TABLE 5.7. Paralinear distance matrix for $F(t)$ under non-stationary and non-homogeneous assumptions

Applying the `hclust` function on this distance matrix, gives the merge matrix

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ -3 & 1 \\ 2 & 3 \end{bmatrix}, \tag{5.8}$$

and a vector of *heights* = (0.17, 0.86, 1.41, 1.80). Note that this height vector and the distance matrix in Table 5.7 gives the same edge lengths. The tree describing the merge matrix (5.8) and the distance matrix in Table 5.7 (after dividing it be the largest distance (1.80)) is shown in Figure 14.

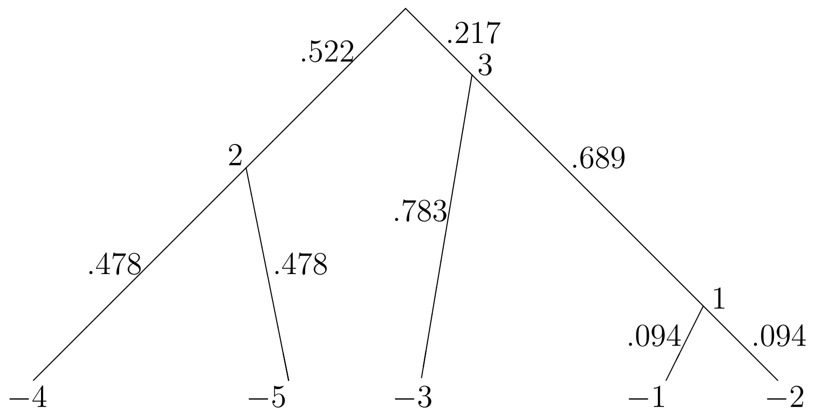


Figure 14. Ultrametric tree from the merge matrix (5.8) and the vector of heights after dividing it by 1.80

We can see that the `hclust` function under this case of non-stationary and homogeneity gives the exact tree topology but not the exact heights we used to generate the joint distribution array as we can see in Figure 14.

Example 5.6. Consider the case of non-stationary and non-homogeneous processes for the same tree topology as in the previous example. Let

$$\pi_X = f_0 = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,\ \pi_Y = (0.1,\ 0.1,\ 0.1,\ 0.7)^T,$$

$R^X = S_X \Pi_X$ and $R^Y = \rho S_Y \Pi_Y$, where $\rho = 3$, the s-vector and the height vector are defined as in Example 5.1 and S_X and S_Y are calculated using (2.13). Finding the distance matrix from the joint distribution of this model, gives the result shown in Table 5.8.

	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.31			
Seq 3	2.55	2.55		
Seq 4	2.22	2.22	2.22	
Seq 5	2.22	2.22	2.22	0.60

TABLE 5.8. Paralinear distance matrix for $F(t)$ under non-stationary and non-homogeneous assumptions

Applying the `hclust` function on the distance matrix in Table 5.8, gives the merge matrix

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ 1 & 2 \\ -3 & 3 \end{bmatrix}, \tag{5.9}$$

and the height vector, *heights* = (0.31, 0.60, 2.22, 2.39). Note that this height vector and the distance matrix in Table 5.8 with respect to the merge matrix (5.9), gives different edge lengths. The unrooted tree describing by the merge matrix (5.9) and the distance matrix in Table 5.8 is shown in Figure 15. Note that since the tree is not ultrametric we can take the root at any point on the edge of length 0.345

We can see that the `hclust` function under this case of non-stationary and non-homogeneity did not give the correct tree topology or heights we used to find the joint distribution array.

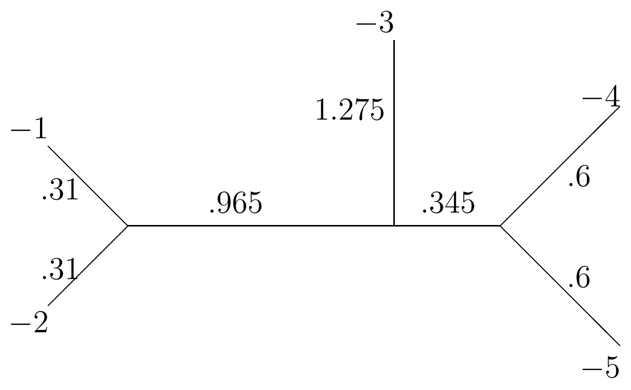


Figure 15. Additive tree

Using the the merge matrix (5.9) and the vector of heights we get from the `hclust` function (after dividing it by the maximum length) give an ultrametric phylogenetic tree as we can see in Figure 16.

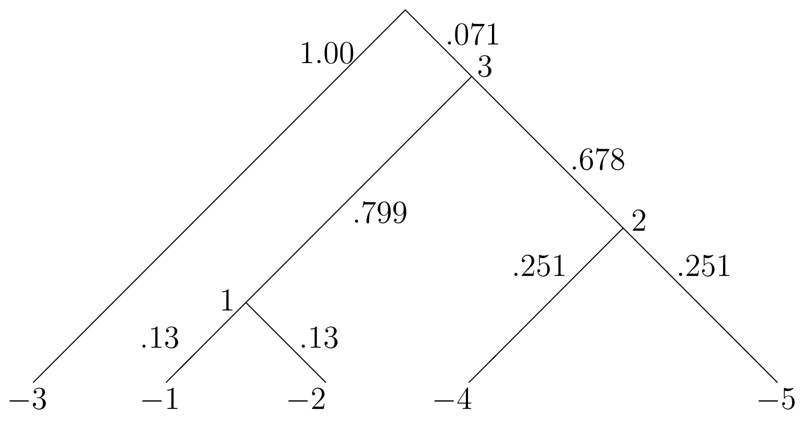


Figure 16. Ultrametric tree from the merge matrix (5.9) and the vector of heights after dividing it by 2.39

Remark: From the previous examples we can see that all the paralinear distances were additive, and the `hclust` function gives the correct tree topology under homogeneous and stationary or non-stationary models, where the non-stationarity came from $\pi_X = \pi_Y \neq f_0$.

Example 5.7. Consider the case of a non-stationary and non-homogeneous processes for a five leaf tree described by the merge matrix in (5.10). Let

$$\pi_X = \pi_Y = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,\ f_0 = (0.1,\ 0.1,\ 0.1,\ 0.7)^T.$$

Define the S matrix as in equation (3.1) and the matrix of rates, which was defined in Chapter 2 Section 2.6.2, as

$$rate = \begin{bmatrix} .1 & .2 \\ .8 & .4 \\ .5 & .6 \\ .7 & .3 \end{bmatrix} \quad merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ -3 & 1 \\ 2 & 3 \end{bmatrix}. \tag{5.10}$$

Finding the distance matrix for the joint distribution function under this case, gives Table 5.9.

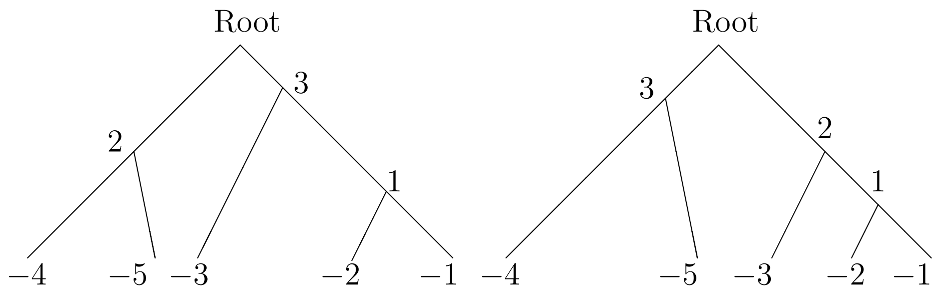
	Seq 1	Seq 2	Seq 3	Seq 4
Seq 2	0.24			
Seq 3	1.00	1.08		
Seq 4	2.12	2.20	2.12	
Seq 5	1.79	1.87	1.79	1.05

TABLE 5.9. Paralinear distance matrix for $F(t)$ under non-stationary and non-homogeneous assumptions

Applying the `hclust` function on the distance matrix in Table 5.9 gives the merge matrix

$$merge = \begin{bmatrix} -1 & -2 \\ -3 & 1 \\ -4 & -5 \\ 2 & 3 \end{bmatrix}, \tag{5.11}$$

and the height vector, $heights = (0.24,\ 1.04,\ 1.05,\ 1.98)$. Note that this height vector and the distance matrix in Table 5.9 with respect to the merge matrix (5.11), gives different edge lengths. We can see that the merge matrix (5.11) gives the same topology as the merge matrix in 5.10 (from which the joint distribution array was generated) except for the order of connection between the nodes as we can see in Figure 17.



A. Using the merge in (5.10) B. Using merge (5.11)

Figure 17. Phylogenetic tree

Using the distance matrix in Table 5.9 and the merge matrix (5.11) we can draw a rooted tree for these additive distances (Figure 18). Also, using the height vector after we divide it by the maximum height and the merge matrix (5.11) from the `hclust` function we can draw an ultrametric tree (Figure 19). Note that the trees in Figure 18 and 19 have the same unrooted topology.

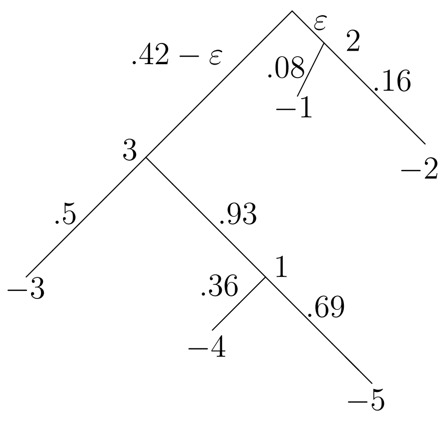


Figure 18. Additive tree

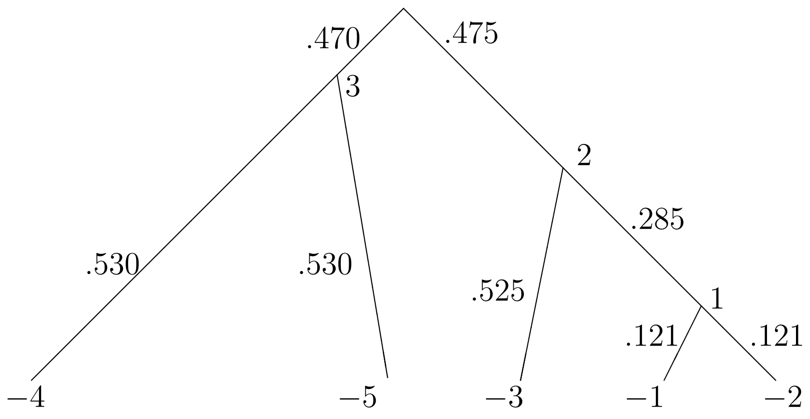


Figure 19. Ultrametric tree

In summary, these experiments show that it is necessary for the assumptions of stationary, reversibility and homogeneity to be met if the hierarchical clustering method is to be used.

5.5. Analysis Based on Real Nucleotide Sequences

In this section, we will apply the paralinear distance to two data sets of real nucleotide sequences.

Example 5.8. Consider the bacterial data described in Example 4.7, finding the distance matrix for this data gives the result in Table 5.10.

	A	B	D	Ts
B	0.972			
D	1.222	0.846		
Ts	0.867	0.889	0.767	
Ta	0.721	0.815	0.983	0.672

Table 5.10. Paralinear distance matrix for bacterial data -A: *Aquifex*, B: *Bacillus*, D: *Deinococcus*, Ts: *Thermus*, Ta: *Thermotoga*

Applying the `hclust` function on the distance matrix, gives the following merge matrix

$$merge = \begin{bmatrix} -4 & -5 \\ -1 & 1 \\ -2 & -3 \\ 2 & 3 \end{bmatrix} \tag{5.12}$$

and a vector of heights,

$$heights = (0.67,\ 0.79,\ 0.85,\ 0.94),$$

where node -1 represents *Aquifex*, -2 represents *Bacillus*, -3 represents *Deinococcus*, -4 represents *Thermus* and node -5 represents *Thermotoga*.

In Chapter 4 Example 4.7, we showed that the statistical test suggested that the set of *Aquifex, Thermus, and Thermotoga* are very close together and have the same stationary distribution, as does the set of

Bacillus and *Deinococcus.* So the tree topology results from the hierarchical clustering method is in agreement with the test result, as we can see from the merge matrix 5.12, which puts the set *Aquifex, Thermus, and Thermotoga* in on side of the tree and the set of *Bacillus* and *Deinococcus* in the other side of the tree, see Figure 20.

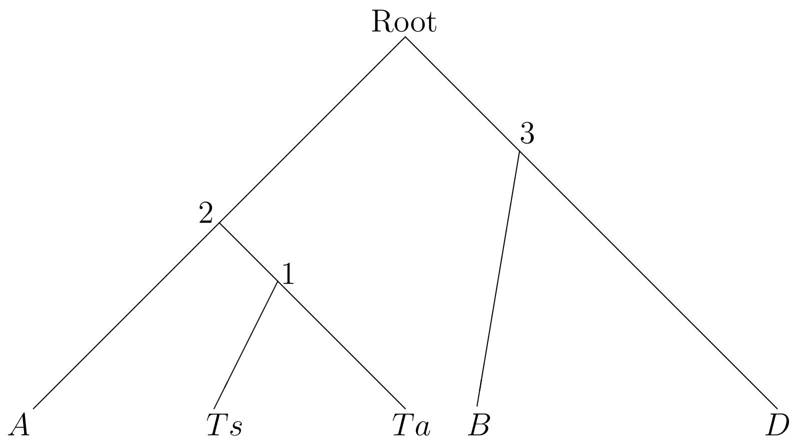


Figure 20. Bacterial data A: *Aquifex*, B: *Bacillus*, D: *Deinococcus*, Ts: *Thermus*, Ta: *Thermotoga*

Example 5.9. Consider the hominoid data described in Example 4.9. Finding the distance matrix for this data gives the result in Table 5.11.

	Hu	Bo	Ch	Go	Or	Gi
Bo	0.22					
Ch	0.24	0.14				
Go	0.33	0.26	0.27			
Or	0.45	0.44	0.43	0.50		
Gi	0.46	0.43	0.44	0.47	0.50	
Ma	0.70	0.67	0.68	0.72	0.72	0.59

TABLE 5.11. Paralinear distance matrix for hominoid data - Hu: *Human*, Bo: *Bonobo*, Ch: *Chimpanzee*, Go: *Gorilla*, Or: *Orangutan*, Gi: *Gibbon*, Ma: *Macaque*

Applying the `hclust` function on this distance matrix, gives the merge matrix

$$merge = \begin{bmatrix} -2 & -3 \\ -1 & 1 \\ -4 & 2 \\ -6 & 3 \\ -5 & 4 \\ -7 & 5 \end{bmatrix}, \tag{5.13}$$

and a vector of heights,

$$heights = (0.14,\ 0.23,\ 0.29,\ 0.45,\ 0.46,\ 0.68),$$

where node -1 represents *Human*, -2 represents *Chimpanzee*, -3 represents *Bonobo*, -4 represents *gorilla*, node -5 represents *Orangutan*, node -6 represents *Gibbon* and node -7 represents *Macaque*.

In Chapter 4 Example 4.9, we showed that none of the three statistical tests indicate a lack of symmetry between the genes of *Human*, *Chimpanzee*, *Bonobo*, *Gorilla*, *Orangutan*, *Gibbon* and *Macaque*, which indicate that this data is consistent with stationarity and homogeneity assumptions. However, inspection of the distance matrix in Table 5.11 shows that the three-point condition equation (5.1) is not met, which implies that the distances are not ultrametric. It is therefore possible that the tree inferred using the `hclust` algorithm (Figure 21) may be misleading. Use of the Neighbor-joining method from PHYLIP (Felsenstein, 2004b), which does not assume ultrametric distances, indeed changed the position of the Gibbon and the Orangutan (Figure 22). The models we are considering are on rooted trees and we have used the UPGMA or hierarchical clustering (using `hclust`) which is really appropriate only for homogeneous models. It may be better to use the Neighbor-joining method more widely when models are not homogeneous.

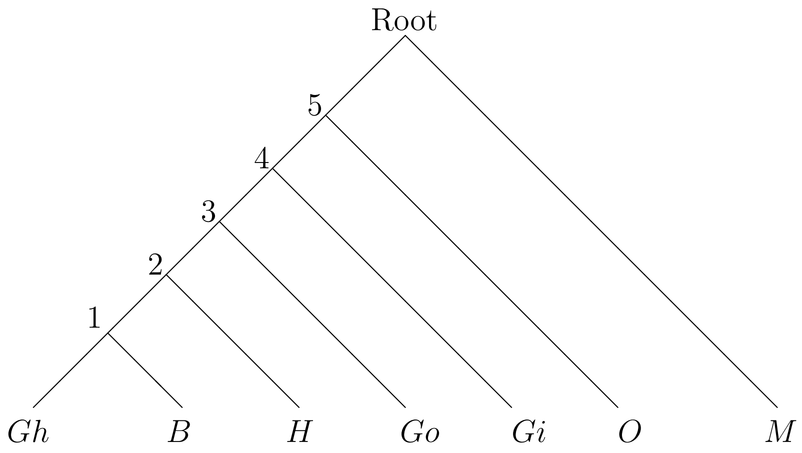


Figure 21. Hominoid data tree topology H: *Human*, Gh: *Chimpanzee*, B: *Bonobo*, Go: *Gorilla*, O: *Orangutan*, Gi: *Gibbon* M: *Macaque*

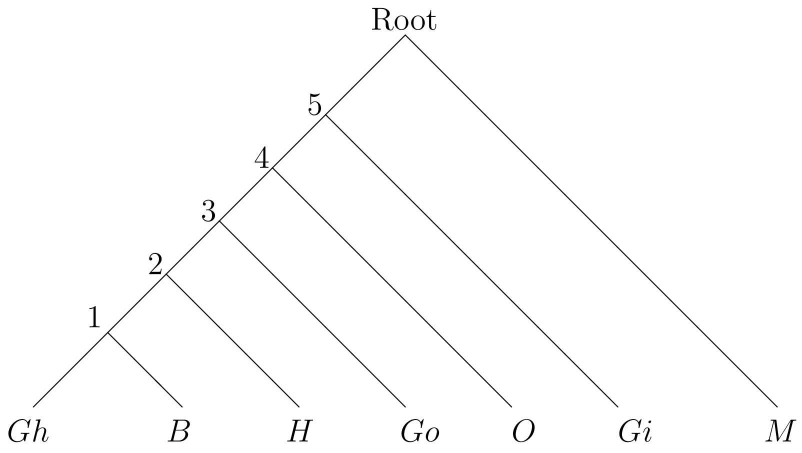


Figure 22. Hominoid data tree topology H: *Human*, Gh: *Chimpanzee*, B: *Bonobo*, Go: *Gorilla*, O: *Orangutan*, Gi: *Gibbon* M: *Macaque*

Chapter 6

Estimation of Evolutionary Parameters

6.1. Introduction

Models of nucleotide substitution were constructed for combined analysis of heterogeneous sequence data from a set of species. The models account for different aspects of heterogeneity in the evolutionary process of different genes, such as differences in nucleotide frequencies, in substitution rate bias and in the extent of rate variation across sites.

Consider a Markov process $X(t)$ on any edge of a phylogenetic tree. It is a stationary process on that edge if $P(X(t) = i) = \pi_i$, for any $t,\ t \geq 0,\ i = 1,\ \cdots, 4$. It is a homogeneous process on an edge if $P(X(t) = j|X(0) = i) = P_{ij}(t)$ satisfies $P(t) = (P_{ij}(t)) = e^{Rt}$. A process is homogeneous over a tree if it is homogeneous on each edge of the tree and R is constant on all the edges. A stationary process is reversible if $\Pi P(t) = P(t)^T\Pi$ or $\pi_i P_{ij}(t) = \pi_j P_{ji}(t)$; this is also equivalent to $\pi_i R_{ij} = \pi_j R_{ji}$ for a homogeneous process or $R = S\Pi$, where S is a symmetric matrix.

We will start with processes that are stationary, homogeneous and reversible on the tree. Then we generalize to non-stationary processes and allow heterogeneity over the tree. We restrict attention to homogeneous processes on edges allowing π to differ on some edges, $P(X(0) = i) = f_{0i} \neq \pi$, but only permitting S to vary by a single scalar multiplier (so on each edge $R = \rho S\Pi$, where ρ can vary from edge to edge). We also keep S symmetric.

In this chapter, we will discuss estimation using maximum likelihood. First we will discuss the estimation method for a two leaf tree under stationary, homogeneous and reversible conditions, then we will extend to K matched sequences and allow for non-stationary and non-homogeneous processes. We describe a general optimizer method for the log likelihood ratio using the statistical package R. We point out how the topology of the phylogenetic tree is chosen using the methods from Chapters 4 and 5. Then we describe the method of estimation under the general Markov model using the gn program, describe in Section 2.6.2 and show how to use the gn program to estimate under simpler models like the Jukes-Cantor model and the general time-reversible Markov model.

We proceed by giving different examples examining the accuracy of the method by generating joint distribution functions $F(t)$ with known parameters and from these joint distributions simulating data sets. We obtain estimates from these data sets to examine the properties of the method of estimation. Some of these examples will illustrate the conditions under which the estimation is possible and we examine some cases where we cannot get estimates.

Then we apply the method of estimation to two real data sets discussed previously in Chapter 4. Next we consider non-parametric and parametric bootstrap methods to obtain bias and standard deviations for the estimates. We apply these bootstrap methods to the simulated data sets, for which we consider estimation to examine the properties under conditions where the models hold. Then we use these bootstrap methods to give information about the estimates of the parameters for the real data sets. Finally, we summarize the results of this chapter.

