Abstract

WILSON, SAMUEL. Molecular Evolution Model Selection and Averaged Estimators. (Under the direction of Spencer Muse.)

A common research interest in molecular evolution is the estimation of substitution rates across a given phylogeny. Many current methods of selecting models to estimate these rates rely on *a priori* designated models or selection methods such as hierarchical likelihood ratio testing (hLRT), information loss analyses, and genetic algorithms (GA). However, model selection methods face multiple challenges. *A priori* selected models implicitly assume that the variability across models and the difference between the selected model and the truth is zero. Furthermore, general methods selecting distinct models have an increased risk that the optimal model may easily change for relatively minor differences in underlying characteristics of the phylogeny or the empirical data [Crandall, 1999, Huelsenbeck et al., 2004, Posada and Buckley, 2004].

This dissertation analyzes model averaged estimates from the traverse of a GA through the set of reversible substitution models. The analysis of these estimators will show a robust method that mitigates the risks and challenges of estimating substitution rates from many of the current methods of model selection. Further discussions on the way ahead and future research are included. Secondary analyses include a thorough mapping on the census of nucleotide reversible models for a broad range of simulated conditions. This will include a discussion on the general behavior observed for the population of reversible models. Additionally, exploratory analyses will include various methods of GA implementation for the set of reversible codon models.

[Model Selection; Model Average Estimate; Phylogenetic Inference; Molecular Evolution; Reversible Markov Process]

Molecular Evolution Model Selection and Averaged Estimators

by
Samuel Wilson

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Statistics

Raleigh, North Carolina

2016

APPROVED BY:

Alison Motsinger	Brian Reich
Hua Zhou	Cynthia Hemenway

Spencer Muse
Chair of Advisory Committee

Dedication

For Kris, Bailey, and Tobo: I love you more than I can express.

Biography

Sam Wilson was born in Orange County, California in 1976 to Haylee and Jerry Wilson. Following his graduation from high school in 1994, he enrolled in the United States Military Academy at West Point and in 1998 earned a Bachelor's Degree in Mathematics.

For the next six years, he served as an armor officer in the US Army where he participated in multiple overseas missions including Operations Desert Spring (Kuwait, 2000), Joint Forge (Bosnia and Herzegovina, 2002), and Enduring Freedom (Iraq, 2003). Following his discharge from the military in 2003, he enrolled in the 2004 Spring semester at Kansas State University and earned a Masters Degree in Statistics in 2006. Before enrolling in the PhD program at North Carolina State University, he did an internship at Hills Science Diet where he enjoyed playing with the dog and cat colonies while working on various statistical principles for the design of experiments.

While working toward his PhD, Sam joined the pharmaceutical industry as a statistician in 2008 and has since worked for various companies in the field, managing statistical teams for a wide range of clinical trials of all phases. Currently, he is an Associate Director of Health Economic and Clinical Outcomes Research (HECOR) for Astellas Pharma. In this capacity he is performing a wide range of statistical duties for the Medical Affairs department that include protocol development, power and sample size analyses, survival, mixed-model, longitudinal, and survival analyses. His experience spans all phases of clinical trials and a wide range of therapeutic areas including Oncology, Transplant, Neurology, Immunology, Hematology, Internal Medicine, Cardiovascular, Dermatology, Emergency Medicine, Endocrinology, Ophthalmology, and various pharmacokinetic, dose response and bioequivalence analyses.

Acknowledgements

This report is only possible due to the time, support, expertise, mentorship, and patience of so many people.

I would like to thank Kris and Bailey for putting me back together following the war and your compassion and support getting me through my post-graduate education. You have sacrificed in nearly every way imaginable to get to this point in time: I love you.

Thank you to Tobo for your unwavering dedication and love through many many late nights at school, work, and every aspect of my life.

Thank you to my dear brother, Kobbe, for your support, review, and personal time that you generously gave to me for this dissertation. Thank you to my parents, Jerry and Haylee, and brother, Simon, for your unwavering support through all the long years of my education and life. You have been with me since the very beginning and given me everything. Thank you to the rest of my family: Banjo, Myles, Millie, Nora, and Deputy for your love.

Thank you to my advisor, Dr. Spencer Muse, who provided an endless supply of patience, guidance, and consistency to keep me on track to complete this dissertation despite the extended time it has taken and my manic personality. I also want to thank Drs. Alison Motsinger, Brian Reich, Hua Zhou, and Cynthia Hemenway for their time, review comments, and expertise throughout this process.

Thank you to the statistics departments at Kansas State University and North Carolina State University for the training, mentorship, and second chances to get here.

Table of Contents

List of Tables viii

List of Figures ix

Chapter 1 Background 1
1.1 Introduction 1
1.2 Modeling the Evolution of DNA Sequences 4
1.2.1 Data and Tree Structure 4
1.2.2 Nucleotide Substitution Models 6
1.2.3 Codon Substitution Models 10
1.2.4 Model Space 14
1.3 Likelihood 16
1.3.1 Maximizing Likelihood 18
1.4 Estimation 19
1.4.1 Transition/Transversion Rates 19
1.4.2 Nonsynonymous/Synonymous Rates 21
1.4.3 Implementation 24
1.5 Model Selection 24
1.5.1 Hierarchical Likelihood Ratio Tests 25
1.5.2 Bayesian Markov Chain Monte Carlo 26
1.5.3 Information Loss 27
1.5.4 Genetic Algorithms 29
1.5.5 CodonTest 31
1.5.6 Model Averaging 33
1.6 Discussion 36

Chapter 2 Nucleotide Analyses 38
2.1 Data Simulation Settings 38
2.2 Simulation Results 43
2.2.1 Transition/Transversion Ratios 43
2.2.2 Variance - Transition/Transversion Ratios 64
2.2.3 Euclidean Distance Comparisons 67
2.2.4 Empirical Results 74
2.3 Discussion 77

Chapter 3 Codon Analyses 80
3.1 Data Simulation Settings 81
3.2 Simulation Results 83
3.2.1 Nonsynonymous/Synonymous Ratios 83
3.2.2 Information Loss by Nonsynonymous Clusterings 90
3.2.3 Genetic Algorithm Performance 95
3.3 Discussion 98

Chapter 4 Conclusions 100

References 103

Appendices 111
Appendix A Nucleotide Simulation Results 112
A.1 Transition/Transversion Estimates 112
A.1.1 HKY85 Data Analyses 112
A.1.2 TN93 Data Analyses 125
A.1.3 GTR Data Analyses 138
A.2 Variance of Transition/Transversion Estimates 151
A.2.1 HKY85 Variance Analyses 151
A.2.2 TN93 Variance Analyses 168
Appendix B Codon Simulation Results 173
B.1 Nonsynonymous/Synonymous Ratios 173
B.2 Information Loss by Nonsynonymous Clusterings 178
Appendix C HyPhy 187
Appendix D Nucleotide Modeling Programs 190
D.1 SimoData.R 190
D.1.1 Utility Function: translate.R 192
D.1.2 Utility Function: generatetaxa.R 193
D.2 NucModelSpace.h 193
D.2.1 Utility Function: pop.h 197
D.2.2 Utility Function: MatrixMapNuc.h 203
D.2.3 Utility Function: sum.h 204
D.3 nuc table.h 204
Appendix E Codon Modeling Programs 217
E.1 SimoData.R 217
E.1.1 Utility Function: translate.R 223
E.1.2 Utility Function: transl nuc codon.R 225
E.1.3 Utility Function: generate taxa.R 226
E.1.4 Utility Function: amino test cpu.R 226
E.1.5 Utility Vector: s cnt.txt 229
E.1.6 Utility Vector: n cnt.txt 230
E.2 GA HYPHY codon.h 230
E.2.1 Utility Function: CreateTemplates.h 246
E.2.2 Utility Function: MatrixMap codon.h 248
E.2.3 Utility Function: MatrixMap dNdS marker.h 250
E.2.4 Utility Function: MatrixMap codon EFM.h 251
E.2.5 Utility Function: IsChildViable.h 255
E.2.6 Utility Function: MakeStringCanonical.h 255
E.2.7 Utility Function: ModelGeneticCode.h 256
E.2.8 Utility Function: one step dNdS.h 257
E.2.9 Utility Function: SimpleCodonFreq.h. 259

