Computational Statistics in Molecular Phylogenetics

William Albert James Fletcher

Centre for Mathematics and Physics in the Life Sciences and Experimental Biology (CoMPLEX)

&

Research Department of Genetics, Evolution and Environment (GEE)

University College London (UCL)

2010

Submitted for the degree of Doctor of Philosophy

Authorship Declaration

I, William Albert James Fletcher, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Simulation remains a very important approach to testing the robustness and accuracy of phylogenetic inference methods. However, current simulation programs are limited, especially concerning realistic models for simulating insertions and deletions (indels). In this thesis I implement a new, portable and flexible application, named INDELible, which can be used to generate nucleotide, amino acid and codon sequence data by simulating indels (under several models of indel length distribution) as well as substitutions (under a rich repertoire of substitution models).

In particular, I introduce a simulation study that makes use of one of INDELible's many unique features to simulate data with indels under codon models that allow the nonsynonymous/synonymous substitution rate ratio to vary among sites and branches. This data is used to quantify, for the first time, the precise effects of indels and alignment errors on the false-positive rate and power of the widely used branch-site test of positive selection. Several alignment programs are used and assessed in this context. Through the simulation experiment, I show that insertions and deletions do not cause the test to generate excessive false positives if the alignment is correct, but alignment errors can lead to unacceptably high false positives. Previous selection studies that use inferior alignment programs are revisited to demonstrate the applicability of my results in real world situations.

Further work uses simulated data from INDELible to examine the effects of tree-shape and branch length on the alignment accuracy of several alignment programs, and the impact of alignment errors on different methods of phylogeny reconstruction. In particular, analysis is performed to explore which programs avoid generating the kind of alignment errors that are most detrimental to the process of phylogeny reconstruction.

For my parents

Jeff and Sue

Acknowledgements

First and foremost, I would like to thank my supervisor Ziheng Yang for his guidance, direction and tutelage, without which this PhD would never have reached anything approaching this conclusion.

Secondly, I would like to thank my other half Jacqueline for her love, patience and support which was crucial in successfully maintaining my happiness, sanity and motivation over the last four or so years.

Also worthy of particular thanks are the department (CoMPLEX) & research council (EPSRC) that funded my PhD, and everyone responsible for conceiving and maintaining the vast computational resource that is UCL's high-performance computing cluster LEGION. Without the former I would never have embarked on this PhD, and without the latter it either wouldn't have been completed before the deadline, or would have had to be greatly simplified in its scope and complexity.

I also wish to thank my mother Sue, my brother Sam, and the many other friends, family and colleagues whose company I have enjoyed over the last four years and who have shown me their patience and encouraged or helped me in too many different ways to name. Thank you to you all.

At this stage I am also required to mention that the work presented in chapters 2 and 3 of this thesis has been published in the following publications:

Fletcher, W. and Yang, Z. 2009. **INDELible: a flexible simulator of biological sequence evolution.** Mol. Biol. and Evol. 2009 26(8):1879-1888

Fletcher, W. and Yang, Z. 2009. **The Effect of Insertions, Deletions and Alignment Errors on the Branch-Site Test of Positive Selection.** Mol. Biol. and Evol. 2010 27 (10): 2257-2267.

Table of Contents

List of Tables

List of Figures

Abbreviations Used

AIC: Akaike Information Criterion

AUC: Area Under Curve

BEB: Bayes Empirical Bayes

BIC: Bayes

ECM: Empirical Codon Model

FST: Finite State Transducer

LRT: Likelihood Ratio Test

ML: Maximum Likelihood

MP: Maximum Parsimony

MSA: Multiple Sequence Alignment

NJ: Neighbour Joining

PBR: Proportion of Branches Recovered

PSD: Positive Semi-Definite

ROC: Receiver Operating Characteristic

SPS: Sum-of-Pairs Score

TB: Tree Building

TC: Total Column score.

Chapter 1

Introduction

1.1 Background

Describing, organising and explaining the earth's biological diversity has always been a goal of naturalists, but it was a goal pursued in a framework that was independent of evolutionary thought for many centuries. That changed on July 1st 1858 when Darwin and Wallace's theory of evolution by natural selection was read at the Linnean Society of London. The central tenet of this theory is that heritable traits that will confer an advantage on an organism, making it more likely to survive and successfully reproduce, will become more prevalent in a population over successive generations. The publication of this controversial idea, *The Origin of Species* (1859), revolutionised biological thinking and sparked fierce scientific, moral and political debate.

1.1.1 Studying Evolutionary History

Within just a few years of *The Origin of Species*, scientists were already attempting to reconstruct the evolutionary history of all the Earth's organisms and describe them in the form of an evolutionary tree, or phylogeny (Haeckel 1866), and thus the field of Phylogenetics was born. Unfortunately, most early phylogenetic studies were little more than inspired guesswork. Generally they were amalgamations of plausible assertions from experts on particular taxonomic groups, and were based on few (if any) objective criteria. This did not change for nearly a century because systematists at that time were primarily concerned with questions about species classification, species diversity, speciation, and geographic variation rather than with questions about phylogenetics and the estimation of the evolutionary history that underlies that diversity.

However, this began to change in the middle of the 20th century. Scientists such as Willi Hennig (1950; 1966) began to define objective methods for rebuilding phylogenies based on the common characteristics and attributes (morphology) of both extinct and extant organisms. These methods were cultivated into explicit criteria for estimating phylogenies in the 1960's and computer programs were soon written to implement algorithms based on these criteria, allowing analysis of large and complex datasets. Around the same time, advances in molecular biology meant that molecular sequences began to take over as the primary source of information which scientists could

use to inform their attempts at reconstructing evolutionary history. Early statistical methods for analysing evolutionary history which made use of molecular sequences fell into the cladistic paradigm begun by Hennig. In this type of study, scientists sought the most parsimonious evolutionary tree that could account for the descent of certain characteristics (e.g. the states of the four nucleotides, Fitch 1971) whilst inferring the minimum number of evolutionary steps. Such maximum parsimony methods, which make use of this minimum change criterion, have been in widespread use for decades. However, they suffer from an inability to account for well-known phenomena such as convergent or parallel evolution. There may be many reasons why two organisms share a trait that was not present in their last common ancestor. Therefore, taking the existence of this shared trait as evidence of an evolutionary relationship could often lead one to infer an incorrect evolutionary history.

More recent likelihood methods (Felsenstein 1981) take a different approach and aim to model sequence evolution explicitly as a probabilistic process. The parameters that define the process are then estimated and the evolutionary tree which maximises the probability of observing the sequence data is chosen (Yang et al. 1995). Such approaches have advantages over parsimony analyses because, for example, they allow the use of likelihood ratio tests to compare different evolutionary hypotheses in order to see which hypothesis fits the data best (e.g. Golding, Felsenstein 1990; Huelsenbeck, Bull 1996). In addition, the process of parametric bootstrapping (Felsenstein 1985) can give a rough error associated with the estimate of the tree (e.g. Huelsenbeck et al. 1996a).

Over the decades since the introduction of the cladistic and likelihood approaches, molecular phylogenetics research has experienced phenomenal growth. This is largely due to the concurrence of advances in computer manufacturing and molecular sequencing. Indeed, both the amount of sequenced molecular data available for analysis and the computational power available with which to perform such analyses has been growing exponentially for over 30 years. Furthermore, this trend shows no sign of abating (Goldman, Yang 2008) which makes it an exciting time to be involved in the field.

1.1.2 Studying the Mechanisms behind Evolution

Darwin knew that natural selection could only be properly described in terms of heritable traits, but his theory lacked a mechanism to explain the transfer of biological information between generations. Furthermore, modes of heredity were poorly understood at that time. It was an Augustinian monk, Gregor Mendel, who first shed light on the matter when he published work based on his breeding experiments involving pea plants (Mendel 1866).

Mendel observed that the inheritance patterns of particular traits obeyed simple statistical rules, and whilst not all features displayed these patterns, now known as *Mendelian inheritance*, his work was the first to show that applying statistics to inheritance could prove highly useful. Mendel's work would remain largely unnoticed by his contemporaries. It wasn't until the turn of the 20th century that it was re-discovered but, even then, most evolutionary discussion and research still used Darwin as a starting point.

This changed in the early 20th century with the birth of modern quantitative genetics, marked by Ronald Fisher's (1918) publication of a genetic model that suggested Mendelian inheritance could be used to explain continuous variation of characters. Through the 1920's Fisher, and other pioneers in population genetics such as Sewall Wright and J.B.S. Haldane, laid the foundations for the theory of *neo-Darwinism* which emerged in the 1930's. This theory demonstrated that Mendelian genetics could be consistent with evolution and Darwinian natural selection. Therefore, it was sometimes known as the *modern evolutionary synthesis* (Huxley 1942), because it served to bridge the gap between naturalists and experimental geneticists. Natural selection began to be seen as the predominant force in shaping genetic makeup, and mutation was accepted as the major cause of genetic variation. This remains, to some extent, the paradigm in which modern evolutionary biology operates.

The landmark discovery of the structure of DNA (Deoxyribonucleic acid) (Watson, Crick 1953) that followed directly led to the *central dogma of molecular biology* (Crick 1958; 1970); that DNA makes RNA, RNA makes proteins, and proteins determine the traits of an organism. DNA replication provides the biological mechanism (that Darwin and Mendel had lacked) to explain the heritability of those traits, and changes to DNA during replication can explain genetic variation from

generation to generation. Subsequent advances in molecular techniques opened the door for scientists to challenge Darwin's hypothesis by searching for evidence of natural selection on the molecular level. A paper by Mitoo Kimura (1968), and another by Jack King and Thomas Jukes (1969), provocatively titled "Non-Darwinian Evolution", marked the birth of the *neutral* theory of evolution (Kimura 1983). This theory argued, by means of species comparisons, that amino-acid substitutions were mostly selectively neutral and the result of random genetic drift, as opposed to natural selection.

This contradiction of Darwin's theory was seen as controversial at the time since neo-Darwinism dominated mainstream evolutionary thought. However, the neutral theory has become generally accepted in the present-day and is commonly used in evolutionary biology as a null hypothesis that scientists seek to reject when they are searching for adaptive mutations. Thus, by the 1970's, scientists were already pursuing one of the two main goals of modern studies of evolution at the molecular level, namely, investigating the forces and mechanisms behind the evolutionary process itself. Much progress has been made since then.

We now know that DNA does not evolve only by means of single point substitutions that change the nucleotide on the copied strand, but also by means of insertions and deletions (collectively known as *indels*) of nucleotides that add or remove nucleotides from the copied strand. Whilst less understood than substitutions, it is generally accepted that indels tend to be small in size and that their length can be well described by a power law (e.g. Fan et al. 2008) – i.e. an indel of length u occurs with a probability that is proportional to u^{-a} for some value of the parameter a. There is also evidence that the two evolutionary processes interact – the level of nucleotide divergence has been found to be significantly correlated with the number and size of nearby indel events in a variety of genomes including those of primates, rodents, fruit fly and yeast (Tian et al. 2008). DNA can also evolve by large scale rearrangements of the DNA known as recombination events (McGlynn 2004; Kreuzer 2005) although the mechanisms for these types of events are probably the least well understood. Whilst attempts have been made to mathematically model the processes of insertion and deletion (see section "1.4.8 Statistical Approaches to Multiple Sequence Alignment" below), and recombination (e.g. Ringrose et al. 1998), the process of point mutations is by far the most extensively studied (and modelled) and shall be described in more detail in the following sections of this chapter.

1.2 Mathematical Models of Evolution

There are two main approaches to modelling the evolution of biological sequences. One approach is to construct an *empirical* model by making comparisons between large numbers of real sequences, and calculating certain properties of, or observed patterns between, the sequences. These patterns (e.g. relative rates of mutation between different amino-acids) are then fixed in the model and there is little to no way for the model to be influenced by the data which it is used to analyse. Most models of amino-acid substitution are empirical. An alternative approach is to define a *mechanistic* model whose form and structure is motivated by the biological or chemical properties of the type of sequence data that it is used to analyse. The obvious advantage of this type of model is that parameter values are not fixed and may be independently derived from the dataset in question. The majority of nucleotide models are mechanistic.

It is generally accepted that such mechanistic models provide a statistically superior fit to observed sequence data than empirical models. However, empirical models still have their place. In particular, they are far less computationally intensive and appear to be at least equally efficient as parametric models when it comes to reconstructing phylogenetic trees (p41, Yang 2006). Regardless of whether a probabilistic model of evolution is empirical or mechanistic in nature they are normally implemented by means of a continuous time Markov chain (CTMC). Below is some background about CTMCs in the context of biological sequence evolution that will be useful in the later sections which describe specific models.

1.2.1 Continuous Time Markov Chains

Normally, positions in a sequence are assumed to evolve independently of each other, and substitutions at any particular position are then described by a CTMC with a finite number of states. The defining feature of the CTMC is the 'memoryless' or 'Markovian' property that: the conditional probability distribution describing the states of the process depends only on the present state, i.e. "given the present, the future does not depend on the past". Given this basic assumption, the parameters of the substitution model are used to define an instantaneous rate matrix $\boldsymbol{Q} = \{q_{ij}\}$ that

describes the rate of change of the probability that a given sequence position is in a given state (i.e. $q_{ij}\delta_t$ can be thought of as the probability that a given sequence position changes from state *i* to state *j* in an infinitesimally small time interval δ_t). The diagonals of this matrix are defined by the mathematical requirement that the rows sum to zero, so $q_{ii} = -\sum_{j \neq i} q_{ij}$.

This rate matrix is then used to calculate a second matrix $\boldsymbol{P}(t) = \{p_{ij}(t)\}$ that describes the probability $p_{ij}(t)$ that state *i* changes to state *j* over some time interval $t > 0$. These transition probabilities can be calculated from the rate matrix by solving a set of linear ordinary differential equations known as the *Kolmogorov* forward equations:

$$\frac{dp_{ij}(t)}{dt} = \sum_k q_{ij} p_{jk}(t) \quad \forall\, i, j$$

or in matrix notation:

$$\frac{d\boldsymbol{P}(t)}{dt} = \boldsymbol{P}(t) \cdot \boldsymbol{Q}$$

If we exponentiate $\boldsymbol{Q}$, and remember that in the case of no evolution ($t = 0$) we have $\boldsymbol{P}(0) = \boldsymbol{I}$ (the identity matrix), then the matrix of transition probabilities is given by:

$$\boldsymbol{P}(t) = \{p_{ij}(t)\} = \exp(\boldsymbol{Q}\, t) = \sum_{k=0}^{\infty} \frac{(\mathbf{Q}t)^k}{k!}$$

If the CTMC is time-homogenous, so that it is described by a single rate matrix $\boldsymbol{Q}$, then the *limiting distribution* of the CTMC is defined as: $\lim_{t \to \infty} p_{ij}(t) = \pi_j$ where $\sum_j \pi_j = 1$.

In other words, as $t \to \infty$, π_j is the probability that a given site in a sequence is in state *j* regardless of the initial state at that site. For this reason the limiting distribution is also known as the *stationary* or *equilibrium* distribution.

If the CTMC is *time-reversible* then for all *t* it will obey the *detailed balance* condition $\pi_j p_{ji}(t) = \pi_i p_{ij}(t)$. The reversed CTMC has the same transition probability matrix as the original CTMC. Equivalently, this may be written $\pi_j q_{ji} = \pi_i q_{ij}$, i.e. the expected amount of change from state

i to state *j* is equal to the expected amount of change from state *j* to state *i*. Most substitution models are time-reversible.

Markov chain models satisfy an equation known as the *Chapman-Kolmogorov* theorem which states that the probability that a site in state *i* will be in state *k* a time $t_1 + t_2$ later is equal to the sum over all potential states *j* that could have occurred at time t_1, i.e.

$$p_{ik}(t_1 + t_2) = \sum_j p_{ij}(t_1) p_{jk}(t_2)$$

Hence the transition probabilities $\boldsymbol{P}(t) = \{p_{ij}(t)\}$ account for all possible evolutionary paths that a given site in a sequence might have taken. This is one of the main advantages that Markovian models have over simpler parsimony models. Parsimony models cannot, for example, accommodate the possibility that a site may have changed multiple times in a given time period, or may infer no change has taken place if a site changes state then reverts back to its original state.

It should be noted that the rate ($\boldsymbol{Q}$) and time (t) only appear as a product ($\boldsymbol{Q}t$) in the transition probability matrix. Therefore $\boldsymbol{Q}$ specifies relative rates only and it is impossible to distinguish between a sequence evolving at a given rate for a given time, or the same sequence evolving at twice the rate for half the time. In addition, the rate matrices are over parameterised, so one of the q_{ij} is normally set equal to 1 and the rate matrix $\boldsymbol{Q}$ is rescaled, prior to calculating $\boldsymbol{P}(t)$, by dividing every entry by a scale factor *s* that corresponds to the mean instantaneous substitution rate. For a homogenous-sites model this is given by: $s = \sum_i \sum_{j>i} (\pi_i q_{ij} + \pi_j q_{ji})$. This has the effect that the average instantaneous substitution rate of $\boldsymbol{Q}$ becomes equal to 1, and means that *t* now represents the evolutionary *distance* measured in expected number of substitutions per site.

1.2.2 Mechanistic Models of Nucleotide Substitution

When modelling nucleotide substitution, the state space of the Markov chain is given by the four nucleotides: thymine (T), cytosine (C), adenine (A) and guanine (G). The rate matrices $\boldsymbol{Q} = \{q_{ij}\}$ of four of the simplest nucleotide substitution models are shown in Figure 1.1.

The first (and simplest) Markov model of molecular evolution is known as JC69 (Jukes, Cantor 1969) and assumes that the rate of substitution of any one nucleotide to another is equal. For the JC69 model, solving the Kolmogorov equations implies that the probability that a site in state *i* changes to state *j* after a given time *t* is given by:

$$p_{ij}(t) = \begin{cases} \frac{1}{4} + \frac{3}{4}e^{-t} & (i = j) \\ \frac{1}{4} - \frac{1}{4}e^{-t} & (i \neq j) \end{cases}$$

We can see that as $t \to \infty$, the equilibrium probability of a site being in state *j* is given by:

$p_{ij}(t) \to \pi_j = 1/4$ for JC69 no matter what state *j* is.

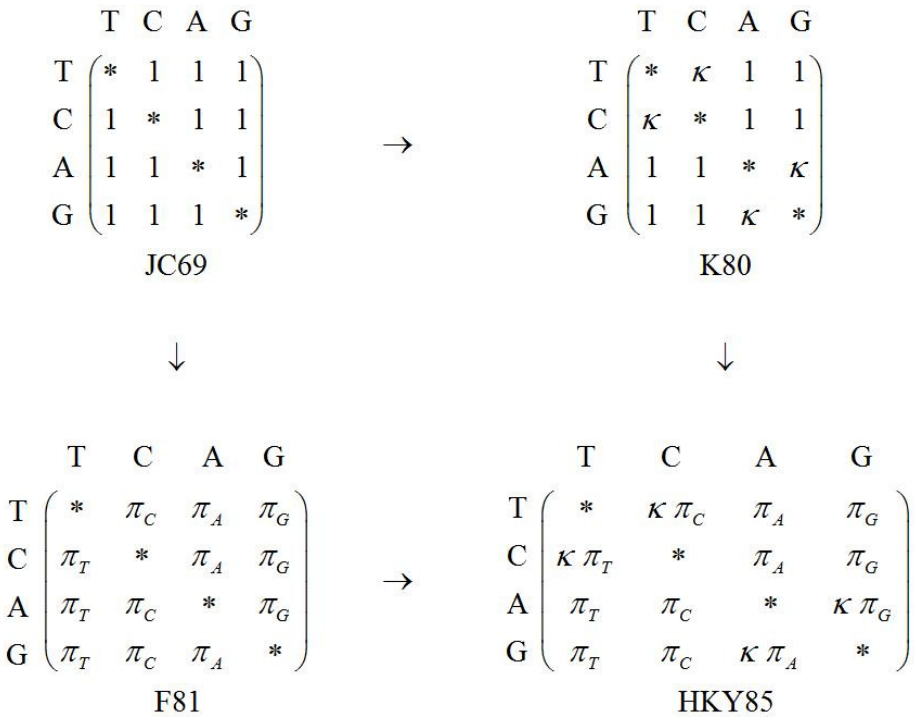


Figure 1.1: The rate matrices of four simple models of nucleotide substitution

The rate matrices on the bottom row allow unequal stationary nucleotide frequencies, whereas those on the top row do not. Similarly, the rate matrices on the right allow for transitions and transversions to occur at a different rate, whilst those on the left do not. The diagonal elements have been replaced by *, but they are defined so that the sum of each row is zero.

A second model known as F81 (Felsenstein 1981) extends JC69 by allowing distinct stationary base frequencies π_T, π_C, π_A and π_G that act as weighting factors in the model, causing certain bases to be more likely to arise when a substitution happens. This can be seen as an improvement over JC69 since it has long been known that the four nucleotides do not occur in equal proportions (e.g. Subak-Sharpe 1967). A third model, known as K80 (Kimura 1980), extends JC69 in a different manner by allowing the rate of transition substitutions between two biochemically similar pyrimidines (T $\leftrightarrow$ C) or purines (A $\leftrightarrow$ G) to differ from the rate of transversion substitutions between the two groups (purine $\leftrightarrow$ pyrimidine). κ is therefore referred to as the transition/transversion rate ratio. Again, this is an increase in realism because there exists a well documented excess of observed transitions over observed transversions in real data (e.g. Jukes 1987; but for a counterexample see Keller et al. 2007) .

The HKY85 model is the logical integration of the K80 and F81 models and allows both unequal nucleotide frequencies and a transition/transversion rate ratio that is different to 1. There is another model, known as F84 (Felsenstein, DNAML program since 1984, PHYLIP version 2.6) that is essentially equivalent to the HKY85 model, although it is calculated in a slightly different manner. It should be noted that as a substitution model becomes more complex the solution is still found in the same manner, i.e. by solving the Kolmogorov equations. Further generalisations of HKY85 exist, such as the TN93 model (Tamura, Nei 1993) that allows the transition rate between pyrimidines to differ from the transition rate between purines. The least specific reversible rate matrix is that of the General Time Reversible (GTR) model (Tavaré 1986) given by:

$$\boldsymbol{Q} = \{q_{ij}\} = \boldsymbol{S}\Pi = \begin{pmatrix} * & a & b & c \\ a & * & d & e \\ b & d & * & f \\ c & e & f & * \end{pmatrix} \cdot \begin{pmatrix} \pi_T & 0 & 0 & 0 \\ 0 & \pi_C & 0 & 0 \\ 0 & 0 & \pi_A & 0 \\ 0 & 0 & 0 & \pi_G \end{pmatrix} = \begin{pmatrix} * & a\,\pi_C & b\,\pi_A & c\,\pi_G \\ a\,\pi_T & * & d\,\pi_A & e\,\pi_G \\ b\,\pi_T & d\,\pi_C & * & f\,\pi_G \\ c\,\pi_T & e\,\pi_C & f\,\pi_A & * \end{pmatrix}$$

where $\boldsymbol{S}=\{s_{ij}\}$ is a symmetrical matrix and $\Pi = \mathrm{diag}\{\pi_T, \pi_C, \pi_A, \pi_G\}$ is the diagonal matrix containing the equilibrium nucleotide frequencies.

The obvious conclusion of this generalisation process is UNREST – the irreversible and unrestricted model of Yang (1994a) which places no constraint on the form of $\boldsymbol{Q}$. The equilibrium frequencies (π_i) for this model must be calculated by solving the system of simultaneous equations $\sum_i \pi_i q_{ij} = 0$ for all j, subject to the constraint $\sum_i \pi_i = 1$. Yang used this model to estimate the pattern of nucleotide substitution, which had previously only been done in a parsimony framework (e.g. Gojobori et al. 1982).

Functional proteins are produced from coding DNA sequences consisting of nucleotide triplets called codons (see section 1.2.5). There is much evidence that the evolutionary rates of sites at each position in the codon are highly auto-correlated (Yang 1995b) with a three position offset. That is to say, a nucleotide at position 1, 2 or 3 in a particular codon will tend to evolve at the same rate as a nucleotide at position 1, 2 or 3 in another codon. The third codon position evolves the fastest and so is most likely to provide useful phylogenetic signal when sequences are closely related and highly similar. However, substitutions at the third position can quickly become saturated so instead the first and second codon positions (which evolve more slowly) can be used to infer relationships between more divergent species. There is some evidence to suggest that such *codon position* models are statistically superior to nucleotide models (Shapiro et al. 2005; Bofkin, Goldman 2006).

1.2.3 Empirical Models of Amino-Acid Substitution

When modelling protein sequences, the unit of evolution is the amino-acid and the state space of the Markov chain is the twenty standard amino-acids used by cells in protein biosynthesis.
Amino-acid sequences are less prone to substitutional saturation than nucleotide sequences, making them preferable to use when inferring deep-level phylogenies (e.g. Ren et al. 2005). However, the much larger state space alphabet results in a computational problem of far greater complexity. The result of this is that the small datasets typically used in phylogenetic analyses may not contain enough information to reliably estimate amino-acid replacement rates using a mechanistic type model such as GTR (GTR amino-acid models have 208 parameters that must be estimated versus only 8 parameters for GTR nucleotide models). However, simple and unrealistic mechanistic models do exist. Two

examples are the Poisson and Equalin models which are essentially generalisations of the nucleotide models JC69 and F81 respectively. Therefore, fixed empirical amino-acid rate matrices that have been pre-estimated from large datasets are normally used instead. Such empirical models are constructed under the GTR framework given in the last section and the $s_{ij} = s_{ji}$ are often referred to as the *amino-acid exchangeablilities* (Whelan, Goldman 2001).

The grandfather of all empirical amino-acid substitution models is that constructed by Dayhoff *et al*. (1978). They analysed 71 protein sequences and used the principle of parsimony to estimate a phylogeny then infer ancestral sequences at the interior nodes. To reduce the effect of multiple hits and parallel and backward substitutions, which parsimony ignores, Dayhoff *et al.* chose only to analyse closely related sequences which shared > 85% identity. They then tabulated the relative frequencies of the 1572 different observed amino-acid changes and used this to approximate $\boldsymbol{P}(0.01)$, the transition probability matrix for an expected distance of 0.01, also known as *1 PAM* (Percent Accepted Mutation). Other PAM matrices were constructed by repeatedly multiplying the PAM1 matrix by itself, for example, PAM250 is equivalent to an average of 2.5 substitutions per amino-acid site. The PAM matrices remained the defacto standard for over a decade and were commonly used to score matches and mismatches during the process of amino-acid alignment.

In 1992 a flurry of activity saw three new scoring matrices published, motivated largely by the great increase in size of protein databases available by that time. The first, the JTT matrix (Jones et al. 1992) was constructed in the same manner as Dayhoff and colleagues but using a larger collection of 16,130 sequences containing some 59,190 mutations. The second matrix, known as the Gonnet matrix (Gonnet et al. 1992), resulted from the exhaustive matching of the entire 1991 SwissProt database with itself and included sequences which differed by 6.4 to 100 PAM units – this matrix is used for amino-acid alignments in the latest version of perhaps the most widely used alignment program ClustalW (Thompson et al. 1994). The third set of matrices are known as the BLOSUM (BLOcks of AminoAcid SUbstitution Matrix) matrices (Henikoff, Henikoff 1992) and were constructed by scanning the BLOCKS database for conserved regions of protein families then using these to calculate the amino-acid frequencies and substitution rates. There are different BLOSUM matrices corresponding to different divergence thresholds used in the Henikoff's clustering

method, the most common being BLOSUM62, where sequences with greater than 62% identity were clustered together and their contribution to the relative replacement rates was downweighted. This reduced the relative influence of the many highly related sequences that may have biased construction of the PAM matrices. In addition, matrices such as BLOSUM62 are all constructed from analysis of observed alignments which is in stark contrast to the analogous PAM250 matrix which is extrapolated from a matrix constructed from very closely related sequences. These two reasons appear to have resulted in the BLOSUM matrices being more suitable for matching divergent sequences (Altschul 1991) and the vast majority of alignment programs in use today implement BLOSUM62 to score matches and mismatches between protein sequences. Interestingly, it was shown in 2008 that the computer code used to construct the BLOSUM matrices actually contained errors, which had gone unnoticed for 15 years, and the published matrices are actually quite different from those that should have been produced by the Henikoff's algorithm, if it had been implemented correctly. Whilst computer bugs are quite a common occurrence this case is worthy of note because the BLOSUM matrices, in particular BLOSUM62, are ubiquitous in comptational biology and, curiously, the "incorrect" matrices perfom better than the "intended" matrices (Styczynski et al. 2008)!

One of the main problems with the aforementioned parsimony approach to estimating amino-acid replacement rates is that the method has no time structure, such that the probability of a substitution occurring is equally likely in a short or a long time interval. This means that, even though a tree topology is used in the analysis, inferred amino-acid changes are merged across all branches with no regard for the branch length, which is obviously very important when it comes to determining the number of changes that have occurred on a given branch. This lead Adachi and Hasegawa (1996) to estimate amino-acid replacement rates using the same maximum likelihood method employed by Yang (1994a) for nucleotides, and this is the general procedure that has been employed for most published matrices since then (although see Müller, Vingron 2000 who use the 'resolvent' method). The WAG model is an update to the earlier PAM and JTT matrices that was constructed using the superior maximum likelihood methodology, and outperformed both of the earlier matrices with respect to maximum-likelihood values of database protein families (Whelan, Goldman 2001). In a similar manner, the more recent LG model provides further subsequent improvement over the WAG

model (Le, Gascuel 2008), as judged by the Akaike information criterion, or AIC (Akaike 1974), and frequently leads to inference of different tree topologies.

The models described so far have all been based on analysis of nuclear proteins, and replacement rates in other types of proteins can differ considerably. This has lead to a plethora of other models. Some early examples were mtREV (Adachi, Hasegawa 1996) and MTMAM (Yang et al. 1998), which were based on vertebrate and mammalian mitochondrial proteins respectively, and CpREV (Adachi et al. 2000) which was based on chloroplast proteins. Since the replacement rates for these models are fixed, and the models were estimated from quite specific datasets, this may lead to concerns about how applicable the models are to other types of dataset. Such concerns directly lead to the development of models for mitochondrial proteins for different datasets such as arthropods (MtArt: Abascal et al. 2007; MtPan: Carapelli et al. 2007), narrower datasets such as fish (MtPip: Kitchovitch et al. 2009), or broader datasets such as metazoans (MtZoa: Rota-Stabelli et al. 2009). Specific models have also been developed for viral reverse transcriptase proteins (RtREV: Dimmic et al. 2002), HIV-1 viral genes (HIVb and HIVw: Nickle et al. 2007), and the influenza virus (I09: Dang et al. 2009). All of these models purport to be a statistically better fit to the type of data they were developed for, and this is likely to fuel the development of further empirical models in the future.

A common variation to these empirical models is to replace the equilibrium frequencies (the π_j) of the model with the observed frequencies of the different amino-acids present in the data being analysed, whilst still using the amino-acid exchangeabilities (the s_{ij}) from the model (Cao et al. 1994). This often considerably improves the fit of the model to the data. Such variations are denoted with a suffix "+F" so, for example, the WAG model variation would be called WAG+F. Other types of empirical amino-acid model have also been developed based on hydrophobicity of different amino acids (e.g. Kyte, Doolittle 1982) or their structure (e.g. Bordo, Argos 1991) although these types of model are rarely used in phylogenetics. Another, potentially more useful, group of matrices aim to detect frameshift mutations that have created new coding sequences, but may also be used to identify sequencing errors (Claverie 1993).

1.2.4 Rate Heterogeneity Among Sites

The models described so far all assume that the rate of substitution is constant over sites in a sequence. This assumption may not be valid for real data where mutation rates may vary over sites, and mutations may accumulate at different rates in different parts of a sequence due to structural or functional constraints. There is much evidence in the literature that suggests variable substitution rates is a real phenomenon (e.g. Fitch, Margoliash 1967b; Uzzell, Corbin 1971; Holmquist et al. 1983; Fitch 1986) and failing to account for rate variation has been shown to lead to drastic effects such as underestimation of evolutionary distance or transition rate bias, as well as incorrect reconstruction of phylogenies (Yang 1996).

One approach to dealing with this is to assume that sites belong to categories which have different rates. The simplest model of this type assumes that a proportion of sites are invariant while the others are evolving at a constant rate (the "+I" model: Hasegawa et al. 1985), but it has been found that a two-rate-category model could not provide an adequate fit to real data (Wakeley 1993). One problem with this approach is that the estimate of p_0, the proportion of invariant sites, is very sensitive to the sampling of taxa. This is because p_0 will never be larger than the observed proportion of constant sites, and so when more divergent sequences are added to the analysis the proportion of observed 'constant' sites becomes lower and estimates of p_0 tend to decrease as well (pp. 113-114, Yang 2006). A three-rate-category approach has also been implemented (Hasegawa et al. 1993; Felsenstein, Churchill 1996) but appears to introduce too many parameters to be estimated.

A second approach is to assume that sites have rates drawn from a continuous distribution such as a log-normal distribution (Olsen 1987; Waddell et al. 1997). Alternatively, the gamma distribution has been found to provide a statistically acceptable fit to data (Wakeley 1993) and has been used in many studies that aimed to estimate distances between two sequences (e.g. Tamura, Nei 1993). However, it was Yang (1993) who first described the use of a continuous gamma distribution in a joint likelihood analysis (the "+Γ" model). The gamma distribution is given by:

$$g(r,\alpha,\beta) = \beta^{\alpha} e^{-\beta r} r^{\alpha-1} / \Gamma(\alpha)$$

where $\alpha > 0$ is the shape parameter and $\beta > 0$ is the scale parameter. The mean and variance of this

distribution are α/β and α/β^2 respectively. To avoid the use of too many parameters, Yang chose to set $\beta = \alpha$ such that the mean of the distribution is 1 and the variance is $1/\alpha$. Thus the shape parameter completely describes the new distribution, and is now inversely related to the degree of rate variation among sites (see figure 1.2) making the model easy to interpret. If $\alpha = 1$, then the distribution reduces to the exponential distribution with rate parameter $\beta = 1$. If $\alpha > 1$, then the distribution is bell-shaped, and as $\alpha \to \infty$ the distribution collapses into the model of a constant rate for all sites. If $\alpha \leq 1$, then the distribution is highly skewed with most sites evolving at a very low rate and some sites evolving with high rates.

Advantages of the continuous gamma model are that it is easy to interpret and the rate variation is described completely by one parameter. However, it is very computationally intensive and so is not practical to use on large datasets. This was the motivation behind the discrete gamma model

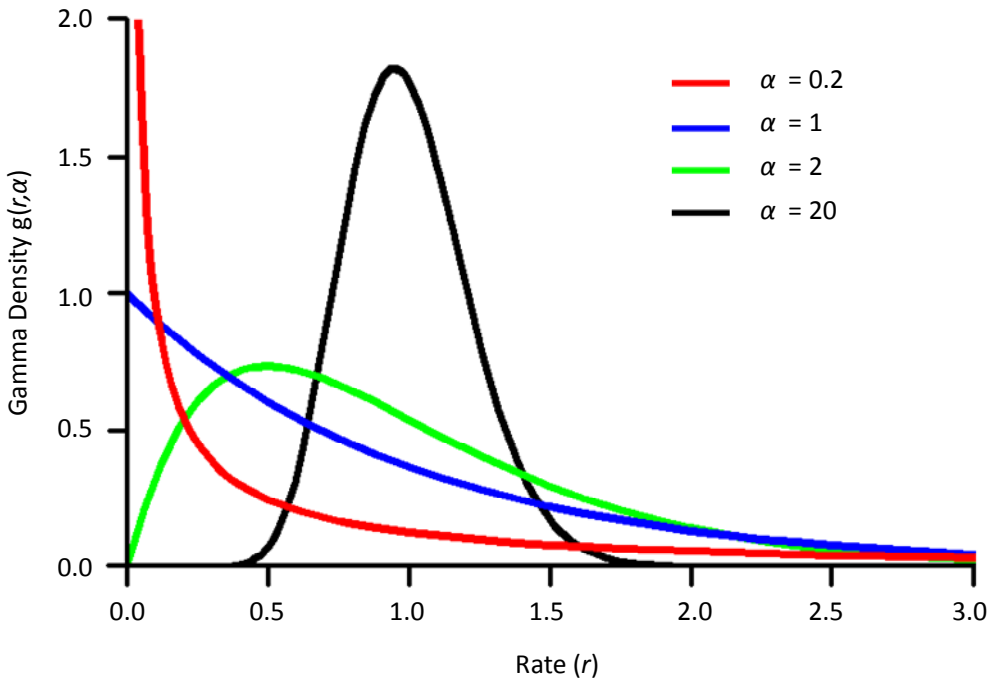


Figure 1.2: Probability density function of the Gamma distribution.

The scale parameter of the gamma distribution for variable rates among sites is fixed such that the mean rate is always equal to 1. Therefore, the distribution is fully specified by the shape parameter α. The *x*-axis is the relative substitution rate, and the *y*-axis is proportional to the number of sites with that rate. Adapted from Yang (2006).

of Yang (1994b) that uses *K* categories of equal proportion to approximate the continuous gamma distribution (denoted "$+\Gamma_K$") . The mean rate of a category is used to represent all rates in the category (figure 1.3). Yang (1994b) found that as few as four categories could be sufficient to provide an optimal or near-optimal fit to data. This approach appears to capture the best features of the continuous gamma model whilst also achieving a similar computational efficiency to the simpler discrete rate class models. As a result the method has been widely used.

A common variation of these models is to combine the invariable sites model with gamma-distributed rates for other sites (the "+I+Γ" and "$+I+\Gamma_K$" models) (Gu et al. 1995). However, this approach may not be worthwhile since the gamma distribution already permits very low rates at some sites (Golding 1983). The result of combining both methods in this way is that there is a strong correlation between p_0 and α so it is generally impossible to estimate both parameters accurately (Sullivan et al. 1999). An alternative approach to implementing a multi-modal distribution of rates is to use a mixture of gamma densities (Mayrose et al. 2005).

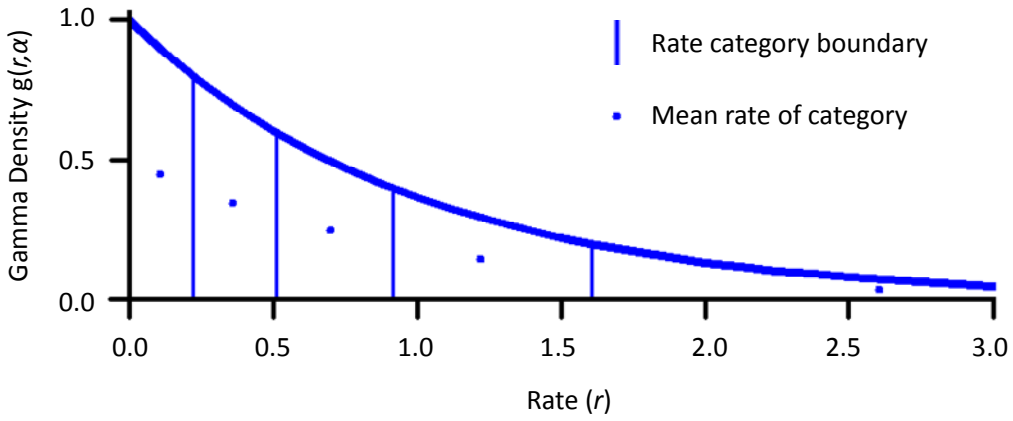


Figure 1.3: Discrete gamma model of variable rates across sites with $\alpha = 1$

An example of the discrete-gamma model of variable rates among sites with five equal-probability categories used to approximate the continuous gamma distribution. Shown here is the probability density function with shape parameter $\alpha = 1$. The four vertical lines are at *r* = 0.22, 0.51, 0.92 and 1.61 and represent the 20th, 40th, 60th, and 80th percentiles of the distribution, separating the rates in to five categories each with proportion 0.2. The mean rates in the five categories, given here by 0.11, 0.36, 0.70, 1.22, and 2.61, are used to represent all rates in that class.

1.2.5 Models of Codon Substitution

Functional proteins are produced from coding DNA sequences consisting of nucleotide triplets called codons. Since there are four possible nucleotides, this means there are 64 possible codons. During protein production an organism's genetic code is used to translate each codon into one of the 20 possible amino-acids. Thus, the genetic code is degenerate and more than one codon can code for the same amino-acid (table 1.1). Rather than being static the genetic code has evolved (Osawa, Jukes 1989), so it is reasonable for one to ask why the code has evolved into this degenerate form. Far from being wasteful, the degenerate assignment of amino-acids to multiple codons is highly non-random and extremely efficient – indeed, it has been suggested that only 1 in every million random alternative codes is more efficient (Freeland, Hurst 1998). It appears that as the code evolved, biochemically or physically similar amino-acids were assigned to codons close together in the code, presumably as a form of error-minimisation in order to lessen the impact of mutations and mistranslations of the reading frame (Freeland et al. 2003). Since the genetic code is degenerate this leads to two types of underlying nucleotide substitution – the synonymous or silent substitution (where the encoded amino-acid is conserved), and the non-synonymous (amino-acid altering) substitution (see figure 1.4).

Table 1.1: The 20 amino-acids that the 64 codons specify (Standard Genetic Code)

		SECOND BASE									
		T		C		A		G			
FIRST BASE	T	TTT	F (Phenylalanine)	TCT	S (Serine)	TAT	Y (Tyrosine)	TGT	C (Cysteine)	T	THIRD BASE
		TTC		TCC		TAC		TGC		C	
		TTA	L (Leucine)	TCA		TAA	Stop codon	TGA	Stop codon	A	
		TTG		TCG		TAG		TGG	W (Tryptophan)	G	
	C	CTT	L (Leucine)	CCT	P (Proline)	CAT	H (histidine)	CGT	R (Arginine)	T	
		CTC		CCC		CAC		CGC		C	
		CTA		CCA		CAA	Q (Glutamine)	CGA		A	
		CTG		CCG		CAG		CGG		G	
	A	ATT	I (Isoleucine)	ACT	T (Threonine)	AAT	N (Asparagine)	AGT	S (Serine)	T	
		ATC		ACC		AAC		AGC		C	
		ATA		ACA		AAA	K (Lysine)	AGA	R (Arginine)	A	
		ATG	(M) Methionine	ACG		AAG		AGG		G	
	G	GTT	V (Valine)	GCT	A (Alanine)	GAT	D (Aspartic acid)	GGT	G (Glycine)	T	
		GTC		GCC		GAC		GGC		C	
		GTA		GCA		GAA	E (Glutamic acid)	GGA		A	
		GTG		GCG		GAG		GGG		G	

NOTE – The codon ATG codes for the amino-acid Methionine but is also a start codon.

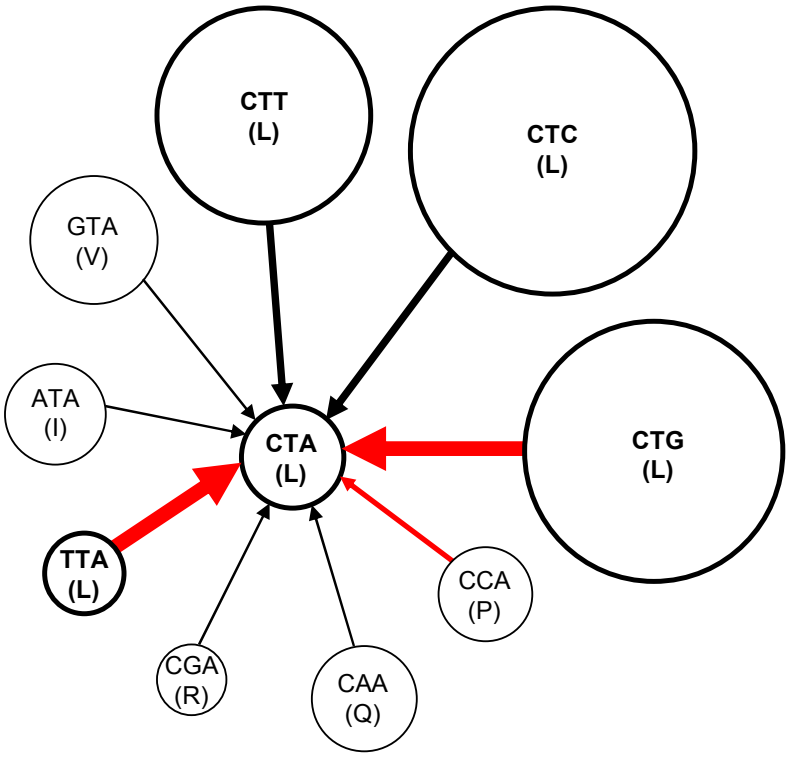


Figure 1.4: Substitution rates to the target codon CTA from its nine neighbours.

This is an example of which codons may instantaneously change into the codon CTA by means of just one nucleotide substitution. The amino-acid each codon specifies is shown in parentheses. The size of the circles represent the relative codon frequencies calculated from the base compositions at the three codon positions in a dataset of five α and β mammalian globin gene sequences (dataset abglobin.nuc in PAML, Yang 2007b). Codons that specify the same amino-acid as codon CTA are shown with bolder circles. Therefore, synonymous substitutions are represented by arrows from bold circles, and nonsynonymous substitutions are represented by arrows from normal circles. Red and black arrows indicate if the nucleotide substitution was a transition or a transversion respectively. The thickness of the arrows indicates the relative rates of substitution. This diagram was drawn using $\omega = 1/3$ and $\kappa = 2$ such that the four rates to the codon CTA are in the ratio 1:2:3:6, for nonsynonymous transversion ($q_{i,CTA} = \omega\pi_{CTA}$), nonsynonymous transitions ($q_{i,CTA} = \kappa\omega\pi_{CTA}$), synonymous transversion ($q_{i,CTA} = \pi_{CTA}$) and synonymous transition ($q_{i,CTA} = \kappa\pi_{CTA}$) respectively.

Adapted from Yang (2006).

Instead of modelling amino-acid substitution empirically at the protein level, it is possible to construct a mechanistic model at the codon level. For codon models the state space of the Markov chain consists of the sense codons of the genetic code, (e.g., 61 sense codons for the *universal* or *standard* genetic code, see table 1.1). Stop codons are not allowed inside a functional protein and so they are not considered in the Markov chain.

The basic codon model assumes mutations occur independently at the three codon positions and specifies the instantaneous rate of substitution from codon *i* to *j* as

$$q_{ij} = \begin{cases} 0, & \text{if } i \text{ and } j \text{ differ at two or three positions,} \\ \pi_j, & \text{if } i \text{ and } j \text{ differ by a synonymous transversion,} \\ \kappa\pi_j, & \text{if } i \text{ and } j \text{ differ by a synonymous transition,} \\ \omega\pi_j, & \text{if } i \text{ and } j \text{ differ by a nonsynonymous transversion,} \\ \omega\kappa\pi_j, & \text{if } i \text{ and } j \text{ differ by a nonsynonymous transition,} \end{cases}$$

where ω is the nonsynonymous/synonymous rate ratio, κ is the transition/transversion rate ratio, and π_j is the equilibrium frequency of codon *j* (Goldman, Yang 1994; Yang, Nielsen 1998).

The substitution rate between codons is proportional to the equilibrium frequency (the π_j) of the target codon. Several choices can be made about these parameters. The simplest and most unrealistic option would be to presume that the frequencies of all codons are equal (known as Fequal). Instead, the frequencies can be calculated from the observed frequencies of the three nucleotides, tabulated over the sequence as a whole (known as F1X4 - 1 set of 4 frequencies) or at each of the three codon positions (known as F3X4 - 3 sets of 4 frequencies). Explicitly, for a codon consisting of the nucleotide triplet *xyz* the frequencies are given by $\pi_{xyz} = \pi_x\pi_y\pi_z / k_1$ and $\pi_{xyz} = \pi_x^1\pi_y^2\pi_z^3 / k_3$ where π_m is the frequency of nucleotide *m* at all three codon positions, π_m^n is the frequency of nucleotide *m* at the *n*th codon position, and k_1 and k_3 are normalizing constants accounting for the presence of stop codons. The final and most parameter rich choice is to allow the frequencies of all codons to be different, subject only the constraint that they sum to 1 (known as F61).

It is difficult to compare nucleotide, amino-acid and codon models because of their differing data structure. However, methods for converting the lower-dimensional nucleotide and amino-acid models in to 64-dimensional codon models with nucleotide triplet substitution have been developed

(Seo, Kishino 2009). This allows comparison of these different types of models using traditional model selection criteria such as the Akaike information criterion (AIC) or the Bayesian information criterion (BIC), which are described further in section 1.3.7. Such comparisons suggest that codon models are generally superior to nucleotide or amino-acid models, but the increased dimensionality means the computational burden is too great to make tree searches in large datasets feasible.

However, the true utility of codon models lies in testing for adaptive evolution. If $\omega \approx 1$ this would imply that mutations that change the amino-acid are just as likely to occur as those that preserve the amino-acid, and if $\omega < 1$ this can interpreted as purifying selection acting to preserve the amino-acid sequence. If Kimura's theory of neutral evolution was true then we would not expect to find $\omega > 1$ as this represents evidence of positive selection where advantageous amino-acid changes have become fixed in the protein. Therefore, the null hypothesis in tests for adaptive evolution does not allow $\omega > 1$, whilst in the alternative hypothesis this is permitted. Thus, the null is nested within the alternative hypothesis and a Likelihood Ratio Test (LRT) can be used to compare whether the alternative is a significantly better statistical fit to the observed sequence data (see 1.3.7). More complicated models have been developed (Yang 1998) but they all share the common feature that the null model contains a distribution to describe $\omega \leq 1$ and is nested within the alternative model which contains an extra category allowing values of $\omega > 1$. Even more complicated models allow ω to vary along branches in the tree, among sites in the gene, or both along branches and among sites. These models are discussed further in chapter 3.

1.2.6 Improvements to Standard Codon Models

Despite the success of the standard codon model it does have limitations. For example, it has been known for some time that amino-acid residues that share similar physico-chemical properties such as composition, polarity and molecular volume, tend to replace each other more frequently than those that are dissimilar (e.g. Grantham 1974). This basic codon model ignores this fact, although attempts have been made before to incorporate dependence among codons because of the protein structure (Robinson et al. 2003). Similarly, the basic codon model does not allow instantaneous multiple

nucleotide substitutions, although this has been shown to be a real phenomenon (Averof et al. 2000) and their inclusion can lead to statistically significant improvements in the fit of models to data (the SDT model: Whelan, Goldman 2004). The unrestricted Empirical Codon Model (ECM), was constructed, in the same manner as the maximum likelihood empirical amino-acid models mentioned above, whilst permitting multiple nucleotide changes, and is also reported to outperform comparable mechanistic models (Kosiol et al. 2007).