6.2. Maximum Likelihood Estimation

Suppose that $X_1, X_2, \cdots, X_n$ have a joint frequency distribution $f(x_1, \cdots, x_n; \theta)$, where θ is a vector of unknown parameters that is to be estimated. Now given the observed values $x_1,\ x_2, \cdots, x_n$, the likelihood as a function of θ for given $x_1, \cdots, x_n$, is defined as

$$L(\theta) = f(x_1, \cdots, x_n; \theta).$$

The maximum likelihood estimate of θ is the value of θ that maximizes the likelihood. Assuming $X_1, \cdots, X_n$ are independent and identically distributed random variables, the likelihood will be

$$L(\theta) = \prod_{i=1}^{n} f(x_i; \theta).$$

Rather than maximizing the likelihood, it is easier to maximize the natural logarithm of the likelihood

$$l(\theta) = log(L(\theta)) = \sum_{i=1}^{n} log(f(x_i; \theta)).$$

Now for any phylogenetic tree with K tips, let $N_1, \cdots, N_m$ follow a multinomial distribution with a total count of n and cell probabilities $f_1, \cdots, f_m$, where $n = \sum_{i=1}^{m} N_i$, $m = 4^K$ and $f = f_i(x_i; \theta)$. We replace the likelihood $L(\theta)$ by the likelihood of the sufficient statistic $N_1, \cdots, N_m$. The joint frequency function of $N_1, \cdots, N_m$ is

$$P(\mathbf{N} = \mathbf{n}; f(\theta)) = \frac{n!}{n_1! \cdots n_m!} f_1^{n_1} \cdots f_m^{n_m},$$

where $\mathbf{n} = (n_1, \cdots, n_m)$ and $\mathbf{N} = (N_1, \cdots, N_m)$.

The log likelihood in this case is

$$\begin{aligned} l(f_1, \cdots, f_m) &= log[P(\mathbf{N} = \mathbf{n}; f_1, \cdots, f_m)] \\ &= \sum_{i=1}^{m} n_i log(f_i) - \sum_{i=1}^{m} log(n_i!) + log(n!). \end{aligned}$$

If we do not restrict $f_1, \cdots, f_m$ to be functions of θ, then the maximum likelihood is

$$\begin{aligned} l(\hat{f}_1, \cdots, \hat{f}_m) &= max(\sum_{i=1}^{m} n_i log(f_i) - \sum_{i=1}^{m} log(n_i!) + log(n!)) \\ &= (\sum_{i=1}^{m} n_i log(\hat{f}_i) - \sum_{i=1}^{m} log(n_i!) + log(n!)), \end{aligned}$$

where $\hat{f}_i = n_i/n$. If the model gives $f_i = f_i(\theta)$, then the log likelihood would be

$$\begin{aligned} l(\hat{\theta}) &= max_\theta l(\theta) \\ &= max_\theta(\sum_{i=1}^{m} n_i log(f_i(\theta)) - \sum_{i=1}^{m} log(n_i!) + log(n!)). \end{aligned}$$

The log likelihood ratio is then

$$\begin{aligned} \Re(\hat{\theta}) &= l(\hat{\theta}) - l(\hat{f}_1, \cdots, \hat{f}_m) \\ &= max_\theta \sum_{i=1}^{m} n_i log(\frac{f_i(\theta)}{\hat{f}_i}). \end{aligned} \tag{6.1}$$

When we estimate under the exact joint distribution array $F(t)$, we are actually maximizing the limit of $\Re(\theta)/n$, that is

$$max_\theta(\lim_{n\to\infty}(\sum_{i=1}^{m} \hat{f}_i log(\frac{f_i(\theta)}{\hat{f}_i}))).$$

6.3. Estimating the Evolutionary Parameters for a Two Leaf Tree

For a two leaf tree, we will use the maximum likelihood method to estimate the parameters under the case where we assume stationary, homogeneous and reversible processes for an arbitrary value of t.

Knowing that $\sum_j^4 \sum_i^4 f_{ij}(t) = 1$ and $F(t)$ is symmetric since it is reversible, let the log likelihood function of the joint distribution function $f_{ij}(t)$ be denoted by $l(f_{ij}(t))$. Using Lagrange multipliers, γ, the log likelihood for $F(t)$ is

$$l(f_{ij}(t)) = \sum_i^4 \sum_j^4 \frac{N_{ij} + N_{ji}}{2} log(f_{ij}(t)) + \gamma(\sum_i^4 \sum_j^4 f_{ij}(t) - 1).$$

Now in order to estimate the parameters we find the derivatives of the log likelihood $l(f_{ij}(t))$ with respect to γ and $f_{ij}(t)$ and set them equal to zero. Solving the two derivative equations gives

$$\hat{f}_{ij}(t) = \frac{N_{ij} + N_{ji}}{2N}. \tag{6.2}$$

Now if $F(t)$ is a one-to-one function with the parameter θ, this implies that $\hat{\theta} = \theta(\hat{f})$, where θ is a function of the symmetric matrix S and Π, that is $\theta = (S, \Pi)$. Here we mean that θ contains the six off-diagonal elements of S and the first three elements of π, the other elements being obtained from $\Pi\mathbf{1} = \mathbf{1}$ and $S\Pi = 0$. Now in order to find an estimate for S and Π, let

$$\begin{aligned} F(t)\mathbf{1} &= P^T(t)F(0)P(t)\mathbf{1} \\ &= P^T(t)F(0)\mathbf{1}. \end{aligned}$$

Let $F(0) = \Pi$, this implies that

$$\begin{aligned} F(t)\mathbf{1} &= (P(t))^T\Pi\mathbf{1} \\ &= \pi, \end{aligned}$$

so

$$\begin{aligned} \hat{\pi} &= \hat{F}(t)\mathbf{1} \\ &= (\frac{N_{ij} + N_{ji}}{2N})\mathbf{1} \\ &= (\frac{N_{i.} + N_{.i}}{2N}). \end{aligned}$$

In Chapter 2, equation (2.26) we showed that

$$\begin{aligned} \Pi^{-1/2}F(t)\Pi^{-1/2} &= \sum_{l=1}^{4} e^{2\lambda_i t}U_l U_l^T \\ &= \Pi^{1/2}\mathbf{1}\mathbf{1}^T\Pi^{1/2} + \sum_{l=2}^{4} e^{2\lambda_l t}U_l U_l^T, \end{aligned}$$

where λ_l are the eigenvalues of $\Pi^{1/2}S\Pi^{1/2}$ and U_l are the eigenvectors of $\Pi^{1/2}S\Pi^{1/2}$, $l = 1, \cdots, 4$. Solving for λ and U we get

$$\begin{aligned} \lambda_l t &= \frac{1}{2} log(eigenvalue_l(\Pi^{-1/2}F(t)\Pi^{-1/2})) \\ U_l &= eigenvector_l(\Pi^{-1/2}F(t)\Pi^{-1/2}), \end{aligned}$$

We know the estimate of $F(t)$, so we can find the estimate of S, such that

$$\hat{S} = \hat{\Pi}^{-1/2}(\sum_{l=1}^{4} \hat{\lambda}_l \hat{U}_l \hat{U}_l^T)\hat{\Pi}^{-1/2},$$

where

$$\begin{aligned} \hat{\lambda}_l &= \frac{1}{2} log(eigenvalue_l(\hat{\Pi}^{-1/2}\hat{F}(t)\hat{\Pi}^{-1/2})) \\ \hat{U}_l &= eigenvector_l(\hat{\Pi}^{-1/2}\hat{F}(t)\hat{\Pi}^{-1/2}). \end{aligned}$$

Example 6.1. Consider the case of stationary, homogeneous and reversible processes for a two leaf tree. Let $\pi = f_0 = (.25, .25, .25, .25)^T$, the s-vector $= (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47)$ and the two leaves have the same length, that is $t_1 = t_2 = 1$. In this case the joint distribution function $F(t)$ will depend on nine free parameters. Finding the estimates for the nine parameters under this model gives exact estimates, that is

$$\hat{\pi} = (.25, .25, .25, .25)$$

and

$$\hat{s}\text{-vector} = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47).$$

Now using a simulated data set of size 1000, finding the estimates for the nine parameters gives

$$\hat{\pi} = (.230, .256, .249, .266)$$

and

$$\hat{s}\text{-vector} = (0.234,\ 0.315,\ 0.922,\ 0.018,\ 0.934,\ 0.482).$$

Increasing the simulated data set size to 10000, and finding the estimates gives us more accurate estimates, that is

$$\hat{\pi} = (.249, .252, .251, .250)$$

and

$$\hat{s}\text{-vector} = (0.207,\ 0.370,\ 0.760,\ 0.012,\ 0.94,\ 0.472).$$

6.4. Optimization using the R Package

In this section we will discuss the optimization function we will use in order to find the estimates for our model parameters. In the statistical package R, we will use the general-purpose optimization `optim` function, in order to maximize the log likelihood ratio. This function is given by

$$\texttt{optim}(\ par,\ fn,\ gr = \text{ NULL},\ \text{method} = \text{“}BFGS\text{”}, ...)$$

The description given for this function in the statistical package R is a “ ... general-purpose optimization based on Nelder-Mead, quasi-Newton and conjugate-gradient algorithms. It includes an option for box-constrained optimization and simulated annealing where
par : Initial values for the parameters to be optimized over.
fn : A function to be minimized, with first argument the vector of parameters over which minimization is to take place. It should return a scalar result.
gr: A function to return the gradient for the “$BFGS$”. If it is 'NULL', a finite-difference approximation will be used.
Method “$BFGS$” is a quasi-Newton method (also known as a variable metric algorithm). This uses function values and gradients to build up a picture of the surface to be optimized.” For more details, see the R package help file.

The function to be minimized, fn, in our case is $-\Re(\theta)/n$, which is equivalent to maximizing $\Re(\theta)/n$, but since the `optim` function performs minimization by default, we will minimize the function $-\Re(\theta)/n$. In the case of the exact joint distribution function $F(t)$, we are actually minimizing the limit of $-\Re(\theta)/n$, that is

$$min_\theta(\lim_{n\to\infty}(\sum_{i=1}^{m}\hat{f}_i\ log(f_i(\theta)/\hat{f}_i))).$$

Remark: For the remainder of this chapter, the log likelihood ratio value means $-\Re(\hat{\theta})/n$.

6.5. Choosing the Phylogenetic Tree Topology

In order to find the parameter estimates for K matched sequences we need to know the topology of the tree associated with these sequences together. In order to determine the phylogenetic tree topology for K matched sequences we will use two methods: the testing method described in Chapter 4 and the paralinear distance method described in Chapter 5. In Chapter 4 we demonstrated that the testing method did not give any significant differences in the case of stationary processes whether we have homogeneous processes or not except for the case when the internal symmetry was significant. However, the tests of symmetry, marginal symmetry and internal symmetry give significant results in the case of non-stationary processes. In Chapter 5 we showed that the paralinear distance method gives an ultrametric tree with correct topology for the cases of stationary and homogeneous processes. However, the paralinear distance gives the correct unrooted tree topology in the case of non-homogeneous processes. Throughout the remainder of this chapter we will discuss in more detail the way in which we choose the tree topology.

6.6. Estimating the General Model Parameters using the gn Program

In Chapter 2, we discussed different models, most of which assumed stationarity and homogeneity, as in, for example, Jukes and Cantor (1969), Kimura (1980) and Felsenstein (1981). The gn program is based on a model that can be non-stationary and non-homogeneous.

As in Chapter 2, the model of the gn program divides any tree of K matched sequences into two parts depending on the first bifurcation from the root. Knowing that the last row of the merge matrix gives the two nodes bifurcating from the root, we used transitions based on $P^X(t) = e^{S\Pi_X t}$ for all edges associated with the node given by the first element in the last row of the merge matrix, and $P^Y(t) = e^{\rho S\Pi_Y t}$ was used for the edges on the other side of the tree; we also used the initial distribution, f_0, for the root. This implies that for the K matched sequences we will have a set of parameters, θ, where θ consists of π_X, π_Y, f_0, the s-vector, ρ, and a vector of heights containing $K-2$ lengths. For the parameters π_X, π_Y and f_0 we will have three free parameters for each one of them, since the sum of each of them should equal one. In total, for K matched sequences we will have $16+(K-2)$ free parameters to estimate.

Throughout this section we will estimate under the general Markov model, which we described in Section 2.7.6, except Section 6.6.2, where we will discuss simpler models.

During the optimization process, the general optimizer function `optim` searches for a set of parameters that minimize the log likelihood ratio. However, several parameters are required to satisfy the constraints. We need $\sum_{i=1}^{n} \pi_{X_i} = \sum_{i=1}^{n} \pi_{Y_i} = \sum_{i=1}^{n} f_{0_i} = 1$, $i = 1, \cdots, 4$ and all parameter values must be positive. To accomplish this within the gn program we return zero whenever $\sum_{i=1}^{n} |\pi_{X_i}| > 1$, $\sum_{i=1}^{n} |\pi_{Y_i}| > 1$, $\sum_{i=1}^{n} |f_{0_i}| > 1$, $\rho < 0$, any $S_{ij} < 0$ and any t_i lies outside (0,1], $i = 1, 2, \cdots, (K-2)$. So the minimum is forced to occur subject to these constraints.

Remark: We will use Section 6.3 to get initial values for the parameters $\pi_X = \pi_Y = f_0$ and the s-vector by taking any two edges from the tree and estimating these parameters using the method for a two leaf tree. We take the height vector results from the hierarchical cluster function `hclust` after dividing it by the highest value of this height vector as initial values for the height vector.

6.6.1. Estimation under the Case of Stationary and Homogeneous Processes. We will give here different examples to show how the estimation process works for different cases. In the case of stationary and homogeneous processes, we showed in Chapter 4 that the statistical tests did not give any information about the tree topology. In addition, we showed in Chapter 5 that the paralinear distance for the joint distribution array $F(t)$ is an ultrametric distance under these assumptions, and the hierarchical clustering using `hclust` function on the paralinear distance matrix gives the exact tree topology.

Example 6.2. Consider the case of stationary and homogeneous processes for a tree with five matched sequences, described by the merge matrix (4.2). The tree is given in Figure 23. Let

$$\pi_X = \pi_Y = f_0 = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,$$

$$s\text{-vector } = (0.2,\ 0.2,\ 0.2,\ 0.2,\ 0.2,\ 0.2).$$

Take the height vector to be $(0.1, 0.5, 0.8, 1.0)$. Under the general Markov model we have 19 free parameters to estimate. Using the joint distribution function array $F(t)$ and the merge matrix (4.2), the estimates for the parameters, using the `optim` function are given in Table 6.1.

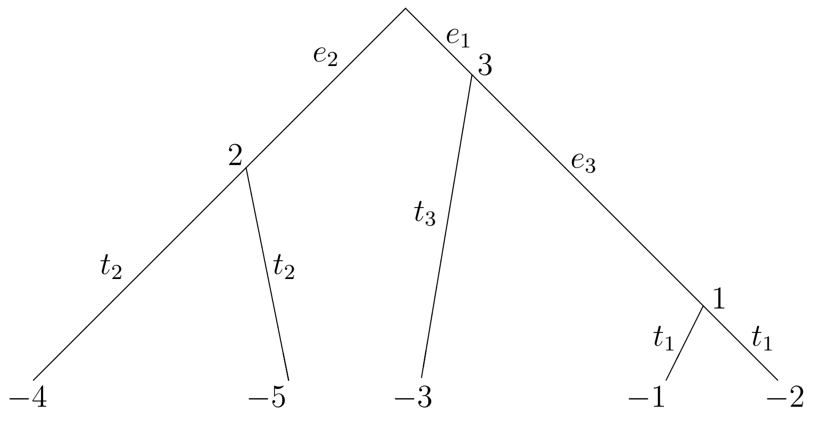


Figure 23. Rooted five tipped tree

The value of the log likelihood ratio in this case is $8.6e-12$. The estimated parameters in Table 6.1 and the exact parameters we used to calculate the joint distribution function array satisfy the following equations,

$$\hat{s_i} \times \hat{\rho} \times \hat{t_j} = s_i \times \rho \times t_j \tag{6.3}$$

for the edges in the right side of the root and

$$\hat{s_i} \times \hat{t_j} = s_i \times t_j \tag{6.4}$$

for the edges in the left side of the root, where $\hat{s_i}$, $\hat{\rho}$, and $\hat{t_j}$ are the estimated values for the parameters s_i, ρ and t, respectively, $i = 1, \cdots, 6$, $j = 1, 2, 3$. Note that these parameters are not estimable. Two sets of parameters satisfying the Equations (6.3) and (6.4), give the same joint distribution at the end nodes. We can see that we have exact estimates for the parameters π_X, π_Y and f_0 but for the parameters, ρ, s-vector and the length vector, the exact and the estimated parameters should satisfy Equations (6.3) and (6.4), such that

$$\begin{aligned} t_1 &= (\hat{s_i} \times \hat{\rho} \times \hat{t_1})/(s_i \times \rho) \\ &= (0.239 \times 0.677 \times 0.124)/(0.2 \times 1) \\ &= 0.1, \end{aligned}$$

where $\hat{s_i}$ is any value from the $\hat{s}$-vector. In the same way we can find $t_2 = 0.5$, $t_3 = 0.8$ and $s_i = 0.2$, $i = 1, \cdots, 6$.

Repeating the estimation process for the same joint distribution function $F(t)$ and using the same merge matrix, but in this case changing the values of the starting parameters in the `optim` function, gave the estimates in Table 6.2, with log likelihood ratio value equal to $2.7e-11$. In the same way see that the exact parameters and there estimates satisfy the Equations (6.3) and (6.4).

Parameters	Estimates
$\hat{\pi}_X$	0.250, 0.250, 0.250
$\hat{\pi}_Y$	0.250, 0.250, 0.250
$\hat{f}_0$	0.250, 0.250, 0.250
$\hat{s}$-vector	0.239, 0.239, 0.239, 0.239, 0.239, 0.239
$\hat{\rho}$	0.677
height	0.124, 0.419, 0.991

TABLE 6.1. Estimates of parameters of the exact joint distribution for five matched sequences evolved under stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.250, 0.250, 0.250
$\hat{\pi}_Y$	0.250, 0.250, 0.250
$\hat{f}_0$	0.250, 0.250, 0.250
$\hat{s}$-vector	0.172, 0.172, 0.172, 0.172, 0.172, 0.172
$\hat{\rho}$	1.325
height	0.088, 0.581, 0.702

TABLE 6.2. Estimates of parameters of the exact joint distribution for five matched sequences evolved under stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Example 6.3. Consider the same tree topology, assumptions and the same parameters as in Example 6.2, but in this case take the s-vector to be

$$s\text{-vector } = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47).$$

Estimation under the general Markov model depending on the joint distribution array $F(t)$ and the merge matrix (4.2), gives the parameter estimates shown in Table 6.3, with log likelihood ratio equal to $3.1e{-}11$.

Knowing the estimated values and using Equations (6.3) and (6.4) we can find the exact values of the parameters.

Changing the starting values for the optimization function, `optim`, will give different estimates for the parameters as we can see in Table 6.4 with log likelihood ratio equal to $8.2e-8$. As before, the exact and the estimated values satisfy the Equations (6.3) and (6.4).

Parameters	Estimates
$\hat{\pi}_X$	0.250, 0.250, 0.250
$\hat{\pi}_Y$	0.250, 0.250, 0.250
$\hat{f}_0$	0.250, 0.250, 0.250
$\hat{s}$-vector	0.191, 0.334, 0.753, 0.010, 0.887, 0.448
$\hat{\rho}$	1.098
height	0.096, 0.524, 0.764

Table 6.3. Estimates of parameters of the exact joint distribution for five matched sequences evolved under stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.250, 0.250, 0.250
$\hat{\pi}_Y$	0.250, 0.250, 0.250
$\hat{f}_0$	0.250, 0.250, 0.250
$\hat{s}$-vector	0.229, 0.402, 0.906, 0.012, 1.066, 0.534
$\hat{\rho}$	0.744
height	0.087, 0.586, 0.698

Table 6.4. Estimates of parameters of the exact joint distribution for five matched sequences evolved under stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Example 6.4. Using the joint distribution array $F(t)$ calculated in Example 6.3 and using one of the simulation methods described in Chapter 3, we simulated a data set of size $n = 1000$ and found the observed divergence array N. Applying the statistical tests described in Chapter 4 on this data gives no significant differences and applying the `hclust` function from the distance method gives the merge matrix (4.2). Using the merge matrix (4.2) and the observed divergence array, N, we estimated the parameters under the general Markov model using the `optim` function, which gives the results shown in Table 6.5, with log likelihood ratio value equal to 0.203.

Parameters	Estimates
$\hat{\pi}_X$	0.282, 0.241, 0.245
$\hat{\pi}_Y$	0.225, 0.281, 0.270
$\hat{f}_0$	0.256, 0.247, 0.249
$\hat{s}$-vector	0.157, 0.296, 0.998, 0.001, 0.2986, 0.451
$\hat{\rho}$	1.000
height	0.095, 0.501, 0.803

TABLE 6.5. Estimates of parameters from data comprising five matched sequences of length $n = 1000$ evolved under stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Simulating a larger data set with $n = 10000$ from the same joint distribution and estimate as before gives the result in Table 6.6, with log likelihood ratio value equal to 0.039.

When we estimate parameters using the exact joint distribution function $F(t)$ for the case of stationary and homogeneous processes, we obtain the exact stationary distribution for each side of the tree, π_X and π_Y, and the exact initial distribution f_0, but for the other parameters, ρ, the s-vector and the length vector, we get different estimates satisfying Equations (6.3) and (6.4), as we saw in Examples 6.2

Parameters	Estimates
$\hat{\pi}_X$	0.226, 0.270, 0.250
$\hat{\pi}_Y$	0.244, 0.251, 0.250
$\hat{f}_0$	0.255, 0.256, 0.252
$\hat{s}$-vector	0.197, 0.363, 0.799, 0.015, 0.953, 0.473
$\hat{\rho}$	1.019
height	0.095, 0.500, 0.807

TABLE 6.6. Estimates of parameters from data comprising five matched sequences of length $n = 10000$ evolved under stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

and 6.3. This means that not all the parameters are estimable (that is we cannot estimate both ρ and the length vector under stationary and homogeneous processes). This means that in Figure 23 we can move the root of the tree around to any point between node 3 and 2. This is in agreement with the ideas of Felsenstein (1981).