E.2.10 Utility Function: One Step Cnts Matrix.h 260
E.2.11 Utility Function: SumNucCnts.h 260
E.2.12 Utility Function: dNdS est utility.h 261
E.2.13 Utility Function: EFM assign.h 262
E.2.14 Utility Function: sum.h 263
E.3 Results Data loops.R 264
E.4 Results Data Analysis.R 265

List of Tables

Table 1.1 Example nucleotide data on 5 taxa 4

Table 2.1 Sequence Divergence 40
Table 2.2 Sequence Divergence for Extended Values of Case I 41
Table 2.3 Reversible Nucleotide Models 44
Table 2.4 HKY85 Simulation Settings 46
Table 2.5 Estimated Ts/Tv Comparisons for HKY85 Data with E(To) = 0.138 to 1.376 47
Table 2.6 Estimated Ts/Tv Comparisons for HKY85 Data with E(To) = 1.72 to 4.128 48
Table 2.7 TN93 Simulation Settings 52
Table 2.8 Estimated Ts/Tv Comparisons for TN93 Data 53
Table 2.9 Estimated Ts/Tv Comparisons for TN93 Data for E(To) = 1.607 to 3.858 54
Table 2.10 GTR Simulation Settings 58
Table 2.11 Estimated Ts/Tv Comparisons for GTR Data 59
Table 2.12 Estimated Ts/Tv Comparisons for GTR Data for E(To) = 1.289 to 3.75 . 60
Table 2.13 Estimated L2 Norm Comparisons for HKY85 Data for E(To) = 0.138 to 1.376 68
Table 2.14 Estimated L2 Norm Comparisons for HKY85 Data for E(To) = 1.72 to 4.128 69
Table 2.15 Estimated L2 Norm Comparisons for TN93 Data for E(To) = 0.129 to 1.286 70
Table 2.16 Estimated L2 Norm Comparisons for TN93 Data for E(To) = 1.607 to 3.858 71
Table 2.17 Estimated L2 Norm Comparisons for GTR Data for E(To) = 0.125 to 1.250 72
Table 2.18 Estimated L2 Norm Comparisons for GTR Data for E(To) = 1.289 to 3.75 73
Table 2.19 Empirical Analyses 75

Table 3.1 dN/dS Estimate Results for a Two Leaf Tree 84
Table 3.2 dN/dS Estimate Results for a Four Leaf Tree 84
Table 3.3 GA Performance - Two Leaf Tree 96
Table 3.4 GA Performance - Four Leaf Tree 97

List of Figures

Figure 1.1 Example evolutionary tree 5
Figure 1.2 Markov Process 6
Figure 1.3 Purine and Pyrimidine Substitutions 9
Figure 1.4 Rooted evolutionary tree 18
Figure 1.5 Unrooted evolutionary tree 18

Figure 2.1 Huelsenbeck and Hillis [1993] Unrooted Four-Taxa Phylogenetic Tree . . . 40
Figure 2.2 Ts/Tv Estimates on HKY85 Data with Sequence Length 500 49
Figure 2.3 Ts/Tv Estimates on HKY85 Data with Sequence Length 1000 50
Figure 2.4 Ts/Tv Estimates on TN93 Data with Sequence Length 500 55
Figure 2.5 Ts/Tv Estimates on TN93 Data with Sequence Length 3000 56
Figure 2.6 Ts/Tv Estimates on GTR Data with Sequence Length 500 61
Figure 2.7 Ts/Tv Estimates on GTR Data with Sequence Length 3000 62
Figure 2.8 Variance Histograms on GTR Data with Sequence Length 500 65
Figure 2.9 Variance Boxplots on GTR Data with Sequence Length 500 and E(To) = 0.125 to 1.250 66
Figure 2.10 Empirical Analysis: Rat/Mouse Data 75
Figure 2.11 Empirical Analyses 76

Figure 3.1 dN/dS Estimate Results for a Two Leaf Tree for Clusters of Size 2 to 7 . 85
Figure 3.2 dN/dS Estimate Results for a Two Leaf Tree for Clusters of Size 8 to 12 . 86
Figure 3.3 dN/dS Estimate Results for a Four Leaf Tree for Clusters of Size 2 to 7 . 87
Figure 3.4 dN/dS Estimate Results for a Four Leaf Tree for Clusters of Size 8 to 11 . 88
Figure 3.5 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree for Clusters of Size 2 to 7 91
Figure 3.6 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree for Clusters of Size 8 to 12 92
Figure 3.7 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree 93
Figure 3.8 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree 94

Figure A.1 Ts/Tv Estimates on HKY85 Data with Sequence Length 250 and E(To) = 0.138 to 1.376 113
Figure A.2 Ts/Tv Estimates on HKY85 Data with Sequence Length 250 and E(To) = 1.72 to 4.128 114
Figure A.3 Ts/Tv Box Plots on HKY85 Data with Sequence Length 250 and E(To) = 0.069 to 1.376 115
Figure A.4 Ts/Tv Estimates on HKY85 Data with Sequence Length 500 and E(To) = 0.138 to 1.376 116
Figure A.5 Ts/Tv Estimates on HKY85 Data with Sequence Length 500 and E(To) = 1.72 to 4.128 117

Figure A.6 Ts/Tv Box Plots on HKY85 Data with Sequence Length 500 and E(To) = 0.069 to 1.376 . . . 118
Figure A.7 Ts/Tv Estimates on HKY85 Data with Sequence Length 1000 and E(To) = 0.138 to 1.376 . . . 119
Figure A.8 Ts/Tv Estimates on HKY85 Data with Sequence Length 1000 and E(To) = 1.72 to 4.128 . . . 120
Figure A.9 Ts/Tv Box Plots on HKY85 Data with Sequence Length 1000 and E(To) = 0.069 to 1.376 . . . 121
Figure A.10 Ts/Tv Estimates on HKY85 Data with Sequence Length 3000 and E(To) = 0.138 to 1.376 . . . 122
Figure A.11 Ts/Tv Estimates on HKY85 Data with Sequence Length 3000 and E(To) = 1.72 to 4.128 . . . 123
Figure A.12 Ts/Tv Box Plots on HKY85 Data with Sequence Length 3000 and E(To) = 0.138 to 1.376 . . . 124
Figure A.13 Ts/Tv Estimates on TN93 Data with Sequence Length 250 and E(To) = 0.129 to 1.286 . . . 126
Figure A.14 Ts/Tv Estimates on TN93 Data with Sequence Length 250 and E(To) = 1.607 to 3.858 . . . 127
Figure A.15 Ts/Tv Box Plots on TN93 Data with Sequence Length 250 and E(To) = 0.129 to 1.286 . . . 128
Figure A.16 Ts/Tv Estimates on TN93 Data with Sequence Length 500 and E(To) = 0.129 to 1.286 . . . 129
Figure A.17 Ts/Tv Estimates on TN93 Data with Sequence Length 500 and E(To) = 1.607 to 3.858 . . . 130
Figure A.18 Ts/Tv Box Plots on TN93 Data with Sequence Length 500 and E(To) = 0.129 to 1.286 . . . 131
Figure A.19 Ts/Tv Estimates on TN93 Data with Sequence Length 1000 and E(To) = 0.129 to 0.286 . . . 132
Figure A.20 Ts/Tv Estimates on TN93 Data with Sequence Length 1000 and E(To) = 1.607 to 3.858 . . . 133
Figure A.21 Ts/Tv Box Plots on TN93 Data with Sequence Length 1000 and E(To) = 0.129 to 1.286 . . . 134
Figure A.22 Ts/Tv Estimates on TN93 Data with Sequence Length 3000 and E(To) = 0.129 to 1.286 . . . 135
Figure A.23 Ts/Tv Estimates on TN93 Data with Sequence Length 3000 and E(To) = 1.607 to 3.858 . . . 136
Figure A.24 Ts/Tv Box Plots on TN93 Data with Sequence Length 3000 and E(To) = 0.129 to 1.286 . . . 137
Figure A.25 Ts/Tv Estimates on GTR Data with Sequence Length 250 and E(To) = 0.125 to 1.250 . . . 139
Figure A.26 Ts/Tv Estimates on TN93 Data with Sequence Length 3000 and E(To) = 1.563 to 3.750 . . . 140
Figure A.27 Ts/Tv Box Plots on GTR Data with Sequence Length 250 and E(To) = 0.125 to 1.250 . . . 141