However, this debate is far from closed as others have argued the case for successive single compensatory changes instead (e.g. Bayzkin et al. 2004). A different approach is that of the mechanistic-empirical-combined (MEC) method (Doron-Faigenboim, Pupko 2007) which allows the assimilation of empirically derived amino-acid replacement probabilities into a codon substitution matrix. The justification for such an approach is twofold. Firstly, differing rates of nonsynonymous/synonymous substitutions and transition/transversion substitutions can be accommodated together with the observation that distinct amino-acids differ in their replacement rates. Secondly, the model may be considered somewhat context-dependent as highly-specific amino-acid substitution matrices can be employed – possibly making the model more relevant to the data being analysed for the same reasons described in section 1.2.3. Again, the authors of this approach report an improved fit of the model to data.

Since synonymous codons that code for the same amino-acid occur at different rates in protein coding genes (Ikemura 1981; Ikemura 1985), it is a controversial and hotly debated topic as to whether mutation or selection is responsible for the origin or perpetuation of this codon usage bias (e.g. Duret 2002). This points to another problem with standard codon models, namely that they do not explicitly consider the separate effects of the two processes and so are incapable of inferring which is responsible for evolution at silent sites. The novel model FMutSel (Yang, Nielsen 2008) attempts to address this deficiency by separating the processes via the introduction of distinct codon-fitness and mutation-bias parameters. As before, an improved fit to data was observed as a result.

1.2.7 Context Dependent Substitution

Most of the simplifying assumptions originally introduced to methods of phylogenetic inference have been relaxed, resulting in models of increased power and realism. For example, the assumption that nucleotides substitute for each other at the same average rate and that this average rate is constant across sites, both gave way to more realistic alternatives. One assumption that largely persists to the present day – despite biological evidence to the contrary – is that sites in a sequence evolve independently of each other. However, the rate of nucleotide substitution has been found to vary as much as 10-fold for different pairs of flanking bases, with context-dependent rates varying by as much as 50-fold (Blake et al. 1992; Hess et al. 1994). The most dramatic instance of this variation is found in so-called CpG dinucleotide "hotspots" (Ehrlich, Wang 1981; Zhang et al. 2007).

The two immediate neighbours of a given site exert the strongest influence in such context effects. But, under certain conditions, the effect is also detectable for the second and third pairs of flanking bases (Morton et al. 1997), and subtle effects may even extend as far as 200 bp (Zhao, Boerwinkle 2002). In addition, observed dinucleotide frequencies significantly deviate from the expectation values calculated from the frequencies of individual nucleotides (Karlin, Mrazek 1997), in particular, the dinucleotide CpG is observed only 23% as much as expected (Gentles, Karlin 2001). It has been suggested that context biases may be correlated with this bias in dinucleotide frequencies that is observed in the human genome (Zhang, Gerstein 2003). There is strong evidence that simultaneous doublet-nucleotide substitutions occur with high frequency (Averof et al. 2000). The *doublet* model (Schoniger, von Haeseler 1994) attempts to account for this in a naïve manner by allowing changes from one doublet to another in a two-step process. First, one nucleotide is substituted for another according to a standard nucleotide model, then the matching nucleotide evolves according to the same model. Thus, there are no instantaneous doublet substitutions; a common doublet is only replaced by another doublet via an intermediate rare doublet.

Codon models are another simple way to model nucleotide context as they allow dependence between different sites of the codon triplet. However, context effects have been shown to be significant across codon boundaries (Siepel, Haussler 2004) and very low CpG frequencies are sometimes seen across codon boundaries which violates the hypothesis of independence among

codons (Pedersen et al. 1998) upon which most codon models rely (but see Robinson et al. 2003). Some researchers have argued that considering context dependence leads to better fitting models than using richer substitution models, or allowing rate heterogeneity across sites, under the assumption of site independence (Siepel, Haussler 2004). Despite this there have been relatively few attempts to model the more complicated forms of context-dependent substitution that are apparent in real biological data (for examples see: Jensen, Pedersen 2000; Hwang, Green 2004; Siepel, Haussler 2004; Lindsay et al. 2008; Misawa, Kikuno 2009). Therefore, this may be an active avenue of model development in the years to come.

1.2.8 Practical Calculation of the Transition Probability Matrix

Whilst closed form solutions of $P(t)$ exist for the simpler nucleotide models, the calculation becomes extremely cumbersome, if not completely intractable, when there are a larger number of states such as in codon and amino-acid models. Therefore, $P(t)$ is normally calculated through numerical computation of the eigenvalues and eigenroots of the rate matrix $\boldsymbol{Q}$ (Yang, 1995).

This is done by first noting that Q is similar to the symmetrical matrix $B = \Pi^{1/2} Q \Pi^{-1/2}$ (where $\Pi^{1/2} = \mathrm{diag}\left[\pi_1^{1/2}, \pi_2^{1/2}, \ldots, \pi_n^{1/2}\right]$, $\Pi^{-1/2}$ is the inverse of $\Pi^{1/2}$, and π_j is the jth stationary frequency). Since similar matrices have identical eigenvalues, one can also define $\Lambda = \mathrm{diag}\left[\lambda_1, \lambda_2, \ldots, \lambda_n\right]$ as the matrix with the common eigenvalues of Q and B. The matrix B can then be constructed and diagonalised as $B = R \Lambda R^{-1}$.

$$\Rightarrow Q = \Pi^{-1/2} B \Pi^{1/2} = \left(\Pi^{-1/2} R\right) \Lambda \left(R^{-1} \Pi^{1/2}\right)$$

where the right and left eigenvectors corresponding to the eigenvalues of Q in Λ are the columns of $U = \Pi^{-1/2} R$, and the rows of $U^{-1} = R^{-1} \Pi^{1/2}$, respectively. The transition probability matrix is then:

$$P(t) = \exp(Qt) = U \exp(\Lambda t) U^{-1} = U \mathrm{diag}\left[\exp\left(\lambda_1 t\right), \exp\left(\lambda_2 t\right), \ldots, \exp\left(\lambda_n t\right)\right] U^{-1}$$

This method of calculating $P(t)$ is not an advisable approach for irreversible rate matrices as, despite $P(t)$ being real, Q may have complex eigenroots and the numerical algorithm may become unstable

when complex numbers are involved. A simple workaround used for calculating the rate matrices of irreversible models, such as UNREST, is to use the slower and less accurate method of repeated matrix squaring (Yang 2006, pp 68-70). This approach makes use of the relationship:

$$e^{Qt} = \left(e^{Qt/m}\right)^m \approx A^m = \left(I + \frac{Qt}{m} + \frac{1}{2}\left(\frac{Qt}{m}\right)^2\right)^m$$

For a reasonably large *m,* the matrix *A* will have positive elements only, and will yield an accurate approximation to $P(t)$. Given a method of calculating $P(t)$ for a certain set of parameter values, the question then becomes which set of parameter values are "best" for a given dataset. The likelihood method (section 1.3) discussed below is an ideal method for determining "good" parameter values.

1.3 The Likelihood Method

Maximum Likelihood (ML) estimation is a methodology used to estimate parameters under a given model; it is a cornerstone of statistics and computational statistics and has been widely used in molecular phylogenetics since it was introduced by Felsenstein (1981). Formally, a full phylogenetic model consists of a tree topology τ containing branch length parameters $t = \{t_0, t_1, \ldots, t_n\}$ for the *n* branches of τ, and substitution model parameters (collectively termed θ) such as the state frequencies and transitition/transversion rate ratio. Informally the term 'model' is often loosely used to refer to the substitution model alone. In this section I shall begin by describing how to compute the likelihood for a model in simple cases where parameters are known, then extend this to the problem of parameter estimation, before discussing applications.

1.3.1 Computing the Likelihood for a Given Tree and Substitution Model

The assumption of independence among sites means that the likelihood *L* for a given set of sequences of length *n* is simply the product of the probability at each site. Given the substitution parameters θ and the branch lengths $t = \{t_0, t_1, \ldots, t_5\}$ the likelihood of observing the states $x^k = \{x_0, x_1, x_2, x_3\}$ at position *k* of the sequences at the tips of the rooted four taxon tree shown in figure 1.5 is

$$f_k = f\left(x^k \mid \theta, t\right) = \sum_{y_0}\sum_{y_1}\sum_{y_2}\left[\pi_{y_0} p_{y_0 y_1}\left(t_0\right) p_{y_0 y_2}\left(t_1\right) p_{y_1 x_0}\left(t_2\right) p_{y_1 x_1}\left(t_3\right) p_{y_2 x_2}\left(t_4\right) p_{y_2 x_3}\left(t_5\right)\right]$$

where π_{y_0} is the equilibrium frequency of state y_0 in the model, and $p_{ij}\left(t\right)$ is the probability of state *i* changing to state *j* after time *t* as defined by the transition probability matrix. The quantity in the square brackets is simply the probability of observing state y_0 at the root, multiplied by six different transition probabilities for the substitutions that took place in order to give rise to the observed site pattern x^k at the tips. Since $y = \left\{y_0, y_1, y_2\right\}$ is unknown, the likelihood is given by summing over all possible ancestral states. Thus, given the assumption of independence among sites, the likelihood *L,* for the four sequences of length *n* at the tips, is given by the product:

$$L = f_1 \times f_2 \times \ldots \times f_k \times \ldots \times f_n = \prod_{k=1}^{n} f_k$$

The likelihood becomes vanishingly small very quickly, and in any large analysis it would certainly become too small to be represented in computer memory. As a result the log-likelihood (log of the likelihood) is normally used instead and the product over sites becomes a sum over sites:

$$\ell = \log(L) = \log(f_1) + \log(f_2) + \ldots + \log(f_k) + \ldots + \log(f_n) = \sum_{k=1}^{n} \log(f_k)$$

1.3.2 The Likelihood Function When Rates Among Sites Are Variable

If the substitution model assumes gamma distributed rate heterogeneity then calculation of the probability of the observed data, given the phylogeny and the substitution model, proceeds in a similar manner. However, now the branch lengths are multiplied by a scale factor *r* (representing the faster or slower relative rate of evolution – see section 1.2.4), and the probability of the site pattern must be multiplied by the probability of drawing the value *r* from the gamma distribution. Thus, if the gamma distribution has shape parameter α and probability density $g_{r\alpha} = g(r, \alpha, \alpha)$, then the probability of observing the site pattern shown in figure 1.5 would now be given by:

$$f\left(x^k \mid \theta, t, r\right) = \sum_{y_0}\sum_{y_1}\sum_{y_2}\left[\pi_{y_0} p_{y_0 y_1}\left(r \cdot t_0\right) p_{y_0 y_2}\left(r \cdot t_1\right) p_{y_1 x_0}\left(r \cdot t_2\right) p_{y_1 x_1}\left(r \cdot t_3\right) p_{y_2 x_2}\left(r \cdot t_4\right) p_{y_2 x_3}\left(r \cdot t_5\right) g_{r\alpha}\right]$$

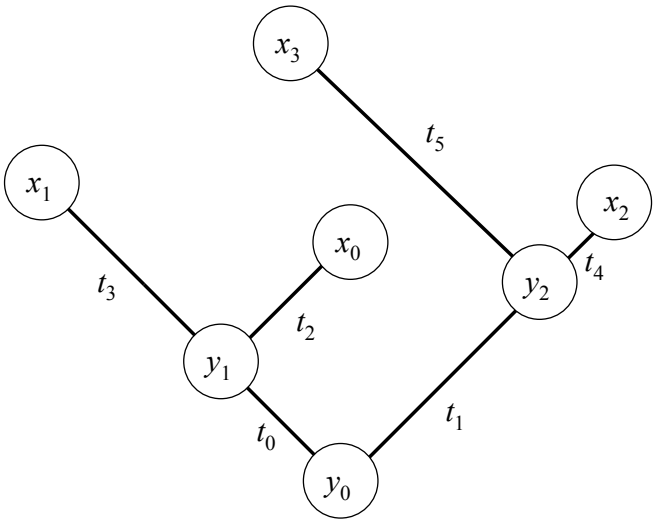


Figure 1.5: An example of one site in a rooted tree of 4 taxa

The observed states are given by the x_i, the ancestral states are given by the y_i, and the branch lengths in expected number of substitutions per site are given by the t_i.

1.3.3 Felsenstein's Pruning Algorithm and Other Computational Shortcuts

A rooted tree of *m* species has *m* – 1 internal nodes and 2*m* – 2 branches. Therefore, for data with *c* possible states there are c^{m-1} ancestral states on the tree which need to be averaged. Fully expanding a likelihood calculation of the form given by f_k leads to $c^{m-1} - 1$ additions and $c^{m-1} \times (2m-2)$ multiplications. For codon data with 61 states, on the four taxa tree in Figure 1.5 this amounts to a staggering $61^3 - 1 > 220,000$ additions and $61^3 \times 3 > 1,300,000$ multiplications.

However, we can drastically reduce the number of operations we need to perform by restructuring the likelihood sum and moving the summation $\sum$ symbols over to the right as far as possible.

$$f_k = f\left(x^k \mid \theta, t\right) = \sum_{y_0} \pi_{y_0} \left[\sum_{y_1} \left[p_{y_0 y_1}\left(t_0\right) p_{y_1 x_0}\left(t_2\right) p_{y_1 x_1}\left(t_3\right) \right] \times \sum_{y_2} \left[p_{y_0 y_2}\left(t_1\right) p_{y_2 x_2}\left(t_4\right) p_{y_2 x_3}\left(t_5\right) \right] \right]$$

In this simpler format our toy example now only involves the use of $5c + 1 = 306$ multiplications and $3c - 3 = 180$ additions, for the same single codon site on the four taxon tree. For datasets with more taxa and larger trees the computational savings are even more dramatic. This "nesting" principle amounts to performing the likelihood calculation by working backwards from the tips of the tree to

the root, calculating the probabilities of successive subtrees on the way (as opposed to starting the calculation at the root of the tree). Approaching the likelihood calculation in this labour-saving manner is known as *Felsenstein's Pruning Algortihm* (1981).

It should also be noted that for substitution models with homogenous rates the transition probability matrix $\boldsymbol{P}(t)$ is the same for all sites and so need only be calculated once for each branch, leading to futher computational efficiency. Also, the probability of a particular *site pattern* (e.g. $x_0x_1x_2x_3$) at the tips of the tree is the same whenever it occurs so need only be calculated once. For this reason, sequence data is normally collapsed into counts of different site patterns before the likelihood calculation is performed. Let there be *s* different site patterns for a set of sequences of length *n*. If the likelihood and count of the *k*th site pattern is given by p_k and N_k respectively, where $\sum_{k=1}^{s} N_k = n$, the likelihood calculation for all sites becomes $\log(L) = \sum_{k=1}^{s} N_k \log(p_k)$.

1.3.4 The Pulley Principle

Time-reversible substitution models obey the detailed balanced condition $\pi_{y_0} p_{y_0 y_1}(t_0) = \pi_{y_1} p_{y_1 y_0}(t_0)$ and the Chapman-Kolmogorov equation tells us that $p_{ik}(t_1 + t_2) = \sum_j p_{ij}(t_1) p_{jk}(t_2)$. Thus we have

$$\sum_{y_0} \pi_{y_0} p_{y_0 y_1}(t_0) p_{y_0 y_2}(t_1) = \sum_{y_0} \pi_{y_1} p_{y_1 y_0}(t_0) p_{y_0 y_2}(t_1) = \pi_{y_1} \sum_{y_0} p_{y_1 y_0}(t_0) p_{y_0 y_2}(t_1) = \pi_{y_1} p_{y_1 y_2}(t_0 + t_1).$$

The original likelihood calculation for the rooted tree in figure 1.5 has now become:

$$f_k = f\left(x^k \mid \theta, t\right) = \sum_{y_1} \sum_{y_2} \left[\pi_{y_1} p_{y_1 y_2}(t_0 + t_1) p_{y_1 x_0}(t_2) p_{y_1 x_1}(t_3) p_{y_2 x_2}(t_4) p_{y_2 x_3}(t_5) \right]$$

As a result Felsenstein's pruning algorithm involves *c* less multiplications and *c* – 1 less additions:

$$f_k = f\left(x^k \mid \theta, t\right) = \sum_{y_1} \left[\pi_{y_1} p_{y_1 x_0}(t_2) p_{y_1 x_1}(t_3) \times \sum_{y_2} \left[p_{y_1 y_2}(t_0 + t_1) p_{y_2 x_2}(t_4) p_{y_2 x_3}(t_5) \right] \right]$$

Therefore, for time reversible substitution models one can move the root of the tree to any node on the tree (in the equation above it has been moved to the node containing state y_1) and the likelihood

calculation gives the same result. Felsenstein called this the *pulley principle* (1981). Figure 1.6 shows the unrooted tree that corresponds to the rooted tree shown in figure 1.5.

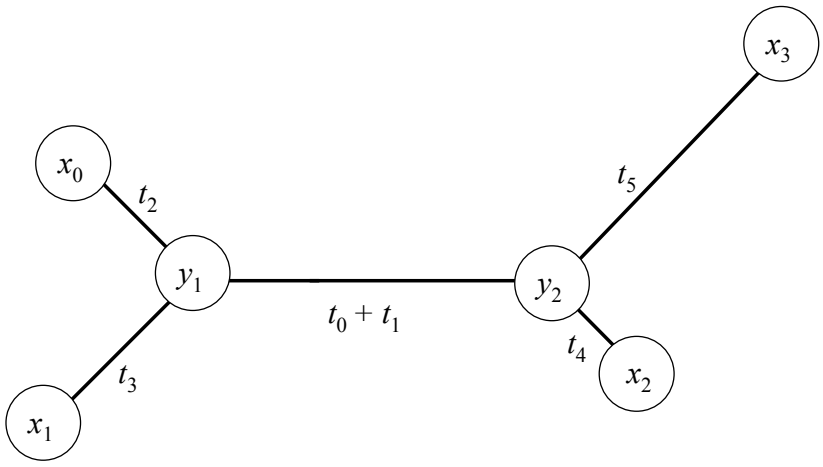


Figure 1.6: An example of one site in an unrooted tree of 4 taxa

This is the unrooted tree that corresponds to the rooted tree in figure 1.5.

1.3.5 Determining the Position of the Root of a Tree

A direct consequence of the pulley principle is that the reversible model in the toy example was over parameterised with six branch lengths – there is only enough data to estimate $t_0 + t_1$ but not to estimate t_0 and t_1 separately. Indeed, with reversible substitution models the position of the root cannot be determined without making further restrictive assumptions. For example, if one assumes that substitution rates have been constant over the whole tree, i.e. over time (known as the *molecular clock*), then the root can be determined since every tip sequence is now equidistant from the root. Conversely, it is theoretically possible to determine the root position using an irreversible model (e.g. Yap, Speed 2005) as the likelihood of the data given the model and tree will change depending on the lengths of the two branches from the root. However, in real data analyses the parameter values are not known and it may be the case that, if there are not enough data, uncertainties in parameter estimates may lead to greater changes in the log-likelihood than changing the position of the root. In this situation estimates of the root position by this method would be inaccurate at best.

1.3.6 Maximum Likelihood Estimation

The examples above have demonstrated how to compute the likelihood for a given set of parameter values. In practice we do not know the "real" values of the branch length parameters or the substitution model parameters. These values must be estimated from the data (for now knowledge of the correct tree is still assumed). This is done by searching for the parameter values that maximise the likelihood. In other words, if the probability of observing the data *x* given the parameter θ is $f(x \mid \theta)$ then the maximum likelihood estimate (MLE) of the parameter, $\hat{\theta}$, is given by:

$$\ell(\hat{\theta}) = \arg\max_{\theta} \log\left[f(x \mid \theta) \right]$$

Most phylogenetic analyses involve a large number of parameters for branch lengths and the substitution model and, in practice, non-linear multidimensional optimisation problems of this kind can almost never be solved analytically (but see Yang 2000 for one example). Therefore, numerical methods are used instead. Newton's method of minimising a multi-dimensional function begins by assuming that the function can be locally approximated by a quadratic function found by taking a Taylor expansion about the current estimate. The quadratic function involves the gradient vector of partial first order derivatives, and the Hessian matrix of partial second order derivatives. In most cases it is not possible to calculate the Hessian, and often it is not possible to calculate the gradient too. Thus *Quasi-Newton* methods that approximate the Hessian (and the gradient if needs be) are widely used instead. Explicitly, if x_k is the estimate of the minimum at iteration *k* of the algorithm, and Δx_k is a step away from that point, then the function *f* to be minimised can be approximated by the quadratic function:

$$f(x_k + \Delta x_k) \approx f(x_k) + \left[\nabla f(x_k)\right]^T \Delta x_k + \frac{1}{2}\left[\Delta x_k\right]^T B_k \Delta x_k$$

where $f(x_k)$ is the function, $\nabla f(x_k)$ is the gradient vector of f (partial first derivatives) and B_k is an approximation of the Hessian matrix (partial second order derivatives), evaluated at the point x_k. The Newton search direction is then found by solving $\Delta x_k = -B_k \left[\nabla f(x_k)\right]$ (Gill et al. 1981).

A line search is then performed along the Newton search direction until the minimum along that

direction has been found. At this new point, if convergence has not been achieved, the approximation to the Hessian B_k is updated to B_{k+1}, a new search direction is calculated and the algorithm begins another iteration. Many methods of updating B_k have been devised, but perhaps the most commonly used is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method which calculates a first order approximation to the Hessian by computing the change in the gradient along the search direction (see Fletcher 1987). If the form of the gradient cannot be calculated then it can also be approximated using difference formulae. Thus, *Quasi-Newton* methods allow researchers to tackle multi-dimensional and highly non-linear optimisation problems in molecular phylogenetics using the powerful maximum likelihood framework, even when the derivatives are too expensive or impossible to calculate.

1.3.7 Likelihood Ratio Tests and Hypothesis Testing

The maximised log-likelihood value of an evolutionary model can be used to compare it to other evolutionary models, with the highest likelihood value representing the model that fits the data best. Consider two models H_0 and H_1, with n_0 and n_1 parameters respectively, and maximised log-likelihood values of ℓ_0 and ℓ_1. If H_0 is a special case of H_1 (with $n_1 > n_0$), we would also expect the more general model to be better, i.e. $\ell_1 \geq \ell_0$ even when the nested model H_0 is true. Thus, a simple comparison of the log-likelihoods can be misleading. However, in this scenario a likelihood ratio test (LRT) for composite hypotheses (e.g. Kalbfleisch 1985) can be performed. Twice the difference in log-likelihood between the models follows a χ^2 distribution with $n_1 - n_0$ degrees of freedom, i.e. $2(\ell_1 - \ell_0) \sim \chi^2_{n_1 - n_0}$

If the test statistic $2(\ell_1 - \ell_0)$ is greater than the $\chi^2_{n_1 - n_0}$ critical value at the chosen level of significance then we reject H_0 and accept H_1. It is important to note that there are certain regularity conditions required for the χ^2 approximation to hold. Firstly, the two models need to be `nested', i.e. the more complex model can be transformed into the simpler model by imposing some linear set of constraints on its parameters. Even if the models are nested other issues can arise the cause the regularity conditions to be violated. For example, the complex mixture model used for detecting

positive selection by Zhang *et al.* (2005) contains a parameter which is not identifiable at the boundary of the parameter space in the simpler model. Thus, the χ^2 approximation cannot hold, and the correct asymptotic distribution is unknown. However, if those conditions are not satisfied then the correct null distribution can be still be derived from Monte Carlo simulations (Goldman 1993).

LRTs have been widely employed in molecular phylogenetics, for example, in tests to detect positive selection (see chapter 3), in procedures of substitution model selection (e.g. Posada, Crandall 1998), in tests of monophyly (Huelsenbeck et al. 1996b), in tests to detect conflicting phylogenetic signal (Huelsenbeck, Bull 1996), and in tests for the adequacy of gamma distributed rate heterogeneity (Goldman, Whelan 2000).

Alternatives to the LRT include the Akaike Information Criterion (AIC: Akaike 1974) and the Bayeisan Information Criterion (BIC: Schwarz 1978) given by $\mathrm{AIC} = -2\ell + 2p$ and $\mathrm{BIC} = -2\ell + p\log(n)$ where ℓ is the log-likelihood, p is the number of parameters and n is the sequence length. Both of these quantities are perhaps more flexible than LRTs because they can be used to test many nested or non-nested models simultaneously, and can also allow for model-averaged inference of parameters or phylogenies (Posada, Buckley 2004), but there is evidence that some LRT methods perform better overall (Posada, Crandall 2001).

1.3.8 Phylogeny Reconstruction

Different phylogenies represent distinct and non-nested statistical models. However, the optimised likelihood can still be used to assess different candidate phylogenies once they are found. This may be done by a simple comparison of the log-likelihoods or by using a likelihood based statistical test such as the KH-test (Hasegawa, Kishino 1989; Kishino, Hasegawa 1989) – although it has been shown that in most circumstances the SH-test (Shimodaira, Hasegawa 1999) is more appropriate (Goldman et al. 2000). The main difficulty in choosing an optimal tree is that finding a candidate set of approximately optimal trees to test is a far from trivial task (Neyman 1971). The number of bifurcating rooted trees for n species (and the number of bifurcating unrooted trees for $n + 1$ species) is given by $T_n = \frac{(2n-3)!}{2^{n-2}(n-2)!}$. As an example, for as few as 10 species there are 34.5 million possible rooted

trees. For 20 species there are $\sim 8\times10^{21}$ different rooted trees. If one considers that this is more than 16,000 times the number of seconds since the beginning of the universe (14 billion years $\approx 5\times10^{17}$ seconds) then one can understand why Felsenstein (1978) stated that one of the two main uses of these type of calculations in the future would be to "frighten taxonomists"! Unsurprisingly, finding the optimal phylogeny has been classified as a member of a particularly difficult class of optimisation problems known as NP-complete (Day 1987).

Tree-space is highly non-linear and novel search algorithms have to be implemented in order to traverse the tree-space and find candidate topologies. Three such procedures are known as *nearest-neighbour interchange* (NNI), *subtree pruning and regrafting* (SPR), and *tree bisection and reconnection* (TBR). Figure 1.7 shows an example of each type of move. In a bifurcating tree every internal branch separates four subtrees – two on one side of the branch and two on the other side of the branch. An NNI move involves swapping one of the subtrees on one side of the branch with one of the subtrees from the other side of the branch. An SPR move involves pruning a subtree and reattaching it to another branch on the tree. A TBR move involves removing a branch from the tree to leave two subtrees, then rejoining the two subtrees with a newly created branch. The SPR operation may be of particular interest as it could represent biological processes such as recombination or horizontal gene transfer (Allen, Steel 2001) but it is not the most complete move. NNI rearrangements are a subset of SPR rearrangements which are themselves a subset of TBR rearrangements Therefore, TBR can be considered the most thorough algorithm for traversing tree-space, whilst also being the most computationally expensive (Maddison 1991).

A typical likelihood based attempt at phylogeny reconstruction would proceed by choosing an initial starting tree (possibly a random tree or the maximum parsimony tree) and optimising the model parameters (such as branch lengths and parameters from the substitution model) to obtain an optimised log-likelihood value. Then a new tree is proposed (through e.g. TBR) and the parameters are re-estimated to obtain a new optimised log-likelihood. The new tree is then accepted or rejected based on the optimality criterion of the algorithm at work, and the process continues. A "greedy" hill-climbing algorithm will always accept/reject trees that result in better/worse log-likelihood values,

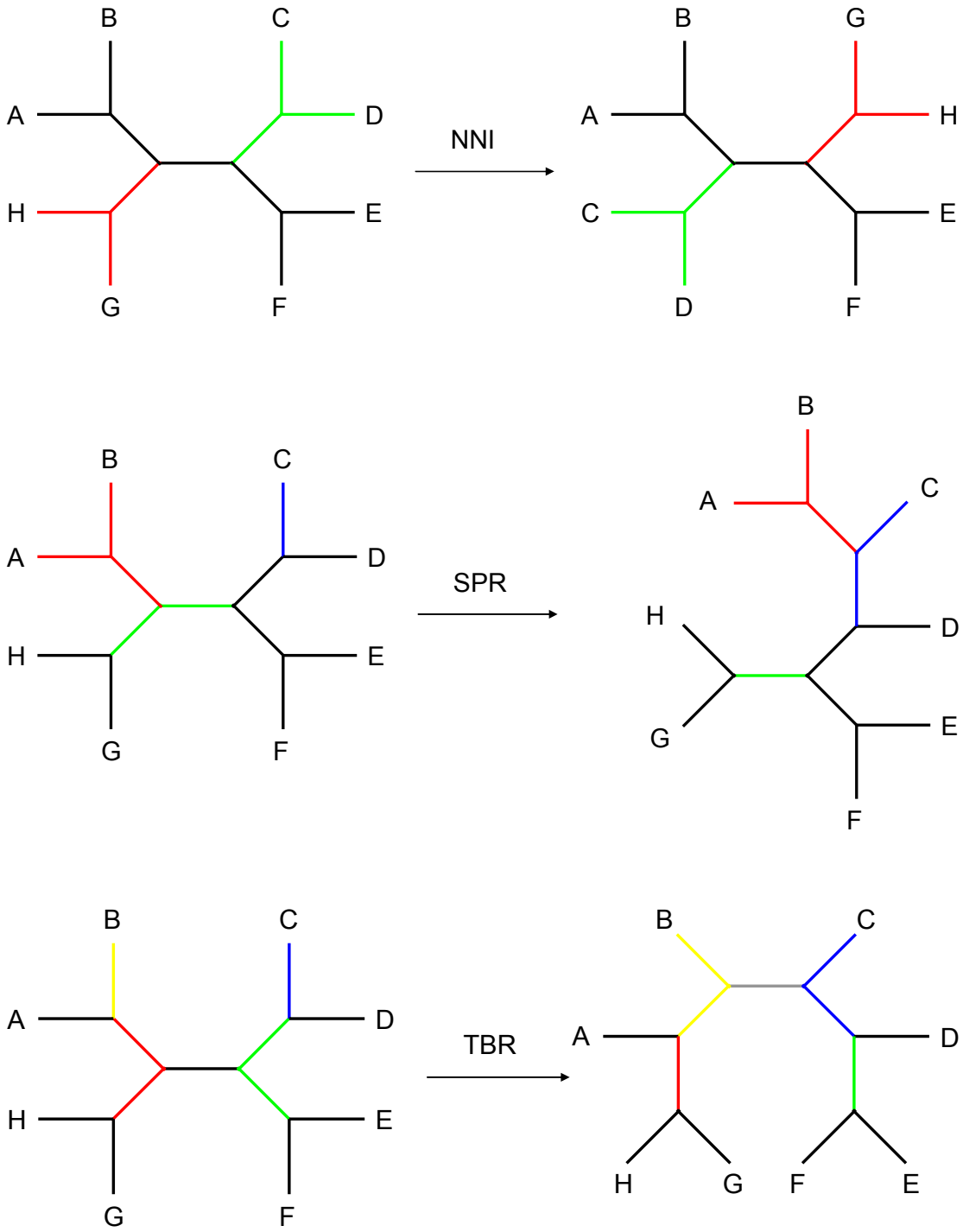


Figure 1.7: Examples of moves used to traverse "tree-space"

Shown here are the *nearest-neighbour interchange* (NNI), *subtree pruning and regrafting* (SPR), and *tree bisection and reconnection* (TBR) moves. A to F can represent different taxa or subtrees.

whereas other algorithms such as *genetic algorithms* (e.g. Lewis 1998) or *simulated annealing* (e.g. Stamatakis 2005) may allow "downhill" moves in an attempt to avoid getting stuck in local peaks in the likelihood surface.

An alternative (and far faster) approach to phylogeny reconstruction is that adopted by the so-called *distance-based* methods. These methods break the problem in to two steps: calculation of the evolutionary distances between each pair of sequences, followed by reconstruction of the tree from the matrix of pairwise distances using a clustering algorithm. In a likelihood framework the first step is relatively simple; each pairwise distance is simply the optimised branch length obtained from maximising the likelihood, given the substitution model and the sequence data, on a two taxon tree. An example of a clustering algorithm required for step two is the well known *neighbour-joining* (NJ) method (Saitou, Nei 1987) based on the minimum-evolution criterion. The method begins by assuming a star-tree (a tree with only one internal node at the root joining all branches). Then, an extra internal node is created, chosen such that it joins together the two leaf nodes that will yield the greatest reduction in tree length. The dimension of the distance matrix is thus reduced by one, and the algorithm proceeds to choose another pair of nodes to join, and so on. NJ is widely used because it is elegant and fast, and when given the true distances the method is guaranteed to recover the true tree topology, and will still do so when there are small errors in the distance estimates (Atteson 1997). One drawback of any distance-matrix based method is that large distances can be poorly estimated. Therefore, several improvements to NJ have been suggested such as weighting the distances to account for these errors (WEIGHBOR: Bruno et al. 2000), or accounting for the variances in large distance estimates (BIONJ: Gascuel 1997). Phylogeny reconstruction is discussed in greater detail in chapter 4.

1.3.9 Bayesian Methods

Any review of the use of computational statistics in molecular phylogenetics would be remiss not to mention the Bayesian paradigm. Thus I shall introduce it here briefly for completeness, but will not go into detail because I do not implement or investigate any such methods during later chapters of this

thesis. The interested reader may wish to consult a more thorough review of Bayesian phylogenetics (e.g. Huelsenbeck et al. 2001; Holder, Lewis 2003; Yang 2005; chapter 5 of Yang 2006).

The likelihood method, as described above, is a cornerstone of *classical* or *frequentist* statistical theory. In this paradigm the probability of an event is defined as the expected frequency with which that event will occur in repeated random draws from some (real or imaginary) population, and parameter values are considered to be unknown constants that need to be determined. The likelihood function is deemed to contain all the information in the data about the parameters and is interpreted as the probability of observing the data given that a particular model is correct. Bayesian inference, once known as *inverse probability*, turns this principle on its head by asking: what is the probability that this model is correct given that I have observed the data? When compared to likelihood the central difference of Bayesian methods is the concept that parameters have probability distributions and that probability now represents the uncertainty about the parameters. Before the data is analysed the parameters are assumed to follow a *prior* distribution, then Bayes theorem is used to calculate the *posterior* distribution of the parameters, that is, the conditional distribution of the parameters given the data. If θ are the parameters and *x* are the data then Bayes Theorem is given by:

$$f(\theta \mid x) = \frac{f(\theta) f(x \mid \theta)}{f(x)} = \frac{f(\theta) f(x \mid \theta)}{\int f(\theta) f(x \mid \theta)\, d\theta}$$

where $f(\theta \mid x)$ is the posterior, $f(\theta)$ is the prior, $f(x \mid \theta)$ is the likelihood and $f(x)$ is the marginal probability of the data, averaged over all parameter values (a normalising constant such that the posterior integrates to one). All inferences about parameters are based on the posterior. For example, the mean, median or mode (the *maximum a posteriori probability* or *MAP* estimate) of the posterior distribution can be used as a point estimate of a parameter value instead of the maximum likelihood estimate. This may be useful, for example, if one is interested in investigating the mechanisms behind the evolutionary process and therefore would like to estimate the values of particular parameters in the substitution model. However, sometimes the ultimate goal may be something else, such as the estimation of an underlying phylogeny. In that scenario, the substitution model parameters are simply an annoyance that have to be accommodated. The Bayesian method

provides a natural way of handling such nuisance parameters. Explicitly, the joint conditional distribution of two sets of parameters, θ and λ, is given by:

$$f(\theta,\lambda \mid x) = \frac{f(\theta,\lambda) f(x \mid \theta,\lambda)}{f(x)} = \frac{f(\theta,\lambda) f(x \mid \theta,\lambda)}{\int\int f(\theta,\lambda) f(x \mid \theta,\lambda)\, d\theta d\lambda}$$

If the θ are the parameters of interest, then the nuisance parameters λ can be integrated out and the marginal posterior density of θ can be obtained as: $f(\theta \mid x) = \int f(\theta,\lambda \mid x)\, d\lambda$

In statistics in general there is much controversy about how one should arrive at a choice of prior distribution (e.g. basing it on subjective beliefs versus past observations) or, indeed, the need to specify the prior in the first place. Aside from these conceptual and philosophical issues a major reason that Bayesian methods were not widely used until recently is that the normalising constant $f(x)$ is rarely analytically tractable, excepting simple toy examples, and numerical methods must normally be used to compute the (normally high-dimensional) integral. However, Bayesian computation has become widespread in recent years after efficient Markov-Chain Monte Carlo algorithms (MCMC: Metropolis et al. 1953; Hastings 1970) were first popularised when Gelfand and Smith (1990) demonstrated such approaches were viable for analyses of non-trivial problems[1].

When the target distribution has multiple peaks, as is the case with the highly non-linear tree-space, MCMC algorithms can get stuck in *local* peaks and fail to find the true mode. One of the most successful strategies for dealing with this problem is the so-called Metropolis-Coupled MCMC or MC^3 method (Geyer 1991) where multiple chains are run in parallel. The first chain is the *cold* chain with the desired target density, and the remaining chains are *incrementally heated* by raising the density to increasingly smaller powers T (with $T < 1$). This has the effect of flattening out the peaks of the distribution, allowing the hotter chains to traverse between peaks in an easier fashion than the cold chain can on the original distribution. During the MCMC run states are sometimes swapped

[1] It should be noted that even before the Metropolis algorithm was conceived there existed a means of sampling from an arbitrary target distribution known as the "*rejection method" (von Neumann, J. 1951. Various techniques used in connection with random digits. National Bureau of Standards Applied Mathematics Series 12:36-38.)*. However, in practice, the method was limited to low dimensional parameter spaces.

between chains, allowing the cold chain to occasionally jump over valleys, leading to better mixing. At the end of the run, only the output from the cold chain is used to determine the posterior. Further details of this, and other, MCMC methods will not be covered here. The keen reader may wish to refer to one of the many excellent textbooks on the subject (e.g. Gilks et al. 1996).

Bayesian methods were first introduced to phylogenetics more than a decade ago (Rannala, Yang 1996; Yang, Rannala 1997). It did not take long for efficient MCMC algorithms for phylogeny reconstruction to be devised (Larget, Simon 1999), implemented (e.g. MrBayes: Huelsenbeck, Ronquist 2001; Ronquist, Huelsenbeck 2003) and used to infer species relationships such as the origin of land plants (Karol et al. 2001), or the radiation of different mammalian orders (Murphy et al. 2001). Since then the method has become very popular and widely used, particularly in the processes of phylogeny reconstruction (e.g. Huelsenbeck et al. 2001), estimation of species divergence times under relaxed molecular clocks (e.g. Thorne et al. 1998; Kishino et al. 2001; Yang, Rannala 2006), or inference of ancestral population sizes (e.g. Rannala, Yang 2003). Proponents of the Bayesian approach commonly point to the fact that it provides an attractive alternative to maximum likelihood since posterior probabilities are easy to interpret whilst also circumventing some of the controversies surrounding interpretation of the nonparametric bootstrap (Felsenstein 1985) – the most common method of assessing phylogenetic uncertainty.

However, critics have noted that the method often produces surprisingly high posterior probabilities for trees or clades (e.g. Suzuki et al. 2002) and this was even noted in the very first Bayesian phylogenetic study (Rannala, Yang 1996). Subsequent work has showed that the choice of prior distribution, evolutionary model and type of data can all strongly affect the posterior probability of trees (Yang, Rannala 2005; Yang 2007a; Yang 2008, also see Appendix A) to the point where the method may infer different trees, sometimes with strong support, even when they are contradictory**.** One proposal that has been shown to reduce such conflicts is to make use of a data size-dependent prior (Yang 2008) where the prior mean approaches zero faster than $1/\sqrt{n}$ but slower than $1/n$ where n is the sequence length. This type of prior was found to give weaker support for unstable relationships and could therefore be useful in reducing apparent conflicts in the results of Bayesian analyses, or making the method less sensitive to model violations.

1.4 Multiple Sequence Alignment

Most phylogenetic methods used today, such as those mentioned earlier in this chapter, require as input a hypothesis about homology between sequence sites in the form of an alignment of sequences of equal length. However, one unavoidable consequence of the evolutionary processes of insertion and deletion is that evolutionarily related biological sequences end up having different lengths. Multiple sequence alignment (MSA) is the name given to any procedure that converts such a group of sequences into a set of sequences of equal length, by constructing a hypothesis about which characters from each sequence are homologous. Explicitly, a MSA method infers where gaps should be placed in each sequence such that homologous characters are subsequently located at the same position in each sequence. Therefore, constructing an MSA is a crucial first step in most phylogenetic studies.

However, MSAs themselves can often be the ultimate goal of researchers since, for example, they can allow identification of conserved protein regions that may be of functional or structural importance (e.g. Thomas et al. 2003; Minovitsky et al. 2007). Given their widespread use and applicability, the fundamental procedure of constructing an MSA can be considered one of the foundations of modern computational biology as a whole. Perhaps the clearest confirmation of this self evident fact is that, at the time of writing, one widely used MSA program (ClustalW: Thompson et al. 1994) has been cited over 30,000 times.

The performance of many MSA methods and their effect on downstream phylogenetic analyses form part of the work presented in later chapters of this thesis. Therefore, the following pages will be devoted to describing approaches to MSA in detail.

1.4.1 Pairwise Sequence Alignment and Scoring Schemes

The simplest alignment problem which can be considered is that of determining the *pairwise* homology between the sites of just two sequences. In order to choose between competing alignments it is necessary to define an objective scoring function that quantifies how ‘good’ or ‘bad’ a certain alignment is. Then, depending on how the function is defined, the goal of the alignment method is to maximise or minimise the resulting score. These objective functions can take many forms, but perhaps the simplest type would involve three types of score: a (positive) score for the benefit of

aligning a pair of sites that contain the same character, a (negative) score for the cost of aligning a pair of sites that contain different characters, and a (negative) score for the cost of aligning a character from one sequence opposite a gap. For this function the goal of the alignment method would be to maximise the resulting score. Another simple scoring scheme could be to use the *edit distance,* which is simply defined as the number of changes that are necessary to change one sequence into another. Therefore, in this case, an alignment method would aim to minimise the edit distance as it searched for the 'best' alignment between two sequences.

But even in a pairwise alignment scenario, using a simplistic scoring scheme such as those described above, an attempt at finding the 'best' alignment (as determined by the scoring function) through some sort of brute force search is misguided. If one sequence has length *n* and the other has length *m* then the number of distinct alignments of the two sequences has been calculated by Torres *et al*. (2003) as:

$$f(n,m) = \sum_{k=0}^{\min\{n,m\}} 2^k \binom{m}{k}\binom{n}{k}$$

As an example, this implies that there are nearly 40 million possible alignments between one sequence of length 16, and one sequence of length 8. Biological sequences of interest are normally far longer than this. Therefore, a brute force search very quickly becomes completely impractical, even with two sequences. Luckily, given a particular objective scoring function, a technique known as *dynamic programming* is guaranteed to find the optimal alignment of two sequences and will do it in far less time than a brute force search.

1.4.2 Dynamic Programming

In essence, this technique involves approaching complex problems by breaking them down into smaller overlapping subproblems and solving them recursively whilst checking at each stage if the solution to a particular subproblem has already been found. In the context of pairwise sequence alignment this amounts to calculating the alignment score by breaking the problem down into the combination of choosing how to align the single sites at the end of the two sequences with their optimally aligned subsequences (Eddy 2004). The algorithm is best described via a worked example.

Consider two sequences x and y that we wish to align, which are of lengths m and n respectively. Let x_i denote the i-th position of sequence x and y_j denote the j-th position of sequence y. Also, let us define $s(a,b)$ to be a score for aligning two characters a and b, and let λ be some gap-penalty for aligning a character opposite a gap. Finally, let $S(i,j)$ be the score for the optimal alignment of characters 1 to i in sequence x with characters 1 to j in sequence y.

Clearly there are only three ways that the last character of each of the two sequences, x_m and y_n, can be aligned. Either x_m is opposite a gap with y_n further back in the alignment (score of λ), or y_n is opposite a gap with x_m further back in the alignment (score of λ), or x_m and y_n are aligned together (with a score of $s(x_m, y_n)$). Thus, the optimal total alignment will have the largest score out of these three possibilities: $S(m-1,n)+\lambda$, $S(m,n-1)+\lambda$, and $S(m-1,n-1)+s(x_m,y_n)$. In order to find out which of these alignments has the highest score we need to calculate the three scores for the three slightly smaller subproblems, namely $S(m-1,n)$, $S(m,n-1)$, and $S(m-1,n-1)$.

Crucially, to calculate these three scores the exact same logic can be applied again. For example, to calculate $S(m-1,n)$ we first consider the three possible ways to align x_{m-1} with y_n, and then must calculate the three scores $S(m-2,n)$, $S(m-1,n-1)$ and $S(m-2,n-1)$, and so on. We can continue in this manner, applying the logic recursively, until we reach very small subproblems with obvious solutions, e.g. the score for aligning nothing with nothing $S(0,0)=0$. Therefore we can write a generalised recursive definition for the partial optimal alignment scores $S(i,j)$:

$$S(i,j) = \max \begin{cases} S(i-1,j-1)+s(x_i,y_j) \\ S(i-1,j)+\lambda \\ S(i,j-1)+\lambda \end{cases}$$

Although relatively simple to explain, this top-down approach leads to many redundant determinations of the optimal subsequence alignments. Infact, the number of redundant calculations grows exponentially as we move deeper into the recursion. Obviously it would make sense to keep track of the different subproblems that had already been solved so there was no wastage. This is the central advantage that dynamic programming algorithms have over naive simple recursive algorithms: the optimal scores $S(i,j)$ of the different subproblems are stored as the algorithm proceeds. For a

pairwise alignment algorithm optimal scores are tabulated in a two-dimensional matrix (see figure 1.8). To fill in the matrix efficiently most algorithms take a "bottom-up" approach and begin by calculating the simplest subproblems first, i.e. the top left cell $S(0,0) = 0$ (nothing aligned with nothing) first, followed by the left-most column $S(i,0) = i\,\lambda$ (i characters from x aligned against i gaps scores i times λ, $1 \le i \le m$) and the top row $S(0,j) = j\,\lambda$ (j characters from y aligned against j

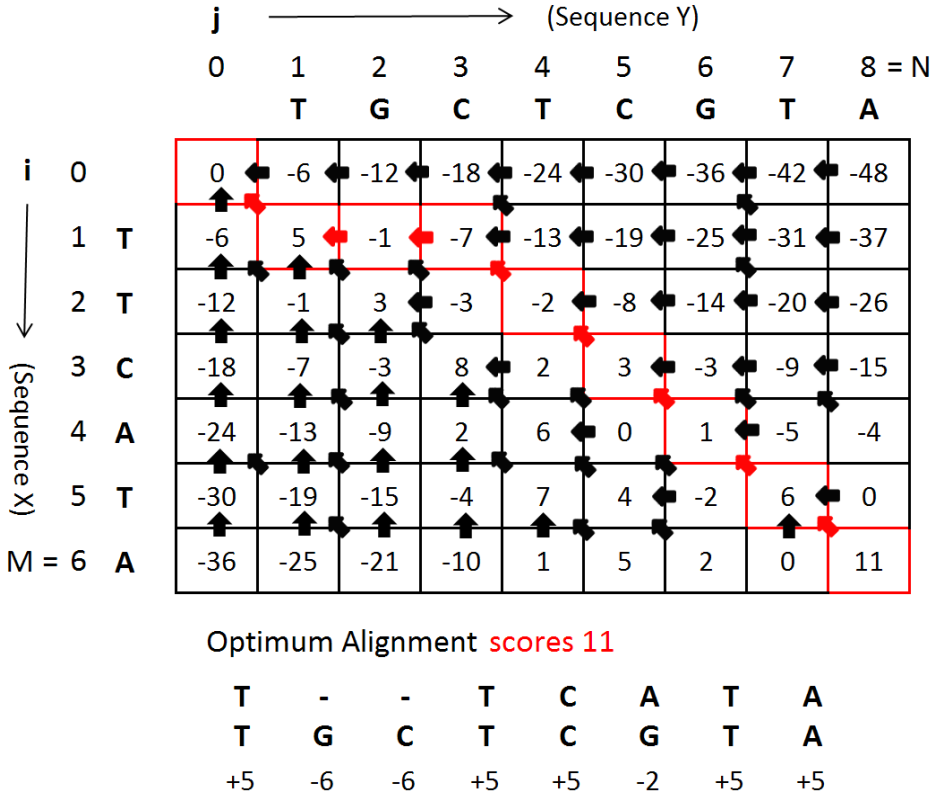


Figure 1.8: Dynamic Programming Matrix

An example of a filled dynamic programming matrix for two DNA sequences x = TTCATA and y = TGCTCGTA, under a scoring system with +5 for a match, -2 for a mismatch and -6 for each insertion or deletion. The cells for the optimum path are shown in red. The arrows are "backtracing pointers" that indicate which of the three cases were optimal for reaching that cell. It should be noted that some cells can be reached by more than one optimal path of equal score. Dynamic programming algorithms tend to resolve such conflicts by arbitrarily choosing one path. However, in this example the optimal path is unique. Example reproduced from Eddy (2004).

gaps scores j times λ, $1 \le j \le n$). Then, the remaining cells can be filled in using the recursive definition of $S(i, j)$ from page 55, and the scores from the three neighbouring cells to the upper left, i.e cells $(i-1, j)$, $(i-1, j-1)$ and $(i, j-1)$. Once the matrix is filled in we know the score of the optimal alignment $S(m, n)$ as it is simply the score of the bottom-right cell (11 in our example), but we don't know the optimal alignment itself. To find this we *backtrace* through the matrix in the top-down manner originally described. First we begin in cell m, n and determine which of the three previous cases, $(m-1, n)$, $(m-1, n-1)$ or $(m, n-1)$, was used to get to that cell (i.e. by repeating the same 3 calculations), record that choice as part of the optimal alignment, then repeat the process over and over until we reach cell (0,0) and the optimal alignment is fully constructed.

Although there had been other attempts at automating sequence alignment (e.g. Fitch 1966) it was this elegant dynamic programming algorithm solution proposed by Needleman and Wunsch (1970) that is considered antecedent to all others. The efficiency of this general approach has been refined and improved many times since. For example, the algorithm of Gotoh (1982) involves far fewer computational steps, and Myers and Miller (1988) extended Gotoh's algorithm such that the memory requirements were lower. However, the general alignment principle implemented in each algorithm essentially remained the same and is still used in many programs to this day. Despite the lack of wastage in the algorithm it is still very computationally demanding since filling in the programming matrix takes time proportional to mn so, for example, the alignment of two sequences of length 200 will take four times as long as for two sequences of length 100. For this reason, much research is devoted to finding good approximations to the dynamic programming algorithm and this shall be discussed in later sections.

