An Approach to Answer: "How Tree-Like is a Network"

An approach to answer: "How Tree-Like is a Network"?

A dissertation submitted to the University of Manchester for the degree of MSc in the Faculty of Life Sciences

Masters Thesis

Geraint James Duck
geraint.duck@postgrad.manchester.ac.uk

Submitted: 2010

Supervisors:
Dr. Simon Whelan
Dr. Cathy Walton

Faculty of Life Sciences
The University of Manchester
Manchester
M13 9PL

Contents

List of Figures

List of Tables

List of Abbreviations

Γ	Gamma – Site Rate Model
2D	Two-Dimensional
3D	Three-Dimensional
App.	Application
AU	Approximately Unbiased Test
BEAST	Bayesian Evolutionary Aanalysis by Sampling Trees – Library
DNA	Deoxyribonucleic Acid
GNU LGPL	GNU Lesser General Public License
GTR	General Time Reversible – Substitution Model
HKY	Hasegawa-Kishono-Yano – Substitution Model
JC	Jukes-Cantor – Substitution Model
JTT	Jones-Taylor-Thornton – Amino-Acid Substitution Model
K2P	Kimura Two Parameter – Substitution Model
LG	Le-Gascuel – Amino-Acid Substitution Model
LS	Least-Squares
MCMC	Markov Chain Monte Carlo

mtDNA	Mitochondrial DNA
NJ	Neighbor-Joining
PAM	Point Accepted Mutation – Amino-Acid Substitution Model
PAML	Phylogenetic Analysis by Maximum Likelihood – Software
PHYLIP	PHYLogeny Inference Package – Software
RAXML	Randomized AXelerated Maximum Likelihood – Software
REV	Synonym for GTR
RMSD	Root Mean Squared Deviation
SH	Shimodaira-Hasegawa Test
SS	Sum of Squares
WAG	Whelan and Goldman – Amino-Acid Substitution Model

Chapter 1

Abstract

This report aims to answer the question of "how networky is my data?". This question can have many repercussions on all types of phylogenetic data analysis helping to highlight biologically significant events. These can include hybridization, reticulation and gene flow. These events are all biologically interesting, but can all cause the underlying analysis assumption of a tree to no longer hold. The result of this project is a tested and verified methodology on which to test for data "networkyness" through a parametric bootstrap approach providing a statistical measure to answer that question. This entirely novel methodology is packaged up within a Java application for easy, everyday use. This application has been thoroughly tested with each component of the methodology validated in turn.

Chapter 2

Preamble

2.1 Declaration

Declaration: I, *Geraint James Duck*, declare that no portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

2.2 Copyright Statement

2.3 Acknowledgements

The author would firstly like to thank Dr. Simon Whelan for his continued help and support through the project. He would also like to thank all the various creators of any software or tools used within this project and importantly the BEAST developers. Thanks also goes out to his fellow friends and colleges of the *Bioinformatics Touchdown Room* for support, advice and friendship. Finally, an acknowledgement goes out to the BBSRC for funding his Master Course and thus, this project and research.

2.4 The Author

Geraint Duck has a First Class degree in *BSc Computer Science* from the *University of Warwick*, awarded in July 2009. His current research experience consists of a 3rd Year Project (as part of the BSc in Computer Science) and this MSc dissertation. His undergraduate project consisted of the generation of a piece of software which aimed to simulate the spread of an infection within an agent based population. He aims to go on to complete a PhD at *Manchester University* starting September 2010 after completion of this project.

Chapter 3

Introduction

3.1 Classical Phylogenetics

3.1.1 Phylogenetic Trees

Phylogenetic trees provide a way to describe which order a set of sequences diverged from each other. For review see chapter 8 of the book by Higgs & Attwood (2005). This report focuses on *bifurcating* trees – trees where there is a two-way split at each branch point. Trees are split into two classes – rooted and unrooted. Both are directed graphs providing a trace from one sequence to another and it is this direction that provides order.

Rooted Trees Rooted trees, when drawn from the bottom up, can give an indication of divergence times (and can be drawn proportionately to represent this,) between two sequences by their vertical branches (though horizontal branches are there to just space out the tree and have no such property). The branch lengths can represent time, evolutionary change or distance, or none of the above if such information is unknown. These trees can also be manipulated to produce equivalent trees that are either rooted in the same place or equivalent trees rooted in a different place (though this will change the divergence ordering).

Unrooted Trees In the alternative case of an unrooted tree, internal nodes represent potential ancestors and the only time related inference possible is to say that such an internal ancestor node must have occurred prior to their external children. Unrooted trees can be rooted with sufficient information about sequence divergence order or the use of an outgroup (a sequence sufficiently diverged from the others to be the root).

3.1.2 Sequence Alignments

Sequence alignment plays a substantial role in evolutionary biology. It forms the basis of a wide variety of analyses including homology modelling, phylogenetic reconstruction and profiling as well as structural and functional

prediction based on sequence conservation (Sierk et al. 2010, Notredame et al. 2000).

Multiple sequence alignment aligns more than two sequences together, often through an iterative approach, using a guide phylogenetic tree to dictate the order in which to align the sequences and to do this often requires the use of heuristics (Penn et al. 2010, Higgins & Sharp 1988). There are already efficient algorithms for solving this problem in polynomial time that guarantee the correct solution using dynamic programming – the most well known and used solution is that of Needleman & Wunsch (1970). The heuristics are required because of the high computational nature of perfect multiple pairwise sequence alignment which adds a great deal of additional complexity to basic pair-wise alignment (from $O(n^2)$ to $O(c^n)$). It is much harder to accurately align more diverged sequences than those more closely related – producing *some* alignment is easy (just putting any two sequences one above the other produces *an* alignment – just not a good or insightful one).

Phylogenetic trees can be inferred directly from their sequence alignments, though alignments are also based on guide trees. The more nucleotide or amino-acid substitutions that there are between two sequences, then the more diverged these two sequences are and the further apart they appear on an appropriate tree. Each base substitution can be corrected to give a better measure of distance through the rate matrices as described later in section 3.1.4.

3.1.3 Biological Insight

A tree provides the primary way to view ancestoral history, whereas a sequence alignment provides an explicit way to see where sequences have diverged at an individual site level as well as (and often more importantly,) where these sites are conserved – this is the basis of site-wise homology. Homology is loosely defined as similarly or relationship and can be used in a wide range of fields. In the case of site-wise homology, it is being used specifically at the phylogenetic sequence level, in which homology then directly implies *common ancestry* (for an in depth discussion on homology see Sattler (1984)).

So, if a specific site shows homology, then the sequences must have, at one point in time, had a common ancestor and this can be displayed on a tree.

Site conservation can give an idea of structural and functional relationships on a per nucleotide or amino-acid basis and gives an easy way to confirm sequence relationships. For example, if a site is completely conserved within a protein family alignment, then that site can be inferred to have a functional impact on the protein whereas if, within the same protein family, a group of sites have the same amino-acid properties (all acidic or all hydrophobic, for example,) then it is likely those residues have a high structural impact on the final protein shape (though there could once again also be functional reasons). Different amino-acids have different effects and occurrence probabilities within protein secondary-structures such as alpha-helices and beta-sheets as well as other protein super-structures.

It is important, however, to distinguish between pure structural homology and evolutionary homology (i.e. site-wise homology). Evolutionary homology is where sequences that appear well aligned have indeed evolved from a common ancestor maintaining some similarity and conservation. The contrast to this is structural homology – where sequences are related by structure, and possibly function, but not necessarily by sequence. In this case, it is possible that two different sequences, never related by a common ancestor, have separately evolved to the same structure because of some advantage it gives. So, it might be a more efficient structure or some sort of "optimum". Sequence alignments can help to highlight the differences between these two types of homology. Though both types can be the same (for example, if two sequence's nucleotides are conserved for their structural traits), this is not always the case (as two separate sequences can converge to the same structure without ever having had a common ancestor).

3.1.4 Substitution Models

Rate Matrices

Substitution models describe changes in character states, so either the rate of change from one nucleotide to another (models of nucleotide substitution)

or the rate of change from one amino-acid to another (models of amino-acid substitution) (Posada & Crandall 2001*b*). These models help pair-wise sequence distance estimations and this resulting distance is then directly proportional to time (Yang 1994). This provides a way to develop straight parsimony scores (sequence site difference) into something more meaningful trying to take into account multiple substitutions at a single site which would not be represented by parsimony. E.g. $A \rightarrow C \rightarrow G$ (which just looks like $A \rightarrow G$ – a single substitution) or $A \rightarrow C \rightarrow A$ (which does not look like a change at all).

Many models have been developed over the years with each improving and relaxing the assumptions of the previous. The most commonly used nucleotide model (and the model used within this application) is the General Time Reversible (GTR or REV) model as first introduced by Tavaré (1986). It is well defined within the paper by Yang (1994) and is the model with the most free parameters.

Each of these free parameters signifies an available variable used to try to better describe the underlying biology within sequence evolution. By enabling simulations to run with more biologically representative models, then the more accurate the results and the better the overall understanding of the underlying processes will be. Because of this, it is the GTR model used within this application because it is best able to describe any sequence alignment supplied to the program, and the more accurate the resulting analysis will be.

The GTR model has a single parameter for each of the base frequencies (3 free, one constrained by the sum of the others being equal to one): $\pi_A, \pi_C, \pi_G, \pi_T$. It then has an additional 6 *exchangability* parameters describing each of the base substitution rates: $R_{AC}, R_{AG}, R_{AT}, R_{CG}, R_{CT}, R_{GT}$ (though the last is not free but instead a scalar for the other 5 giving 8 free parameters in total). Note that because the model is reversible, the rate in one direction is the same as the rate in the reverse direction, so: $R_{AC} = R_{CA}$. The resulting Q matrix is shown below in equation 3.1. See the paper by Yang (1994) for more information.

$$Q = \begin{pmatrix} \cdot & R_{AC}\pi_C & R_{AG}\pi_G & R_{AT}\pi_T \\ R_{AC}\pi_A & \cdot & R_{CG}\pi_G & R_{CT}\pi_T \\ R_{AG}\pi_A & R_{CG}\pi_C & \cdot & R_{GT}\pi_T \\ R_{AT}\pi_A & R_{CT}\pi_C & R_{GT}\pi_G & \cdot \end{pmatrix} \tag{3.1}$$

There are several other nucleotide substitution models, each of which is a constrained version of the GTR model, and each of which was discovered prior to the GTR model (in general, the more constrained the model, the earlier it was found). The first restriction placed on the GTR model, sets several base rate exchangability parameters equal, just leaving allowance for biases between transitions and transversions. The result is the Hasegawa-Kishono-Yano (HKY) model by Hasegawa et al. (1985). The HKY model sets transitions equal to each other and both higher than transversions, so: $R_{AG} = R_{CT} > R_{AC} = R_{AT} = R_{CG} = R_{GT}$. Following on from this, is the Kimura Two Parameter (K2P) model by Kimura (1980) which adds the restriction that all nucleotide frequencies are equal, so $\pi_A = \pi_C = \pi_G = \pi_T = 0.25$. The final restriction removes the bias between transitions and transversions setting all rates to be equal ($R_{XY} = 1.0$), and this is the Jukes-Cantor (JC) model (Jukes & Cantor 1969).

As well as the various nucleotide models already described, there are also various amino-acid models of substitution (though none are used within this project). The first is the Point Accepted Mutation (PAM) matrix by (Dayhoff et al. 1978). This model is based upon an observed estimate of a collection of aligned protein sequences and, because of the relatively high number of amino-acids, has far more free parameters than any of the nucleotide models. This model was later improved through a bigger initial dataset to become the JTT (T.Jones et al. 1992) model. As with nucleotides, further models have also since been developed with various assumptions and accuracies when compared to those before it. Two common ones are the WAG model (Whelan & Goldman 2001) and the LG model (Le & Gascuel 2008).

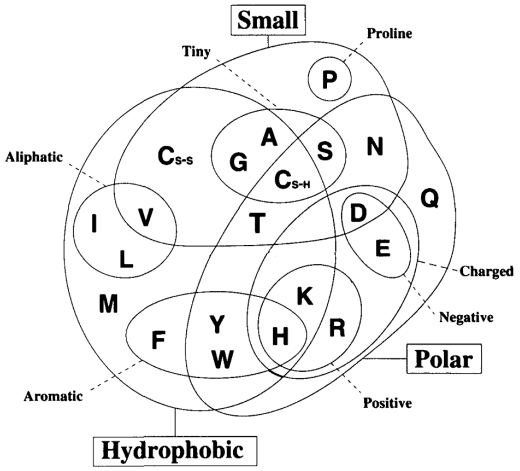


Figure 3.1: This figure shows the various relationships between amino-acid properties through a Venn-Diagram. The figure is adapted from Livingstone & Barton (1993).

Modelling Spatial Heterogeneity in Rate

It is well known that there can be mutational "hot spots" in many genes (Golding 1983) where some sites evolve at a quicker rate than others. This is an inevitable consequence of variation in functional and structural impact as described before. Sites with no impact on function or structure have no constraints on their evolution, whereas a specific site that is essential to either is fixed and is very unlikely to change. This provides a continuous distribution with these extremes to each side. This "give" is implicit in the genetic code with some nucleotide positions having greater impact on the resulting amino-acid than others. For example, changes in the third codon position often have no effect on the resulting amino-acid, whereas changes in the second codon position can have large effects changing amino-acids from one extreme to the other (e.g. based on size or hydrophilic properties) – see table 3.1 for the code and figure 3.1 for the various amino-acid properties. Changes that affect the amino-acid composition are known as non-synonymous changes whereas those that do not are synonymous changes.

This site variability can be modelled by a gamma (Γ) distribution. Though it may not directly represent any biological process, it has been shown to hold

	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Table 3.1: The standard nucleotide to amino-acid genetic code read in the order first codon down the left, second codon along the top and third codon down the right. E.g. UGG is Trp.

on many occasions (for example, see Wakeley's (1993) paper), and can be either a continuous or discrete based distribution (Yang 1996). In the discrete model, each rate comes from one of a given number of rate parameter classes, whereas in the continuous model the number of rates classes is instead infinite and the results rely on the integration of the area under the curve of each class. Each of these classes is internally described by a single value – most often the mean or median value of the range, and this helps avoid issues with the infinite curve in either direction. The gamma distribution involves a single parameter *alpha* (α) to describe the shape and rate variation across sites (Yang 1996). The standard gamma distribution also has an additional parameter *beta* (β) which is a scalar. In the case of the gamma distribution for site variability: $\alpha = \beta$ which implies $\mu = 1$ and $\alpha = \frac{1}{\sigma}$ where μ is the mean and σ is the variance. The given gamma distribution can be used within the various model formulae (e.g. JC or GTR) to correct for variations within each sequence providing a single additional free model parameter overall.

3.1.5 Statistical Inference in Phylogenetics

Maximum-likelihood

The most common statistical inference approach is the maximum-likelihood selection process. This calculates a likelihood value (the probability of something happening by chance) of some data being produced (calculated through simulation) for any given set of parameters. This can be converted to the likelihood of some parameter values given the data through Bayes theorem (Joyce 2008). The maximum-likelihood value, then, is the model with the highest likelihood. This is a consistent method, unlike many before it, which means that, given enough initial data, it will converge to the correct optimal solution.

In this case, it is calculating the likelihood of the model parameters and potential tree given some sequence alignment data (after applying Bayes rule) and finds the evolutionary tree with the highest probability of having generated the data (Shimodaira 2002, Felsenstein 1981), then extending this approach to additionally optimise the model parameters as well as the tree.