Figure A.28 Ts/Tv Estimates on GTR Data with Sequence Length 500 and E(To) = 0.125 to 1.250 . . . 142
Figure A.29 Ts/Tv Estimates on GTR Data with Sequence Length 3000 and E(To) = 1.563 to 3.750 . . . 143
Figure A.30 Ts/Tv Box Plots on GTR Data with Sequence Length 500 and E(To) = 0.125 to 1.250 . . . 144
Figure A.31 Ts/Tv Estimates on GTR Data with Sequence Length 1000 and E(To) = 0.125 to 1.250 . . . 145
Figure A.32 Ts/Tv Estimates on GTR Data with Sequence Length 3000 and E(To) = 1.563 to 3.750 . . . 146
Figure A.33 Ts/Tv Box Plots on GTR Data with Sequence Length 1000 and E(To) = 0.125 to 1.250 . . . 147
Figure A.34 Ts/Tv Estimates on GTR Data with Sequence Length 3000 and E(To) = 0.125 to 1.250 . . . 148
Figure A.35 Ts/Tv Estimates on GTR Data with Sequence Length 3000 with E(To) = 1.563 to 3.750 . . . 149
Figure A.36 Ts/Tv Box Plots on GTR Data with Sequence Length 3000 and E(To) = 0.125 to 1.250 . . . 150
Figure A.37 Variance Histograms on HKY85 Data with Sequence Length 250 . . . 152
Figure A.38 Variance Histograms on HKY85 Data with Sequence Length 250 for E(To) = 1.72 to 4.128 . . . 153
Figure A.39 Variance Boxplots on HKY85 Data with Sequence Length 250 for E(To) = 0.138 to 1.376 . . . 154
Figure A.40 Variance Boxplots on HKY85 Data with Sequence Length 250 and E(To) = 1.720 to 4.128 . . . 155
Figure A.41 Variance Histograms on HKY85 Data with Sequence Length 500 with E(To) = 0.138 to 1.376 . . . 156
Figure A.42 Variance Histograms on HKY85 Data with Sequence Length 500 for E(To) = 1.72 to 4.128 . . . 157
Figure A.43 Variance Boxplots on HKY85 Data with Sequence Length 500 with E(To) = 0.138 to 1.376 . . . 158
Figure A.44 Variance Boxplots on HKY85 Data with Sequence Length 500 and E(To) = 1.720 to 4.128 . . . 159
Figure A.45 Variance Histograms on HKY85 Data with Sequence Length 1000 with E(To) = 0.138 to 1.376 . . . 160
Figure A.46 Variance Histograms on HKY85 Data with Sequence Length 1000 for E(To) = 1.72 to 4.128 . . . 161
Figure A.47 Variance Boxplots on HKY85 Data with Sequence Length 1000 with E(To) = 0.138 to 1.376 . . . 162
Figure A.48 Variance Boxplots on HKY85 Data with Sequence Length 1000 and E(To) = 1.720 to 4.128 . . . 163
Figure A.49 Variance Histograms on HKY85 Data with Sequence Length 3000 . . . 164
Figure A.50 Variance Histograms on HKY85 Data with Sequence Length 3000 for E(To) = 1.72 to 4.128 . . . 165

Figure A.51 Variance Boxplots on HKY85 Data with Sequence Length 3000 with E(To) = 0.138 to 1.376 . . . 166
Figure A.52 Variance Boxplots on HKY85 Data with Sequence Length 3000 and E(To) = 1.720 to 4.128 . . . 167
Figure A.53 Variance Histograms on TN93 Data with Sequence Length 250 and E(To) = 0.129 to 1.286 . . . 169
Figure A.54 Variance Histograms on HKY85 Data with Sequence Length 250 for E(To) = 1.607 to 3.858 . . . 170
Figure A.55 Variance Boxplots on TN93 Data with Sequence Length 250 and E(To) = 0.129 to 1.286 . . . 171
Figure A.56 Variance Boxplots on TN93 Data with Sequence Length 250 and E(To) = 1.607 to 3.858 . . . 172

Figure B.1 dN/dS Estimate Results for a Two Leaf Tree for Clusters of Size 2 to 7 . 174
Figure B.2 dN/dS Estimate Results for a Two Leaf Tree for Clusters of Size 8 to 12 . 175
Figure B.3 dN/dS Estimate Results for a Four Leaf Tree for Clusters of Size 2 to 7 . 176
Figure B.4 dN/dS Estimate Results for a Four Leaf Tree for Clusters of Size 8 to 11 . 177
Figure B.5 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree for Clusters of Size 2 to 7 . . . 179
Figure B.6 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree for Clusters of Size 8 to 12 . . . 180
Figure B.7 Population Best-Fit mBIC by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree . . . 181
Figure B.8 Population Best-Fit mBIC by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree . . . 182
Figure B.9 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree . . . 183
Figure B.10 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree . . . 184
Figure B.11 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree . . . 185
Figure B.12 Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree . . . 186

Chapter 1

Background

1.1 Introduction

Molecular evolution is the study of the evolution of macromolecules and the reconstruction of gene and organism history. An area of focus in this field is the estimation of substitution rates. This dissertation presents model selection and model-averaged estimators for these rates as comparators to current methods, and describes the properties of point estimation and variability for these estimates.

A better understanding of modeling the molecular evolution of deoxyribonucleic acid (DNA) sequences will provide practical benefits to science. For example, the prediction of substitutions within organisms could be valuable to current work by the Centers for Disease and Control (CDC) in their preparation for each annual influenza season. Additionally, better prediction and knowledge of the evolution observed within the Human Immunodeficiency Virus (HIV) could aide in efforts to eradicate or minimize its effects. A better understanding of substitution rates can also aid in the identification of positive selective pressures for a given phylogeny as well as provide scale for estimates of evolutionary distance between and within organisms.

Relevant biological characteristics play a major role in the evolution of molecular sequences and the rates of substitutions. The usage of these characteristics and the identification of their

presence will be used to efficiently traverse candidate models in this dissertation. Specifically, at the nucleotide level, possible substitutions are often parameterized on whether they are transitions versus transversions [Kimura, 1980, Hasegawa et al., 1985, Tamura and Nei, 1993]. At the codon level, substitutions are often parameterized on whether or not the change results in the encoding of a different amino acid (nonsynonymous versus synonymous) [Muse and Gaut, 1994, Goldman and Yang, 1994].

Regarding the evolution of nucleotides, numerous methods have been presented to describe the dynamics of observed sequence alignments. The simplest of the nucleotide models was defined by Jukes and Cantor [1969] and uses a single parameter to describe the relative frequency of any nucleotide change. Since then, various models have been developed that are based on biological characteristics [e.g. Kimura, 1980, Hasegawa et al., 1985, Tamura and Nei, 1993, Tavare, 1986]. Additionally, given the assumptions of a Markov process, the full set of 203 models is well defined with many of the named ones being special cases of the general Markov process.

For the codon case, substitution modeling has different challenges. Computationally, the enumeration of possible models is too large to be traversed (unlike the nucleotide case's 203 models). Also, the frequency of relative substitution rates take into account additional biological constraints that include stop codon differences and within-codon correlations [Muse and Gaut, 1994, Muse, 1996, Schneider et al., 2005, Rodrigue et al., 2008]. Given these challenges, many modeling methods rely on deductive biological reasoning that includes characteristics such as transitions, transversions, nonsynonymous, and/or synonymous substitutions. Additionally, many models use narrowly focused parameterizations of the mathematically possible substitution rates of the phylogeny.