1.4.3 Global and Local Alignment Strategies

The alignment strategy described above is known as a *global* alignment algorithm as it assumes that the two sequences to be aligned are sequentially homologous and tries to simultaneously find the optimal alignment for all the sites from both sequences. This assumption may not be valid for sequences that have experienced large scale rearrangements such as inversions, where part of one

sequence is reversed compared to the second sequence, or transpositions, where part of one sequence has changed location relative to other sites in the sequence (e.g. see Kent et al. 2003). In these cases only certain subsections of the two sequences will be truly homologous and a global approach could lead to a poor alignment. An alternative approach is *local* alignment whereby the alignment method, rather than considering the two sequences as a whole, only attempts to align different subsections of each sequence at any one time with no regard for the overall global structure of the alignment. This approach is useful for very dissimilar sequences as regions of high similarity can be aligned whilst leaving highly divergent regions unaligned.

The archetypal local alignment algorithm is that of Smith and Waterman (1981). It is actually a variation of the Needleman-Wunsch algorithm described previously – the main difference being that the negatively scoring gap and mismatch elements of the scoring matrix are set equal to zero, thus rendering the remaining positively scoring local alignments 'visible' to the alignment algorithm. Since the Smith-Waterman algorithm is a derivative of the Needleman-Wunsch algorithm, the approach also benefits from the desirable properties of dynamic programming, namely that the optimal local alignment (with respect to the scoring function) is guaranteed to be found.

As an aside, it is interesting to note that the biologically motivated advances in local alignment algorithms have cross-pollinated other areas of research. For example, sequence alignment methods are now sometimes used in natural language processing during the first step of comparative reconstruction of ancient languages (Covington 1996; Covington 1998; Barzilay, Lee 2002).

1.4.4 The Objective Scoring Function

The scoring functions at the centre of most modern alignment programs tend to score matches and mismatches between characters using complicated sums based on the substitution matrices of probabilistic Markov models such as those described in section 1.2 (e.g. HKY or K80 are widely used for nucleotide alignments, and BLOSUM62 and WAG are commonly used for amino-acid alignments). The scoring of gaps is normally carried out in a far less complicated framework. The simplest way to score gaps is to simply assign a constant value ($-k$) for a gap of any length. Since it is known that insertions and deletions tend to be small in size it makes more sense to penalise large gaps

to a greater degree than small gaps in an alignment. The simplest scoring system for penalising gaps that respects this intuition is a simple linear function that is proportional to the number of gaps (n) being inserted, e.g. $f(n) = -kn$ for some constant k.

A common extension of this approach is the so-called *convex* or *affine* gap penalty where different penalties are used for opening gaps in the alignment as opposed to extending them (the former normally being more costly than the latter), e.g. $f(n) = -\left[k_1 + k_2 n\right]$, $k_1 > k_2$. Gotoh's algorithm makes use of affine gap costs and it is the efficiency of this algorithm that has perpetuated the use of this scoring scheme (as opposed to the scheme's biological realism). The observation that indel sizes can be well described by a power law (e.g. Fan et al. 2008) led many researchers to propose that logarithmic gap penalties

$$f(n) = -\left[k_1 + k_3 \ln(n)\right]$$

were more realistic than affine gap penalties (Gonnet et al. 1992; Gu, Li 1995). More recently this assumption has been questioned and it may be possible that log-affine gap penalties (Cartwright 2006) are more appropriate, i.e.

$$f(n) = -\left[k_1 + k_2 n + k_3 \ln(n)\right].$$

It is important to underline once again that the elegant Needleman-Wunsch and Smith-Waterman algorithms (and their variations) are by no means guaranteed to find the alignment that represents the true homology between characters in homologous sequences. In contrast, they are only guaranteed to find the optimal alignment that is implied by the objective function they are using to score the potential alignments being examined. Of course, different functions can imply different 'optimal' alignments, so the scoring function used is perhaps the single most important factor in obtaining an alignment of good quality. Therefore, it is no surprise that the effect on alignment quality that the choice of gap penalties (Altschul 1998; Liu et al. 2009a), gap scoring model (Cartwright 2006) or substitution matrix (Edgar 2009) have is an active avenue of research.

1.4.5 Database Searching

Bioinformaticians often desire to search DNA and protein databases for sequences with regions of similarity to some target sequence of interest. For example, if a new gene sequence is discovered in the rat genome, then a researcher may wish to search a database to see if there are sequences of the human genome that are similar to the rat gene sequence. It was realised at an early stage that to be of any use this process would need to be automated and an algorithm to facilitate this search would need to be devised (e.g. Korn et al. 1977; Wilbur, Lipman 1983). The need for good search algorithms is perhaps even more of an issue in the modern day with the preponderance of sequence data now available since querying a database in this manner could potentially involve performing hundreds of thousands of pairwise alignments. For example, it is predicted that the EMBL Nucleotide Sequence Database (Kulikova et al. 2004) will contain over a trillion nucleotides in the near future (Goldman, Yang 2008). Thus, the need to search a sequence database has yielded many improvements in local alignment algorithms over the years.

One such algorithm is FASTA (Lipman, Pearson 1985; Pearson 1988). This algorithm was so successful that compatibility with the FASTA file format is now offered by almost every other database search or sequence alignment tool. Another algorithm, perhaps the most widely known, is BLAST – the Basic Local Alignment Search Tool (Altschul et al. 1990; Altschul et al. 1997), which has become so ubiquitous to database searching that "BLAST" is commonly used as a verb in the same way that GOOGLE is when one wants to refer to searching the web. FASTA and BLAST both make use of "word" (sometimes called "*k*-tuple" or "*k*-mer") heuristic search methods that begin by identifying a series of nonoverlapping subsequences (words) of length *k* in the test sequence, and then try to match these words to candidate sequences from the database. If multiple distinct words are found at the same relative position in a database sequence as in the test sequence (allowing for some fixed offset) then further alignment criteria are evaluated. Thus, both methods avoid making many unnecessary sequence comparisons and it is this emphasis of speed over sensitivity that makes them practical to use on large databases – BLAST is over 50 times faster than the exhaustive Smith-Waterman algorithm that it approximates.

1.4.6 Progressive Alignment

Dynamic programming algorithms, whilst time-consuming, are at least tractable for pairwise alignment. In theory the approach can be extended to alignment of multiple sequences but quickly becomes unusable. This is because, when aligning *n* sequences, it requires the use of an *n*-dimensional matrix to keep track of the optimal scores. In addition, the growth of the number of possible alignments of multiple sequences as the number of taxa increases is truly staggering (Slowinski 1998), and is at least as dramatic as the incredible growth of the number of possible phylogenetic trees as the number of taxa increases that was mentioned previously (Felsenstein 1978). In fact, like tree reconstruction, the problem of MSA has been shown to be NP-complete (Wang, Jiang 1994). For all these reasons, MSA algorithms must use heuristic search strategies.

The heuristic that is most prevalent among alignment programs is the *progressive* alignment strategy first described by Feng and Doolittle (1987). A variation on this algorithm is used in many popular alignment programs such as PRANK (Löytynoja, Goldman 2005), MAFFT (Katoh, Toh 2008b), MUSCLE (Edgar 2004), ProbCons (Do et al. 2005), T-Coffee (Notredame et al. 2000), DIALIGN-TX (Subramanian et al. 2008) and ClustalW (Larkin et al. 2007). The method typically begins by calculating the $n(n-1)/2$ possible pairwise alignments of the *n* sequences to be aligned using, for example, the Needleman-Wunsch algorithm. These pairwise alignments are then used to calculate the distances between each pair of sequences and a clustering algorithm such as NJ (Neighbour Joining) or UPGMA (Unweighted Pair Group Method with Arithmetic Mean) is used to construct a phylogenetic tree. Using this tree as a guide the two most similar sequences are aligned with a pairwise algorithm and then sequences are progressively added to the alignment, one at a time, based on their position in the tree. Some methods (e.g. Muscle and MAFFT) seek further speed increases by skipping the initial pairwise alignment stage and approximating the distances between sequences using the number of shared *k*-tuples instead.

One issue that has been noticed concerning progressive alignments is that the order in which sequences are added can bias downstream phylogeny reconstruction (Lake 1991; Thorne, Kishino 1992). Avoiding this artefact of the alignment process is one potential advantage of a one-step process that attempts to align all sequences simultaneously rather than progressively (Lipman et al. 1989;

Stoye et al. 1997; Stoye 1998), although standalone one-step alignment programs of this kind are not widely used in phylogenetic studies (but see comments on simultaneous alignment and phylogeny reconstruction below, in section 1.4.8).

1.4.7 Improving and Refining a Progressive Alignment

Another problem with progressive alignment is that it is what is known as a *greedy* algorithm. A greedy algorithm makes the locally optimal choice at each stage, and any mistakes made in a particular stage are fixed during later stages and so propagate through to the final output. One technique commonly employed to try and correct or minimise such mistakes is known as *iterative refinement* and has been shown to be a successful strategy (Wallace et al. 2004). This process involves taking a progressive alignment and trying to improve it in some way. For example, an alignment method may repeatedly divide the sequences into two randomly chosen groups and then realign the subalignments (Berger, Munson 1991). A better performing approach offered by some programs (e.g. MAFFT) is to pursue a weighted sum-of-pairs score where the weights are introduced to correct for uneven representations of particular subgroups of sequences (Gotoh 1996). Many common alignment methods (e.g. MAFFT, Muscle, PRANK, DIALIGN) implement a different type of iterative refinement where the initial progressive alignment is used to construct a second guide tree which is in turn used to guide a new progressive alignment. This procedure may be repeated more than once in some of the programs.

Iterative refinement attempts to fix the mistakes made by progressive alignment strategies. An alternative approach is to try to prevent those mistakes occurring in the first place. This is the ultimate goal of *consistency based* alignment methods and was pioneered by the novel program T-Coffee, and implemented later in other programs such as ProbCons. The standard T-Coffee alignment process begins by calculating the global pairwise alignments between each pair of sequences using ClustalW. These are then used, along with the 10 ‘best’ local alignments associated with each sequence pair, calculated with Lalign (Huang, Miller 1991), to create a library of pairs of characters that could be aligned, each with an associated weight. This initial library is then searched for pairwise similarities that support each other, i.e. similarities involving more than two sequences. These overlapping

pairwise similarities are then used to construct a second library, which is then used to guide the normal progressive alignment strategy in place of the traditional guide tree. ProbCons is similar to T-Coffee, however, it uses a probabilistic consistency scoring function and then employs a Hidden Markov Model (see below) to construct the alignment by maximising the expected accuracy. DIALIGN is another program that can make use of both local and global alignment features, allowing areas of locally aligned subsequences to be interrupted by other globally aligned or unaligned subsequences.

Several other interesting T-Coffee variants have been also been designed. For example, M-Coffee (Wallace et al. 2006) uses other packages (such as Clustal, MAFFT, ProbCons, Muscle, etc) to generate alignments and uses the same consistency based methods to combine them into one unique final alignment. Other variants can additionally constrain the alignment process by using non-sequence based sources of information such as protein structure (3D-Coffee/Expresso: Poirot et al. 2004; Armougom et al. 2006) or RNA secondary structures (R-Coffee: Wilm et al. 2008). The ability to incorporate structural information in this way has also been added to MAFFT (Katoh, Toh 2008a) as well.

Perhaps the most significant variation on the decades old progressive alignment strategy are the improvements found in the algorithm at the core of PRANK that allow the program to correctly handle insertions and deletions. The problems other programs experience stem from the fact that in a pairwise alignment the processes of insertion and deletion are indistinguishable and, as mentioned above, the order of the backtracking progressive alignment strategy used by most alignment methods is based on initial pairwise alignments (or approximations to them). This has the effect that a gap for a deletion, and its associated penalty, are created only once whereas a gap for an insertion has to be opened many times. Essentially this leads to a full penalty being assigned to each of these gap opening events, resulting in excessive penalisation of single insertion events (Löytynoja, Goldman 2008). Other alignment methods (such as ClustalW and Muscle) implicitly tackle this problem by lowering the penalty for opening gaps at alignment positions that already contain gaps. However, it has been shown that this strategy of site-specific gap penalties, far from solving the problem, actually encourages the incorrect matching of independent insertions. This can result in systematically biased

alignments that suggest implausible insertion-deletion histories containing an excess of deletions and substitutions and too few insertions (Löytynoja, Goldman 2008). In stark contrast to this, PRANK uses an algorithm that identifies the gaps in an alignment as coming from either an insertion or a deletion. Then, in subsequent rounds of the progressive alignment, the algorithm chooses to allow gaps to be added opposite pre-existing insertions without penalty in each new sequence. In addition, the program can use closely related sequences to infer some sites as "permanent" insertions that cannot be matched in other sequences. The means that independent insertion events that occurred at the same position in the alignment will be correctly kept apart rather than attracted to each other as in normal progressive alignment strategies. The alignments that PRANK produces tend to be more accurate and less compressed as a result (see chapter 3).

1.4.8 Statistical Approaches to Multiple Sequence Alignment

An alternative to the character-based alignment methods mentioned above is to use a statistical estimation procedure based on the likelihood method. Although statistical approaches were first suggested a long time ago (Bishop, Thompson 1986) they were slow to take off, mostly due to the heavy computational burden that these methods impose.

The first tractable statistical models of the insertion-deletion process suffered from unrealistic assumptions such as assuming that indel events involve only one character (TKF91: Thorne et al. 1991) or that sequences are made from non-overlapping indivisible fragments (TKF92: Thorne et al. 1992). The structure of both these models induces a *pair hidden Markov model* or *pair HMM* (Durbin et al. 1998) that can assign likelihoods to all of the possible combinations of matches, mismatches and gaps to determine the most likely MSA or set of MSAs. Furthermore, by optimising the parameters of the model for every sequence pair in a given dataset it is possible to directly construct a neighbour joining tree without knowledge of the underlying multiple alignments (Thorne et al. 1992). Most subsequent attempts at statistical alignment have used these two models as a starting point.

Shortly after HMMs were formally developed further (Baldi et al. 1994; Krogh et al. 1994) the first software packages for statistical alignment began to appear. Two early examples were HMMER (Eddy 1995) and SAM (Hughey, Krogh 1995). Further work led to the use of statistical

alignment for phylogenetic profiling (Durbin et al. 1998) and homology testing (Hein et al. 2000). Other work by Mitchison (1999) described an early statistical attempt at co-estimating alignment and phylogeny and helped pave the way for one of the first Bayesian alignment software packages Handel (Holmes, Bruno 2001). In these works the concept of a *pair HMM* was extended by Holmes and Bruno into that of an *Evolutionary HMM* or *Tree HMM* that associated the TKF91 pair HMM with each branch in a phylogenetic tree. That tree was then used as a guide to combine suitable pair HMMs in a progressive manner analogous to classical progressive alignment described above.

Some models and methods developed in the following years relaxed the fixed-fragment assumption of TKF91 and TKF92 (Knudsen, Miyamoto 2003; Metzler 2003; Miklós et al. 2004). Meanwhile, the idea of simultaneous estimation of phylogeny and alignment has been pursued further by using, for example, direct optimisation to minimise the edit cost for the given data (POY: Wheeler et al. 2003), simulated annealing (Fleissner et al. 2005), a Bayesian framework (Lunter et al. 2005; Redelings, Suchard 2005; BAli-Phy: Suchard, Redelings 2006), or a Maximum Likelihood framework (SATé: Liu et al. 2009b). One of the latest developments in modelling indels is the use of *Finite State Transducers* or *FSTs* from automata theory which are very similar to pair HMMs in many ways (Bradley, Holmes 2007). Kim and Sinha (2007) applied FSTs to the problem of reconstructing ancestral indel histories, but more recently the same framework has been applied in a novel manner with the goal of alignment-free phylogeny reconstruction (Schwarz, Fletcher *et al.*, 2010). The performance of this approach shall be examined and compared to more traditional two-phase approaches in chapter 4.

1.4.9 Determining Accuracy of Multiple Sequence Alignment Methods

Accuracy of a particular alignment method is normally assessed by inspecting its performance when aligning a "gold-standard" dataset whose true homology is presumed known (e.g. Thompson et al. 1999a; Notredame 2002; Wallace et al. 2004). The best known benchmark of this kind is probably BAliBASE (Thompson et al. 1999b; Bahr et al. 2001; Thompson et al. 2005) which consists of alignment test cases constructed from three dimensional structural superpositions that have been manually refined, in order to ensure correct alignment of functional residues. Indeed, all of the

alignment programs mentioned above have been compared with each other using one benchmark dataset or another. However, these benchmarks contain alignments that represent functional or structural similarities between sites in different sequences, which are not necessarily congruent with evolutionary homologies. Therefore, an alignment method performing well on such benchmark tests for structure prediction is not necessarily an indication that the alignments it produces are suitable for phylogenetic analyses. In addition, the quality of some of these benchmarks has been called into question recently because there are often significant portions of a database that have unknown, contradictory or conflicting structure annotations (Edgar 2010).

For pairwise alignment, efforts have been made to try and determine the distribution of optimal local alignment scores for random sequences (e.g. Altschul et al. 2001; Pang et al. 2005b) in order to assess the statistical significance of a given alignment. However, no equivalent methodology is available for MSA methods, although recent studies have sought a method of scoring MSA accuracy without knowledge of the true alignment (Landan, Graur 2007; Hall 2008a; Penn et al. 2010b) with mixed results. As a result, simulation is the most important way to assess the accuracy of MSA methods (e.g. Pollard et al. 2004; Rosenberg 2005a; Nuin et al. 2006; Ogden, Rosenberg 2006) and has been used by the authors of just about every MSA method to demonstrate the improvements that their program brings to the field.

1.5 Project Aims

There exists a large variety of methods and computer programs for aligning multiple sequences, reconstructing phylogenetic trees and estimating evolutionary parameters. However, true phylogenetic relationships are rarely known with any certainty. Therefore, simulated data are used to investigate the accuracy, robustness and efficiency of phylogenetic inference procedures instead. In a limited number of experimental studies known phylogenies have been generated *in vivo* using a system based on the serial propagation of fast-evolving bacteriophages (Hillis et al. 1992; Sousa et al. 2008), but even then simulation may be used to reinforce the findings of the study (Sousa et al. 2008).

To give some other examples, simulated data has been used to investigate the accuracy and efficiency of phylogenetic reconstruction methods(von Neumann 1951; Huelsenbeck 1995), ancestral sequence reconstruction methods (e.g. Blanchette et al. 2004), tests for positive selection (Anisimova et al. 2003; Zhang et al. 2005), and methods of MSA (see last section, 1.4.7). They can also be used in parametric bootstrap analysis to calculate confidence intervals for parameter estimates or to estimate the null distribution for hypothesis testing (e.g. Goldman 1993). Simulation can also be used to examine the robustness of the analytical method to model misspecification, by simulating data under a complex model and analyzing them under a simplistic incorrect model (e.g. Lemmon, Moriarty 2004 and chapter 3). When the simulation does not incorporate indels, there will be no need for sequence alignment and thus an important step that may contribute significantly to errors in inference is ignored. However, current simulation programs are limited, especially concerning realistic models for simulating insertions and deletions.

In the next chapter of this thesis I will introduce a portable and flexible application, named INDELible, for generating nucleotide, amino acid and codon sequence data by simulating insertions and deletions (indels) as well as substitutions. Indels are simulated under several models of indel length distribution. The program implements a rich repertoire of substitution models, including the general unrestricted model and non-stationary non-homogeneous models of nucleotide substitution, mixture and partition models that account for heterogeneity among sites, and codon models that allow the nonsynonymous/synonymous substitution rate ratio to vary among sites and branches. In later chapters I will use some of INDELible's many unique features to evaluate the performance of different inference methods, including those for multiple sequence alignment, phylogenetic tree inference and detection of positive selection.

Chapter 2

INDELible: A Powerful and Flexible Simulator of Biological Sequence Evolution

2.1 Introduction

Many methods exist for reconstructing phylogenies from molecular sequence data, but few phylogenies are known and can be used to check their efficacy. Therefore, simulation remains the most important approach to testing the accuracy and robustness of methods of phylogenetic inference. When the simulation does not incorporate insertions and deletions (indels), there is no need for sequence alignment and thus an important step that may contribute significantly to inference errors is ignored. However, existing programs for simulating molecular sequence evolution are often found lacking, especially concerning the simulation of insertions and deletions.

Two widely used programs, Evolver (Yang 1997) and Seq-Gen (Rambaut, Grassly 1997), do not include indels at all. EvolveAGene (Hall 2008a) is inflexible and only allows the use of the spontaneous mutational spectrum of *Escherichia coli*, and Rose (Stoye et al. 1998)1859 has an unrealistic model of indel formation that does not allow simulation of, for example, transmembrane regions where low-frequency indels occur in regions with high substitution rates. Similarly, GSimulator (Varadarajan et al. 2008) does not use continuous branch lengths nor does it implement the most commonly used substitution models; it must be "trained" before use and only comes pre-trained with estimates based on the Drosophila genome. DAWG (Cartwright 2005) cannot simulate under amino acid or codon models whilst indel-Seq-Gen (Strope et al. 2007) and SIMPROT (Pang et al. 2005a) cannot simulate nucleotide or codon sequences. Only MySSP (Rosenberg 2005b) can simulate under non-stationary and non-homogenous models, while Evolver (Yang 1997) is the only program that can simulate under codon models. Furthermore, several of these programs are known to handle insertions and deletions incorrectly (Strope et al. 2009) and this is discussed further in chapter 4. Thus I have developed INDELible to plug these gaps and to provide a powerful and flexible tool for simulating molecular sequence evolution. This chapter will describe the program in full.

2.2 Outline of the Simulation Algorithm

The main difficulty in dealing with indels, especially when developing a likelihood model for inference (e.g., Bishop, Thompson 1986; Thorne et al. 1991), lies in the lack of independence of data

amongst sites in the sequence. However, if one chooses to view the entire sequence (instead of one nucleotide, amino acid or codon in the sequence) as the basic unit of evolution, then the change from one sequence to another is described by a Markov chain, with the whole sequence being the state of the chain. Thus, sequence evolution through insertions and deletions as well as substitutions can be simulated using the standard algorithm for simulating Markov chains, that is, by generating exponentially-distributed waiting times and then sampling from the jump chain (Yang 2006: pp. 303-4). This is also known as Gillespie's algorithm (Gillespie 1977).

Consider a simulation of the evolution of a sequence along a branch on a phylogeny, with both the branch length (t) and the sequence at the start of the branch given. Let $\lambda = I + D + S$ be the total event rate for the current sequence, where I, D, and S are the total insertion, deletion and substitution rates, respectively. The waiting time until the next event s_1 is generated by sampling from the exponential distribution with mean $1/\lambda$. If $s_1 > t$, then no event occurs before the end of the branch. Otherwise an event occurs at time s_1, and is randomly chosen to be an insertion, deletion, or substitution with probabilities I/λ, D/λ, or S/λ, respectively. Similarly, the location of the event is determined by random sampling with probabilities proportional to the rates. If the event is an indel (insertion or deletion), then the location is drawn uniformly from the pool of all possibilities and the length of the indel is drawn from the indel-length distribution (see below). If the event is a substitution, a site is chosen at random with a probability that is proportional to the substitution rate at that site, and the new state at the site is chosen using the transition matrix of the jump chain J (see below). Thus the new sequence at time s_1 is generated, and the rates for the new sequence and the sequence length L are updated. The time remaining for the branch ($= t - s_1$) is calculated. The next waiting time s_2 is then generated based on the total rate for the current sequence. This procedure is repeated until the end of the branch is reached, i.e. until $s_1 + s_2 + \ldots > t$.

Ideally the root sequence length L should be sampled from the distribution of sequence lengths that is implied by the model of insertions and deletions (Thorne et al. 1991). However, sampling from this distribution is complicated because of the arbitrary nature of the indel-size distributions implemented in INDELible. Instead L must be specified by the user. The root sequence is then generated by randomly sampling L characters (nucleotides, amino acids or codons) from the

equilibrium distribution of the substitution model at the root. For models of rate heterogeneity among sites, the rates at sites are generated from the rate distribution. The Gillespie algorithm is then used to simulate the evolution of the sequence from the root along the branches towards the tips of the tree. The sequences at the tips of the tree constitute a replicate dataset.

The models I have implemented assume that insertion and deletion rates are constant among sites in the sequence. The substitution process is independent of insertions and deletions, so substitutions can therefore be simulated separately from insertions and deletions. Thus, an alternative procedure is to use the Gillespie algorithm to simulate indels only, with substitutions simulated afterwards by sampling from the transition probability matrix for the branch (Yang 2006: p. 303). This will be referred to in this chapter as method 1 and is the method used by Cartwright (2005). The method described above, of simulating waiting times for substitutions as well as insertions and deletions, is referred to as method 2. For most models, method 2 is less efficient than method 1 but the opposite is true when using models of continuous rate variation among sites (see Results). However, method 2 provides a way of simulating sequences under more complex models in which the insertion and deletion rates may depend on the local sequence context and vary along the sequence (see Discussion)

2.3 Simulation of Substitutions

Substitutions are assumed to be independent among sites, and are described by a continuous-time Markov chain (see Chapter 1), characterized by the instantaneous rate matrix

$$\boldsymbol{Q} = \begin{pmatrix} q_{11} & q_{12} & \cdots & q_{1c} \\ q_{21} & q_{22} & \cdots & q_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ q_{c1} & q_{c2} & \cdots & q_{cc} \end{pmatrix}, \tag{2.1}$$

where c is the number of characters (equal to 4, 20 and 64 for nucleotides, amino acids and codons, respectively). The diagonal elements of the matrix are defined as $q_{ii} = -\sum_{j \neq i} q_{ij}$, whereas the off-diagonal elements are specified by the model. INDELible rescales the rate matrices such that branch lengths represent the expected number of substitutions per site (or the average expected number of substitutions per site under a heterogeneous-sites model).

Method 1 requires the transition probability matrix $P(t) = e^{Qt}$ for a branch of length t. As mentioned in Chapter 1, for reversible models this is calculated by numerical computation of the eigenvectors and eigenvalues of Q (Yang 1995a), while for non-reversible models it is calculated by repeated matrix squaring (Yang 2006, pp 68-70).

Method 2 requires the calculation of substitution rates at individual sites. Given Q, the rate "away" from state i is defined as $q_i = -q_{ii}$. Thus, the total substitution rate for the entire sequence is $S = \sum_{k=1}^{L} q_{i_{(k)}} r_k$ where r_k is the relative rate at site k and $i_{(k)}$ is the state at site k. Given that a substitution occurs at site k, the new state is sampled using the transition matrix of the jump chain, $M = \{m_{ij}\}$, where $m_{ij} = q_{ij}/q_i$ if $i \neq j$ and $m_{ij} = 0$ otherwise (Yang 2006, equation 9.7). In other words, if a site is currently in state i, the probability that the new state is j is simply given by m_{ij}.

2.3.1 Nucleotide Substitution Models

The most general model of nucleotide substitution possible places no constraint on the rate matrix Q. This is known as the UNREST model (Yang 1994a) and, in INDELible, it is specified by using 11 relative rate parameters (the off-diagonal elements of the rate matrix Q). INDELible calculates the equilibrium frequencies (π_i) by solving the system of simultaneous equations $\sum_i \pi_i q_{ij} = 0$ for all j, subject to the constraint $\sum_i \pi_i = 1$ (e.g., Yang 2006, p. 32). It should be noted that the rate matrix for this model is often described incorrectly in the literature (e.g. Swofford et al. 1996). In addition, INDELible also implements the general time-reversible model (GTR or REV, Tavaré 1986; Yang 1994a) and many of the commonly used models that are its special cases, such as JC69 (Jukes, Cantor 1969), K80 (Kimura 1980), K81 (Kimura 1981), F81 (Felsenstein 1981), F84 (Felsenstein, DNAML program since 1984, PHYLIP Version 2.6), HKY85 (Hasegawa et al. 1984; Hasegawa et al. 1985), T92 (Tamura 1992), and TN93 (Tamura, Nei 1993). The rates under GTR may be written as $q_{ij} = s_{ij}\pi_j$, with $s_{ij} = s_{ji}$, where s_{ij} is also known as the exchangeability between i and j (Whelan, Goldman 2004). Thus GTR is specified using the nucleotide frequencies π_j and the exchangeability parameters s_{ij}.

Table 2.1: Empirical amino-acid substitution models implemented in INDELible

Model	Source of alignment	Reference
DAYHOFF	Nuclear proteins	(Dayhoff et al. 1978)
JTT	Nuclear proteins	(Jones et al. 1992)
WAG	Nuclear proteins	(Whelan, Goldman 2001)
VT	Nuclear proteins	(Müller, Vingron 2000)
DAYHOFF (DCMUT)	Nuclear proteins	(Kosiol, Goldman 2005)
JTT (DCMUT)	Nuclear proteins	(Kosiol, Goldman 2005)
LG	Nuclear proteins	(Le, Gascuel 2008)
BLOSUM62	Nuclear proteins	(Henikoff, Henikoff 1992)
MTMAM	Mammalian mitochondrial proteins	(Yang et al. 1998)
mtREV	Vertebrate mitochondrial proteins	(Adachi, Hasegawa 1996)
MtArt	Arthropod mitochondrial proteins	(Abascal et al. 2007)
CpREV	Chloroplast proteins	(Adachi et al. 2000)
RtREV	Viral reverse transcriptase proteins	(Dimmic et al. 2002)
HIVb and HIVw	HIV-1 viral genes	(Nickle et al. 2007)
I09	I09 Influenza proteins	(Dang et al. 2009)
MtZoa	Broad Metazoan proteins	(Rota-Stabelli et al. 2009)

2.3.2 Amino-Acid Substitution Models

Currently INDELible incorporates sixteen empirical amino-acid substitution models, which were derived from analysis of protein alignments from a variety of sources (table 2.1). All of these models are time-reversible, and are specified using the stationary amino acid frequencies π_j and the amino acid exchangeabilities s_{ij} (see description above). It is also possible for the user to supply a time-reversible substitution rate matrix. INDELible also implements the Poisson model of protein evolution, which assumes that the substitution rates between any two amino acids are the same.

2.3.3 Among-Site Heterogeneity

INDELible incorporates a number of different random-sites models for simulating rate heterogeneity among sites in a sequence. Under these models the relative rates are independent and identically distributed among sites. If INDELible is simulating under a non-homogeneous model then different branches may have different models, and thus the rate for a site may change as a result of the changed model. Otherwise, the relative rate at each site is held constant throughout the simulation process with daughter sites inheriting the rate of their parent. For nucleotide and amino-acid simulations, variable

substitution rates among sites can be simulated using any of the following models: (a) a constant rate for all sites, (b) a proportion of invariable sites plus a constant rate for all other sites (+I, Hasegawa et al. 1985), (c) a continuous or discrete-gamma distribution of rates among sites (the "+Γ" and "+Γ_5" models) (Yang 1993; 1994b), and (d) a proportion of invariable sites plus gamma-distributed rates for other sites ("+I+Γ" and "+I+Γ_5" models) (Gu et al. 1995).

2.3.4 Codon Substitution Models

For codon models the state space consists of the sense codons of the genetic code, e.g., 61 sense codons for the universal code or 60 for the vertebrate mitochondrial code. Stop codons are not allowed inside a functional protein and so they are not considered in the chain. INDELible currently supports 17 genetic codes: codes 1-6, 9-16 and 21-23 listed in GenBank. The basic codon model that (as described in chapter 1) specifies the instantaneous rate of substitution from codon *i* to *j* as

$$q_{ij} = \begin{cases} 0, & \text{if } i \text{ and } j \text{ differ at two or three positions,} \\ \pi_j, & \text{if } i \text{ and } j \text{ differ by a synonymous transversion,} \\ \kappa\pi_j, & \text{if } i \text{ and } j \text{ differ by a synonymous transition,} \\ \omega\pi_j, & \text{if } i \text{ and } j \text{ differ by a nonsynonymous transversion,} \\ \omega\kappa\pi_j, & \text{if } i \text{ and } j \text{ differ by a nonsynonymous transition,} \end{cases} \tag{2.2}$$

where ω is the nonsynonymous/synonymous rate ratio, κ is the transition/transversion rate ratio, and π_j is the equilibrium frequency of codon *j* (Goldman, Yang 1994; Yang, Nielsen 1998). INDELible also allows the use of two empirical codon models (ECMs, Kosiol et al. 2007). The first (ECMrest) was constructed under the assumption that only one codon position can change instantaneously, as in equation (2.2). The second (ECMunrest) was constructed whilst allowing instantaneous doublet and triplet changes as well.

INDELible also implements several advanced models of codon substitution, which allow the selective pressure on the protein-coding gene (as measured by the nonsynonymous/synonymous rate ratio ω) to vary among branches in the tree, among sites (codons) in the gene, or among both branches and sites (see Anisimova, Kosiol 2009 for a recent review). The site models allow ω to vary among sites (Nielsen, Yang 1998; Yang et al. 2000). All the site models are special cases of model M3

(discrete), which assumes a general discrete distribution for ω (Yang et al. 2000). M3 is implemented in INDELible by specifying the number of site classes, and the ω ratios and proportions for those site classes. A small script is included with INDELible, which can calculate the discrete ω values from the parameters of the other models M4-M13 of Yang *et al*. (2000).

The branch models (Yang 1998) and branch-site models (Yang, Nielsen 2002; Yang et al. 2005; Zhang et al. 2005) are implemented in INDELible as well. The latter allows the ω ratio to vary among branches and among sites. While the branch-site model described by Yang *et al.* (2005) allows only two types of branch (the foreground and background branches) and only four site classes, INDELible allows an arbitrary number of site classes and branch types to be used.

Those codon models are widely used in likelihood ratio tests of natural selection affecting the evolution of protein-coding genes. Implementation of those models in INDELible makes it possible to evaluate the impact of alignment errors and of insertions and deletions on the robustness of those methods and that will be done, for the first time, in chapter 3 of this thesis.

2.4 Non-Stationary and Non-Homogeneous Processes

Most models currently used in phylogenetic analysis assume stationarity and homogeneity of the substitution process across the whole tree; that is to say, substitutions have occurred according to the same rate matrix Q, and nucleotide, amino-acid or codon frequencies have remained more or less constant, during the course of evolution. Sequences from distantly related species have often been noted to have different nucleotide or amino acid frequencies, which is a clear indication of violation of these assumptions. Only a few attempts have been made to implement nonhomogeneous models (Yang, Roberts 1995; Galtier, Gouy 1998; Blanquart, Lartillot 2006) for phylogenetic inference. Therefore, data simulated under such non-homogeneous and non-stationary conditions should be useful for testing the robustness of phylogenetic reconstruction methods.

The branch and branch-site models of codon substitution mentioned above may be considered examples of non-homogeneous models, because the ω ratio and thus the rate matrix Q varies along branches. INDELible allows any parameter or any aspect of the evolutionary model to change along

branches in the tree. Each branch may have its own insertion/deletion rates and size distributions, equilibrium frequencies, substitution model or level of rate heterogeneity among sites. Under codon models the genetic code is also allowed to vary between branches, as long as all the codes used on the tree share the same stop codons. Parameters are also allowed to change at arbitrary points along a branch. This is achieved by specifying a tree with an internal node that has only one daughter branch.

2.5 Simulation of Insertions and Deletions

2.5.1 Indel Formation

INDELible treats insertions and deletions as separate processes, each with its own size distribution and instantaneous rate. The model assumes that insertions and deletions occur at the fixed rates λ_I and λ_D, respectively, at every site in the sequence. I define one time unit as one expected substitution per site, so λ_I and λ_D are then the expected numbers of indels per substitution. In simulation under codon models, a site refers to a codon, and so only indels of whole codons are permitted.

Insertions are relatively simple to simulate. If a sequence has L sites then there are $L + 1$ possible positions for insertion (including both ends of the sequence). Therefore, the total rate of insertions is given by $I = \lambda_I(L + 1)$. Insertions at the two ends of the sequence are allowed, and the sequence has an "immortal link" at the beginning (Thorne et al. 1991). If an insertion occurs, the insertion-size distribution is used to generate the size of the insertion (u). Then u characters (nucleotides, amino acids or codons) are generated by randomly sampling from the equilibrium distribution of the substitution model to form the insertion sequence. For site-heterogeneous models, the rates for the u sites are then generated by sampling at random from the rate distribution.

Deletions are more complex to simulate as one has to make somewhat arbitrary decisions concerning deletions at the ends of the sequence. I choose to follow the procedure of Cartwright (2005), and consider that the simulated sequence, of length L, lies within a larger sequence, of length N, with $N \gg L$. Let the maximum deletion length be M, with $M \ll N$. Therefore, a deletion of size u in the larger sequence will only delete some of the smaller sequence if it occurs at any of the L sites of the smaller sequence or at any of the $u - 1$ sites preceding the smaller sequence. Deletions are

assumed to occur uniformly in the larger sequence so the probability that a deletion of size u in the larger sequence deletes some sites from the smaller sequence is given by $(u - 1 + L)/N$.

Thus the probability that a deletion in the larger sequence deletes some sites in the smaller sequence is $P_D = (\overline{u}_D - 1 + L)/N$, where $\overline{u}_D$ is the mean deletion size (Cartwright 2005). The total rate of deletion in the larger sequence is $N\lambda_D$ where λ_D is the rate of deletion per site, so that the total rate of deletion in the smaller sequence is $D = N\lambda_D P_D = \lambda_D (\overline{u}_D - 1 + L)$. This is independent of N.

2.5.2 Indel Size Distributions

INDELible uses two independent distributions to model the sizes of insertions and deletions. For the sake of simplicity, here I use indel size distribution to refer to both. Several different indel size distributions are implemented in INDELible.

The first is the negative binomial distribution, for which the probability that the indel has size u is

$$f(u) = \binom{r+u-2}{u-1}(1-q)^r q^{u-1},\ u = 1, 2, \ldots, \tag{2.2}$$

where the parameters are the integer r and probability q. This mean of this distribution is given by $\overline{u} = 1 + rq/(1 - q)$ and the variance is $rq/(1 - q)^2$. If $r = 1$, the distribution reduces to the geometric distribution.

The second model is a power law or Zipfian distribution, for which indel length u has probability

$$f(u) = \frac{u^{-a}}{\zeta(a)},\ u = 1, 2, \ldots, \tag{2.3}$$

where $a > 1$ is a parameter of the distribution and $\zeta(a) = \sum_{v=1}^{\infty} v^{-a}$ is the Riemann Zeta function. This distribution has a very heavy tail, and the mean is infinite if $a < 2$ and the variance is infinite if $a < 3$. If $a > 2$, the mean is $\overline{u} = \zeta(a - 1)/\zeta(a)$, and if $a > 3$, the variance is $\zeta(a - 2)/\zeta(a) - \overline{u}^2$. Empirical estimates of a range from 1.5 to 2, with infinite variance (Benner et al. 1993; Gu, Li 1995; Zhang, Gerstein 2003; Chang, Benner 2004; Yamane et al. 2006; Cartwright 2009). There is also evidence that parameter a, which is inversely related to the indel size, differs for insertions and deletions (Gu,

Li 1995; Zhang, Gerstein 2003), so INDELible's ability to allow different length distributions for insertions and deletions may prove useful.

The third model is the Lavalette distribution, an extension of Zipf's law that was originally proposed to describe the distribution of journal impact factors (Lavalette 1996; Popescu et al. 1997; 2003). For this distribution the probability of an indel of size u is given by:

$$f(u) \propto \left(\frac{u\,M}{M-u+1} \right)^{-a}, \quad u = 1, 2, \ldots, M, \tag{2.4}$$

where M is the maximum indel size and a is the parameter. The proportionality constant is determined such that the probabilities sum to 1. This model has two desirable features. Firstly, the mean and variance are always finite because of the maximum length M. Secondly, it can be made to approximate the Zipf distribution arbitrarily well by the use of a large value for M. This is because the two distributions differ only by the factor $\varphi = [M/(M-u+1)]^{-a}$, apart from the normalizing constants, and $\varphi \approx 1$ when $M \gg 1$. Figure 2.1 shows the distribution for a few different values of M.

Besides these three models above, INDELible also allows the user to define an indel size distribution.

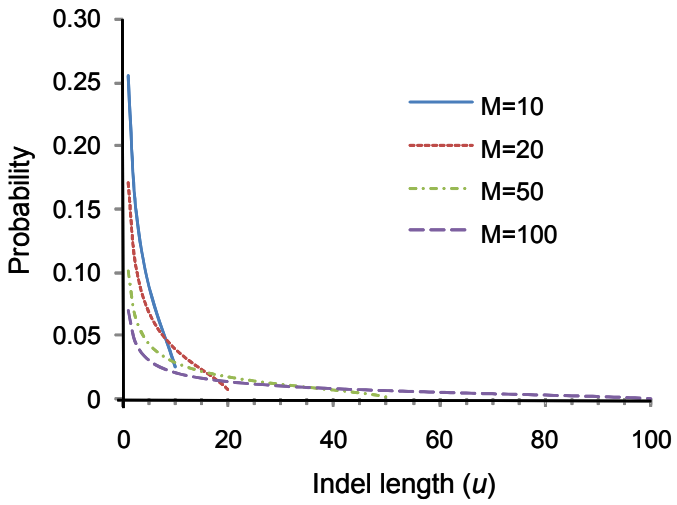


Figure 2.1: The Lavalette distribution

Indel lengths are plotted for different values of the maximum indel length M, with $a = 0.5$ fixed (see eq. 2.4). N.B. u can take integer values 1, 2, …, M only.

A number of authors have attempted to estimate empirical indel size distributions. Gu and Li (1995) suggested that the geometric model was an inadequate fit to the data and found to the power-law model to fit the data much better. Many other studies have also found that the power-law fitted a variety of datasets reasonably well (Benner et al. 1993; Zhang, Gerstein 2003; Chang, Benner 2004; Yamane et al. 2006). Qian and Goldstein (2001) used a mixture of four exponential distributions to describe indel lengths, which was later adapted into a distance-dependent indel length distribution for use in the simulation program SIMPROT (Pang et al. 2005a). This distribution appears to be more complicated than necessary.

2.6 Program Validation

I conducted extensive simulations to confirm the validity of this simulation program. Relying on the consistency of maximum likelihood estimates, I chose to validate the implementation of the substitution models by simulating larger and larger datasets (e.g. with 10^6 or 10^7 sites) before confirming that the parameter estimates were close to the true values by analysing them under the same model using BASEML and CODEML in the PAML package (Yang 1997). It is more difficult to validate simulations under the different models of insertions and deletions, as correct analytical results are lacking. However, I did compare the observed indel size distribution of the simulated datasets with the true distribution and found that they were a close match. I also simulated datasets on trees of 2, 8 or 40 taxa with insertions only, with deletions only, or with both insertions and deletions, using many different rates, parameters and length distributions. The proportions of columns in the true alignment that had 0, 1, 2, … gaps were calculated and compared with the correct proportions generated using a simple and independent simulation program that kept track of the sequence lengths only. In all combinations that were investigated there was good agreement between the two.

My extensive comparisons with DAWG revealed a few problems with DAWG version 1.1.2 and earlier. For example, there are two biological mechanisms that can generate columns with all gaps in the true alignment: (a) deleted insertions, i.e., deletion of part of an earlier insertion on the same branch, and (b) parallel deletions, i.e., deletion of the same nucleotides but along different lineages.

DAWG keeps track of (b) but not of (a). Furthermore, the true alignment produced by DAWG was sometimes incorrect with nucleotides from parallel insertions misaligned. Those bugs have been fixed in version 1.2 of the program (R. Cartwright, pers. comm.).

2.7 Results

DAWG (Cartwright 2005) is the simulation program that is most similar to INDELible. While DAWG does not have some of the advanced features available in INDELible, it is possible to simulate data under the same nucleotide-substitution models in order to make a fair comparison. Thus I conducted several computer simulations to examine the computational efficiency of the two programs. Sequence data was simulated under the HKY model, with $\kappa = 2$ and base frequencies 0.4 (T), 0.3 (C), 0.2 (A) and 0.1 (G). In the base model I set the insertion and deletion rates to $\lambda_I = \lambda_D = 0.1$ per substitution, with the indel length following a negative binomial distribution with $q = 0.25$ and $r = 1$ (the geometric distribution). The phylogenetic tree used was symmetric with 32 taxa and all branch lengths were set to be 0.1 substitutions per site. Substitution rates over sites were either constant or followed the gamma distribution with shape parameter $\alpha = 1$. The number of replicate datasets was 100. I then explored several variations of this base simulation scheme in order to examine the impact that different factors had on simulation efficiency. INDELible (methods 1 and 2) and DAWG were used to generate the data, and the results are shown in figure 2.2 (panels A to E). Factors explored include the number of taxa (A), the amount of evolution measured by the branch length (B), the sequence length at the root (C), the insertion/deletion rate ratio λ_I/λ_D (D) and the average indel length (E).

DAWG is faster than INDELible in simple circumstances, such as when simulating short sequences with low insertion rates and small insertions on small trees with few taxa and short branches. However, with increased simulation complexity, the time taken by DAWG increases much faster than the time taken by INDELible. The exception to this pattern is simulations involving INDELible method 2, which is sensitive to the average branch length as longer branches need simulation of more rounds of exponential-waiting times in the main algorithm. However, method 2 has a speed advantage over

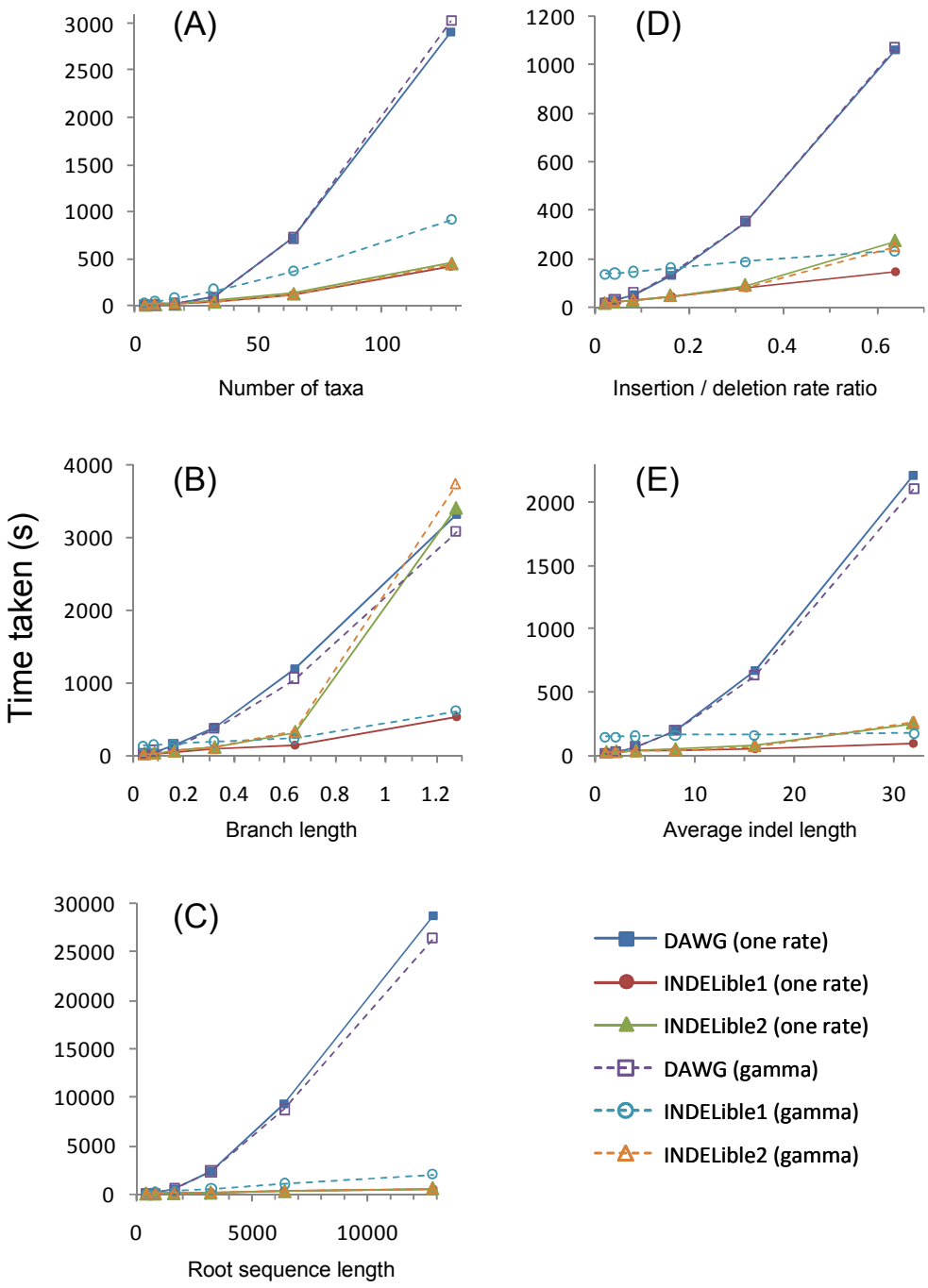


Figure 2.2: Speed comparison between DAWG and INDELible

Comparisons are made with and without continuous gamma rate heterogeneity. The base simulation model used is specified by the settings in figure 2.6. In each plot one factor is varied at a time to see its impact. INDELible simulation under methods 1 and 2 is referred to as INDELible1 and INDELible2 respectively. The tests were carried out on a SunFire Opteron X4600M2 server running Linux.

method 1 and DAWG for simulation under the continuous gamma model of variable rates among sites. Under this model, every site has a distinct rate. This means the transition probability matrix $P(t)$ needs to be calculated for every site on every branch. Conversely, the transition matrix of the jump chain (M in method 2, page 68) is the same for all sites and so does not need to be calculated for every site, leading to an increase in computational efficiency.

These speed differences between DAWG and INDELible are largely a matter of programming design. Both programs are written in C++, and both programs store sequence information in the STL vector container. However, INDELible implements insertions via a modified lookup table approach whose execution time is largely independent of simulation complexity, but can be slow in very simple simulations. In contrast, DAWG implements insertions via the C++ function vector::insert, the speed of which is proportional to the number of elements between the insertion position and the end of the vector (due to moving) plus the number of elements to be inserted (due to copying). This shall be discussed in a more detail in the next section.