For further information on the algorithm itself, see Felsenstein's (1981) paper.

Sequence Simulation

Sequence Simulation is the automated creation of a sequence alignment through a process of artificial evolution. It is implicit within likelihood analysis as well as within Monte Carlo simulation. It is of great importance to molecular evolutionary analysis by aiding understanding of divergence mechanisms (Strope et al. 2007). Simulation enables the user to vary the conditions and assumptions of a model providing the ability to have a known evolution, and thus both a true tree and true alignment for various methods and ideas to be compared to allowing model accuracies to be calculated and quantified (Strope et al. 2009).

Sequence simulation is performed by initially constructing a root sequence and then simulating evolution across the tree using the specified models of evolution (both rate matrices and/or site rate variation – see section 3.1.4). This has the advantage that every ancestor node has a known sequence and ensures that the baseline "true alignment" is of truly homologous sequences (Rosenberg 2005).

Sequence simulation provides a way to calculate likelihoods. The likelihood of some sequence, given some parameters, is exactly how often a given sequence is generated from those parameters (generally very small probabilities and this is why they are often calculated as *log*-likelihoods). However, there are now more efficient ways to calculate likelihoods without needing to do multiple huge sequence simulations.

3.1.6 Hypothesis Testing

P-Values

The application of statistical methods to hypothesis testing is now wide and the most common approach is the *P-Value*. The p-value defines the probability of obtaining a result under the *null hypothesis* that is equal to or more extreme than the observed data (Goodman 1999).

The p-value is often chosen to be at either the 0.05 or 0.01 (5% or 1%) level. This means that there is only a 5% chance that the null hypothesis is true and so the null hypothesis is then rejected in favour of the *alternative hypothesis* (i.e. the given result is in the extreme).
The null hypothesis can now be defined as follows:

$$H_0 = The\ initial\ data\ is\ treelike \tag{3.2}$$

and the alternate hypothesis can be defined as:

$$H_1 = The\ initial\ data\ is\ not\ treelike \tag{3.3}$$

Likelihood Test Ratio

Given that the null and alternative hypotheses (L_0, L_1) are nested, the significance of improvment between the two models can be tested with the *likelihood test ratio* (Whelan et al. 2001). Nested hypotheses are where the null hypothesis is a simplification of the alternative hypothesis (e.g. the GTR model over the JC model of substitution). This is important because the alternative hypothesis will generally come out as better, but not always significantly better. The test ratio (equation 3.4) provides a p-value confidence allowing the null hypothesis to be rejected in favour of the alternative hypothesis.

$$2\delta = 2ln\left(\frac{L_1}{L_0}\right) = 2\left(lnL_1 - lnL_0\right) \tag{3.4}$$

Bootstrapping

An additional approach for hypothesis testing is that of bootstrapping which is required when hypotheses are *non-nested* (Whelan et al. 2001). This approach enables a distribution under the null hypothesis to be generated using Monte Carlo simulation as data can be randomly and repeatedly generated which obeys to the null hypothesis and then a statistic or confidence interval to then be calculated (Goldman 1993*b*).

As described in Ostaszewski & Rempala's (2000) paper, the bootstrap was first introduced by Efron (1979) to try and improve on *jackknifing* as part

of the wider field of resampling. Bootstrapping aims to help calculate both standard error and a distribution statistic on which to base a quantitative level of confidence. There are two different versions of the bootstrap – the *nonparametic* and the *parametric* bootstrap. The difference is that in the nonparamtric case, a discrete distribution with equal probability is used, whereas in the parametric case, the parameters have to first be estimated.

This report and project makes use of the parametric bootstrapping approach which relies on data simulation to provide the null hypothesis model expected result for comparison (Rambaut & Grassly 1997, Goldman 1993*b*). This approach has been widely used already in the past (e.g. Adell & Dopazo 1994, Goldman 1993*a*).

Bootstrapping, for example, is used to provide branch and taxon or taxa topology support values in various phylogenetic trees. It does this through a process of random resampling with replacement of the initial data with reanalysis providing the bootstrap support value. For further reference, see Dopazo (1994) or Felsenstein (1985).

Additional Statistical Tests

Some commonly used statistical tests for trees are the Shimodaira & Hasegawa (1999) (SH) and Shimodaira's (2002) Approximately Unbiased (AU) tests. These tests aim to reduce test bias produced from bootstrapping support values to maximum-likelihood tree selection and aim to provide the final tree topology with a confidence value.

3.2 Alternative Phylogenetics

It is becoming clear that phylogenetic trees alone can not fully describe all the complexities in real datasets. To resolve this, phylogenetic trees are being defined in a new way (but without changing the principals or ideas behind them) to allow the introduction of *phylogenetic networks*. These expand the ideas behind phylogenetic trees further to better describe the data available. Networks are able to describe datasets that otherwise do not follow a tree

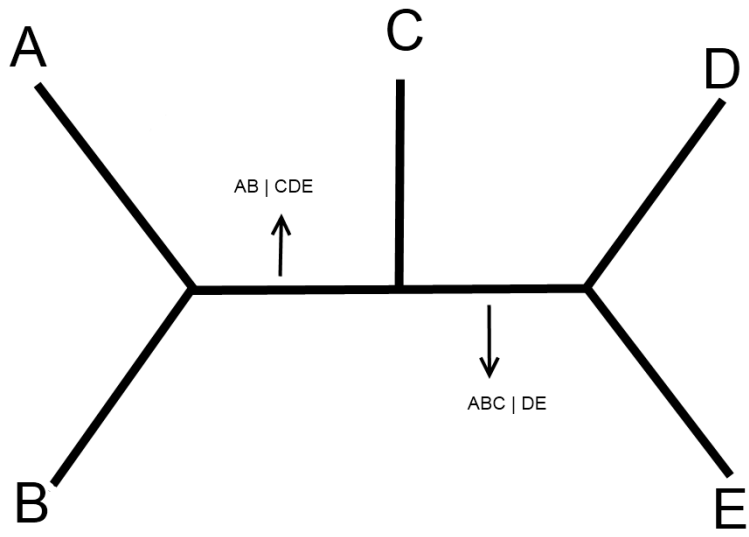


Figure 3.2: 5 taxa tree with two possible non-trivial splits highlighted.

based evolution.

3.2.1 Terminology

Contrasting phylogenetic trees and phylogenetic networks requires some new terminology to be defined. These definitions are the same as those from Huson & Bryant's (2005) paper.

Most importantly, a *split* is defined as

> "A partition of the taxa into two nonempty subsets, such as when we remove a branch from a phylogenetic tree."

So, for example, given the taxa {A,B,C,D}, then all possible splits are: {AB|CD}, {AC|BD} and {AD|BC}, alongside the four trivial splits of just one taxon each. This can be expanded for any number of taxa and two example splits have been labelled on figure 3.2.

3.2.2 Phylogenetic Trees: The Return

Now it is worth redefining phylogenetic trees in terms of splits. Every branch on a tree defines a split between groups of taxa (this is clearer on an unrooted

than rooted tree). This has been done before with the alternative and less commonly used "partitions" format. Phylogenetic trees then, are a set of *compatible* splits (or at least, the set of most informative compatible splits). A perfect phylogenetic tree will be fully compatible and have no conflicting information. So, for example, the tree is figure 3.2 is defined as the set of compatible splits: {AB|CDE}, {ABC|DE} and the 5 trivial single taxon splits (e.g. {A|BCDE}). However, there are also incompatible splits not represented in the tree shown (e.g. {AC|BDE} or {BE|ACD}) and this is where phylogenetic networks come in.

3.2.3 Phylogenetic Networks

A *phylogenetic network* is defined by Huson & Bryant (2005) as

> ""Any" network in which taxa are represented by nodes and their evolutionary relationships are represented by edges. (For phylogenetic trees, edges are referred to as branches.)"

Networks are important because there are a number of biological evolutionary processes that cause a given dataset to no longer follow a tree based history. These are all primarily based around reticulation and gene-flow (e.g. hybridization, recombination, gene transfer and duplication-loss and introgression), but can also arise from homoplasy as well as from methodological issues in data collection and analysis (Morrison 2010). If these issues are present within a dataset, then this can void any further tree based phylogenetic analysis and so are important to note prior to any such analysis.

There are two primary types of network – *implicit* and *explicit* networks. The difference between them is that within an explicit network, every node represents either a specific taxon, or an evolutionary event (for example, hybridization or recombination) (Huson & Bryant 2005). This is not the case within an implicit network and thus is not the case with split networks.

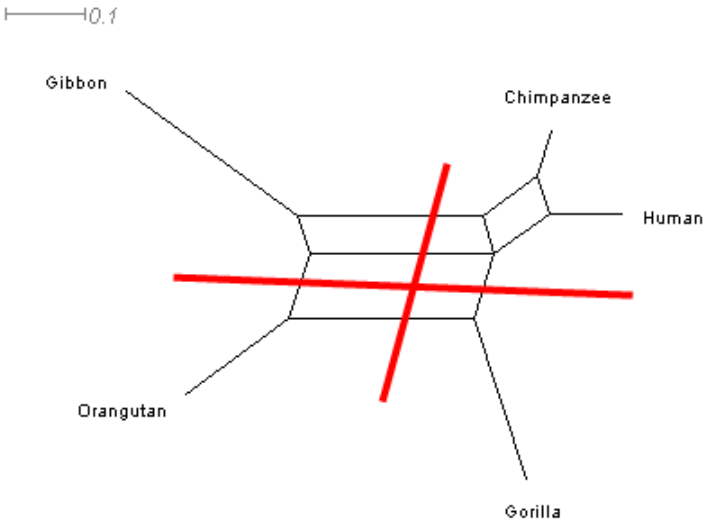


Figure 3.3: Network representation of a dataset generated by simulation down a tree which should be 100% treelike but still has noise causing a visible networkyness aspect (as highlighted in red). Drawn in SplitsTree4 (Huson & Bryant 2005).

Implicit Networks

Split Networks Split networks are a group of phylogenetic networks that use the concept of the "split" (as defined previously). They find such splits in the data and then show these as edges within the drawn network. They extend phylogenetic trees by enabling the user to way to describe all possible splits represented within the data including all *incompatible* splits. If a phylogenetic network only has compatible splits, it is, by definition, a phylogenetic tree – a network with no cycles. The important point here is to have a way to statistically analyse the data and say if the incompatible splits are important or purely down to stochastic noise. This point is highlighted in figure 3.3 where an alignment produced from simulation (and is thus tree based) can still be shown as a network but with noise (the opposing but supported splits have been shown in red) – though this is an extreme example due to the short simulation sequence length used.

Neighbor-Net Neighbor-Net is an algorithm designed by Bryant & Moulton (2004). It aims to provide an efficient method for constructing phylogenetic networks, based on the Neighbor-Joining (NJ) algorithm (Saitou & Nei 1987). Neighbor-Net works in a similar way to that of *Split Decomposition* (Bandelt & Dress 1992), though thought to be more resolved. A set of weighted splits is constructed from a distance matrix and then represented as a graph. It is worth noting that both NJ and Neighbor-Net are consistent meaning that if a purely additive distance matrix is input into the methods, then the correct corresponding tree or set of splits will be returned. The Neighbor-Net algorithm is implemented in the SplitsTree4 software package by Huson & Bryant (2005) provides a way to allow the user to visualise various networks on a 2D screen. As such, it is not a definitive description but more of a sufficient interpretation.

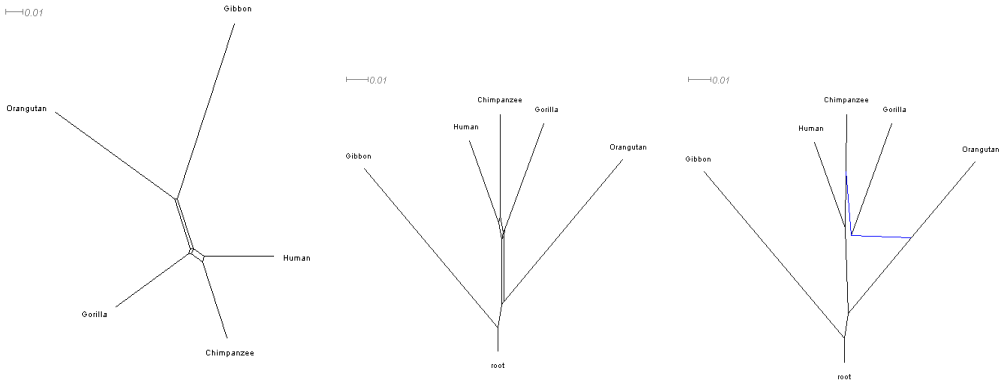
Other Split Networks Other networks based on splits include *Median-Networks* and *Median-Joining Networks* (Posada & Crandall 2001*a*). Median-networks convert sequences into binary data and then groups and weights each split support. A vector is then created for each of these groups producing the final network. Median-joining networks, on the other hand, combine the minimum-spanning trees and then add median vectors to the network on a parsimony basis. This approach requires there to be no recombination and so is limited in its applications.

Explicit Networks

Reticulate Networks Reticulate networks provide the alternate *explicit* representation of the data. Reticulate networks are usually rooted, which is in contrast to most split networks. A *Hybridization Network* is an example of a reticulate network (Huson & Bryant 2005).

Various Networks Example

Figure 3.4 shows the same dataset in three different network representations. The dataset is the Brown dataset by Brown et al. (1982) and is further des-



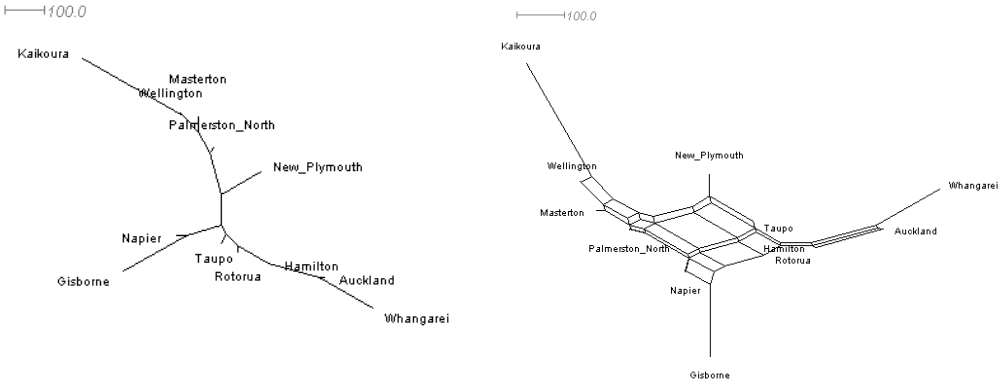
(a) Splits (b) Rooted (c) Hybrid/ Reticulate Net

Figure 3.4: Figure shows the same Brown dataset (see Section 4.2.1) for three different phylogenetic network types. The first is as a normal Splits-Network, the second as the same network, but rooted, and the third as a Hybridization or Reticulation Network.

cribed in section 4.2.1 later in the report. The three representations are the normal Splits-Network, a rooted Splits-Network and a Hybridization or Reticulate Network (they produce the same output in this case). The resulting graphs are produced using Huson & Bryant's (2005) SplitsTree4 program.