These nucleotide and codon models all specify their parameterizations *a priori*. The implicit assumption for this methodology of pre-selected models is that the variability across models and the difference between the selected model and the truth is zero [Hjort and Claeskens, 2003]. This methodology has risk that resulting estimators will have higher bias, and statistical tests

may not be size alpha.

Given the assumptions required for *a priori* selected models, numerous model selection methods have been proposed over the years. These span a variety of approaches including hierarchical likelihood ratio testing (hLRT) [Posada and Crandall, 1998, Anisimova et al., 2001, Posada, 2008, Yang, 1997, 2007], Bayesian Markov chain Monte Carlo (MCMC) estimation [Suchard et al., 2001, Huelsenbeck and Ronquist, 2001, Huelsenbeck et al., 2004, Rodrigue et al., 2007, Drummond and Rambaut, 2007, Ronquiet et al., 2012], information loss [Buckland et al., 1997, Posada and Crandall, 2001, Posada and Buckley, 2004, Abascal et al., 2005, Burnham and Anderson, 2010], and genetic algorithms (GAs) [Kosakovsky Pond and Frost, 2004, Kosakovsky Pond et al., 2006, 2007, Delport et al., 2010b]. However, several investigators have identified common situations across these methods consisting of selected models changing based on the particular phylogeny and/or empirical dataset analyzed [Crandall, 1999, Posada and Buckley, 2004, Huelsenbeck et al., 2004]. This represents one of many challenges facing researchers in this field.

The objective of this dissertation is to use model averaging within a model selection method in order to find robust estimates of quantities shared by all of the models in the candidate set used in the selection method. Specifically, the performance of model averaged estimates that use the traverse of a GA through the reversible codon substitution models will be analyzed for different phylogenetic and data settings in order to take advantage of the unique benefits that have been noted for GAs within molecular evolution [Kosakovsky Pond and Frost, 2004, Kosakovsky Pond et al., 2006, 2007, Delport et al., 2010b]. The secondary objective is the analysis of the computationally simple nucleotide models in order to analyze the benefits of model averaging over the census of reversible models as a proof of concept for the codon case.

Chapters two and three of this dissertation provide the results of these nucleotide and codon analyses. The data simulation settings and scale of simulated molecular evolutionary rates are based on published results and methodologies. Additionally, the empirical analyses included in this dissertation are based on sourced data from GenBank®, a publicly available National

Institutes of Health database of DNA sequence data. Chapter four will include a discussion of the overall findings, contributions to the field of molecular evolution, and the way forward for future research.

1.2 Modeling the Evolution of DNA Sequences

In order to understand the use of model averaging and GAs over reversible substitution models, a review of the assumptions, models, and parameterizations are included.

1.2.1 Data and Tree Structure

Consider the nucleotide sequences of the four taxa shown in table 1.1. Each data point in the table represents a nucleotide site with abbreviations that will be used throughout this dissertation: A is Adenine, C is Cytosine, G is Guanine, and T is Thymine. Nucleotide sites are represented by the columns in the sequence (e.g. the first site in table 1.1 is $\{A, A, A, A\}$). Note that the first change across the four taxa is in the 17^{th} column (first column highlighted in yellow) with Human - G, Chimpanzee - A, Gorilla - G, and Orangutan - A. Let y_g^k represent the nucleotide for the k^{th} site ($k = 1, 2...K$) located at node g ($g = 1, 2..., G$), such that $y_g^k \in \{A, C, G, T\}$.

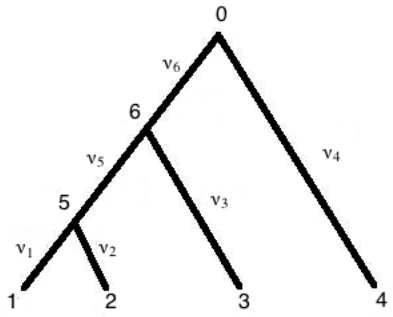
Table 1.1: Example nucleotide data on 5 taxa

Human	A	A	G	C	T	T	C	A	C	C	G	G	C	G	C	A	G	T	C	A	T	T	C	T	C	A	T	A	A	T	C	G
Chimpanzee	A	A	G	C	T	T	C	A	C	C	G	G	C	G	C	A	A	T	T	A	T	C	C	T	C	A	T	A	A	T	C	G
Gorilla	A	A	G	C	T	T	C	A	C	C	G	G	C	G	C	A	G	T	T	G	T	T	C	T	T	A	T	A	A	T	T	G
Orangutan	A	A	G	C	T	T	C	A	C	C	G	G	C	G	C	A	A	C	C	A	C	C	C	T	C	A	T	G	A	T	T	G

Consider the phylogenetic tree in figure 1.1. This tree consists of four terminal nodes that represent modern DNA sequences and three internal intersections/nodes. For terminology

purposes, the generic term "node" refers to any of the points $g = 0 - 6$ in the figure. "Internal nodes" (also called "ancestral nodes") are represented by $g = 5$ and 6. The "root node" is represented at $g = 0$, and the "tips" (or "leaves") of the tree at nodes $g = 1 - 4$ are the observed data for the taxa as represented in table 1.1. Note that the relationship of the rows in the data of table 1.1 could be represented with figure 1.1. This relationship between the data for each taxa and tree is represented such that the human (row one) is genetically closer to the Chimpanzee (row two) than the Orangutan (row four).

Figure 1.1: Example evolutionary tree



The values $\nu_1, \nu_2, ...\nu_6$ represent the amount of sequence change (or "divergence"). These quantities include the effects of both time and rate of sequence evolution. The details of these quantities will be detailed later.

While the observed genetic data may only be available for the taxa at nodes one through four, if the data for the complete phylogenetic tree were known it would consist of an alignment with seven rows. Note that research into HIV and some viruses with high substitution rates may use phylogenetic trees with data available at all nodes.

1.2.2 Nucleotide Substitution Models

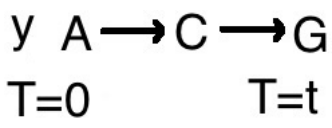
In order to carry out statistical inferences, it is necessary to describe state changes for sequences over time with Markov processes. Assumptions for the simplest models for DNA sequence evolution are the following:

1. substitutions are realized at an instantaneous point in evolutionary time
2. sites in the genetic sequence are independent
3. substitution rates are homogeneous across the sites in the sequence
4. the substitutions follow a reversible Markov process
5. the phylogenetic tree is known and the same for all sites in the alignment.

The assumption of instantaneous nucleotide substitutions (1) relates to the point in time that a site change has been adopted by the entire population or taxon under consideration. Note that this does not imply evolutionary changes occur instantaneously for the population.

The assumption on the Markov process (4) is illustrated in figure 1.2. The reversibility of this is assumption is defined as $\pi_{y_1} P(y_1 \to y_2; t) = \pi_{y_2} P(y_2 \to y_1; t)$, where the stationary distribution of the Markov process for the four nucleotide states is π_y for $y \in \{A, C, G, T\}$.

Figure 1.2: Markov Process



Each of these assumptions may be relaxed with particular cases discussed later in this dissertation.

The General Time Reversible Model

The generalized nucleotide substitution model is known as the general time reversible (GTR) model Tavare [1986]. For this model, the stationary distribution is not assumed equivalent and each possible reversible nucleotide substitution is modeled with a separate parameter (i.e., no substitutions are assumed equivalent). Therefore, the generalized substitution model, GTR, is represented with the following instantaneous rate matrix:

$$R = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} * & \theta_0\pi_C & \theta_1\pi_G & \theta_2\pi_T \\ \theta_0\pi_A & * & \theta_3\pi_G & \theta_4\pi_T \\ \theta_1\pi_A & \theta_3\pi_C & * & \theta_5\pi_T \\ \theta_2\pi_A & \theta_4\pi_C & \theta_5\pi_G & * \end{pmatrix} \end{array}$$

For this rate matrix, the π_i; $i \in \{A, C, G, T\}$ are the frequencies of the stationary distribution and the θ_i; $i \in \{0, 1, ..., 5\}$ are the corresponding rates of change between the nucleotides. For example, the substitution rate for a change of $A \leftrightarrow C$ is modeled with θ_0, the change from $A \leftrightarrow G$ is modeled with θ_1, etc.