In practice, this means that we cannot estimate both rate and time, which explains why most trees published in the literature are drawn with edge lengths as a product of time and rate.

6.6.2. Estimation under Simpler Evolutionary Models. As discussed in Chapter 2, the gn program allows different rate matrices for each side of the root. This means that we can estimate under models having more constraints. In this section, we will give two cases where we reduce the gn program to simpler models.

6.6.2.1. *Estimation under the General Time-Reversible Markov Model.* The general time-reversible Markov model assumes that the processes are stationary, so $\pi_X = \pi_Y = f_0$, and homogeneous, so $\rho = 1$; thus, in this case, we have $(9 + (K - 2))$ free parameters for K matched sequences. In order to make the gn program suitable to fit the parameters of the general time-reversible Markov model, we need to restrict some parameters. We wrote a small program to do this that takes as input a vector of parameters, θ_1, and a merge matrix describing the tree topology. For a tree of K matched sequences the vector of parameters, θ_1, contains the following elements: the first three values of π, six values for the s-vector and $K-2$ values for the edges lengths. Inside the program, we increase the number of parameters in θ_1 to be exactly the same as θ (the input for the gn program) by taking $\pi_X = \pi_Y = f_0 = \pi$ and $\rho = 1$. The new program can then use the gn program to calculate the joint distribution array $F(t)$ for any tree of K matched sequences.

Example 6.5. Consider the case of stationary and homogeneous processes for a tree with five matched sequences, described by the merge matrix

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ -3 & 1 \\ 2 & 3 \end{bmatrix}. \tag{6.5}$$

Let

$$\pi = (0.1,\ 0.1,\ 0.1,\ 0.7)^T,$$

$$s\text{-vector } = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47).$$

and the height vector to be $(0.1, 0.5, 0.8, 1.0)$. Using the joint distribution array $F(t)$ generated under the general time-reversible Markov model and the merge matrix (6.5), we simulate a data set of size $n = 1000$. Depending on the observed divergence array N, the estimation under the general time-reversible Markov model gives the parameter estimates shown in Table 6.7, with log likelihood ratio value equal to 0.1394.

Estimating the same observed divergence array N under the general Markov model gives the parameter estimates shown in Table 6.8, with log likelihood ratio value equal to 0.1384.

Comparing Table 6.7 and Table 6.8, we can see that the estimates inferred in Table 6.7 are closer to those used to generate the data.

Parameters	Estimates
$\hat{\pi}$	0.096, 0.095, 0.101
$\hat{s}$-vector	0.017, 0.309, 0.805, 0.0007, 1.023, 0.452
height	0.096, 0.516, 0.762

TABLE 6.7. Estimates of parameters from data comprising five matched sequences of length n = 1000. The data were generated by simulation under a general time-reversible Markov model. The estimates were obtained assuming the general time-reversible Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.066, 0.098, 0.124
$\hat{\pi}_Y$	0.074, 0.100, 0.123
$\hat{f}_0$	0.126, 0.093, 0.090
$\hat{s}$-vector	0.057, 0.309, 0.808, 0.0001, 0.842, 0.329
$\hat{\rho}$	1.377
height	0.081, 0.614, 0.646

TABLE 6.8. Estimates of parameters from data comprising five matched sequences of length n = 1000. The data were generated by simulation under a general time-reversible Markov model. The estimates were obtained assuming the general Markov model

This implies that if we invoke parameters that did not exist during the generation of the data (as in Table 6.8), then we get imprecise estimates, and may be led to believe that there are differences between parameter estimates where there are in fact none.

6.6.2.2. *Estimation under the Jukes-Cantor Model.* The Jukes-Cantor (1969) model assumes stationarity and homogeneity with $\pi_i = 0.25$, $i = 1, \cdots, 4$ and one free parameter α for the case of two matched sequences. For the case of K matched sequences the Jukes-Cantor model has $(1 + (K - 2))$ free parameters to estimate, α and $(K - 2)$ values for the edge lengths. In order to make the gn program suitable to fit the parameters of the Jukes-Cantor model we wrote a small program taking as input, a vector of parameters θ_2 and a merge matrix describing the tree topology.

For a tree of K matched sequences the vector of parameters θ_2 contains the following elements: the evolutionary rate α and $K-2$ values for the edge lengths. Inside the program we increase the number of parameters in θ_2 to be exactly the same as θ (the input for the gn program) by taking $\pi_X = \pi_Y = f_0 = (0.25, 0.25, 0.25, 0.25)$, $\rho = 1$ and the s-vector $= (\alpha, \alpha, \alpha, \alpha, \alpha, \alpha)$. The new program can use the gn program to calculate the joint distribution array $F(t)$ for any tree of K matched sequences.

Example 6.6. Consider the case of stationary and homogeneous processes for a tree with five matched sequences, described by the merge matrix (6.6). Let $\pi = (0.25,\ 0.25,\ 0.25,\ 0.25)^T$, $\alpha = 0.2$ and the height vector to be $(0.1, 0.5, 0.8, 1.0)$. Using the joint distribution array $F(t)$ generated under the Jukes-Cantor model and the merge matrix (6.5), we simulated a data set of size $n = 1000$. Depending on the observed divergence array N, the estimation under the Jukes-Cantor model gives the parameter estimates shown in Table 6.9, with log likelihood ratio value equal to 0.1471. Estimating the same observed divergence array, N, under the general Markov model gives the parameter estimates shown in Table 6.10, with log likelihood ratio value equal to 0.1394.

Comparing Table 6.9 and Table 6.10, we can see, in the same way as in Example 6.5, that the estimates inferred in Table 6.9, adjusted using Equations (6.3) and (6.4) are closer to those used to generate the data.

6.6.3. Estimation under the Case of Stationary and Non-Homogeneous Processes. We showed earlier in Chapter 4 that the statistical tests did not give any information about the tree topology if we have stationary processes. Also, we showed in Chapter 5, that

Parameters	Estimates
$\hat{\alpha}$	0.207
height	0.121, 0.462, 0.917

TABLE 6.9. Estimates of parameters from data comprising five matched sequences of length n = 1000. The data were generated by simulation under the Jukes-Cantor model. The estimates were obtained assuming the Jukes-Cantor model

Parameters	Estimates
$\hat{\pi}_X$	0.151, 0.266, 0.270
$\hat{\pi}_Y$	0.227, 0.242, 0.225
$\hat{f}_0$	0.253, 0.249, 0.241
$\hat{s}$-vector	0.173, 0.146, 0.161, 0.157, 0.101, 0.119
$\hat{\rho}$	1.941
height	0.094, 0.684, 0.692

TABLE 6.10. Estimates of parameters from data comprising five matched sequences of length n = 1000. The data were generated by simulation under the Jukes-Cantor model. The estimates were obtained assuming the general Markov model

under the case of stationary and non-homogeneous processes the paralinear distance for the exact joint distribution function $F(t)$ is not an ultrametric distance and applying the hierarchical clustering using `hclust` function from R on the paralinear distance matrix did not give the correct rooted tree topology, but it gave the correct unrooted tree topology. Throughout this section in order to determine the correct tree topology, we will use different combinations of the merge matrix, which we get from the `hclust` function, where we will change the location of the root while keeping the unrooted tree topology.

Example 6.7. Consider the case of stationary and non-homogeneous processes for a tree with five matched sequences, described in Example

6.2 with a height vector equal to $(0.1, 0.5, 0.8, 1.0)$. Let

$$\pi_X = \pi_Y = f_0 = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,$$

$R^X = S_X \Pi_X$ and $R^Y = \rho S_Y \Pi_Y$, where the six free parameters in S_X and S_Y matrices are given by the

$$s\text{-vector } = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47)$$

and $\rho = 3$. As we showed in Example 5.3, applying the `hclust` function on the paralinear distance matrix of this model gives the merge matrix (5.5), which describes the topology of the tree shown in Figure 24, Finding the parameter estimates using the merge matrix (5.5), which we get from the `hclust` function, gives the result in Table 6.11, with log likelihood ratio equal to 0.022.

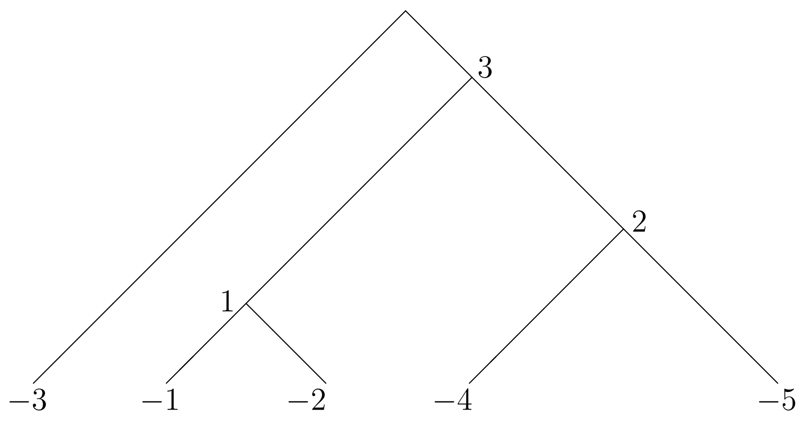


Figure 24. Rooted Five Tipped Tree

We can see from Table 6.11 that estimation using the merge matrix (5.5) did not give the correct estimate for the parameters. Taking the root to be at any position between node 2 and 3 in Figure 24, gives the correct tree topology described by the merge matrix given in Equation (5.6), which we used to calculate the joint distribution array.

Applying the optimization function, `optim`, on the joint distribution array with respect to the original tree topology we generate the data

Parameters	*Estimates*
$\hat{\pi}_X$	0.247, 0.246, 0.247
$\hat{\pi}_Y$	0.262, 0.238, 0.264
$\hat{f}_0$	0.246, 0.224, 0.227
$\hat{s}$-*vector*	0.159, 0.290, 0.750, 0.001, 0.886, 0.416
$\hat{\rho}$	2.970
height	0.111, 0.135, 0.801

TABLE 6.11. Estimates of parameters of the exact joint distribution for five matched sequences evolved under stationary and non-homogeneous processes. The estimates were obtained assuming the general Markov model

from, gives the parameter estimates shown in Table 6.12, with log likelihood ratio value equal to $1.22e - 10$.

Parameters	Estimates
$\hat{\pi}_X$	0.250, 0.250, 0.250
$\hat{\pi}_Y$	0.250, 0.250, 0.250
$\hat{f}_0$	0.250, 0.250, 0.250
$\hat{s}$-vector	0.298, 0.521, 1.176, 0.015, 1.385, 0.700
$\hat{\rho}$	1.686
height	0.119, 0.336, 0.956

TABLE 6.12. Estimates of parameters of the exact joint distribution for five matched sequences evolved under stationary and non-homogeneous processes. The estimates were obtained assuming the general Markov model

Repeating the estimation process as before, with respect to the same merge matrix (4.2), but with different starting values for the `optim` function, gives the result in Table 6.13, with log likelihood ratio value equal to $1.4e - 10$.

The exact values of ρ, the s-vector and the height vector and there estimated values in both Tables 6.12 and 6.13, satisfy Equations (6.3) and (6.4). Note that with the correct topology we estimate π_X, π_Y

Parameters	Estimates
$\hat{\pi}_X$	0.250, 0.250, 0.250
$\hat{\pi}_Y$	0.250, 0.250, 0.250
$\hat{f}_0$	0.250, 0.250, 0.250
$\hat{s}$-vector	0.260, 0.455, 1.026, 0.013, 1.208, 0.610
$\hat{\rho}$	2.080
height	0.111, 0.385, 0.888

TABLE 6.13. Estimates of parameters of the exact joint distribution for five matched sequences evolved under stationary and non-homogeneous processes. The estimates were obtained assuming the general Markov model

and f_0 exactly but we are unable to estimate ρ, the height vector and the s-vector since these are not estimable in this case, where we have stationarity.

Example 6.8. In this example we simulated a data set of size $n = 1000$ from the joint distribution array $F(t)$ in Example 6.7 and found the observed divergence array, N. Using the observed divergence array, N, and the merge matrix used to calculate the joint distribution array, we applied our method of estimation which, gave the result shown in Table 6.14, with log likelihood ratio value equal to 0.318.

Now assume that we do not know the tree topology, then applying the statistical tests on N shows no differences, and applying the `hclust` function from the distance method gives the incorrect tree topology described by the merge matrix (5.5). Using this merge matrix and the observed divergence array N, we can estimate the parameters under the general model using the `optim` function, which gives the results shown in Table 6.15, with log likelihood ratio value equal to 0.587.

Parameters	Estimates
$\hat{\pi}_X$	0.231, 0.233, 0.228
$\hat{\pi}_Y$	0.274, 0.230, 0.237
$\hat{f}_0$	0.260, 0.262, 0.269
$\hat{s}$-vector	0.172, 0.323, 0.593, 0.014, 0.651, 0.392
$\hat{\rho}$	4.089
height	0.097, 0.692, 0.706

Table 6.14. Estimates of parameters from data comprising five matched sequences of length $n = 1000$ evolved under stationary and non-homogeneous processes. The estimates were obtained assuming the general Markov model with the correct tree topology

Parameters	Estimates
$\hat{\pi}_X$	0.259, 0.261, 0.256
$\hat{\pi}_Y$	0.266, 0.242, 0.241
$\hat{f}_0$	0.255, 0.279, 0.294
$\hat{s}$-vector	0.346, 0.519, 1.377, 0.001, 2.473, 1.536
$\hat{\rho}$	6.045
height	0.020, 0.036, 0.777

Table 6.15. Estimates of parameters from data comprising five matched sequences of length $n = 1000$ evolved under stationary and non-homogeneous processes. The estimates were obtained assuming the general Markov model withe the incorrect tree topology from `hclust`

Notice that estimation using the correct topology gives small log likelihood ratio value compared to the estimation using the other topology. Also, using the correct tree topology gives good estimates for π_X, π_Y and f_0 but ρ, the height vector and s-vector are not estimable, however, they satisfy Equations (6.3) and (6.4) approximately.

We can see that when we estimate using the exact joint distribution function $F(t)$ and the correct tree topology we used to calculate $F(t)$, the estimation under the case of stationary and non-homogeneous processes is similar to the estimation under the case of stationary and homogeneous processes, since in both cases we can estimate the exact stationary distribution for each side of the tree, π_X and π_Y, and the exact initial distribution f_0, but for the other parameters, ρ, the s-vector and the length vector, we can estimate different combination between them, given that Equations (6.3) and (6.4) should satisfied.

6.6.4. Estimation under the Case of Non-Stationary and Homogeneous Processes. In Chapter 5, we showed that in the case of homogeneous processes the paralinear distance matrix for the exact joint distribution array $F(t)$ is ultrametric and performing hierarchical clustering on the paralinear distance matrix using the `hclust` function gives the exact tree topology; also, performing hierarchical clustering on the distance matrix from observed divergence matrix N gives the correct tree topology.

Knowing that, for the case of homogeneous processes, hierarchical clustering gives the correct tree topology for the exact and observed joint distribution functions, for the purpose of estimation we will use the correct merge matrix we used to calculate the joint distribution array.

Example 6.9. Consider the case of non-stationary and homogeneous processes for a tree with five matched sequences, described in Example 6.2 with a height vector equal to $(0.1, 0.5, 0.8, 1.0)$. Let

$$\pi_X = \pi_Y = \pi = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,\ f_0 = (0.1,\ 0.1,\ 0.1\ ,0.7)^T$$

$R^X = R^Y = S\Pi$, where the six free parameters in the S matrix are given by the

$$s\text{-vector } = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47).$$

We showed in Example 5.5, that the `hclust` function for the paralinear distance matrix of the joint distribution array for this model gives the same merge matrix that we used to calculate the joint distribution array $F(t)$. Now applying the `optim` function on the joint distribution array with respect to the correct merge matrix, the merge matrix we used to generate the data, gives the parameter estimates shown in Table 6.16 with log likelihood ratio value equal to $2.54e-10$.

Parameters	Estimates
$\hat{\pi}_X$	0.250, 0.250, 0.250
$\hat{\pi}_Y$	0.250, 0.250, 0.250
$\hat{f}_0$	0.100, 0.100, 0.100
$\hat{s}$-vector	0.200, 0.350, 0.790, 0.010, 0.930, 0.470
$\hat{\rho}$	1.000
height	0.100, 0.500, 0.800

TABLE 6.16. Estimates of parameters of the exact joint distribution for five matched sequences evolved under non-stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Example 6.10. Using the joint distribution array $F(t)$ calculated in the previous example we simulated a data set of size $n = 1000$ and found the observed divergence array N. Applying the statistical tests described in Chapter 4 on this data gives no significant difference, and applying the `hclust` function from the distance method gives the exact merge matrix we used to calculate the joint distribution array $F(t)$.

Using this merge matrix and the observed divergence array, N, we calculated the parameter estimates under this model using the `optim` function, see Table 6.17, with log likelihood ratio equal to 0.197.

Parameters	Estimates
$\hat{\pi}_X$	0.233, 0.304, 0.238
$\hat{\pi}_Y$	0.228, 0.294, 0.222
$\hat{f}_0$	0.095, 0.090, 0.090
$\hat{s}$-vector	0.138, 0.346, 0.945, 0.001, 0.628, 0.504
$\hat{\rho}$	1.078
height	0.094, 0.548, 0.803

TABLE 6.17. Estimates of parameters from data comprising five matched sequences of length $n = 1000$ evolved under non-stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

In the case of non-stationary and homogeneous processes, we can see that we estimate the exact value for the parameters using the exact joint distribution function $F(t)$ and very good estimate using simulated data with sample size $n = 1000$.

6.6.5. Estimation under the Case of Non-Stationary and Non-Homogeneous Processes. In Chapter 4, we showed that in the case of non-stationary processes the statistical tests may give significant differences when we apply them to an observed divergence array, N. Also we showed in Chapter 5 that the hierarchical clustering for the paralinear distance, using the the `hclust` function gives the correct rooted tree topology in the case of homogeneous processes.

For the case of non-stationary and non-homogeneous processes, the paralinear distances for the exact joint distribution function are not ultrametric, so the hierarchical clustering using the `hclust` function can lead to a wrong topology. However, in this case we calculate the marginal distributions and find which group of sequences having the same marginal distributions, so we can connect them together in one side of the root.

Throughout this section to determine the exact tree topology of the exact joint distribution function $F(t)$, we will use both the marginal distributions and the hierarchical clustering for the paralinear distance. However, to determine the exact tree topology for the observed divergence array, N, we will use the statistical tests and the hierarchical clustering for the paralinear distance using the hclust function.

Example 6.11. Consider the case of non-stationary and non-homogeneous processes for a tree with five matched sequences, as described in Example 6.2 with a vector of heights equal to (0.1, 0.5, 0.8, 1.0). Let

$$\pi_X = \pi_Y = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,\ f_0 = (0.1,\ 0.1,\ 0.1\ ,0.7)$$

$R^X = S\Pi_X$ and $R^Y = \rho S\Pi_Y$, where $\rho = 3$ and the six free parameters in the S matrix are given by the

$$s\text{-vector } = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47).$$

Assume that we do not know the correct tree topology. Applying the `hclust` function on the paralinear distance matrix of the joint distribution array for this model gives the incorrect rooted tree topology described by the merge matrix (5.5) and Figure 24. Finding the marginal distribution for the five nodes suggests that the nodes $-1, -2$ and -3 should be on one side of the root and the nodes -4 and -5 on the other side of the root. Using the results from the `hclust` function and the marginal distributions we can conclude that the tree topology should be in the form

$$merge = \begin{bmatrix} -1 & -2 \\ -4 & -5 \\ -3 & 1 \\ 2 & 3 \end{bmatrix}, \tag{6.6}$$

which is the same merge matrix we used to calculate the joint distribution array.

Applying the `optim` function on the joint distribution array with respect to the correct tree topology described by merge matrix (6.6), gives us the parameter estimates shown in Table 6.18, with log likelihood ratio value equal to $3.30e-15$.

Parameters	Estimates
$\hat{\pi}_X$	0.250, 0.250, 0.250
$\hat{\pi}_Y$	0.250, 0.250, 0.250
$\hat{f}_0$	0.100, 0.100, 0.100
$\hat{s}$-vector	0.200, 0.350, 0.790, 0.010, 0.930, 0.470
$\hat{\rho}$	3.00
height	0.1, 0.5, 0.8

Table 6.18. Estimates of parameters of the exact joint distribution for five matched sequences evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$. The estimates were obtained assuming the general Markov model

Now finding the parameter estimates using the merge matrix (5.5), which we got from the paralinear distance method, gives the result in Table 6.19, with log likelihood ratio value equal to 0.045.

Parameters	Estimates
$\hat{\pi}_X$	0.0002, 0.002, 0.001
$\hat{\pi}_Y$	0.268, 0.239, 0.251
$\hat{f}_0$	0.090, 0.113, 0.110
$\hat{s}$-vector	0.152, 0.273, 0.616, 0.001, 0.793, 0.404
$\hat{\rho}$	2.926
height	0.117, 0.228, 0.711

Table 6.19. Estimates of parameters of the exact joint distribution for five matched sequences evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$. The estimates were obtained assuming the general Markov model

We can see that estimation using the merge matrix (5.5) did not give the correct estimates for our parameters.

Example 6.12. Using the joint distribution array $F(t)$ calculated in the previous example we simulated a data set of size $n = 1000$ and found the observed divergence array N. Applying the statistical tests described in Chapter 4 on this data suggested that the nodes $-1, -2$ and -3 should be on one side of the root and the nodes -4 and -5 on the other side of the root. Applying the `hclust` function from the distance method gives the correct unrooted tree topology, so using both results we conclude that we should use the merge matrix (6.6), which is the same as the merge matrix we used to calculate the joint distribution array. Using the observed divergence array N and the merge matrix (6.6), we calculated the parameter estimates under this model using the `optim` function, which gives the result in Table 6.20, with log likelihood ratio value equal to 0.302.