3.2.4 Networks verses Trees

As networks provide a way of describing multiple incompatible splits (whereas trees can only show a single set of compatible splits), networks provide a way of viewing the multiple trees consistent with the data all at once within a single structure. This enables the user to question if a single tree really does have greater support than any others. Some of the issues of correct data representation can be resolved through bootstrap analysis of the phylogenetic tree, giving an indication of how much confidence can be placed in a single branch. So, if there is conflicting data, one might expect it to have a lower bootstrap support value, however this may not be the case because bootstrapping provides analysis after tree building whereas exploratory data analysis (such as building a network for review) provides it prior to tree building (Morrison 2010). This distinction has been acknowledged and discussed



(a) Tree (b) Network

Figure 3.5: Figure shows how any dataset can be represented as both a tree (a) and a network (b) using the "North" dataset present in SplitsTree4 (Huson & Bryant 2005).

in the paper by Wägele & Mayer (2007).

The issue here is that if a tree based structure is chosen, as it often is, without regard for the true nature of the dataset, this can render any further analysis redundant (for more information and examples see: Morrison 2010). Many further analysis tools make assumptions about the data that they are analysing and often one of these assumptions is that the supplied data is indeed "treelike" without any speciation or reticulation events. It is important to know if this assumption holds – for example, figure 3.5 shows how any network can be still represented as a tree, though it clearly (by visual inspection)is better represented by a network because of the conflicting data present.

3.3 Related Work

Not much work has been previously done in this area regarding any easy way to compare the "treelikeness" of a given dataset with any statistical output. However, one notable exception is the very recent, and as yet unpublished, work done by Savva et al. (unpublished) which describes much of the processes used within the application resulting from this project (thanks goes to

the authors for allowing us the use of the paper). It describes the appropriate parametric bootstrapping approach used as well as a suggested test-statistic of difference between the tree-based distances and the true distances also suggesting that a least-squares (LS) measure is a good one. However, as a purely theoretical paper, it only analyses the uses of these suggested statistics and the method itself without providing any way of doing this test for yourself. With their permission, this project aims to do just that.

3.4 Aims and Objectives

To summarise, the primary aims of the project are to produce a working piece of software that can produce a statistic defining the "networkyness" of a given dataset (when a guide tree is supplied) testing the evolutionary process of the data. This requires optimisating the pair-wise distance parameters for the true pair-wise distance matrix and optimisation of both model and tree parameters to create the tree-based distance matrix. It also requires alternative sequence alignment generation for the null hypothesis distribution and a distribution calculation and statistic providing a confidence in the treelikeness of the data. Finally, the correctness and accuracy of the software must be tested by comparison to alternative software tools along side additional self testing and validation.

The secondary aims of the project are to use this software to analyse a sample dataset with some preliminary results and to additionally provide a user interface to the software application.

The project is split into 3 main sections, and they are as follows:

3.4.1 Initial Perl Pipeline

The project started out with the aim to analyse a specific dataset for "networkyness" (the two mosquito datasets – see section 4.2.2). That is, to determine if the data would be better represented as a phylogenetic tree or as a phylogenetic network. Though many programs are available to display a dataset in either form (e.g. Schmidt et al. 2002, Huson & Bryant 2005, Felsenstein 2009*b*),

there did not appear to be any statistical measure of the difference implemented in the programs and this lack has been pointed out before by Gauthier & Lapointe (2007). This is contradictory to the literature in which such a measure has been proposed on numerous occasions (e.g. Huson & Bryant 2005, Gauthier & Lapointe 2007, Posada & Crandall 2001*a*).

Due to the apparent lack of availability of a program to calculate a "networkyness" statistic, a Perl pipeline was created which used a variety of software already available with the intention to achieve this goal. This enabled the methodology to be easily tested before any such application developed ensuring that the test statistic works and is valid. Once the idea had its proof of concept, and a test bed on which to be based, then an application incorporating the ideas and individual components into a single piece of software could be developed. This Perl script then acts as a validation tool for the final application ensuring consistent and accurate results.

3.4.2 Primary Java Application

The Java application aims to provide an easy way to ask "How Networky is My Data?" by bringing together all the various components used within the Perl pipeline into a single application for both convenience and ease of use. It provides a way to test the evolutionary process of a dataset through a parametric bootstrapping approach, as described above. This application aims to fill what appears to be a currently unexploited niche within the field of phylogenetics. This application is the primary focus of this report and project, though more emphasis has been placed on methodology rather than software-engineering where possible.

3.4.3 Miscellaneous

This section tries to outline additional work done for this project that did not form part of either the Perl pipeline or Java application. This includes a Java class generated to perform the optimal least-squares calculation based on the paper by Bulmer (1991). However, though this approach worked correctly for optimising the branch lengths, it made it hard to additionally

optimise the GTR+Γ parameters. As such, it became more logical to optimise the GTR+Γ parameters at the same time as the branch lengths within the same class and this initial class was no longer required (though some of the mathematical short-hands are still used elsewhere).

Chapter 4

Materials and Methods

4.1 Methodology and Implementation

So, given the null and alternative hypothesis in equations 4.1 and 4.2 (reproduced below), the project needs to provide a way to compare the two hypothesis and accept or reject them as appropriate.

$$H_0 = The\ initial\ data\ is\ treelike \tag{4.1}$$

$$H_1 = The\ initial\ data\ is\ not\ treelike\ (implies\ networky) \tag{4.2}$$

To do this requires a test of the difference between the networkyness and treelikeness of the data given. To do this, the Least-Squares (LS) statistic has been used which compares the difference between the true network based distance matrix and the tree based distance matrix. The LS distance is defined as:

$$LS_{dist} = \sum_i \sum_{j>i} (d_{i,j} - p_{i,j})^2 \ for\ \forall i, j \in Sequences \tag{4.3}$$

This measure of difference then provides a quantitative way to compare the generated datasets to the initial dataset. If there is sufficient difference between the two measures, then the null hypothesis model can be rejected and the alternative hypothesis accepted with some confidence level.

This section of the report aims to describe each step performed in the analysis of a sequence alignment for networkyness. The underlying aim here is to generate alternative sequence data under the null hypothesis model, and then compare the resulting LS_{dist} values accepting or rejecting the null hypothesis as appropriate. The process used is described in the flow chart in figure 4.1 and is described in more detail in each section below.

4.1.1 Sequence Pair-Wise Distances

Method The first step is to calculate the true pair-wise distance matrix for the given sequence alignment. This is done by calculating the number of differences in nucleotides between each sequence pair and adjusting this

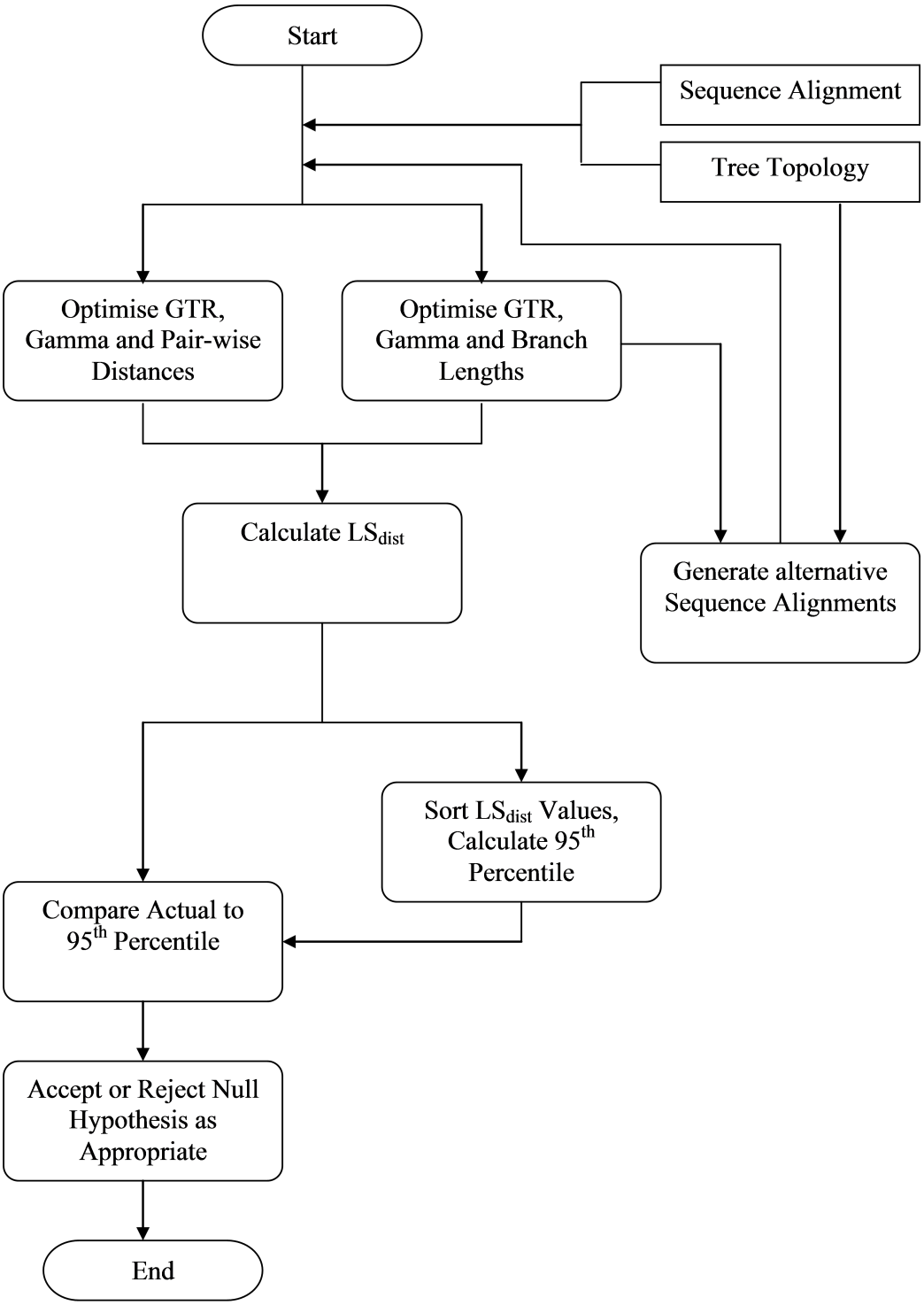


Figure 4.1: Figure shows the various methodology steps performed during the analysis of a sequence alignment.

raw parsimony based score through a model of substitution – in this case the GTR model.

This provides a way to calculate the networkyness of the data – the alternative hypothesis. The pair-wise distances capture the complete and true distance between two sequences and includes all conflicting data and splits that may be present. This is required to calculate the LS distances.

Implementation The supplied data is used for a maximum-likelihood based optimisation which, through a gradient search, aims to optimise both the GTR+Γ model parameters as well as each individual pair-wise distance. Though not an easy problem to solve, the output of this optimiser is each true distance between each sequence pair and is used for the LS calculation. This is done by optimising the single branch distance between two nodes along side each GTR model and gamma distribution parameters.

4.1.2 Tree Based Distance Matrix

Method The tree branch lengths describe how quickly evolution has taken place between the two branch nodes (along the branch). This can be described as a time, as a rate per unit time (for example, substitutions per year) or as a rate per site (for example, mean number of substitutions per site). These tree based distances then provides the basis of the null hypothesis model and are required for the LS distances calculation.

Implementation As well as estimating the GTR model and gamma distribution parameters again, the program will also estimate and optimise the branch lengths of the supplied tree topology based on the same maximum-likelihood approach. This time the program optimises each individual branch length, from which the additive tree based distance matrix is then calculated. This distance is the sum of the branch lengths encountered going from the first sequence node to the second sequence node. For example, in figure 4.2, the distance from taxon A to taxon D is: $0.1 + 0.3 + 0.5 + 0.6 = 2.0$. The output of this alternative optimiser is each individual branch length used for

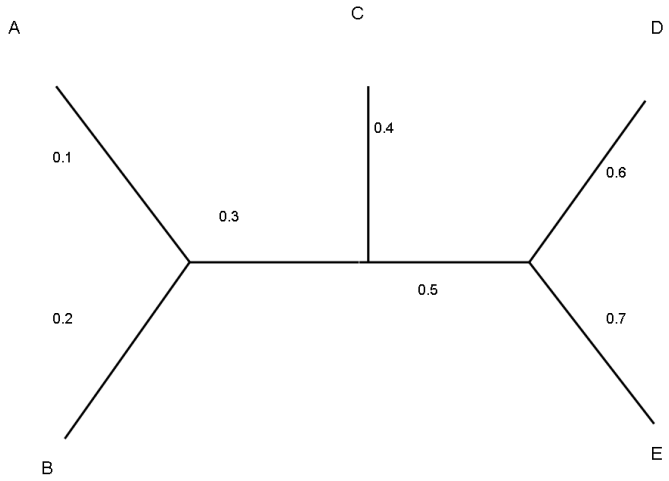


Figure 4.2: Example unrooted phylogenetic tree with branch lengths and 5 taxa.

the LS calculation as well as the GTR+Γ model parameters used for the sequence simulation.

An efficient way of calculating these tree distances given the tree and branch lengths has been implemented within the application (adapted from Bulmer's (1991) paper). As the tree topology never changes, the tree is traversed only once and the branches linking each taxon pair are stored in a 2D matrix of size *taxon pair's* by *total number of branches*. The number of taxon pairs can be defined as:

$$\frac{species(species-1)}{2} \tag{4.4}$$

So for figure 4.2, there are $5 \times (5-1)/2 = 10$ pairs and 7 branches. This matrix contains either a 1 (one), if the branch is part of the route, or a 0 (zero) if it is not.

This matrix is then multiplied by the column vector of tree branch lengths to produce a column vector of the summed distances requiring only simple one/zero multiplication and addition. So, continuing the example from figure

4.2:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0.1 \\ 0.2 \\ 0.4 \\ 0.6 \\ 0.7 \\ 0.3 \\ 0.5 \end{pmatrix} = \begin{pmatrix} 1 \times 0.1 + 1 \times 0.2 + \cdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \cdots + 0 \times 0.3 + 0 \times 0.5 \end{pmatrix} = \begin{pmatrix} 0.3 \\ 0.8 \\ 1.5 \\ 1.6 \\ 0.9 \\ 1.6 \\ 1.4 \\ 1.5 \\ 1.6 \\ 1.3 \end{pmatrix} \quad (4.5)$$

4.1.3 Least Squares Distances

Method This distance measure is what the primary statistic is based on. It is the measure of difference between the true distances (well represented as a network) and the tree based distances (which, if hold, show that the data evolved in a treelike manor). In this case it is chosen to be the LS distance between the two distance matrices, as given in equation 4.3 and again in equation 4.6 below.

Implementation The LS distances are calculated by calculating the LS sum of distances between each possible sequence alignment pairing by comparing the true distance to the tree based distance, as given in equation 4.3, and reproduced below:

$$LS_{dist} = \sum_{i} \sum_{j>i} (d_{i,j} - p_{i,j})^2 \ for \ \forall i, j \in Sequences \quad (4.6)$$

So, the total LS distance is just the squared difference between the true distance (pair-wise distance) and the tree based distance, summed for every sequence pair.

4.1.4 The Bootstrap

Method Having calculated the various model parameters (branch lengths, GTR model and gamma distribution), and the LS distance estimate for the initial dataset, the program then uses the estimated parameters from the tree based optimisation to generate alternative sequence alignments. This process uses the substitution and site rate parameters to calculate where each sequence evolves and which nucleotide to change from and to and then uses the tree topology and branch lengths to work out what total distance the sequences should be apart. The result is a sequence alignment that is generated based upon a 100% treelike process. This process does not allow for any of the potential difficulties found in real datasets that require the need for phylogenetic networks instead. Given these new sequence alignments, the LS distances are then calculated for each new sequence as before. Each of these new LS distances would be the expected LS_{dist} value if the original data satisfied the null hypothesis.