Symmetry of the substitution matrix is a byproduct of reversibility of the Markov process, and the matrix diagonal $(*)$ is defined as $R_{ii} = -\sum_{\{j:j\neq i\}} R_{ij}$; $i, j \in \{1, 2, 3, 4\}$ such that the row sums of R are zero. This approach of representation for nucleotide models can be used to identify the full set of 203 distinct, reversible models [Crandall, 1999].

For finite, non-infinitesimal lengths of time, the matrix of transition probabilities is

$$P(t) = e^{tR}. \tag{1.1}$$

Equation 1.1 has been extensively studied and calculated with various methods that are discussed later [Moler and Van Loan, 1978, Muse, 1996, Moler and Van Loan, 2003].

Single Parameter Nucleotide Models

The simplest nucleotide model is JC69. This model uses a single parameter to describe the relative frequency of any nucleotide change and assumes that the stationary distribution of nucleotides is equivalent across levels ($\pi_A = \pi_C = \pi_G = \pi_T = \frac{1}{4}$) [Jukes and Cantor, 1969]. The 203 reversible models considered in this dissertation do not include this assumption on the stationary distribution. Therefore models assuming an equivalent stationary will not be discussed further.

Felsenstein described a similar model with the same assumption on the rate of nucleotide substitutions (shared, single parameter) but did not assume equal weights for the ancestor distribution of nucleotides [Felsenstein, 1981]. Therefore, the substitution matrix is defined as

$$R^{\text{F81}} = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} * & \theta_0\pi_C & \theta_0\pi_G & \theta_0\pi_T \\ \theta_0\pi_A & * & \theta_0\pi_G & \theta_0\pi_T \\ \theta_0\pi_A & \theta_0\pi_C & * & \theta_0\pi_T \\ \theta_0\pi_A & \theta_0\pi_C & \theta_0\pi_G & * \end{pmatrix} \end{array}$$

For the transition probabilities, note that these may be numerically solved using equation 1.1. However, Felsenstein solved for these from first principles.

$$P_{ij}^k(t) = \pi_i + (1 - \pi_i)e^{-t\theta_0},\ i \neq j$$
$$P_{ij}^k(t) = \pi_i - \pi_i e^{-t\theta_0},\ i = j \tag{1.2}$$

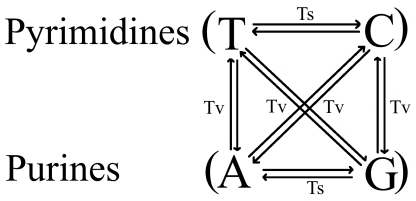
Note that since the stationary distribution is not assumed as equal across states, $\pi_0 \in \{\pi_A, \pi_C, \pi_G, \pi_T\}$ and is represented by the ancestor nucleotide state for the given substitution.

Multi-Parameter Nucleotide Models

In general, substitution models are differentiated by how they parameterize the substitution rates into fixed categories. For example, F81 categorizes all substitution rates together and GTR lets each substitution rate be categorized separately. Let this be described as "clustering", where the different substitution rates are set equivalent within each of the possible classes (or "clusters"). This method of defining substitution models is useful for comparing model averaged estimates across candidate subsets from the census. Further details will be discussed in Section 1.5.

Per figure 1.3, define purines as nucleotide states of Adenine (A) and Guanine (G) and pyrimidines as Cytosine (C) and Thymine (T). Transversions (Tv) are defined as nucleotide state changes resulting in a substitution from purine to pyrimidine or visa versa. Transitions (Ts) are those nucleotide site changes that do not result in a substitution of a purine to pyrimidine or visa versa. Therefore, $A \leftrightarrow G$ and $C \leftrightarrow T$ are transitions. All other substitutions fall into the category of transversions. Note that under the assumption of reversibility (assumption 4), each state change is considered equivalent in both directions (i.e., $A \rightarrow C = C \rightarrow A$).

Figure 1.3: Purine and Pyrimidine Substitutions



Hasegawa et al. [1985] defined a two parameter model (HKY85) that is based on the distinction between transitions and transversions. This model clusters nucleotide state changes into two parameters: one for transition substitution rates, θ_1, and one for transversion substitutions,

θ_0. The substitution matrix is therefore defined as

$$R^{\text{HKY85}} = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} * & \theta_0\pi_C & \theta_1\pi_G & \theta_0\pi_T \\ \theta_0\pi_A & * & \theta_0\pi_G & \theta_1\pi_T \\ \theta_1\pi_A & \theta_0\pi_C & * & \theta_0\pi_T \\ \theta_0\pi_A & \theta_1\pi_C & \theta_0\pi_G & * \end{pmatrix} \end{array}$$

Tamura and Nei [1993] specified a similar model to HKY85 with the exception that it clustered the two different transitions separately (TN93). Specifically, transversion nucleotide substitutions are modeled with a single rate, say θ_0, while transitions are modeled with two different parameters: $A \leftrightarrow G$ using θ_1 and $C \leftrightarrow T$ using θ_2.

Nucleotide Model Discussion

The key concept regarding the development of substitution models is the importance of biological characteristics. Note that each of the multi-parameter, named models estimates the transition and transversion rates separately (i.e., these models do not cluster a transition substitution parameter with a transversion substitution parameter). The proof of concept analyses in Chapter 2 will provide a detailed description of the computational relevance for these biologically significant characteristics. This information will be used with the model averaging results to obtain a methodology for model averaging in Chapter 3.

1.2.3 Codon Substitution Models

Analyses over all 203 reversible nucleotide substitution models is feasible from a computational standpoint. This is not currently possible for the codon case. There are two reasons for this: the enumeration of possible models is large and the computational resources required for the likelihood calculations are significantly greater for the codon case (both discussed further later).

Due to these challenges, codon modeling methods rely on deductive reasoning to analyze a narrow subset or a single point in the model space. Analogous to the nucleotide case, this deductive reasoning is based on biological characteristics.

For the assumptions in Section 1.2.3, note that if codons are considered sites, independence holds per assumption 2. However, for the nucleotides within a codon, independence does not hold. Additionally, of the $4^3 = 64$ possible codons, three are stop codons used to signal the termination of translation for a genetic sequence. Therefore, an additional assumption is used in this dissertation that does not consider the stop codons for substitution rate modeling and results in the reduction of considered codons to 61.

Empirical Codon Substitution Models

Empirical amino acid substitution models have a relatively robust history with Dayhoff and Eck [1968] and Dayhoff et al. [1978] providing some of the earliest methodologies. An important distinction for these models is that they estimate large numbers of parameters requiring a significant amount of data [Kosiol et al., 2007].

Schneider et al. [2005] published the first empirical model for codons; it is based on more than 17,000 alignments of five vertebrate genomes with 8.3 million codons. Additionally, the empirical substitution matrix does not use assumption 4 from Section 1.2.3 (reversibility of the Markov Process). Therefore, the model estimates all 3660 off-diagonal elements of the 61×61 substitution matrix. The method provides a log-odds substitution matrix based on simple frequencies using the sequence data. Schneider et al. [2005] use sequences with relatively small divergence; this is done purposely because a constraint to the method as alignments with long sequence divergences are better estimated with amino acid matrices.

Even with more than 17 thousand alignments and 8 million codons, the most rare substitution (TGG to GAG) had a count of only 45 and was nearly 34 thousand times less frequent than the most frequent substitutions observed in the data. Therefore, even with the large database used for this empirical estimation, the model has a relatively high risk of producing estimates

that may be unstable or unestimable with anything less than an enormous amount of data [Schneider et al., 2005].

There are fundamental differences and restrictive uses of empirical models. Therefore, we will not explore empirical models further in this dissertation. Moving forward, Markov process models will be the only models considered.