Parameters	Estimates
$\hat{\pi}_X$	0.224, 0.151, 0.264
$\hat{\pi}_Y$	0.213, 0.223, 0.302
$\hat{f}_0$	0.149, 0.156, 0.095
$\hat{s}$-vector	0.325, 0.363, 0.717, 0.009, 1.123, 0.380
$\hat{\rho}$	3.074
height	0.095, 0.536, 0.822

Table 6.20. Estimates of parameters from data comprising five matched sequences of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$. The estimates were obtained assuming the general Markov model

Simulating a larger data set with $n = 10000$ from the same joint distribution and estimated as before gives the result in Table 6.21, with log likelihood ratio value equal to 0.050.

Parameters	Estimates
$\hat{\pi}_X$	0.240, 0.267, 0.246
$\hat{\pi}_Y$	0.247, 0.249, 0.252
$\hat{f}_0$	0.113, 0.095, 0.094
$\hat{s}$-vector	0.214, 0.339, 0.746, 0.012, 0.863, 0.442
$\hat{\rho}$	3.112
height	0.097, 0.524, 0.792

Table 6.21. Estimates of parameters from data comprising five matched sequences of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$. The estimates were obtained assuming the general Markov model

Example 6.13. Consider the case of non-stationary and non-homogeneous processes for five matched sequences as in Example 6.11, but for this case take $\pi_X = (0.1,\ 0.1,\ 0.1,\ 0.7)^T$, $\pi_Y = (0.3,\ 0.3,\ 0.3\ ,0.1)$ and $f_0 = (0.2,\ 0.2,\ 0.2\ ,0.4)$. Let $R^X = S_X\Pi_X$ and $R^Y = \rho S_Y \Pi_Y$, where $\rho = 1$ and the s-vector is given by

$$s\text{-vector } = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47).$$

Applying the `hclust` function on the paralinear distance matrix of the joint distribution array for this model, gives the correct rooted tree topology described by the merge matrix (6.6). Finding the marginal distribution for the five nodes suggests that nodes $-1, -2$ and -3 should be on one side of the root and the nodes -4 and -5 in the other side of the root, this result is in agreement with the result from the distance method. Applying the `optim` function on the joint distribution array with respect to the correct tree topology described by merge matrix (6.6), gives the exact parameter estimates shown in Table 6.22, with log likelihood ratio value equal to $3.19e - 15$.

Example 6.14. Using the joint distribution array $F(t)$ calculated in the previous example, we simulated a data set of size $n = 1000$ and

Parameters	Estimates
$\hat{\pi}_X$	0.100, 0.100, 0.100
$\hat{\pi}_Y$	0.300, 0.300, 0.300
$\hat{f}_0$	0.200, 0.200, 0.200
$\hat{s}$-vector	0.200, 0.350, 0.790, 0.010, 0.930, 0.470
$\hat{\rho}$	1.00
height	0.1, 0.5, 0.8

Table 6.22. Estimates of parameters of the exact joint distribution for five matched sequences evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

found the observed divergence array, N. Applying the statistical tests and the `hclust` function from the paralinear distance method gives the correct tree topology described by the merge matrix (6.6), which is the same merge matrix we used to calculate the joint distribution array. Using the observed divergence array, N, and this merge matrix, we calculated the parameter estimates under this model using the `optim` function, which gives the result in Table 6.23, with log likelihood ratio value equal to 0.302.

Simulating a larger data set with $n = 10000$ from the same joint distribution and estimate as before gives the result in Table 6.24, with log likelihood ratio value equal to 0.034.

Example 6.15. Consider the same tree topology, assumptions, and the same set of parameters as in Example 6.13, but in this case take $\rho = 3$, so the non-homogeneity will come from two reasons, having different stationary distributions for each side of the tree and also having ρ not equal to one on one side of the tree. Applying the `hclust` function on the paralinear distance matrix of the joint distribution array for this model, gives incorrect rooted tree topology described by the merge matrix (5.5). Finding the marginal distribution for the five

Parameters	Estimates
$\hat{\pi}_X$	0.096, 0.072, 0.110
$\hat{\pi}_Y$	0.363, 0.261, 0.267
$\hat{f}_0$	0.160, 0.231, 0.198
$\hat{s}$-vector	0.170, 0.223, 0.662, 0.0117, 0.915, 0.408
$\hat{\rho}$	1.288
height	0.092, 0.570, 0.695

TABLE 6.23. Estimates of parameters from data comprising five matched sequences of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.104, 0.103, 0.095
$\hat{\pi}_Y$	0.298, 0.317, 0.290
$\hat{f}_0$	0.201, 0.204, 0.197
$\hat{s}$-vector	0.186, 0.360, 0.795, 0.007, 0.938, 0.482
$\hat{\rho}$	0.957
height	0.103, 0.480, 0.833

TABLE 6.24. Estimates of parameters from data comprising five matched sequences of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

nodes suggests that nodes $-1, -2$ and -3 should be on one side of the root and the nodes -4 and -5 on the other side of the root. Using the results from the `hclust` function and the statistical tests, we can conclude that the tree topology described by the merge matrix (6.6) is the correct tree topology. Now, applying the `optim` function on the joint distribution array with respect to the correct tree topology described by merge matrix (6.6), gives the exact parameter estimates shown in Table 6.25, with log likelihood ratio value equal to $2.98e-15$.

Parameters	Estimates
$\hat{\pi}_X$	0.100, 0.100, 0.100
$\hat{\pi}_Y$	0.300, 0.300, 0.300
$\hat{f}_0$	0.200, 0.200, 0.200
$\hat{s}$-vector	0.200, 0.350, 0.790, 0.010, 0.930, 0.470
$\hat{\rho}$	3.00
height	0.1, 0.5, 0.8

Table 6.25. Estimates of parameters of the exact joint distribution for five matched sequences evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

Finding the parameter estimates using the merge matrix (5.5), which we get from the paralinear distance method, gives the result in Table 6.26, with log likelihood ratio value equal to 0.380.

Parameters	Estimates
$\hat{\pi}_X$	0.001, 0.0001, 0.008
$\hat{\pi}_Y$	0.271, 0.219, 0.260
$\hat{f}_0$	0.198, 0.273, 0.231
$\hat{s}$-vector	0.141, 0.276, 0.694, 0.001, 0.820, 0.417
$\hat{\rho}$	2.893
height	0.101, 0.150, 0.698

Table 6.26. Estimates of parameters of the exact joint distribution for five matched sequences evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

We can see that estimation using the merge matrix from the `hclust` function did not give the correct estimate for our parameters.

Example 6.16. Using the joint distribution array $F(t)$ calculated in the previous example, we simulated a data set of size $n = 1000$ and found the observed divergence array, N. Applying the statistical tests on the joint distribution array suggested that nodes $-1, -2$ and -3 should be on one side of the root and the nodes -4 and -5 on the other side of the root. Applying the `hclust` function on the paralinear distance matrix of the joint distribution array for this model, gives the incorrect rooted tree topology described by the merge matrix (5.5).

Parameters	Estimates
$\hat{\pi}_X$	0.083, 0.128, 0.074
$\hat{\pi}_Y$	0.292, 0.305, 0.292
$\hat{f}_0$	0.227, 0.209, 0.206
$\hat{s}$-vector	0.194, 0.392, 0.788, 0.001, 0.968, 0.473
$\hat{\rho}$	3.010
height	0.094, 0.526, 0.776

TABLE 6.27. Estimates of parameters from data comprising five matched sequences of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

Using the results from the `hclust` function and the statistical tests, we can conclude that the tree topology described by the merge matrix (6.6) is the correct tree topology. Applying the `optim` function on the observed divergence array N with respect to the correct tree topology described by merge matrix (6.6), gives the parameter estimates shown in Table 6.27, with log likelihood ratio value equal to 0.244.

6.7. Estimation Based on Real Nucleotides Sequences

In this section we find the parameter estimates for two real nucleotide sequences using the log likelihood ratio method.

Example 6.17. Consider the case of the bacterial data described in Example 4.7. We showed in this example that applying the statistical test to this bacterial data, suggested that the divergence matrices for the set *Aquifex, Thermus,* and *Thermotoga* show symmetry, as does the divergence matrix for *Bacillus* and *Deinococcus*, but all divergence matrices for pairs between these sets are highly asymmetric. So we can conclude from the tests that the most parsimonious model must satisfy a lack of stationary, all terminal edges to *Aquifex, Thermus* and *Thermotoga* have the same rate matrix R_1 whereas the terminal edges to *Bacillus* and *Deinococcus* have the same rate matrix R_2 and $R_1 \neq R_2$. Now, since the gn program allows us to have different stationary distribution for the two sides of the root, we will put the set *Aquifex, Thermus,* and *Thermotoga* on one side of the root and the set *Bacillus* and *Deinococcus* on the other side.

In Example 5.8, we showed that the `hclust` function for the paralinear distance matrix of this bacterial data gives a tree topology described by the merge matrix which as we can see suggested that the nodes $-1, -4, -5$ are on side of the root and the nodes $-2, -3$ are on the other side, where node -1 represents *Aquifex*, -2 represents *Bacillus*, -3 represents *Deinococcus*, -4 represents *Thermus* and node -5 represents *Thermotoga*.

$$merge = \begin{bmatrix} -4 & -5 \\ -1 & 1 \\ -2 & -3 \\ 2 & 3 \end{bmatrix}, \tag{6.7}$$

We can see that both the statistical tests and the distance method suggested the same tree topology described by the merge matrix (6.7), see Figure 25.

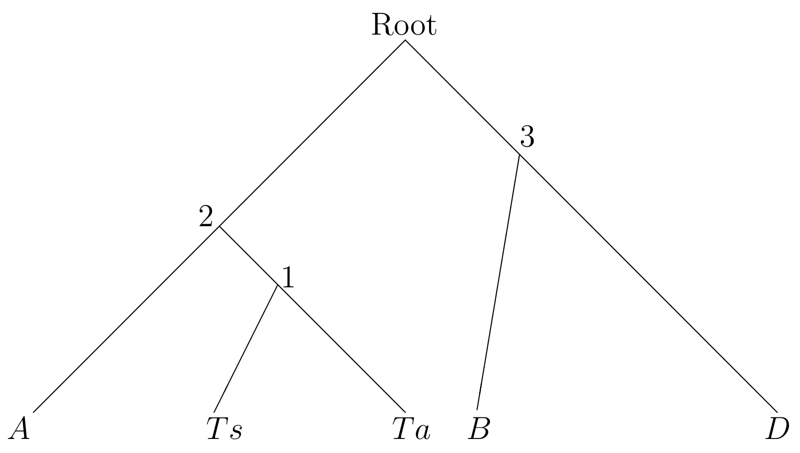


Figure 25. Bacterial data A: *Aquifex*, B: *Bacillus*, D: *Deinococcus*, Ts: *Thermus*, Ta: *Thermotoga*

Maximizing the log likelihood ratio using the `optim` function, using the observed divergence array, N, for this bacterial data and with respect the merge matrix (6.7), gives the parameter estimates shown in Table 6.28, with log likelihood ratio value equal to 0.332.

Parameters	Estimates
$\hat{\pi}_X$	0.200, 0.293, 0.328
$\hat{\pi}_Y$	0.413, 0.097, 0.149
$\hat{f}_0$	0.216, 0.279, 0.358
$\hat{s}$-vector	0.067, 0.135, 0.056, 0.221, 0.235, 0.087
$\hat{\rho}$	1.710
height	0.872, 0.999, 0.681

Table 6.28. Estimates of the bacterial data parameters. The estimates were obtained assuming the general Markov model

Discussion of the accuracy for this data will be given in a later section, when we discuss bootstrap methods.

Example 6.18. Consider the hominoid data described in Chapter 4 Example 4.9. We showed earlier in Chapter 4 that none of the three statistical tests indicate a lack of symmetry between the genes of *Human*, *Chimpanzee*, *Bonobo*, *Gorilla*, *Orangutan*, *Gibbon* and *Macaque*, which indicates that the model for this data satisfy the assumptions of stationarity and homogeneity. Finding the marginal distribution divided by the sequence length (1206) gives the following results: *Chimpanzee*: (0.284, 0.277, 0.136, 0.303), *Bonobo*: (0.279, 0.281, 0.141, 0.299), *Human*: (0.278, 0.287, 0.142, 0.294), *Gorilla*: (0.273, 0.281, 0.144, 0.302), *Gibbon*: (0.288, 0.288, 0.136, 0.289), *Orangutan*: (0.278, 0.299, 0.138, 0.285) and *Macaque*: (0.290, 0.280, 0.139, 0.290). From these marginal distributions we can see that all of them have approximately the same stationary distribution, which supports the idea that this data satisfies stationarity and homogeneity assumptions. In Example 5.9 the `hclust` function for the paralinear distance matrix of this hominoid data gave a tree topology described by the merge matrix (6.8), where node -1 represents *Human*, -2 represents *Chimpanzee*, -3 represents *Bonobo*, -4 represents *gorilla*, node -5 represents *Orangutan*, node -6 represents *Gibbon* and node -7 represents *Macaque*.

$$merge = \begin{bmatrix} -2 & -3 \\ -1 & 1 \\ -4 & 2 \\ -6 & 3 \\ -5 & 4 \\ -7 & 5 \end{bmatrix}, \tag{6.8}$$

We can see that both the marginal distributions and the distance method suggested the same tree topology described by merge matrix (6.8), see Figure 26.

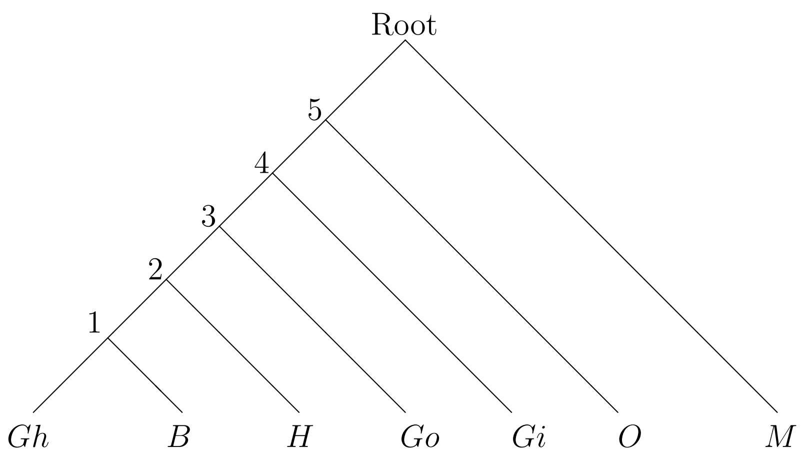


Figure 26. Hominoid data tree topology H: *Human*, Gh: *Chimpanzee*, B: *Bonobo*, Go: *Gorilla*, O: *Orangutan*, Gi: *Gibbon* M: *Macaque*

Maximizing the log likelihood ratio under the general Markov model and using the `optim` function using the observed divergence array N for this hominoid data and with respect the merge matrix (6.8), gave the parameter estimates shown in Table 6.29, with log likelihood ratio value equal to 0.2586.

Parameters	Estimates
$\hat{\pi}_X$	0.198, 0.285, 0.081
$\hat{\pi}_Y$	0.166, 0.161, 0.301
$\hat{f}_0$	0.298, 0.299, 0.129
$\hat{s}$-vector	0.099, 0.246, 0.042, 0.029, 0.285, 0.004
$\hat{\rho}$	1.016
height	0.181, 0.326, 0.421, 0.650, 0.660

Table 6.29. Estimates of the hominoid data parameters. The estimates were obtained assuming the general Markov model

Under the stationary and homogeneous model, we have 14 free parameters. Maximizing the log likelihood ratio using the `optim` function depending on the observed divergence array N for this hominoid data and with respect the merge matrix (6.8), gave the parameter estimates shown in Table 6.30, with log likelihood ratio value equal to 0.278.

Parameters	Estimates
$\hat{\pi}$	0.274, 0.275, 0.150
$\hat{s}$-vector	0.063, 0.327, 0.041, 0.038, 0.274, 0.012
height	0.179, 0.312, 0.419, 0.660, 0.679

Table 6.30. Estimates of the hominoid data parameters. The estimates were obtained assuming the general time-reversible Markov model

In this case we would have expected, on the basis of the statistical tests of symmetry, that π_X, π_Y and f_0 would have similar estimates but this is not the case. In addition the difference between the two log likelihood ratios, 0.2586 and 0.278, is larger than would be expected if the general time-reversible Markov model fitted the data. We conjecture that although the marginal probabilities at all the leaves are homogeneous the model assuming stationarity and homogeneity does not fit the data.

6.8. Bootstrap

The bootstrap, developed by Efron (1979), is a computer-based method frequently used to assess the accuracy of many statistical estimates.

6.8.1. Non-Parametric Bootstrap.

6.8.1. Non-Parametric Bootstrap. Felsenstein (1985) describes the method of the bootstrap as the following. Start with data points $x_1, \cdots, x_n$, which are assumed to be drawn independently from the same distribution. From these values, we can apply our method of statistical estimation to obtain an estimate for the parameter of interest. If the exact distribution of the data was known and if our function were tractable, an estimate of the standard deviation could be obtained. Otherwise, the bootstrap is a useful alternative. The bootstrap procedure suggests that the data be resampled to construct several pseudo data sets. Each of the bootstrap data sets is constructed by resampling n points from the original data with replacement. For each bootstrap data sets, we compute the estimates of interest. The standard deviation of the estimates of interest can be calculated from the bootstrapped estimates. A non-parametric bootstrap sample is created from the original alignment by sampling the columns of the alignment with replacement. Thus, the new alignment may have several repeated columns and several missing columns from the original alignment. The main advantage of the non-parametric bootstrap is that it does not base its replicates on any assumptions about the model but only that the sites are independent and identically distributed.

Example 6.19. Consider the case of non-stationary and homogeneous processes for a tree with five matched sequences, as described in Example 6.2. Let the vector of heights equal to $(0.1, 0.5, 0.8, 1.0)$ and take

$$\pi_X = \pi_Y = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,\ f_0 = (0.1,\ 0.1,\ 0.1\ ,0.7)^T$$

$R^X = R^Y = S\Pi$, where the six free parameters in the S matrix are given by the

$$s\text{-vector } = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47).$$

Simulating a data set of size $n = 1000$ from the joint distribution $F(t)$ of this model and finding the parameter estimates for the observed divergence array N gives the result shown in Table 6.31, with log likelihood ratio equal to 0.201

From the observed divergence array N, we sample with replacement 20 data sets each of size $n = 1000$. For each data set we estimate the parameters under the general Markov model using the `optim` function.

Parameters	Estimates
$\hat{\pi}_X$	0.220, 0.240, 0.240
$\hat{\pi}_Y$	0.252, 0.269, 0.256
$\hat{f}_0$	0.076, 0.110, 0.081
$\hat{s}$-vector	0.198, 0.348, 0.787, 0.0003, 0.927, 0.470
$\hat{\rho}$	0.989
height	0.087, 0.520, 0.782

TABLE 6.31. Estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

The mean and the standard deviation of these estimates are given in Table 6.32. The mean of the log likelihood ratio is 0.311 with a standard deviation 0.025.

Simulating a larger data set with $n = 10000$ from the same joint distribution function $F(t)$ and finding the parameter estimates for the observed divergence array N gives the result shown in Table 6.33, with log likelihood ratio equal to 0.038.

From the observed divergence array, N, we sampled with replacement 20 data sets each of size $n = 10000$. The mean and the standard deviation for the parameter estimates of these data sets is given in Table 6.34. The log likelihood ratio mean is equal to 0.0637 with standard deviation 0.0024.

We note that in the case of $n = 1000$ the parameter estimates are reasonable and are very accurate when the sample size $n = 10000$. Also, we can see that n times the log likelihood ratio is smaller for the case when $n = 1000$ than the case when $n = 10000$. Further, the log likelihood ratio for the bootstrap data sets is 50% greater than that for the simulated data.

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.195, 0.328, 0.216
sd of $\hat{\pi}_X$	0.053, 0.131, 0.074
mean of $\hat{\pi}_Y$	0.185, 0.322, 0.214
sd of $\hat{\pi}_Y$	0.053, 0.124, 0.061
mean of $\hat{f}_0$	0.095, 0.086, 0.087
sd of $\hat{f}_0$	0.022, 0.020, 0.014
mean of $\hat{s}$-vector	0.124, 0.379, 1.068, 0.003, 0.599, 0.561
sd $\hat{s}$-vector	0.056, 0.127, 0.358, 0.006, 0.183, 0.230
mean of $\hat{\rho}$	1.150
sd of $\hat{\rho}$	0.174
mean of *height*	0.091, 0.564, 0.788
sd of *height*	0.011, 0.053, 0.047

TABLE 6.32. Non-parametric estimates of parameters from simulated data of length n=1000 evolved under non-stationary and non-homogeneous processes. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.258, 0.222, 0.274
$\hat{\pi}_Y$	0.265, 0.219, 0.292
$\hat{f}_0$	0.092, 0.109, 0.096
$\hat{s}$-vector	0.222, 0.332, 0.755, 0.0006, 1.046, 0.442
$\hat{\rho}$	0.921
height	0.108, 0.489, 0.832

TABLE 6.33. Estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.254, 0.244, 0.264
sd of $\hat{\pi}_X$	0.017, 0.016, 0.014
mean of $\hat{\pi}_Y$	0.253, 0.235, 0.275
sd of $\hat{\pi}_Y$	0.014, 0.011, 0.013
mean of $\hat{f}_0$	0.092, 0.105, 0.098
sd of $\hat{f}_0$	0.005, 0.006, 0.004
mean of $\hat{s}$-vector	0.206, 0.346, 0.759, 0.001, 0.956, 0.456
sd $\hat{s}$-vector	0.016, 0.024, 0.040, 0.002, 0.061, 0.020
mean of $\hat{\rho}$	0.973
sd of $\hat{\rho}$	0.054
mean of *height*	0.107, 0.497, 0.817
sd of *height*	0.005, 0.011, 0.023

TABLE 6.34. Non-parametric estimates of parameters from simulated data of length n=10000 evolved under non-stationary and non-homogeneous processes. The estimates were obtained assuming the general Markov model

Example 6.20. Consider the case of non-stationary and non-homogeneous processes for five matched sequences as in the previous example. Take the height vector equal to $(0.1, 0.5, 0.8, 1.0)$ and let

$$\pi_X = \pi_Y = (0.25,\ 0.25,\ 0.25,\ 0.25)^T,\ f_0 = (0.1,\ 0.1,\ 0.1\ ,0.7)^T.$$

Let $R^X = S\Pi_X$ and $R^Y = \rho S\Pi_Y$, where the six free parameters in the S matrix are given by the

$$s\text{-vector } = (0.2,\ 0.35,\ 0.79,\ 0.01,\ 0.93,\ 0.47),$$

and $\rho = 3$. Simulating a data set of size $n = 1000$ from the joint distribution $F(t)$ and finding the parameter estimates for the observed divergence array N gives the result shown in Table 6.35, with log likelihood ratio equal to 0.302.