2.8 Discussion

2.8.1 Computational Efficiency of INDELible – Data Storage

The efficient mechanism for storing data in INDELible is best explained through some examples. Figure 2.3 demonstrates how sequence data is stored and manipulated in INDELible as a root sequence is created and evolves through a chain of 5 indel events in a simulation (from top to bottom). This particular simulation begins with the creation of a root sequence of size 40. This root sequence is stored in a vector of 40 integers representing the states at the different sites (blue squares). At the same time a vector of size 41 is created whose elements are all also vectors of integers – i.e. a "vector of vectors". The 41 positions are the 41 possible positions where insertions can occur between sites in the root. At first, all of these 41 vectors are empty. The first indel event (1) is a deletion of size 3 in the root sequence. This deletion involves 3 *write* operations to the root vector in C++ to label these sites as deleted sites that should be ignored for the rest of the simulation. The second event is an insertion of size 3 occuring within the root sequence (2). This insertion involves copying 3 integers to

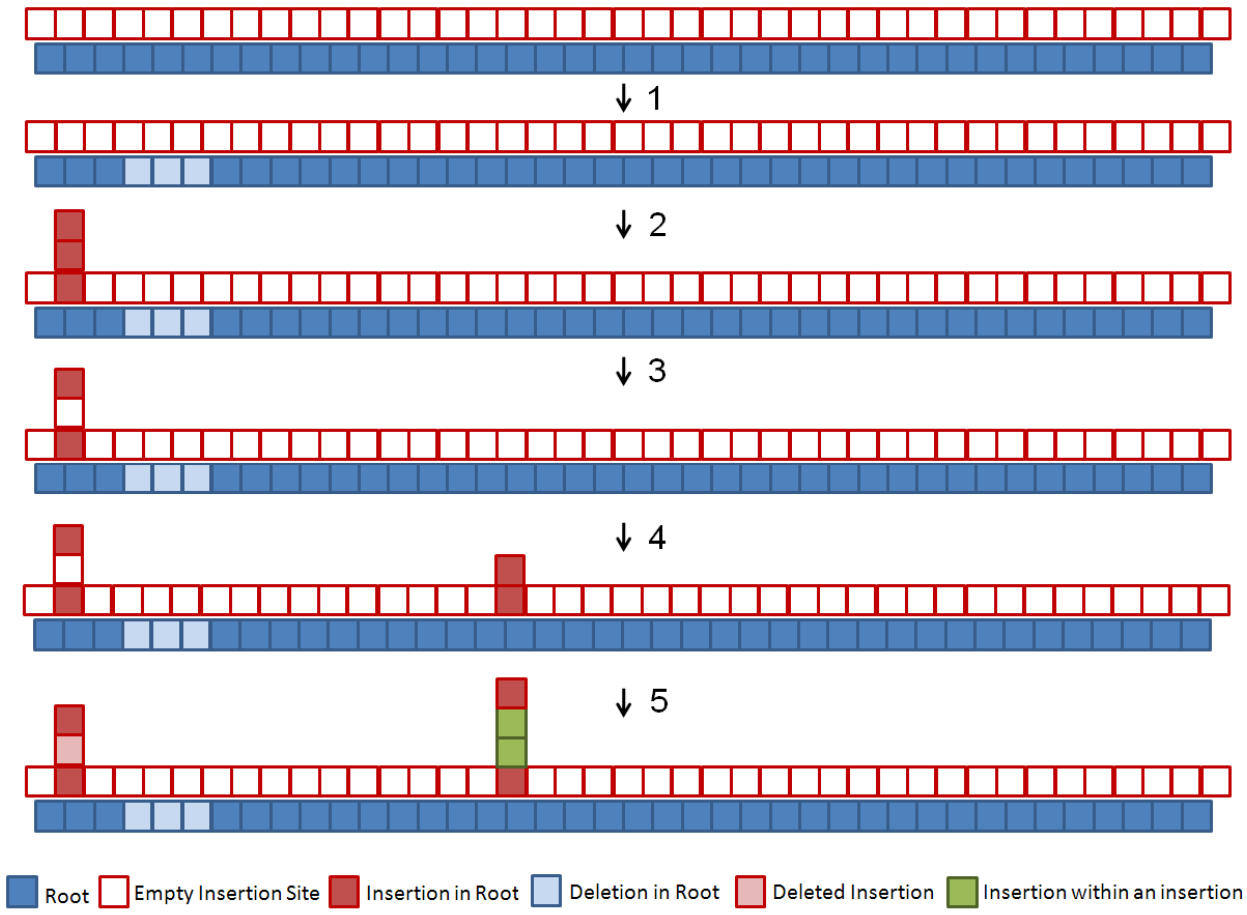


Figure 2.3: Storage of sequence data in INDELible during a simulation

This toy example shows a sequence of initial length 40 being subjected to 5 indel events in a simulation (from top to bottom). Full explanation of the data storage is in the main text. The first indel event is a deletion of size 3 at position 4 in the sequence (1). This is followed by an insertion of length 3 between positions 1 and 2 of the sequence (2). Next we have a deletion occurring at position 3 of the new sequence, i.e. position 2 of the previous insertion (3). This is followed by an insertion of size 2 between sites 15 and 16 in the "current" sequence, i.e. between positions 13 and 14 of the initial root sequence (4). Finally there is an insertion between positions 16 and 17 of the "current" sequence, i.e. between positions 1 and 2 of the previous insertion (5). Thus, this toy example encompasses each kind of indel event that can occur during a simulation: deletion of root sites, insertion between root sites, deletion of inserted sites, and insertions within pre-existing insertions.

the end of the empty vector at position 2 (of 41) in the "insertion" container, i.e. 3 *copy* operations. The third event is a deletion of length 1 occurring within that insertion (3). Again, this just involves one *write* operation to the insertion vector at position 2, in order to label that site as deleted. The fourth event is another insertion in the root, this time of length 2, so involves 2 *copy* operations. The fifth and final event is an insertion of length 2 within a previous insertion. This is performed using the C++ vector::insert function, and so involves 1 *move* operation to move the second inserted site 2 spaces to make room (as elements of a C++ vector must be contiguous and in order in computer memory), and 2 *copy* operations to copy over the newly inserted sites to the vector.
So, for INDELible the total cost of this simulation was 3 write + 3 copy + 1 write + 2 copy + (1 move + 2 copy) = 4 writes + 7 copies + 1 move.

Other programs, such as DAWG, use the in-built vector::insert and vector::delete functions to literally carry out insertion and deletion operations on a single vector containing the sequence data. Thus, these programs are much slower because of the requirement that the C++ vector container stores its elements, in order, in contiguous memory (it is this property that gives vectors their excellent access speed). So, using the same example, we can work out the consequence of using this convenient strategy. The first deletion (1) of size 3 now involves 34 *move* operations as the entire back of the sequence is moved to ensure the vector is contiguous in memory (vector::insert and vector::delete operations work by moving the back end of the sequence in memory, with the portion of the vector containing the first element remaining static). This move is shown in figure 2.4 (top). The sequence vector now has length 37. The second event (2), an insertion of size 3, involves 36 *move* operations as the back of the sequence is moved to make room for the insertion, followed by 3 *copy* operations to place that actual insertion within the sequence container. This move is also shown in figure 2.4 (bottom). The vector now has length 40 again. If we continue in this manner we find that the computational cost was 34 moves + (36 moves + 3 copies) + 38 moves + (24 moves + 2 copies) + (25 moves + 2 copies) = 157 moves + 7 copies. Clearly this is a contrived example and the cost will be less if indels occur towards the back of the sequence, but it serves to show that it is all this literal moving of sequence sites around in the computer memory that makes other programs get exponentially slower. If the root (or inserted) sequences, and therefore the vectors, are much longer

there is obviously a lot more elements that need moving about (or copying) and this takes longer, and if the insertion/deletion rates are higher or branches are longer then there will be that many more reshuffling events going on behind the scene. This explains the performance curves that can be seen in figure 2.2

Despite these speed increases there is another advantage to storing sequence data in this manner. Whilst the method may be more memory intensive it allows for the true alignment to be directly read from the sequence vectors, without any ambiguity. If a program employs a method where sequence data is literally inserted and deleted from memory then elaborate "colouring" schemes need to be used in order to identify what happened at a given site through the simulation

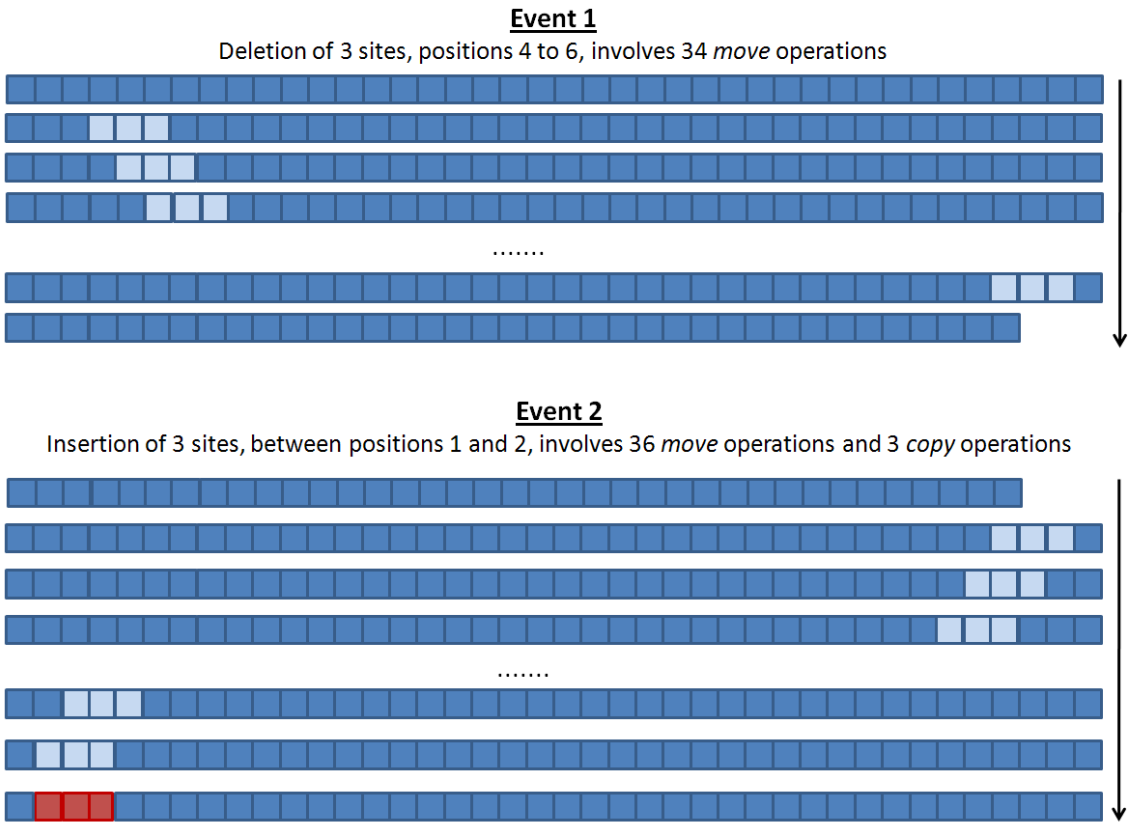


Figure 2.4: Inefficient memory management in other programs

The first two events of the same simulation process (figure 2.3) are shown here as step-by-step memory moves. Because the C++ vector container stores its elements in contiguous memory every insert and delete operation involves a lot of unnecessary shuffling that can be avoided.

history so that the true alignment can be reconstructed. Such colouring schemes are undoubtedly very clever, but leave the door open for errors because of the complicated nature of phylogenetic simulations. It was such an error that was responsible for the aforementioned bug that I found in earlier versions of the DAWG program (R. Cartwright, pers. comm. and DAWG website).

2.8.2 Computational Efficiency of INDELible – Locating a Site

When simulating using constant insertion and deletion rates with a program that literally inserts and deletes sites in a vector, determining the location where an insertion or deletion will occur is very simple. First, for a sequence of length L the algorithm generates a random integer u between 1 and L if a deletion has occurred, or between 1 and $L+1$ if an insertion has occurred, and then calls either the vector::delete function (using the boundaries u and $u+n$), or the vector::insert function (with position u), when the indel was sampled to have length n.

In contrast, the speed efficiency of INDELible's data storage procedures comes at the expense of convenience. The problem that must be overcome by INDELible is to find where the "real" nth position in the sequence is given that there will be many deleted sites in memory, i.e. there will be sites that do not exist in the simulation sequence that is being evolved but that still exist in computer memory (so as to avoid unnecessary memory shuffling to gain a speed advantage, and to enable accurate reconstruction of the true alignment unambiguously and correctly). Therefore the site positions in the sequence do not correspond with the site positions in memory. So now the algorithm must determine the site j for which $\sum_{k=1}^{j} v_k = u \ \ (j \geq u)$ where v_k is a binary variable denoting whether a site in memory has been deleted or not.

A similar problem occurs when using method 2 for substitutions, regardless of the data structure in memory. This is because, if the total substitution rate for the whole sequence is $S = \sum_{k=1}^{L} q_{i_{(k)}} r_k$ then a substitution site is chosen by sampling $s = S \times u$ where u is a random number between 0 and 1. This means that the substitution occurs at the site j where $\sum_{k=1}^{j-1} q_{i_{(k)}} < s \leq \sum_{k=1}^{j} q_{i_{(k)}}$.

For both cases, a naive approach would just iterate from the start of the vector and count the number of non-deleted sites, or add the individual substitution rates from non-deleted sites, in order to find the chosen site *j*. This would involve a large number of additions and accesses to the vector memory and would go some way to negate the speed increases that the efficient memory management facilitates when using constant indel rates across sites (N.B. for varying indel rates across sites, or when using method 2 for substitutions, this is a problem that is unavoidable).

To circumvent this difficulty INDELible constructs and maintains a decision-tree indexing system to navigate the sites quickly (see figure 2.5). In principle this is constructed as follows.

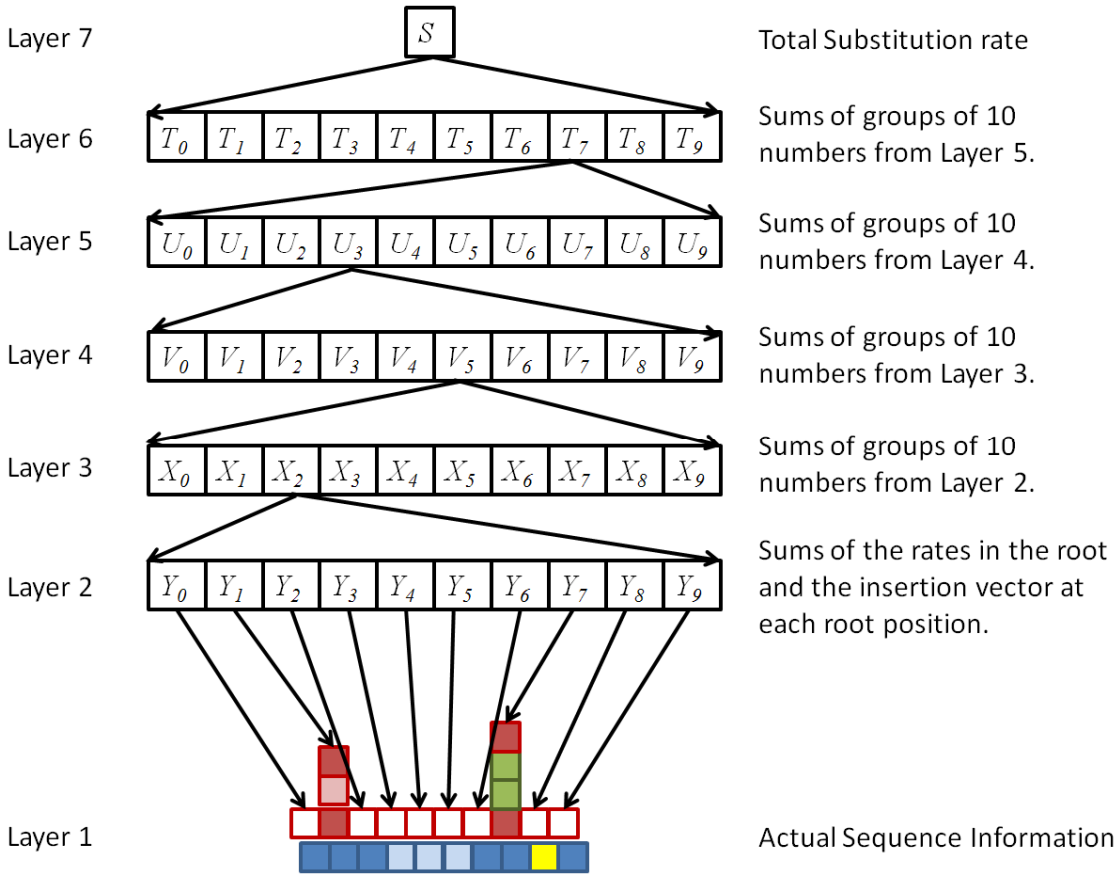


Figure 2.5: Decision tree structure for quickly navigating sequence sites

A pictorial example of how one can navigate very quickly through sequence substitution rates in order to find a particular area where a substitution has randomly been chosen to occur (when using method 2). The exact same approach is used for locating the area where an indel occurs (see main text).

Firstly, in layer 1, we have the actual sequence information which is site specific. In layer 2, vectors are constructed that contain the sum of the rates at each insertion vector site and root site position. In layer 3 a smaller vector is constructed where each element is the sum of 10 numbers from layer 2. Then in layer 4 an even smaller vector is constructed where each element is the sum of 10 numbers from layer 3 and so on. This process is repeated, layer after layer, until there is one vector at the top of the decision tree containing only one element that is either the total substitution rate, or total insertion/deletion rate of the whole sequence.

` Once the tree has been constructed it is simple to update once an evolutionary event occurs. For example, if we define $T_k^* = \sum_{j=0}^{k} T_j$ then instead of using $\sum_{k=1}^{j-1} q_{i_{(k)}} < s \le \sum_{k=1}^{j} q_{i_{(k)}}$ to find the site we want, we begin by comparing numbers in the second highest layer (layer 6 in the example in figure 2.5). If, for example, we find that $T_6^* < s \le T_7^*$, we have narrowed down the search and can look now in layer 5, in the section that T_7 holds the total for. A few more additions and comparisons and we may find $U_2^* < s - T_6^* \le U_3^*$ allowing us to look in the corresponding section of the layer below. Continuing on we find $V_4^* < s - T_6^* - U_2^* \le V_5^*$, then $X_1^* < s - T_6^* - U_2^* - V_4^* \le X_2^*$, and finally we find that $Y_7^* < s - T_6^* - U_2^* - V_4^* - X_1^* \le Y_8^*$ which has narrowed down the substitution to occurring at the site marked in yellow.

Thus, in this example, we have navigated a sequence with a length of approximately one million sites (or more) using, on average, 25 additions and 25 comparisons, as opposed to a naive strategy of exhaustively cycling through the actual sequence where we would have needed, on average, 500,000 additions and comparisons! Once the substitution has occurred the tree is easily maintained. If the difference between the old and new substitution rates at the yellow site is s^* then we simply add s^* to the elements Y_8, X_2, V_5, U_3, T_7, S and the process may begin again. The principle is exactly the same for insertion/deletion rates whether they are constant across sites or not. For method 1 of evolving sequences such a decision tree structure is not necessary. However, the use of the decision-tree facilitates the use of the creative data structure mentioned in section 2.8.1 which is what gives INDELible its speed advantage over competitors. Furthermore, this new approach would

be utterly necessary if one wanted to implement indel rates that are not constant across sites, or wanted to simulate under context-dependent models using method 2 (see below). The fact that the algorithm is adaptable in this way is one of the features that makes INDELible unique and innovational among simulation programs. General features of the program are discussed below.

2.8.3 Features of INDELible

INDELible is driven by a control file (figure 2.6). The program is designed to be flexible, and so a wide range of options can be specified in this file to control different aspects of the simulation including, but not limited to, the substitution model, the heterogeneous-rates model, indel models and indel-size distributions, and the underlying phylogeny. No constraints are placed on the size and structure of the tree, the sequence length, or the values of model parameters. The tree with branch lengths (measured by the expected number of substitutions per site) may be specified by the user or created at random from the birth-death process with species sampling (Yang, Rannala 1997). The user can also choose to generate random ultrametric or non-ultrametric branch lengths for a given topology, or can choose to rescale the tree to have a given depth or length, or rescale the tree to have a given maximum pairwise distance between sequences at the tips. Furthermore, the user can choose to make branches on the tree randomly larger or smaller in order to simulate deviations from the molecular clock.

INDELible also offers the ability to simulate data in multiple partitions where different partitions can have different substitution models, indel lengths, or heterogeneous rate distributions and may also evolve on different trees (e.g. to simulate gene-tree/species-tree conflict). Deletions are not allowed to span different partitions, the different partitions must have the same data type (nucleotide, amino-acid or codon) and the tree must have the same number of leaves. Apart from those restrictions, every other parameter or setting is permitted to vary between partitions. The history of any insertions and deletions that occur is maintained during the course of the simulation. To avoid possible complications when outputting sequences, deletions are not actually removed from the computer memory but are simply marked as deletions and ignored during the remainder of the

simulation. For the same reason, inserted bases/residues are stored in separate memory containers to those in the original sequence at the root. Thus, at the end of the simulation, sites are immediately recognisable as either core sites that evolved from the root, deleted core sites, insertions, or deleted insertions, and the true alignment can be assembled and output easily, without the need for any translation or interpretation. INDELible also offers the option to print inserted residues in lower-case and print core residues that evolved from the root in upper-case, or to distinguish between gaps placed opposite insertions and gaps that are deleted insertions. In addition, codon sequences can also be translated into amino-acid sequences for output, using the genetic code of that branch.

```
[TYPE] NUCLEOTIDE 1               // nucleotide simulation using algorithm 1

[MODEL] m1
  [submodel]    HKY 2             // HKY with kappa = 2
  [basefreq]    0.4 0.3 0.2 0.1   // frequencies for T C A G
  [rates]       0  1  0           // pInv alpha Ngammacat
  [indelrate]   0.1               // insertion rate = deletion rate = 0.1
                                  //  (relative to average substitution rate of 1)
  [indelmodel]  NB 1 0.25         // Geometric length distribution
                                  //  with mean indel length of 4

[TREE] t2  (a:0.1,b:0.1);         // user defined guide tree

[PARTITIONS] p1                   // one partition with root length
  [t2 m1 1000]                    // of 1000 that uses tree t2

[EVOLVE]  p1 100 outputname       // produce 100 replicates of partition p1
```

Figure 2.6: An example input file for INDELible.

In this example the substitution model has been set to HKY+Γ with a transition/transversion rate ratio of $\kappa = 2$ and stationary base frequencies of 0.4 (T), 0.3 (C), 0.2 (A), and 0.1 (G). Continuous gamma rate variation has been used with shape parameter $\alpha = 1$. Insertions and deletions have both been set to have an instantaneous rate of 0.1 (relative to an average substitution rate of 1), and have been assigned the same geometric length distribution with a mean length of 4. Next the phylogeny with branch lengths is specified. In the simulations for the speed tests, a symmetric, 32-taxa, strictly bifurcating tree with all branch lengths equal to 0.1 was used instead. This simulation will create 100 replicate datasets each containing one partition with a randomly created root sequence of 1000 bases.

A summary of features of INDELible in comparison with other simulation programs is provided in table 2.2. INDELible is unique in its implementation of codon models and non-stationary and non-homogeneous models among programs of indel simulation.

Table 2.2: Comparison of simulation programs

Feature	Seq-Gen v1.3.2	Evolver v4	Rose v1.3	DAWG v1.1.2	MySSP v1.0	Indel-Seq-Gen v1.0.3	EvolveAGene v3	GSimulator v1.1	SIMPROT v1.01	INDELible v1.04
GTR	x	x		x	x					x
UNREST										x
Empirical amino acid models	6	10[a]				3			3	16[a]
Empirical codon models										2
Codon *site* model		x								x
Codon *branch* model		x								x
Codon *branch-site* model		x								x
Non-stationary models					x					x
Discrete gamma	x	x								x
Continuous gamma	x	x		x	x				x	x
Proportion of invariant sites	x			x		x				x
Indels			x	x	x	x	x	x	x	x
Ancestral sequences	x	x	x	x	x	x	x	x		x
Batch mode		x		x	x					x
Multi-gene mode	x				x	x			x	x
Platform										
Unix	x	x	x	x		x	x	x	x	x
Mac OS X	x	x	x	x		x	x			x
Win32	x	x		x	x		x		x	x

Note.— [a]Evolver and INDELible can also use user-defined amino-acid substitution models.

2.8.4 Correct Simulation Under a Model and Biological Realism

It is important for any indel-simulation program to simulate data correctly under a model that includes insertions and deletions, as well as substitutions. That is to say, it must generate datasets with the correct probability distribution under such a model. Most existing indel-simulation programs do not appear to have achieved this goal as they often involve somewhat arbitrary manipulations of the simulation process that cannot be justified under any model. It is often claimed that those manipulations improve the biological realism of the generated data. A common mistake is to fix the sequence at the root of the tree to be a real sequence rather than generating a sequence at random (as

in EvolveAGene). However, in a model of insertions, deletions and substitutions, the sequence at the root is a random realisation of the model and should be allowed to vary among datasets.

While it is certainly important for any simulation to represent real-data scenarios, this goal should be achieved by specifying representative values of the parameters in the model (such as substitution rates, sequence length, base or amino acid frequencies, and the size and shape of the tree, etc). Most parameters, such as substitution rates, stationary frequencies, or heterogeneous rate distributions, can be easily estimated via maximum likelihood using standard phylogenetic software (e.g. PAML: Yang 1997). Parameters for indel formation and indel length distributions present more of a problem. INDELible is designed to be a simulation program and so does not include methods for estimating any model parameters from real data, which is the remit of an inference tool. However, a number of studies have produced estimates of the insertion and deletion rates (λ_I and λ_D) relative to the substitution rate (λ_S), with $\lambda_S/(\lambda_I + \lambda_D)$ normally estimated to be around 13-15 (Silva, Kondrashov 2002; Britten et al. 2003; Ogurtsov et al. 2004). Estimates have also suggested that insertions occur less often than deletions, with λ_D/λ_I ranging from 1.3 to 4 (Gu, Li 1995; Zhang, Gerstein 2003; Arndt, Hwa 2004), although it is noted that Mills *et al.* (2006) estimated $\lambda_D/\lambda_I \approx 1$ in a comparison of chimpanzee and human genomes. Thus, the ability of INDELible to specify separate insertion and deletion length distributions, and separate insertion and deletion rates (λ_I, λ_D), and to permit those parameters to change at any point on the tree, may be of importance for realistic simulation of molecular sequence evolution.

2.8.5 Extending the Evolutionary Model

In the future, INDELible could be improved upon in a number of ways, by incorporating important features of sequence or genome evolution. Indeed the current incarnation of INDELible is mainly designed for generating sequences suitable for phylogenetic comparison, and does not currently include models of genome rearrangements such as inversion, duplication, and translocation.
To evaluate methods that attempt to reconstruct ancestral genomes (Blanchette et al. 2004), it may be important to simulate such large-scale events. Also, repetitive elements appear to have very high

insertion and deletion rates. For example, the ALU sequence in humans is about 300 base pairs long, and recurs 300,000 times throughout the DNA. This causes a conspicuous spike in the observed indel size distribution at around ≈300bp when the human genome is compared against other genomes (Kent et al. 2003). Even shorter sequences may be repeated as many as 10^6 times. Such repetitive sequences can create indel hotspots and clearly violate the assumption of uniform insertion/deletion rates.

Similarly, substitution or mutation rate is known to depend on the local sequence context. As mentioned in chapter 1, the most dramatic instance of such a context effect is found in the so-called CpG dinucleotide "hotspots" (e.g. Ehrlich, Wang 1981). Codon models incorporate the context effect to some extent by accounting for dependence between the different positions of the codon triplet, but they cannot deal with context effects across codon boundaries (Pedersen et al. 1998; Siepel, Haussler 2004). There is also some evidence that rates of substitution, insertion, and deletion are positively correlated, meaning that genomic regions with high substitution rates also display high insertion and deletion rates (Waterston 2002). It has also been found that there is a specific enrichment of CpG dinucleotides in close proximity to insertion events and that both insertions and deletions are more common in higher G+C content sequences (Taylor *et al*., 2004).

It should be straightforward to extend INDELible to simulate genome-rearrangement events, to accommodate insertions/deletions of repetitive elements, substitutional context effects, or correlated substitution and indel rates, as long as precise models for these processes can be formulated. Note that simulation of the evolutionary process by Gillespie's algorithm (INDELible method 2 but not method 1 or DAWG) is still possible as long as one can generate a sequence at the root of the tree and calculate the instantaneous rates; there is no need for matrix-exponential solutions to the transition probabilities, contra Varadarajan *et al.* (2008) and their justification for GSimulator's design. Even with dependence among sites in the sequence, the evolution from one sequence to another can still be described by a Markov chain, and the instantaneous rates of various events are easy to calculate. Thus, it should be straightforward to simulate the process. Nevertheless, such processes are poorly understood at present, and the lack of any suitable inference tools to analyze real data makes it difficult to obtain reliable parameter estimates under such models.

2.9 Implementation Details and Program Availability

INDELible, now in version 1.04, is written in standard ANSI C++, and has been tested on Windows, Mac OS X and Linux systems. Pre-compiled executables are provided for Mac OS X and Windows while the C++ source code is provided for compilation on UNIX systems. The program is distributed free of charge (for academic use) at the web site http://abacus.gene.ucl.ac.uk/software/indelible/ which also contains an extensive manual and tutorial with examples.

Chapter 3

The Effect of Insertions, Deletions and Alignment Errors on the Branch-Site Test of Positive Selection

3.1 Introduction

The nonsynonymous to synonymous substitution rate ratio (ω) can be used to measure the selective pressure on the protein. A ratio $\omega < 1$ indicates purifying selection acting to preserve the amino-acid sequence, whereas a neutrally evolving sequence will exhibit $\omega \approx 1$, and $\omega > 1$ represents positive selection driving the fixation of amino-acid changes.

Many methods have been developed that aim to detect positive selection that affects specific lineages (Messier, Stewart 1997; Zhang et al. 1997; Yang 1998) or a subset of sites (Nielsen, Yang 1998; Suzuki, Gojobori 1999; Yang et al. 2000), but both approaches may lack power. In the branch test, positive selection is detected on the branch only if ω averaged over all sites is significantly greater than 1, and similarly the site test will detect positive selection only if the ω ratio averaged over all branches on the tree is greater than 1. As a result both tests have generally been superseded by more powerful tests that are designed to detect episodic positive selection that affects only a few amino-acid residues on a few lineages (Yang, Nielsen 2002; Guindon et al. 2004; Yang et al. 2005). The original branch-site test (Yang, Nielsen 2002) was found to generate excessive false positives when model assumptions were violated (Zhang 2004). However, a modified version (Yang et al. 2005) was found to have reasonable power and an acceptable false positive rate under a variety of selection schemes (see Zhang et al. 2005 and below). This modified test has been widely used, for example, to investigate the adaptive evolution of genes underlying schizophrenia (Crespi et al. 2007) and possible positive selection affecting human disease genes (Vamathevan et al. 2008).

Although previous studies noted that different alignment methods may lead to different conclusions concerning detection of positively selected sites (Wong et al. 2008), and that alignment problems as well as poor sequence quality can cause spurious detection of positive selection by the branch-site test (Schneider et al. 2009; Mallick et al. 2010), the effects of insertions, deletions and alignment errors on the branch-site test have not been systematically examined. In this chapter I use the recently-developed simulation program INDELible (chapter 2, Fletcher, Yang 2009) to generate datasets under codon models incorporating indels to examine the performance of the test. The study is an update of Zhang *et al.* (2005). The effect of indels is examined by analysis of the true

alignments and the effect of alignment errors by analysis of alignments generated using alignment programs, including PRANK (Löytynoja, Goldman 2005; Löytynoja, Goldman 2008), MUSCLE (Edgar 2004), MAFFT (Katoh, Toh 2008b) and ClustalW (Larkin et al. 2007).

3.2 Method

3.2.1 The Branch-Site Test of Positive Selection

I refer the reader to the original papers (Yang, Nielsen 2002; Yang et al. 2005) for further details of the branch-site test of positive selection. The model assumes that the branches on the phylogeny are divided a priori into foreground branches where some sites may be under positive selection and background branches where positive selection is absent. The model assumes four site classes (table 3.1). Site class 0 (with proportion p_0) includes codons that evolve under purifying selection on all lineages, with $0 < \omega_0 < 1$. Site class 1 (with proportion p_1) includes codons that evolve neutrally throughout the tree, with $\omega_1 = 1$. Codons in site classes 2a and 2b (with proportion $1 - p_0 - p_1$) are under positive selection on the foreground branches, with $\omega_2 > 1$, but are conserved or neutral on the background branches. The model involves four parameters in the ω distribution that are estimated from the data: p_0, p_1, ω_0 and ω_2. This branch-site model is the alternative hypothesis in the likelihood ratio test (LRT), and the null hypothesis is the same model but with $\omega_2 = 1$ fixed.

Table 3.1: The Branch-Site Model

Site Class	Proportion	Background	Foreground
0	p_0	$0 < \omega_0 < 1$	$0 < \omega_0 < 1$
1	p_1	$\omega_1 = 1$	$\omega_1 = 1$
2a	$(1 - p_0 - p_1)\, p_0 / (p_0 + p_1)$	$0 < \omega_0 < 1$	$\omega_2 \geq 1$
2b	$(1 - p_0 - p_1)\, p_1 / (p_0 + p_1)$	$\omega_1 = 1$	$\omega_2 \geq 1$

NOTE – This model is the alternative hypothesis for the branch-site test of positive selection. The null model is the same except $\omega_2 = 1$ is fixed.

If the null hypothesis is true, twice the difference in log-likelihood between the two models ($2\Delta\ell$) should follow an asymptotic distribution that is a 50:50 mixture of point mass 0 and χ^2_1, with critical values of 2.71 and 5.41 at the 5% and 1% levels, respectively (e.g. Self, Liang 1987). I follow Zhang *et al.* (2005) and use χ^2_1 to conduct the test, with critical values of 3.84 and 5.99. This makes the test more conservative.

If the null hypothesis is rejected, a Bayes empirical Bayes (BEB) approach can be used to calculate the posterior probabilities that each site has evolved under positive selection on the foreground lineages (Yang et al. 2005).

3.2.2 Computer Simulation

INDELible (Fletcher, Yang 2009) was used to generate both the unaligned sequences and the true alignment. For easy comparison I followed Zhang *et al.* (2005) and used the two rooted trees shown in figure 3.1. One branch on the tree is designated the foreground branch while the others are the background branches. The transition/transversion rate ratio is fixed at $\kappa = 4$. Different from Zhang *et al.* (2005), the number of replicates used is 1000 (instead of 200), the root sequence length is 300 codons (instead of 200), and the number of sites in each site class is random instead of being fixed. The stationary codon frequencies are those calculated from the base compositions at the three codon positions in a dataset of five α and β mammalian globin gene sequences (dataset abglobin.nuc in PAML, Yang 2007b).

As in Zhang *et al.* (2005), the simulation model assumes 10 site classes. The background branch always uses selection scheme X, while the foreground branches use one of schemes X, Y, Z, U or V (table 3.2). The ω values for site classes under the different selection schemes are listed in table 3.2 Schemes X, Y, and Z do not allow any sites under positive selection with $\omega > 1$ while schemes U and V do. Scheme X assumes some neutral sites (with $\omega = 1$) and other sites subject to varying degrees of negative selection (with $\omega < 1$). Scheme Y represents a partial relaxation of functional constraints where some sites have higher ω values than in scheme X. In scheme Z all sites have $\omega = 1$,

representing a complete relaxation of functional constraints. This is a very unrealistic scheme for any functional protein, but is included partly because it may cause the test to generate false positives. In scheme U, some sites that experienced purifying selection in scheme X become positively selected, whilst scheme V differs from scheme X in a more complicated manner with some sites having lower ω and some having higher ω.

The molecular clock (rate constancy) holds for the synonymous substitution rate in both trees. In this study, as in Zhang *et al.* (2005), branch lengths are defined as the number of synonymous substitutions per synonymous site (d_S). For example, each branch in tree I represents about 10% of divergence at synonymous sites and, for the background scheme X with average $\omega = 0.5$ (table 3.2), 5% of divergence at nonsynonymous sites.

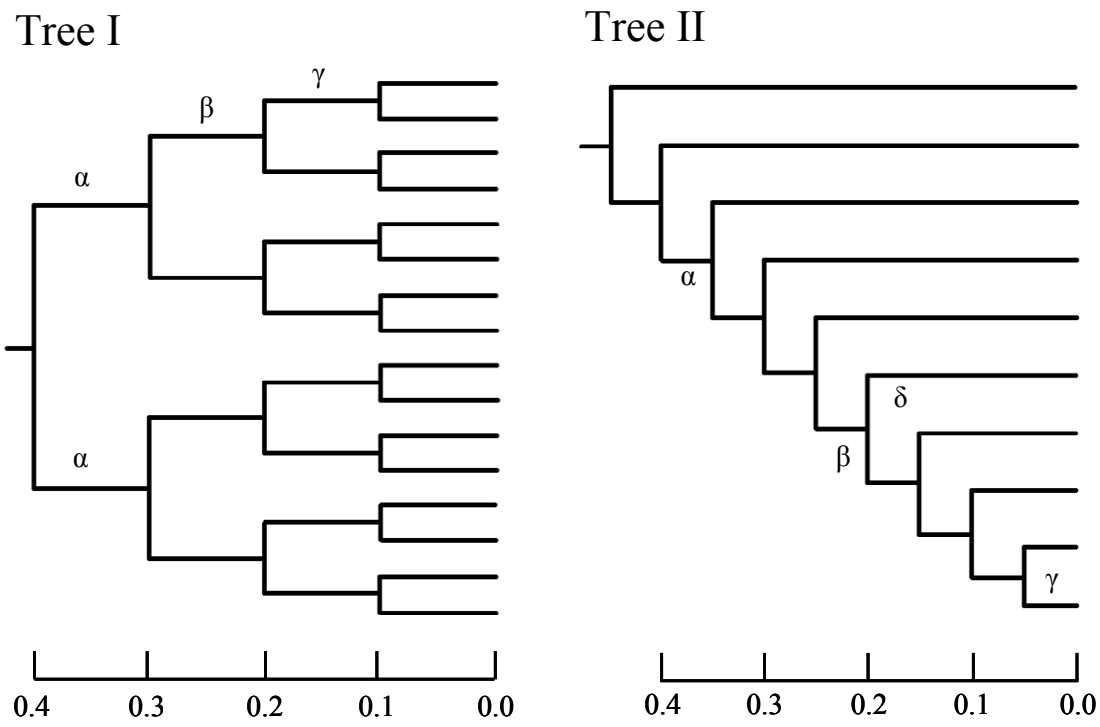


Figure 3.1: Two model trees used in computer simulation

Branch lengths are drawn to scale, in terms of the number of synonymous substitutions per synonymous site. Greek letters indicate the foreground branches used in the simulation.

However, INDELible defines branch lengths as the average number of substitutions per codon (t). They are related approximately by

$$t = 3(Nd_N + Sd_S) = 3d_S(\omega N + S) \approx 3d_S(0.5 \times 0.7 + 0.3) = 1.95d_S$$

(Yang, Nielsen 2000), where N and S are the proportions of nonsynonymous and synonymous sites with $S \approx 0.3$ when $\kappa = 4$ (see fig. 3 of Yang, Nielsen 1998), and $\omega = 0.5$ is the average ω for the background scheme X. Therefore, for example, when a branch length of $d_S = 0.1$ is quoted in this study I have used $t = 0.195$ in INDELible. For simulations that included indels, the rates of insertion and deletion were set to be equal ($\lambda_I = \lambda_D$), and the ratio of substitutions to indels was similar to estimates in the literature, with $\lambda_S/(\lambda_I + \lambda_D) = 10$ (e.g. Ogurtsov et al. 2004). A geometric distribution was used to model insertion and deletion lengths, with parameter $q = 1 - p = 0.35$ chosen as it was deemed an adequate fit to published data on indels for protein coding sequences in mammalian genomes (e.g. Taylor et al. 2004). The mean of this distribution is $1/p = 1.54$ codons, and the standard deviation is $\sqrt{q}/p = 0.91$.

Table 3.2: The ω Values for the Different Selection Schemes Used in Computer Simulation

	Selection Schemes				
Site Class	X	Y	Z	U	V
1	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00	0.70
3	0.80	1.00	1.00	4.00	4.00
4	0.80	0.90	1.00	0.80	0.80
5	0.50	1.00	1.00	2.00	2.00
6	0.50	0.75	1.00	0.50	0.50
7	0.20	1.00	1.00	0.20	0.30
8	0.20	0.60	1.00	0.20	0.20
9	0.00	1.00	1.00	0.00	0.10
10	0.00	0.50	1.00	0.00	0.00
Average ω	0.50	0.88	1.00	0.97	0.96

NOTE – The proportion of each site class is 1/10.

A random DNA sequence of 300 codons was generated at the root of the tree, by sampling from the stationary distribution and from the site classes specified in the ω scheme. The sequence is then "evolved" along the branches of the tree by simulating insertions and deletions, as well as substitutions. To preserve the reading frame, only insertions and deletions of whole codons are allowed. When new codons are inserted they are assigned to one of the 10 site classes at random. INDELible records the insertions and deletions that occurred on the tree, generating the sequences for the tips of the tree as well as the true alignment.

The sequences at the tips of the tree are aligned using the default options of the programs PRANK (version 081202, Löytynoja, Goldman 2005), MAFFT (version 6.716, Katoh, Toh 2008b), MUSCLE (versions 3.7 and 4, Edgar 2004) and ClustalW (version 2.0.11, Larkin et al. 2007). The guide tree is calculated by those programs, and I did not provide the true tree to the alignment program as this option is not often available in real data analysis. To avoid out-of-frame indels, the codon sequences were translated into amino-acid sequences, aligned, and then "back-translated" into codon alignments. In addition I used PRANK's "codon" option that uses the empirical codon model (ECM, Kosiol et al. 2007) to directly align the codon sequences whilst preserving the reading frame. This shall be referred to as PRANK (codon), with PRANK (aa) referring to the amino-acid based alignment.

The aligned codon sequences were analyzed using codeml from the PAML package (Yang 2007b). Besides the estimated alignments, I also analyzed the true alignments. This allowed me to evaluate the impact of insertions and deletions (in the true alignments) separately from that of alignment errors. Alignment gaps are either removed or kept. In the latter case, they are treated as missing data. The correct tree topology and correct identification of the foreground branches was assumed. The branch lengths on the unrooted tree were estimated by maximum likelihood without assuming the molecular clock. Then LRTs were performed at the 5% significance level, as described earlier. Analysis of each dataset was conducted three times, with different initial values used for the numerical optimization to guard against codeml getting stuck in local maxima. In ~98% of datasets all three analyses produced identical log-likelihood values, while in the remaining cases the largest log-likelihood value was used in the LRT.

3.2.3 Measures of Alignment Quality

In order to investigate the effect of alignment errors on the branch-site test I used two measures of alignment accuracy: the total column score (TC) and the sum of pairs score (SPS). TC is the proportion of columns from the true alignment that are reproduced exactly in the test alignment, and SPS is the proportion of aligned codon pairs from the true alignment that are also aligned together in the test alignment (Thompson et al. 1999a). TC is more stringent than SPS. For the true alignment, TC = SPS = 1.

3.3 Results and Discussion

3.3.1 False-Positive Rate of the Branch-Site Test Under Models of Relaxed Constraints

I investigated the false-positive rate of the test when the data were simulated using one of schemes X, Y or Z as the foreground scheme (table 3.2). The background scheme was always X. Thus the data were generated without positive selection but the null hypothesis of the test is violated because the selection scheme is more complex than assumed by the null model. Note that the branch-site model is designed to test whether any residues in the protein experienced positive selection along the foreground branches (with $\omega > 1$), and is not intended to infer the detailed selection scheme on every branch or to estimate the ω ratio for every site and every branch. The latter task is hardly achievable due to lack of information in typical datasets (c.f. Guindon *et al.* 2004).

I analyzed the data first with alignment gaps treated as missing data, and then with gaps removed. The results are presented in table 3.3. The first set of analyses (column headed "No indels") was conducted on datasets generated without indels in order to establish a baseline by which I could evaluate the effects of indels and alignment errors. The false positive rate ranged from 0-4%, all below the nominal 5%. The results are similar to those of Zhang *et al.* (2005), even though there are differences in the two simulation experiments (in the number of codons, in codon usage frequencies and in the use of fixed vs. random number of sites in each site class). A second set of analyses was conducted on the true alignments of data generated with indels (column headed "True Alignment"). The false positive rate was also low, at 0-4%. Next I analyzed the estimated alignments. The false

Table 3.3:

Frequencies of Cases in Which Positive Selection for Foreground Branches Is Erroneously inferred by the Branch-Site Test (Type I Error)

					Gaps Kept							Gaps Removed						
Tree	FGB[a]	FGS[b]	ZNY 2005[c]	No indels	True Alignment	PRANK (codon)	PRANK (aa)	MAFFT	Muscle v4	Muscle v3.7	ClustalW 2.0.11	True Alignment	PRANK (codon)	PRANK (aa)	MAFFT	Muscle v4	Muscle v3.7	ClustalW 2.0.11
I	α	X[d]	-	-	0.013	0.095	0.251	0.650	0.580	0.728	0.988	0.013	0.078	0.212	0.447	0.372	0.526	0.985
I	α	X	0.015	0.009	0.016	0.115	0.252	0.580	0.501	0.625	0.998	0.014	0.079	0.184	0.367	0.325	0.414	0.973
I	α	X[e]	-	-	0.009	□.130	0.288	0.414	0.410	0.473	0.943	0.006	0.075	0.193	0.275	0.270	0.307	0.851
I	β	X	0.020	0.012	0.016	0.106	0.179	0.325	0.320	0.383	0.841	0.014	0.077	0.144	0.236	0.227	0.247	0.670
I	γ	X	0.000	0.013	0.012	0.078	0.170	0.305	0.392	0.375	0.600	0.007	0.050	0.129	0.208	0.298	0.265	0.408
II	α	X	0.010	0.010	0.008	0.112	0.189	0.192	0.220	0.255	0.540	0.008	0.092	0.156	0.137	0.176	0.186	0.463
II	β	X	0.025	0.003	0.008	0.056	0.131	0.153	0.176	0.155	0.340	0.009	0.051	0.104	0.102	0.127	0.109	0.230
II	γ	X	0.020	0.008	0.007	0.021	0.036	0.075	0.279	0.093	0.118	0.010	0.024	0.029	0.058	0.243	0.077	0.100
II	δ	X	0.020	0.012	0.015	0.115	0.249	0.408	0.508	0.436	0.693	0.013	0.102	0.211	0.333	0.463	0.350	0.613
II	β[f]	X	0.030	0.008	0.015	0.076	0.119	0.149	0.175	0.168	0.340	0.010	0.054	0.094	0.092	0.118	0.107	0.241
II	β[g]	X	0.015	0.005	0.005	0.083	0.128	0.147	0.168	0.155	0.342	0.004	0.066	0.090	0.100	0.115	0.095	0.240
I	α	Y	0.030	0.019	0.018	0.045	0.103	0.407	0.361	0.493	0.999	0.020	0.049	0.096	0.255	0.230	0.319	0.989
I	β	Y	0.010	0.024	0.022	0.046	0.100	0.288	0.297	0.351	0.899	0.028	0.047	0.090	0.176	0.181	0.212	0.690
I	γ	Y	0.025	0.026	0.021	0.071	0.123	0.235	0.331	0.289	0.592	0.028	0.067	0.091	0.185	0.227	0.192	0.385
II	α	Y	0.020	0.016	0.016	0.104	0.254	0.256	0.248	0.302	0.685	0.016	0.078	0.194	0.159	0.176	0.192	0.580
II	β	Y	0.040	0.021	0.018	0.068	0.135	0.169	0.188	0.193	0.433	0.025	0.060	0.098	0.111	0.141	0.110	0.297
II	γ	Y	0.055	0.030	0.029	0.045	0.072	0.096	0.312	0.116	0.154	0.034	0.046	0.060	0.095	0.273	0.109	0.126
II	δ	Y	0.035	0.025	0.019	0.046	0.092	0.197	0.259	0.245	0.523	0.020	0.050	0.091	0.165	0.240	0.200	0.439
I	α	Z	0.025	0.030	0.022	0.033	0.064	0.388	0.322	0.486	0.999	0.036	0.043	0.080	0.246	0.221	0.333	0.978
I	β	Z	0.045	0.039	0.023	0.058	0.119	0.298	0.295	0.329	0.902	0.035	0.072	0.118	0.191	0.187	0.198	0.698
I	γ	Z	0.065	0.031	0.029	0.056	0.112	0.202	0.307	0.298	0.595	0.038	0.054	0.106	0.164	0.215	0.195	0.393
II	α	Z	0.010	0.014	0.011	0.129	0.251	0.236	0.242	0.331	0.702	0.017	0.074	0.199	0.149	0.181	0.213	0.600
II	β	Z	0.040	0.031	0.036	0.089	0.139	0.176	0.222	0.224	0.465	0.029	0.072	0.109	0.112	0.151	0.131	0.288
II	γ	Z	0.075	0.043	0.038	0.045	0.061	0.102	0.309	0.106	0.140	0.056	0.074	0.067	0.105	0.289	0.106	0.125
II	δ	Z	0.050	0.043	0.037	0.045	0.078	0.183	0.219	0.217	0.471	0.040	0.049	0.085	0.160	0.207	0.189	0.397

[a] FGB = Foreground Branch. [b] FGS = Foreground Scheme. The background scheme is X in all cases. See table 3.2 for details of selection schemes X, U and V.
[c] "ZNY 2005" refers to the results from Zhang, Nielsen and Yang (2005). [d e] In these schemes there were insertions only (with no deletions)[d] or deletions only (with no insertions)[e].
[f] Foreground branch β is not under positive selection (scheme X) but the 2 neighbouring internal branches are both under positive selection with scheme V.
[g] Foreground branch β is not under positive selection (scheme X) but all 3 neighbouring branches (2 internal, 1 terminal) are under positive selection with scheme V.

positive rates were 2-13% for PRANK (codon), 4-29% for PRANK (aa), 7-65% for MAFFT, 17-58% for MUSCLE v4, 9-73% for MUSCLE v3.7, and 12-100% for ClustalW. Finally another set of analyses was performed on the same alignments after removing columns that contained gaps (using the cleandata option in codeml). With this approach, the false positive rates were 0-6% for the true alignments, and 2-10% for PRANK (codon), 3-21% for PRANK (aa), 6-45% for MAFFT, 11-46% for MUSCLE v4, 8-53% for MUSCLE v3.7, and 10-99% for ClustalW.