Given the expected numbers produced from the null hypothesis model, the actual number can then be compared to test the hypotheses via this bootstrapping approach.

Implementation This is done through an adapted "Seq-Gen Class" from within the BEAST library (Drummond & Rambaut 2007, Rambaut & Grassly 1997). The original class within the library was designed to be stand alone with many pre-set parameters and so it was adjusted for use within this application instead to allow the parameters to be defined during runtime dependant on previous calculations.

The class also includes parameters for both sequence "Substitution Rate" and for sequence "Damage Rate". This substitution rate is separate from the substitution models described earlier and is a scaling factor for the branch lengths and, as such, is left set at 1.0. It is there to allow the user to transform the branch lengths into the mean number of substitutions per site (Rambaut & Grassly 1997). The damage rate represents the average amount of damage per nucleotide and is used to add such a number of substitutions to the

nucleotide sites to represent substitutions due to DNA damage and as such, it is left set at 0.0 (Ho et al. 2007).

Once each new strictly tree based sequence has been generated, each sequence is then fed back into the two optimisers previously discussed to calculate each sequence's own LS_{dist} value providing the various LS_{dist} values expected from the null hypothesis.

4.1.5 Statistical Confidence – P-Values

Method Given the null hypothesis based LS_{dist} scores, the 95th percentile of this distribution can be calculated. This gives a comparator from which a p-value score can be placed into the "treelikeness" of the data. If the original data is sufficiently treelike, then the evolved data will have roughly equal LS_{dist} scores to the primary data. Whereas, if the primary data is going to be better represented as a network (so a tree is not sufficient), then there will be a discrepancy between the results for the two datasets (primary and evolved) and the true LS_{dist} score will be in the extreme and outside of the 95th percentile.

Implementation This is done quite simply. Given the LS_{dist} values of the alternative sequences, they are sorted and the 95th percentile calculated. By a direct comparison of the real data LS distance to this percentile, the resulting statistic can be output and the analysis completed. If the true data LS distance is less than the simulated data LS distance, then the data is described as treelike. On the other hand, if the true data LS distance is more than the simulated LS distance, then the LS value must be in the extreme and does not conform to the null hypothesis and is thus not treelike.

4.2 Datasets Used

4.2.1 Mitochondrial DNA of Primates

Throughout the various application tests done during development, the Brown et al. (1982) dataset was used as a self-contained small dataset for quick re-

sults and easily comparison between the various programs for validation. It was obtained from the PAML software package (section 4.3.2). It is mitochondrial DNA (mtDNA) from a set of primates (Chimpanzee, Gibbon, Gorilla, Human and Orangutan) and is 896 base pairs in length.

4.2.2 Mitochondrial DNA of Mosquitoes

Two separate alignments of both *Anopheles annularis* and *Anopheles jeyporiensis* are used as an example to show how analysis could be performed with the software. As of the start of the study, it is unknown if this data is better represented as a network or if a tree is sufficient. The data was donated to this project by Dr. Cathy Walton. It is also mtDNA, like the Brown dataset and is 636 base pairs long, containing 123 sequences for *An. annularis* and only 86 sequences for the *An. jeyporiensis* dataset.

Both *An. annularis* and *An. jeyporiensis* are well documented and both are known to have ties to malaria (Dash et al. 2008, Gunasekaran et al. 1989).

4.3 Software Used

The following is a description of all the software and tools used during this project along with appropriate citations and a brief description of how they are used within this project as a whole.

4.3.1 BEAST

BEAST (Bayesian Evolutionary Aanalysis by Sampling Trees) is a Markov Chain Monte Carlo (MCMC) based Bayesian statistical framework for sequence data parameter estimation and hypothesis testing (Drummond & Rambaut 2007). It is designed to allow the user to analyse molecular sequences that are related by an evolutionary tree. Though primarily packaged as its own stand-alone application, the full source library is available to use under the GNU Lesser General Public License (GNU LGPL) license. It is this library that is used and adapted for use within this project.

This library is used to aid the implementation of trees, sequence alignments, substitution models, site models and is also used for its classes defining an optimiser (gradient search) and for tree-likelihood calculations.

4.3.2 Other

The software cited below is all used to form part of the initial pipeline created in Perl for validation of the final Java application or as application validation separate from the pipeline. They do not form part of the final Java application itself.

TREE-PUZZLE TREE-PUZZLE (formally just PUZZLE) is a piece of parallelized software that allows a variety of tree based phylogenetic analysis of sequence data (Schmidt et al. 2002, Strimmer & von Haeseler 1996). It has options to perform maximum-likelihood phylogenetic analysis on both DNA and protein based sequences.

TREE-PUZZLE is used to generate the sequence pairwise distance matrix using the GTR+Γ model of evolution with estimated parameters. This distance matrix is then used as input for Fitch (see below).

RAXML RAXML (Randomized AXelerated Maximum Likelihood) is a parallel and sequential program which is particularly efficient at yielding the best tree phylogenies, with bootstrap support, using maximum-likelihood (Stamatakis 2006, Stamatakis et al. 2010).

The program is used to calculate the maximum-likelihood phylogenetic starting tree given a sequence alignment and some possible tree topologies on which to test. The best tree is then used as input to Fitch (see below) along side the pairwise sequence distance matrix from TREE-PUZZLE. This step is not replicated within the Java application and a starting tree is requested as input instead.

***PHYLIP* - Fitch** PHYLIP (PHYLogeny Inference Package) is a software package developed by Felsenstein (2009*b*) and contains a variety of tools for

phylogenetic inference. Within this package is *Fitch*, which is a program that can carry out LS based tree estimations using the *Fitch-Margoliash* method (Felsenstein 2009*a*).

This project uses Fitch to calculate the LS branch length distances for the final additive tree given a initial starting tree from RAXML and a pairwise distance matrix from TREE-PUZZLE (see above).

***PAML* - baseml & evolver** PAML (Phylogenetic Analysis by Maximum Likelihood) is a collection of software packages designed to do maximum-likelihood based phylogenetic analysis on both DNA and protein sequences (Yang 2007). It is very good at parameter estimation with a large variety of evolutionary models implemented. Of particular note though, are the two internal software programs *baseml* and *evolver*. Baseml uses maximum-likelihood based analysis to estimate various substitution model parameters from a sequence alignment. Evolver then uses set parameters to generate and evolve a sequence alignment - see section 3.1.5.

Both baseml and evolver are used for the reasons described for this project within the Perl pipeline as well as direct validation tools for the Java application.

Seq-Gen Seq-Gen (or Sequence-Generator) is an alternative program to PAML's evolver performing a very similar function (Rambaut & Grassly 1997). Like evolver, Seq-Gen will generate a sequence alignment given a substitution model and its parameters as well as an optional tree to perform the generation along. In addition, it is a version of this Seq-Gen program that is implemented in a Java class file within the BEAST Java Library (see section 4.3.1).

This is used as a validation tool for the Java application only and not within the Perl pipeline. The BEAST Library Seq-Gen class file *is* used within the Java application.

BioPerl BioPerl is an Open-Source project which has helped create a large Perl based library of various models for use primarily within the Bioinforma-

tics field (Stajich et al. 2002).

The BioPerl model for tree-based analysis is used within the Perl pipeline to help calculate the bootstrapped distribution.

Chapter 5

Results

5.1 Methodology Validation

The application takes in both a set of aligned sequences in Phylip format (Felsenstein 2009*b*) and a tree described in the Newick format (Olsen 1990). This tree does not require branch lengths, but appropriate estimate branch lengths may help to speed up the computation.

The final output will be a significance score describing how much confidence can be placed in a phylogenetic tree of that data. If this score is low, then the data is better off described by a network instead.

Note that during some testing sections, additional program components may also be inadvertently tested due to the complex interlinking of these components within the program. However, where ever possible, each component (as shown in figure 4.1) will be tested individually.

The overall methodology and program will be split into three main sections; Initial Set-up, Optimisation and Statistical aspects. All values and results are given to 5 decimal places.

5.1.1 Initial Set-up

Base State Frequencies The first test is to just ensure that the program calculates average base frequencies correctly from the data, as this forms the basis for all models that do not assume equal base probabilities (e.g. HKY and GTR). The sequence alignment library within BEAST can do this calculation directly from the alignments. As a comparison, the frequencies will also be calculated using baseml from the PAML package (Yang 2007). The overall frequencies are calculated by averaging the probabilities of each individual base within each sequence. The results are shown in table 5.1.

These results suggest that the calculation is correct as they are equivalent results to 5 decimal places after accounting for rounding.

Likelihood calculation This can be tested by fixing both the GTR model parameters to the JC model by setting all the GTR parameters to 1.0, fixing the branch lengths and by not using the gamma distribution. This also requires setting all the base frequencies to 0.25 rather than estimating them

	Java App.	baseml (PAML)
π_A	0.31196	0.31196
π_C	0.32894	0.32894
π_G	0.10682	0.10682
π_T	0.25229	0.25229

Table 5.1: Tables describes the results of the base frequency calculations from both the Java application and PAML's baseml program (Yang 2007).

Java App.	baseml (PAML)
-4047.38265	-4047.38265

Table 5.2: Testing the likelihood calculation by setting the model to JC, no gamma and fixing the branch lengths using the Brown dataset and the baseml PAML program (Yang 2007) for comparison.

from the dataset. This tries to ensure that it is just the likelihood calculation being compared as there are no longer any free model parameters needing optimisation. The two likelihood results from the Java application and PAML's baseml package (Yang 2007) are shown in table 5.2 and are identical to 6 decimal places confirming the validity of the likelihood calculation.

5.1.2 Optimisation

The optimiser class within the program uses a non-linear conjugate gradient search using the Beale-Sorenson and Hestenes-Stiefel approach to update direction (for more information on this method see MacKay (2004)). It tests each parameter in turn a little in each direction to decide which direction to go and by about how much moving into a new direction as appropriate (represented as a vector). The bounds used within the program limits each parameter value between 0.0 and 50.0.

GTR+Γ Parameter Estimation

GTR Parameter Validation By setting the branch lengths explicitly (and thus not optimising them) as well as turning off the gamma distribution, the optimiser can be validated on its GTR parameter estimation only. Table

		Java App.	baseml (PAML)
GTR	R_{AC}	0.13866	0.13865
	R_{AG}	1.0	*N/A*
	R_{AT}	0.07866	0.07864
	R_{CG}	0.10335	0.10333
	R_{CT}	7.65749	7.65631
	R_{GT}	0.0	0.00001
Gamma	α	*N/A*	*N/A*
Distances (set at)	b_C	0.1	
	b_{Go}	0.2	
	b_O	0.3	
	b_{Gi}	0.4	
	$b_{H,C}$	0.5	
	$b_{O,Gi}$	0.6	
	b_H	0.7	
Likelihood	(ln)	-3213.79005	-3213.79015

Table 5.3: Table comparing the optimised GTR model parameters for the Brown dataset using both the Java application and the baseml package from within the PAML program (Yang 2007). The branch length are set to fixed values to reduce free model parameters.

5.3 shows how the optimised values are comparable to that of baseml to at least 3 decimal places validating the GTR optimiser.

Γ Parameter Validation Testing the gamma parameter is quite difficult because of the different implementations of the distribution within different programs. The difference here is that one describes each internal rate class using its mean value, the other uses its median value. This will produce similar likelihood and classification values, but not identical. To test this, the GTR model parameters will be kept constant, as will the branch lengths. The comparison will once again be made to the baseml program within the PAML package (Yang 2007). Initially, the alpha value, which describes the gamma distribution, will also be kept constant at 0.5 and the number of rate categories varied to test how close the resulting likelihood values are. These will also be compared to the likelihood value without a gamma distribution to ensure an increase in the value. Results for this initial test are in table

	Likelihoods (ln)	
Rate Categories	Java App.	baseml (PAML)
Off	-4047.38265	-4047.38265
2	-3274.16148	-3274.28840
4	-3229.33540	-3208.97859
6	-3216.02753	-3198.79466
8	-3209.62531	-3195.36731
10	-3205.85124	-3193.80540

Table 5.4: Table giving likelihood results of the baseml program from the PAML package (Yang 2007) and the Java application using the Brown dataset.

5.4 which suggests a good level of relationship between the two alternative programs with a significant level of improvement over the likelihoods without using the gamma distribution.

Next, the alpha value will be optimised, keeping the GTR parameters and branch lengths constant for the Brown dataset along side baseml from PAML (Yang 2007). Although the results will not be identical, they should be at least comparable. Finally, a sequence will be generated with a gamma distribution within the Java application and then analysed using the same application, this way the resulting gamma values should be equal as they will have been calculated through the same method.

However, during this test it was discovered that there was an issue with the gamma alpha parameter optimiser. The optimiser function would repeatedly try to test $\alpha = 0.0$, which is an undefined value resulting in undefined likelihood values. This instantly resulted in the Java application failing to optimise the value. Despite this, it is clear that alpha does have an optimal value but, without additional work, the application is unable to optimise it. The various likelihood values of different values of alpha are plotted in figure 5.1 showing the optimal to be at around 0.13 given the Brown dataset and other fixed parameters.

As a test, based on the surface graph in figure 5.1, the alpha value has been artificially bound at 0.02 to try to avoid the problem of the optimiser trying $\alpha = 0.0$, and the application rerun on the Brown dataset. This successfully

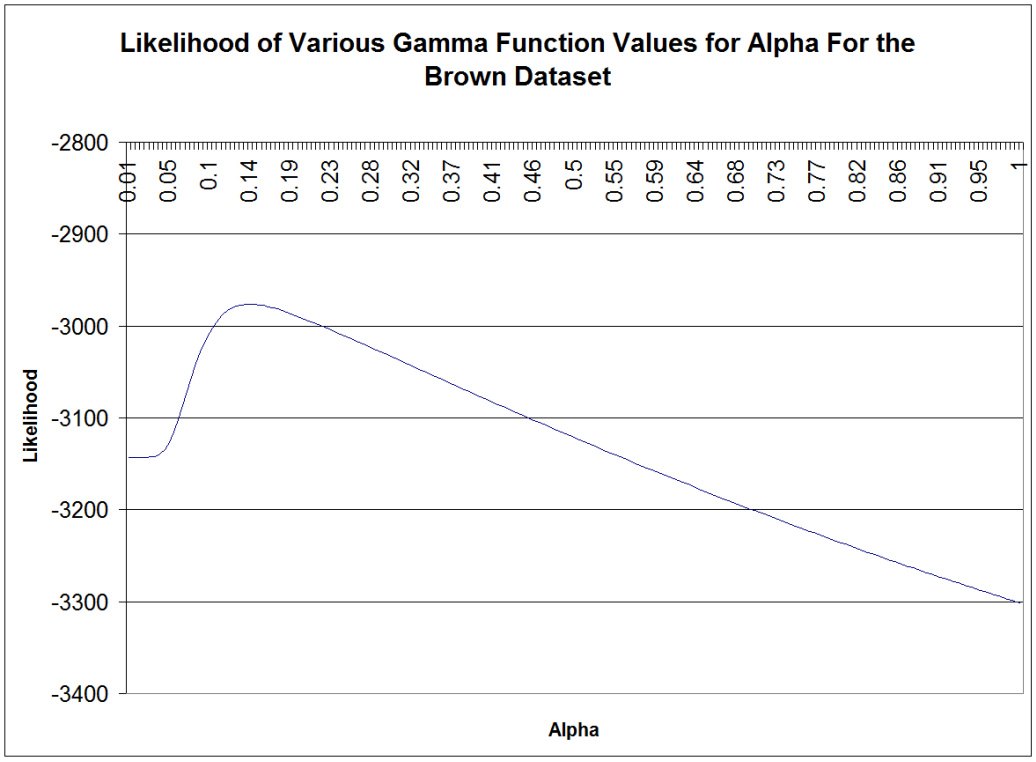


Figure 5.1: Graph showing the calculated likelihood values of the Brown dataset using the JC model but without equal base frequencies and not estimating branch lengths. The only parameter being adjusted is the value of the alpha shape parameter for the gamma distribution. This shows that there is an optimal value of alpha at around 0.13.