Parameters	Estimates
$\hat{\pi}_X$	0.224, 0.151, 0.264
$\hat{\pi}_Y$	0.212, 0.223, 0.303
$\hat{f}_0$	0.149, 0.192, 0.087
$\hat{s}$-vector	0.325, 0.363, 0.717, 0.009, 1.124, 0.375
$\hat{\rho}$	3.075
height	0.095, 0.536, 0.822

TABLE 6.35. Estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model.

Sample with replacement 20 data sets each of size $n = 1000$ from the observed divergence array N. Estimate the parameters under the general Markov model using the `optim` function for each data set. The mean and the standard deviation of these estimates are given in Table 6.36. The mean of the log likelihood ratio is 0.458 with a standard deviation 0.017.

Simulating a larger data set with $n = 10000$ from the same joint distribution function $F(t)$ and finding the parameter estimates for the observed divergence array N gives the result shown in Table 6.37, with log likelihood ratio equal to 0.050.

From the observed divergence array N, we sample with replacement 20 data sets each of size $n = 10000$. The mean and the standard deviation for the parameter estimates of these data sets is given in Table 6.38. The log likelihood ratio mean is equal to 0.084 with standard deviation 0.0031.

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.246, 0.157, 0.275
sd of $\hat{\pi}_X$	0.102, 0.082, 0.064
mean of $\hat{\pi}_Y$	0.217, 0.213, 0.323
sd of $\hat{\pi}_Y$	0.034, 0.046, 0.069
mean of $\hat{f}_0$	0.140, 0.209, 0.080
sd of $\hat{f}_0$	0.074, 0.112, 0.018
mean of $\hat{s}$-vector	0.385, 0.367, 0.801, 0.018, 1.320, 0.416
sd $\hat{s}$-vector	0.206, 0.094, 0.260, 0.029, 0.551, 0.105
mean of $\hat{\rho}$	2.893
sd of $\hat{\rho}$	1.028
mean of *height*	0.098, 0.516, 0.827
sd of *height*	0.014, 0.129, 0.080

TABLE 6.36. Non-parametric estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.241, 0.269, 0.246
$\hat{\pi}_Y$	0.247, 0.249, 0.252
$\hat{f}_0$	0.131, 0.095, 0.094
$\hat{s}$-vector	0.214, 0.339, 0.746, 0.017, 0.862, 0.441
$\hat{\rho}$	3.154
height	0.097, 0.525, 0.756

TABLE 6.37. Estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.244, 0.274, 0.241
sd of $\hat{\pi}_X$	0.013, 0.017, 0.017
mean of $\hat{\pi}_Y$	0.248, 0.250, 0.248
sd of $\hat{\pi}_Y$	0.006, 0.010, 0.008
mean of $\hat{f}_0$	0.111, 0.094, 0.095
sd of $\hat{f}_0$	0.009, 0.014, 0.006
mean of $\hat{s}$-vector	0.208, 0.339, 0.735, 0.014, 0.854, 0.453
sd $\hat{s}$-vector	0.0257, 0.043, 0.069, 0.010, 0.070, 0.054
mean of $\hat{\rho}$	3.202
sd of $\hat{\rho}$	0.472
mean of *height*	0.097, 0.531, 0.762
sd of *height*	0.007, 0.043, 0.037

TABLE 6.38. Non-parametric estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

We can see the accuracy of the parameter estimates is increased when we increase the sample size from $n = 1000$ to $n = 10000$. Again, as in the previous example, we can can see that n times the log likelihood ratio is smaller for the case when $n = 1000$ than the case when $n = 10000$ and the log likelihood ratio for the bootstrap data sets is 50% greater than that for the simulated data.

Example 6.21. Consider the same case of non-stationarity and non-homogeneity for five matched sequences as in the previous example, but in this case let the non-homogeneity come from having different stationary distribution for each side of the tree root by taking

$$\pi_X = (0.1,\ 0.1,\ 0.1,\ 0.7)^T,\ \pi_Y = (0.3,\ 0.3,\ 0.3\ ,0.1)^T,$$

$$f_0 = (0.2,\ 0.2,\ 0.2,\ 0.4)^T,$$

$R^X = S_X \Pi_X$ and $R^Y = S_Y \Pi_Y$. From the joint distribution $F(t)$, we simulate a data set of size $n = 1000$. Using the `optim` optimizer, we find the parameter estimates for the observed divergence array N as we can see in Table 6.39, with log likelihood ratio equal to 0.178.

Now, we sample with replacement 20 data sets each of size $n = 1000$ from the observed divergence array N. For each data set we estimate the parameters under the general Markov model using the optimizer function `optim`. The mean and the standard deviation of these estimates are given in Table 6.40. The mean of the log likelihood ratio is 0.256 with a standard deviation 0.019.

For a larger simulated data set with $n = 10000$, the parameter estimates for the observed divergence array N are given in Table 6.41, with log likelihood ratio equal to 0.034. From the observed divergence array N, we sampled with replacement 20 data sets each of size $n = 10000$. The mean and the standard deviation for the parameters estimates of these data sets is given in Table 6.42. The log likelihood ratio mean is equal to 0.055 with standard deviation 0.0025.

Comparing the results in Tables 6.40 and 6.42, we can see that we get more accurate estimates for the case of $n = 10000$. Also, we can see that we had relatively small errors in the case of $n = 1000$ and very small errors for the case when $n = 10000$. The relation between the log likelihood ratios commented in Examples 6.19 and 6.20 holds here also.

Parameters	Estimates
$\hat{\pi}_X$	0.096, 0.072, 0.110
$\hat{\pi}_Y$	0.363, 0.261, 0.267
$\hat{f}_0$	0.160, 0.232, 0.198
$\hat{s}$-vector	0.172, 0.223, 0.662, 0.012, 0.915, 0.408
$\hat{\rho}$	1.288
height	0.091, 0.570, 0.694

TABLE 6.39. Estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.101, 0.071, 0.106
sd of $\hat{\pi}_X$	0.031, 0.025, 0.023
mean of $\hat{\pi}_Y$	0.364, 0.253, 0.174
sd of $\hat{\pi}_Y$	0.057, 0.047, 0.042
mean of $\hat{f}_0$	0.168, 0.238, 0.196
sd of $\hat{f}_0$	0.027, 0.022, 0.020
mean of $\hat{s}$-vector	0.202, 0.238, 0.687, 0.014, 0.426, 0.561
sd $\hat{s}$-vector	0.051, 0.066, 0.142, 0.015, 0.157, 0.067
mean of $\hat{\rho}$	1.204
sd of $\hat{\rho}$	0.449
mean of *height*	0.097, 0.544, 0.756
sd of *height*	0.020, 0.120, 0.112

TABLE 6.40. Non-parametric estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.104, 0.103, 0.095
$\hat{\pi}_Y$	0.298, 0.317, 0.290
$\hat{f}_0$	0.202, 0.204, 0.197
$\hat{s}$-vector	0.186, 0.360, 0.795, 0.007, 0.938, 0.482
$\hat{\rho}$	0.957
height	0.103, 0.480, 0.833

Table 6.41. Estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.104, 0.103, 0.093
sd of $\hat{\pi}_X$	0.006, 0.005, 0.006
mean of $\hat{\pi}_Y$	0.297, 0.314, 0.292
sd of $\hat{\pi}_Y$	0.013, 0.019, 0.015
mean of $\hat{f}_0$	0.201, 0.205, 0.197
sd of $\hat{f}_0$	0.007, 0.008, 0.005
mean of $\hat{s}$-vector	0.192, 0.360, 0.790, 0.007, 0.931, 0.476
sd $\hat{s}$-vector	0.016, 0.031, 0.039, 0.004, 0.066, 0.031
mean of $\hat{\rho}$	0.978
sd of $\hat{\rho}$	0.107
mean of *height*	0.102, 0.483, 0.830
sd of *height*	0.008, 0.024, 0.048

Table 6.42. Non-parametric estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

Example 6.22. Consider the parameter estimates for the bacterial data in Example 6.17. In this example we found the parameter estimates under the general Markov model, as we can see in Table 6.43 with a log likelihood ratio value equal to to 0.332. In order to see if the general Markov model fits this data we applied the non-parametric bootstrap technique on this data, by sampling from the original bacterial data with replacement 20 data sets each of the same size as the original data set, that is $n = 1238$. For each data set we estimated the parameters under the general Markov model using the optimizer function `optim`. The mean and the standard deviation of these estimates are given in Table 6.44, with log likelihood ratio mean equal to 0.420 and a standard deviation 0.028.

In this case the log likelihood ratio for the bootstrap data sets is larger than that for the original data by 25% (0.420, 0.332). This is less than the 50% noted for simulated data of similar sizes.

Parameters	Estimates
$\hat{\pi}_X$	0.200, 0.293, 0.328
$\hat{\pi}_Y$	0.413, 0.097, 0.149
$\hat{f}_0$	0.216, 0.279, 0.358
$\hat{s}$-vector	0.067, 0.135, 0.056, 0.221, 0.235, 0.087
$\hat{\rho}$	1.710
height	0.872, 0.999, 0.681

Table 6.43. Estimates of the bacterial data parameters. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.227, 0.263, 0.334
sd of $\hat{\pi}_X$	0.074, 0.035, 0.048
mean of $\hat{\pi}_Y$	0.437, 0.085, 0.142
sd of $\hat{\pi}_Y$	0.102, 0.015, 0.027
mean of $\hat{f}_0$	0.217, 0.274, 0.359
sd of $\hat{f}_0$	0.010, 0.011, 0.013
mean of $\hat{s}$-vector	0.066, 0.130, 0.056, 0.234, 0.252, 0.092
sd $\hat{s}$-vector	0.021, 0.038, 0.011, 0.029, 0.070, 0.026
mean of $\hat{\rho}$	1.781
sd of $\hat{\rho}$	0.135
mean of *height*	0.845, 0.999, 0.682
sd of *height*	0.034, 0.0003, 0.032

TABLE 6.44. Non-parametric estimates of the bacterial data parameters. The estimates were obtained assuming the general Markov model

Example 6.23. Consider the parameter estimates for the hominoid data in Example 6.18. In this example we found the parameter estimates under the general time-reversible Markov model as we can see in Table 6.45 with a log likelihood ratio value equal to to 0.278. In order to see if general time-reversible Markov model fits this data we applied the non-parametric bootstrap technique on this data, by sampling from the original hominoid data with replacement 20 data sets each of the same size as the original data set, that is $n = 1206$. For each data set we estimated the parameters under the general time-reversible Markov model using the optimizer function `optim`. The mean and the standard deviation of these estimates are calculated and given in Table 6.46, with log likelihood ratio mean equal to 0.332 and a standard deviation 0.035.

Parameters	Estimates
$\hat{\pi}$	0.274, 0.275, 0.150
$\hat{s}$-vector	0.063, 0.327, 0.041, 0.038, 0.274, 0.012
height	0.179, 0.312, 0.419, 0.660, 0.679

Table 6.45. Estimates of the hominoid data parameters. The estimates were obtained assuming the general time-reversible Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}$	0.271, 0.276, 0.153
sd of $\hat{\pi}$	0.012, 0.011, 0.012
mean of $\hat{s}$-vector	0.060, 0.330, 0.043, 0.037, 0.274, 0.010
sd $\hat{s}$-vector	0.007, 0.045, 0.008, 0.012, 0.026, 0.007
mean of *height*	0.168, 0.312, 0.425, 0.666, 0.700
sd of *height*	0.022, 0.039, 0.034, 0.053, 0.073

Table 6.46. Non-parametric estimates of the hominoid data parameters. The estimates were obtained assuming the general time-reversible Markov model

Here, the bootstrap log likelihood ratio is approximately 20% larger than that for the hominoid data set (0.332, 0.278). Again this is not as great as the 50% noted for simulated data. The bootstrap indicates that the estimates are unbiased and have relatively small errors.

6.8.2. Parametric Bootstrap. Unlike the non-parametric bootstrap, the parametric bootstrap creates new samples (replicates) by simulation involving maximum likelihood estimates. First, a model of evolution is assumed. Then the parameters of the model are estimated from the data. Then we use these parameter estimates to generate a joint distribution array. Using this joint distribution array we generate several bootstrap data sets. Each of the bootstrap data sets is constructed by sampling n points from the joint distribution array. For each bootstrap data sets we compute the estimates of interest. The mean and standard deviation of the estimates of interest can be calculated from the bootstrapped estimates.

Note: To make it easy to compare between the parameter estimates and the parametric bootstrap estimates, we will repeat the tables of the parameter estimates for the observed divergence arrays.

Example 6.24. Consider the case of the observed divergence array N of size $n = 1000$ given in Example 6.19. Depending on the parameter estimates of this observed divergence array N, shown below in Table 6.47 with log likelihood ratio equal to 0.201.

We calculate the joint distribution array $F(t)$, then from this joint distribution array, we simulate 20 data sets, each of size $n = 1000$. The mean and the standard deviation for the parameter estimates of the 20 data sets are calculated and the result are given in Table 6.48. The log likelihood ratio mean is equal to 0.215 with standard deviation 0.013.

Now consider the observed divergence array N of size $n = 10000$ given in Example 6.19 and the parameter estimates shown in Table 6.49, with log likelihood ratio equal to 0.038. Using these parameter estimates we calculate the joint distribution array $F(t)$. From this joint distribution array we simulate 20 data sets each of size $n = 10000$ and we find the parameter estimates for each of these data sets using the optimizer function `optim`. The mean and the standard deviation of these parameter estimates are given in Table 6.50 with log likelihood ratio mean equal to 0.039 and standard deviation 0.002.

We note that in the case of $n = 1000$ the parameter estimates are reasonable and are very accurate when the sample size $n = 10000$. Also, we can see that, as may be expected, the log likelihood ratio for the parametric bootstrap is very close to that obtained from the simulated data. Further, the log likelihood ratio for the bootstrap data sets is 70% greater than that for the simulated data.

Parameters	Estimates
$\hat{\pi}_X$	0.220, 0.240, 0.240
$\hat{\pi}_Y$	0.252, 0.269, 0.256
$\hat{f}_0$	0.076, 0.110, 0.081
$\hat{s}$-vector	0.198, 0.348, 0.787, 0.0003, 0.927, 0.470
$\hat{\rho}$	0.989
height	0.087, 0.520, 0.782

Table 6.47. Estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.239, 0.226, 0.242
sd of $\hat{\pi}_X$	0.055, 0.059, 0.050
mean of $\hat{\pi}_Y$	0.264, 0.253, 0.262
sd of $\hat{\pi}_Y$	0.051, 0.059, 0.050
mean of $\hat{f}_0$	0.073, 0.116, 0.081
sd of $\hat{f}_0$	0.022, 0.026, 0.013
mean of $\hat{s}$-vector	0.209, 0.380, 0.820, 0.008, 0.973, 0.485
sd $\hat{s}$-vector	0.071, 0.109, 0.240, 0.016, 0.220, 0.115
mean of $\hat{\rho}$	0.990
sd of $\hat{\rho}$	0.210
mean of *height*	0.095, 0.493, 0.818
sd of *height*	0.015, 0.056, 0.057

Table 6.48. Parametric estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.258, 0.222, 0.274
$\hat{\pi}_Y$	0.265, 0.219, 0.292
$\hat{f}_0$	0.092, 0.109, 0.096
$\hat{s}$-vector	0.222, 0.332, 0.755, 0.0006, 1.046, 0.442
$\hat{\rho}$	0.921
height	0.108, 0.489, 0.832

TABLE 6.49. Estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.263, 0.233, 0.265
sd of $\hat{\pi}_X$	0.019, 0.023, 0.017
mean of $\hat{\pi}_Y$	0.264, 0.227, 0.280
sd of $\hat{\pi}_Y$	0.022, 0.019, 0.019
mean of $\hat{f}_0$	0.093, 0.106, 0.096
sd of $\hat{f}_0$	0.006, 0.007, 0.005
mean of $\hat{s}$-vector	0.210, 0.331, 0.746, 0.003, 1.002, 0.452
sd $\hat{s}$-vector	0.017, 0.020, 0.053, 0.005, 0.096, 0.023
mean of $\hat{\rho}$	0.951
sd of $\hat{\rho}$	0.050
mean of *height*	0.109, 0.496, 0.826
sd of *height*	0.005, 0.012, 0.020

TABLE 6.50. Parametric estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and homogeneous processes. The estimates were obtained assuming the general Markov model

Example 6.25. Consider the case of non-stationary and non-homogeneous sequences discussed in Example 6.20. Using the observed divergence array N of size $n = 1000$ and the parameter estimates of this observed divergence array N, shown in Table 6.51, with log likelihood ratio equal to 0.302, we calculate the joint distribution array $F(t)$. From this joint distribution array, we simulate 20 data sets, each of size $n = 1000$. The mean and the standard deviation for the parameter estimates of the 20 data sets are calculated and the result are shown in Table 6.52. The log likelihood ratio mean is equal to 0.308 with standard deviation 0.016.

Consider the observed divergence array N of size $n = 10000$ given in Example 6.20 and the parameter estimates shown in Table 6.53, with log likelihood ratio equal to 0.050.

Using these parameter estimates we calculate the joint distribution array $F(t)$. From this joint distribution array we simulate 20 data sets each of size $n = 10000$ and we find the parameter estimates for each of these data sets using the optimizer function `optim`. The mean and the standard deviation of these parameter estimates are given in Table 6.54 with log likelihood ratio mean equal to 0.047 and standard deviation 0.002.

Comparing Tables 6.52 and 6.54, we can see that when we increase the sample size, from $n = 1000$ to $n = 10000$, we got more accurate estimates. Also, the mean of n times the log likelihood ratio increases when we increase the sample size.

Parameters	Estimates
$\hat{\pi}_X$	0.224, 0.151, 0.264
$\hat{\pi}_Y$	0.212, 0.223, 0.303
$\hat{f}_0$	0.149, 0.192, 0.087
$\hat{s}$-vector	0.325, 0.363, 0.717, 0.009, 1.124, 0.375
$\hat{\rho}$	3.075
height	0.095, 0.536, 0.822

TABLE 6.51. Estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.223, 0.166, 0.265
sd of $\hat{\pi}_X$	0.055, 0.060, 0.063
mean of $\hat{\pi}_Y$	0.215, 0.226, 0.299
sd of $\hat{\pi}_Y$	0.029, 0.029, 0.038
mean of $\hat{f}_0$	0.147, 0.191, 0.092
sd of $\hat{f}_0$	0.040, 0.068, 0.017
mean of $\hat{s}$-vector	0.293, 0.335, 0.676, 0.013, 1.010, 0.372
sd $\hat{s}$-vector	0.112, 0.103, 0.195, 0.020, 0.352, 0.140
mean of $\hat{\rho}$	3.719
sd of $\hat{\rho}$	1.417
mean of *height*	0.097, 0.621, 0.793
sd of *height*	0.018, 0.195, 0.080

TABLE 6.52. Parametric estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.241, 0.269, 0.246
$\hat{\pi}_Y$	0.247, 0.249, 0.252
$\hat{f}_0$	0.131, 0.095, 0.094
$\hat{s}$-vector	0.214, 0.339, 0.746, 0.017, 0.862, 0.441
$\hat{\rho}$	3.154
height	0.097, 0.525, 0.756

TABLE 6.53. Estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.252, 0.270, 0.250
sd of $\hat{\pi}_X$	0.022, 0.020, 0.018
mean of $\hat{\pi}_Y$	0.250, 0.248, 0.249
sd of $\hat{\pi}_Y$	0.007, 0.011, 0.008
mean of $\hat{f}_0$	0.110, 0.095, 0.093
sd of $\hat{f}_0$	0.012, 0.014, 0.006
mean of $\hat{s}$-vector	0.201, 0.318, 0.709, 0.014, 0.841, 0.435
sd $\hat{s}$-vector	0.024, 0.027, 0.073, 0.006, 0.067, 0.038
mean of $\hat{\rho}$	3.331
sd of $\hat{\rho}$	0.401
mean of *height*	0.096, 0.543, 0.751
sd of *height*	0.007, 0.038, 0.031

TABLE 6.54. Parametric estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X = \pi_Y$ and $\rho = 3$. The estimates were obtained assuming the general Markov model

Example 6.26. For the case of non-stationary and non-homogeneous sequences given in Example 6.21. Using the parameter estimates of the observed divergence array, N, of size $n = 1000$, shown in Table 6.55, with log likelihood ratio equal to 0.178, we calculate the joint distribution array $F(t)$.

Parameters	Estimates
$\hat{\pi}_X$	0.096, 0.072, 0.110
$\hat{\pi}_Y$	0.363, 0.261, 0.267
$\hat{f}_0$	0.160, 0.232, 0.198
$\hat{s}$-vector	0.172, 0.223, 0.662, 0.012, 0.915, 0.408
$\hat{\rho}$	1.288
height	0.091, 0.570, 0.694

TABLE 6.55. Estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

From this joint distribution array we simulate 20 data sets, each of size $n = 1000$. The mean and the standard deviation for the parameter estimates of the 20 data sets are calculated and the result are shown in Table 6.56. The log likelihood ratio mean is equal to 0.171 with standard deviation 0.011.