MG94 and GY94 Models

The earliest named codon models, not based on empirical estimation, were published simultaneously by Muse and Gaut [1994] (MG94) and Goldman and Yang [1994] (GY94). These models cluster all non-stop, single-nucleotide change codon substitutions into one of two biological possibilities (synonymous versus nonsynonymous substitutions). This is analogous to the partitioning of transitions and transversions in the HKY85 model.

The two models are represented by the following instantaneous rates:

$$R_{ij}^{\text{MG94}} = \begin{cases} \alpha\pi_{y_j} & \text{single change from codon } i \text{ to } j \text{ is synonymous} \\ \beta\pi_{y_j} & \text{single change from codon } i \text{ to } j \text{ is nonsynonymous} \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

$$R_{ij}^{\text{GY94}} = \begin{cases} \alpha\mathcal{F}_j & \text{single from change codon } i \text{ to } j \text{ is synonymous} \\ \beta\mathcal{F}_j & \text{single from change codon } i \text{ to } j \text{ is nonsynonymous} \\ 0 & \text{otherwise.} \end{cases} \tag{1.4}$$

Here π_{y_j} represents the stationary probability of the "target" nucleotide whose change resulted in codon substitution (i to j) in the GY94 model, and $\mathcal{F}_k : k = 1, 2, ..., 61$ are the codon frequencies. Although Goldman and Yang [1994] did not specify a particular method of estimating $\mathcal{F}_k$, modern methods typically consist of

$$\frac{\hat{\pi}_y^1 \hat{\pi}_y^2 \hat{\pi}_y^3}{1 - \hat{\Pi}_{stop}}. \tag{1.5}$$

where $\hat{\pi}_y^{1-3}$ are the estimated stationary nucleotide probabilities at positions 1, 2, and 3 of the codon and $\hat{\Pi}_{stop} = [\hat{\pi}_T \times \hat{\pi}_A \times \hat{\pi}_G] + [\hat{\pi}_T \times \hat{\pi}_G \times \hat{\pi}_A] + [\hat{\pi}_T \times \hat{\pi}_A \times \hat{\pi}_A]$. The two models, published simultaneously in 1994, are similar in their use of the same underlying assumptions and biological differences between nonsynonymous and synonymous substitutions. However, the primary difference between MG94 and GY94 is that the former scales substitution rates using the "target" stationary probabilities while the latter scales the rates using the codon frequencies.

A common modification to MG94 and GY94 is the inclusion of an additional parameter for the transition and transversion substitution rates for the single-nucleotide changes [Goldman and Yang, 1994, Yang and Nielsen, 2002, Huelsenbeck et al., 2006, Rodrigue et al., 2007]. In this case, it is referenced as κ and results in the following substitution matrices:

$$R_{ij}^{\text{MG94}\kappa} = \begin{cases} \alpha\kappa\pi_{y_j} & \text{single from change codon } i \text{ to } j \text{ is synonymous transition} \\ \alpha\pi_{y_j} & \text{single change from codon } i \text{ to } j \text{ is synonymous transversion} \\ \beta\kappa\pi_{y_j} & \text{single change from codon } i \text{ to } j \text{ is nonsynon ymous transition} \\ \beta\pi_{y_j} & \text{single change from codon } i \text{ to } j \text{ is nonsynonymous transversion} \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

$$R_{ij}^{\text{GY94}\kappa} = \begin{cases} \alpha\kappa\mathcal{F}_j & \text{single change from codon } i \text{ to } j \text{ is synonymous transition} \\ \alpha\mathcal{F}_j & \text{single change from codon } i \text{ to } j \text{ is synonymous transversion} \\ \beta\kappa\mathcal{F}_j & \text{single change from codon } i \text{ to } j \text{ is nonsynonymous transition} \\ \beta\mathcal{F}_j & \text{single change from codon } i \text{ to } j \text{ is nonsynonymous transversion} \\ 0 & \text{otherwise.} \end{cases} \tag{1.7}$$

Codon Model Discussion

Rodrigue et al. [2008] compared the MG94 and GY94 models along with numerous changes that included differences in estimating the stationary distribution, heterogeneity of the rates across

codons, use of a Dirichlet process prior, codon and amino acid preference weights, and the use of the transition-to-transversion ratio parameter (κ). The MG94 parameterization generally outperformed the GY94 version across the different changes to the models. For this reason, this dissertation scales the substitution rates with the "target" nucleotides (MG94 method) in the simulation of data and estimation of the substitution matrix.

The development of codon substitution models is similar to the nucleotide case where both focus on the biological characteristics of interest. In this case, the nonsynonymous and synonymous substitution rates are used directly or with the ratio in order to define positive selective pressures. Also similar to the nucleotide case, these *a priori* models do not account for variability across different models and do not account for differences between the model and the true process that originated the observed data.

Delport et al. [2010b] published a comprehensive framework for using GAs to traverse a subset of the class of reversible substitution models. This was used with a forward selection scheme to select both the number and parameterization of nonsynonymous substitution rate clusters. This GA framework is the basis of the selection of candidate sets of models for use with model averaging in this dissertation. The objective of this framework and implementation is to illustrate a robust method, relative to pre-selected models, for estimating substitution rates.

1.2.4 Model Space

The substitution models considered in this dissertation estimate clusters of rate parameters from among the discrete possibilities. The possible ways to cluster a set of N numbers into n groups or clusters is done using a Stirling number of the second kind [Miller et al., 1999, Kosakovsky Pond et al., 2007]:

$$S(N,n) = \frac{1}{n!}\sum_{x=1}^{n}(-1)^x \binom{n}{x}(n-x)^N.$$

Therefore, the possible number of models is calculated by the sum of the Stirling numbers over each of the possible cluster sizes. As previously stated, in the case of nucleotide substitution models there are relatively few and enumerated by the sum of each of the possible clustering sizes, $\sum_1^6 S(6,n) = 203$ models.

n the case of codon substitution models this set of possibilities is combinatorically large. Consider that there are 4032 off-diagonal elements of a 64×64 substitution matrix. This results in a massive $\sum_1^{4032} S(4032,n) = 1.35 \times 10^{9796}$ possible clusterings. The assumption removing stop codons reduces the matrix to 61×61 with 3660 off-diagonal elements and a model space of $\sum_1^{3660} S(3660,n) = 2.68 \times 10^{8762}$ models. Additionally, the assumption of a reversible Markov process results in a symmetric substitution matrix with $\frac{(61+1)61}{2} - 61 = 1830$ upper-diagonal elements and a reduction of the model space to $\sum_1^{1891} S(1891,n) = 8.26 \times 10^{4073}$ models. For this dissertation, we also use the assumption from MG94 and GY94 that codon substitutions resulting in more than one nucleotide state change occur with rate zero. This assumption is included for computational convenience as it reduces the possible non-zero, upper-diagonal elements of the substitution matrix to 263. Enumeration of the model space, as it applies to this dissertation, is therefore $\sum_1^{263} S(263,n) = 2.76 \times 10^{386}$ models. Therefore, despite the final model space being more than 25 orders of magnitude smaller than the full model space, it still results in a value that is larger than the number of elementary particles in the known universe (approximately 10^{79} [Champion, 1998]) by more than four.

Note that computational problems exist even with the use of very restrictive limitations. For example, if only two possible parameter clusterings are permitted among the 263 possibilities, the number of possible models is $\sum_1^2 S(263,n) = 7.41 \times 10^{78}$. MG94 is just one of the 7.41×10^{78} possible ways. Related to this problem, the next section (1.3) will discuss implementation of the likelihood and maximization for each of these possible codon models. Essentially, the issue is that there are no non-trivial subsets with fewer models than 7×10^{78}, and the next section will illustrate the computational intensity required for each of these models.

The scale of the codon model space is computationally problematic. No matter how re-

strictive the model limitations may be for clustering size, the resulting number of models is computationally problematic for a census. Therefore, the emphasis of this dissertation is to explore the model space efficiently and implement model averaging on that set of models.