The results indicate that insertions and deletions do not cause the branch-site test to generate excessive false positives if the alignment is correct: the false positive rates were at or below 5% in all cases, and were very similar for the data generated with and without indels. However, false positives were often unacceptably high when the alignments were generated using the alignment programs. PRANK (codon) consistently produces the lowest false positives. This may be because the codon-based ECM model it uses to score substitutions is inherently superior to those used to score amino-acids in the other methods. However, I would be remiss not to mention that it could be, at least in part, that the codon-based ECM model is simply the most appropriate scoring scheme for aligning the sequences generated under the similar codon-based models used in INDELible. At any rate PRANK can still be considered superior since PRANK (aa), which does not share this potential advantage, tends to produce the second lowest false positives. The two versions of MUSCLE, and MAFFT, come in third, fourth and fifth (none being clearly superior), and ClustalW generally performs worst. PRANK was the only method that had any false positive rates below 5%. Compared with the true alignment, PRANK has room for improvement.

I conducted two simulations to investigate the robustness of the branch-site test to more complex variations in the ω ratio across lineages in the tree. I tested whether the branch-site test would be misled to produce false positives, in the presence of indels, when the foreground branch is not under positive selection but is surrounded in the phylogeny by background branches that are under positive selection. As in Zhang *et al*. (2005) I used tree II with foreground branch β and foreground scheme X. All other branches are background branches evolving under scheme X, except for the two internal branches adjacent to branch β which evolved under scheme V (table 3.2). In the second simulation, all three branches neighbouring branch β evolved under scheme V. With the true alignment the false

positive rate was ~0.01 in both simulations. As in Zhang *et al*. (2005), the branch-site test is robust and not misled by positive selection on branches close to the foreground branch of interest, even in the presence of indels and unequal codon frequencies.

Alignment quality is known to be closely related to sequence divergence. To investigate the effect of sequence divergence on the false-positive rate of the test, I kept the indel/substitution rate ratio constant, and proportionally decreased the branch lengths in tree I to generate data at different divergence levels, using foreground branch α and foreground scheme X. Figure 3.2 shows the false positive rate and alignment quality plotted against the sequence divergence, measured by the synonymous branch length d_S (note that all branches in tree I have the same length). The results when

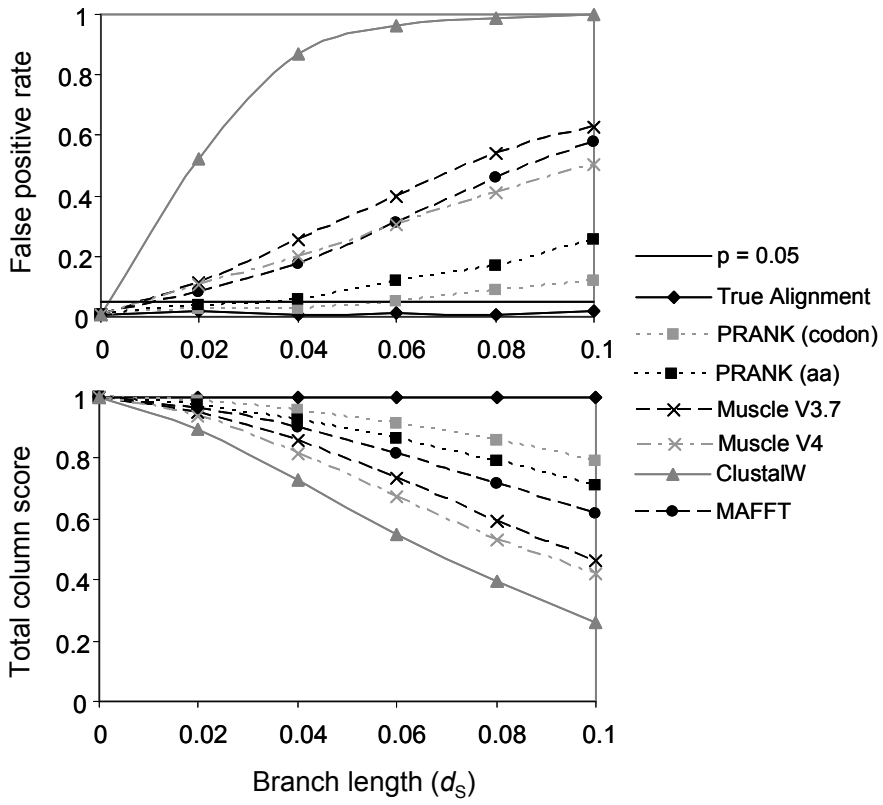


Figure 3.2: False positive rate and alignment accuracy

The false positive rate (top) and alignment accuracy (bottom) for the different alignment methods, plotted against sequence divergence as measured by the synonymous branch length (d_S) on Tree I. Branch α in Tree I was the foreground branch, with scheme X used for all branches. Alignment gaps were treated as missing data.

alignment gaps were removed were very similar and thus not shown. For all alignment methods, the false positive rate decreased and alignment accuracy increased as sequences became less divergent.

I also simulated data with insertions but no deletions and with deletions but no insertions. MUSCLE, MAFFT and ClustalW were found to generate fewer false positives when there were deletions only, but more false positives when there were insertions only (table 3.3). This result is consistent with the observation that a major problem with programs like ClustalW, MAFFT and MUSCLE is that they do not deal with insertions properly, penalizing the same insertion event multiple times during the progressive alignment algorithm, while they deal with deletions more or less appropriately (Löytynoja, Goldman 2005). For PRANK the pattern was the opposite. At any rate, whether there were deletions or insertions only, PRANK (codon) always had the lowest false positives, PRANK (aa) always came second, and ClustalW had the highest false positives, with MAFFT and the two versions of MUSCLE falling in-between.

3.3.2 Power of the Branch-Site Test in Detecting Positive Selection

To test how often the branch-site test correctly identifies positive selection on the foreground branches, I conducted simulations using either scheme U or V for the foreground branch. Scheme X is always used for the background branches. The results are shown in table 3.4. Again, use of the true alignments for data with indels produced very similar results to simulations without indels. For tree I, the power was 5-11% for data generated without indels, and 4-11% for data with indels (table 3.4). Similarly, in tree II the power was 1-32% without indels, and 1-33% with indels. Whilst these are relatively low detection rates, it should be noted that the average ω across all sites on the foreground branch under both schemes U and V is <1, and the average ω over all branches on the tree is never greater than 1 for any site class.

Next I analyzed the alignments generated with the six alignment methods. The power was 10-67% for PRANK (codon), 12-85% for PRANK (aa), 17-95% for MAFFT, 21-93% for MUSCLE v4, 20-94% for MUSCLE v3.7, and 22-100% for ClustalW. If the gaps were removed, the power became 1-26% for the true alignments, 6-49% for PRANK (codon), 10-74% for PRANK (aa), 12-85% for

Table 3.4:

Frequencies of Cases in Which Positive Selection for Foreground Branches Is Correctly Inferred by the Branch-Site Test (Power)

					Gaps Kept							Gaps Removed						
Tree	FGB[a]	FGS[b]	ZNY 2005[c]	No indels	True Alignment	PRANK (codon)	PRANK (aa)	MAFFT	Muscle v4	Muscle v3.7	ClustalW 2.0.11	True Alignment	PRANK (codon)	PRANK (aa)	MAFFT	Muscle v4	Muscle v3.7	ClustalW 2.0.11
I	α (0.2)	U	0.145	0.106	0.114	0.245	0.419	0.767	0.723	0.823	0.999	0.089	0.186	0.336	0.534	0.494	0.593	0.988
I	β (0.1)	U	0.110	0.055	0.042	0.146	0.270	0.451	0.441	0.492	0.892	0.042	0.107	0.189	0.289	0.269	0.300	0.718
I	γ (0.1)	U	0.095	0.054	0.061	0.165	0.285	0.452	0.538	0.523	0.750	0.058	0.128	0.225	0.317	0.376	0.354	0.525
II	α (0.05)	U	0.025	0.016	0.015	0.139	0.203	0.227	0.238	0.275	0.578	0.014	0.097	0.152	0.167	0.177	0.196	0.496
II	β (0.05)	U	0.040	0.026	0.022	0.111	0.171	0.189	0.208	0.197	0.394	0.024	0.081	0.135	0.117	0.156	0.120	0.295
II	γ (0.05)	U	0.095	0.062	0.065	0.102	0.126	0.175	0.379	0.202	0.231	0.049	0.064	0.101	0.138	0.311	0.163	0.171
II	δ (0.2)	U	0.220	0.182	0.188	0.359	0.530	0.659	0.749	0.721	0.871	0.134	0.257	0.427	0.556	0.662	0.600	0.783
II	β (0.15)	U	0.130	0.077	0.079	0.279	0.421	0.567	0.557	0.535	0.787	0.056	0.185	0.339	0.373	0.404	0.326	0.638
II	β (0.45)	U	0.180	0.192	0.149	0.670	0.851	0.946	0.925	0.937	0.997	0.124	0.492	0.744	0.845	0.816	0.844	0.990
II	γ and δ	U	0.330	0.309	0.331	0.471	0.590	0.681	0.774	0.714	0.864	0.258	0.323	0.455	0.555	0.680	0.566	0.753
I	α (0.2)	V	0.115	0.105	0.102	0.277	0.435	0.777	0.711	0.817	0.998	0.076	0.196	0.331	0.530	0.481	0.574	0.983
I	β (0.1)	V	0.075	0.063	0.052	0.165	0.292	0.486	0.470	0.516	0.918	0.046	0.109	0.217	0.299	0.288	0.320	0.732
I	γ (0.1)	V	0.075	0.071	0.066	0.181	0.270	0.433	0.530	0.504	0.731	0.055	0.112	0.180	0.297	0.381	0.337	0.515
II	α (0.05)	V	0.055	0.013	0.014	0.138	0.224	0.235	0.266	0.288	0.575	0.012	0.093	0.182	0.167	0.189	0.196	0.498
II	β (0.05)	V	0.035	0.027	0.031	0.101	0.164	0.200	0.241	0.204	0.417	0.028	0.072	0.122	0.122	0.162	0.116	0.292
II	γ (0.05)	V	0.120	0.062	0.079	0.100	0.123	0.171	0.396	0.208	0.222	0.066	0.082	0.098	0.131	0.329	0.148	0.171
II	δ (0.2)	V	0.300	0.174	0.181	0.364	0.527	0.676	0.761	0.733	0.896	0.139	0.261	0.429	0.569	0.697	0.612	0.809
II	β (0.15)	V	0.110	0.077	0.067	0.260	0.437	0.535	0.568	0.540	0.776	0.055	0.197	0.320	0.361	0.413	0.341	0.619
II	β (0.45)	V	0.190	0.154	0.147	0.615	0.831	0.949	0.928	0.936	0.997	0.119	0.462	0.725	0.821	0.808	0.803	0.985
II	γ and δ	V	0.435	0.321	0.298	0.441	0.547	0.638	0.764	0.687	0.831	0.224	0.307	0.410	0.520	0.677	0.540	0.733
II	β[d] (0.15)	V	0.195	0.159	0.169	0.619	0.800	0.902	0.878	0.867	0.963	0.114	0.427	0.694	0.748	0.745	0.677	0.902
II	β[e] (0.15)	V	0.500	0.226	0.251	0.438	0.588	0.672	0.687	0.657	0.877	0.173	0.311	0.468	0.487	0.498	0.458	0.729

[a] FGB = Foreground Branch. The length of the foreground branch (the number of synonymous substitutions per synonymous site) is shown in parentheses (see figure 3.1).
[b] FGS = Foreground Scheme. The background scheme is X in all cases. See table 3.2 for details of selection schemes X, U and V.
[c] "ZNY 2005" refers to the results from Zhang, Nielsen and Yang (2005).
[d] In this case the root sequence was 900 codons in length instead of 300.
[e] In this case ω ratios were doubled for classes 3 and 5 of scheme V.

MAFFT, 16-82% for MUSCLE v4, 12-84% for MUSCLE v3.7, and 17-99% for ClustalW. Thus methods that had high false positives when there was no positive selection also had high true positives when there was positive selection. The conflicts between the accuracy and power of the test are considered in the next section, but in the rest of this section, I mainly focus on the true alignments and PRANK alignments.

I investigated the effect of sequence divergence on the power of the test. This was done in the same manner as in figure 3.2, except that I now use scheme U in place of scheme X for the foreground branch. The results are shown in figure 3.3. Power decreased and alignment accuracy increased as sequences became less divergent.

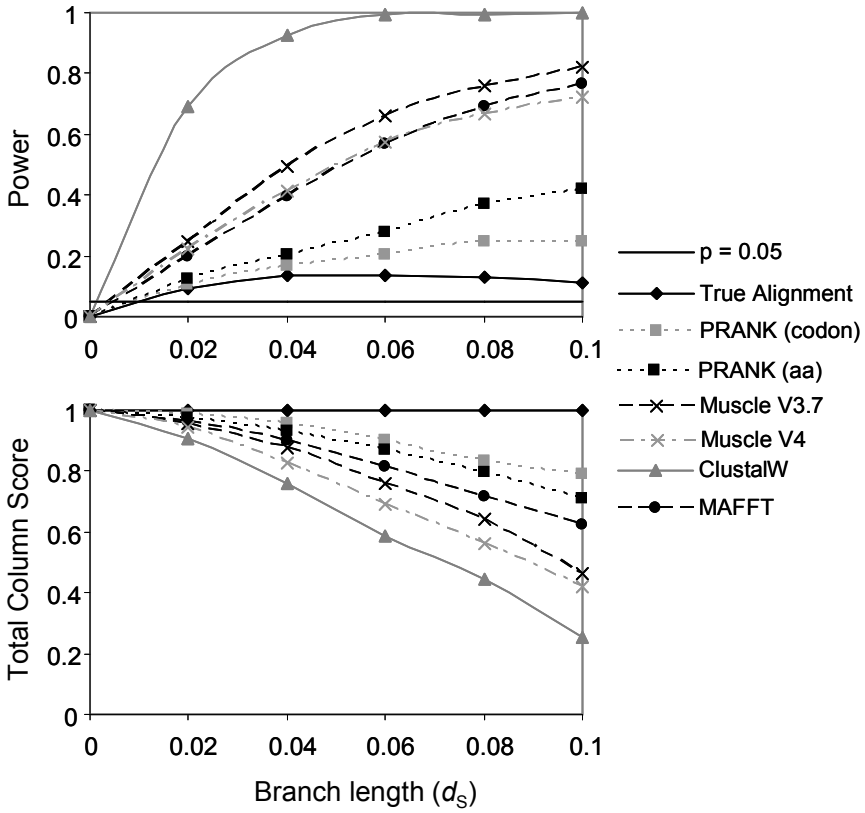


Figure 3.3: Power and alignment accuracy

The power (top) and alignment accuracy (bottom) for the different alignment methods, plotted against sequence divergence as measured by the synonymous branch length (d_s) on Tree I. Foreground branch α and foreground scheme U were used on Tree I. Alignment gaps were treated as missing data.

It may be expected that the method should be able to infer recent substitutions more reliably than ancient ones, and therefore it should be harder to detect positive selection on branches deeper in the phylogeny. This intuition appears to be correct for both schemes on both trees -- in tree I the power is higher for branch γ than for branch β, and in tree II the power is higher for branch γ than for branch β, and higher for branch β than for branch α. However, the length of the foreground branch had a much greater effect -- the branch-site test had greater power for branch α, the deepest branch in tree I, than for either of the shorter and more recent branches β and γ, and in tree II the test had much greater power for the longer branch δ than for any of the shorter branches α, β, and γ.

To investigate the effect of foreground branch length, I performed further simulations with the length of branch β in tree II increased from 0.05 to 0.15, or 0.45. Under these conditions the power of the test increased by twofold to sevenfold over the different alignments. When the foreground branch becomes too long I would expect the power to decrease because of saturation of substitutions. Furthermore, I expect the power to increase when the same sites are under positive selection on several branches. To test this I applied the same selection scheme to branches γ and δ on tree II, and identified both as foreground branches when running codeml. Power was substantially higher than when γ or δ alone was the foreground branch (table 3.4) for both the true alignment and the PRANK (codon) alignment.

I also expect the power to be higher if the positive selection is stronger (with higher ω ratios) or if more sites are under positive selection. This was indeed the case. I conducted two sets of simulations using tree II, foreground branch β (with length 0.15), and foreground scheme V, to examine such effects. In the first set, I increased the ω ratios for site classes 3 and 5 from 2 and 4 to 4 and 8 respectively (table 3.2). For the true alignment power increased by more than threefold, while for the estimated alignments, the increase was 13-68%. In the second set of simulations I increased the length of the root sequence from 300 to 900 codons, with 3 times as much data as before. For the true alignment power increased more than twofold, with similar increases for PRANK.

3.3.3 Which Alignment Method is Best for Detecting Positive Selection?

A commonly used procedure for evaluating a test is the ROC (Receiver Operating Characteristic) plot. An example is shown in figure 3.4 for simulations using branch α on tree I as the foreground branch. Scheme X was used for the foreground to calculate the false positive rate and U was used to calculate the true positive rate. Schemes X and U differ only in the positively selected site classes, and can thus be used for a fair comparison. ROC curves that bulge towards the top left corner indicate a good predictor. It has been suggested that the area under the ROC curve (AUC) can be used as a summary to measure the performance of a method (Ling et al. 2003). Note that AUC = 0.5 for a random guess and =1 for a "perfect" method that makes no mistakes. I then calculated the AUC for each alignment method and foreground branch combination using the trapezoidal method (Hanley, McNeil 1983).

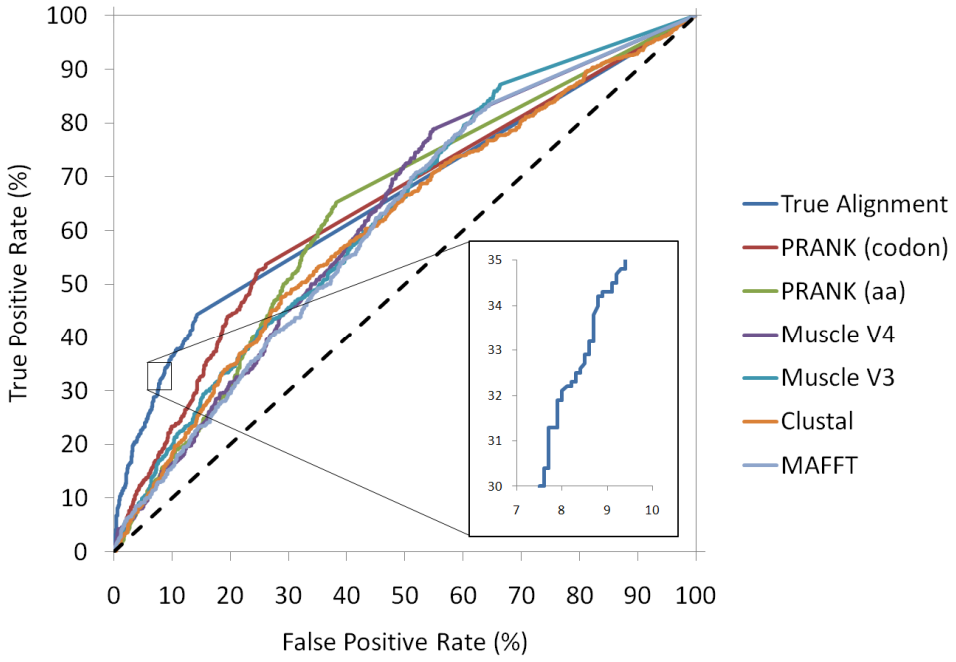


Figure 3.4: Example ROC curves

ROC curves for the the six alignment methods, and the true alignment, when using foreground branch α and tree I. The true positive rate is generated under foreground scheme U, and the false positive rate is generated under foreground scheme X. The box shows a section of the curve for the true alignment. No graphical smoothing has been used. Each individual segment of the curve is a very small vertical (or horizontal) line signifying the branch-site test has correctly (or incorrectly) deemed that positive selection occurred in one further simulated dataset as the significance level of the test is increased.

The results are shown in table 3.5 for analyses where gaps were kept and in table 3.6 for analyses where gaps were removed.

For the true alignment, AUC was smaller when gaps were removed than when they were kept, regardless of the tree or foreground branch. This is consistent with the expectation that given the correctness of the alignment, removing gap columns amounts to reducing the amount of data available. For the estimated alignments, one aim of removing gaps is to reduce the alignment errors

Table 3.5:

AUC values for the ROC analysis (Gaps Removed)

Tree	FGB[a]	True Alignment	PRANK (codon)	PRANK (aa)	MAFFT	Muscle v4	Muscle v3.7	ClustalW 2.0.11
I	α	0.651	0.634	0.641	0.605	0.615	0.617	0.583
I	β	0.593	0.590	0.566	0.551	0.547	0.554	0.521
I	γ	0.625	0.622	0.617	0.600	0.573	0.584	0.579
I	C	0.623	0.615	0.608	0.584	0.578	0.582	0.541
II	α	0.539	0.532	0.518	0.525	0.520	0.511	0.524
II	β	0.551	0.567	0.556	0.554	0.543	0.538	0.546
II	γ	0.646	0.646	0.660	0.666	0.601	0.648	0.645
II	δ	0.699	0.670	0.678	0.641	0.620	0.657	0.606
II	C	0.609	0.602	0.562	0.593	0.571	0.587	0.570

[a] FGB = Foreground Branch. [b] C = Combined dataset of all foreground branches on the tree.

Table 3.6:

AUC values for the ROC analysis (Gaps Kept)

Tree	FGB[a]	True Alignment	PRANK (codon)	PRANK (aa)	MAFFT	Muscle v4	Muscle v3.7	ClustalW 2.0.11
I	α	0.658	0.642	0.632	0.614	0.625	0.630	0.609
I	β	0.599	0.584	0.583	0.570	0.575	0.570	0.532
I	γ	0.643	0.631	0.627	0.604	0.592	0.600	0.577
I	C	0.633	0.619	0.614	0.593	0.596	0.596	0.545
II	α	0.541	0.544	0.523	0.527	0.515	0.514	0.525
II	β	0.569	0.575	0.566	0.551	0.544	0.547	0.531
II	γ	0.666	0.692	0.689	0.680	0.618	0.670	0.666
II	δ	0.746	0.726	0.701	0.654	0.634	0.670	0.616
II	C	0.630	0.631	0.613	0.598	0.578	0.595	0.570

[a] FGB = Foreground Branch. [b] C = Combined dataset of all foreground branches on the tree.

and the false positives of the test. However, it had this effect in only 6 out of the 42 combinations of alignment method, tree, and foreground branch (twice for ClustalW, and once for each of Muscle v4, MAFFT and the two PRANK variations). Removing gaps before applying the branch-site test was thus ineffective in reducing alignment errors or false positives.

I thus focus on table 3.6, where gaps were kept. For tree I the average AUC values were 0.633 for the true alignment, 0.619 for PRANK (codon), 0.614 for PRANK (aa), 0.596 for the two versions of Muscle, 0.593 for MAFFT, and 0.545 for ClustalW. For tree II they were 0.630 for the true alignment, 0.631 for PRANK (codon), 0.613 for PRANK (aa), 0.598 for MAFFT, 0.595 for Muscle v3.7, 0.578 for Muscle v4 and 0.570 for ClustalW. These values are shown in figure 3.5 (for tree I) and figure 3.6 (for tree II) with binomial exact 95% confidence intervals (as vertical bars). Also the standard errors, calculated according to the method of (DeLong et al. 1988), are shown below (in the bar charts). The AUC values thus indicated that PRANK (codon) was the superior alignment method among those tested, followed by PRANK (aa), MAFFT and Muscle, with ClustalW to be the poorest.

I consider the false detection of positive selection (false positive) to be a more serious error than a failure to detect positive selection (false negative), as this may lead to wasted time, effort, and funding being devoted to downstream research. Therefore, I suggest that a test with excessive false positives (with rate >20%, say) be avoided in real data analysis. With this viewpoint, the differences among the methods look even greater than suggested by the AUC values. For example, only PRANK had the false positive rate under control in some datasets (table 3.3), but even PRANK leaves room for improvement. The rankings of the alignment methods are nevertheless the same whether I use the false positives or the AUC values. It should be noted that the validity of AUC has come into question, with some claiming it to be misleading measure (Lobo et al. 2008) because, for example, it summarises the test performance over regions of the ROC space in which one would rarely operate and it weights false negatives in the same way as false positives. However, if I was to give greater weight to certain regions of the ROC curves, to favour avoidance of false positives, the AUC value differences would become more, not less, exaggerated and the same would also happen if we were to employ partial ROC curves (Walter 2005), in the region with false positives < 20% say. Therefore, I feel the use of AUC in this study is justified and the conclusions are sound.

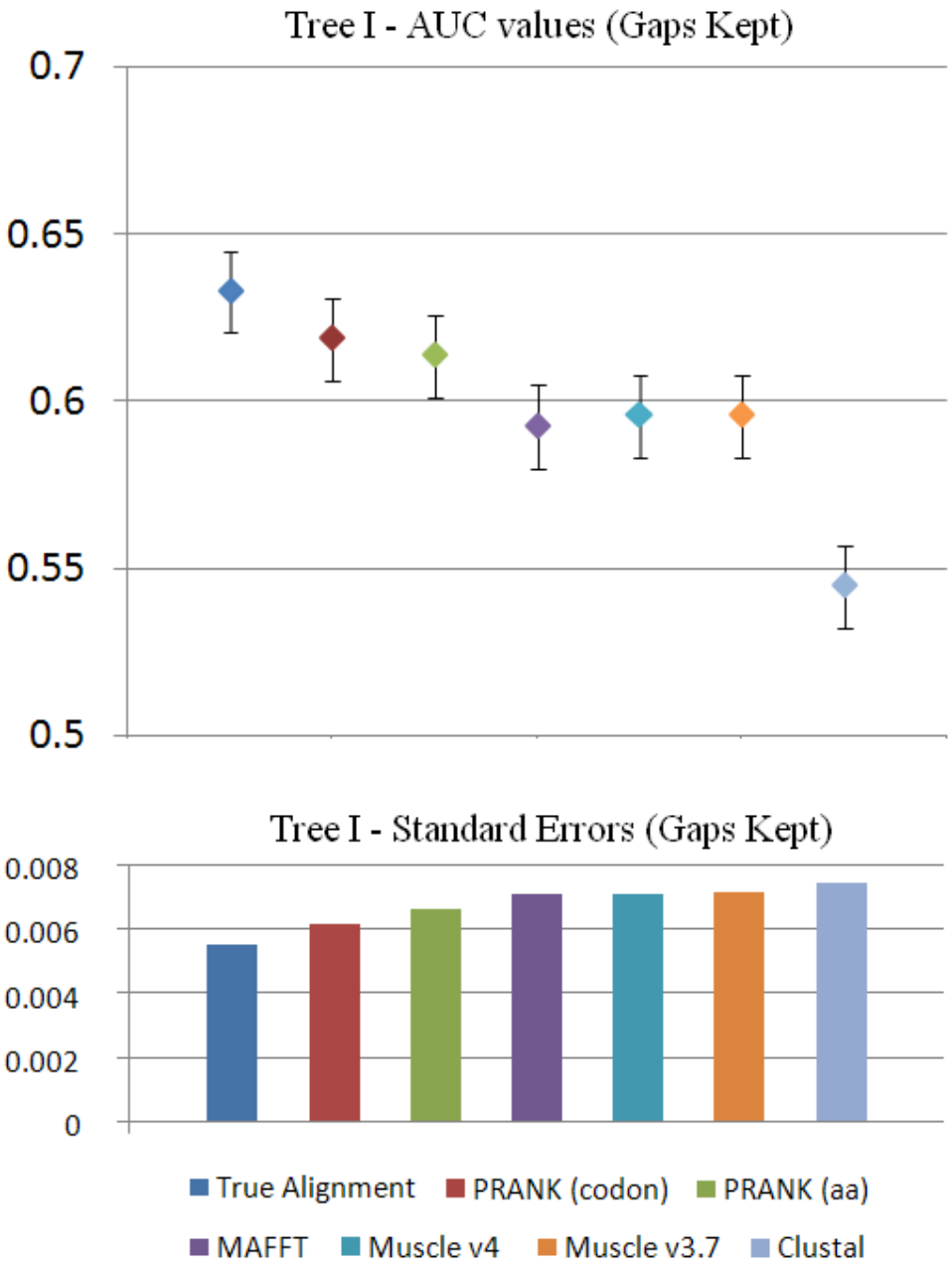


Figure 3.5: AUC values on Tree I when gaps are kept

Average AUC values for each alignment method (and the true alignment) on Tree I. The vertical capped lines are binomial eaxact 95% confidence intervals. The vertical bars below the scatter plot are the standard errors for each alignment.

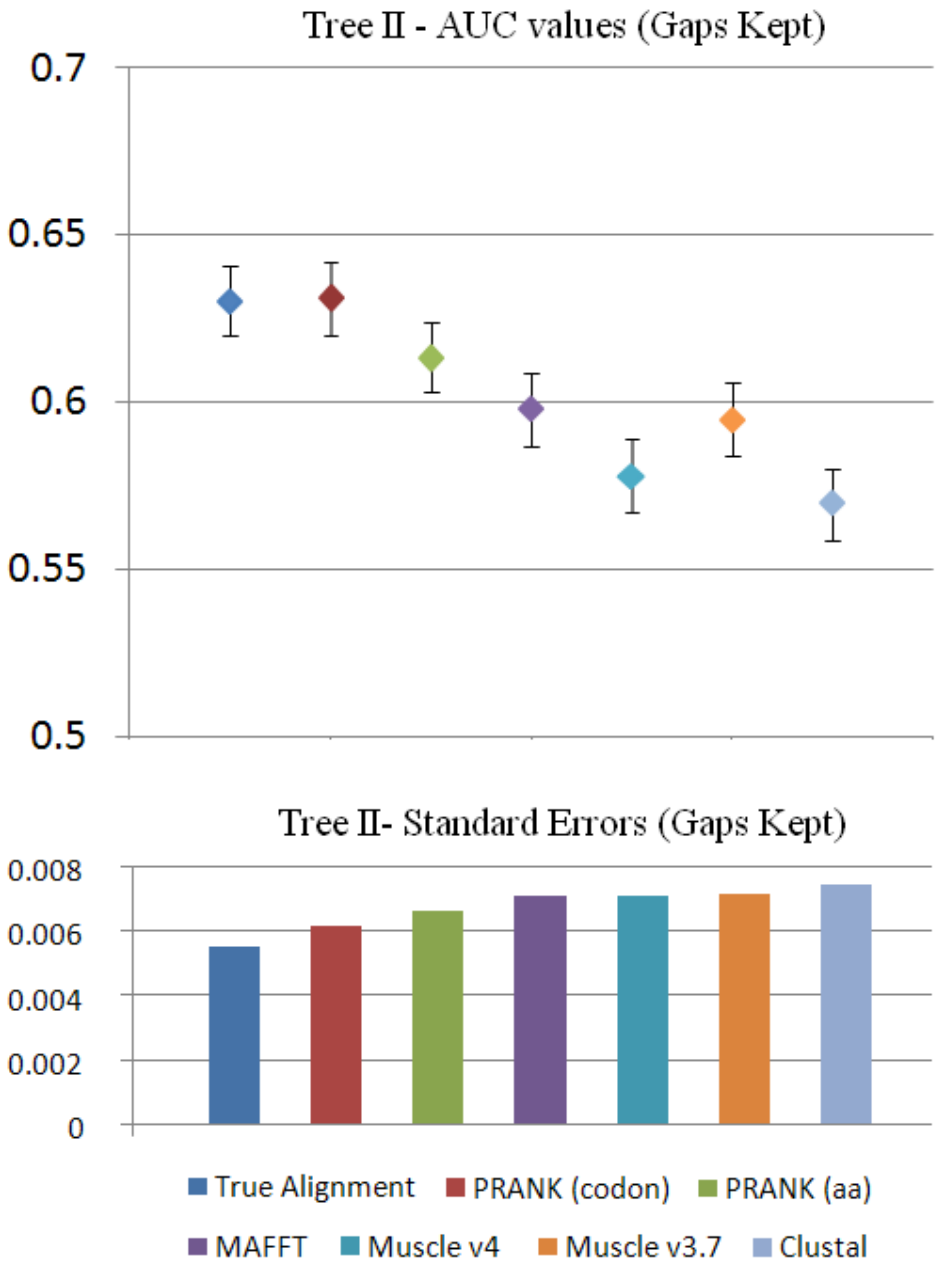


Figure 3.6: AUC Values on Tree II when gaps are kept

Average AUC values for each alignment method (and the true alignment) on Tree II. The vertical capped lines are binomial exact 95% confidence intervals. The vertical bars below the scatter plot are the standard errors for each alignment.

3.3.4 Performance of BEB in Identifying Positively Selected Sites

When the branch-site test of positive selection is significant the BEB procedure (Yang et al. 2005) can be used to calculate the posterior probability that a particular codon belongs to the class of positive selection. A codon with a high posterior probability is likely to have been evolving under positive selection on the foreground branches. I calculated the average frequency at which a codon was identified as being under positive selection using 95% or 99% cut-offs. I analyzed only the true alignments because alignment errors cause sites from different classes to be aligned together, making the calculations difficult. I only performed the BEB analysis on datasets in which the LRT was significant at the 5% level.

If BEB is conservative when evaluated under the Frequentist criterion (see Yang et al. 2005) one would expect the false positives to occur less than 5% of the time at the 95% cut-off, and less than 1% of the time at the 99% cut-off. This was found to be true for datasets generated with indels, and with no positive selection (table 3.3), in 45 of 46 cases.

For the 23 cases where datasets were generated without indels, and under selection schemes X, Y or Z, the false positive rates were sometimes higher – at 6-7% for 2 cases at the 95% cut-off, and 1-2% for 5 cases at the 99% cut-off. For all simulations with positive selection (table 3.4) the false positive rate was very low (<0.1% at both cut-offs). However the power of BEB in detecting positively selected codons was also very low, at ≤1% in all but one case.

3.3.5 Alignment Accuracy

The selection schemes and root sequence length had little effect on alignment accuracy. Thus I averaged over the simulation conditions and present the results for each tree (table 3.7). The alignment accuracy is in the order PRANK (codon) > PRANK (aa) > MUSCLE v4 & MUSCLE v3.7 & MAFFT > ClustalW (table 3.9). This is the case for both trees and for all different simulation conditions (results not shown). This ranking was observed in ~100% (TC) and 88% (SPS) of the replicates for tree I, and 87% (TC) and 43% (SPS) of the replicates for tree II. PRANK (codon) was clearly the best among the alignment methods examined here on most datasets. MAFFT was better

than MUSCLE in most cases, but the relative performance of the two versions of MUSCLE was less clear. On average, TC judged MUSCLE v3.7 as better on tree I but worse on tree II, whilst SPS scored the two methods very similarly for both trees (table 3.9). It is noted that MUSCLE v4 is experimental. Overall, the order of alignment accuracy is exactly the opposite of the order for the false positive rate discussed before.

To understand the nature of the alignment errors, I simulated datasets similarly to figure 3.3, but kept the substitution rate λ_S constant while increasing the indel rate $\lambda_I + \lambda_D$ (with $\lambda_I = \lambda_D$). I then calculated the average alignment length and the average number of distinct codons in a column after removal of columns with gaps. The programs MAFFT, MUSCLE and ClustalW produced much shorter alignments than the true alignments, while PRANK alignments were of a similar length (results not shown). The number of distinct codons in a column is shown in figure 3.7. This ranges from 1 to 16 for my data with 16 sequences. For the true alignment, the number remained roughly constant regardless of the indel rates. For PRANK (codon) and PRANK (aa), this number is similar to that for the true alignment. For MAFFT, MUSCLE and ClustalW, this number is much greater, especially at high indel rates. Those results are consistent with the observation of Loytynoja and Goldman (2005) that the main problem with those poor alignment methods is that they place nonhomologous codons (amino acids) into the same column. As such alignment errors remain after

Table 3.7: Alignment Accuracy for Different Alignment Methods

	Average accuracies				Insertions or Deletions only			
	Total Column Score (TC)		Sum of Pairs Score (SPS)		Insertions (Tree I)		Deletions (Tree I)	
	Tree I	Tree II	Tree I	Tree II	TC	SPS	TC	SPS
True Alignment	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
PRANK (codon)	0.79	0.63	0.95	0.91	0.83	0.96	0.73	0.94
PRANK (aa)	0.71	0.54	0.94	0.90	0.75	0.94	0.64	0.93
MAFFT	0.56	0.47	0.91	0.89	0.64	0.89	0.61	0.91
Muscle v4	0.42	0.44	0.88	0.86	0.42	0.87	0.50	0.90
Muscle v3.7	0.46	0.40	0.88	0.88	0.46	0.89	0.41	0.89
ClustalW 2.0.11	0.25	0.22	0.71	0.75	0.21	0.67	0.35	0.80

NOTE.— Simulations where root sequence length, branch lengths, or insertion/deletion rates were changed are excluded from the average accuracies.

gaps are removed, the strategy of removing gaps to reduce the false positive rate of the test is ineffective (table 3.2). The nature of the alignment errors as discussed above suggests that the site models (Nielsen, Yang 1998; Yang et al. 2000) may be similarly affected by alignment errors, although site models are not examined in this study. The impact of the alignment errors on the branch model (Yang 1998) appears to be more complex and may depend on the location of the foreground branch in the tree or whether errors are introduced when sequences on one side of the foreground branch are aligned against sequences on the other side.

The empirical codon model (ECM, Kosiol et al. 2007) underlying PRANK (codon) was derived from the PANDIT database (Whelan et al. 2006), which has an average ω of 0.192 (Kosiol et al. 2007). Thus PRANK (codon) should be more successful at aligning codons under selective constraint than those under positive selection. Similar bias may be expected for Prank (aa), MAFFT, MUSCLE and ClustalW, as they use empirical amino acid substitution/exchange matrices derived from large databases dominated by purifying selection.

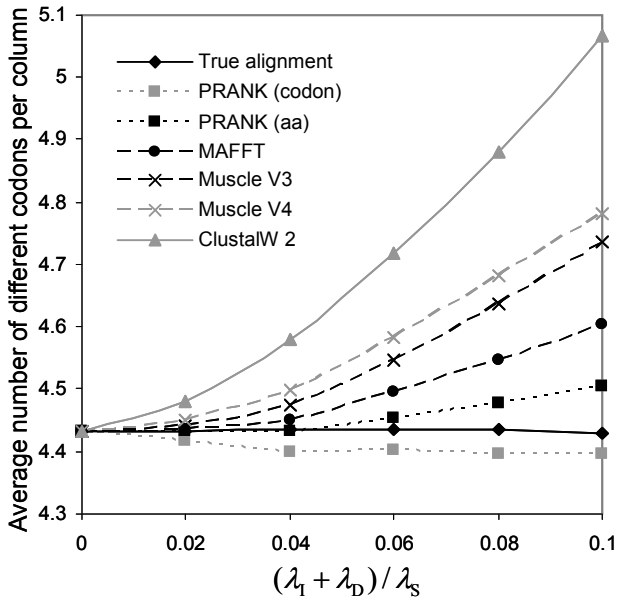


Figure 3.7: Average number of different codons per column for different alignment methods

The number of distinct codons per alignment column plotted against the indel/substitution rate ratio for simulations when the substitution rate λ_S was fixed. Tree I was used with scheme X for both foreground and background branches. Alignment gaps were removed.

Furthermore, conserved amino acids correspond to less variable codons which are easier to align. In sum, codons under positive selection or under weak constraint are expected to be most prone to alignment errors. This prediction was found to be true for all six alignment methods. For example, figure 3.8 shows the alignment accuracy for codons in different site classes of scheme X for tree I for PRANK (codon). Codons in site classes with lower ω ratios were aligned more accurately. The background ω ratios had far greater effects on alignment quality than the foreground ω ratios. However, for a given site class with the same background ω ratio, alignment quality was slightly better for lower foreground ω ratios.

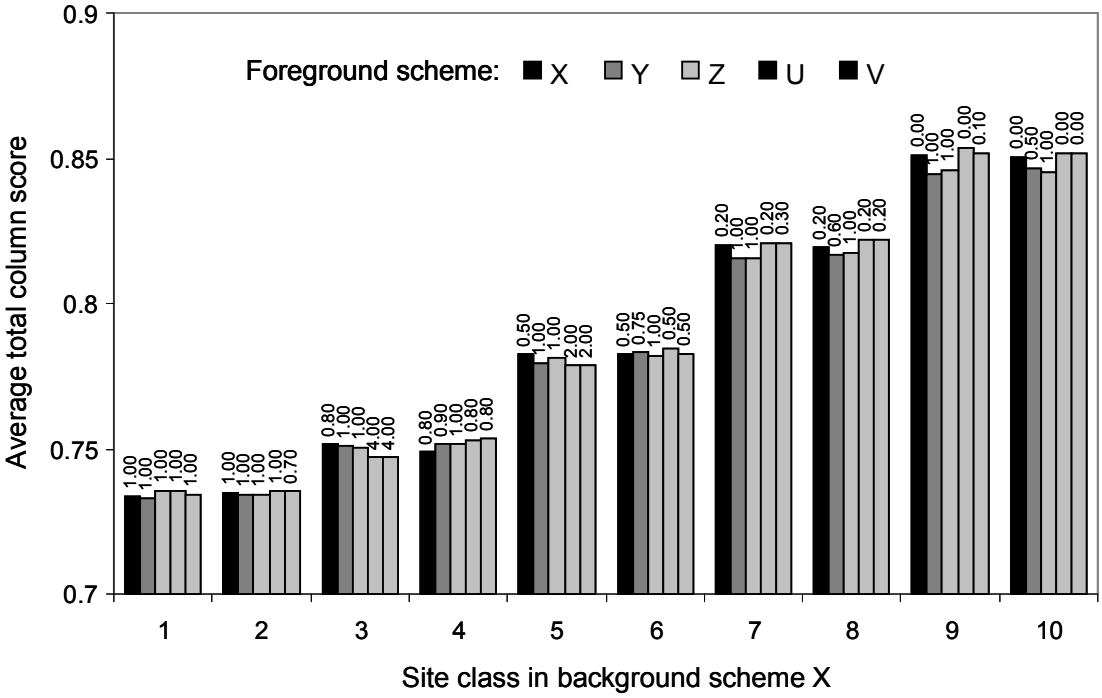


Figure 3.8: Average alignment accuracy in different site classes for PRANK (codon)

Average alignment accuracy for PRANK (codon) for codons in the ten site classes of background scheme X and for different foreground ω ratios. Tree I was used. The number above each column is the foreground ω ratio for that site class. Other alignment programs showed a similar pattern.

Recently Hall (2008b) suggested that a measure of "consistency" known as the Heads-or-Tails (HoT) score (Landan, Graur 2007) had a direct correlation with alignment accuracy, and that it can be used to choose between alignments produced by different methods. The HoT score is the proportion of columns shared between the "Heads" alignment, generated from the original sequences, and the "Tails" alignment, generated from the reversed sequences. On my data the HoT score chose MUSCLE v4 as the best method 93% of the time, while TC consistently favored PRANK (codon). This discrepancy appears to be due to the fact that PRANK breaks ties at random, while MUSCLE v4 makes the same choices so that alignment errors are consistent. I do not recommend the HoT score as a measure of alignment quality.

3.3.6 Implications for Past Studies of Positive Selection

What levels of sequence divergences may cause serious alignment errors and false detection of positive selection? To get a rough idea about this question, I examined two recent studies of positive selection using the branch-site test, one using five mammalian species (human, chimp, dog, mouse and rat) (Vamathevan et al. 2008), and the other using a broader range of vertebrate species (five fishes, the *Xenopus* frog, the chicken, and at least four mammals) (Studer et al. 2008). Both studies used MUSCLE v3 to construct the alignments and both removed columns with gaps before applying the branch-site test. While Studer *et al.* re-aligned some of their genes using MAFFT and obtained highly similar results, I note that consistency between MAFFT and MUSCLE was not a good indication for high alignment quality in my simulations.

The MUSCLE alignments for the 3,081 mammalian genes were provided by Jessica Vamathevan (pers. comm.). The gene sequences were re-aligned using PRANK (codon), and both sets of alignments were analyzed using the branch-site test, with the human or chimpanzee lineages designated as the foreground branch. Following Vamathevan *et al.* (2008), I used the 5% significance level and the Bonferonni correction for multiple testing (Anisimova, Yang 2007). The initial analysis of Vamathevan *et al.* identified 69 (2.2%) and 354 (11.5%) genes under positive selection on the human and chimpanzee lineages, respectively (Vamathevan 2008: table 3.1). The counts from my re-

analysis of the same data were nearly identical, at 70 and 355. My analysis of the PRANK (codon) alignments produced 33 (1.1%) and 83 (2.7%) genes for the human and chimpanzee branches, respectively, much smaller than for the MUSCLE alignments, indicating that the original MUSCLE alignments may involve substantial alignment errors. Problems with the MUSCLE alignments were noted by Vamathevan *et al.*, who applied a manual curation step and reported 54 (1.8%) and 162 (5.3%) positively selected genes for the human and chimpanzee branches, respectively (Vamathevan 2008, table 1). These counts are much smaller than from the original MUSCLE alignments but are still much higher than those from the PRANK (codon) alignments.

The MUSCLE alignments for 767 vertebrate genes were downloaded from http://bioinfo.unil.ch/supdata/Singleton.html (Studer et al. 2008). Both the original alignments and the PRANK (codon) re-alignments were analyzed using the branch-site test, with three foreground branches considered: the mammal lineage, the euteleosts lineage, or the boney vertebrate lineage. Studer *et al*. (2008) used the 1% significance level and identified 8%, 25%, and 31% of genes to be under positive selection along the three branches. My re-analysis of the same original alignments produced 8%, 24% and 30%. The counts were 6%, 16% and 18% from analysis of the PRANK (codon) alignments.

Those comparisons suggest that mammalian and vertebrate gene sequences are divergent enough for the impact of alignment errors on the branch-site test to be a real concern. This conclusion is consistent with the results of Schneider *et al.* (2009) and Mallick *et al.* (2010). Many past studies detecting positive selection in divergent genes may benefit from a re-analysis using alignments generated from a more reliable method such as PRANK (codon).

3.4 Conclusion

In this study I investigated the accuracy and robustness of the branch-site test in the presence of insertions, deletions and alignment errors. My results obtained from analyses of the true alignments suggested that indels have little effect on the performance of the branch-site test. In the presence of indels the test is still robust to violation of model assumptions such as the existence of more than three site classes, more than one site class evolving under positive selection, or more than one kind of

background branch. The BEB method for detecting positively selected sites was noted to have low false positives under such conditions.

This study has highlighted the importance of alignment quality to detection of positive selection using the branch-site test. In particular most current alignment programs tend to place nonhomologous codons (or amino-acids) in the same column, misleading the test into claiming excessive amino acid changes at those sites. Removing alignment gaps helped to reduce false positives only slightly. I found that PRANK was superior to MAFFT, MUSCLE and ClustalW. In particular, PRANK (codon) produced the most accurate alignments, with the lowest false positive rates. Nevertheless, even PRANK (codon) does not have the false positive rate under control. It is hard to imagine tests of positive selection that are tolerant of gross alignment errors, and I suggest that it may be profitable to try and improve current alignment algorithms.

Chapter 4

The Effect of Alignment Accuracy on Methods of Phylogeny Reconstruction

4.1 Introduction

Phylogeny reconstruction is one of the oldest problems in studies of evolution (Haeckel 1866) and is amongst the most challenging statistical problems (Neyman 1971; Felsenstein 1978). Methods based on molecular sequences have been in use for over forty years (e.g. Fitch, Margoliash 1967a) and are now the main technique used for phylogeny inference. The alignment problem, that is estimating the homology between sites in different biological sequences, is confounded with the problem of phylogeny estimation. Therefore, many methods have recently been developed that try to find the solution to both problems simultaneously (e.g. Lunter et al. 2005; Suchard, Redelings 2006; Liu et al. 2009b). However, the most commonly used strategy remains a two-phase process where homology between sites is estimated through the process of multiple sequence alignment (MSA), and the process of tree building (TB) is not considered until after the MSA has been calculated and fixed.

Some studies involving real data have suggested that the choice of MSA method may have more impact on the resulting phylogeny than the choice of TB method (e.g. Morrison, Ellis 1997; Ogden, Whiting 2003). Despite claims like these, MSA methods are still often assessed using simulated data without giving consideration to the problem of TB (e.g. Pollard et al. 2004; Nuin et al. 2006). Similarly, many attempts at using simulation to assess TB have been undertaken whilst ignoring the problem of MSA (e.g. Hillis 1995; Huelsenbeck 1995; Posada, Crandall 2002; Rosenberg, Kumar 2003). Even when biological realism is increased through the inclusion of insertions and deletions the "true" simulated alignment is often used which negates the need for an MSA (e.g. Dwivedi, Gadagkar 2009). Even if a study uses an MSA method to align the simulated data this still tells us nothing about the role that alignment errors may play if no comparison is made between the true alignments and the estimated alignments (e.g. Hall 2005).

The few studies that have attempted to directly assess the role of MSA accuracy on downstream TB are often found lacking. For example, they may have focused on MSA and only used one method of TB (Roshan et al. 2006), or vice versa (Hall 2005; Ogden, Rosenberg 2006), whilst perhaps investigating only small trees (Ogden, Rosenberg 2006) or only balanced trees (Hall 2005). All of these studies generated their simulated data using computer programs (ROSE: Stoye et al. 1998; MySSP: Rosenberg 2005b; EvolveAGene: Hall 2008a) that are known to handle insertions and

deletions incorrectly (Strope et al. 2009). The problem stems from the fact that these programs implement indel rates that depend on the sequence length at the last interior node, rather than rates that depend on the current sequence length as they properly should since the total rate is a sum over sites (see Discussion). This may have had some effect on the conclusions drawn from those studies.

This investigation aims to address many of those issues. Firstly, the data is generated using INDELible (Fletcher, Yang 2009) which is one of the few simulation programs that updates indel rates correctly. Secondly, both balanced and pectinate tree shapes are used in the simulations to test whether alignment errors are more important for TB on pectinate trees as has been suggested before (Ogden, Rosenberg 2006). For each tree shape the complexity of the simulation is increased by progressively adding more taxa, by increasing branch lengths or indel rates, or by deviating from the molecular clock more and more severely. Thirdly, a total of 8 MSA alignment programs and 4 methods of TB are tested. The MSA programs used were: PRANK (Löytynoja, Goldman 2005); MAFFT (Katoh, Toh 2008b); MUSCLE (Edgar 2004); ProbConsRNA (Do et al. 2005); T-Coffee (Notredame et al. 2000); ClustalW (Larkin et al. 2007); and DIALIGN-TX (Subramanian et al. 2008). The TB methods used included: maximum likelihood (ML) with RAxML (Stamatakis 2006); maximum parsimony (MP) with DNAPARS[2]; distance calculation with DNADIST[1] followed by neighbour joining (NJ) with NEIGHBOR[1]; and a novel alignment-free method (FST, see below).