Referring to the observed divergence array N of size $n = 10000$ given in Example 6.21 and the parameter estimates shown in Table 6.57, with log likelihood ratio equal to 0.034, and using these parameter estimates we calculate the joint distribution array $F(t)$. From this joint distribution array we simulate 20 data sets each of size $n = 10000$. The mean and the standard deviation for the parameter estimates of these 20 data sets have been calculated, as we can see in Table 6.58 with log likelihood ratio mean equal to 0.033 and standard deviation 0.002.

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.093, 0.076, 0.119
sd of $\hat{\pi}_X$	0.025, 0.020, 0.032
mean of $\hat{\pi}_Y$	0.359, 0.269, 0.265
sd of $\hat{\pi}_Y$	0.056, 0.050, 0.041
mean of $\hat{f}_0$	0.164, 0.226, 0.197
sd of $\hat{f}_0$	0.024, 0.028, 0.017
mean of $\hat{s}$-vector	0.166, 0.237, 0.660, 0.011, 0.894, 0.390
sd $\hat{s}$-vector	0.059, 0.070, 0.167, 0.012, 0.228, 0.095
mean of $\hat{\rho}$	1.468
sd of $\hat{\rho}$	0.621
mean of *height*	0.091, 0.606, 0.701
sd of *height*	0.020, 0.168, 0.118

TABLE 6.56. Parametric estimates of parameters from simulated data of length $n = 1000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

Parameters	Estimates
$\hat{\pi}_X$	0.104, 0.103, 0.095
$\hat{\pi}_Y$	0.298, 0.317, 0.290
$\hat{f}_0$	0.202, 0.204, 0.197
$\hat{s}$-vector	0.186, 0.360, 0.795, 0.007, 0.938, 0.482
$\hat{\rho}$	0.957
height	0.103, 0.480, 0.833

TABLE 6.57. Estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.104, 0.102, 0.095
sd of $\hat{\pi}_X$	0.006, 0.005, 0.007
mean of $\hat{\pi}_Y$	0.299, 0.316, 0.290
sd of $\hat{\pi}_Y$	0.008, 0.014, 0.009
mean of $\hat{f}_0$	0.204, 0.207, 0.198
sd of $\hat{f}_0$	0.006, 0.008, 0.006
mean of $\hat{s}$-vector	0.190, 0.369, 0.801, 0.006, 0.943, 0.484
sd $\hat{s}$-vector	0.015, 0.035, 0.047, 0.005, 0.062, 0.038
mean of $\hat{\rho}$	0.944
sd of $\hat{\rho}$	0.125
mean of *height*	0.102, 0.478, 0.841
sd of *height*	0.008, 0.032, 0.063

TABLE 6.58. Parametric estimates of parameters from simulated data of length $n = 10000$ evolved under non-stationary and non-homogeneous processes, where $\pi_X \neq \pi_Y$ and $\rho = 1$. The estimates were obtained assuming the general Markov model

We can notice, as in the previous two examples, that in the case of $n = 1000$ the parameter estimates are reasonable and are very accurate when the sample size $n = 1000$. Also, the log likelihood ratio for the parametric bootstrap is very close to that obtained from the simulated data. However, increasing the sample size causes an increase in n times the log likelihood ratios from 171 to 330.

Example 6.27. In this example we applied the parametric bootstrap technique on the bacterial data. Using the parameter estimates of the bacterial data, given in Table 6.59, with log likelihood ratio value equal to 0.332, we calculate the joint distribution array $F(t)$. From this joint distribution array, we simulate 20 data sets each of size $n = 1238$ and find the parameter estimates for each one of these data sets under the general model. The mean and standard deviation of these estimates are given in Table 6.60, with log likelihood ratio mean equal to 0.158 and standard deviation 0.012.

Parameters	Estimates
$\hat{\pi}_X$	0.200, 0.293, 0.328
$\hat{\pi}_Y$	0.413, 0.097, 0.149
$\hat{f}_0$	0.216, 0.279, 0.358
$\hat{s}$-vector	0.067, 0.135, 0.056, 0.221, 0.235, 0.087
$\hat{\rho}$	1.710
height	0.872, 0.999, 0.681

TABLE 6.59. Estimates of the bacterial data parameters. The estimates were obtained assuming the general Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}_X$	0.201, 0.294, 0.317
sd of $\hat{\pi}_X$	0.029, 0.029, 0.022
mean of $\hat{\pi}_Y$	0.419, 0.087, 0.140
sd of $\hat{\pi}_Y$	0.043, 0.015, 0.025
mean of $\hat{f}_0$	0.218, 0.276, 0.361
sd of $\hat{f}_0$	0.010, 0.014, 0.015
mean of $\hat{s}$-vector	0.074, 0.142, 0.055, 0.238, 0.242, 0.091
sd $\hat{s}$-vector	0.011, 0.018, 0.010, 0.024, 0.025, 0.017
mean of $\hat{\rho}$	1.657
sd of $\hat{\rho}$	0.189
mean of *height*	0.842, 0.957, 0.702
sd of *height*	0.051, 0.050, 0.054

TABLE 6.60. Parametric estimates of the bacterial data parameters. The estimates were obtained assuming the general Markov model

Note that the log likelihood ratio for the parametric bootstrap, 0.158, is smaller than the log likelihood ratio for the original bacterial data estimate, 0.332, this may indicate that the the model is not sufficiently complex for this data.

Example 6.28. In this example we applied the parametric bootstrap technique on the hominoid data under the general time-reversible Markov model. Using the parameter estimates of the hominoid data, given in Table 6.61, with log likelihood ratio value equal to 0.278, we calculate the joint distribution array $F(t)$. From this joint distribution array, we simulate 20 data sets each of size $n = 1206$ and find the parameter estimates for each one of these data sets under the general time-reversible Markov model. The mean and standard deviation of these estimates are given in Table 6.62, with log likelihood ratio mean equal to 0.151 and standard deviation 0.016.

Note that the log likelihood ratio for the general time-reversible Markov model, 0.278, is larger than the mean of the log likelihood ratios, 0.151, which indicates that we need more complex model to fit this data.

Parameters	Estimates
$\hat{\pi}$	0.274, 0.275, 0.150
$\hat{s}$-vector	0.063, 0.327, 0.041, 0.038, 0.274, 0.012
height	0.179, 0.312, 0.419, 0.660, 0.679

TABLE 6.61. Estimates of the hominoid data parameters. The estimates were obtained assuming the general time-reversible Markov model

Parameters	Average estimates and standard deviation
mean of $\hat{\pi}$	0.270, 0.277, 0.153
sd of $\hat{\pi}$	0.014, 0.009, 0.009
mean of $\hat{s}$-vector	0.062, 0.328, 0.043, 0.037, 0.280, 0.010
sd $\hat{s}$-vector	0.007, 0.026, 0.007, 0.015, 0.020, 0.006
mean of *height*	0.177, 0.313, 0.420, 0.671, 0.687
sd of *height*	0.022, 0.032, 0.046, 0.043, 0.042

TABLE 6.62. Parametric estimates of the hominoid data parameters. The estimates were obtained assuming the general time-reversible Markov model

6.9. Discussion

In the general case, when we have stationary models, we cannot estimate all the parameters under the general Markov model since the same joint distribution can be generated by a family of parameter values satisfying Equations (6.3) and (6.4) (see Examples 6.2-6.4). In the case when stationarity and homogeneity do not hold the estimation of simulated data sets with sample size $n = 1000$ seems to be quite good and very accurate when the sample size $n = 10000$ (see Examples 6.12, 6.14, 6.15 and 6.17).

We obtain estimates for the real data under these models. Both the tests of homogeneity in Chapter 4 and the distance method in Chapter 5 suggested using the general Markov model for the bacterial data, so we fit the general model to the bacterial data, which gives reasonable estimates (see Example 6.17). For the hominoid data, the tests of homogeneity in Chapter 4 suggest that the data is stationary and homogeneous (see Example 6.18). The general Markov model compared to the general time-reversible Markov model showed that the general time-reversible Markov model is not sufficient to fit this data. We showed that the likelihood in the case of the general Markov model is smaller than in the case of the general time-reversible Markov model which indicates that the general time-reversible Markov model is not sufficient to fit the data. Further we note that the hominoid data has quite few cells where any change occurs (exactly 137 out of $4^7 = 16384$) and this may give poor estimation.

We note that the log likelihood ratio for sample size $n = 10000$ is larger than 0.1 of the log likelihood ratio for the sample size $n = 1000$ (see Examples 6.12, 6.14, 6.15 and 6.17). In this case there are many cells of the observed divergence array N which are zero and so contribute zero to the log likelihood ratio, the number of such cells is smaller when the total sample size is $n = 10000$. So the log likelihood ratio increases as we increase n by less than expected by the usual, but inapplicable, asymptotic theory.

Both the non-parametric and parametric bootstrap showed that the estimation method was unbiased and had relatively small errors in the case of $n = 1000$ and very small error for the case when $n = 10000$. However, there are real differences between these two methods, when examining simulated data sets generated under known models. The parametric bootstrap, as may be expected, gave log likelihood ratio close to that obtained from the simulated data. However, the non-parametric bootstrap gave log likelihood ratio values in general 50% larger. In this case we conjecture that sampling from the observed divergence array increases the number of cells which have zero observation and so the model is not as well fitted as for the original simulated data set (see Examples 6.19-6.26). For, any cell that had zero occurrences for the data, will necessarily have zero occurrences in the non-parametric bootstrap sample and other cells with few occurrences will also correspond to zero occurrences in the non-parametric bootstrap sample. Thus the number of cells in the non-parametric bootstrap sample with zero occurrences will be larger than in the original sample. However, for the parametric bootstrap no such bias will exist and cells with zero occurrences in the original sample can have non-zero occurrences in the parametric bootstrap sample. If the model does describe the data then the parametric bootstrap samples will be similar to the original sample and so estimates will be approximately unbiased.

We did not apply the bootstrap methods to the cases where we have stationary processes, since not all the parameters are estimable as we showed in Sections 6.6.1 and 6.6.3.

For the bacterial data the log likelihood ratio for the general model is 0.332, for the non-parametric bootstrap the average of 20 log likelihood ratios is 0.420 and for the parametric bootstrap the average of 20 log likelihood ratios is 0.158. The fact that the log likelihood ratio for the parametric bootstrap is much smaller than for the real data indicates that the model is not sufficiently complex for this data. We note that for simulated data these two log likelihood ratios are approximately the same. The inflation of the log likelihood for the non-parametric bootstrap, which occurs for the simulated data, occurs here also for the real data.

For the hominoid data the log likelihood ratio for the general time-reversible Markov model is 0.278, for the non-parametric bootstrap the average of 20 log likelihood ratios is 0.332 and for the parametric bootstrap the average of 20 log likelihood ratios is 0.151. The fact that the log likelihood ratio for the parametric bootstrap is much smaller than for the real data indicates that the model is not sufficiently complex for this data.

The methods proposed here represent a step toward more realistic models than those in common use, where stationarity, homogeneity and reversibility are assumed. These method are successful for data generated from known models and hence they provide a method of testing such models. This is clear by looking at the bacterial data where we get improved fit, but clearly the model proposed is not sufficient to fit this data. A number of generalizations of these methods is possible but this will involve more parameters than for those involved here. The method could be extended to cases where each edge had different rate matrix and different stationary distribution and to the case where the rate matrices did not have the property of reversible processes. Each of these generalizations, however, involves an increase in the number of parameters, thus increasing numerical difficulties. In addition other techniques need to be developed to deal with rate heterogeneity over sites.

References

[1] Agresti, A. (1990). *Categorical Data Analysis*, Wiley Series in Probability and Mathematical Statistics, New York.

[2] Bhapkar, V. P. (1966). A note on the equivalence of two test criteria for hypotheses in categorical data. *Journal of the American Statistical Association* 61, 228-235.

[3] Bowker, A. H. (1948). A test for symmetry in contingency tables. *Journal of the American Statistical Association* 43, 572-574.

[4] Conant, G. C. and Lewis, P. O. (2001). Effects of nucleotide compositional bias in the success of the parsimony criterion in phylogenetic inference. *Molecular Biology and Evolution* 18, 1024-1033.

[5] Efron, B. (1979). Bootstrap method: Another look at the jackknife. *Annals of statistics* 7, 1-26

[6] Eisen, J., A. (1998). Phylogenomics: Improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Reserch* 8, 163-167.

[7] Felsenstein, J. (1973). Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. *Systematic Zoology* 22, 240-240.

[8] Felsenstein, J. (1981). Evolutionary trees from DNA sequences: A maximum likelihood approach. *Molecular Biology and Evolution* 17, 368-376.

[9] Felsenstein, J. (1985). Confidence limits on phylogenies: An approach using the bootstrap. *Evolution* 39, 783-791.

[10] Felsenstein, J. (1996). Inferring phylogenies from protein sequence by parsimony distance, and likelihood methods. *Methods in Enzymology* 24, 418-427.

[11] Felsenstein, J. (2004a). *Inferring Phylogenies.* Sinauer, Sunderland, Massachusetts, USA.

[12] Felsenstein, J. (2004b). *PHYLIP (Phylogeny Inference Package)*, version 3.62. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

[13] Huelsenbeck, J. P. (1995). Performance of phylogenetic estimation in simulation. *Systematic Biology* 44, 17-48.

[14] Galtier, N. and Gouy, M. (1995). Inferring phylogenies from DNA sequences of unequal base compositions. *Proceedings of the National Academy of Sciences of the United States of America* 92, 11317-11321.

[15] Gaut B. S., Lewis P. O. (1995). Success of maximum likelihood phylogeny inference in the four-taxon case. *Molecular Biology and Evolution* 12, 152-162.

[16] Goldman, N. (1990). Maximum likelihood inference of phylogenetic trees, with special reference to a poisson process model of DNA substitution and to parsimony analysis. *Systematic Zoology* 39, 345-361.

[17] Hasegawa, M., Kishino, H. and Yano, T. (1985). Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. *Journal of Molecular Evoltion* 22, 160-174.

[18] Hillis, D. *et al.* (1996). *Molecular systematics* Sunderland, Chapter 11, pp 447-466.

[19] Ho, S. Y. W. and Jermiin, L. S. (2004). Tracing the decay of the historical signal in biological sequence data. *Systematic Biology* 53, 423-437.

[20] Iosifescu, M. (1980). *Finite Markov processes and their applications* Wiley Series in Probability and Mathematical Statistics, New York.

[21] Ireland, C., Ku, H. H. and Kullback, S. (1969). Symmetry and marginal homogeneity of an $r \times r$ contingency table. *Journal of the American Statistical Association* 64, 1323-1341.

[22] Jermiin, L. S., Ho, S. Y. W, Ababneh, F., Robinson, J. and Larkum, A. W. D. (2003). Hetero: A program to simulate the evolution of DNA on a four-taxon tree. *Applied Bioinformatics* 2, 159-163.

[23] Jermiin, L. S., Ho, S. Y. W., Ababneh, F., Robinson, J. and Larkum, A. W. D. (2004). The biasing effect of compositional heterogeneity on phylogenetic estimates may be underestimated, *Systematic Biology* 53, 638-643.

[24] Jukes, T. H. and Cantor, C. R. (1969). Evolution of protein molecules. In H. N. Munro (Ed.), *Mammalian protein metabolism*, pp. 21-123, Academic Press, New York.

[25] Kelly, F. P. (1979). *Reversibility and stochastic networks*, Wiley Series in Probability and Mathematical Statistics, New York.

[26] Kimura, M. (1980). A simple method for estimating evolutionary rates of base substitution through comparative studies of nucleotide sequences. *Journal of Molecular Biology* 16, 111-120.

[27] Kolaczkowski, B. and Thornton, J.W. (2004). Performance of maximum parsimony and likelihood phylogenetics when evolution is heterogeneous. *Nature* 431, 980-984.

[28] Lake, J. A. (1994). Reconstructing evolutionary trees from DNA and protein sequences: Paralinear distances. *Proceedings of the National Academy of Sciences of the United States of America* 91, 1155-1159.

[29] Lanave, C. and Pesole, G. (1993). Stationary MARKOV processes in the evolution of biological macromolecules. *Binary* 5, 191-195.

[30] Lanave, C., Preparata, G., Saccone, C. and Serio, G. (1984). A new method for calculating evolutionary substitution rates. *Journal of Molecular Evolution* 20, 86-93.

[31] Lockhart, P. J., Steel, M. A., Hendy, M. D., Penny, D. (1994). Recovering evolutionary trees under a more realistic model of sequence evolution. *Molecular Biology and Evolution* 11, 605-612.

[32] O'Neill, M. E. (1975). *Problems in Dependence*, PhD Thesis, University of Sydney.

[33] Pagel, M., Meade, A. (2004). A phylogenetic mixture model for detecting pattern-heterogeneity in gene sequence or character-state data. *Systematic Biology* 53, 571-581.

[34] Penny, D., Hendy, M. D. and Steel, M. A. (1992). Progress with methods for constructing evolutionary trees. *Trends Ecology Evolution* 7, 73-79.

[35] Rambaut, A. and Grassly, N. C. (1997). Seq-Gen: an application for the monte carlo simulation of DNA sequence evolution along phylogenetic trees. *Computer Applications in the Biosciences* 13, 235-238.

[36] Saitou, N. and Nei, M. (1987). The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evoltion* 4, 406-425.

[37] Sokal, R. R. and Michener, C. D. (1958). A statistical method for evaluating systematic relationship. *University of Kansas Scientific Bulletin* 28, 1409-1438.

[38] Steel, M. A. (1993). Distributions on bicoloured binary trees arising from the principle of parsimony. *Discrete Applied Mathematics* 41, 245-261.

[39] Stuart, A. (1955). A test for homogeneity of the marginal distributions in a two-way classification. *Biometrika* 42, 412-416.

[40] Swofford, D. L., Olsen G. J., Waddell P. J., Hillis D.M. (1996). Phylogenetic inference. Pp 407-514. In: *Molecular systematics*, 2nd Ed. (Eds. Hillis DM, Moritz D, Mable BK), Sinauer, Sunderland, Massachusetts, USA.

[41] Swofford, D. L. *et al.* (2001). Bias in phylogenetic estimation and its relevance to the choice between parsimony and likelihood methods. *Systematic Biology* 50, 525-539.

[42] Tavaré, S. (1986). Some probabilistic and statistical problems on the analysis of DNA sequences. *Lectures on Mathematics in the*

Life Sciences 17, 57-86.

[43] Van Den Bussche, R. A., Baker, R. J., Huelsenbeck, J. P. and Hillis, D. M. (1998). Base compositional bias and phylogenetic analyses: a test of the "flying DNA" hypothesis, *Molecular Phylogenetics and Evolution* 10, 408-416.

[44] Waddell, P. J. and Steel, M. A. (1997). General time reversible distances with unequal rates across sites: Mixing Γ and inverse gaussian distributions with invariant sites. *Molecular Phylogenetics and Evolution* 8, 398-414.

[45] Waddell, P. J., Cao, Y., Hauf, J. and Hasegawa, M. (1999). Using novel phylogenetic methods to evaluate mammalian mtDNA, including amino acid-invariant sites-LogDet plus site stripping, to detect internal conflicts in the data, with special reference to the positions of hedgehog, armadillo, and elephant. *Systematic Biology* 48, 31-53.

[46] Zharkikh, A. (1994). Estimation of evolutionary distances between nucleotide sequences. *Journal of Molecular Evolution* 39, 315-329.

Appendix (A)

The Symmetric Matrix S

Description:
This function calculates the symmetric matrix S.
Usage:
Smatrix(s, pix)
Arguments:
s: a vector of variables containing the six free parameters in the S matrix.
pix: a vector giving the stationary probabilities for the four nucleotides A, C, G and T.
Details:
This function calculates the matrix S, which we used to calculate the rate matrix R.
See Chapter 2 Section 2.1.1 for more details.
Value:
A 4×4 symmetric matrix.
See Also:
Pt, Fmatrix, gn ,gn2.

`Example:`

s=c(.1,.2,.3,.4,.5,.6)

pi=c(.1,.1,.1,.7)

Smatrix(s, pi)

```
     [,1] [,2] [,3] [,4]
[1,] -2.4  0.1  0.2  0.3
[2,]  0.1 -4.0  0.4  0.5
[3,]  0.2  0.4 -4.8  0.6
[4,]  0.3  0.5  0.6 -0.2
```

The Transition Probability Function

Description:

This function calculates the transition probability function for a process during a period of time.

Usage:

Pt(S, Pi, t)

Arguments:

S: a 4×4 symmetric matrix.

Pi: a diagonal matrix containing the stationary distribution for the process.

t: a period of time describing the length of the process.

Details:

This function needs the 4×4 symmetric matrix S, Π and the process length t in order to find the transition probability over that process, where $P_{ij}(t)$ is the probability that the i^{th} nucleotide changes to the j^{th} nucleotide during the period of t.

See Chapter 2 Section 2.1 for more details.

Value:

A 4×4 matrix containing the transition probabilities for a process.

See Also:

Smatrix.

Example:

Pi=diag(c(.1,.1,.1,.7)), S=Smatrix(c(.3,.3,.3,.3,.3,.3),diag(Pi)), t=1

Pt(S, Pi, t)

```
           [,1]       [,2]       [,3]      [,4]
[1,] 0.76673640 0.02591818 0.02591818 0.1814272
[2,] 0.02591818 0.76673640 0.02591818 0.1814272
[3,] 0.02591818 0.02591818 0.76673640 0.1814272
[4,] 0.02591818 0.02591818 0.02591818 0.9222455
```

Joint Distribution for Two Matched Sequences

`Description:`

This function calculates the joint distribution function for two edge tree.

`Usage:`

Fmatrix(t1, t2, f0, Sx, Sy, Pix, Piy)

`Arguments:`

t1: represents the length from the tree root to the first node.

t2: represents the length from the tree root to the second node.

f0: the initial distribution for the four nucleotides.

Sx: a 4×4 symmetric matrix related to the first edge.

Sy: a 4×4 symmetric matrix related to the second edge.

Pix: a diagonal matrix for the stationary distribution of the first edge.

Piy: a diagonal matrix for the stationary distribution of the second edge.

`Details:`

This function calculates the joint distribution function for a two edge tree with different edge lengths, stationary distributions and different S matrices.