1.3 Likelihood

Efforts to obtain generalized maximum likelihood estimates led to the definition of the phylogenetic likelihood function for sequence data. Among the earliest efforts were those of Neyman [1971] and Holmquist [1972] with the Neyman method being restricted to three taxa. Prior to 1981, multiple other methods were developed that defined the likelihood function on sequence data, however, these methods did not generalize the function, were ad-hoc adjustments to existing non-generalized methods, or were computationally infeasible [Felsenstein, 1973, Kashyap and Subas, 1974, Farris et al., 1979, Kaplan and Langley, 1979].

The first likelihood and corresponding maximization method that was both generalized and computationally feasible was developed by Felsenstein [1981]. The following is a brief description of his likelihood definition in order to provide context for detailing his method of iterative maximum likelihood.

Using figure 1.1, consider the following notation of a probability on a site change between two nodes (say nodes 1 and 5)

$$P(Y_1^k = y_1^k, \nu_1 | Y_5^k = y_5^k). \tag{1.8}$$

Note that ν is the product of substitution rate and time. Per equation 1.1, this represents the confounded, multiplicative values that are exponentiated. For notational convenience, further references to these probabilities will simply be $P_{51}^k(\nu_1)$.

Using assumptions 2 and 3 (all sites are independent and and substitution rates are homogeneous across the tree), it follows from equation (1.8) that the likelihood for the entire tree

(across all nodes of the alignment) at the k^{th} site is

$$L(y^k) = \pi_{y_0} P^k_{06}(\nu_6) P^k_{65}(\nu_5) P^k_{63}(\nu_3) P^k_{51}(\nu_1) P^k_{52}(\nu_2) P^k_{04}(\nu_4). \tag{1.9}$$

where π_{y_0} is the stationary probability. However, because the data at nodes 0, 5, and 6 are typically unknown, the computational burden calculating the likelihood increases significantly with the following:

$$L(y^k) = \sum_{y^k_6} \sum_{y^k_0} \sum_{y^k_5} \pi_{y_0} P^k_{06}(\nu_6) P^k_{65}(\nu_5) P^k_{63}(\nu_3) P^k_{51}(\nu_1) P^k_{52}(\nu_2) P^k_{04}(\nu_4). \tag{1.10}$$

Note that a key finding from Felsenstein [1981] is that the likelihood simplifies further due to probability terms being free of particular summands. Therefore, by gathering terms in (1.10) the likelihood reduces to

$$L(y^k) = \sum_{y^k_0} \pi_{y_0} \left\{ \left[P^k_{04}(\nu_4) \right] \sum_{y^k_6} \left[P^k_{06}(\nu_6) P^k_{63}(\nu_3) P^k_{65}(\nu_5) \right] \sum_{y^k_5} \left[P^k_{51}(\nu_1) P^k_{52}(\nu_2) \right] \right\}. \tag{1.11}$$

Based on the assumption of independence (Section 1.2, assumption 2), the complete phylogenetic likelihood is the product of N site likelihoods

$$L(y) = \prod_{k=1}^{N} L(y^k). \tag{1.12}$$

The data analyses of this dissertation do not include likelihoods with an unknown phylogeny. Therefore, the above likelihood calculations are considered conditional upon the relationship between the rows of the alignment (evolutionary tree). Therefore, this dissertation will assume the evolutionary tree is fixed and known for all cases.

1.3.1 Maximizing Likelihood

Modern methods of estimating substitution rates across the phylogenetic tree are based on the principles of maximum likelihood (ML).

Felsenstein's original maximization of the likelihood function used a computational tool that reduces the number of nodes to maximize the likelihood over by one. This method essentially unroots the likelihood of the evolutionary tree from the root ancestor node. By using the Chapman-Kolmogorov equation, together with the reversibility of the Markov process, Felsenstein proved that the class of all unrooted versions of a rooted tree have equivalent likelihoods to the ancestrally rooted version. He termed this property "The Pulley Principle" and applied it as a method of unrooting evolutionary trees at any computationally convenient location [Felsenstein, 1981].

As an example, consider figure 1.5 and note that branches ν_6 and ν_4 have been unrooted from the rooted version in figure 1.4. Furthermore, note that the likelihoods for these two trees under Felsenstein's method are equivalent.

Figure 1.4: Rooted evolutionary tree

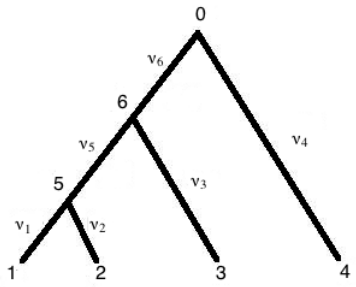
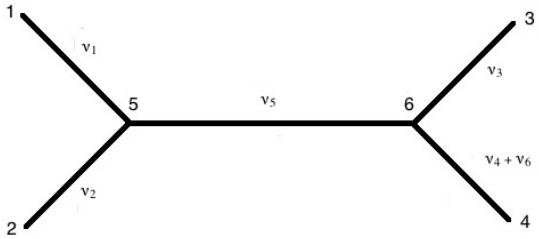


Figure 1.5: Unrooted evolutionary tree



The Pulley Principle also implies that the likelihood in equation (1.11) depends on ν_6 and ν_4 only through their sum. Using this with some clever grouping of likelihood terms (in this example, for node 6), Felsenstein showed the following holds true:

$$L(y^k) = \pi_{y_6} L(y_6^k) \sum_{y_0^k} P(y_4^k, \nu_4 | y_0^k) P(y_6^k, \nu_6 | y_0^k) = \pi_{y_6} L(y_6^k) P(y_6^k, \nu_4 + \nu_6 | y_4^k). \tag{1.13}$$

Felsenstein noted that the practical assumption of the "Pulley Principle" is that the true root is immediately contiguous to a particular node joining the combined branches of the likelihood. He then used this tool to implement an iterative method of maximizing the likelihood of a given unrooted tree one branch at a time. Therefore, the Felsenstein [1981] method of maximizing the phylogenetic likelihood follows from the stepwise traverse of the phylogenetic tree, maximizing each corresponding conditional likelihood. Note that this method does not guarantee a global maximum.

1.4 Estimation

1.4.1 Transition/Transversion Rates

Many of the early models of nucleotide substitutions were based on the biological characteristics of transition and transversion differences [Kimura, 1980, Hasegawa et al., 1985, Tamura and Nei, 1993]. Estimation of the transition-to-transversion ratio (κ) is based on the expected number of transitions and transversions. In order to describe this ratio, note that the expected total substitutions is the summand of off-diagonal elements from the substitution matrix and scaled

by the stationary frequencies and time:

$$\begin{aligned} E(\text{To}) &= t\sum_{i=1}^{4}\pi_{y_i}\sum_{y_j:y_j\neq y_i}R_{y_iy_j} \\ &= t\times\big[\pi_A\theta_0\pi_C+\pi_A\theta_1\pi_G+\pi_A\theta_2\pi_T+\pi_C\theta_3\pi_G+\pi_C\theta_4\pi_T+\pi_G\theta_5\pi_T \\ &+\pi_C\theta_0\pi_A+\pi_G\theta_1\pi_A+\pi_T\theta_2\pi_A+\pi_G\theta_3\pi_C+\pi_T\theta_4\pi_C+\pi_T\theta_5\pi_G\big] \\ &= t\times 2\times\big[\theta_0\pi_A\pi_C+\theta_1\pi_A\pi_G+\theta_2\pi_A\pi_T+\theta_3\pi_C\pi_G+\theta_4\pi_C\pi_T+\theta_5\pi_G\pi_T\big] \end{aligned} \tag{1.14}$$

As all nucleotide substitutions are either transitions or transversions, the sum of the expected transitions and expected transversions is the expected total substitutions.