	Java App.	baseml (PAML)
α	43.21695	999.00000
γ_1	0.82998	0.96008
γ_2	0.94692	0.98944
γ_3	1.04364	1.00998
γ_4	1.17945	1.04050
likelihood	-5668.68044	-5668.43492

Table 5.5: Table providing results of comparison for the alpha value, the final likelihood and the internal gamma distribution of 4 categories for a simulated dataset with known initial parameters (where $\alpha = 0.8$) after applying the artificial lower bound of 0.02 to the alpha parameter. Program results are being compared to those computed by the baseml program from within the PAML package (Yang 2007).

ran without failing being able to optimise alpha (at 0.20776) along side all other GTR parameters and the branch lengths. This is roughly comparable to the alpha value produced by PAML's baseml program (Yang 2007) of 0.23294 (bearing in mind the different gamma calculation methods) with near equal maximum-likelihood values of -2632.61582 verses -2632.62148.

However, it does not appear that this function is completely working yet. If a sequence is generated *within* the application using set GTR parameter values as well as set branch lengths and a set alpha value for the gamma distribution, the program should correctly optimise these parameters back to their initial values with little error. Though the branch lengths and GTR parameters are indeed optimised back to their initial set values, the alpha value is not. It was initially set to 0.8 for the sequence generation, but instead was optimised back to 43.21695. When the same generation alignment was analysed in baseml from the PAML package (Yang 2007), alpha was optimised to 999 exactly (suggesting that it may have hit the upper bound value). On the other hand, the likelihood values and internal rates (of the various gamma distribution categories) do look substantially more comparable than the final alpha values do (see table 5.5). This clearly requires additional investigation, but, due to time constraints, is now left to future work.

		Java App.	TREE-PUZZLE
GTR	R_{AC}	0.16139	*N/A*
	R_{AG}	1.0	*N/A*
	R_{AT}	0.08895	*N/A*
	R_{CG}	0.07785	*N/A*
	R_{CT}	0.92426	*N/A*
	R_{GT}	0.05731	*N/A*
Gamma	α	*N/A*	*N/A*
Distances	d_{H-C}	0.09489	0.09489
	d_{H-Go}	0.11182	0.11183
	d_{H-O}	0.19078	0.19078
	d_{H-Gi}	0.22265	0.22264
	d_{C-Go}	0.11611	0.11611
	d_{C-O}	0.20596	0.20596
	d_{C-Gi}	0.23420	0.23419
	d_{Go-O}	0.19959	0.19958
	d_{Go-Gi}	0.23448	0.23447
	d_{O-Gi}	0.23644	0.23643
Likelihood	(ln)	-16404.06188	*N/A*

Table 5.6: Table comparing the results for the true pair-wise distances for the Brown dataset between the Java application and TREE-PUZZLE (Schmidt et al. 2002) for the same GTR model parameters.

Sequence Pair-wise Distance Optimisation To test just the pair-wise distance optimisation component, the application was run and the various optimised parameters from the pair-wise distance optimiser function taken for the Brown dataset. The outputs were the six GTR model parameters, the optional gamma parameter (not used in this case to reduce the free variables), the various pair-wise distances and finally the state frequencies as calculated from the data.

The resulting GTR parameters were then fed into TREE-PUZZLE (Schmidt et al. 2002), which estimates pair-wise sequence distances, as the corresponding GTR parameters with gamma off and making no use of a phylogenetic tree. It was also set to calculate exact distances and estimate base frequencies from the data. The results of TREE-PUZZLE were then compared to the primary application and are show in table 5.6.

		Java App.	baseml (PAML)
GTR	R_{AC}	0.15140	0.15137
	R_{AG}	1.0	*N/A*
	R_{AT}	0.08299	0.08299
	R_{CG}	0.07421	0.07420
	R_{CT}	0.96176	0.96155
	R_{GT}	0.03837	0.03835
Gamma	α	*N/A*	*N/A*
Distances	d_H	0.04188	0.04188
	b_C	0.05319	0.05319
	b_{Go}	0.05751	0.05751
	b_O	0.09969	0.09969
	b_{Gi}	0.14051	0.14051
	$b_{H,C}$	0.01684	0.01684
	$b_{O,Gi}$	0.05516	0.05516
Likelihood	(ln)	-2672.03213	-2672.03213

Table 5.7: Table comparing the results for the optimised branch lengths and optimised GTR model parameters for the Brown dataset between the Java application and baseml within the PAML package (Yang 2007).

As you can see, these results are easily comparable to at least 4 decimal places of accuracy suggesting that the application is estimating pair-wise distances correctly.

Branch Lengths Estimation and Optimisation To test the branch length optimiser, the outputs of this function were once again calculated for the Brown dataset without the gamma distribution using the GTR model and base frequencies as estimated from the data. For comparison, the same data was analysed using the baseml program within the PAML package (Yang 2007) providing the results shown in table 5.7.

Once again, these results are easily comparable to four or five decimal places. Please note, however, that the R_{AG} rate is implicitly set to 1.0 within the baseml package as it is not an additional free variable but instead a scalar. In addition, the maximum final log-likelihood values can be compared which suggest an equal optimisation resting place for both the programs.

Pair	Distance	Branch	Length
d_{H-C}	0.09489	b_H	0.04188
d_{H-Go}	0.11182	b_C	0.05319
d_{H-O}	0.19078	b_{Go}	0.05751
d_{H-Gi}	0.22265	b_O	0.09969
d_{C-Go}	0.11611	b_{Gi}	0.14051
d_{C-O}	0.20596	$b_{H,C}$	0.01684
d_{C-Gi}	0.23420	$b_{O,Gi}$	0.05516
d_{Go-O}	0.19959		
d_{Go-Gi}	0.23448		
d_{O-Gi}	0.23644		

Table 5.8: Table shows the optimised pair-wise distances and optimised branch lengths for the Brown dataset. These were directly recalculated by the program but could equally have been taken from previous results tables above. These are then used during both the automatic and manual least-squares calculation from equation 4.3 described previously.

5.1.3 Statistical

Least Squares Distances Calculation There are two main steps to this, both of which just use basic mathematics. The first converts the optimised branch lengths into distances and the second does the LS addition on the two matrices (see equation 4.3).

To test this, the optimised pair-wise distances and branch lengths will be taken from the program, then the LS statistic calculated by hand and compared to the one calculated within the program. The optimised pair-wise distances and branch lengths (for full GTR model) are shown in table 5.8 for the Brown dataset. The correct distances from the optimised data and the program tree based distances were compared as came out as the same, as expected. The LS distances were then also calculated by hand once again providing an equal result of 0.00355.

Alternate Sequence Generation and Evolution By setting the internal sequence generator within the application (a class based on the Seq-Gen program by Rambaut & Grassly (1997) within the BEAST library,) to use specific model parameters, these can then be analysed within both baseml

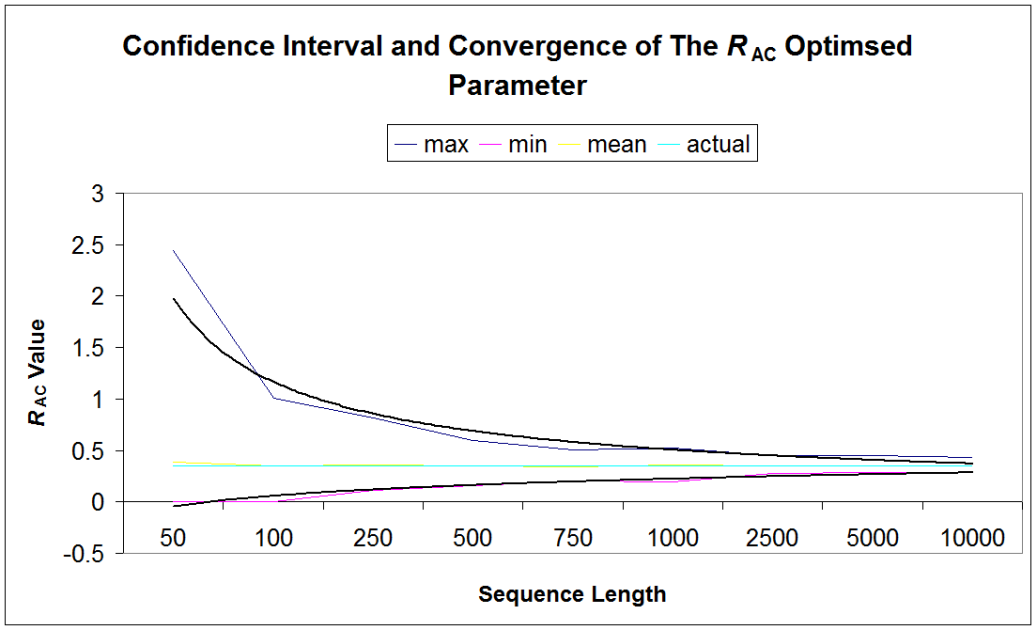


Figure 5.2: Graph showing how the optimised parameter value converges as the sequence length gets longer. This graph is based on the GTR R_{AC} parameter value which was set at 0.35 for each sequence generation based on 100 sequence repeats at each length. The data for all GTR and branch length parameters can be found in appendix A.

from PAML (Yang 2007) and the Java application optimisers to test if the initial parameters come out again (as they should with the generated sequence being tree-like). The fact that each branch length and model parameter converges to its starting place shows that the sequence evolver is generating sequences correctly. For an example illustration of R_{AC} see figure 5.2 and for the convergence graph based on the calculated Root Mean Squared Deviations (RMSD) (calculated using the formula in equation 5.1 below), see figure 5.3. The data obtained is available in Appendix A.

$$RMSD = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}} \tag{5.1}$$

The Test Statistic To test the test statistic and distribution analyses, the program was ran with a single input file, and the various LS estimates of the

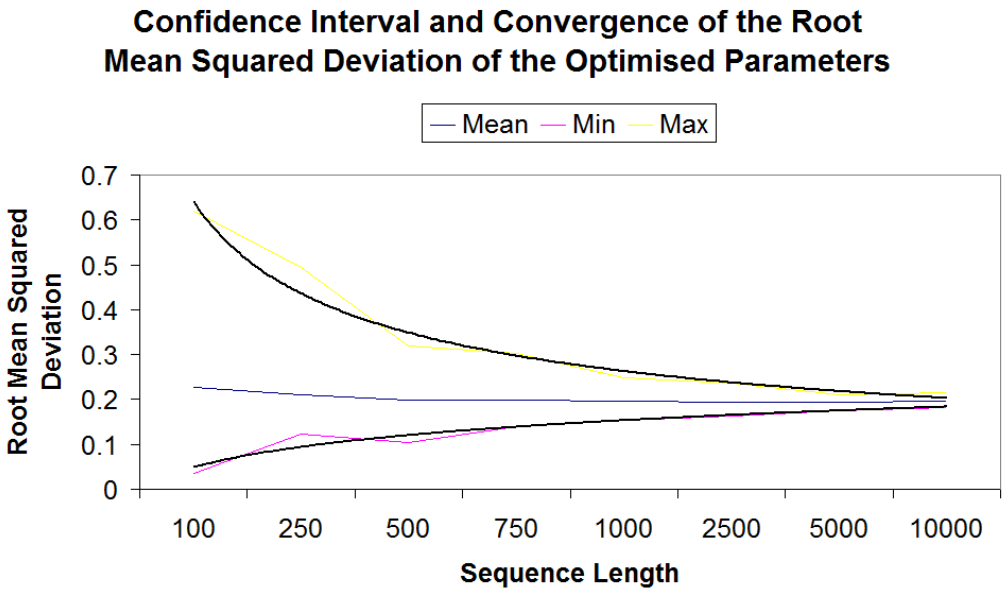


Figure 5.3: Graph showing how the optimised parameter values converge as the sequence length gets longer. This graph uses all the optimised GTR model parameters (except for R_{AG}) and branch lengths for each sequence generation based on 100 sequence repeats at each length combined using the root mean standard deviation statistic in equation 5.1. The data for all GTR and branch length parameters can be found in appendix A.

generated distributions were then taken and plotted onto a graph. This was done for both a treelike dataset (as generated from Seq-Gen (Rambaut & Grassly 1997)) as well as the Brown dataset. The resulting graphs are shown in figures 5.4 to 5.7 showing both the data in the order it was generated (random) and after sorting it (sorted). The sorted data shows how the true LS value (for the initially entered data) is compared to the generated values (the 95% mark). With the treelike data, the true point falls well below the 95% mark whereas the non-treelike data, the resulting LS point is well off the chart suggesting an extreme case and thus is networky. They also show the range of values of LS fit that could result from within treelike data and without too many extreme values suggesting that this is indeed perhaps an appropriate test statistic.

The Seq-Gen alignment was generated using equal base frequencies, the JC model of evolution, no gamma, the tree ((A:0.1, B:0.2):0.6, C:0.3, (D:0.4, E:0.5):0.7) and a sequence length of 895 (same as that of the Brown dataset).

5.2 Optimiser Consistency Check

To ensure that the optimiser always converges to the same optimal values, the program will be run 10 times from different randomised start points. The results of this gave identical final values in each case to within the degree of accuracy that the program calculates showing program consistency.

5.3 Additional Testing Notes

Some additional overall tests were also performed. The main additional test here was to take a simulated sequence alignment (treelike) and to ensure that this came back as treelike by the program. This was tested several times with various internally simulated sequence lengths (for the distribution) of 50 and 1000 and the program consistently reported the initial alignment as treelike, as expected.

However, upon testing against a simulated sequence length of 5000, the

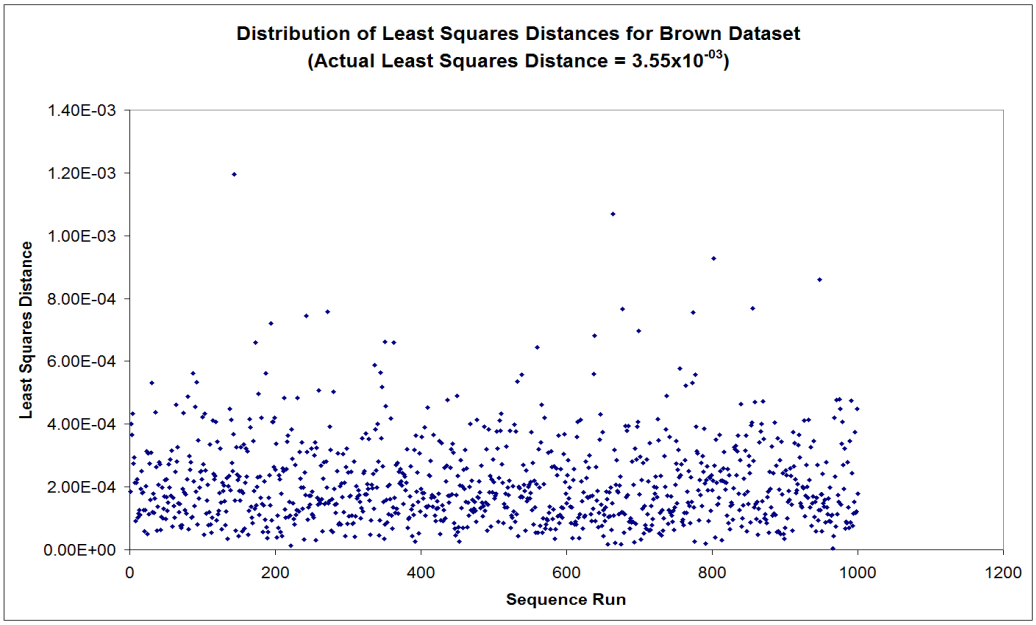


Figure 5.4: Graph showing the least-squares values of 1000 randomly generated sequences based on the Brown dataset after parameter optimisation.