Comparison of the estimated alignments with the true alignment allows identification of which MSA methods are more accurate, and which types of dataset are harder to align. TB performed on these estimated alignments allows identification of which methods generate alignments that are conducive to recovering the most accurate tree. The additional use of the true alignment allows proper assessment of the accuracy of different TB methods when there are no alignment errors, and provides a benchmark for TB performed on the estimated alignments. Therefore, this study allows a full assessment of which methods are most accurate for each phase of the process; permits identification of the types of datasets that are particularly problematic (or otherwise) for each phase; and allows me to identify whether, in some situations, one of the phases is particularly important (or unimportant).

---

[2] DNAPARS, DNADIST and NEIGHBOR are all part of the PHYLIP package (Felsenstein, 2010)

4.2 Method

4.2.1 Simulation Process

Two basic ultrametric trees of 16 taxa were used as the starting point for all simulations. The basic balanced tree (figure 4.1a) had equal branch lengths of $t_B = 0.025$ giving an overall tree length of 0.75. The basic pectinate tree (figure 4.1b) was scaled to also have a tree length of 0.75, giving a basic branch length unit of $t_P = 1/180 \approx 0.0056$. Five sets of simulations were performed. In the first set of simulations the number of taxa on each tree was changed. Explicitly, data was simulated on ultrametric balanced trees with 4, 16, 32, 64, 128 or 256 taxa, or on ultrametric pectinate trees with 4, 6, 8, …, 46, 48 taxa, whilst keeping the values of t_B and t_P mentioned above constant. Figure 4.1 shows how the 32 taxa balanced tree and the 18 taxa pectinate tree are related to the two basic 16 taxa

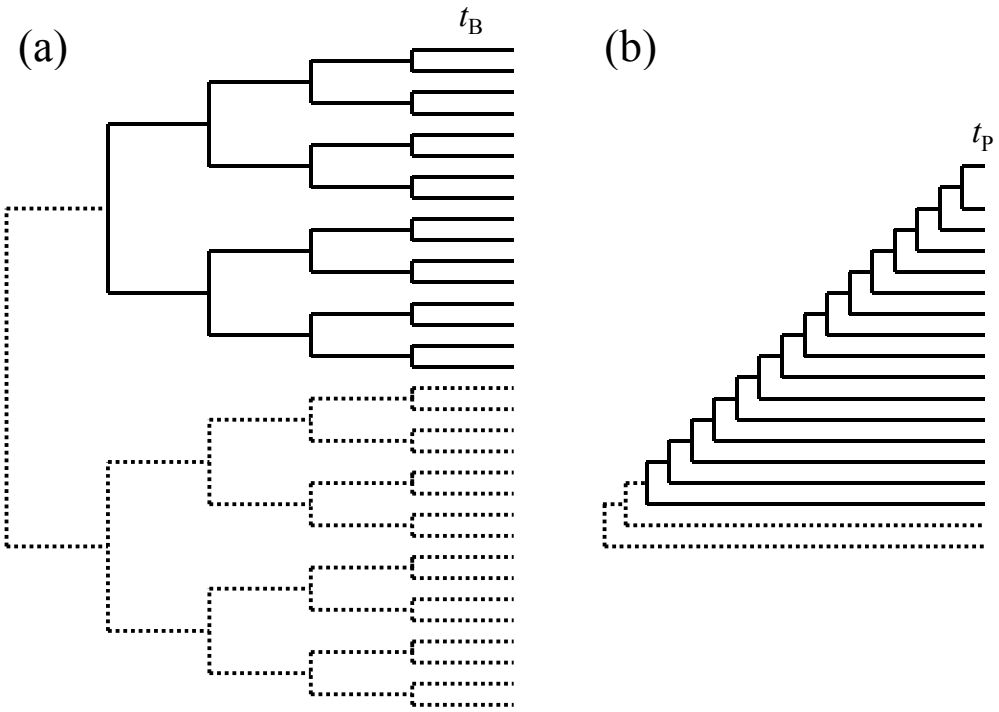


Figure 4.1: The two tree shapes used in simulation

Two basic 16 taxa trees were used (solid lines). Both of these trees have a total length of 0.75. The basic branch lengths used were $t_B = 0.025$ and $t_P = 0.0056$. An example of how taxa were added to each of the trees is represented by the dotted lines. N.B. the estimated trees are unrooted.

trees. In the second set of simulations sequence divergence was increased by repeatedly adding 0.025 or 0.0056 to the two basic branch length units t_B and t_P, i.e. doubling the branch lengths, then tripling them, and so on. Branch lengths $t_B \leq 0.6$ and $t_P \leq 0.1$ were considered. The third set of simulations was similar, except only the relative rate of insertion and deletion was increased such that the branch length measured in expected substitutions remained constant.

The fourth and fifth sets of simulations were designed to generate datasets that deviated from the molecular clock using two new features available in INDELible v1.04. In the fourth set, the trees became less and less ultrametric whilst keeping sequence divergence roughly constant. This was done, by making each branch length on the tree bigger or smaller, with equal probability, and repeating the process for each replicate dataset. The deviation factor (*F*) from the basic branch length units t_B and t_P was either ±10%, ±20%, …, ±100%, i.e. in the most extreme case branch lengths were either doubled or collapsed to zero. The fifth set of simulations was a little more complicated as they combined increasing sequence divergence with increasingly severe violation of the molecular clock. This was performed by randomly multiplying or dividing branch lengths (with equal probability) by a divergence factor $k = 2, 3, …., 10$. Again, this process was repeated for each replicate dataset.

For all combinations of tree shape and branch lengths the choice of substitution model and indel model was the same. For substitutions I used HKY (Hasegawa et al. 1984; Hasegawa et al. 1985) with a transition/transversion rate ratio of $\kappa = 4$ and nucleotide frequencies of $\pi_T = 0.1$, $\pi_C = 0.2$, $\pi_A = 0.3$ and $\pi_G = 0.4$. Gamma rate variation (Yang 1994b) was used with shape parameter $\alpha = 0.5$ and five discrete categories. Insertion and deletion lengths were both modelled by a truncated power law distribution with parameter $a = 1.8$ which is consistent with empirical estimates (Benner et al. 1993; Gu, Li 1995; Zhang, Gerstein 2003; Chang, Benner 2004; Yamane et al. 2006; Cartwright 2009) and a maximum indel size of 100, giving a mean indel length of approximately 4.4. Insertion and deletion rates were both set at 0.05 relative to the substitution rate giving a substitution/indel rate ratio of 10. INDELible was then used to generate both the unaligned sequences and the true alignment for 1000 replicate datasets of each combination of model and tree, with an initial root sequence of length 1000, sampled from the stationary nucleotide frequencies.

4.2.2 Multiple Sequence Alignment Methods and Measuring Alignment Accuracy

The unaligned sequences generated by INDELible were subjected to MSA with one of 8 alignment programs. The MSA programs used were: PRANK (version 091016, Löytynoja, Goldman 2005); MAFFT (version 6.716b, Katoh, Toh 2008b); MUSCLE (versions 3.7 and 4, Edgar 2004); ClustalW (version 2.0.11, Larkin et al. 2007); ProbConsRNA (version 1.1, Do et al. 2005); T-Coffee (version 8.14, Notredame et al. 2000); and DIALIGN-TX (version 1.0.2, Subramanian et al. 2008).
In each case the program's default options were used unless a "slow and thorough" or "accurate" mode was available (e.g. "+F" in PRANK, or "--localpair" in MAFFT) .

In order to investigate the effect of MSA errors on TB I used two measures of alignment accuracy: the total column score (TC) and the sum of pairs score (SPS). TC is the proportion of columns from the true alignment that are reproduced exactly in the test alignment, and SPS is the proportion of aligned nucleotide pairs from the true alignment that are also aligned together in the test alignment (Thompson et al. 1999a). I use both because one can think of scenarios where the two scores could disagree dramatically. For example, consider first a dataset that contains very many closely related species and one very distantly related "orphan" sequence. The resulting alignment may align the large number of similar sequences with great accuracy but sites in the one orphan sequence may be placed incorrectly. Such an alignment would have a high SPS score but a low TC score, and one may feel that TC penalizes such an alignment too heavily. In another case, consider a dataset containing two non-homologous groups of sequences, where sequences within each group are homologous and closely related. In the true alignment the sub-alignments for each group would be placed opposite gaps in the other group. If the two sub-alignments are incorrectly placed opposite each other in an estimated alignment then the TC score would be low. However, the SPS score would be very close to that of the true alignment. In this scenario SPS might be seen to not penalize the alignment enough. Therefore, SPS may be considered a less stringent measure than TC and the two measures reveal different aspects of an alignment method's accuracy. Note that the true alignment has TC = SPS = 1.

4.2.3 Phylogeny Reconstruction Methods and Measuring Their Performance

The main TB methods used are: maximum likelihood (ML) with RAxML (version 7.0.4, Stamatakis 2006) using the GTRGAMMA model; maximum parsimony (MP) with DNAPARS; or distance calculation with DNADIST using the F84 model, followed by neighbour joining (NJ) with NEIGHBOR. The three programs DNAPARS, DNADIST and NEIGHBOR are all part of the PHYLIP package (version 3.69, Felsenstein 2010). Only DNADIST required model parameter values as input, so the correct values were provided where necessary. Specifically, the correct coefficient of variation for the gamma distribution $\mathrm{CV} = \alpha^{-1/2} = \sqrt{2} \approx 1.41$ and average transition/transversion rate ratio $R = \left(\pi_{\mathrm{T}}\pi_{\mathrm{C}} + \pi_{\mathrm{A}}\pi_{\mathrm{G}}\right)\kappa \Big/ \left[\left(\pi_{\mathrm{T}} + \pi_{\mathrm{C}}\right)\left(\pi_{\mathrm{A}} + \pi_{\mathrm{G}}\right)\right] = 8/3 \approx 2.67$ (p. 18, Yang 2006) were given to DNADIST. The true nucleotide frequencies were not provided, and DNADIST used the empirical frequencies instead.

In addition, a novel and (currently) unpublished method of distance-based alignment-free phylogeny reconstruction (Schwarz, Fletcher *et al.*, 2010) based on finite-state transducers was also tested (FST). This method will not be described here in detail but is centred on applying the positive semi-definite (PSD) rational kernel framework of Cortes *et al* (2004) to the problem of calculating a weight matrix. The method has theoretical advantages over other distance methods because the kernel score incorporates information about indels as well as substitutions, and implicitly takes all possible alignments between each pair of sequences into account. Furthermore, the calculated weights obey the triangle inequality and so can be considered as relative distances in the mathematical sense. The FST method takes the unaligned sequences as input and outputs a weight matrix. The final inferred tree was generated from this weight matrix using NEIGHBOR.

For each replicate dataset, the true alignment and the alignments from the 8 MSA programs were subjected to the three main methods of TB, and the original unaligned sequences were subjected to the FST method of TB. For the MP analyses, if there were many tied trees then the strict consensus tree was taken as the inferred tree. Thus a total of 28 trees were estimated for each replicate dataset. The accuracy of each method was calculated as the average proportion of branches from the true tree that were correctly recovered in the 1000 inferred trees, hereafter referred to as PBR.

4.3 Results

4.3.1 Alignment Accuracy

The data that was generated on the two basic trees did not present a difficult challenge to any of the MSA methods. On the balanced tree the average TC was 0.93 for PRANK, 0.91 for the two versions of Muscle, 0.90 for T-Coffee and MAFFT, and 0.87 for Clustal and DIALIGN-TX, with all methods having an average SPS ≥ 0.97. For the pectinate tree the average TC was 0.92 for PRANK and Muscle v4, 0.91 for ProbConsRNA and Muscle v3, 0.89 for MAFFT, 0.86 for DIALIGN-TX and 0.85 for Clustal. The average SPS for each method was ≥ 0.98. However, in nearly every one of the more complicated simulations either PRANK or Muscle v4 produced the most accurate alignments. Therefore, much of the following comparisons will be concerning those programs only. The relative performance of these two methods for selected simulations is shown in table 4.1 below.

Table 4.1: Alignment Accuracy of PRANK and Muscle v4 in Selected Cases

			Average CS		% of the time that MSA method was most accurate as judged by TC		Average SPS		% of the time that MSA method was most accurate as judged by SPS	
Tree	Set	Details	PRANK	Muscle	PRANK	Muscle	PRANK	Muscle	PRANK	Muscle
Balanced	-	Basic Tree	0.93	0.91	82	5	0.99	0.99	59	15
	1	64 taxa	0.82	0.73	100	0	0.98	0.97	94	6
		256 taxa	0.75	0.51	100	0	0.98	0.97	100	0
	2	$t_B = 0.3$	0.27	0.13	100	0	0.36	0.43	6	94
		$t_B = 0.6$	0.12	0.03	92	0	0.06	0.08	11	90
	3	$(\lambda_I + \lambda_D)/\lambda_S = 0.5$	0.74	0.56	100	0	0.93	0.91	98	2
		$(\lambda_I + \lambda_D)/\lambda_S = 0.8$	0.64	0.40	100	0	0.86	0.84	95	6
	4	$F = \pm 50\%$	0.93	0.91	81	5	0.99	0.99	58	18
		$F = \pm 100\%$	0.93	0.90	80	6	0.99	0.99	60	17
	5	$k = 5$	0.76	0.67	96	3	0.95	0.94	60	33
		$k = 10$	0.51	0.39	98	2	0.82	0.84	28	71
Pectinate	-	Basic Tree	0.92	0.92	36	41	0.99	0.99	22	46
	1	32 taxa	0.70	0.72	26	74	0.97	0.98	2	96
		48 taxa	0.50	0.52	22	79	0.93	0.95	0	100
	2	$t_P = 0.05$	0.26	0.27	43	57	0.63	0.76	0	100
		$t_P = 0.1$	0.16	0.11	64	2	0.25	0.39	0	100
	3	$(\lambda_I + \lambda_D)/\lambda_S = 0.5$	0.68	0.63	90	10	0.94	0.95	10	82
		$(\lambda_I + \lambda_D)/\lambda_S = 0.8$	0.57	0.49	96	4	0.88	0.90	8	92
	4	$F = \pm 50\%$	0.92	0.92	40	39	0.99	0.99	25	43
		$F = \pm 100\%$	0.90	0.90	34	44	0.99	0.99	20	50
	5	$k = 5$	0.64	0.68	26	65	0.94	0.96	8	84
		$k = 10$	0.32	0.38	15	79	0.82	0.88	1	97

NOTE – percentages may not sum to 100 if other methods were most accurate on some replicates.

Figures 4.2 to 4.6 show the accuracy of the 8 methods of MSA, for each of the five sets of simulations that build on the two basic trees. From the first set of simulations it was found that as the number of taxa increased, alignment accuracy decreased (figure 4.2) for both shapes of tree. PRANK performed much better on the balanced trees. As the number of taxa increases PRANK recovers a significantly greater number of columns correctly when compared to Muscle v4, yet the SPS scores are roughly similar with both methods correctly recovering the vast majority of the nucleotide pairs that are aligned together in the true alignment (table 4.1). However, in the true 256 taxa alignments there were approximately 30 million aligned nucleotide pairs on average. Therefore, the difference of 1% in the SPS scores actually represents the fact that Muscle misaligns several hundred thousand nucleotide pairs more than PRANK. The majority of these errors occurred for nucleotide pairs that were on

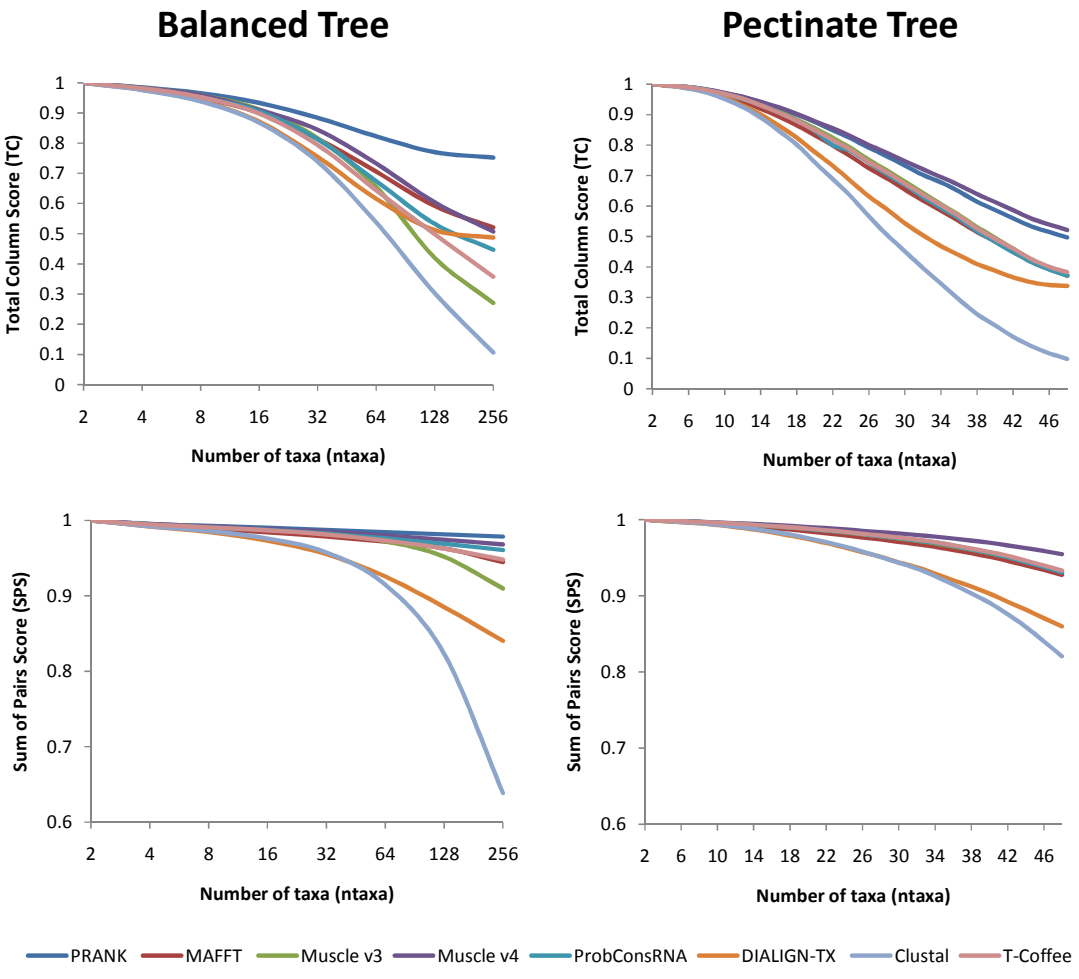


Figure 4.2: MSA Accuracy when Number of Taxa Increases

opposite sides of the deepest branch in the phylogeny. This accounts for the small difference in SPS and the large difference in TC between the two methods. For the pectinate tree shape Muscle v4 appeared to perform better than PRANK (table 4.1). The differences in MSA accuracy scores for the two methods on the pectinate tree are small but consistent. This change in relative performance of the two methods on balanced and pectinate trees is a pattern that is repeated on the other datasets (this shall be discussed further below). Explicitly, PRANK tends to have the best TC on the balanced tree and Muscle v4 tends to have the best SPS on the pectinate tree, with SPS on the balanced tree and TC on the pectinate tree judging the two methods to be roughly similar.

This general pattern also held true for the simulations where sequence divergence increased (figure 4.3, table 4.1). It is interesting to note that when sequences are very divergent DIALIGN-TX, with its local alignment strategy, begins to perform better than most other methods when it comes to

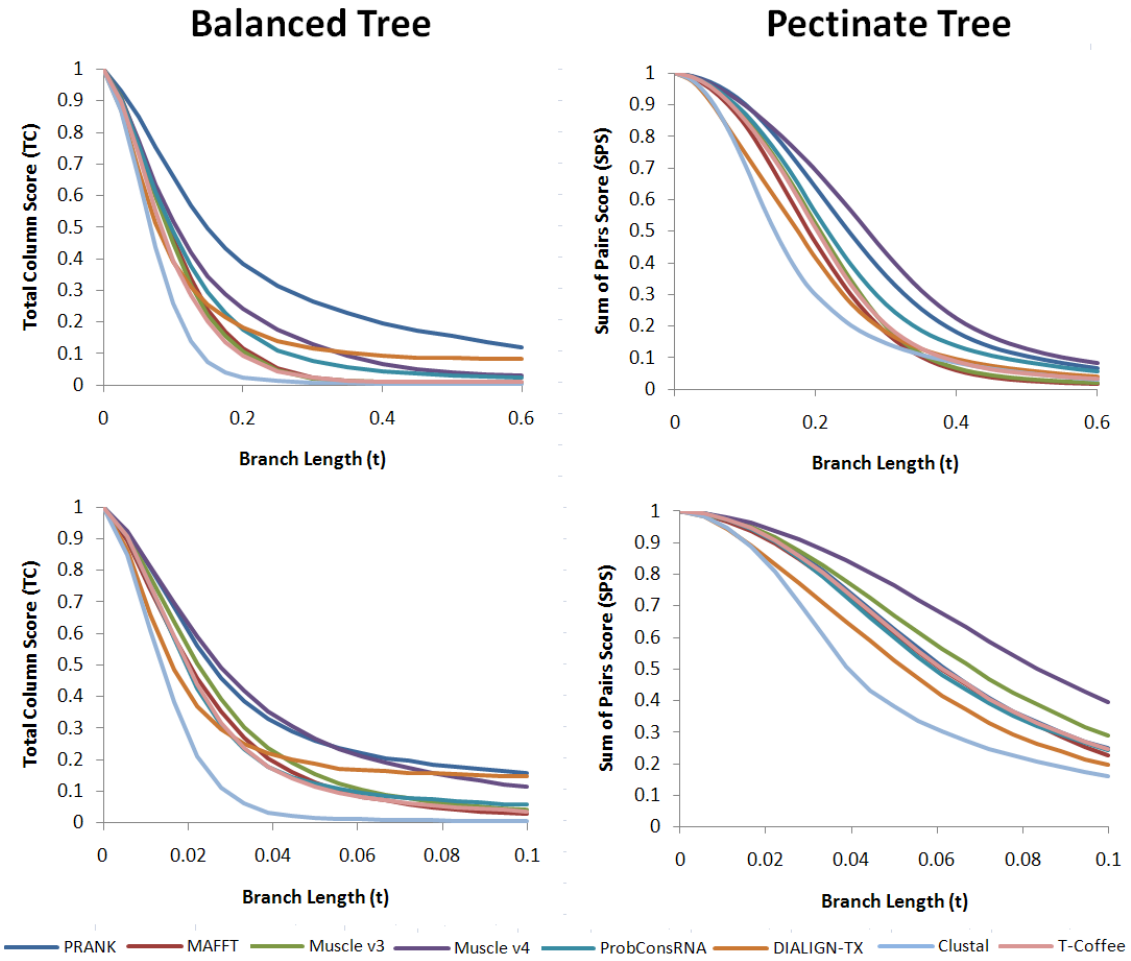


Figure 4.3: MSA Accuracy when Sequence Divergence Increases

recovering alignment columns correctly. For example, when $t_B = 0.6$ and $t_P = 0.1$, DIALIGN-TX has the second highest average TC scores, 0.08 and 0.15, with the method being most accurate on 8% and 34% of the replicates, on the balanced and pectinate trees respectively. SPS scores for DIALIGN were only 0.04 and 0.20 for the same two cases. These results are consistent with the observation that local alignment algorithms become increasingly appropriate and accurate as sequences become more divergent. Such algorithms are good at aligning locally conserved regions of sequences and they focus on this goal with less regard for alignment quality over the alignment as a whole (see chapter 1). It is for this reason that they are commonly used in database search algorithms that compare divergent sequences looking for regions that may be conserved due to functional or structural constraints.

For the third set of simulations the rate of insertion/deletion relative to substitutions was increased, but the branch lengths remained the same (figure 4.4). Thus, when compared to the last set

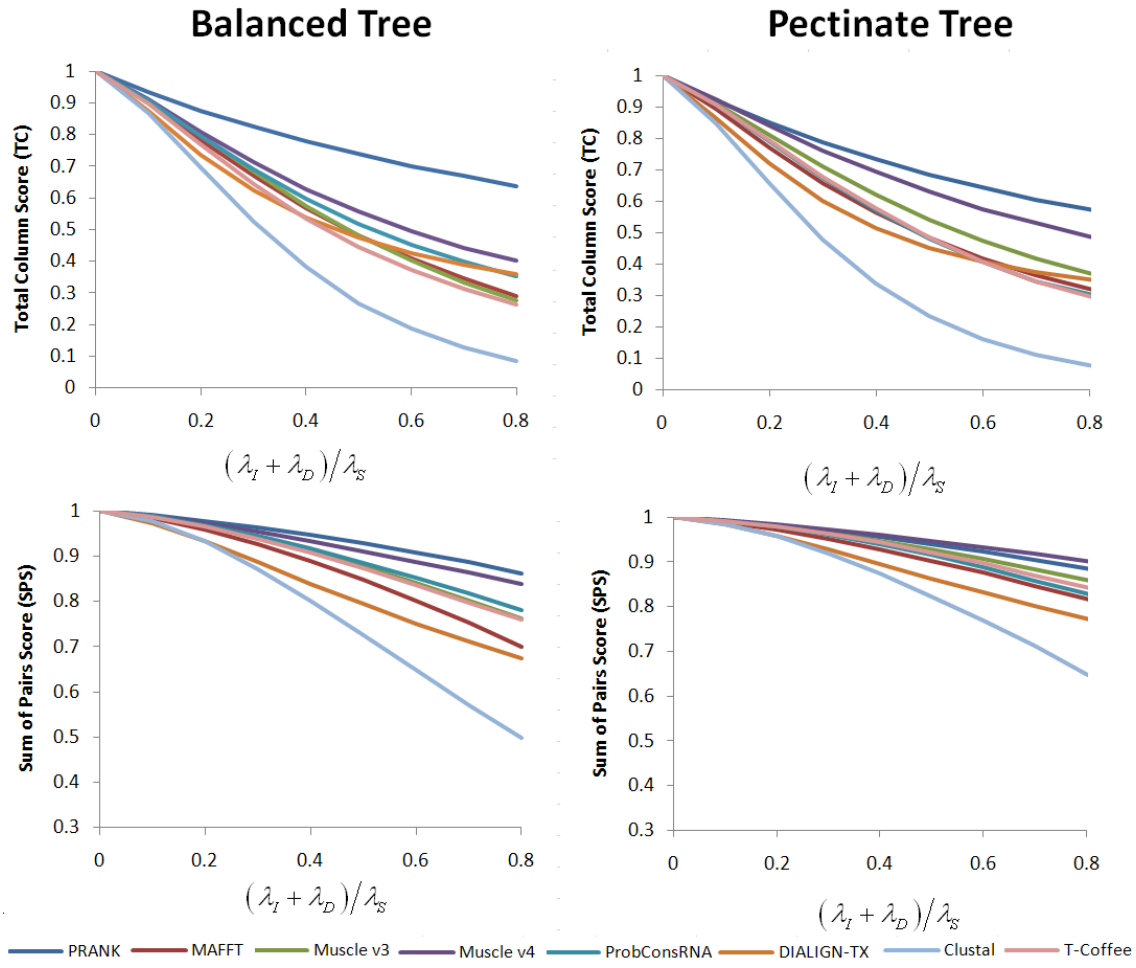


Figure 4.4: MSA Accuracy when Indel Rate Increases

of simulations, this set places more emphasis on the accuracy of the procedures for handling indels as opposed to the scoring scheme used for nucleotide substitutions. As a result this set of simulations favours PRANK more than the last set and this is reflected in the results (table 4.1). Indeed, PRANK can be seen to be better as judged by TC on both trees, whilst also being better for SPS on the balanced tree. Muscle remains superior as judged by SPS on the pectinate trees.

The fourth set of simulations involved datasets that deviated from the molecular clock in an increasingly severe fashion whilst keeping the overall level of divergence roughly constant (figure 4.5). The degree that simulation guide trees deviated from ultrametricity had little effect on alignment accuracy (table 4.1), with all methods performing roughly the same as they did on the two basic trees. Again PRANK performed better on the balanced tree or when judged by TC, and Muscle performed better on the pectinate tree or when judged by SPS.

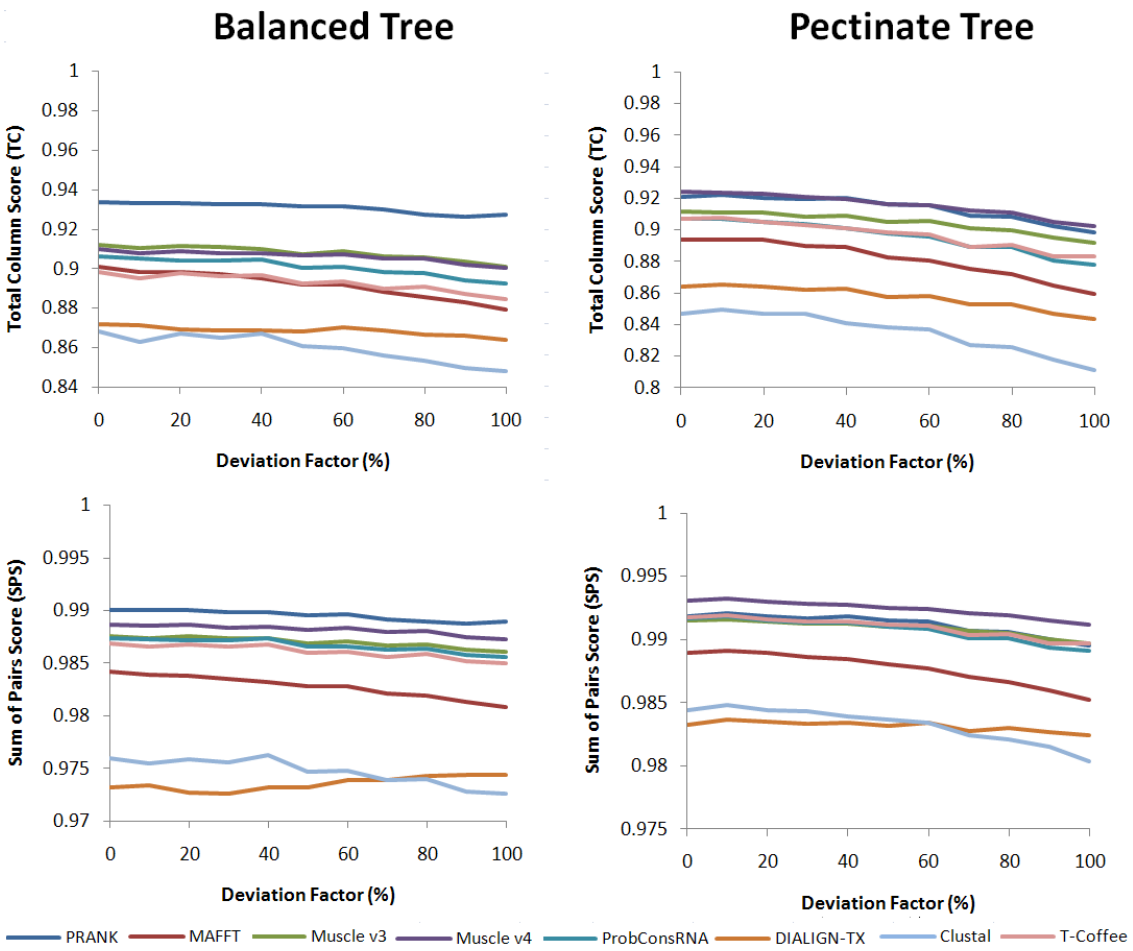


Figure 4.5: MSA Accuracy when Molecular Clock is Violated

The final set of simulations combined increasing sequence divergence with increasing deviation from the molecular clock (figure 4.6). The results are broadly similar to those where sequence divergence increased but the molecular clock held true ($k = 10$ corresponds to $t_B \approx 0.12$ and $t_P \approx 0.03$ in figure 4.3), and the different methods were generally ranked in the same order.

In summary, increasing the number of taxa in a dataset, increasing the sequence divergence, or increasing the insertion/deletion to substitution ratio all cause a decrease in alignment accuracy, as one would expect. Conversely, increasing violation of the molecular clock, while keeping sequence divergence low and roughly constant, results in datasets that are no harder to align than their ultrametric counterparts. Also, a consistent pattern emerged: PRANK generates more accurate alignments for data generated from a balanced tree or, more generally, when alignments are judged by TC. This is in contrast to Muscle v4 which tends to generate more accurate alignments on pectinate trees and tends to appear relatively more accurate when judged by SPS.

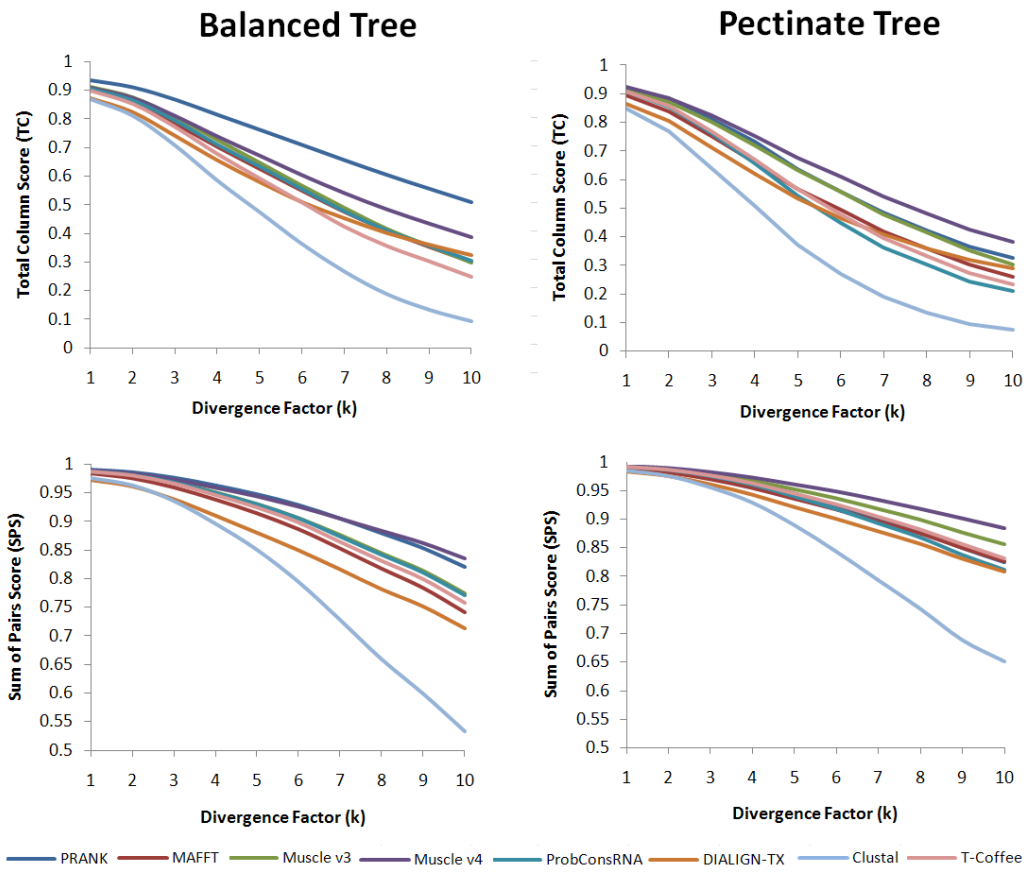


Figure 4.6: MSA Accuracy when Molecular Clock is Violated and Divergence Increases

4.3.2 Phylogeny Reconstruction Accuracy

I shall now discuss the accuracy of the different methods of TB, first focusing on the analyses where there are no alignment errors, before proceeding to discuss the TB performance on the generated alignments.

For the basic balanced tree the PBR was > 0.99 for the true alignment, and for each of the generated alignments, regardless of the TB method, meaning that inferred trees were nearly always 100% correct. For the basic pectinate tree there was a little more variation. For the true alignment the PBRs were 0.92 (ML), 0.84 (MP) and 0.85 (NJ), whilst for the generated alignments the PBRs were 0.91 to 0.92 (ML), 0.79 to 0.83 (MP), and 0.82 to 0.84 (NJ). Thus, on this relatively easy dataset we can see that ML seems to be the superior TB method. The FST method had a PBR of 0.88. Note how this is better than any of the other PBR scores based on NJ reconstruction of a distance matrix, even that which was based on the true alignment.

Figures 4.7 to 4.11 show the results for the four different TB methods when there are no alignment errors to contend with, and figures 4.12 to 4.16 show the results for each MSA and TB combination. Selected PBR values and comparisons are reported in table 4.2. PBR values are reported for the ML, MP and NJ analyses performed on the true alignment, and for the three "best" performing inference combinations from the remainder of the analyses (PRANK+ML, Muscle v4+ML, FST+NJ, one of these was the most accurate for each set of simulation conditions). Three sets of comparisons are also reported in table 4.2. Firstly, I compare the performance of the different TB methods when there was no MSA to perform (C1), then I compare the performance of each TB method on the true alignment compared to the estimated alignments (C2), and finally I compare the relative performance of the three "best" inference methods mentioned above (C3).

Phylogeny Reconstruction Accuracy in the Absence of Alignment Errors

Comparison of the ML, MP, NJ analyses of the true alignments, and of the FST analyses of the original sequences, allows for a fair comparison of TB methods, and which types of dataset and tree are more problematic, when there are no alignment errors. The results for the four TB methods and the five sets of simulations are shown in figures 4.7 to 4.11

Table 4.2: Phylogenetic Reconstruction Accuracy

Tree	Set	Details	Average Proportion of Branches Recovered (PBR)						Comparison 1 (C1) [a]				Comparison 2 (C2) [b]			Comparison 3 (C3) [c]		
			PRANK (ML)	Muscle v4 (ML)	True (ML)	True (MP)	True (NJ)	FST	True (ML)	True (MP)	True (NJ)	FST	ML	MP	NJ	PRANK (ML)	Muscle v4 (ML)	FST
Balanced	-	Basic Tree	1.00	1.00	1.00	1.00	1.00	1.00	100	99	99	100	100	100	100	100	100	100
	1	64 taxa	1.00	1.00	1.00	1.00	1.00	1.00	100	95	96	99	100	97	96	100	99	99
		256 taxa	1.00	1.00	1.00	1.00	1.00	1.00	99	81	65	92	90	81	70	90	89	84
	2	$t_B = 0.3$	0.98	0.98	1.00	1.00	0.99	1.00	99	99	90	95	99	99	100	83	81	95
		$t_B = 0.6$	0.54	0.67	0.99	0.99	0.06	0.79	88	99	0	5	100	100	98	5	28	80
	3	$(\lambda_I + \lambda_D)/\lambda_S = 0.5$	1.00	1.00	1.00	1.00	1.00	1.00	100	97	99	100	100	97	99	100	100	100
		$(\lambda_I + \lambda_D)/\lambda_S = 0.8$	1.00	1.00	1.00	1.00	1.00	1.00	100	97	97	100	100	97	97	100	100	100
	4	$F = \pm 50\%$	1.00	1.00	1.00	0.99	0.99	0.99	97	92	86	88	98	95	91	96	96	88
		$F = \pm 100\%$	0.63	0.63	0.63	0.51	0.52	0.58	73	8	14	44	46	44	43	39	35	31
	5	$k = 5$	0.79	0.79	0.79	0.71	0.71	0.75	63	26	20	43	39	42	46	36	32	30
		$k = 10$	0.72	0.72	0.70	0.68	0.65	0.73	45	29	20	61	24	31	50	25	25	39
Pectinate	-	Basic Tree	0.91	0.92	0.92	0.84	0.85	0.88	74	27	26	45	68	63	50	54	60	39
	1	32 taxa	0.79	0.82	0.83	0.59	0.62	0.66	97	2	2	5	39	38	34	16	32	2
		48 taxa	0.64	0.70	0.73	0.45	0.46	0.49	99	0	0	1	54	50	49	5	37	0
	2	$t_P = 0.05$	0.68	0.69	0.87	0.82	0.74	0.78	64	40	12	23	91	84	85	23	29	70
		$t_P = 0.1$	0.47	0.32	0.78	0.82	0.58	0.57	46	67	7	3	98	99	90	28	5	85
	3	$(\lambda_I + \lambda_D)/\lambda_S = 0.5$	0.85	0.87	0.90	0.82	0.79	0.86	68	30	16	41	53	60	52	28	37	34
		$(\lambda_I + \lambda_D)/\lambda_S = 0.8$	0.80	0.83	0.88	0.80	0.75	0.84	64	31	16	39	55	60	67	25	38	39
	4	$F = \pm 50\%$	0.86	0.87	0.87	0.77	0.78	0.67	80	27	30	9	63	63	54	57	61	7
		$F = \pm 100\%$	0.58	0.58	0.58	0.48	0.56	0.31	68	17	55	3	40	68	47	43	41	3
	5	$k = 5$	0.66	0.69	0.70	0.56	0.61	0.32	78	19	33	2	43	58	47	30	42	1
		$k = 10$	0.50	0.57	0.62	0.48	0.54	0.24	80	17	35	1	49	71	63	20	37	0

[a, b, c] Comparisons C1, C2 and C3 show the percentage of replicates where different methods yielded (perhaps jointly) the most accurate tree. C1[a] compares the four TB methods with each other when there were no MSA errors to deal with. C2[b] shows the percentage of the time that analyses involving the true alignment produced the most accurate tree when compared to the analyses involving the estimated alignments (FST method excluded). C3[c] reports the performance of the three best inference methods when compared with each other and the analyses based on the other generated alignments (analyses involving the true alignment excluded).

Firstly when sequence divergence is relatively low, balanced trees are far easier to recover than pectinate trees as more taxa are added (figure 4.7). For the balanced trees all methods recovered nearly every branch, For the pectinate tree, all methods struggled to recover the deeper branches as they were added, but ML performs better nearly 100% of the time with the other 3 methods consistently failing to recover nearly twice as many branches as ML (table 4.2).

As sequence divergence increases on the balanced tree (figure 4.8) the pairwise distance calculations become less accurate until they reach a point where saturation of substitutions makes the calculations impossible and the NJ method fails to recover any branches correctly at all. Performance

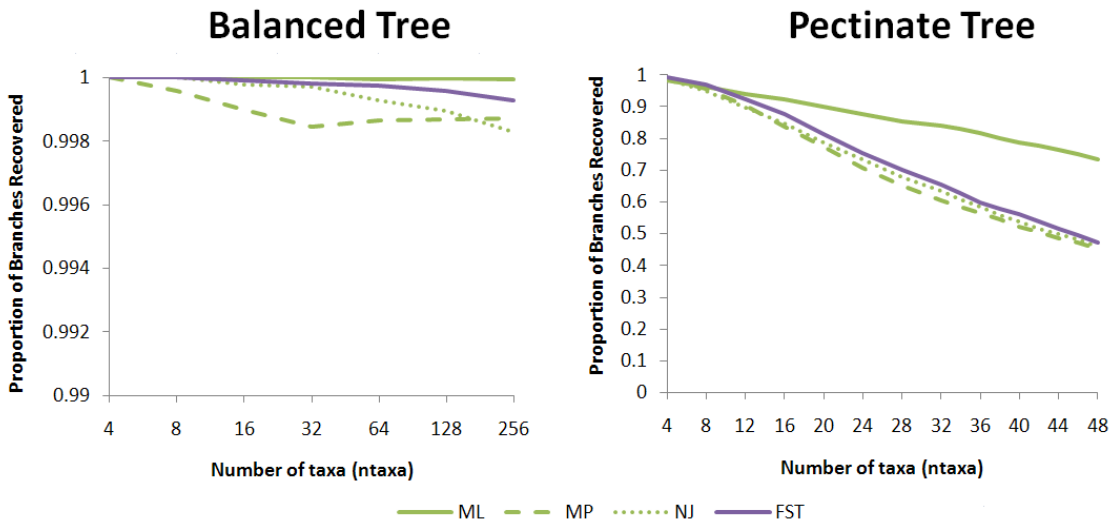


Figure 4.7: TB Accuracy when Number of Taxa Increases

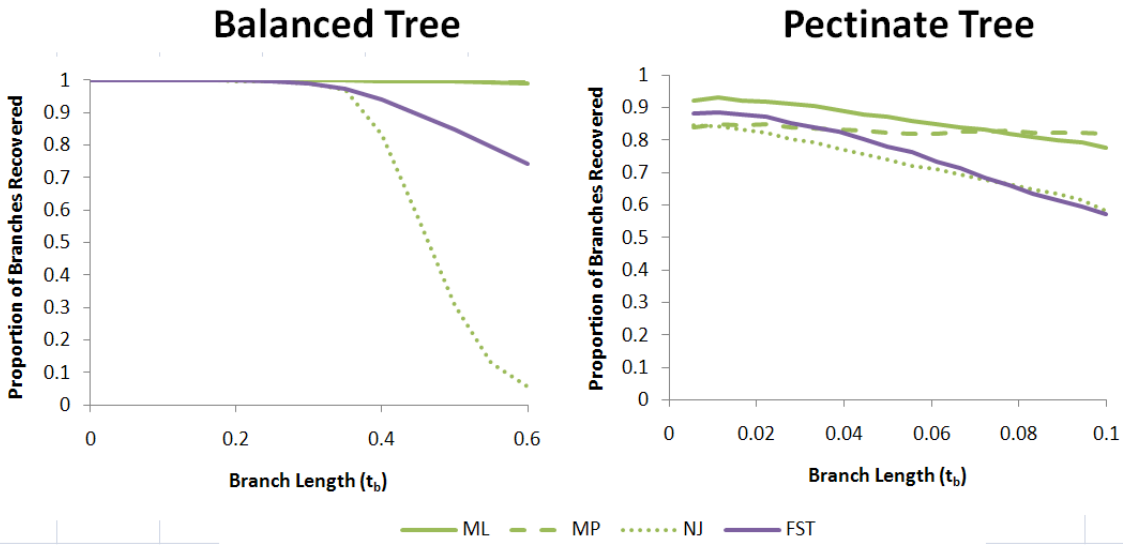


Figure 4.8: TB Accuracy when Sequence Divergence Increases

of the FST method also degrades, albeit at a slower rate, whilst the accuracy of the MP and ML methods barely changes (table 4.2). On the pectinate tree the performance of most methods deteriorates as divergence increases but does not for the MP analyses. Whilst this is true for both tree shapes (figure 4.8) it is perhaps most obvious on the pectinate tree.

When divergence is low changing the indel rate has little effect (figure 4.9). On the balanced tree all methods recover the correct tree with a very high accuracy rate and performance deteriorates only slowly on the pectinate tree. Once again ML was the superior method. In the two sets of simulations where the molecular clock is violated (figures 4.10 and 4.11) ML, MP and NJ are all affected in a similar manner as the trees become less and less ultrametric and the ranking of the relative performance of the three methods remains the same (table 4.2). The most striking result of this set of simulations is the comparatively poor performance of the FST method, which is most striking for the pectinate tree shape. Deviations from the molecular clock seem to cause this method to decrease in accuracy, although it should be noted that the FST method is at a disadvantage compared to the other three methods as it was not provided with any information about the true alignment. Therefore, rather than failing on these last two sets of simulations it may be more appropriate to view the performance of the FST method as surprisingly good on the previous three sets of simulations. Indeed, in those cases, NJ on FST weights is more accurate that NJ on DNADIST distances which were derived from the true alignment with knowledge of the true parameter values.

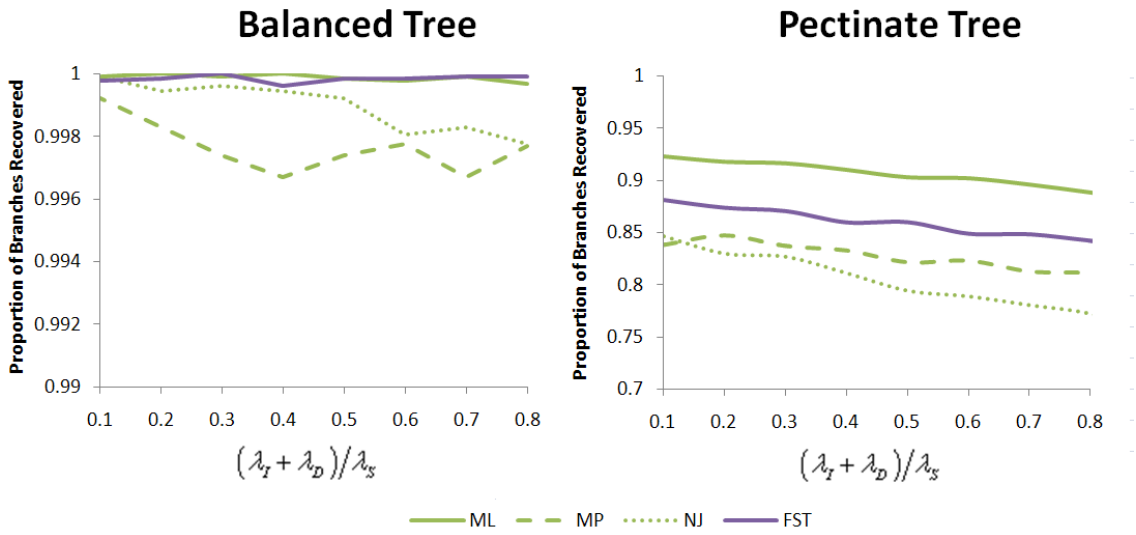


Figure 4.9: TB Accuracy when Indel Rate Increases

In summary, when there are no alignment errors two very clear patterns hold true over all five sets of simulations. Firstly, shallow and balanced phylogenies are easier to recover than deeper pectinate shaped phylogenies for all TB methods. Secondly, ML is clearly the superior TB inference method in general. ML has the largest PBR values (table 4.2) and yields the most accurate trees more often (C1) in nearly every set of simulations. The only two exceptions to this trend are for pectinate trees with long branches (figure 4.8) where MP performs better, and for balanced trees when divergence is high and the molecular clock is severely violated (figure 4.11) where FST performs better.