See Chapter 2 Example 2.1 for more details.

`Value:`

A 4×4 matrix containing the joint edges.

`See Also:`

gn, Smatrix.

Example:

f0=c(.25,.25,.25,.25)

Pi1=diag(c(.2,.2,.2,.4))

Pi2=diag(c(.1,.1,.1,.7))

S1=Smatrix(c(.2,.2,.2,.2,.2,.2),diag(Pi1))

S2=Smatrix(c(.3,.3,.3,.3,.3,.3),diag(Pi2))

Fmatrix(1, .5, f0, S1, S2, Pi1, Pi2)

```
           [,1]       [,2]       [,3]       [,4]
[1,] 0.18732907 0.01115705 0.01115705 0.03129337
[2,] 0.01115705 0.18732907 0.01115705 0.03129337
[3,] 0.01115705 0.01115705 0.18732907 0.03129337
[4,] 0.01946303 0.01946303 0.01946303 0.21880130
```

Joint Distribution for K Matched Sequences

`Description:`

This function calculates the joint distribution array for K matched sequences.

`Usage:`

gn(theta, merge2)

`Arguments:`

theta: a vector of variables containing the following parameters in this order:

1. the first three parameters from π_X vector,
2. the first three parameters from π_Y vector,
3. the first three parameters from f_0 vector,
4. the six off-diagonal free parameters in the S matrix,
5. a scalar ρ,
6. a vector of lengths containing $K-2$ values.

merge2: a $(K-1)\times 2$ matrix describing the tree topology.

`Details:`

This function calculates the joint distribution array for a tree with K matched sequences. it uses the following functions: Pt, Fmatrix and Smatrix.

See Chapter 2 Section 2.6.2 for more details.

`Value:`

A 4^K array containing the joint distribution for the K edges.

`See Also:`

Fmatrix, Pt, Smatrix.

`Example:`

merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)

theta=c(rep(.25,3), rep(.25,3),rep(.25,3), c(.2,.35,.79,.01,.93,.47), 3,.1,.5,.8)

gn(theta, merge2)

Note: This will give 4^5 array.

Joint Distribution for K Matched Sequences (2)

Description:

This function calculates the joint distribution array for K matched sequences.

Usage:

gn2(theta, merge2)

Arguments:

theta: is a vector of variables containing the following parameters in this order:

1. the first three parameters from π_X vector,
2. the first three parameters from π_Y vector,
3. the first three parameters from f_0 vector,
4. the six off-diagonal free parameters in the S matrix,
5. a $(K-1)\times 2$ matrix contains the rate at each edge of the K matched sequences.

merge2: a $(K-1)\times 2$ matrix describing the tree topology.

Details:

This function calculates the joint distribution array for a tree with K matched sequences. it uses the following functions: Pt, Fmatrix and Smatrix.

See Chapter 2 Section 2.6.3 for more details.

Value:

A 4^K array containing the joint distribution for the K edges.

See Also:

Fmatrix, Pt, Smatrix.

`Example:`

```
merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)
rho2=matrix(c(.3,.5,.3,.2,.3,.5,.8,2.7),4,2)
theta=c(rep(.25,3), rep(.25,3),rep(.25,3), c(.2,.35,.79,.01,.93,.47),rho2)
gn2(theta, merge2)
```

Note: This will give 4^5 array.

Generating Samples from a Multinomial Distribution

Description:

Generating random DNA samples from a multinomial distribution.

Usage:

Ntml(n, Ft)

Arguments:

n: sample size

Ft: a 4^K array, containing the joint distribution probabilities for K matched sequences.

Details:

This function generates a 4^K DNA array from a multinomial distribution. It depends on the sample size we need to generate and the 4^K joint distribution array of K matched sequences.

See Chapter 3 Section 3.2 for more details.

Value:

A 4^K observed divergence array.

See Also:

simemb, simapp, gn3sim, gn, gn2, Fmatrix.

Example:

merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)

theta=c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.35,.79,.01,.93,.47), 3,.1,.5,.8)

F1=gn(theta,merge2)

Ntml(1000, F1)

Note: This will give a 4^5 observed divergence array.

Generating Random DNA Samples using an Embedded Markov Chain

Description:

This function generates random DNA samples using embedded chain.

Usage:

simemb(theta, seq Length, merge2)

Arguments:

theta: is a vector of variables containing the following parameters in this order:

1. the first three parameters from π_X vector,
2. the first three parameters from π_Y vector,
3. the first three parameters from f_0 vector,
4. the six off-diagonal free parameters in the S matrix,
5. a scalar ρ,
6. a vector of lengths containing $K-2$ values.

merge2: a $(K-1) \times 2$ matrix describing the tree topology.

seqlength: the length of sequences we need to generate.

Details:

This function generates 4^K DNA array using embedded Markov chain. It depends on a set of variables theta, the sequence length and a merge matrix describing the tree topology.

See Chapter 3 Section 3.3 for more details.

Value:

A $n \times K$ observed divergence matrix.

See Also:

Ntml, simapp, gn3sim, gn, gn2, Fmatrix.

`Example:`

theta=(c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.35,.79,.01,.93,.47), 3,.1,.5,.8))

n=1000

merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)

simemb(theta, n, merge2)

Note: This will give 4^5 observed divergence array.

Generating Random DNA Samples using the Rambaut and Grassly Method

Description:

This function generates random DNA samples using Rambaut and Grassly method.

Usage:

gn3sim(theta, seq Length, merge2)

Arguments:

theta: a vector of variables containing the following parameters in this order:

1. the first three parameters from π_X vector,
2. the first three parameters from π_Y vector,
3. the first three parameters from f_0 vector,
4. the six off-diagonal free parameters in the S matrix,
5. a scalar ρ,
6. a vector of lengths containing $K-2$ values.

merge2: a $(K-1)\times 2$ matrix describing the tree topology.

seqlength: the length of sequences we need to generate.

Details:

This function generates a 4^K DNA array using Rambaut and Grassly, (1997) method. It depends on a set of variables theta, the sequence length and a merge matrix describing the tree topology.

See Chapter 3 Section 3.3.1 for more details.

divergence array.

Value:

A $n \times K$ observed divergence matrix.

See Also:

Ntml, simapp, simemb, gn, gn2, Fmatrix.

`Example:`

theta=(c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.35,.79,.01,.93,.47), 3,.1,.5,.8))

n=1000

merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)

gn3sim(theta, n, merge2)

Note: This will give 4^5 observed divergence array.

Generating Random DNA Samples using an Approximation Method

Description:

This function generates random DNA samples using an approximation method.

Usage:

simapp(theta, seq Length, merge2)

Arguments:

theta: a vector of variables containing the following parameters in this order:

1. the first three parameters from π_X vector,
2. the first three parameters from π_Y vector,
3. the first three parameters from f_0 vector,
4. the six off-diagonal free parameters in the S matrix,
5. a scalar ρ,
6. a vector of lengths containing $K-2$ values.

merge2: a $(K-1)\times 2$ matrix describing the tree topology.

seqlength: the length of sequences we need to generate.

Details:

This function generates a 4^K DNA array using an approximation method. It depends on a set of variables theta, the sequence length and a merge matrix describing the tree topology.

See Chapter 3 Section 3.4 for more details.

Value:

A $n\times K$ observed divergence matrix.

See Also:

Ntml, simemb, gn3sim, gn, gn2, Fmatrix.

`Example:`

```
theta=(c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.2,.2,.2,.2,.2),
3,.1,.5,.8))
n=1000
merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)
simapp(theta, n, merge2)
```

Note: This will give 4^5 observed divergence array.

Transforming 4^K Array to $m \times K$ Matrix

Description:

This function transfers any array to a matrix.

Usage:

artomat(Ft)

Arguments:

Ft: a 4^K array, containing the observed divergent frequencies for K matched sequences.

Details:

This function transfers any 4^K array containing the observed divergent frequencies of K matched sequences to an $m \times K$ matrix, where m is the sum of the frequencies in the 4^K observed divergence array.

See Chapter 3, Example 3.1 for more details.

Value:

An $m \times K$ matrix, where m is the sum of the frequencies in the 4^K divergence

See Also:

gn2, gn, Fmatrix.

Example:

```
F1=gn(c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.35,.79,.01,.93,.47),
3,.1,.5,.8))
N1=Ntml(1000,F1)
artomat(N1)
```

Note: This will give 1000×5 matrix.

Paralinear Distances

Description:

This function calculates the paralinear distance between K matched DNA sequences.

Usage:

Distance(F4)

Arguments:

F4: a 4^K array containing the joint distribution array $F(t)$ or the observed array N.

Details:

This function calculates the paralinear distances between K matched DNA sequences, depending on the joint distribution array for these K sequences or on the observed divergence array N.

See Chapter 5 Section 5.2 for more details.

Value:

A $K \times K$ symmetric matrix distances between the K sequences.

See Also:

gn2, gn, Fmatrix, Ntml.

Example:

merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)

theta=c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.35,.79,.01,.93,.47), 3,.1,.5,.8)

F1=gn(theta,merge2)

Distance(F1)

```
      [,1]  [,2] [,3]  [,4]  [,5]
[1,] 0.000 0.825  6.6 5.500 5.500
[2,] 0.825 0.000  6.6 5.500 5.500
[3,] 6.600 6.600  0.0 5.500 5.500
[4,] 5.500 5.500  5.5 0.000 1.375
[5,] 5.500 5.500  5.5 1.375 0.000
```

Test for Symmetry of Matched DNA Sequences

`Description:`
This function tests for symmetry between all the pairs of K matched DNA sequences.
`Usage:`
TEST2(Nt)
`Arguments:`
Nt: a 4^K array containing the observed divergence array N.
`Details:`
This function calculates Bowker's test for symmetry, Stuart's test for marginal symmetry and the test for internal symmetry. It depends on the 4^K observed divergence array N.
See Chapter 4 Sections 4.2, 4.3 and 4.4, for more details.
`Value:`
A list of three lower triangle matrices:
1. the lower triangle of the matrix contains $(K-1) \times (K-1)$ values shows Bowker's test between all the possible pairs of the K sequences.
2. the lower triangle of the matrix contains $(K-1) \times (K-1)$ values shows Stuart's test between all the possible pairs of the K sequences.
3. the lower triangle of the matrix contains $(K-1) \times (K-1)$ values shows the internal test between all the possible pairs of the K sequences.
`See Also:`
Ntml, simapp, simemb, TEST3.

Example:

merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)

theta=c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.35,.79,.01,.93,.47), 3,.1,.5,.8)

F1=gn(theta,merge2)

N1=Ntml(1000,F1)

TEST2(N1)

```
$Bowker.test
       1      2      3      4
2 0.5757
3 0.1403 0.7638
4 0.3291 0.7729 0.7252
5 0.8371 0.9933 0.8939 0.4990

$Stuart.test
       1      2      3      4
2 0.4860
3 0.3978 0.7549
4 0.2637 0.4012 0.7752
5 0.7695 0.9763 0.9110 0.2742

$Internal.test
       1      2      3      4
2 0.5101
3 0.0825 0.5402
4 0.4022 0.9520 0.4694
5 0.6507 0.9088 0.6306 0.6888
```

Overall Test for Marginal Symmetry

Description:

This function tests for symmetry between K matched DNA sequences.

Usage:

TEST3(Nt)

Arguments:

Nt: a 4^K array containing the observed divergence array N.

Details:

This function calculates overall test for marginal symmetry. It depends on the 4^K observed divergence array N.

See Chapter 4 Sections 4.5, for more details.

Value:

A single value gives the overall test for marginal symmetry between K matched sequences.

See Also:

Ntml, simapp, simemb, TEST2.

Example:

```
merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)
theta=c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.35,.79,.01,.93,.47),
3,.1,.5,.8)
F1=gn(theta,merge2)
N1=Ntml(1000,F1)
TEST3(N1)
[1] 9.217612
```

Negative Log Likelihood Ratio

Description:

This function calculates log likelihood ratio value.

Usage:

likelihood(thetast,fobs,merge2)

Arguments:

thetast: a starting values for the parameter we need to estimate.

fobs: the 4^K joint distribution array for K edge tree.

merge2: a $(K-1) \times 2$ matrix describing the tree topology.

Details:

This function calculates the log likelihood ratio value for $F(t)$. It needs a vector of starting values for the parameters estimate, 4^K observed divergence array and merge matrix describing the tree topology.

See Chapter 6 Sections 6.2, for more details.

Value:

The value of the log likelihood ratio.

See Also:

gn, gn2.

Example:

merge2=matrix(c(-1,-4,-3,2,-2,-5,1,3), 4, 2)

theta=c(rep(.25,3), rep(.25,3), rep(.25,3), c(.2,.35,.79,.01,.93,.47), 3,.1,.5,.8)

F1=gn(theta,merge2)

likelihood(theta,F1,merge2)

```
 [1] 0.00
```

Appendix (B)

The Symmetric Matrix S - Program

```
Smatrix<-function(s, pix)
{
    Pi <- diag(pix)
    Sd <- matrix(0, 4, 4)
    Sd[1, 2:4] <- s[1:3]
    Sd[2, 3:4] <- s[4:5]
    Sd[3, 4] <- s[6]
    for(i in 1:4) {
        Sd[, i] <- Sd[i,  ]
        Sd[i, i] <- ( - sum((Sd[i,  - i]) %*% (diag(Pi)[
            - i])))/diag(Pi)[i]
    }
    Sd
}
```

The Transition Probability Function - Program

```
 Pt<-function(S, Pi, t)
{
    ax <- matrix(0, nrow = 4, ncol = 4)
    fx <- matrix(0, nrow = 4, ncol = 4)
    Rx <- S %*% Pi
    a1 <- eigen((sqrt(Pi)) %*% Rx %*% (solve(sqrt(Pi))),
        symmetric = T)$values
    a2 <- eigen((sqrt(Pi)) %*% Rx %*% (solve(sqrt(Pi))),
        symmetric = T)$vectors
    for(j in 1:4) {
        ax <- exp(a1[j] * (t)) * a2[, j] %*% t(a2[, j])
        fx <- fx + ax
    }
    ptx <- (solve(sqrt(Pi))) %*% fx %*% (sqrt(Pi))
    ptx
}
```

Joint Distribution for Matched Sequences - Program

```
Fmatrix<- function(t1, t2, f0, Sx2, Sy2, Pix, Piy)
{
    F0 <- diag(f0)
    F1 <- t(Pt(Sx2, Pix, t1)) %*% F0 %*% Pt(Sy2, Piy, t2)
    F1
}
```

Joint Distribution for K Matched Sequences - Program

```
 gn<-function(theta, merge2)
{
    pix1 <- theta[1:3]
    piy1 <- theta[4:6]
    f00 <- theta[7:9]
    ss <- theta[10:15]
    rho <- theta[16]
    pix2 <- c(pix1, 1 - sum(pix1))
    Pix2 <- diag(pix2)
    piy2 <- c(piy1, 1 - sum(piy1))
    Piy2 <- diag(piy2)
    f0 <- c(f00, 1 - sum(f00))
    height <- theta[17:length(theta)]
    if(sum(abs(pix2)) > 1) {
        return(0)
    }
    if(sum(abs(piy2)) > 1) {
        return(0)
    }
    if(sum(abs(f0)) > 1) {
        return(0)
    }
    if(rho <= 0) {
        return(0)
    }
    if(any(theta <= 0)) {
        return(0)
    }
    Sx1 <- Smatrix(ss, pix2)
    Sy1 <- rho * Smatrix(ss, piy2)
    k <- dim(merge2)[1]
    if(k == 1) {
        F1 <- Fmatrix(theta[17], theta[17], f0, Sx1, Sy1,
            Pix2, Piy2)
        return(F1)
    }
```

```
else {
    h <- c(0, height, 1)
    me <- merge2[k,  ]
    f1 <- f0
    m <- me
    t0 <- h[k + 1]
    cc <- 0
    if(merge2[, 1][k] > 0) {
        cc <- merge2[, 1][k]
        v <- matrix(0, k, 1)
        v[1] <- merge2[k, 1]
        cur <- 0
        max <- 1
        while(cur < max) {
            cur <- cur + 1
            for(i in c(1, 2)) {
                if(merge2[(v[cur]), i] >
                    0) {
                    max <- max + 1
                    v[max] <- merge2[
                        (v[cur]),
                        i]
                }
            }
            cc <- matrix(0, max, 1)
            for(i in 1:max) {
                cc[i] <- v[i]
            }
        }
    }
    for(i in 1:k) {
        zzz <- 0
        iii <- k - i + 1
        if(iii == k) {
            xx <- 0
            if(any(merge2[k, 1] == cc)) {
                pix <- pix2
                Pix <- Pix2
                piy <- piy2
                Piy <- Piy2
                Sx2 <- Sx1
                Sy2 <- Sy1
    }
    else {
        piy <- pix2
        Piy <- Pix2
        pix <- piy2
        Pix <- Piy2
        Sy2 <- Sx1
        Sx2 <- Sy1
    }
}
else {
    if(any(iii == cc)) {
        pix <- pix2
        Pix <- Pix2
        piy <- pix2
        Piy <- Pix2
        Sx2 <- Sx1
        Sy2 <- Sx1
    }
    else {
        piy <- piy2
        Piy <- Piy2
        pix <- piy2
        Pix <- Piy2
        Sx2 <- Sy1
        Sy2 <- Sy1
    }
}
m <- rev(sort(me))
mm <- merge2[m[1],  ]
if(me[1] > 0) {
    t1 <- t0 - h[me[1] + 1]
}
else {
    t1 <- t0
}
if(me[2] > 0) {
    t2 <- t0 - h[me[2] + 1]
}
else {
    t2 <- t0
}
```

```
                if(t1 < 0 || t1 > 1) {
                    return(0)
                }
                if(t2 < 0 || t2 > 1) {
                    return(0)
                }
                hp <- rev(order(me))
                for(j in 1:(4^(i - 1))) {
                    if(j == 1) {
                        F1 <- Fmatrix(t1, t2, f1[
                            1:4], Sx2, Sy2,
                            Pix, Piy)
                    }
                    else {
                        F1 <- Fmatrix(t1, t2, f1[
                            (4 * (j - 1) + 1):
                            (4 * (j - 1) + 4)],
                            Sx2, Sy2, Pix, Piy)
                    }
                    if(j == 1) {
                        f <- c(as.vector(F1), f1[
                            -1:-4])
                    }
                    else {
                        f <- c(f[1:(4^2 * (j - 1))],
                            as.vector(F1), f1[
                            -1: - (4 * j)])
                    }
                }
                f12 <- array(f, c(rep(4, i + 1)))
                f112 <- aperm(f12, c(hp))
                f1 <- as.vector(f112)
                if(m[1] > 0) {
                    t0 <- h[m[1] + 1]
                }
                else {
                    return(f112)
                }
                me <- c(mm, m[-1])
            }
        }
    }
```

Joint Distribution for K Matched Sequences (2) - Program

```
gn2<-function(theta, merge2)
{
    pix1 <- theta[1:3]
    piy1 <- theta[4:6]
    f00 <- theta[7:9]
    ss <- theta[10:15]
    rho1 <- matrix(theta[16:(15 + length(merge2))],  , 2)
    pix2 <- c(pix1, 1 - sum(pix1))
    Pix2 <- diag(pix2)
    piy2 <- c(piy1, 1 - sum(piy1))
    Piy2 <- diag(piy2)
    f0 <- c(f00, 1 - sum(f00))
    if(sum(abs(pix2)) > 1) {
        return(0)
    }
    if(sum(abs(piy2)) > 1) {
        return(0)
    }
    if(sum(abs(f0)) > 1) {
        return(0)
    }
    Sx1 <- Smatrix(ss, pix2)
    Sy1 <- Smatrix(ss, piy2)
    k <- dim(merge2)[1]
    if(k == 1) {
        F1 <- Fmatrix(theta[16], theta[17], f0, (rho1[1,
            1] * Sx1), (rho1[1, 2] * Sy1), Pix2, Piy2)
        return(F1)
    }
    else {
        merge <- matrix(0, k, 2)
        for(i in 1:k) {
            merge[i,  ] <- rev(sort(merge2[i,  ]))
        }
        me <- merge[k,  ]
        f1 <- f0
```

```
m <- me
cc <- 0
if(merge2[, 1][k] > 0) {
    cc <- merge[, 1][k]
    v <- matrix(0, k, 1)
    v[1] <- merge2[k, 1]
    cur <- 0
    max <- 1
    while(cur < max) {
        cur <- cur + 1
        for(i in c(1, 2)) {
            if(merge2[(v[cur]), i] >
                0) {
                max <- max + 1
                v[max] <- merge2[
                    (v[cur]),
                    i]
            }
        }
        cc <- matrix(0, max, 1)
        for(i in 1:max) {
            cc[i] <- v[i]
        }
    }
}
for(i in 1:k) {
    iii <- k - i + 1
    if(iii == k) {
        if(any(merge2[k, 1] == cc)) {
            pix <- pix2
            Pix <- Pix2
            piy <- piy2
            Piy <- Piy2
            Sx2 <- Sx1
            Sy2 <- Sy1
        }
        else {
            piy <- pix2
            Piy <- Pix2
            pix <- piy2
            Pix <- Piy2
            Sy2 <- Sx1
            Sx2 <- Sy1
        }
    }
    else {
        if(any(iii == cc)) {
            pix <- pix2
            Pix <- Pix2
            piy <- pix2
            Piy <- Pix2
            Sx2 <- Sx1
            Sy2 <- Sx1
        }
        else {
            piy <- piy2
            Piy <- Piy2
            pix <- piy2
            Pix <- Piy2
            Sx2 <- Sy1
            Sy2 <- Sy1
        }
    }
    m <- rev(sort(me))
    mm <- merge[m[1],  ]
    t1 <- rho1[iii,  ][1]
    t2 <- rho1[iii,  ][2]
    hp <- rev(order(me))
    for(j in 1:(4^(i - 1))) {
        if(j == 1) {
            F1 <- Fmatrix(t1, t2, f1[
                1:4], Sx2, Sy2,
                Pix, Piy)
        }
        else {
            F1 <- Fmatrix(t1, t2, f1[
                (4 * (j - 1) + 1):
                (4 * (j - 1) + 4)],
                Sx2, Sy2, Pix, Piy)
        }
        if(j == 1) {
            f <- c(as.vector(F1), f1[
                -1:-4])
        }
```

```
                else {
                    f <- c(f[1:(4^2 * (j - 1))],
                        as.vector(F1), f1[
                        -1: - (4 * j)])
                }
            }
            f12 <- array(f, c(rep(4, i + 1)))
            f112 <- aperm(f12, c(hp))
            f1 <- as.vector(f112)
            if(m[1] > 0)
                t0 <- 9999
            else {
                return(f112)
            }
            me <- c(mm, m[-1])
        }
    }
}
```