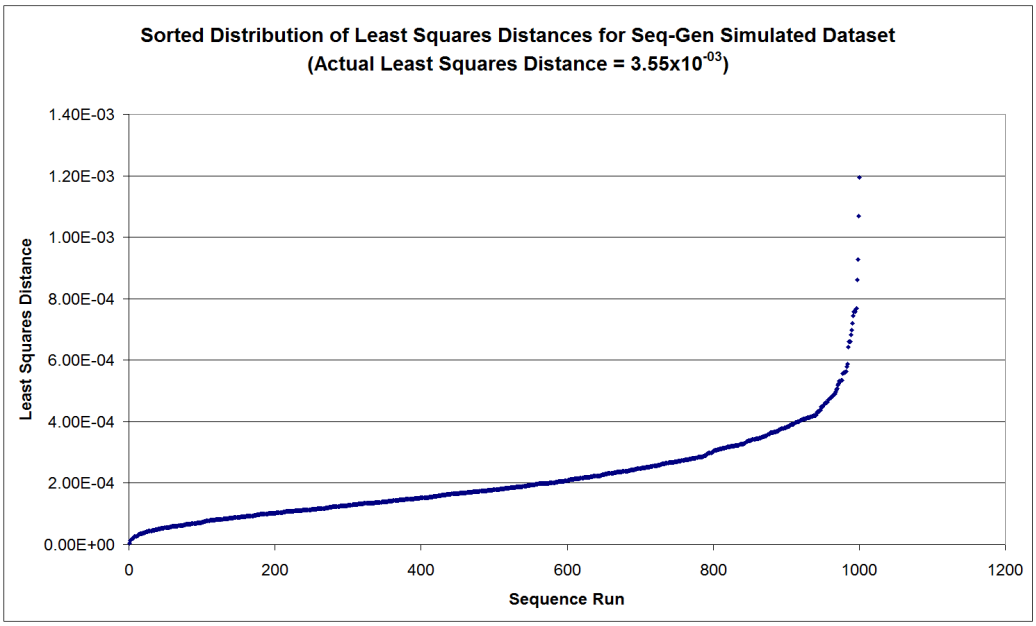


Figure 5.5: Graph showing the Least-squares values of 1000 randomly generated sequences based on the Brown dataset after parameter optimisation.

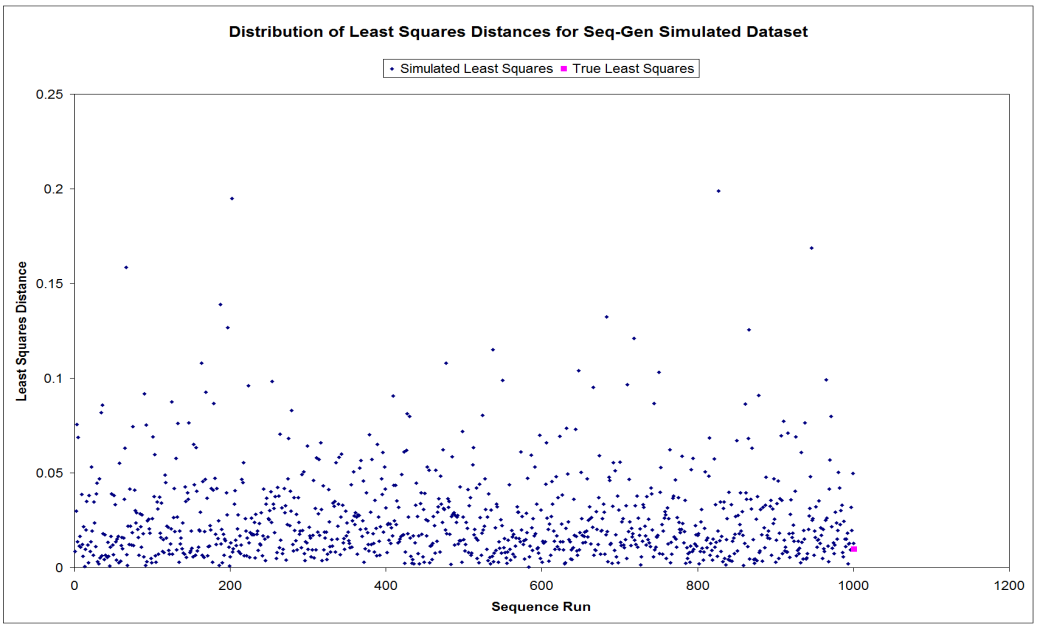


Figure 5.6: Graph showing the least-squares values of 1000 randomly generated sequences based on a sequence alignment generated from Seq-Gen (Rambaut & Grassly 1997) after parameter optimisation.

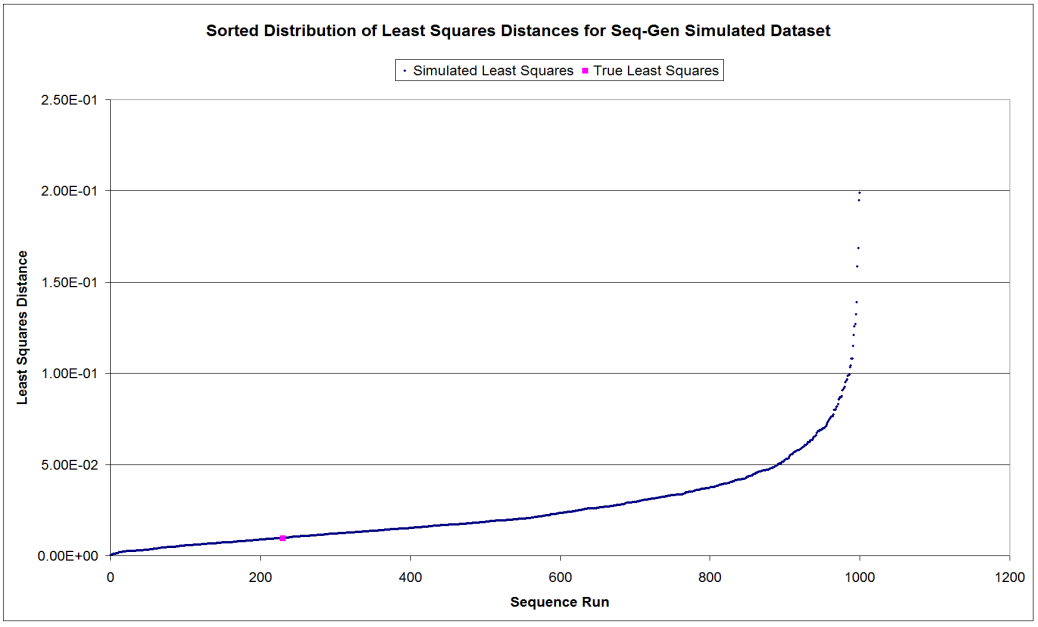


Figure 5.7: Graph showing the least-squares values of 1000 randomly generated sequences based on a sequence alignment generated from Seq-Gen (Rambaut & Grassly 1997) after parameter optimisation.

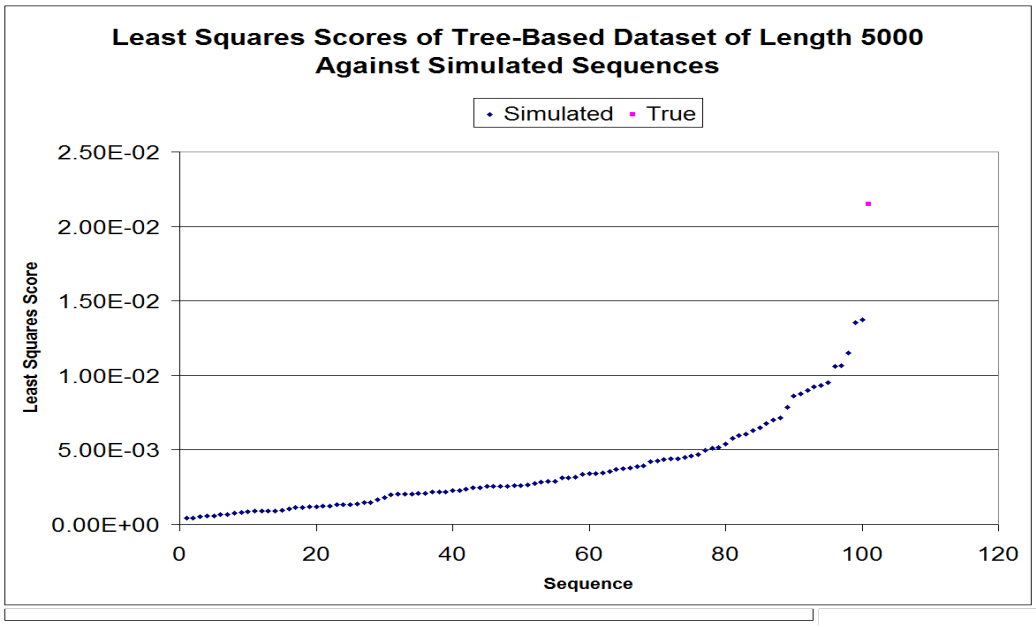


Figure 5.8: Graph showing the least-squares estimates for a simulated sequence of length 1000 against 100 simulated sequences of length 5000 and it appearing to be not treelike despite the initial sequence being generated from a tree.

result came back as *not* treelike, which is surprising for tree based simulated data. The graph in figure 5.8 shows this statistical discrepancy.

Finally, there are some additional points to note which where picked up on during testing. The program will occasionally fail to optimise some of the data parameters as the internal method used fails to converge on an optimum. This appears to most commonly happen when the sequence lengths are small giving too little data on which to optimise the likelihood value. As such, it is suggested to use a minimum sequence length of about 1000 bases to reduce this problem.

5.4 Example Real Data Analysis

Performing a full test optimising GTR and branch lengths (though not the gamma alpha value because it has failed validation) on the Brown dataset resulted in a programmatic analysis of *not treelike*. Figure 5.5 shows this gra-

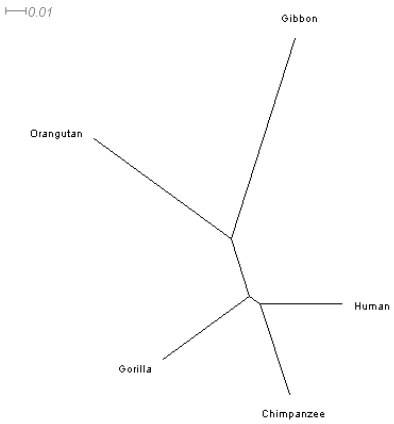


Figure 5.9: Figure shows the SplitsTree4 (Huson & Bryant 2005) representation of the Brown dataset using the Neighbor-Joining algorithm.

phically where the calculated true LS value is well above the range calculated by the null hypothesis. However, if the substitution model parameters are restricted requiring equal base frequencies, the result is then *treelike* instead which is in direct contradiction. The NJ Tree and Neighbor-Net representations of the data are shown in figures 5.9 and 5.10 as drawn in SplitsTree4 (Huson & Bryant 2005).

The *An. annularis* dataset was also analysed with the program. However, the program failed to converge on any parameter values for the dataset after a couple of hours of execution. Due to time constraints and the result of the *An. annularis*, the *An. jeyporiensis* dataset was not analysed. The NJ Tree and Neighbor-Net representations of the *An. annularis* dataset are shown in figures 5.11 and 5.12 as drawn in SplitsTree4 (Huson & Bryant 2005).

5.5 Benchmark Tests

Benchmark tests were run on the Brown dataset with both GTR and branch lengths being optimised. The gamma distribution is off. This will use 100 simulations of sequence length 1000. Results will be taken to the nearest second (as output by the program) and averaged across 10 runs. Various benchmark results are given in table 5.9.

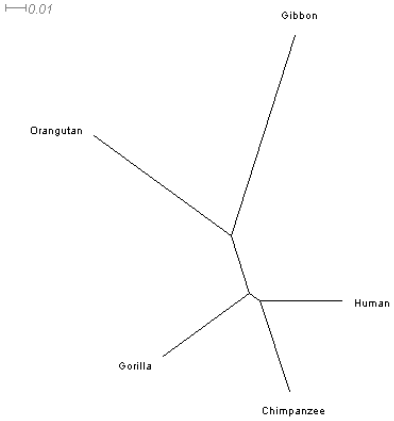


Figure 5.10: Figure shows the SplitsTree4 (Huson & Bryant 2005) representation of the Brown dataset using the Neighbor-Net algorithm.

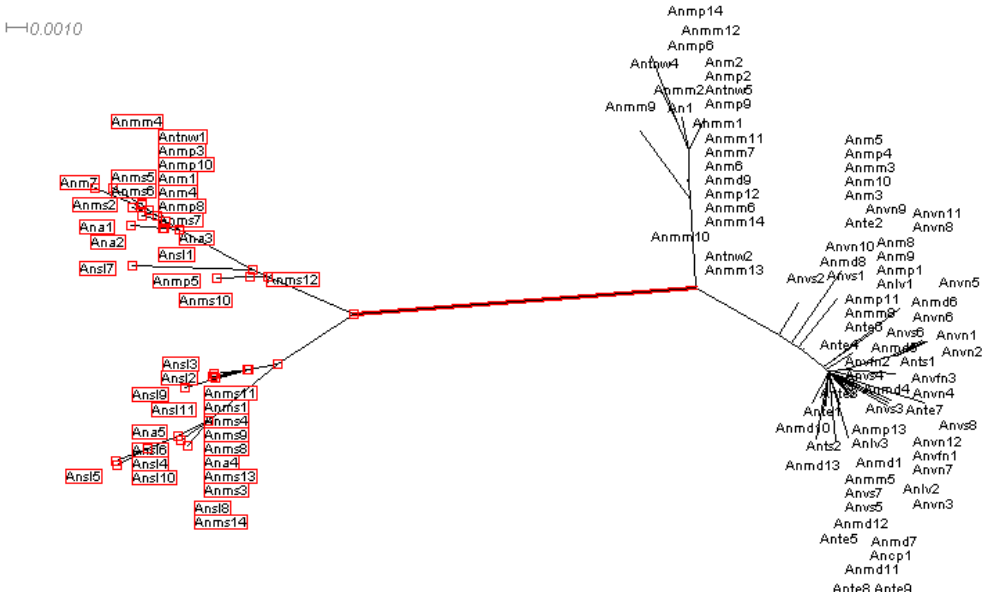


Figure 5.11: Figure shows the SplitsTree4 (Huson & Bryant 2005) representation of the *Anopheles annularis* dataset using the Neighbor-Joining algorithm. There is a clearly defined split between two halves (as highlighted) but otherwise the splits are not so well defined.