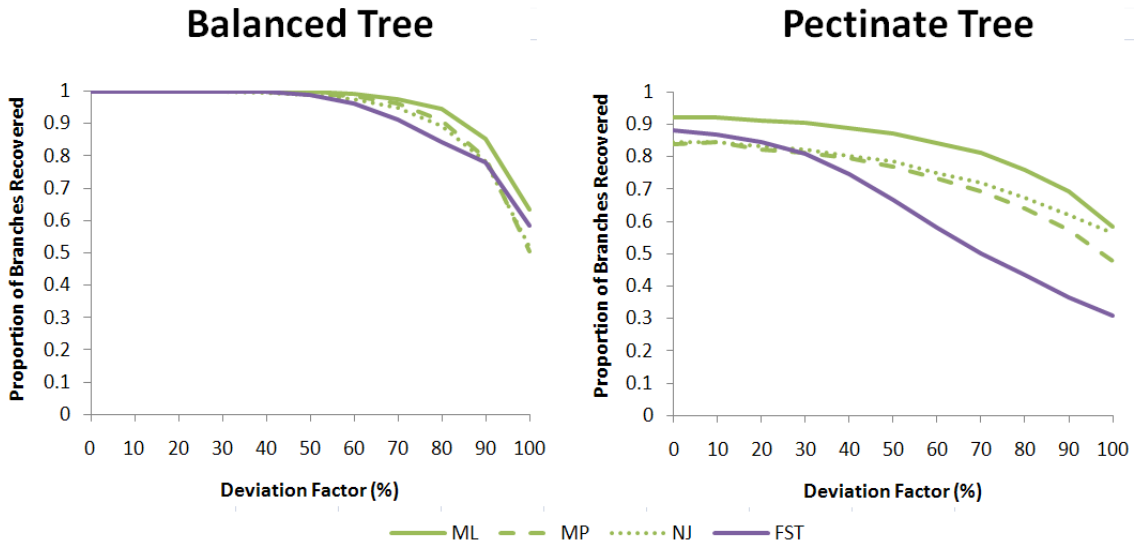


Figure 4.10: TB Accuracy when Molecular Clock is Violated

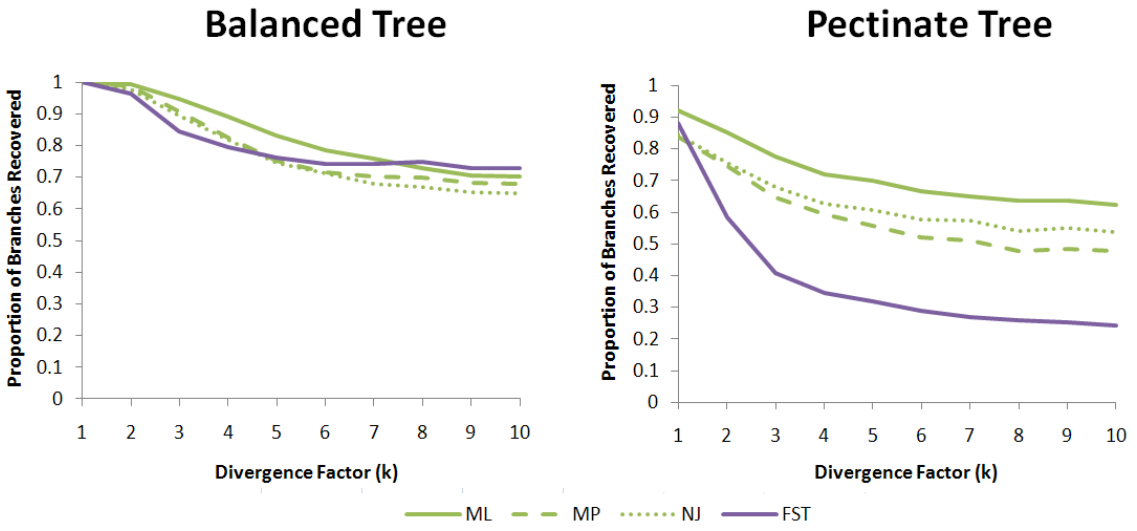


Figure 4.11: TB Accuracy when Molecular Clock is Violated and Divergence Increases

Phylogeny Reconstruction Accuracy In the Presence of Alignment Errors

The results for simulations that involved estimated alignments are shown in figures 4.12 to 4.15, with separate panels for each of the three main TB methods. The FST method is shown on each panel and the results for the true alignments are also included for comparison. For all combinations of MSA method, TB method and simulation conditions, the pectinate trees were once again harder to recover than the balanced trees. Therefore this will not be commented on again in this section and instead I shall focus discussion on the relative performance of the different MSA-TB combinations.

Looking first at the set of simulations where the number of taxa was increased (figure 4.12) it is clear that MSA errors play a key role in the fallibility of each method of TB. On average, for both tree shapes, all three main TB methods were most accurate when given the true alignment as input. In addition, for the ML and NJ analyses involving both tree shapes, the two least accurate alignment methods according to SPS (Clustal and DIALIGN-TX) lead to the least accurate TB performance, on average in both the ML and NJ analyses. On the balanced tree PRANK was the most accurate MSA method and on the pectinate tree Muscle v4 was superior. This was also true for the ML and NJ analyses for both trees. On the balanced tree PRANK was best and very close to the true alignment (PBRs of 1.00 for both in ML, and PBRs of 0.99 for both using NJ) with Muscle v4 and MAFFT coming in second and third, with the other MSA methods trailing behind. For the pectinate tree Muscle v4 was better for both the ML (followed closely by MAFFT) and NJ (followed closely by PRANK) analyses with the other methods performing worse. For the MP analyses the worst alignment methods did not lead to the worst TB performance, and Muscle v4 was actually one of the worst methods on the balanced tree. Again PRANK was best on the balanced tree but MAFFT was slightly better than Muscle v4 on the pectinate tree.

The FST method is worthy of note since it performed better than any of the NJ or MP based analyses, even those that were performed using the true alignment. However, for the ML analyses the FST method was beaten by PRANK, MAFFT and Muscle v4 on the balanced tree and only performed as well as the worst method (Clustal) on the pectinate tree. The fact that FST performs better than any MSA for MP and NJ but as bad as the worst MSA for ML neatly underlines that ML is the superior inference method as the number of taxa is increased and is the least susceptible to the effects of

alignment errors. This is, perhaps, more noticeable for the balanced tree where even the worst ML inference method (DIALIGN-TX + ML) has a PBR of ≈0.99 when there are 256 taxa, compared to ProbConsRNA and Muscle v3 having PBRs ≈0.97 for MP (failing to recover ~ 5 branches on average), and Clustal having a PBR ≈0.86 for NJ (failing to recover ~35 branches on average). This comparison looks more convincing when one considers that PRANK, the best performer for the MP

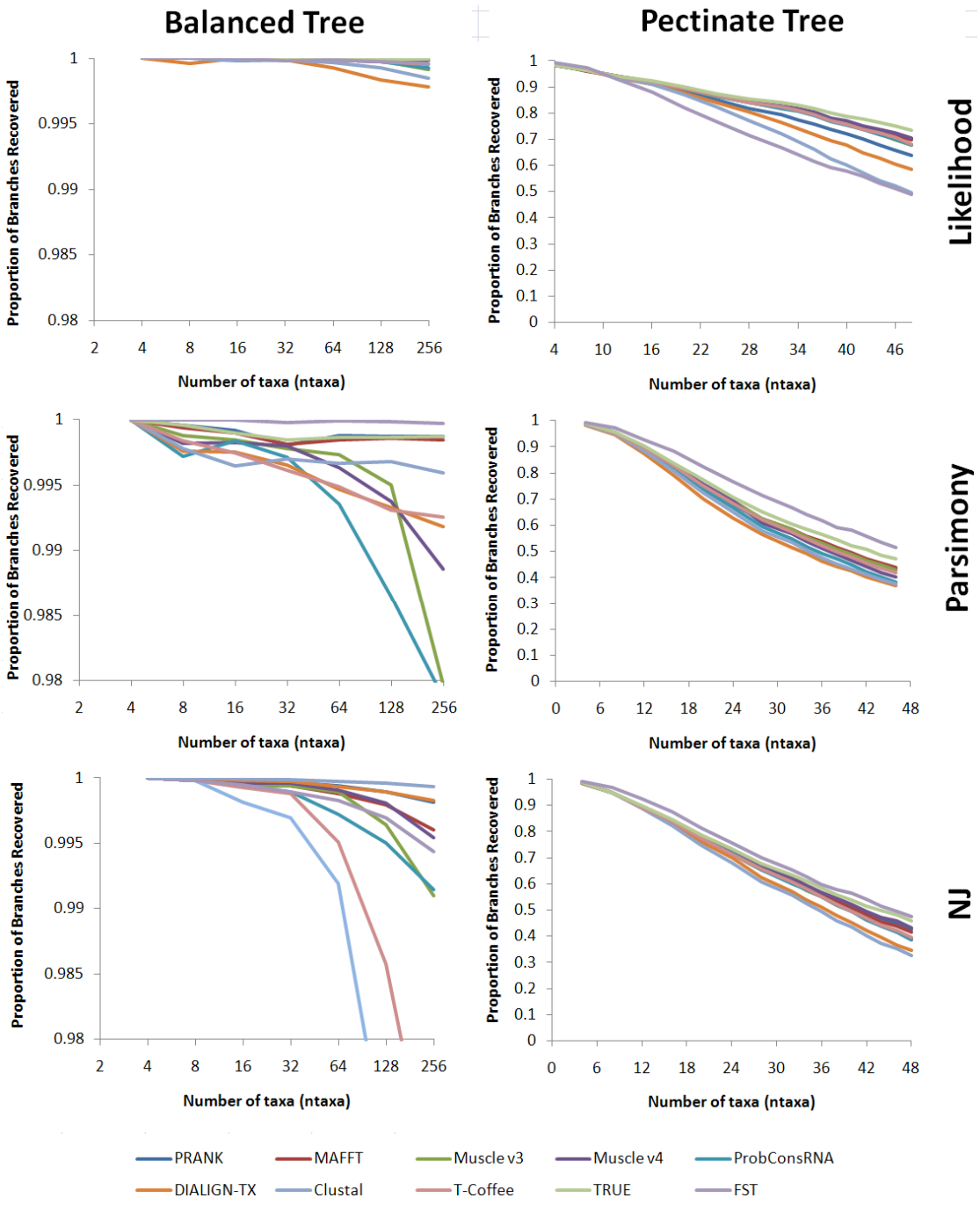


Figure 4.12: MSA+TB Accuracy when Number of Taxa Increases

and NJ analyses, had PBRs > 0.99 which is not much different to the worst ML based score.

In the second set of simulations, where sequence divergence increased (figure 4.13), it is NJ that performs somewhat differently to the other two TB methods. For both ML and MP the relative ranking of the analyses based on the different MSA methods is very similar. For example, on the balanced tree, T-Coffee, MAFFT, Muscle v3 then ProbConsRNA perform worst for both ML and

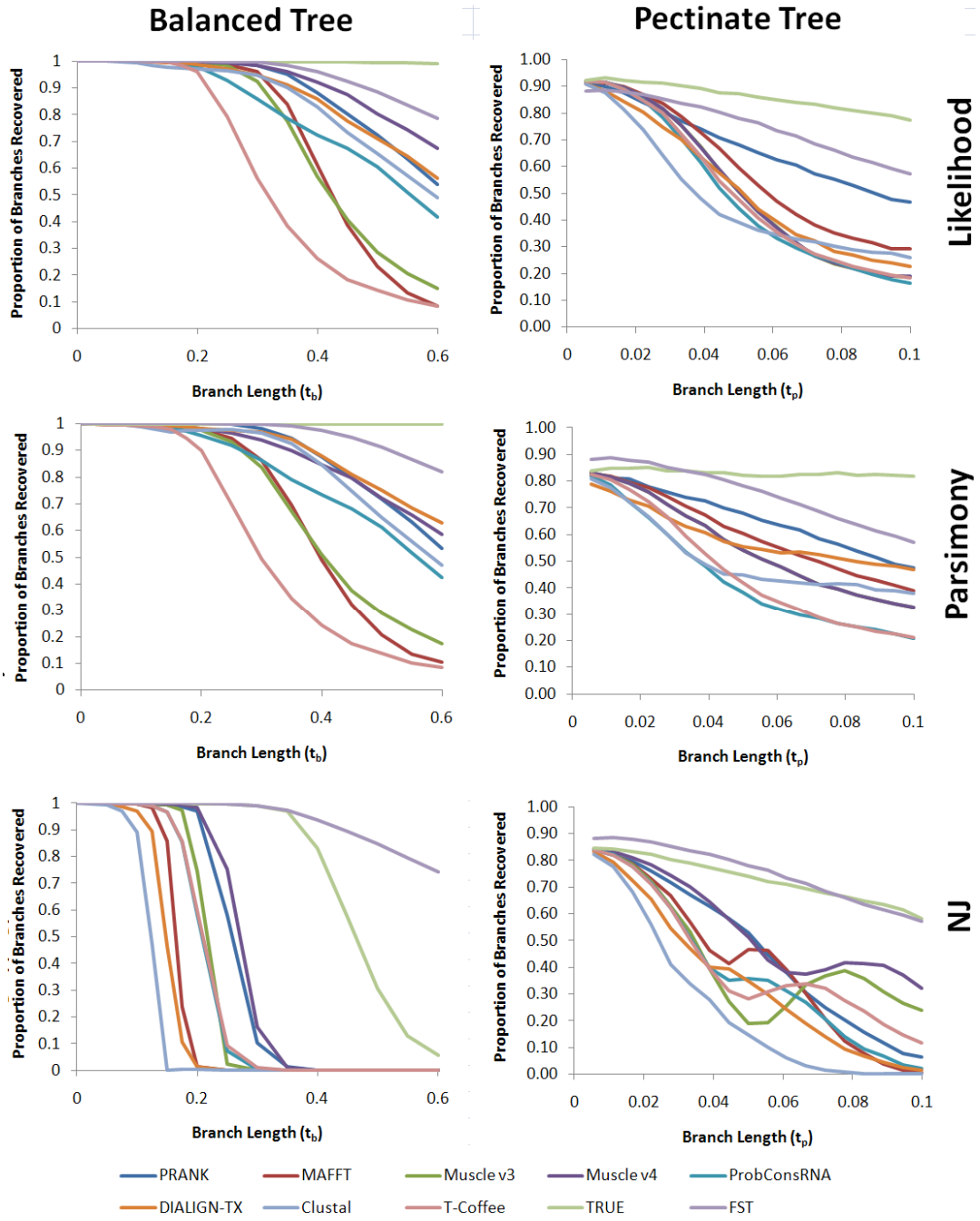


Figure 4.13: MSA+TB Accuracy when Sequence Divergence Increases

MP, and PRANK, Muscle v4 and DIALIGN-TX were better. Therefore, I shall discuss these two TB methods first before moving on to NJ and FST.

Interestingly, the worst MSA method (Clustal for both trees by both TC and SPS) is ranked solidly in the middle for the ML and MP analyses which is clear proof that the *type* of errors that MSA programs make in certain scenarios can be far more important than the *amount* of errors. Another clear example of this principle is that, despite the fact that PRANK had better TC and worse SPS than Muscle v4 (for both tree shapes in both the ML and MP analyses) when divergence was greatest (t_B = 0.6 and t_P = 0.1) Muscle v4 performed better on the balanced tree and PRANK performed better on the pectinate tree. When divergence is high this can be explained as follows. On pectinate trees there are very many "deep" pairwise sequence relationships that pass through the root, and so it is highly probable that a large number of parallel insertions and deletions will have occurred on the different lineages. This obviously favours an algorithm that deals with insertions and deletions correctly, such as that implemented in PRANK. Conversely, on the balanced tree there are many relatively "shallow" pairwise relationships which negates this benefit somewhat.

The NJ analyses in this set of simulations outline the main problem with distance based methods. When sequence divergence reaches a critical point pairwise distance estimation becomes inaccurate then impossible. On the balanced tree each MSA+NJ method collapses, one after the other, until no branches at all are successfully recovered. When t_B = 0.15, the SPS of the 8 MSA methods was 0.46, 0.58, 0.65, 0.70, 0.71, 0.73, 0.78 and 0.80 for Clustal, DIALIGN-TX, MAFFT, T-Coffee, ProbConsRNA, Muscle v3, PRANK and Muscle v4 respectively. This order exactly matches the PBR ranking of the different MSA methods when one looks at the order in which the PBR scores of the 8 methods deteriorate as t_B increases (bottom left of figure 4.13). This is perhaps no surprise as it has been shown before that alignment accuracy can heavily bias the pairwise distance estimation process, particularly those based on the normal greedy progressive alignment algorithms (Rosenberg 2005a). For the pectinate trees the NJ based analyses also performed worse than either of the ML or MP analyses. In particular, alignment errors resulted in the distance estimates becoming inconsistent for the Muscle, MAFFT, T-Coffee and ProbConsRNA analyses which lead to a non-monotonic decrease

in performance of these methods. Once again the FST method performed very well. When the true alignment is excluded from the comparisons, the FST method was better than any of the eight MSA-TB combinations, for both tree shapes, both by average PBR value and by how often the tree was most accurate (table 4.2, comparison C3, and figure 4.13).

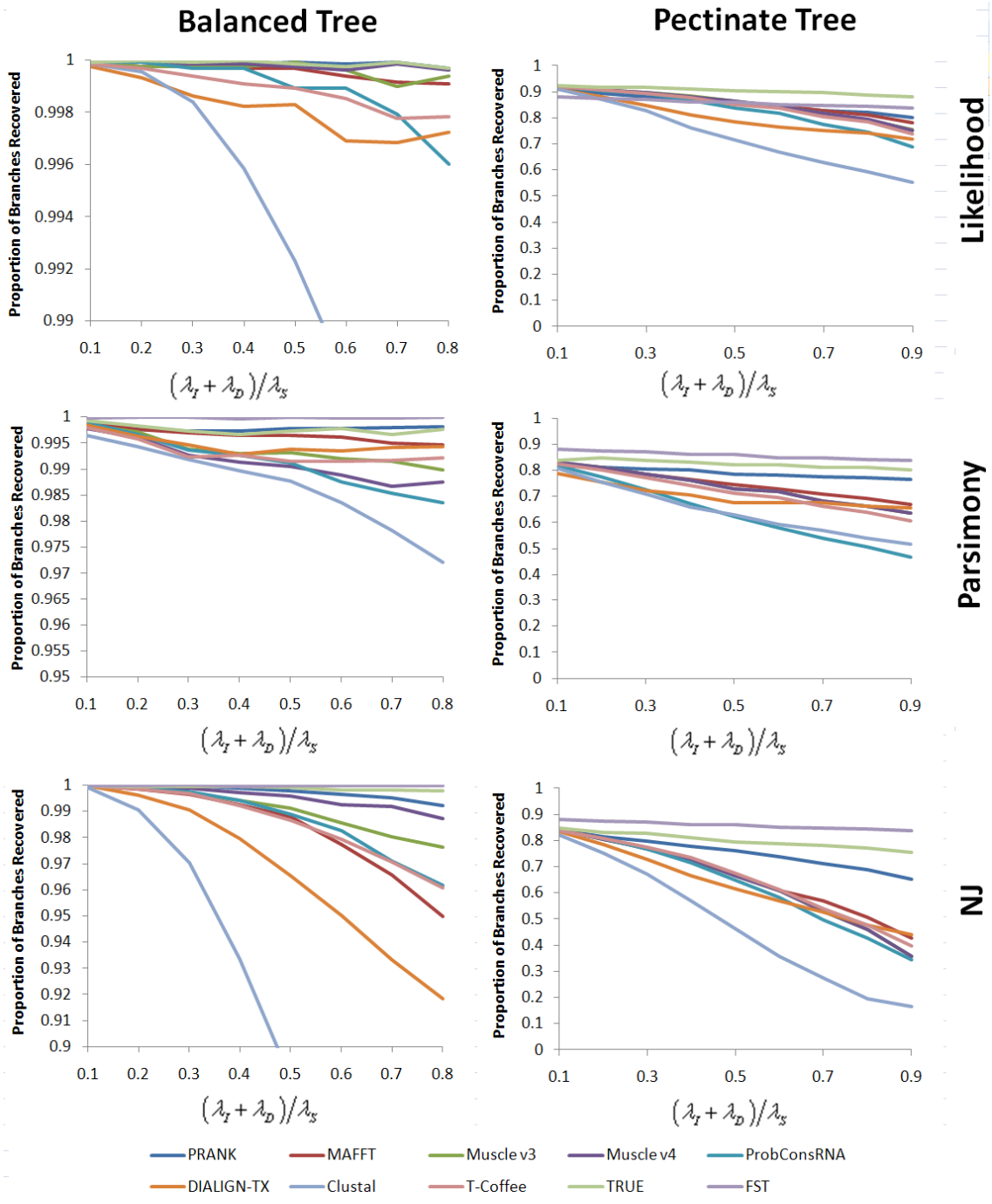


Figure 4.14: MSA+TB Accuracy when Indel Rate Increases

As before, alignment accuracy was important in the third set of simulations (figure 4.14) where the indel rate was increased. The worst alignment method for this set of simulations (Clustal) was worst in 5 of the 6 MSA-TB combinations and, again, the true alignment consistently yielded the most accurate trees with each TB method (table 4.2). In general, PRANK was the most accurate MSA method (figure 4.4) and it was also the most accurate method on the balanced tree, and for the MP analyses of the pectinate tree. However, Muscle v4 performed better in the ML and NJ analyses on the pectinate tree.

The FST method was more accurate than any of the MP or NJ analyses on either tree, including those that involved the true alignment. For ML analyses, FST is only outperformed by the true alignment for the pectinate tree when the indel rate is high. On the balanced tree FST performs equally as well as Muscle v4, PRANK and the true alignment (table 4.2). Once again ML seems least susceptible to alignment errors overall, with all MSA methods yielding more accurate trees than with MP or NJ.

All MSA methods performed well for the fourth set of simulations, generating alignments that were very similar to the true alignment. As a result the performance of each TB method on the generated alignments was virtually identical to that of the true alignment. There was some variation in the accuracy of each TB method that depended on the MSA used, but the differences were small. Therefore, no figure is shown for these results as each panel would essentially just show many nearly superimposed lines that followed the shapes shown in figure 4.10.

In the final set of simulations sequence divergence increased while trees became increasingly nonultrametric as well (figure 4.15). The FST method performed worse than all other methods for this scenario on the pectinate tree. However, on the balanced tree, the FST method was better than the other NJ distance methods, including those involving the true alignment, and it was also better in the MP and ML analyses when the divergence was high. On the pectinate tree the true alignment yielded more accurate trees for all three TB methods, which demonstrates the importance of alignment accuracy once more.

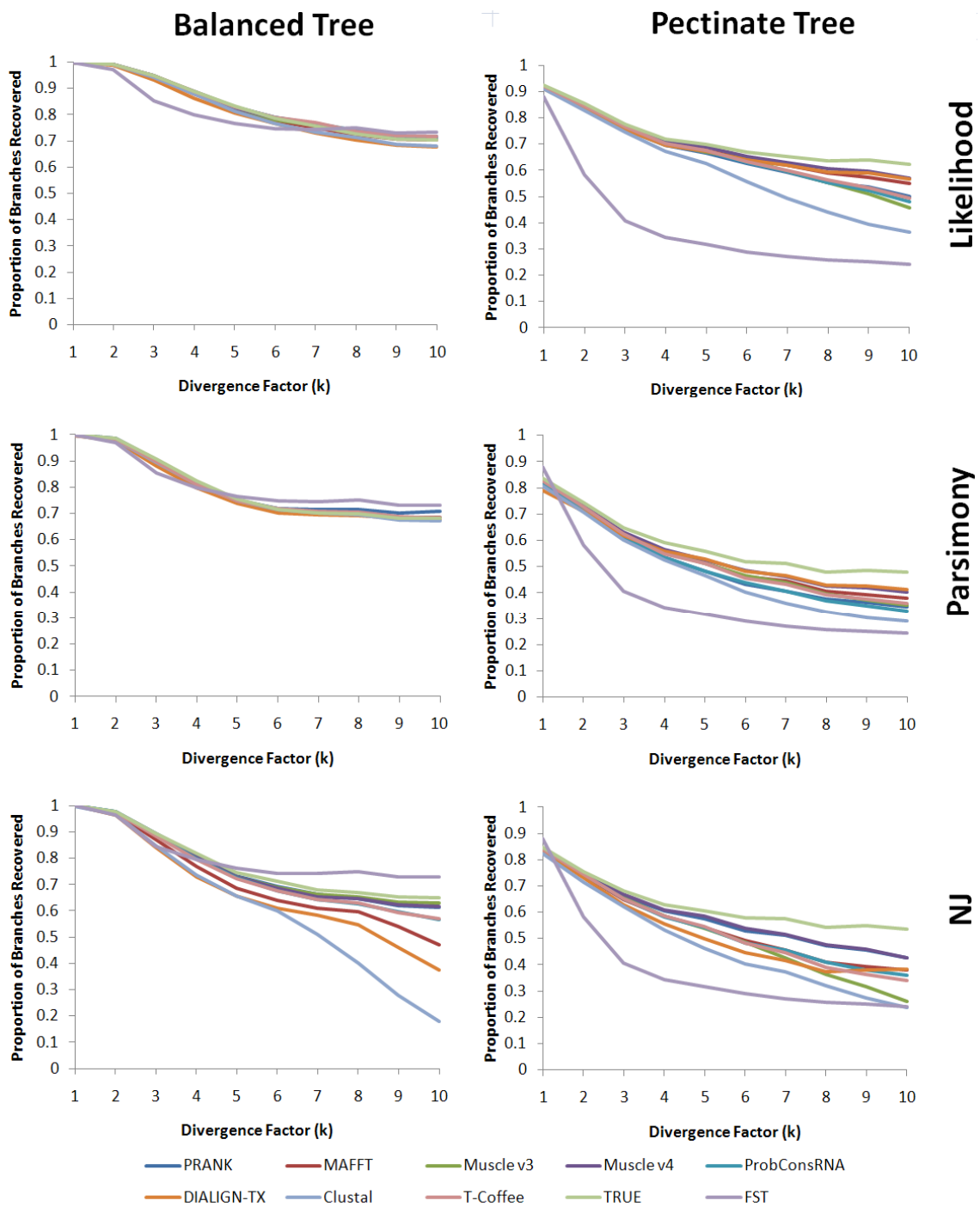


Figure 4.15: MSA+TB Accuracy when Molecular Clock is Violated and Divergence Increases

4.4 Discussion

4.4.1 Alignment Accuracy

Muscle v4 and PRANK were clearly the two most accurate MSA methods over the datasets tested. However, whilst PRANK consistently generated more accurate alignments from data generated on a balanced tree, it performed less well on the pectinate tree and Muscle v4 was more accurate. This change in relative performance between the two methods can be readily explained.

Firstly, PRANK's phylogeny aware gap placement algorithm is very sensitive to the choice of underlying guide tree. When no guide tree is provided, an approximate unrooted tree is calculated and mid-point rooting (MPR: Farris 1972) is used, where the root is placed at the mid-point of the longest distance between two taxa in the tree. MPR has been shown to be less accurate when there is only a single outgroup taxon as is the case for pectinate trees (Hess, De Moraes Russo 2007). Thus, one might expect PRANK's calculated guide tree to be less accurate for datasets generated on pectinate trees. This is indeed the case. For the basic balanced tree (figure 4.1a) the initial PRANK guide tree was rooted correctly 99% of the time, but only 23% of the time for the basic pectinate tree (figure 4.1b). Incorrect root placement will inevitably bias PRANK's choice of gap placement and lessen the impact of the program's advantage gained from handling indels correctly. A second reason that the algorithm will enjoy less of an advantage on pectinate tree shapes is that for a pectinate tree of a given length there are far fewer pairwise evolutionary paths that pass through the root of the tree compared to a balanced tree of a similar length. Therefore there is less chance of an insertion or deletion occurring on the path between any two given taxa, which minimises the benefits that the correct gap placement algorithm brings. Finally, when the influence of gap placement has less effect on an alignment's accuracy the scoring scheme used for matching pairs of nucleotides will become more important. PRANK's default option employs the HKY model with a transitition/transversion rate ratio of $\kappa = 2$. This is quite different from the true value of $\kappa = 4$ used in simulations, and ignoring rate heterogeneity is known to lead to underestimates of κ anyway (Yang 1996), further exacerbating the problem. This incorrect value will mean that nucleotide pairings that imply transversion substitutions would not have been penalised heavily enough by PRANK. It may be that Muscle's choice of match

scoring scheme – which is based on a scoring scheme derived by Chiaromonte *et al.* (2002) for use in the BLASTZ program (Schwartz et al. 2000) – is more appropriate for scoring the patterns of nucleotide substitution used in the simulations in this study.

4.4.2 Does Alignment Accuracy affect Tree Building Accuracy?

In this study I have shown that alignment accuracy can indeed have a large impact on phylogeny reconstruction. For all three methods, analysis of the true alignment yielded more accurate trees in every set of simulations. In addition, the analyses that involved the more accurate generated alignments tended to produce more accurate trees.

However, the *type* of alignment errors that an MSA program makes may be more important than the *number* of errors. For the sets of simulations where divergence was increased Clustal was the worst MSA method by any measure, on both trees, yet Clustal+TB was hardly ever the worst TB method. In addition, PRANK had by far the best TC on the balanced tree, and Muscle had a slightly better SPS. On the pectinate tree, the two methods had very similar TC but Muscle was much better by SPS. Despite this fact, Muscle v4 + ML performed best on the balanced tree and PRANK + ML performed best on the pectinate tree. This suggests that for TB on shallow balanced trees the most important factor in alignment accuracy is accurate alignment of closely related pairs of sequences, with the global alignment quality being less important. This is further evidenced by the fact that, when divergence was low but the number of taxa was high, all MSA+ML methods were very successful in recovering accurate trees. Conversely, when divergence was high on the pectinate tree, PRANK clearly provided better MSAs for use in TB – many deep pairwise relationships between divergent sequences allowing PRANK's superior algorithm to show its true worth. It seems clear that on large datasets with many divergent sequences the PRANK alignments would yield the most accurate trees.

4.4.3 Which Datasets Present the Hardest Problem for Multiple Sequence Alignment and Phylogeny Reconstruction Methods?

It is clear that increasing the insertion/deletion rate, the level of divergence or the number of taxa increases the difficulty of the MSA problem for both balanced and pectinate trees, but didn't always make the TB problem harder. Conversely, increasing the deviation from the molecular clock, and making the tree "harder" by including combinations of long and short internal branches, has little effect on the difficulty of the MSA problem when divergence is low. However, such "hard" trees were increasingly difficult to recover with every TB method, underlining that such datasets provide an easy challenge to MSA methods but a much harder problem to TB methods. The opposite trend was also observed. Raising the indel rate when divergence was low, or increasing the number of taxa, drastically increased the complexity of the MSA problem but had very little effect on ML TB accuracy on the balanced tree which demonstrates that there are also types of dataset where alignment is hard and TB can be relatively easy. A common thread among all simulations is that pectinate trees with "deep" phylogenies are far harder to recover than balanced trees.

4.4.4 Which Method of Phylogeny Reconstruction is Best?

Perhaps the clearest result of this study is that ML outperformed MP and NJ quite substantially. Firstly, the best MSA+TB method was always based on ML and, secondly, the worst MSA+ML inference method was nearly always better than the best MSA+MP or MSA+NJ inference method. This stems from the fact that ML seems far more resistant to the detrimental effects of MSA errors from all the tested MSA methods. Indeed, in almost every set of simulations it is immediately apparent that the errors that some MSA programs make can severely bias MP and NJ inference – especially the distance based methods which suffer hugely when alignment accuracy decreases.

Deserved of special mention is the FST method. Firstly, TB based on FST+NJ outperformed any MSA+NJ method in the first three sets of simulations. This is very impressive given that the FST method was given absolutely no information about the true substitution parameters or structure of the alignment, yet still outperformed DNADIST+NJ (referred to as NJ) which was given the correct

values for *R* and *CV*, and still had worse PBRs even when the distance matrix generated from the true alignment was used! Furthermore, when alignment accuracy was low because sequence divergence was high the FST method was the best performing on both tree shapes – even outperforming the best MSA+ML inference methods, and beaten only by ML on the true alignment. These successes suggest great promise for the FST approach to TB. However, the fact that the method performed relatively poorly in the experiments where the molecular clock did not hold suggest that there are further development avenues to be pursued before the method can compete consistently at the highest level.

4.4.5 Comparison to Other Studies

Ogden and Rosenberg (2006) were among the first to attempt to quantify the effect of tree-shape and alignment accuracy on phylogenetic inference. They concluded that as alignment error increased the accuracy of reconstructed topologies decreased. They also suggested that this was more pronounced for pectinate trees than for balanced ultrametric trees where alignment accuracy had little average effect. One problem with their study is that they simulated data using MySSP which does not update sequence lengths when insertions and deletions take place along a branch. Since they used a deletion rate that was 2.5 times the insertion rate this meant that on the long branches of a pectinate tree the sequence length should be getting smaller and smaller along the branch. Thus, the overall indel rate should also get smaller as a result because it should properly be formulated as a sum over sites in the sequence. However, MySSP only updates indel rates at interior nodes. This has the effect on long branches that the insertion and deletion rate per site in the sequence actually increases along the branch which was not their intention (figure 4.16). This may have contributed to the difficulty in phylogeny reconstruction that they observed and attributed to the pectinate tree shape. However, in this study I have confirmed that the pectinate tree shape is indeed inherently harder by using a simulation program that does ensure indel rates are kept at the desired levels.

Another conclusion drawn by Ogden and Rosenberg (2006) was that whilst alignment accuracy was related to tree building accuracy on average, there was no real relationship on a dataset-to-dataset basis. This conclusion appears to have be an artefact from only having implemented one

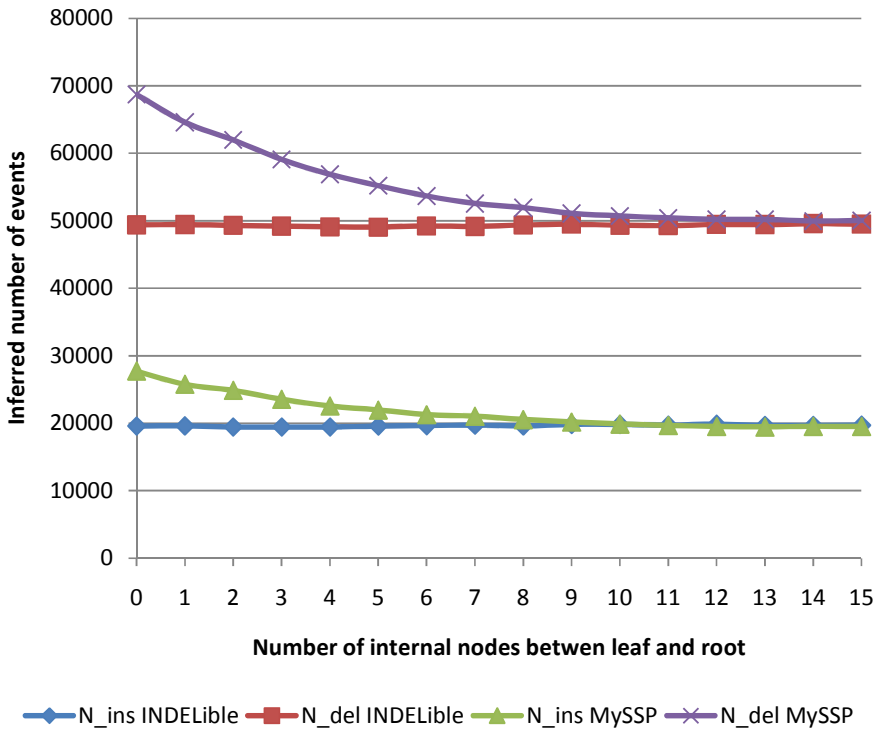


Figure 4.16: Comparison of the number of indel events in INDELible and MySSP

I followed Ogden and Rosenberg (2006) and created 1000 replicate datasets on a 16 taxa pectinate tree with maximum pairwise distance of 2.0 (tree depth of 1.0), with insertions and deletions having rates of 1 per 100 and 1 per 40 substitutions respectively, and a mean length of 4 for both. To calculate the number of events I compared characters from the root sequence with characters from each leaf sequence in turn. If the root has a blank and the leaf does not then I classed this as an insertion, and if the leaf has a blank and the root does not then I classed that as a deletion. Therefore the comparison ignores deleted insertions. The counts for each program were then normalised by dividing by the mean indel length. Although the calculations are not accurate since deleted insertions are ignored they do serve to show how the number of insertion and deletion events increase depending on the number of nodes along a given evolutionary path. This is not what Ogden and Rosenberg (2006) intended and may have contributed to the relative difficulty they observed in reconstructing the phylogenies of datasets generated on pectinate trees. In comparison, the number of indel events that occur along a branch in INDELible remain roughly constant and are independent of the number of internal nodes. This phenomenon was first observed by Strope *et al* (2009).

MSA program. It makes sense that different replicate datasets may be harder or easier to align, or to rebuild trees from, and we should not necessarily expect any absolute correlation between the accuracy of the two processes for a fixed MSA method on different datasets. However, in this study I have shown that for a given fixed dataset there is a strong correlation between the accuracy of a particular MSA method and the accuracy of the tree inferred, regardless of the tree building method applied downstream.

Other studies have also suggested this correlation (Wang et al. 2009) but failed to highlight examples where MSA or TB was particularly important or less important as I did in this study. In addition, Wang *et al* omitted PRANK from their study and Muscle v4 was not available then. The absence of these two accurate MSA methods (the current state-of-the-art in the field), combined with their choice of simulating data using the conceptually incorrect program ROSE, means that their conclusions may not be as valid or relevant as they could have been.

4.5 Conclusion

This study has shown that alignment accuracy is generally important to the accuracy of different methods of phylogenetic reconstruction, but that there are certain types of dataset where the choice of MSA or TB method may be less important. I have shown that balanced trees are easier than pectinate trees to recover, and that Muscle v4 and PRANK are currently the best performing MSA methods for TB, but still leave room for improvement. When sequence divergence was high PRANK's superior algorithm gave better performance when resolving deep phylogenies, but when phylogenies were relatively shallow the choice of substitution scoring scheme used could be more important. The novel FST method does not require alignment and on divergent datasets showed great promise, yet it was still outperformed by ML on the true alignment. The fact that the true alignment yielded the most accurate trees for each method underlines once again the critical role that alignment accuracy plays in phylogeny reconstruction and I suggest that further improvements in alignment quality can only help increase the accuracy of downstream phylogeny reconstruction.

Closing Remarks

In this thesis I have presented a novel simulation program, INDELible, that combines many features only previously found in other disparate programs, or not at all, and allows new and exciting avenues of research to be pursued. At the time of writing (Sep 2010) INDELible has been downloaded by more than 250 unique users at universities and institutes from some 40 countries and has already been used in other published studies (Albayrak et al. 2010; Fletcher, Yang 2010; Kim, Sinha 2010; Penn et al. 2010a; Penn et al. 2010b; Schwarz et al. 2010) with many more works in progress around the world (pers. comm.).

The program continues to evolve (currently in v1.04) as I have added new features at the request of colleagues from around the world and I expect this trend to continue as I wish the program to be a useful tool for the phylogenetics community. In particular, INDELible has demonstrated the potential of the jump-chain formulation of markov chain simulations which can be adapted in the future to allow simulation of complex content-dependent substitution models. This is something that is not possible using the traditional transition probability matrix employed in all other simulation programs unless one makes unrealistic simplifying assumptions such as discretising time (Varadarajan et al. 2008). Given the ever present drive in Phylogenetics for increased biological realism in statistical inference methods the ability to simulate under such models will be important in the years to come. Another factor that makes INDELible particularly useful for researchers is that it can be used to simulate very large datasets with great speed compared to other programs because of the unique design of the program. In particular INDELible can simulate many replicate megabase sized datasets (with indels) in minutes or hours whereas every other simulation program I have tested takes days or weeks to simulate under the same conditions. Real-life datasets grow in size year on year – indeed we are on the cusp of an era of Phylogenetics that could potentially use whole genomes as datasets and so the ability to simulate large datasets in a timely fashion in order to test such methods will only become more important as time goes on.

I have used INDELible to show categorically, for the first time, that the widely used branch-site test of positive selection is unaffected by insertions and deletionswhen the alignment is correct, but that alignment errors mislead the test to generate an unacceptable number of false positives, thus highlighting the need for improved alignment methods in studies of positive selection. It is my hope that INDELible will be used for other novel projects such as this in the future.

Literature Cited

Abascal, F, D Posada, R Zardoya. 2007. MtArt: A new Model of amino acid replacement for Arthropoda. Molecular Biology and Evolution 24:1-5.

Adachi, J, M Hasegawa. 1996. MOLPHY version 2.3: programs for molecular phylogenetics based on maximum likelihood. Computer Science Monographs of Institute of Statistical Mathematics 28:1-150.

Adachi, J, PJ Waddell, W Martin, M Hasegawa. 2000. Plastid genome phylogeny and a model of amino acid substitution for proteins encoded by chloroplast DNA. Journal of Molecular Evolution 50:348-358.

Akaike, H. 1974. A new look at the statistical model identification. IEEE Transactions on Automatic Control 19:716-723.

Albayrak, A, HH Otu, UO Sezerman. 2010. Clustering of protein families into functional subtypes using Relative Complexity Measure with reduced amino acid alphabets. BMC Bioinformatics.

Allen, BL, M Steel. 2001. Subtree Transfer Operations and Their Induced Metrics on Evolutionary Trees. Annals of Combinatorics 5:1-15.

Altschul, SF. 1991. Amino Acid Substitution Matrices from an Information Theoretic Perspective. Journal of Molecular Biology 219:555-565.

Altschul, SF. 1998. Generalized Affine Gap Costs for Protein Sequence Alignment. Proteins: Structure, Function, and Genetics 32:88-96.

Altschul, SF, R Bundschuh, R Olsen, T Hwa. 2001. The estimation of statistical parameters for local alignment score distributions. Nucleic Acids Research 29:351-361.

Altschul, SF, W Gish, W Miller, E Myers, D Lipman. 1990. Basic local alignment search tool. Journal of Molecular Biology 215:403-410.

Altschul, SF, T Madden, A Schaffer, J Zhang, Z Zhang, W Miller, DJ Lipman. 1997. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research 25:3389-3402.

Anisimova, M, C Kosiol. 2009. Investigating protein-coding sequence evolution with probabilistic codon substitution models. Molecular Biology and Evolution 26:255-271.

Anisimova, M, R Nielsen, Z Yang. 2003. Effect of recombination on the accuracy of the likelihood method for detecting positive selection at amino acid sites. Genetics 164:1229-1236.

Anisimova, M, Z Yang. 2007. Multiple hypothesis testing to detect lineages under positive selection that affects only a few sites. Molecular Biology and Evolution 24:1219-1228.

Armougom, F, S Moretti, O Poirot, S Audic, P Dumas, B Schaeli, V Keduas, C Notredame. 2006. Expresso: automatic incorporation of structural information in multiple sequence alignments using 3D-Coffee. Nucleic Acids Research 34:W604-608.

Arndt, PF, T Hwa. 2004. Regional and time-resolved mutation patterns of the human genome. Bioinformatics 20:1482-1485.

Atteson, K. 1997. The performance of the neighbor-joining method of phylogeny reconstruction. Mathematical hierarchies and biology. Providence (RI): American Mathematical Society. p. 133-147.

Averof, M, A Rokas, K Wolfe, P Sharp. 2000. Evidence for a high frequency simultaneous doublet-nucleotide substitutions. Science 287:1283-1286.

Bahr, A, J Thompson, J Thierry, O Poch. 2001. BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. Nucleic Acids Research 29:323-326.

Baldi, P, Y Chauvin, T Hunkapiller, M McClure. 1994. Hidden markov models of biological primary sequence information. Proceedings of the National Academy of Sciences of the United States of America 91:1059-1063.

Barzilay, R, L Lee. 2002. Bootstrapping Lexical Choice via Multiple Sequence Alignment. Conference on Empirical Methods in Natural Language Processing (EMNLP). Philadelphia: Association for Computational Linguistics. p. 164-171.

Bayzkin, G, F Kondrashov, AY Ogurtsov, S Sunyaev, AS Kondrashov. 2004. Positive selection at sites of multiple amino acid replacements since rat-mouse divergence. Nature 429:558-562.

Benner, SA, MA Cohen, GH Gonnet. 1993. Empirical and structural models for insertions and deletions in the divergent evolution of proteins. Journal of Molecular Biology 229:1065-1082.

Berger, M, P Munson. 1991. A novel randomized iterative strategy for aligning multiple protein sequences. Bioinformatics 7:479-484.

Bishop, M, E Thompson. 1986. Maximum likelihood alignment of DNA sequences. Journal of Molecular Biology 190:159-165.

Blake, RD, ST Hess, J Nicholson-Tuell. 1992. The influence of nearest neighbors on the rate and pattern of spontaneous point mutations. Journal of Molecular Evolution 34:189 – 200.

Blanchette, M, ED Green, W Miller, D Haussler. 2004. Reconstructing large regions of an ancestral mammalian genome in silico. Genome Research 14:2412–2423.

Blanquart, S, N Lartillot. 2006. A Bayesian Compound Stochastic Process for Modeling Nonstationary and Nonhomogeneous Sequence Evolution. Molecular Biology and Evolution 23:2058-2071.

Bofkin, L, N Goldman. 2006. Variation in Evolutionary Processes at Different Codon Positions. Molecular Biology and Evolution 24:513-521.

Bordo, D, P Argos. 1991. Suggestions for "safe" residue substitutions in site-directed mutagenesis. Journal of Molecular Biology 217:721-729.

Bradley, RK, I Holmes. 2007. Transducers: an emerging probabilistic framework for modeling indels on trees. Bioinformatics 23:3258-3262.

Britten, RJ, L Rowen, J Williams, RA Cameron. 2003. Majority of divergence between closely related DNA samples is due to indels. Proceedings of the National Academy of Sciences of the United States of America.

Bruno, WJ, ND Socci, AL Halpern. 2000. Weighted Neighbor Joining: A Likelihood-Based Approach to Distance-Based Phylogeny Reconstruction. Molecular Biology and Evolution 17:189-197.

Cao, Y, J Adachi, A Janke, S Pääbo, M Hasegawa. 1994. Phylogenetic relationships among eutherian orders estimated from inferred sequences of mitochondrial proteins: Instability of a tree based on a single gene. Journal of Molecular Evolution 39:519-527.

Carapelli, A, P Liò, F Nardi, E van der Wath, F Frati. 2007. Phylogenetic analysis of mitochondrial protein coding genes confirms the reciprocal paraphyly of Hexapoda and Crustacea. BMC Evolutionary Biology 7(Suppl 2):S8.

Cartwright, RA. 2005. DNA Assembly With Gaps (Dawg): Simulating Sequence Evolution. Bioinformatics 21 Suppl. 3:iii31-iii38.

Cartwright, RA. 2006. Logarithmic gap costs decrease alignment accuracy. BMC Bioinformatics 7:527.

Cartwright, RA. 2009. Problems and Solutions for Estimating Indel Rates and Length Distributions. Molecular Biology and Evolution 26:473-480.

Chang, MSS, SA Benner. 2004. Empirical analysis of protein insertions and deletions determining parameters for the correct placement of gaps in protein sequence alignments. Journal of Molecular Biology 341:617-631.

Chiaromonte, F, VB Yap, W Miller. 2002. Scoring pairwise genomic sequence alignment. Pacific Symposium on Biocomputing 7:115-126.

Claverie, J. 1993. Detecting frame shifts by amino acid sequence comparison. Journal of Molecular Biology 234:1140-1157.

Cortes, C, P Haffner, M Mohri. 2004. Rational Kernels: Theory and Algorithms. Journal of Machine Learning Research 5:1035-1062.

Covington, M. 1996. An algorithm to align words for historical comparison. Computational Linguistics 22:481-496.

Covington, M. 1998. Alignment of multiple languages for historical comparison. 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics. p. 275-279.

Crespi, B, K Summers, S Dorus. 2007. Adaptive evolution of genes underlying schizophrenia. Proceedings of the Royal Society Biological Sciences Series B 274:2801-2810.

Crick, F. 1958. On Protein Synthesis. Symposium of the Society for Experimental Biology 12:138-163.

Crick, F. 1970. Central Dogma of Molecular Biology. Nature 227:561-563.

Dang, CC, SQ Le, SV Le. 2009. Influenza-specific Amino Acid Substitution Model. International Conference on Knowledge and Systems Engineering. p. 19-25.

Darwin, C. 1859. On the Origin of Species by Means of Natural Selection. London: J. Murray.

Day, WH. 1987. Computational complexity of inferring phylogenies from dissimilarity matrices. Bulletin of Mathematical Biology 49:461-467.

Dayhoff, MO, R Schwartz, BC Orcutt. 1978. A model of evolutionary change in protein. Atlas of Protein Sequences and Structure 5:345-352.

DeLong, E, D DeLong, D Clarke-Pearson. 1988. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. Biometrics 44:837-845.

Dimmic, MW, JS Rest, DP Mindell, RA Goldstein. 2002. RtREV: An amino acid substitution matrix for inference of retrovirus and reverse transcriptase phylogeny. Journal of Molecular Evolution 55:65-73.

Do, C, M Mahabhashyam, M Brudno, S Batzoglou. 2005. ProbCons: Probabilistic consistency-based multiple sequence alignment. Genome Research 15:330-340.

Doron-Faigenboim, A, T Pupko. 2007. A Combined Empirical and Mechanistic Codon Model. Molecular Biology and Evolution 24:388-397.

Durbin, R, S Eddy, A Krogh, G Mitchison. 1998. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge, UK: Cambridge University Press.

Duret, L. 2002. Evolution of synonymous codon usage in metazoans. Current Opinion in Genetics & Development 12:640-649.

Dwivedi, B, SR Gadagkar. 2009. Phylogenetic inference under varying proportions of indel-induced alignment gaps. BMC Evolutionary Biology 9:211.

Eddy, S. 1995. Multiple Alignment Using Hidden Markov Models. Proceedings of the Thirs International Conference of Intelligent Systems for Molecular Biology. Menlo Park: AAAI Press. p. 114-120.

Eddy, S. 2004. What is dynamic programming? Nature Biotechnology 22.

Edgar, RC. 2004. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research 32:1792-1797.

Edgar, RC. 2009. Optimizing substitution matrix choice and gap parameters for sequence alignment. BMC Bioinformatics 10:396.

Edgar, RC. 2010. Quality measures for protein alignment benchmarks. Nucleic Acids Research 38:2145-2153.

Ehrlich, M, RY Wang. 1981. 5-Methylcytosine in eukaryotic DNA. Science 212:1350 –1357.

Fan, Y, Q Shi, J Chen, W Wang, H Pang, J Tang, S Tao. 2008. The rates and patterns of insertions, deletions and substitutions in mouse and rat inferred from introns. Chinese Science Bulletin 53:2813-2819.

Farris, J. 1972. Estimating phylogenetic trees from distance matrices. American Naturalist 106:645-667.