Generating Samples from a Multinomial Distribution - Program

```
Ntml<-function(N, Ft) {
    s1 <- length(dim(Ft))
    s2 <- length(Ft)
    x <- array(0, c(rep(4, s1)))
    x1 <- 0
    ft1 <- 0
    x[1] <- rbinom(1, N, Ft[1])
    for(i in 2:(s2 - 1)) {
        x1 <- x1 + x[i - 1]
        ft1 <- ft1 + Ft[i - 1]
        x[i] <- rbinom(1, (N - x1), (Ft[i])/(1 - ft1))
    }
    x1 <- x1 + x[s2 - 1]
    ft1 <- ft1 + Ft[s2 - 1]
    x[s2] <- N - x1
    x
}
```

Generating Random DNA Samples Using an Embedded Markov Chain - Program

```
simemb<-function(theta, seqLength, merge2)
{
    pix1 <- theta[1:3]
    piy1 <- theta[4:6]
    f00 <- theta[7:9]
    ss <- theta[10:15]
    rho <- theta[16]
    pix2 <- c(pix1, 1 - sum(pix1))
    Pix2 <- diag(pix2)
    piy2 <- c(piy1, 1 - sum(piy1))
    Piy2 <- diag(piy2)
    f0 <- c(f00, 1 - sum(f00))
    Sx1 <- Smatrix(ss, pix2)
    Sy1 <- rho * Smatrix(ss, piy2)
    merge <- merge2
    k <- dim(merge)[1]
    height2 <- c(theta[17:length(theta)], 1)
    height <- height2
    h <- c(0, height)
    me <- merge[k,  ]
    m <- me
    t0 <- h[k + 1]
    cc <- 0
    if(merge[, 1][k] > 0) {
        cc <- merge[, 1][k]
        v <- matrix(0, k, 1)
        v[1] <- merge[k, 1]
        cur <- 0
        max <- 1
        while(cur < max) {
            cur <- cur + 1
            for(i in c(1, 2)) {
                if(merge[(v[cur]), i] > 0) {
                    max <- max + 1
                    v[max] <- merge[(v[cur]),
                        i]
                }
            }
            cc <- matrix(0, max, 1)
            for(i in 1:max) {
                cc[i] <- v[i]
            }
        }
    }
    seqr <- matrix(0, seqLength, k - 1)
    fs <- matrix(0, seqLength, k + 1)
    seq1 <- cbind(sample(1:4, seqLength, prob = f0, replace = T)
        )
    for(i in 1:k) {
        iii <- k - i + 1
        if(iii == k) {
            xx <- 0
            if(any(merge[k, 1] == cc)) {
                pix <- pix2
                Pix <- Pix2
                piy <- piy2
                Piy <- Piy2
                Sx2 <- Sx1
                Sy2 <- Sy1
            }
            else {
                piy <- pix2
                Piy <- Pix2
                pix <- piy2
                Pix <- Piy2
                Sy2 <- Sx1
                Sx2 <- Sy1
            }
        }
        else {
            if(any(iii == cc)) {
                pix <- pix2
                Pix <- Pix2
                piy <- pix2
                Piy <- Pix2
                Sx2 <- Sx1
                Sy2 <- Sx1
            }
            else {
                piy <- piy2
                Piy <- Piy2
                pix <- piy2
                Pix <- Piy2
                Sx2 <- Sy1
                Sy2 <- Sy1
            }
        }
        m <- rev(sort(me))
        mm <- merge2[m[1],  ]
        if(me[1] > 0) {
            t1 <- t0 - h[me[1] + 1]
        }
        else {
            t1 <- t0
        }
        if(me[2] > 0) {
            t2 <- t0 - h[me[2] + 1]
        }
        else {
            t2 <- t0
        }
        if(t1 < 0 || t1 > 1) {
            return(0)
        }
        if(t2 < 0 || t2 > 1) {
            return(0)
        }
        hp <- rev(order(me))
        if(i != 1) {
            seq1 <- seqr[, (k - i + 1)]
        }
        seq2 <- genseq4(t1, t2, seq1, Sx2, Sy2, Pix, Piy)
        if(me[1] < 0) {
            fs[, (abs(me[1]))] <- seq2[, 1]
        }
        else {
            seqr[, (abs(me[1]))] <- seq2[, 1]
        }
        if(me[2] < 0) {
            fs[, (abs(me[2]))] <- seq2[, 2]
```

```
        }
        else {
            seqr[, (abs(me[2]))] <- seq2[, 2]
        }
        if(m[1] > 0)
            t0 <- h[m[1] + 1]
        else {
            return(fs)
        }
        me <- c(mm, m[-1])
    }
}

genseq4 <- function(t1, t2, seq, Sx2, Sy2, Pix, Piy) {
    seqx <- matrix(0, nc = 1, nr = length(seq))
    seqy <- matrix(0, nc = 1, nr = length(seq))
    Rx <- Sx2 %*% Pix
    Ry <- Sy2 %*% Piy
    for(j in 1:(length(seq))) {
        Tx <- matrix(0, 1, 10)
        WTx <- matrix(0, 1, 10)
        m <- matrix(0, 1, 10)
        m[1] <- seq[j]
        WTx[1] <- rexp(1, ( - Rx[m[1], m[1]]))
        xx <- m[1]
        for(i in 2:10) {
            Tx[i] <- Tx[i - 1] + WTx[i - 1]
            if(Tx[i] >= t1)
                break
            rrx <- Rx[m[i - 1],  ]/( - Rx[m[i - 1], m[i - 1]])
            rrx[m[i - 1]] <- 0
            m[i] <- sample(1:4, 1, rrx, replace = T)
            WTx[i] <- rexp(1, ( - Rx[m[i], m[i]]))
            xx <- m[i]
        }
        seqx[j] <- xx
    }
    for(k in 1:(length(seq))) {
        Ty <- matrix(0, 1, 10)
        WTy <- matrix(0, 1, 10)
        n <- matrix(0, 1, 10)
        n[1] <- seq[k]
        WTy[1] <- rexp(1, ( - Rx[n[1], n[1]]))
        yy <- n[1]
        for(i in 2:10) {
            Ty[i] <- Ty[i - 1] + WTy[i - 1]
            if(Ty[i] >= t2)
                break
            rry <- Ry[n[i - 1],  ]/( - Ry[n[i - 1], n[i - 1]])
            rry[n[i - 1]] <- 0
            n[i] <- sample(1:4, 1, rry, replace = T)
            WTy[i] <- rexp(1, ( - Ry[n[i], n[i]]))
            yy <- n[i]
        }
        seqy[k] <- yy
    }
    sequ <- cbind(seqx, seqy)
    return(sequ)
}
```

Generating Random DNA Samples using the Rambaut and Grassly Method - Program

```
gn3sim<-function(theta, seqLength, merge2) {
    pix1 <- theta[1:3]
    piy1 <- theta[4:6]
    f00 <- theta[7:9]
    ss <- theta[10:15]
    rho <- theta[16]
    pix2 <- c(pix1, 1 - sum(pix1))
    Pix2 <- diag(pix2)
    piy2 <- c(piy1, 1 - sum(piy1))
    Piy2 <- diag(piy2)
    f0 <- c(f00, 1 - sum(f00))
    Sx1 <- Smatrix(ss, pix2)
    Sy1 <- rho * Smatrix(ss, piy2)
    height <- c(theta[17:length(theta)], 1)
    merge <- merge2
    k <- dim(merge)[1]
    h <- c(0, height)
    me <- merge[k,  ]
    m <- me
    t0 <- h[k + 1]
    cc <- 0
    if(merge[, 1][k] > 0) {
        cc <- merge[, 1][k]
        v <- matrix(0, k, 1)
        v[1] <- merge[k, 1]
        cur <- 0
        max <- 1
        while(cur < max) {
            cur <- cur + 1
            for(i in c(1, 2)) {
                if(merge[(v[cur]), i] > 0) {
                    max <- max + 1
                    v[max] <- merge[(v[cur]), i]
                }
            }
            cc <- matrix(0, max, 1)
        for(i in 1:max) {
            cc[i] <- v[i]
        }
    }
}
seqr <- matrix(0, seqLength, k - 1)
fs <- matrix(0, seqLength, k + 1)
seq1 <- cbind(sample(1:4, seqLength, prob = f0, replace = T))
for(i in 1:k) {
    iii <- k - i + 1
    if(iii == k) {
        xx <- 0
        if(any(merge[k, 1] == cc)) {
            pix <- pix2
            Pix <- Pix2
            piy <- piy2
            Piy <- Piy2
            Sx2 <- Sx1
            Sy2 <- Sy1
        }
        else {
            piy <- pix2
            Piy <- Pix2
            pix <- piy2
            Pix <- Piy2
            Sy2 <- Sx1
            Sx2 <- Sy1
        }
    }
    else {
        if(any(iii == cc)) {
            pix <- pix2
            Pix <- Pix2
            piy <- pix2
            Piy <- Pix2
            Sx2 <- Sx1
            Sy2 <- Sx1
        }
        else {
            piy <- piy2
            Piy <- Piy2
            pix <- piy2
            Pix <- Piy2
            Sx2 <- Sy1
            Sy2 <- Sy1
        }
    }
    m <- rev(sort(me))
    mm <- merge[m[1],  ]
    if(me[1] > 0) {
        t1 <- t0 - h[me[1] + 1]
    }
    else {
        t1 <- t0
    }
    if(me[2] > 0) {
        t2 <- t0 - h[me[2] + 1]
    }
    else {
        t2 <- t0
    }
    if(t1 < 0 || t1 > 1) {
        return(0)
    }
    if(t2 < 0 || t2 > 1) {
        return(0)
    }
    hp <- rev(order(me))
    if(i != 1) {
        seq1 <- seqr[, (k - i + 1)]
    }
    seq2 <- genseq2(t1, t2, seq1, Sx2, Sy2, Pix, Piy)
    if(me[1] < 0) {
        fs[, (abs(me[1]))] <- seq2[, 1]
    }
    else {
        seqr[, (abs(me[1]))] <- seq2[, 1]
    }
    if(me[2] < 0) {
        fs[, (abs(me[2]))] <- seq2[, 2]
    }
    else {
        seqr[, (abs(me[2]))] <- seq2[, 2]
    }
```

```
         if(m[1] > 0)
            t0 <- h[m[1] + 1]
        else {
            return(fs)
        }
        me <- c(mm, m[-1])
    }
}

genseq2 <- function(t1, t2, seq, Sx2, Sy2, Pix, Piy) {
    sequ <- matrix(0, nc = 2, nr = length(seq))
    p1 <- Pt(Sx2, Pix, t1)
    p2 <- Pt(Sy2, Piy, t2)
    for(i in 1:length(seq)) {
        sequ[i, 1] <- sample(1:4, 1, prob = p1[seq[i],  ])
        sequ[i, 2] <- sample(1:4, 1, prob = p2[seq[i],  ])
    }
    return(sequ)
}
```

Generating Random DNA Samples using an Approximation Method - Program

```
simapp<-function(theta, seqLength, merge1)
{
    pix1 <- theta[1:3]
    piy1 <- theta[4:6]
    f00 <- theta[7:9]
    ss <- theta[10:15]
    rho <- theta[16]
    pix2 <- c(pix1, 1 - sum(pix1))
    Pix2 <- diag(pix2)
    piy2 <- c(piy1, 1 - sum(piy1))
    Piy2 <- diag(piy2)
    f0 <- c(f00, 1 - sum(f00))
    Sx1 <- Smatrix(ss, pix2)
    Sy1 <- rho * Smatrix(ss, piy2)
    merge2 <- merge1
    k <- dim(merge2)[1]
    height2 <- c(theta[17:length(theta)], 1)
    height <- height2
    h <- c(0, height)
    me <- merge2[k,  ]
    m <- me
    t0 <- h[k + 1]
    cc <- 0
    if(merge2[, 1][k] > 0) {
        cc <- merge2[, 1][k]
        v <- matrix(0, k, 1)
        v[1] <- merge2[k, 1]
        cur <- 0
        max <- 1
        while(cur < max) {
            cur <- cur + 1
            for(i in c(1, 2)) {
                if(merge2[(v[cur]), i] > 0) {
                    max <- max + 1
                    v[max] <- merge2[(v[cur]),
                        i]
                }
            }
            cc <- matrix(0, max, 1)
            for(i in 1:max) {
                cc[i] <- v[i]
            }
        }
    }
    seqr <- matrix(0, seqLength, k - 1)
    fs <- matrix(0, seqLength, k + 1)
    seq1 <- cbind(sample(1:4, seqLength, prob = f0, replace = T)
        )
    for(i in 1:k) {
        iii <- k - i + 1
        if(iii == k) {
            xx <- 0
            if(any(merge2[k, 1] == cc)) {
                pix <- pix2
                Pix <- Pix2
                piy <- piy2
                Piy <- Piy2
                Sx2 <- Sx1
                Sy2 <- Sy1
            }
            else {
                piy <- pix2
                Piy <- Pix2
                pix <- piy2
                Pix <- Piy2
                Sy2 <- Sx1
                Sx2 <- Sy1
            }
        }
        else {
            if(any(iii == cc)) {
                pix <- pix2
                Pix <- Pix2
                piy <- pix2
                Piy <- Pix2
                Sx2 <- Sx1
                Sy2 <- Sx1
            }
            else {
                piy <- piy2
                Piy <- Piy2
                pix <- piy2
                Pix <- Piy2
                Sx2 <- Sy1
                Sy2 <- Sy1
            }
        }
        m <- rev(sort(me))
        mm <- merge2[m[1],  ]
        if(me[1] > 0) {
            t1 <- t0 - h[me[1] + 1]
        }
        else {
            t1 <- t0
        }
        if(me[2] > 0) {
            t2 <- t0 - h[me[2] + 1]
        }
        else {
            t2 <- t0
        }
        if(t1 < 0 || t1 > 1) {
            return(0)
        }
        if(t2 < 0 || t2 > 1) {
            return(0)
        }
        hp <- rev(order(me))
        if(i != 1) {
            seq1 <- seqr[, (k - i + 1)]
        }
        seq2 <- genseq3(t1, t2, seq1, Sx2, Sy2, Pix, Piy)
        if(me[1] < 0) {
            fs[, (abs(me[1]))] <- seq2[, 1]
        }
        else {
            seqr[, (abs(me[1]))] <- seq2[, 1]
        }
        if(me[2] < 0) {
            fs[, (abs(me[2]))] <- seq2[, 2]
```

```
        }
        else {
            seqr[, (abs(me[2]))] <- seq2[, 2]
        }
        if(m[1] > 0)
            t0 <- h[m[1] + 1]
        else {
            return(fs)
        }
        me <- c(mm, m[-1])
    }
}

genseq3 <- function(t1, t2, seq, Sx2, Sy2, Pix, Piy) {
    sequ <- matrix(0, nc = 2, nr = length(seq))
    seqx<-seq
    seqy<-seq
    SS <- diag(c(1, 1, 1, 1))
    p1 <- SS+Sx2%*%Pix
    p2 <- SS+Sy2%*%Piy
    print(p1)
     for(i in 1:(length(seq) * t1)) {
        km <- sample(1:length(seq), 1)
        seqx <- replace(seqx, km, (sample(1:4, 1, prob = p1[seqx[km],
          ],replace=T)))
    }
    for(i in 1:(length(seq) * t2)) {
        km <- sample(1:length(seq), 1)
        seqy <- replace(seqy, km, (sample(1:4, 1, prob = p2[seqy[km],
          ],replace=T)))
    }
    sequ<-cbind(seqx,seqy)
    return(sequ)
}
```

Transforming 4^K Array to $m \times K$ Matrix - Program

```
artomat<-function(fobs)
{ n1<-fobs nn=length(dim(fobs))
ee=NULL
for(i in 1:nn){ee[[i]]=1:4}
dimnames(n1)<-ee
w<-as.data.frame.table(n1)
w<-as.matrix(w)
w1<-w[,1:(dim(w)[2])-1]
w1<-matrix((as.numeric(w1)),dim(w)[1],dim(w)[2]-1)
w2<-cbind(as.numeric(w[,(dim(w)[2])]))
seq<-NULL
seq1<-NULL for(i in
1:dim(w)[1])
 { seq1<-rep(w1[i,],w2[i,]) seq<-c(seq,seq1) }
fseq<-(matrix(seq,nc=(dim(w1)[2]),byrow=T))
fseq }
```

Paralinear Distances - Program

```
Distance<-function(F4)
{
    L1 <- length(dim(F4))
    Dis <- matrix(0, L1, L1)
    for(i in 1:L1) {
        for(j in 1:L1) {
            if(i != j) {
                fij <- apply(F4, c(i, j), sum)
                fi <- apply(fij, 1, sum)
                fj <- apply(fij, 2, sum)
                Dis[i, j] <-  - log(det(diag((fi)^
                    -0.5) %*% fij %*% diag(
                    (fj)^-0.5)))
            }
            else {
                Dis[i, j] <- 0
            }
        }
    }
    Dis
}
```

Test for Symmetry of Matched DNA Sequences - Program

```
TEST2<-function(f)
{
    n <- length(dim(f))
    B1 <- NULL
    S1 <- NULL
    I1 <- NULL
    for(i in 1:n) {
        B <- matrix(0, n, 1)
        S <- matrix(0, n, 1)
        I <- matrix(0, n, 1)
        for(j in 1:n) {
            if(j != i) {
                k <- apply(f, c(i, j), sum)
                testr <- TEST(k)
            }
            else {
                testr <- matrix(0, 3, 1)
            }
            B[j] <- testr[1]
            S[j] <- testr[2]
            I[j] <- testr[3]
        }
        B1 <- cbind(B1, B)
        S1 <- cbind(S1, S)
        I1 <- cbind(I1, I)
    }
    list(Bowker.test = as.dist(round(B1, 4)), Stuart.test =
        as.dist(round(S1, 4)), Internal.test = as.dist(
        round(I1, 4)))
}


TEST<-function(Nt)
{
    QB <- 0
    QS <- 0
    QR <- 0
    PB <- 0
    PR <- 0
    PS <- 0
    ZB <- 0
    ZS <- 0
    ZR <- 0
    V <- matrix(0, 3, 3)
    d <- matrix(0, 3, 1)
    for(i in 1:4) {
        for(j in 1:4) {
            if((i < j))
                if(Nt[i, j] + Nt[j, i] == 0) {
                }
                else {
                    QB <- QB + (((Nt[i, j] -
                        Nt[j, i])^2)/(Nt[
                        i, j] + Nt[j, i]))
                }
        }
    }
    for(i in 1:3) {
        d[i] <- (sum(Nt[i,  ]) - sum(Nt[, i]))
        for(j in 1:3)
            if(i == j) V[i, j] <- (sum(Nt[i,  ]) + sum(
                    Nt[, i]) - 2 * sum(Nt[i,
                    i])) else V[i, j] <- -1 *
                    (Nt[i, j] + Nt[j, i])
    }
    QS <- t(d) %*% (solve(V)) %*% d
    QR <- QB - QS
    PB <- (1 - pchisq(QB, 6))
    Z <- qnorm(PB)
    PS <- (1 - pchisq(QS, 3))
    ZS <- qnorm(PS)
    PR <- (1 - pchisq(QR, 3))
    ZR <- qnorm(PR)
    r <- matrix(c(PB, PS, PR), 3, 1)
    r
}
```

Overall Test for Marginal Symmetry - Program

```
TEST3<-function(Farray)
{
    s1 <- sum(Farray)
    Farray <- Farray/s1
    v <- NULL
    m <- length(dim(Farray))
    r <- dim(Farray)[1]
    n <- NULL
    one <- rep(1, r * (m - 1))
    J <- matrix(0, 4 * (m - 1), 4 * (m - 1))
    for(i in 1:(m - 1)) {
        J[(r * (i - 1) + 1):(r * (i - 1) + r), (r * (i -
            1) + 1):(r * (i - 1) + r)] <- 1
    }
    L1 <- t(matrix(t(rep(diag(c(1, 1, 1, 1)), (m - 1))), r,
        (r * (m - 1))))
    L <- cbind(L1,  - (diag(one)))
    for(i in 1:m) {
        vc1 <- NULL
        for(k in 1:m) {
            vc <- NULL
            if(i == k) {
                fi <- apply(Farray, i, sum)
                vc <- diag(fi)
                n1 <- matrix(fi, r, 1)
                n <- rbind(n, n1)
            }
            if(i < k) {
                fij <- apply(Farray, c(i, k), sum)
                fr <- apply(fij, 1, sum)
                fc <- apply(fij, 2, sum)
                vc <- fij
            }
            if(i > k) {
                fij <- apply(Farray, c(i, k), sum)
                fr <- apply(fij, 1, sum)
                fc <- apply(fij, 2, sum)
                vc <- fij
            }
            vc1 <- cbind(vc1, vc)
        }
        v <- rbind(v, vc1)
    }
    Ts <- (t(L %*% n) %*% solve((L %*% v %*% t(L)) + J) %*% (
        L %*% n)) * s1
    Ts
}
```

Log Likelihood Ratio - Program

```
likelihood<-function(thetast,fobs,merge2)
{
   fobss=fobs; thetss=thetast; merge22=merge2
    fst <- gn(thetass,merge22)
    if(min(fst)<=0){return(lik<-4)}
    lik <- sum(fobss * log(fobss/fst))
    lik
}
```