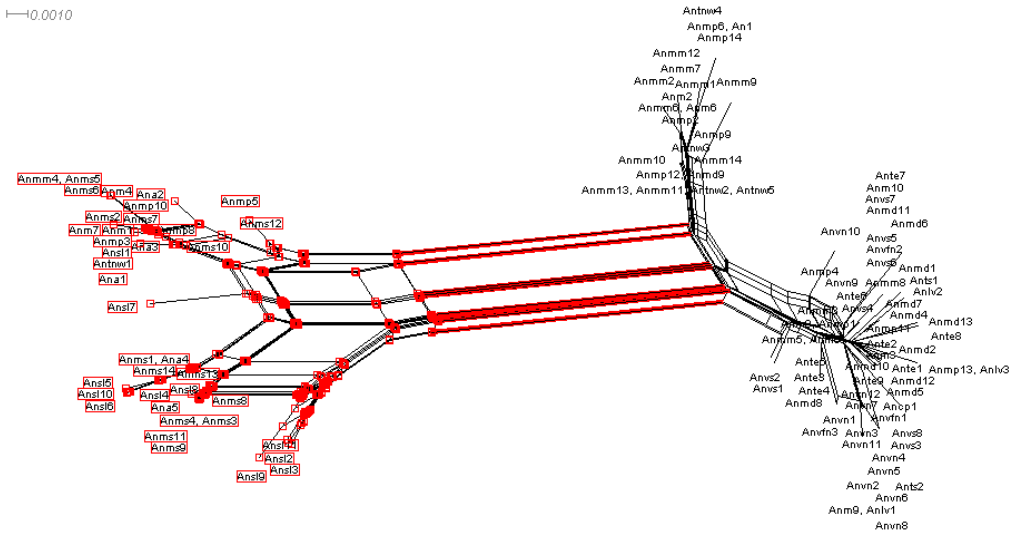


Figure 5.12: Figure shows the SplitsTree4 (Huson & Bryant 2005) representation of the *Anopheles annularis* dataset using the Neighbor-Net algorithm. There is a clearly defined split between two halves (as highlighted) but otherwise the splits are not so well defined.

Processor	Time (secs)			Runs Failed
	Mean	High	Low	
Intel i7 @ 2.66GHz	85.11	86	84	1
AMD QL-66 @ 2.20GHz	246	274	227	0

Table 5.9: Tables giving run times of the program on various Processors. This uses the Brown dataset with no gamma with 100 simulations of size 1000 optimising GTR and branch lengths. Results averaged across 10 runs.

The number of runs failed gives an idea of how often the program errors from having a sequence length that is too small as discussed already in section 5.3. Though rare with a sequence length of 1000 (much more likely to fail within one of the 100 simulated runs than the initial primary run due to there being 100 times the chance), it does still happen.

Chapter 6

Interpretation and Discussion

6.1 Analysis of the Methodology

The methodology as a whole appears sound – using the parametric bootstrap to generate a treelike distribution on which to compare the real data. However, there are some potential issues with the specifics of the method. Firstly, the measure of difference (equation 4.3) is of great importance. If this measure is not appropriate, then the results gained from the analysis may be void and irrelevant. The LS test seems appropriate and there is some evidence to support this (see: Savva et al. unpublished) however it can also clearly be improved. The result of "not treelike" from the program with the tree based dataset (see figure 5.8) shows that there must still be error in the overall analysis.

Given the nature of the methodology, where a long enough sequence will cause the program to optimise to the correct initial parameter values, care needs to be taken when deciding how long to make each of the simulated sequences. If they are generated to be too long, then the variance of the resulting parameter optimisation is greatly reduced causing the final distance comparsion to be directly back to the initial parameters with little or no variance or noise. This could be causing some problems with the internal LS comparison and potentially be voiding the significance calculation. However, a sequence length that is too small causes the program to fail to converge at all, and so a balance must be made between the two (and this is before taking into account the additional time required to do the additional analysis).

6.2 Application Completion

The results of the testing and analysis show that the program works sufficiently well without use of the gamma function. With the gamma function working correctly, the program would be complete and very useful in an initial form although improvements could then still be made (as discussed in section 7).

It is currently unknown as to why (if at all) the gamma function is not working correctly and this is in need of further investigation. Without being

able to fully prove the correctness of the function, it must be assumed to be invalid and so this leaves the application partly incomplete.

Finally, coming back to the aims and objectives of the project, the application successfully optimises all GTR and branch length parameters as well as the sequence pair-wise distances although it does not necessarily correctly optimise the alpha value of the gamma distribution. It does, on the other hand, successfully generate the sequence alignments that describe the null hypothesis (treelike) and will perform the correct statistical test against this to accept or reject the null hypothesis, providing a way of measuring the networkyness of a dataset given a guide tree. The correctness and accuracy of the program have also been analysed and the results seem promising.

However, the secondary aims of the project have not been successfully met as an example dataset analysis (though has been done on the Brown dataset in a little depth,) was not completed for either the *An. annularis* or the *An. jeyporiensis* datasets available. Additionally, no user interface has been implemented into the program at present and is left for future work.

6.3 Example Dataset Discussion

The initial and primary result for the Brown dataset of not treelike is a surprising one because the sequence data is mtDNA which, by its very nature, has no reticulation within. This would suggest that the sequences are likely to be treelike but the analysis result suggests otherwise and if this mtDNA without chance of reticulation is not treelike, then this suggests that very little sequence data, if any, is ever treelike unless it has been artificially generated. However, this has been contradicted by the alternative result when the base frequencies are all set to equal probabilities of 0.25. What I believe that this shows is just that to have the most accurate and reliable result, the most accurate models need to be used. Because of this, the result above has the potential to change once retested using the optimised alpha value of the gamma distribution and only then should the application result be analysed with confidence.

As a result of this, the Brown dataset was additionally analysed using the

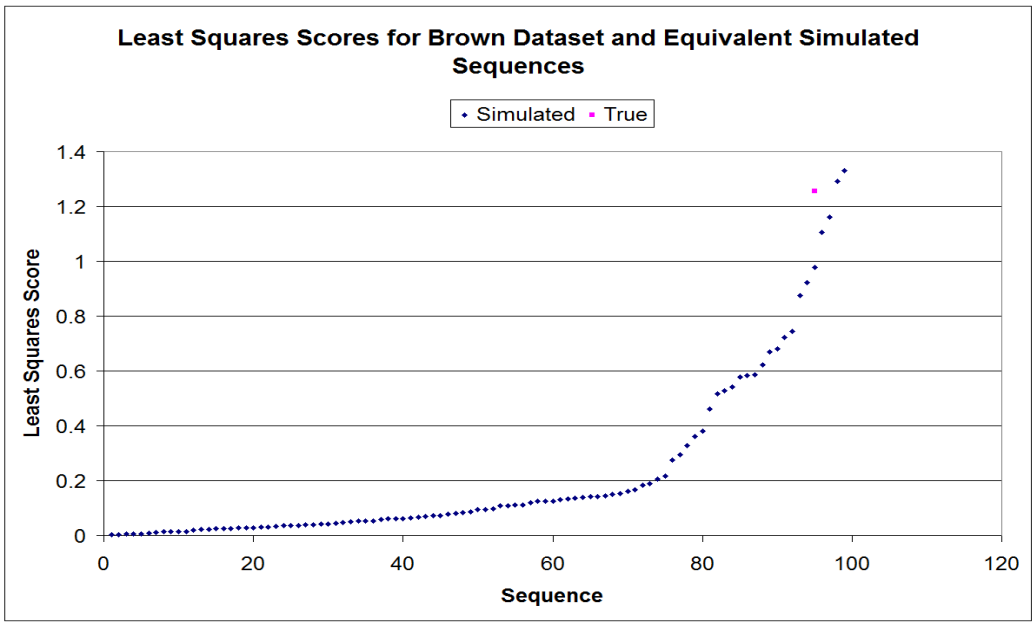


Figure 6.1: Graph showing the least-squares estimates for the Brown dataset against 100 simulated sequences of length 895 using the full GTR+Γ model. Though much closer than before, the final result is still *not treelike*.

full GTR model with gamma (given that it appears to work for the Brown dataset) and a simulated sequence length of 895 for 100 simulated sequences. This resulted in the distribution given in figure 6.1 and the result of *not treelike* produced again, though the simulated and true LS_{dist} values were far closer than before.

In addition to this, there is still the underlying concern with the methodology and this may also be the cause of this discrepancy. After all, looking at the SplitsTree4 network of the Brown data, there is nothing that makes it obviously network like (see figure 5.10).

There could be many reasons as to why the *An. annularis* dataset did not converge within the program. Firstly, it could be a general error with the program that caused this. This could be down to the dataset being too similar and causing the program to fail to converge (it is known that there are a few identical sequences within the dataset and these may be causing issues and will certainly be slowing the progress), or the lack of use of the gamma distribution could also be having an effect. Secondly it could be a more

general issue with the dataset. If the dataset is analysed with the baseml program from the PAML package (Yang 2007), then the optimised branch lengths all come back as very similar in length suggesting high sequence similarity and thus little information on which to optimise the various model parameters.

6.4 Scope and Limitations

At present the application does not optimise tree topology, just tree branch lengths along side GTR model parameters. Due to the complex nature of tree topology optimisation, it was outside the scope of this project.

The application only aims to give a statistical analysis of a given dataset's level of treelikeness – it does not say what needs to be done as a consequence of the analysis or why the results may arise. It is down to the user to investigate the reason for the result and any implications it may have on any future analysis the user may decide to undertake.

Chapter 7

Future Work and Extensions

There are a few ways in which the application could be improved. The first step would be to ensure that all aspects of the program work – most notably the gamma distribution optimiser which is otherwise having some current issues. Once this is up and running, it could also be extended to allow for the invariant sites model. This is where a proportion of sites do not evolve at all and can be used separate or together with the gamma distribution (for overview see: Jayaswal et al. 2007). Additionally, there are various places in which the program could be further optimised for both memory and time efficiency. As well as various code optimisations, one major idea here would be to adjust the program for multi-threading to make use of modern multi-cored computers. This could be done, for example, by either splitting up the tree and pair-wise optimisation steps into two separate threads or by splitting the distribution generation and optimisation into separate threads (or both).

Secondly, the program could also enable the maximum-likelihood estimation of the best tree to describe the data as well as the parameters rather than requiring a guide tree from the user. This would require an implementation similar to that of RAXML (Stamatakis 2006) to enable the best tree to be found rather than assuming that this is going to be prior knowledge before analysis.

As previously mentioned, a suggested extension of the program is to in-

clude a user interface to the software for ease of use as this has not already been implemented. This would have clear usability advantages potentially allowing the application to reach a larger target audience and wider user base.

Separately from this, further alternative distance measures or statistics could be implemented into the program along side the least-squares statistic currently used. Two alternatives compared within the literature is that of *Tree Fit* and *Net Fit* as described by Savva et al. (unpublished). Tree Fit is shown in equation 7.3 and Net Fit is shown in equation 7.4. Both measures require the additional definitions of Sum of Squares (SS) measures given in equations 7.1 and 7.2

$$SS_T = \sum_i \sum_{j>i} (d_{ij} - t_{ij})^2 \tag{7.1}$$

$$SS_N = \sum_i \sum_{j>i} (d_{ij} - n_{ij})^2 \tag{7.2}$$

where $t_{ij} \in T$ given T is a Neighbor-Joining tree and $n_{ij} \in N$ where N is a Neighbor-Net both estimated from a pair-wise alignment distance matrix, D, and thus $d_{ij} \in D$.

$$TF = SS_T \tag{7.3}$$

$$NF = \frac{SS_T - SS_N}{SS_N} \tag{7.4}$$

These measures have been tested within the paper and are thus thought to be better measures for the difference perhaps providing a better test statistic.

Finally, the program at present only works in the one direction – testing for treelikeness. It may, however, also be appropriate to try the method the other way around by generating maximum-likelihood networks instead of maximum-likelihood trees. This approach has been described and detailed in Jin et al.'s (2006) paper and is likely worthy of some investigation.

Additionally, more benchmark results need to be added to the current table and benchmark results of memory usage also need to be included.

Chapter 8

Conclusion

In conclusion, the methodology described here seems sufficient as a test base for data networkyness. The bootstrap approach provides a way to generate datasets adhering to the null hypothesis model against which to compare the initial data allowing a statistic and p-value level of confidence to be placed in the final result of the full hypothesis test. This approach makes use of tree based simulated sequence alignments modelled on the null hypothesis and fully functional parameter optimisation functions for both the branch lengths, pair-wise distances and GTR+Γ parameters. The final application produced provides an easy way to quickly analyse your data for networkyness allowing you to know if you data fulfils the various assumptions of any further phylogenetic based analysis that may be undertaken. Though there is still room for improvement, the application filled most of its primary aims and objectives successfully. This has been verified through in depth module tests of each component in turn by comparing each section to its equivalent program as well as through internal test runs. With a little more work, this application could be of great use within the field of phylogenetics for many years to come. This has been shown through an application run using an example dataset testing for networkyness and a brief discussion of the implications arising from such a result.

The source code for the program is available on request from the author and is also attached to the back of this report.

Bibliography

Adell, J. C. & Dopazo, J. (1994), 'Monte carlo simulation in phylogenies: An application to test the constancy of evolutionary rates', *Journal of Molecular Evolution* **38**, 305–309.

Bandelt, H.-J. & Dress, A. W. (1992), 'A canonical decomposition theory for metrics on a finite set', *Advances in Mathematics* **92**, 47–105.

Brown, W. M., Pragerand, E. M., Wang, A. & Wilson, A. C. (1982), 'Mitochondrial dna sequences of primates: Tempo and mode of evolution', *Journal of Molecular Evolution* **18**, 225–239.

Bryant, D. & Moulton, V. (2004), 'Neighbor-net: An agglomerative method for the construction of phylogenetic networks', *Molecular Biology and Evolution* **21**(2), 255–265.

Bulmer, M. (1991), 'Use of the method of generalized least squares in reconstructing phylogenies from sequence data', *Molecular Biology and Evolution* **8**(6), 868–883.

Dash, A. P., Valecha, N., Anvikar, A. R. & Kumar, A. (2008), 'Malaria in india: Challenges and opportunities', *Journal of Biosciences* **33**(4), 583–592.

Dayhoff, M., Schwartz, R. & B. C, O. (1978), A model of evolutionary change in proteins, *in* 'Atlas of Protein Sequence and Structure', pp. 345–352.

Dopazo, J. (1994), 'Estimating errors and confidence intervals for branch lengths in phylogenetic trees by a bootstrap approach', *Journal of Molecular Evolution* **38**, 300–304.

Drummond, A. J. & Rambaut, A. (2007), 'BEAST: Bayesian evolutionary analysis by sampling trees', *Evolutionary Biology* **7**, 214. Version 1.5.4. `http://beast.bio.ed.ac.uk/Main_Page`.

Efron, B. (1979), 'Bootstrap methods: Another look at the jackknife', *The Annals of Statistics* **7**(1), 1–26.

Felsenstein, J. (1981), 'Evolutionary trees from DNA sequences: A maximum likelihood approach', *Journal of Molecular Evolution* **17**, 368–376.

Felsenstein, J. (1985), 'Confidence limits on phylogenies: An approach using the bootstrap', *Evolution* **39**(4), 783–791.

Felsenstein, J. (2009*a*), 'Fitch – Fitch-Margoliash and Least-Squares distance methods', World-Wide-Web Reference. Version 3.69 - Part of the PHYLIP Package. `http://evolution.genetics.washington.edu/phylip/doc/fitch.html`.