Felsenstein, J. 1978. The Number of Evolutionary Trees. Systematic Zoology 27:27-33.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of Molecular Evolution 17:368-376.

Felsenstein, J. 1985. Confidence limit on phylogenies: An approach using the bootstrap. Evolution 39:783-791.

Felsenstein, J. 2010. PHYLIP (Phylogeny Inference Package) version 3.69. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

Felsenstein, J, GA Churchill. 1996. A Hidden Markov Model Approach to Variation Among Sites in Rate of Evolution. Molecular Biology and Evolution 13:93-104.

Feng, D, R Doolittle. 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. Journal of Molecular Evolution 60:351-360.

Fisher, RA. 1918. The Correlation Between Relatives on the Supposition of Mendelian Inheritance. Proceedings of the Royal Society of Edinburgh 52:399-433.

Fitch, W. 1966. An improved method of testing for evolutionary homology. Journal of Molecular Biology 16:9-16.

Fitch, W. 1971. Toward Defining Course of Evolution - Minimum Change for a Specific Tree Topology. Systematic Zoology 20:406-416.

Fitch, W. 1986. The estimate of total nucleotide substitutions from pairwise differences is biased. Philosophical Transactions of the Royal Society of London B Biological Sciences 312:317-324.

Fitch, W, E Margoliash. 1967a. Construction of Phylogenetic Trees. Science 155:279-284.

Fitch, W, E Margoliash. 1967b. A method for estimating the number of invariant amino acid coding positions in a gene, using cytochrome c as a model case. Biochemical Genetics 1:65-71.

Fleissner, R, D Metzler, A von Haeseler. 2005. Simultaneous Statistical Multiple Alignment and Phylogeny Reconstruction. Systematic Biology 54:548-561.

Fletcher, R. 1987. Practical Methods of Optimization. New York: Wiley.

Fletcher, W, Z Yang. 2009. INDELible: A flexible simulator of biological sequence evolution. Molecular Biology and Evolution 26:1879-1888.

Fletcher, W, Z Yang. 2010. The Effect of Insertions, Deletions, and Alignment Errors on the Branch-Site Test of Positive Selection. Molecular Biology and Evolution 27:2257-2267.

Freeland, SJ, LD Hurst. 1998. The Genetic Code is One in a Million. Journal of Molecular Evolution 47:238-248.

Freeland, SJ, T Wu, N Keulmann. 2003. The Case for an Error Minimizing Standard Genetic Code. Origins of Life and Evolution of the Biosphere 33:457-477.

Galtier, N, M Gouy. 1998. Inferring pattern and process: maximum-likelihood implementation of a nonhomogeneous model of DNA sequence evolution for phylogenetic analysis. Molecular Biology and Evolution 15:871-879.

Gascuel, O. 1997. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. Molecular Biology and Evolution 14:685-695.

Gelfand, AE, AFM Smith. 1990. Sampling-based approaches to calculating marginal densities. Journal of the American Statistical Association 85:398-409.

Gentles, AJ, S Karlin. 2001. Genome-scale compositional comparisons in eukaryotes. Genome Research 11:540-546.

Geyer, CJ. 1991. Markov chain Monte Carlo maximum likelihood. In: EM Keramidas, editor. Computing Science and Statistics: Proceedings of the 23rd Symposium of the Interface: Interface Foundation, Fairfax Station, VA. p. 156-163.

Gilks, WR, S Richardson, DJ Spiegelhalter, editors. 1996. Markov Chain Monte Carlo in Practice. London: Chapman & Hall.

Gill, P, W Murray, M Wright. 1981. Practical Optimization. London: Academic Press.

Gillespie, DT. 1977. Exact stochastic simulation of coupled chemical reactions. Journal of Physical Chemistry 81:2340-2361.

Gojobori, T, WH Li, D Graur. 1982. Patterns of nucleotide substitution in pseudogenes and functional genes. Journal of Molecular Evolution 18:360-369.

Golding, B, J Felsenstein. 1990. A maximum likelihood approach to the detection of selection from a phylogeny. Journal of Molecular Evolution 31:511-523.

Golding, G. 1983. Estimates of DNA and protein sequence divergence: an examination of some assumptions. Molecular Biology and Evolution 1:125-142.

Goldman, N. 1993. Statistical tests of models of DNA substitution. Journal of Molecular Evolution 36:182-198.

Goldman, N, JP Anderson, AG Rodrigo. 2000. Likelihood-Based Tests of Topologies in Phylogenetics. Systematic Biology 49:652-670.

Goldman, N, S Whelan. 2000. Statistical Tests of Gamma-Distributed Rate Heterogeneity in Models of Sequence Evolution in Phylogenetics. Molecular Biology and Evolution 17:975-978.

Goldman, N, Z Yang. 1994. A codon-based model of nucleotide substitution for protein-coding DNA sequences. Molecular Biology and Evolution 11:725-736.

Goldman, N, Z Yang. 2008. Statistical and computational challenges in molecular phylogenetics and evolution. Philisophical Transactions of The Royal Society B 363:3889-3892.

Gonnet, GH, MA Cohen, SA Benner. 1992. Exhaustive matching of the entire protein sequence database. Science 256:1443-1445.

Gotoh, O. 1982. An improved algorithm for matching biological sequences. Journal of Molecular Biology 162:705-708.

Gotoh, O. 1996. Significant Improvement in Accuracy of Multiple Protein Sequence Alignments by Iterative Refinement as Assessed by Reference to Structural Alignments. Journal of Molecular Biology 264:823-838.

Grantham, R. 1974. Amino acid difference formula to help explain protein evolution. Science 185:862-864.

Gu, X, YX Fu, WH Li. 1995. Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. Molecular Biology and Evolution 12:546-557.

Gu, X, WH Li. 1995. The size distribution of insertions and deletions in human and rodent pseudogenes suggests the logarithmic gap penalty for sequence alignment. Journal of Molecular Evolution 40:464-473.

Guindon, S, AG Rodrigo, KA Dyer, JP Huelsenbeck. 2004. Modelling the site-specific variation of selection patterns along lineages. Proceedings of the National Academy of Sciences of the United States of America 101:12957-12962.

Haeckel, E. 1866. Generelle Morphologie der Organismen: Allgemeine Grundzüge der organischen Formen-Wissenschaft, mechanisch begründet durch die von Charles Darwin reformirte Descendenz-Theorie. Berlin: Georg Riemer.

Hall, BG. 2005. Comparison of the Accuracies of Several Phylogenetic Methods Using Protein and DNA Sequences. Molecular Biology and Evolution 22:792-802.

Hall, BG. 2008a. EvolveAGene 3: A DNA coding sequence evolution simulation program. Molecular Biology and Evolution 25:688-695.

Hall, BG. 2008b. How well does the HoT score reflect sequence alignment accuracy? Molecular Biology and Evolution 25:1576-1580.

Hanley, JA, BJ McNeil. 1983. A Method of Comparing the Areas under Receiver Operating Characteristic Curves Derived fom the Same Cases. Radiology 148:839-843.

Hasegawa, M, H Kishino. 1989. Confidence limits on the maximum-likelihood estimate of the hominoid tree from mitochondrial-DNA sequences. Evolution 43:672-677.

Hasegawa, M, H Kishino, T-A Yano. 1985. Dating of the human-ape splitting by a molecular clock of mitchondrial DNA. Journal of Molecular Evolution 22:160-174.

Hasegawa, M, A Rienzo, T Kocher, A Wilson. 1993. Toward a more accurate time scale for the human mitochondrial DNA tree. Journal of Molecular Evolution 37:347-354.

Hasegawa, M, T-A Yano, H Kishino. 1984. A New Molecular Clock of Mitochondrial DNA and the Evolution of Hominoids. Proceedings of the Japan Academy Series B Physical and Biological Sciences 60:95-98.

Hastings, WK. 1970. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. Biometrika 57:97-109.

Hein, J, C Wiuf, B Knudsen, M Moller, G Wibling. 2000. Statistical alignment: computational properties, homology-testing and goodness-of-fit. Journal of Molecular Biology 302:265-279.

Henikoff, S, JG Henikoff. 1992. Amino acid substitution matrices from protein blocks. Proceedings of the National Academy of Sciences of the United States of America 89.

Hennig, W. 1950. Grundzüge einer Theorie der phylogenetischen Systematik. Berlin: Deutscher Zentralverlag.

Hennig, W. 1966. Phylogenetic Systematics. Urbana: University of Illinois Press.

Hess, PN, CA De Moraes Russo. 2007. An empirical test of the midpoint rooting method. Biological Journal of the Linnean Society 92:669-674.

Hess, ST, JD Blake, RD Blake. 1994. Wide variations in neighbor-dependent substitution rates. Journal of Molecular Biology 236:1022-1033.

Hillis, DM. 1995. Approaches for assessing phylogenetic accuracy. Systematic Biology 44:3-16.

Hillis, DM, JJ Bull, ME White, BM R., IJ Molineux. 1992. Experimental phylogenetics: generation of a known phylogeny. Science 255:589-592.

Holder, M, PO Lewis. 2003. Phylogeny estimation: traditional and Bayesian approaches. Nature Reviews Genetics 4:275-284.

Holmes, I, W Bruno. 2001. Evolutionary HMMs: a Bayesian approach to multiple alignment. Bioinformatics 17:803-820.

Holmquist, R, M Goodman, T Conry, J Czelusniak. 1983. The spatial distribution of fixed mutations within genes coding for proteins. Journal of Molecular Evolution 19:137-448.

Huang, X, W Miller. 1991. A time-efficient, linear-space local similarity algorithm. Advances in Applied Mathematics 12:337-357.

Huelsenbeck, JP. 1995. Performance of phylogenetic methods in simulation. Systematic Biology 44:17-48.

Huelsenbeck, JP, JJ Bull. 1996. A Likelihood Ratio Test to Detect Conflicting Phylogenetic Signal. Systematic Biology 45:92-98.

Huelsenbeck, JP, DM Hillis, R Jones. 1996a. Parametric bootstrapping in molecular phylogenetics: applications and performance. In: JDP Ferraris, S.R., editor. Molecular Zoology: Advances, Strategies, and Protocols. New York: Wiley-Liss. p. 19-45.

Huelsenbeck, JP, DM Hillis, R Nielsen. 1996b. A Likelihood-Ratio Test of Monophyly. Systematic Biology 45:546-558.

Huelsenbeck, JP, F Ronquist. 2001. MrBayes: Bayesian inference of phylogenetic trees. Bioinformatics 17:754-755.

Huelsenbeck, JP, F Ronquist, R Nielsen, JP Bollback. 2001. Bayesian inference of phylogeny and its impact on evolutionary biology. Science 294:2310-2314.

Hughey, R, A Krogh. 1995. SAM: Sequence alignment and modelling software system. Santa Cruz, CA: University of California.

Huxley, J. 1942. Evolution: The Modern Synthesis.

Hwang, DG, P Green. 2004. Bayesian Markov chain Monte Carlo sequence analysis reveals varying neutral substitution patterns in mammalian evolution. Proceedings of the National Academy of Sciences of the United States of America 101.

Ikemura, T. 1981. Correlation between the abundance of *Escherichia coli* transfer RNAs and the occurence of respective codons in its protein genes. Journal of Molecular Biology 146:1-21.

Ikemura, T. 1985. Codon usage and the tRNA content in unicellular and multicellular organisms. Molecular Biology and Evolution 2:13-34.

Jensen, JL, A-MK Pedersen. 2000. Probabilistic models of DNA sequence evolution with context dependent rates of substitution. Advances in Applied Probability 32:499-517.

Jones, DT, WR Taylor, JM Thornton. 1992. The rapid generation of mutation data matrices from protein sequences. Computer Applications in the BioSciences 8:275- 282.

Jukes, TH. 1987. Transitions, Transversions, and the Molecular Evolutionary Clock. Journal of Molecular Evolution 26:87-98.

Jukes, TH, CR Cantor. 1969. Evolution of protein molecules. In: MN Munro, editor. Mammalian Protein Metabolism. New York: Academic Press. p. 21-132.

Kalbfleisch, J. 1985. Statistical Inference. New York: Springer-Verlag.

Karlin, S, J Mrazek. 1997. Compositional differences within and between eukaryotic genomes. Proceedings of the National Academy of Sciences of the United States of America 94.

Karol, KG, RM McCourt, MT Cimino, CF Delwiche. 2001. The closest living relatives of land plants. Science 294:2351-2353.

Katoh, K, H Toh. 2008a. Improved accuracy of multiple ncRNA alignment by incorporating structural information into a MAFFT-based framework. BMC Bioinformatics 9:212.

Katoh, K, H Toh. 2008b. Recent developments in the MAFFT multiple sequence alignment program. Briefings in Bioinformatics 9:286-298.

Keller, I, D Bensasson, RA Nichols. 2007. Transition-Transversion Bias Is Not Universal: A Counter Example from Grasshopper Pseudogenes. PLoS Genetics 3:e22.

Kent, WJ, R Baertsch, A Hinrichs, W Miller, D Haussler. 2003. Evolution's cauldron: duplication, deletion, and rearrangement in the mouse and human genomes. Proceedings of the National Academy of Sciences of the United States of America 100:11484-11489.

Kim, J, S Sinha. 2007. Indelign: a probabilistic framework for annotation of insertions and deletions in a multiple alignment. Bioinformatics 23:289-297.

Kim, J, S Sinha. 2010. Towards realistic benchmarks for multiple alignments of non-coding sequences. BMC Bioinformatics 11:doi:10.1186/1471-2105-1111-1154.

Kimura, M. 1968. Evolutionary Rate at the Molecular Level. Nature 217:624-626.

Kimura, M. 1980. A simple model for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. Journal of Molecular Evolution 16:111-120.

Kimura, M. 1981. Estimation of evolutionary distances between homologous nucleotide sequences. Proceedings of the National Academy of Sciences of the United States of America 78:454-458.

Kimura, M. 1983. The Neutral Theory of Molecular Evolution: Cambridge University Press.

King, JL, TH Jukes. 1969. Non-Darwinian Evolution. Science 164:788-798.

Kishino, H, M Hasegawa. 1989. Evaluation of the maximum likelihood estimate of the evolutionary tree topologies from DNA sequence data, and the branching order in Hominoidea. Journal of Molecular Evolution 29:170-179.

Kishino, H, JL Thorne, WJ Bruno. 2001. Performance of a divergence time estimation method under a probabilistic model of rate evolution. Molecular Biology and Evolution 18:352-361.

Kitchovitch, S, Y Song, R van der Wath, P Liò. 2009. Substitution Matrices and Mututal Information Approaches to Modeling Evolution. Learning and Intelligent Optimization. Berlin: Springer. p. 259-272.

Knudsen, B, M Miyamoto. 2003. Sequence alignments and pair hidden Markov models using evolutionary history. Journal of Molecular Biology 333:453-460.

Korn, L, C Queen, M Wegman. 1977. Computer analysis of nucleic acid regulatory sequences. Proceedings of the National Academy of Sciences of the United States of America 74:4401-4405.

Kosiol, C, N Goldman. 2005. Different versions of the Dayhoff rate matrix. Molecular Biology and Evolution 22:193-199.

Kosiol, C, I Holmes, N Goldman. 2007. An empirical codon model for protein sequence evolution. Molecular Biology and Evolution 24:1464-1479.

Kreuzer, KN. 2005. Interplay between DNA replication and recombination in prokaryotes. Annual Review of Microbiology 59:43-67.

Krogh, A, M Brown, I Mian, K Sjolander, D Haussler. 1994. Hidden markov models in computational biology: Applications to protein modeling. Journal of Molecular Biology 235.

Kulikova, T, P Aldebert, N Althorpe, et al. 2004. The EMBL Nucleotide Sequence Database. Nucleic Acids Research 32:D27-D30.

Kyte, J, R Doolittle. 1982. A simple method for displaying the hydropathic character of a protein. Journal of Molecular Biology 157:105-132.

Lake, JA. 1991. The Order of Sequence Alignment Can Bias the Selection of Tree Topology. Molecular Biology and Evolution 8:378-385.

Landan, G, D Graur. 2007. Heads or tails: A simple reliability check for multiple sequence alignments. Molecular Biology and Evolution 24:1380-1383.

Larget, B, DL Simon. 1999. Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. Molecular Biology and Evolution 16:750-759.

Larkin, M, G Blackshields, N Brown, et al. 2007. Clustal W and Clustal X Version 2.0. Bioinformatics 23:2947-2948.

Lavalette, D. 1996. Facteur d'impact: impartialité ou impuissance?: Institut Curie - Recherche, Bât. 112, Centre Universitaire, 91405 Orsay, France.

Le, SQ, O Gascuel. 2008. LG: An Improved, General Amino-Acid Replacement Matrix. Molecular Biology and Evolution 25:1307–1320.

Lemmon, AR, EC Moriarty. 2004. The importance of proper model assumption in Bayesian phylogenetics. Systematic Biology 53:265–277.

Lewis, PO. 1998. A Genetic Algorithm for Maximum-Likelihood Phylogeny Inference Using Nucleotide Sequence Data. Molecular Biology and Evolution 15:277-283.

Lindsay, H, VB Yap, H Ying, GA Huttley. 2008. Pitfalls of the most commonly used models of context dependent substitution. Biology Direct 3:52.

Ling, CX, J Huang, H Zhang. 2003. AUC: A Better Measure than Accuracy in Comparing Learning Algorithms. Advances in Artificial Intelligence: Springer Berlin / Heidelberg. p. 329-341.

Lipman, D, R Pearson. 1985. Rapid and sensitive protein similarity searches. Science 227:1435-1441.

Lipman, DJ, SF Altschul, JD Kececioglu. 1989. A tool for multiple sequence alignment. Proceedings of the National Academy of Sciences of the United States of America 86:4412-4415.

Liu, K, S Nelesen, S Raghavan, CR Linder, T Warnow. 2009a. Barking Up The Wrong Treelength: The Impact of Gap Penalty on Alignment and Tree Accuracy. IEEE Transactions on Computational Biology and Bioinformatics 6:7-21.

Liu, K, S Raghavan, S Nelesen, CR Linder, T Warnow. 2009b. Rapid and Accurate Large-Scale Coestimation of Sequence Alignments and Phylogenetic Trees. Science 19:1561-1564.

Lobo, JM, A Jiménez-Valverde, R Real. 2008. AUC: a misleading measure of the performance of predictive distribution models. Global Ecology and Biogeography 17:145-151.

Löytynoja, A, N Goldman. 2005. An algorithm for progressive multiple alignment of sequences with insertions. Proceedings of the National Academy of Sciences of the United States of America 102:10557-10562.

Löytynoja, A, N Goldman. 2008. Phylogeny-aware gap placement prevents errors in sequence alignment and evolutionary analysis. Science 320:1632-1635.

Lunter, G, I Miklós, A Drummond, JL Jensen, J Hein. 2005. Bayesian Coestimation of Phylogeny and Sequence Alignment. BMC Bioinformatics 6:83.

Maddison, DR. 1991. The Discovery and Importance of Multiple Islands of Most-Parsimonious Trees. Systematic Zoology 40:315-328.

Mallick, S, S Gnerre, P Muller, D Reich. 2010. The difficulty of avoiding false positives in genome scans for natural selection. Genome Research 19:922-933.

Mar, JC, TJ Harlow, MA Ragan. 2005. Bayesian and maximum likelihood phylogenetic analyses of protein sequence data under relative branch-length differences and model violation. BMC Evolutionary Biology 5:8.

Mayrose, I, N Friedman, T Pupko. 2005. A Gamma mixture model better accounts for among site rate heterogeneity. Bioinformatics 21:ii151-ii158.

McGlynn, P. 2004. Links between DNA replication and recombination in prokaryotes. Current Opinion in Genetics & Development 14:107-112.

Mendel, G. 1866. Experiments on Plant Hybridization. Proceedings of the Natural History Society of Brünn 4:3-47.

Messier, W, C-B Stewart. 1997. Episodic adaptive evolution of primate lysozymes. Nature 385:151-154.

Metropolis, N, AW Rosenbluth, MN Rosenbluth, AH Teller, E Teller. 1953. Equations of State Calculations by Fast Computing Machines. Journal of Chemical Physics 21:1087-1092.

Metzler, D. 2003. Statistical alignment based on fragment insertion and deletion models. Bioinformatics 19:490-499.

Miklós, I, G Lunter, I Holmes. 2004. A "Long Indel" Model For Evolutionary Sequence Alignment. Molecular Biology and Evolution 21:529-540.

Mills, RE, CT Luttig, CE Larkins, A Beauchamp, C Tsui, WS Pittard, SE Devine. 2006. An initial map of insertion and deletion (INDEL) variation in the human genome. Genome Research 16:1182-1190.

Minovitsky, S, P Stegmaier, A Kel, AS Kondrashov, I Dubchak. 2007. Short sequence motifs, overrepresented in mammalian conserved non-coding sequences. BMC Genomics 8:378.

Misawa, K, RF Kikuno. 2009. Evaluation of the effect of CpG hypermutability on human codon substitution. Gene 431:18-22.

Mitchison, G. 1999. A probabilistic treatment of phylogeny and sequence alignment. Journal of Molecular Evolution 49:11-22.

Morrison, DA, JT Ellis. 1997. Effects of nucleotide sequence alignment on phylogeny reconstruction: a case study of 18S rDNAs of apicomplexa. Molecular Biology and Evolution 14:428-441.

Morton, BR, VM Oberholzer, MT Clegg. 1997. The influence of specific neighboring bases on substitution bias in noncoding regions of the plant chloroplast genome. Journal of Molecular Evolution 45:227-231.

Müller, T, M Vingron. 2000. Modeling amino acid replacement. Journal of Computational Biology 7:761-776.

Murphy, WJ, E Eizirik, SJ O'Brien, et al. 2001. Resolution of the early placental mammal radiation using Bayesian phylogenetics. Science 294:2348-2351.

Myers, E, W Miller. 1988. Optimal alignments in linear space. Computer Applications in the BioSciences 4:11-17.

Needleman, S, C Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48:443-453.

Neyman, J, editor. 1971. Molecular studies of evolution: a source of novel statistical problems. New York: Academic Press.

Nickle, DC, L Heath, MA Jensen, PB Gilbert, JI Mullins, SL Kosakovsky Pond. 2007. HIV-Specific Probabilistic Models of Protein Evolution. PloS One 2:e503.

Nielsen, R, Z Yang. 1998. Likelihood models for detecting positively selected amino acid sites and applications to the HIV-1 envelope gene. Genetics 148:929-936.

Notredame, C. 2002. Recent progresses in multiple sequence alignment: a survey. Pharmacogenomics 31:131-144.

Notredame, C, D Higgins, J Heringa. 2000. T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment. Journal of Molecular Biology 302:205-217.

Nuin, PAS, Z Wang, ERM Tillier. 2006. The accuracy of several multiple sequence alignment programs for proteins. BMC Bioinformatics 24:471.

Ogden, T, MS Rosenberg. 2006. Multiple sequence alignment accuracy and phylogenetic inference. Systematic Biology 55:314-328.

Ogden, T, MF Whiting. 2003. The problem with ‘‘the Paleoptera Problem:’’ sense and sensitivity. Cladistics 19:432-442.

Ogurtsov, AY, S Sunyaev, AS Kondrashov. 2004. Indel-based evolutionary distance and mouse-human divergence. Genome Research 14:1610-1616.

Olsen, G. 1987. Earliest phylogenetic branchings: comparing rRNA-based evolutionary trees inferred with various techniques. Cold Spring Harbor Symposia on Quantitative Biology 52:825-837.

Osawa, S, TH Jukes. 1989. Codon reassignment (codon capture) in evolution. Journal of Molecular Evolution 28:271-278.

Pang, A, AD Smith, PAS Nuin, ERM Tillier. 2005a. SIMPROT: using an empirically determined indel distribution in simulations of protein evolution. BMC Bioinformatics 27:236.

Pang, H, J Tang, S-S Chen, S Tao. 2005b. Statistical distributions of optimal global alignment scores of random protein sequences. BMC Bioinformatics 6:257.

Pearson, R. 1988. Improved tools for biological sequence comparison. Proceedings of the National Academy of Sciences of the United States of America 85:2444-2448.

Pedersen, A-MK, C Wiuf, FB Christiansen. 1998. A codon-based model designed to describe lentiviral evolution. Molecular Biology and Evolution 15.

Penn, O, E Privman, H Ashkenazy, G Landan, D Graur, T Pupko. 2010a. GUIDANCE: a web server for assessing alignment confidence scores. Nucleic Acids Research.

Penn, O, E Privman, G Landan, D Graur, T Pupko. 2010b. An alignment confidence score capturing robustness to guide-tree uncertainty. Molecular Biology and Evolution.

Poirot, O, K Suhre, C Abergel, E O'Toole, C Notredame. 2004. 3DCoffee@igs: a web server for combining sequences and structures into a multiple sequence alignment. Nucleic Acids Research 32:W37-40.

Pollard, DA, CM Bergman, J Stoye, SE Celniker, MB Eisen. 2004. Benchmarking tools for the alignment of functional noncoding DNA. BMC Bioinformatics 5:6.

Popescu, I-I. 2003. On a Zipf's law extension to impact factors. Glottometrics 6:83-93.

Popescu, I-I, M Ganciu, MC Penache, D Penache. 1997. On the Lavalette Ranking Law. Romanian Reports in Physics 49:3-27.

Posada, D, T Buckley. 2004. Model selection and model averaging in phylogenetics: advantages of Akaike information criterion and Bayesian approaches over likelihood ratio tests. Systematic Biology 53:793-808.

Posada, D, KA Crandall. 1998. MODELTEST: testing the model of DNA substitution. Bioinformatics 14:817-818.

Posada, D, KA Crandall. 2001. Selecting the best-fit model of nucleotide substitution. Systematic Biology 50:580-601.

Posada, D, KA Crandall. 2002. The Effect of Recombination on the Accuracy of Phylogeny Estimation. Journal of Molecular Evolution 54:396-402.

Qian, B, RA Goldstein. 2001. Distribution of indel lengths. Proteins: Structure, Function, and Genetics 45:102-104.

Rambaut, A, N Grassly. 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.

Rannala, B, Z Yang. 1996. Probability distribution of molecular evolutionary trees: a new method of phylogenetic inference. Journal of Molecular Evolution 42:587-596.

Rannala, B, Z Yang. 2003. Bayes Estimation of Species Divergence Times and Ancestral Population Sizes Using DNA Sequences From Multiple Loci. Genetics 164:1645-1656.

Redelings, BD, MA Suchard. 2005. Joint Bayesian estimation of alignment and phylogeny. Systematic Biology 54:401-418.

Ren, F, H Tanaka, Z Yang. 2005. An Empirical Examination of the Utility of Codon-Substitution Models in Phylogeny Reconstruction. Systematic Biology 54:808-818.

Ringrose, L, V Lounnas, L Ehrlich, F Buchholz, R Wade, AF Stewart. 1998. Comparative kinetic analysis of FLP and cre recombinases: mathematical models for DNA binding and recombination. Journal of Molecular Biology 284:363-384.

Robinson, DM, DT Jones, H Kishino, N Goldman, JL Thorne. 2003. Protein Evolution with Dependence Among Codons Due to Tertiary Structure. Molecular Biology and Evolution 20:1692-1704.

Ronquist, F, JP Huelsenbeck. 2003. MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19:1572-1574.

Rosenberg, MS. 2005a. Multiple sequence alignment accuracy and evolutionary distance estimation. BMC Bioinformatics 6:278.

Rosenberg, MS. 2005b. MySSP: Non-stationary evolutionary sequence simulation, including indels. Evolutionary bioinformatics online 1.

Rosenberg, MS, S Kumar. 2003. Heterogeneity of nucleotide frequencies among evolutionary lineages and phylogenetic inference. Molecular Biology and Evolution 20:610-621.

Roshan, U, DR Livesay, S Chikkagoudar. 2006. Improving progressive alignment for phylogeny reconstruction using parsimonious guide-trees. Sixth IEEE Symposium on BionInformatics and BioEngineering. Washington DC, USA.

Rota-Stabelli, O, Z Yang, MJ Telford. 2009. MtZoa: A general mitochondrial amino acid substitutions model for animal evolutionary studies. Molecular Phylogenetics and Evolution 52:268-272.

Saitou, N, M Nei. 1987. The Neighbour-joining Method: A New Method for Reconstructing Phylogenetic Trees. Molecular Biology and Evolution 4:406-425.

Schneider, A, A Souvorov, N Sabath, G Landan, GH Gonnet, D Graur. 2009. Estimates of positive Darwinian selection are inflated by errors in sequencing, annotation and alignment. Genome. Biol. Evol. 1:114-118.

Schoniger, M, A von Haeseler. 1994. A stochastic model and the evolution of autocorrelated DNA sequences. Molecular Phylogenetics and Evolution 3:240-247.

Schwartz, S, Z Zhang, KA Frazer, A Smit, C Riemer, J Bouck, R Gibbs, RC Hardison, W Miller. 2000. PipMaker—a Web server for aligning two genomic DNA sequences. Genome Research 10:577-586.

Schwarz, G. 1978. Estimating the Dimension of a Model. Annals of Statistics 6:461-464.

Schwarz, R, W Fletcher, F Förster, B Merget, M Wolf, J Schultz, F Markowetz. 2010. Evolutionary distances in the twilight zone - a rational kernel approach. PloS One 5:e15788.

Self, SG, K-Y Liang. 1987. Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under non-standard conditions. J. Am. Stat. Assoc. 82:605-610.

Seo, T-K, H Kishino. 2009. Statistical Comparison of Nucleotide, Amino Acid, and Codon Substitution Models for Evolutionary Analysis of Protein-Coding Sequences. Systematic Biology 58:199-210.

Shapiro, B, A Rambaut, A Drummond. 2005. Choosing Appropriate Substitution Models for the Phylogenetic Analysis of Protein-Coding Sequences. Molecular Biology and Evolution 23:7-9.

Shimodaira, H, M Hasegawa. 1999. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. Molecular Biology and Evolution 16:1114-1116.

Siepel, A, D Haussler. 2004. Phylogenetic estimation of context-dependent substitution rates by maximum likelihood. Molecular Biology and Evolution 21:468-488.

Silva, JC, AS Kondrashov. 2002. Patterns in spontaneous mutation revealed by human–baboon sequence comparison. Trends in Genetics 18:544–547.

Slowinski, J. 1998. The Number of Multiple Alignments. Molecular Phylogenetics and Evolution 10:264-266.

Smith, T, M Waterman. 1981. Identification of Common Molecular Subsequences. Journal of Molecular Biology 147:195-197.

Sousa, A, L Zé-Zé, P Silva, R Tenreiro. 2008. Exploring tree-building methods and distinct molecular data to recover a known asymmetric phage phylogeny. Molecular Phylogenetics and Evolution 48:563-573.

Stamatakis, A. 2005. An Efficient Program for phylogenetic Inference Using Simulated Annealing. 19th International Parallel and Distributed Processing Symposium (IPDPS 2005). Denver, Colorado.

Stamatakis, A. 2006. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics 22:2688-2690.

Stoye, J. 1998. Multiple sequence alignment with the divide-and-conquer method. Gene 211:GC45-GC56.

Stoye, J, D Evers, F Meyer. 1998. Rose: Generating sequence families. Bioinformatics 14:157-163.

Stoye, J, V Moulton, A Dress. 1997. DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. Computational Applications in the Biosciences 13:625-626.

Strope, CL, K Abel, SD Scott, EN Moriyama. 2009. Biological sequence simulation for testing complex evolutionary hypotheses: indel-Seq-Gen version 2.0. Molecular Biology and Evolution 26:2581-2593.

Strope, CL, SD Scott, EN Moriyama. 2007. indel-Seq-Gen: A new protein family simulator incorporating domains, motifs, and indels. Molecular Biology and Evolution 24:640-649.

Studer, RA, S Penel, L Duret, M Robinson-Rechavi. 2008. Pervasive positive selection on duplicated and nonduplicated vertebrate protein coding genes. Genome Research 18:1393-1402.

Styczynski, MP, KL Jensen, I Rigoutsos, G Stephanopoulos. 2008. BLOSUM62 miscalculations improve search performance. Nature Biotechnology 26:274-275.

Subak-Sharpe, J. 1967. Base doublet frequency patterns in the nucleic acid and evolution of viruses. British Medical Bulletin 23:161.

Subramanian, AR, M Kaufmann, B Morgenstern. 2008. DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment. Algorithms for Molecular Biology 3:6.

Suchard, MA, BD Redelings. 2006. BAli-Phy: simultaneous Bayesian inference of alignment and phylogeny. Bioinformatics 22:2047-2048.

Sullivan, J, D Swofford, G Naylor. 1999. The Effect of Taxon Sampling on Estimating Rate Heterogeneity Parameters of Maximum-Likelihood Models. Molecular Biology and Evolution 16:1347-1356.

Suzuki, Y, GV Glazko, M Nei. 2002. Overcredibility of molecular phylogenies obtained by Bayesian phylogenetics. Proceedings of the National Academy of Sciences of the United States of America 99:16138-16143.

Suzuki, Y, T Gojobori. 1999. A method for detecting positive selection at single amino acid sites. Molecular Biology and Evolution 16:1315-1328.

Swofford, D, G Olsen, PJ Waddell, DM Hillis. 1996. Phylogeny Inference. In: DM Hillis, C Moritz, B Mable, editors. Molecular Systematics. Sunderland (MA): Sinauer Associates. p. 411-501.

Tamura, K. 1992. Estimation of the number of nucleotide substitutions when there are strong transition-transversion and G+C-content biases. Molecular Biology and Evolution 9:678-687.

Tamura, K, M Nei. 1993. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. Molecular Biology and Evolution 10:512-526.

Tavaré, S. 1986. Some probabilistic and statistical problems in the analysis of DNA sequences. Lectures on Mathematics in the Life Sciences 17:57-86.

Taylor, MS, CP Ponting, RR Copley. 2004. Occurrence and consequences of coding sequence insertions and deletions in mammalian genomes. Genome Research 14:555-566.

Thomas, JW, JW Touchman, RW Blakesley, et al. 2003. Comparative analyses of multi-species sequences from targeted genomic regions. 424:788-793.

Thompson, J, D., F Plewniak, O Poch. 1999a. A comprehensive comparison of multiple sequence alignment programs. Nucleic Acids Research 27:2682-2690.

Thompson, J, D Higgins, T Gibson. 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Research 22:4673-4680.

Thompson, J, P Koehl, R Ripp, O Poch. 2005. BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark. Proteins 61:127-136.

Thompson, J, F Plewniak, O Poch. 1999b. BAliBASE: A benchmark alignment database for the evaluation of multiple sequence alignment programs. Bioinformatics 15:87-88.

Thorne, JL, H Kishino. 1992. Freeing Phylogenies from Artifacts of Alignment. Molecular Biology and Evolution 9:1148-1162.

Thorne, JL, H Kishino, J Felsenstein. 1991. An evolutionary model for maximum likelihood alignment of DNA sequences. Journal of Molecular Evolution 33:114–124.

Thorne, JL, H Kishino, J Felsenstein. 1992. Inching toward reality: an improved likelihood model of sequence evolution. Journal of Molecular Evolution 34:3-16.

Thorne, JL, H Kishino, IS Painter. 1998. Estimating the rate of evolution of the rate of molecular evolution. Molecular Biology and Evolution 15:1647-1657.

Tian, D, Q Wang, P Zhang, H Araki, S Yang, M Kreitman, T Nagylaki, R Hudson, J Bergelson, J-Q Chen. 2008. Single-nucleotide mutation rate increases close to insertions/deletions in eukaryotes. 455:105-108.

Torres, A, A Cabada, J Nieto. 2003. An Exact Formula for the Number of Alignments Between Two DNA Sequences. DNA Sequence 14:427-430.

Uzzell, T, K Corbin. 1971. Fitting discrete probability distributions to evolutionary events. Science 172:1089-1096.

Vamathevan, JJ. 2008. Evolutionary Analysis of Mammalian Genomes and Associations to Human Disease. Biology. London: University College London.

Vamathevan, JJ, S Hasan, RD Emes, et al. 2008. The role of positive selection in determining the molecular cause of species differences in disease. BMC Evolutionary Biology 8:273.

Varadarajan, A, RK Bradley, I Holmes. 2008. Tools for simulating evolution of aligned genomic regions with integrated parameter estimation. Genome Biology 9:R147.

von Neumann, J. 1951. Various techniques used in connection with random digits. National Bureau of Standards Applied Mathematics Series 12:36-38.

Waddell, PJ, D Penny, T Moore. 1997. Hadamard conjugations and modeling sequence evolution with unequal rates across sites. Molecular Phylogenetics and Evolution 8:33-50.

Wakeley, J. 1993. Substitution rate variation among sites in hypervariable region 1 of human mitochondrial DNA. Journal of Molecular Evolution 37:613-623.

Wallace, I, O O'Sullivan, D Higgins. 2004. Evaluation of iterative alignment algorithms for multiple alignment. Bioinformatics 21:1408-1414.

Wallace, I, O O'Sullivan, D Higgins, C Notredame. 2006. M-Coffee: combining multiple sequence alignment methods with T-Coffee. Nucleic Acids Research 34:1692-1699.

Walter, S. 2005. The partial area under the summary ROC curve. Statistics in Medicine 24:2025-2040.

Wang, L-S, J Leebens-Mack, PK Wall, K Beckmann, CW dePamphilis, T Warnow. 2009. The Impact of Multiple Protein Sequence Alignment on Phylogenetic Estimation. IEEE Transactions on Computational Biology and Bioinformatics 99.

Wang, L, T Jiang. 1994. On the complexity of multiple sequence alignment. Journal of Computational Biology 1:337-348.

Waterston, R, Lindblad-Toh, K., Birney, E., et al. (219 co-authors). 2002. Initial sequencing and comparative analysis of the mouse genome. Nature 420:520–562.

Watson, JD, F Crick. 1953. A Structure for Deoxyribose Nucleic Acid. Nature 171:737-738.

Wheeler, W, D Gladstein, J De Laet. 2003. POY, Phylogeny Reconstruction via Optimization of DNA and other Data: American Museum of Natural History.

Whelan, S, PIW de Bakker, E Quevillon, N Rodriguez, N Goldman. 2006. PANDIT: an evolution-centric database of protein and associated nucleotide domains with inferred trees. Nucleic Acids Research 34:D327-D331.

Whelan, S, N Goldman. 2001. A general empirical model of protein evolution derived from multiple protein families using a maximum-likelihood approach. Molecular Biology and Evolution 18:691-699.

Whelan, S, N Goldman. 2004. Estimating the Frequency of Events That Cause Multiple-Nucleotide Changes. Genetics 167:2027-2043.

Wilbur, W, D Lipman. 1983. Rapid similarity searches of nucleic acid and protein data banks. Proceedings of the National Academy of Sciences of the United States of America 80:726-730.

Wilm, A, D Higgins, C Notredame. 2008. R-Coffee: a method for multiple alignment of non-coding RNA. Nucleic Acids Research 36:e52.

Wong, KM, MA Suchard, JP Huelsenbeck. 2008. Alignment uncertainty and genomic analysis. Science 319:473-476.

Yamane, K, K Yano, T Kawahara. 2006. Pattern and Rate of Indel Evolution Inferred from Whole Chloroplast Intergenic Regions in Sugarcane, Maize and Rice. DNA Research 13:197-204.

Yang, Z. 1993. Maximum likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. Molecular Biology and Evolution 10:1396-1401.

Yang, Z. 1994a. Estimating the pattern of nucleotide substitution. Journal of Molecular Evolution 39:105-111.

Yang, Z. 1994b. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. Journal of Molecular Evolution 39:306-314.

Yang, Z. 1995a. On the general reversible Markov-process model of nucleotide substitution: a reply to Saccone et al. Journal of Molecular Evolution 41:254-255.

Yang, Z. 1995b. A Space-Time Process Model for the Evolution of DNA Sequences. Genetics 139:993-1005.

Yang, Z. 1996. Among-site rate variation and its impact on phylogenetic analyses. Trends in Ecology & Evolution 11:367-372.

Yang, Z. 1997. PAML: a program package for phylogenetic analysis by maximum likelihood. Computer Applications in the BioSciences 15:555-556.

Yang, Z. 1998. Likelihood ratio tests for detecting positive selection and application to primate lysozyme evolution. Molecular Biology and Evolution 15:568-573.

Yang, Z. 2000. Complexity of the simplest phylogenetic estimation problem. Proceedings of the Royal Society Biological Sciences Series B 267:109-116.

Yang, Z. 2005. Bayesian Inference in Molecular Phylogenetics. In: O Gascuel, editor. Mathematics of Evolution and Phylogeny. Oxford: Oxford University Press. p. 63-90.

Yang, Z. 2006. Computational Molecular Evolution. Oxford, England.: Oxford University Press.

Yang, Z. 2007a. Fair-balance paradox, star-tree paradox and Bayesian phylogenetics. Molecular Biology and Evolution 24:1639-1655.

Yang, Z. 2007b. PAML 4: Phylogenetic Analysis by Maximum Likelihood. Molecular Biology and Evolution 24:1586-1591.

Yang, Z. 2008. Empirical evaluation of a prior for Bayesian phylogenetic inference. Philisophical Transactions of The Royal Society B 363:4031-4039.

Yang, Z, N Goldman, A Friday. 1995. Maximum likelihood trees from DNA sequences: a peculiar statistical estimation problem. Systematic Biology 44:384-399.

Yang, Z, R Nielsen. 1998. Synonymous and nonsynonymous rate variation in nuclear genes of mammals. Journal of Molecular Evolution 46:409-418.

Yang, Z, R Nielsen. 2000. Estimating synonymous and nonsynonymous substitution rates under realistic evolutionary models. Molecular Biology and Evolution 17:32-43.

Yang, Z, R Nielsen. 2002. Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages. Molecular Biology and Evolution 19.

Yang, Z, R Nielsen. 2008. Mutation-Selection Models of Codon Substitution and Their Use to Estimate Selective Strengths on Codon Usage. Molecular Biology and Evolution 25:568-579.

Yang, Z, R Nielsen, N Goldman, A-MK Pedersen. 2000. Codon-substitution models for heterogeneous selection pressure at amino acid sites. Genetics 155:431-439.

Yang, Z, R Nielsen, M Hasegawa. 1998. Models of amino acid substitution and applications to Mitochondrial protein evolution. Molecular Biology and Evolution 15:1600-1611.

Yang, Z, B Rannala. 1997. Bayesian phylogenetic inference using DNA sequences: a Markov Chain Monte Carlo Method. Molecular Biology and Evolution 14:717-724.

Yang, Z, B Rannala. 2005. Branch-length prior influences Bayesian posterior probability of phylogeny. Systematic Biology 54:455-470.

Yang, Z, B Rannala. 2006. Bayesian estimation of species divergence times under a molecular clock using multiple fossil calibrations with soft bounds. Molecular Biology and Evolution 23:212-226.

Yang, Z, D Roberts. 1995. On the use of nucleic acid sequences to infer early branchings in the tree of life. Molecular Biology and Evolution 12:451-458.

Yang, Z, WSW Wong, R Nielsen. 2005. Bayes empirical Bayes inference of amino acid sites under positive selection. Molecular Biology and Evolution 22:1107-1118.

Yap, VB, T Speed. 2005. Rooting a phylogenetic tree with nonreversible substitution models. BMC Evolutionary Biology 5:2.

Zhang, J. 2004. Frequent false detection of positive selection by the likelihood method with branch-site models. Molecular Biology and Evolution 21:1332-1339.

Zhang, J, S Kumar, M Nei. 1997. Small-sample tests of episodic adaptive evolution: a case study of primate lysozymes. Molecular Biology and Evolution 14:1335-1338.

Zhang, J, R Nielsen, Z Yang. 2005. Evaluation of an improved branch-site likelihood method for detecting positive selection at the molecular level. Molecular Biology and Evolution 22:2472-2479.

Zhang, W, GG Bouffard, SS Wallace, JP Bond. 2007. NISC Comparative Sequencing Program. Estimation of DNA sequence context-dependent mutation rates using primate genomic sequences. Journal of Molecular Evolution 65:207-214.

Zhang, Z, M Gerstein. 2003. Patterns of nucleotide substitution, insertion and deletion in the human genome inferred from pseudogenes. Nucleic Acids Research 31.

Zhao, Z, E Boerwinkle. 2002. Neighboring-nucleotide effects on single nucleotide polymorphisms: A study of 2.6 million polymorphisms across the human genome. Genome Research 12.

Appendix A

A Brief Demonstration of How Bayesian Methods Can Be Sensitive to the Prior

In this section I will present a brief example that demonstrates how Bayesian model selection can be sensitive to the choice of prior when the models being compared are all weakly supported given the data. For this purpose I modified MrBayes v3.12 to implement separate exponential priors for internal and external branch lengths in the same manner as Yang and Rannala (2005) had done to MrBayes v3.02 for an earlier study (modified program available at http://abacus.gene.ucl.ac.uk/software.html).

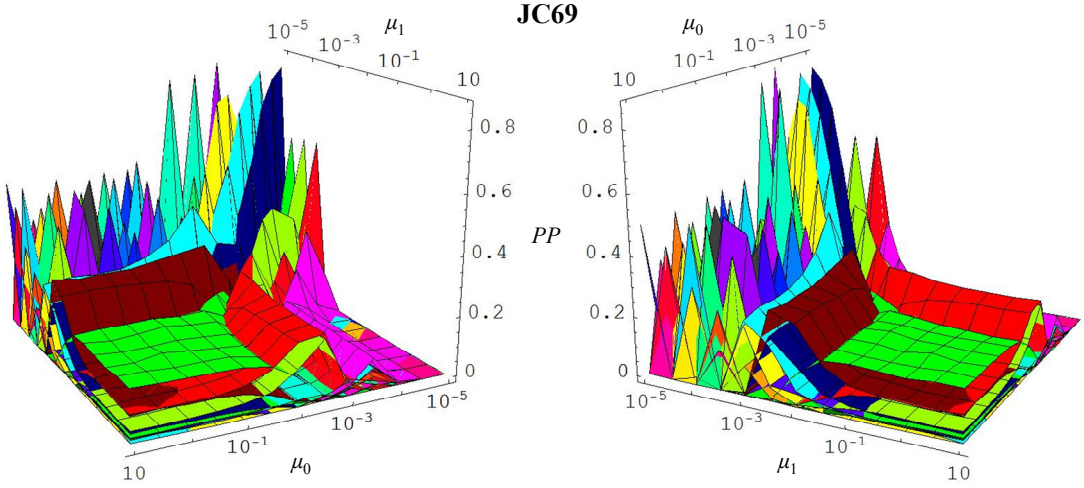
This modified version of MrBayes was used to perform a Bayesian tree search on a relatively uninformative dataset (*bglobin.nex* included in MrBayes, from Yang *et. al.* 2000) using the JC69 and HKY85 models. In addition, the translated amino-acid dataset was analysed under the WAG model. In all three cases, 2 parallel MCMC runs were conducted, each with 4 incrementally heated chains and a different random starting tree. The first 200,000 generations were discarded as burn-in. Tree topologies were sampled from the subsequent 2 million generations, every 100th generation, to contribute to the posterior distribution. The standard deviation between runs was ~ 0.01 by the end of the MCMC runs indicating that convergence had been reached. The posterior distribution resulting from this tree-search is shown in figure 1.8. The resulting MAP tree that is inferred differs depending on the choice of data type (nucleotide or amino-acid), substitution model or combination of prior distribution on branch lengths. In particular, notice that when $\mu_1 \gg \mu_0 \approx 0$ all trees look like the star tree and so the posterior probability of every tree is roughly equal (to zero). Also, when $\mu_1 \approx 0$ external branch lengths in every tree are forced to be ≈ zero – such trees are all unrealistic, causing the chain to rarely leave the random starting tree, resulting in spuriously high posterior probabilities ≈ 0.5.

Poor performance when using extreme exponential branch length priors (with $\mu_1 = \mu_0$) has been observed before (Mar et al. 2005) with extreme values for the branch length priors. However, for less extreme values such as $\mu_1, \mu_0 \geq 10^{-3}$, one can see here that the choice of branch length prior distribution still affects the posterior probability distribution of trees to the point that the method infers different MAP trees. One proposal (that was mentioned in the main text.) that has been shown to reduce such conflicts is to make use of a data size-dependent prior (Yang 2008)

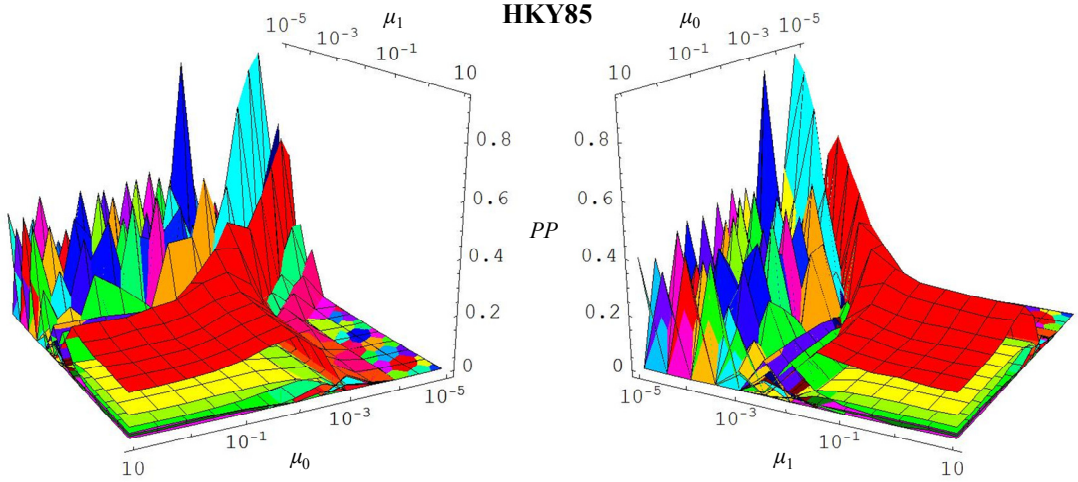
Figure A1: Example of the Effect the Branch-Length Prior can have on the Posterior Tree Distribution obtained during Bayesian Phylogeny Reconstruction

The image is shown on the next page. The three panels (top to bottom) show the results of Bayesian analysis of the nucleotide dataset with the JC69 model, then the HKY85 model, with the bottom panel showing analysis of the translated amino-acid dataset under the WAG model. μ_0 and μ_1 are the means of the exponentially distributed priors on internal and external branch lengths respectively. *PP* is the posterior probability. Each coloured surface represents a distinct tree topology, and the colours are consistent across all of the images (e.g. the red surface in each image represents the same phylogeny). The left and right panel for each model show the same results, but from different angles. The pictures were constructed using custom-written Mathematica code.

JC69



HKY85



WAG