Felsenstein, J. (2009*b*), 'PHYLIP (Phylogeny Inference Package)', Distributed by the author. Department of Genome Sciences, University of Washington, Seattle. Version 3.69. `http://evolution.gs.washington.edu/phylip.html`.

Gauthier, O. & Lapointe, F.-L. (2007), 'Seeing the trees for the network: Consensus, information content, and superphylogenies', *Systematic Biology* **56**(2), 355–363.

Golding, G. (1983), 'Estimates of DNA and protein sequence divergence: An examination of some assumptions', *Molecular Biology and Evolution* **1**(1), 125–142.

Goldman, N. (1993*a*), 'Simple diagnostic statistical tests of models for DNA substitution', *Journal of Molecular Evolution* **37**, 650–661.

Goldman, N. (1993*b*), 'Statistical tests of models of DNA substitution', *Journal of Molecular Evolution* **36**, 182–198.

Goodman, S. N. (1999), 'Toward evidence-based midical statistics. 1: The P value fallacy', *Annals of Internal Medicine* **130**, 995–1004.

Gunasekaran, K., Sahu, S., Parida, S., Sadanandane, C., Jambulingam, P. & Das, P. (1989), 'Anopheline fauna of Koraput district, Orissa state, with particular reference to transmission of malaria', *Indian Journal of Medical Research* **89**, 340–343.

Hasegawa, M., Kishino, H. & aki Yano, T. (1985), 'Dating of the human-ape splitting by a molecular clock of mitochondrial DNA', *Journal of Molecular Evolution* **22**, 160–174.

Higgins, D. G. & Sharp, P. M. (1988), 'CLUSTAL: a package for performing multiple sequence alignment on a microcomputer', *Gene* **73**, 237–244.

Higgs, P. G. & Attwood, T. K. (2005), *Bioinformatics and Molecular Evolution*, Blackwell Publishing, pp. 158–161.

Ho, S. Y. W., Heupink, T. H., Rambaut, A. & Shapiro, B. (2007), 'Bayesian estimation of sequence damage in ancient DNA', *Molecular Biology and Evolution* **24**(6), 1416–1422.

Huson, D. H. & Bryant, D. (2005), 'Application of phylogenetic networks in evolutionary studies', *Molecular Biology and Evolution* **23**(2), 254–267.

Jayaswal, V., Robinson, J. & Jermiin, L. (2007), 'Estimation of phylogeny and invariant sites under the general markov model of nucleotide sequence evolution', *Systematic Biology* **56**(2), 155–162.

Jin, G., Nakhleh, L., Snir, S. & Tuller, T. (2006), 'Maximum likelihood of phylogenetic networks', *Bioinformatics* **22**(21), 2604–2611.

Joyce, J. (2008), Bayes' theorem, *in* E. N. Zalta, ed., 'The Stanford Encyclopedia of Philosophy (Fall 2008 Edition)', World Wide Web: `http://plato.stanford.edu/entries/bayes-theorem/`.

Jukes, T. H. & Cantor, C. R. (1969), Evolution of protein molecules, *in* H. Munro, ed., 'Mammalian Protein Metabolism', Vol. III, Academic Press, New York, pp. 21–132.

Kimura, M. (1980), 'A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences', *Journal of Molecular Evolution* **16**, 111–120.

Le, S. Q. & Gascuel, O. (2008), 'An improved general amino acid replacement matrix', *Molecular Biology and Evolution* **25**(7), 1307–1320.

Livingstone, C. D. & Barton, G. J. (1993), 'Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation', *Computer Applications in the Biosciences* **9**(6), 745–756.

MacKay, D. (2004), 'macopt – a nippy wee optimizer', World-Wide-Web: `http://www.inference.phy.cam.ac.uk/mackay/c/macopt.html`, Last Modified: June 2004, Accessed: September 2010.

Morrison, D. A. (2010), 'Using data-display networks for exploratory data analysis in phylogenetic studies', *Molecular Biology and Evolution* **27**(5), 1044–1057.

Needleman, S. B. & Wunsch, C. D. (1970), 'A general method applicable to the search for similarities in the amino acid sequence of two proteins', *Journal of Molecular Biology* **48**(3), 443–453.

Notredame, C., Higgins, D. G. & Heringa, J. (2000), 'T-Coffee: A novel method for fast and accurate multiple sequence alignment', *Journal of Molecular Biology* **302**, 205–217.

Olsen, G. (1990), "'newick 8:45" tree format standard', World-Wide-Web Reference. `http://evolution.genetics.washington.edu/phylip/newick_doc.html`.

Ostaszewski, K. & Rempala, G. A. (2000), 'Parametric and nonparametric bootstrap in actuarial practice', *www. actuarialfoundation. org/ research_ edu/parametic. pdf* .

Penn, O., Privman, E., Landan, G., Graur, D. & Pupko, T. (2010), 'An alignment confidence score capturing robustness to guide tree uncertainty', *Molecular Biology and Evolution* **27**(8).

Posada, D. & Crandall, K. A. (2001*a*), 'Intraspecific gene genealogies: trees grafting into networks', *Trends in Ecology & Evolution* **16**(1), 37–45.

Posada, D. & Crandall, K. A. (2001*b*), 'Selecting models of nucleotide substitution: An application to Human Immunodeficiency Virus 1 (HIV-1)', *Molecular Biology and Evolution* **18**(6), 897–906.

Rambaut, A. & Grassly, N. C. (1997), 'Seq-Gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees', *Computer Applications in the Biosciences* **13**(3), 235–238. Version 1.3.2. `http://tree.bio.ed.ac.uk/software/seqgen/`.

Rosenberg, M. S. (2005), 'MySSP: Non-stationary evolutionary sequence simulation, including indels', *Evolutionary Bioinformatics Online* **1**, 81–83.

Saitou, N. & Nei, M. (1987), 'The neighbor-joining method: A new method for reconstructing phylogenetic trees', *Molecular Biology and Evolution* **4**(4), 406–425.

Sattler, R. (1984), 'Homology – a continuing challenge', *Systematic Biology* **9**(4), 382–394.

Savva, G., Multon, V., Huber, K. & Dicks, J. (unpublished), A novel statistical measure for testing hypotheses of treelikeness in genomic datasets. Personal Communication – Unpublished 2010: *Systematic Biology*.

Schmidt, H. A., Strimmer, K., Vingron, M. & von Haeseler, A. (2002), 'TREE-PUZZLE: maximum likelihood phylogenetic analysis using quartets and parallel computing', *Bioinformatics* **18**, 502–504. Version 5.2. `http://www.tree-puzzle.de/`.

Shimodaira, H. (2002), 'An approximately unbiased test of phylogenetic tree selection', *Systematic Biology* **51**(3), 492–508.

Shimodaira, H. & Hasegawa, M. (1999), 'Multiple comparisons of log-likelihoods with applications to phylogenetic inference', *Molecular Biology and Evolution* **16**(8), 1114–1116.

Sierk, M. L., Smoot, M. E., Bass, E. J. & Pearson, W. R. (2010), 'Improving pairwise sequence alignment accuracy using near-optimal protein sequence alignments', *Bioinformatics* **11**, 146–160.

Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., Fuellen, G., Gilbert, J. G. R., Korf, I., Lapp, H., Lehväslaiho, H., Matsalla, C., Mungall, C. J., Osborne, B. I., Pocock, M. R., Schattner, P., Senger, M., Stein, L. D., Stupka, E., Wilkinson, M. D. & Birney, E. (2002), 'The Bioperl Toolkit: Perl modules for the life sciences', *Genome Research* **12**, 1611–1618. Version 1.6.1. `http://www.bioperl.org/wiki/Main_Page`.

Stamatakis, A. (2006), 'RAxML-VI-HPC: Maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models', *Bioinformatics* **21**(22), 2688–2690. Version 7.0.4. `http://icwww.epfl.ch/~stamatak/index-Dateien/Page443.htm`.

Stamatakis, A., Hoover, P. & Rougemont, J. (2010), A rapid bootstrap algorithm for the RAxML web-servers. to be published.

Strimmer, K. & von Haeseler, A. (1996), 'Quartet Puzzling: A quartet maximum-likelihood method for reconstructing tree topologies', *Molecular Biology and Evolution* **13**(7), 964–969.

Strope, C. L., Abel, K., Scott, S. D. & Moriyama, E. N. (2009), 'Biological sequence simulation for testing complex evolutionary hypotheses: indel-Seq-Gen version 2.0', *Molecular Biology and Evolution* **26**(11), 2581–2593.

Strope, C. L., Scott, S. D. & Moriyama, E. N. (2007), 'indel-Seq-Gen: A new protein family simulator incorporating domains, motifs, and indels', *Molecular Biology and Evolution* **23**(3), 640–649.

Tavaré, S. (1986), Some probabilistic and statistical problems in the analysis of DNA sequences, *in* 'Lectures on Mathematics in the Life Sciences, vol 17', pp. 57–86.

T.Jones, D., R.Taylor, W. & M.Thornton, J. (1992), 'The rapid generation of mutation data matricies from protein sequences', *Computer Applications in the Biosciences* **8**(3), 275–282.

Wägele, J. W. & Mayer, C. (2007), 'Visualizing differences in phylogenetic information content of alignments and distinction of three classes of long-branch effects', *BMC Evolutionary Biology* **7**, 147.

Wakeley, J. (1993), 'Substitution rate variation among sites in hypervariable region 1 of human mitochondrial DNA', *Journal of Molecular Biology* **37**, 613–623.

Whelan, S. & Goldman, N. (2001), 'A general empirical model of protein evolution derived from multiple families using a maximim-likelihood approach', *Molecular Biology and Evolution* **18**(5), 691–699.

Whelan, S., Liò, P. & Goldman, N. (2001), 'Molecular phylogenetics: state-of-the-art methods for looking into the past', *Trends in Genetics* **17**(5), 262–272.

Yang, Z. (1994), 'Estimating the pattern of nucleotide substitution', *Journal of Molecular Evolution* **39**, 105–111.

Yang, Z. (1996), 'Among-site rate variation and its impact on phylogenetic analyses', *Trends in Ecology & Evolution* **11**(9).

Yang, Z. (2007), 'PAML 4: a program package for phylogenetic analysis by maximum likelihood', *Molecular Biology and Evolution* **24**, 1586–1591. Version 4.4. `http://abacus.gene.ucl.ac.uk/software/paml.html`.

Appendix A

Parameter Convergence to Sequence Length - Data

These table gives the average, actual, minimum and maximum parameter estimates for each parameter from 100 repeats at various simulated sequence lengths showing how the optimiser is consistent. That is, the more data it has available (more sequence) then the closer it will converge to the correct answer. The optimised parameters are all GTR parameters and the branch lengths. The base frequencies were set to: $\pi_A = 0.2$, $\pi_C = 0.3$, $\pi_G = 0.4$ *and* $\pi_T = 0.1$

A.1 GTR Parameters

Length		R_{AC}	R_{AG}	R_{AT}	R_{CG}	R_{CT}	R_{GT}
N/A	Actual	0.35	1	0.25	0.45	0.15	0.05
50	Mean	0.383	1.0	0.237	0.418	0.241	0.042
	Max	2.439	1.0	1.059	1.836	8.395	0.335
	Min	0.000	1.0	0.000	0.000	0.000	0.000
100	Mean	0.354	1.0	0.243	0.473	0.140	0.052
	Max	1.007	1.0	0.717	0.966	0.468	0.312
	Min	0.000	1.0	0.000	0.121	0.000	0.000
250	Mean	0.362	1.0	0.259	0.450	0.155	0.045
	Max	0.812	1.0	0.531	0.732	0.411	0.245
	Min	0.108	1.0	0.000	0.245	0.037	0.000
500	Mean	0.350	1.0	0.252	0.448	0.150	0.051
	Max	0.592	1.0	0.433	0.678	0.281	0.157
	Min	0.157	1.0	0.081	0.300	0.059	0.000
750	Mean	0.342	1.0	0.247	0.443	0.147	0.052
	Max	0.505	1.0	0.442	0.599	0.236	0.116
	Min	0.200	1.0	0.102	0.309	0.066	0.000
1000	Mean	0.355	1.0	0.242	0.457	0.150	0.056
	Max	0.527	1.0	0.366	0.590	0.265	0.115
	Min	0.197	1.0	0.131	0.356	0.080	0.000
2500	Mean	0.347	1.0	0.248	0.449	0.149	0.050
	Max	0.446	1.0	0.360	0.542	0.204	0.083
	Min	0.280	1.0	0.177	0.387	0.114	0.015
5000	Mean	0.347	1.0	0.247	0.449	0.150	0.050
	Max	0.446	1.0	0.328	0.496	0.187	0.073
	Min	0.283	1.0	0.201	0.403	0.113	0.028
10000	Mean	0.347	1.0	0.252	0.450	0.150	0.050
	Max	0.430	1.0	0.309	0.509	0.181	0.074
	Min	0.284	1.0	0.191	0.386	0.114	0.019

Table A.1: Table giving the Mean, Max and Min scores for optimised GTR parameters after 100 simulations at varying sequence lengths. All values to 3 decimal places.

A.2 Branch Lengths

Length		b_H	b_C	b_{Go}	b_O	b_{Gi}	$b_{H,C}$	$b_{O,Gi}$
N/A	Actual	0.1	0.2	0.3	0.4	0.5	0.6	0.7
50	Mean	0.127	0.215	0.551	0.589	0.649	0.772	1.105
	Max	0.642	0.893	5.751	5.424	2.782	3.541	9.846
	Min	0.000	0.000	0.000	0.000	0.000	0.000	0.000
100	Mean	0.116	0.209	0.314	0.429	0.601	0.689	0.804
	Max	0.370	0.487	1.829	1.098	1.682	1.830	2.146
	Min	0.000	0.000	0.000	0.000	0.000	0.065	0.000
250	Mean	0.091	0.214	0.307	0.392	0.564	0.650	0.752
	Max	0.247	0.372	0.761	0.916	1.523	1.343	1.843
	Min	0.000	0.090	0.000	0.000	0.210	0.335	0.307
500	Mean	0.105	0.199	0.301	0.425	0.510	0.612	0.725
	Max	0.179	0.337	0.619	0.722	0.756	1.032	1.247
	Min	0.000	0.109	0.055	0.262	0.243	0.193	0.283
750	Mean	0.105	0.199	0.308	0.403	0.506	0.600	0.740
	Max	0.185	0.276	0.485	0.627	0.711	0.978	1.160
	Min	0.042	0.102	0.098	0.226	0.331	0.428	0.406
1000	Mean	0.095	0.206	0.297	0.400	0.502	0.612	0.714
	Max	0.146	0.301	0.456	0.551	0.666	0.821	0.970
	Min	0.022	0.140	0.162	0.285	0.354	0.426	0.501
2500	Mean	0.099	0.200	0.301	0.398	0.503	0.601	0.703
	Max	0.133	0.245	0.398	0.489	0.598	0.715	0.928
	Min	0.058	0.162	0.196	0.274	0.411	0.494	0.544
5000	Mean	0.101	0.200	0.304	0.402	0.500	0.607	0.702
	Max	0.137	0.241	0.376	0.499	0.576	0.694	0.782
	Min	0.072	0.161	0.224	0.310	0.424	0.503	0.631
10000	Mean	0.102	0.198	0.300	0.401	0.502	0.606	0.711
	Max	0.125	0.233	0.372	0.481	0.582	0.679	0.804
	Min	0.071	0.162	0.229	0.334	0.436	0.534	0.640

Table A.2: Table giving the Mean, Max and Min scores for optimised branch lengths after 100 simulations at varying sequence lengths. All values to 3 decimal places.