$$E(\text{To}) = t\sum_{i=1}^{4}\pi_{y_i}\sum_{y_j:y_j\neq y_i}R_{y_iy_j} = E(\text{Ts})+E(\text{Tv})$$

$$\begin{aligned} \text{where } E(\text{Ts}) &= t\times\big[\pi_AR_{AG}+\pi_CR_{CT}+\pi_GR_{GA}+\pi_TR_{TC}\big] \\ &= t\times\big[\pi_A\theta_1\pi_G+\pi_C\theta_2\pi_T+\pi_G\theta_1\pi_A+\pi_T\theta_4\pi_C\big] \\ &= t\times 2\times\big[\theta_1(\pi_A\pi_G)+\theta_4(\pi_C\pi_T)\big]. \end{aligned} \tag{1.15}$$

$$\begin{aligned} \text{and } E(\text{Tv}) &= t\times\big[\pi_AR_{AC}+\pi_AR_{AT}+\pi_CR_{CG}+\pi_GR_{GT}+\pi_CR_{CA}+\pi_TR_{TA} \\ &+\pi_GR_{GC}+\pi_TR_{TG}\big] \\ &= t\times\big[\pi_A\theta_0\pi_C+\pi_A\theta_2\pi_T+\pi_C\theta_3\pi_G+\pi_G\theta_5\pi_T+\pi_C\theta_0\pi_A+\pi_T\theta_2\pi_A \\ &+\pi_G\theta_3\pi_C+\pi_T\theta_5\pi_G\big] \\ &= t\times 2\times\big[\theta_0(\pi_C\pi_A)+\theta_2(\pi_T\pi_A)+\theta_3(\pi_G\pi_C)+\theta_5(\pi_T\pi_G)\big] \end{aligned} \tag{1.16}$$

As implied by (1.15) and (1.16), $E(\text{Ts})$ and $E(\text{Tv})$ are the values of $t\pi_iR_{ij}$ that correspond to transition and transversion substitutions, respectively.

For example, given the HKY85 model, we have the following:

$$R^{\text{HKY85}} = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} * & \theta_0\pi_C & \theta_1\pi_G & \theta_0\pi_T \\ \theta_0\pi_A & * & \theta_0\pi_G & \theta_1\pi_T \\ \theta_1\pi_A & \theta_0\pi_C & * & \theta_0\pi_T \\ \theta_0\pi_A & \theta_1\pi_C & \theta_0\pi_G & * \end{pmatrix} \end{array} \text{ where } \begin{cases} E(\text{Ts}) = & t2\theta_1 \times \big[(\pi_A\pi_G) + (\pi_C\pi_T)\big] \\ E(\text{Tv}) = & t2\theta_0 \times \big[(\pi_A\pi_C) + (\pi_A\pi_T) + \\ & (\pi_C\pi_G) + (\pi_G\pi_T)\big]. \end{cases}$$

Finally, the ratio of transitions to transversion substitutions is $\kappa = 2E(\text{Ts})/E(\text{Tv})$, and by the property of invariance, $\hat{\kappa} = 2\hat{E}(\text{Ts})/\hat{E}(\text{Tv})$. The scalar multiplication for κ is added to equally weight the numerator and denominator, such that if θ_{0-5} are equivalent, $\kappa = 1$. Note that $E(\text{Ts})$ and $E(\text{Tv})$ are estimated by the property of invariance through the use of MLEs $\hat{\theta}_{0-5}$. Therefore 1.15 and 1.16 are estimated by replacing the component parameters with their corresponding MLEs: $2\big[\hat{t\theta}_1(\hat{\pi}_A\hat{\pi}_G) + \hat{t\theta}_4(\hat{\pi}_C\hat{\pi}_T)\big]$ and $2\big[\hat{t\theta}_0(\hat{\pi}_C\hat{\pi}_A) + \hat{t\theta}_2(\hat{\pi}_T\hat{\pi}_A) + \hat{t\theta}_3(\hat{\pi}_G\hat{\pi}_C) + \hat{t\theta}_5(\hat{\pi}_T\hat{\pi}_G)\big]$.

This generalized method of estimating κ has the favorable property of being defined for every member across the complete set of 203 substitution models. This property facilitates the use of model averaged estimates across any set of models and will be investigated fully in Chapter 2.

1.4.2 Nonsynonymous/Synonymous Rates

In the codon case, estimation of the nonsynonymous-to-synonymous ratio (ω) is not as straightforward as the transitions-to-transversion ratio from the nucleotide case due to the definition of nucleotide site changes resulting in an encoding change of the amino acid (discussed within this section) [Muse, 1996]. Several methods have been presented to estimate this parameter. Most of these methods pre-date the advent of codon models that are ad-hoc, and based on observable quantities and the counts of paired sequence data.

Muse [1996] points out that these methods typically express ω in units of silent (synonymous) substitutions per silent site or replacement (nonsynonymous) substitutions per replacement site. Define an f-fold degenerate site ($f = 1, 2, 3, \text{or } 4$) as the number of nucleotides at a particular site in a codon that result in no encoding change in the amino acid (i.e., nonsynonymous change). For example, the third position of many codons does not result in any change of the encoded amino acid and would therefore be considered four-fold degenerate sites. Next, consider that perfectly silent or replacement sites do not exist. For example, if a site is four-fold degenerate, there still exists the possibility that changes to other sites in the codon may change the degeneracy. Therefore, the degeneracy of a site is stochastic.

Muse [1996] provides a framework for estimating ω based on defining what are the rates of synonymous and nonsynonymous substitutions. The first approach is based on the ratio of expected silent to expected replacement substitutions. The expected number of silent substitutions is defined as $E_s = \sum_{c=1}^{61} \mathcal{F}_c s_c/3$ where s_c is the number of possible single nucleotide synonymous changes for codon c, and $\mathcal{F}_c$ is codon c's frequency. The expected number of replacement substitutions (E_n) follows by substituting into the equation n_c as the number of possible single nucleotide nonsynonymous changes for codon c. Therefore the synonymous and nonsynonymous substitution rates are

$$\begin{aligned}
\mathcal{D}_s &= \frac{t \sum_c \mathcal{F}_{c_s} \sum_{c_s \neq z} S_{c_s z}}{\sum_{c_s} \mathcal{F}_c s_c/3} \\
&= \frac{\alpha t \sum_i \mathcal{F}_i \sum_k s_i \pi_{y_k}}{E_s} \\
&= \alpha t \gamma_s
\end{aligned} \tag{1.17}$$

$$\begin{aligned}
\mathcal{D}_n &= \frac{t \sum_i \mathcal{F}_i \sum_{j \neq i} O_{ij}}{\sum_i \mathcal{F}_i n_i/3} \\
&= \frac{\beta t \sum_i \mathcal{F}_i \sum_k n_i \pi_{y_k}}{E_n} \\
&= \beta t \gamma_n.
\end{aligned} \tag{1.18}$$

S_{ij} is the ij^{th} entry in the i^{th} row and j^{th} column of R that corresponds to a single nucleotide, synonymous codon change and O_{ij} is the nonsynonymous analog. If we assume the MG94 model, $\sum_{j \neq i} S_{ij}$ reduces to $\alpha \sum_k s_i \pi_{n_{ik}}$ and $\sum_{j \neq i} O_{ij}$ reduces to $\beta \sum_k n_i \pi_{n_{ik}}$. Additionally, since γ_n is not necessarily equal to γ_s, for $\alpha = \beta$ the ratio of D_n/D_s is not necessarily equal to one and therefore presents challenges as a parameter for ω [Muse, 1996].

The second method of estimating the synonymous and nonsynonymous substitution rates is based on the definition of parameters for relative frequency of change as they relate to theoretically perfectly silent sites versus replacement sites (Δ_s and Δ_n, respectively). These are analogous to the nucleotide parameters used for κ.

$$\begin{aligned} \Delta_s &= 1 - t \sum_{k \neq i} S_{ik} \pi_{y_k} \\ &= 1 - t\alpha \sum_{k \neq i} \pi_{y_k}^2 \end{aligned} \tag{1.19}$$

$$\begin{aligned} \Delta_n &= 1 - t \sum_{k \neq i} O_{ik} \pi_{y_k} \\ &= 1 - t\beta \sum_{k \neq i} \pi_{y_k}^2 \end{aligned} \tag{1.20}$$

The quantities in (1.17) and (1.18) do not have the same issues as ω, (1.19) and (1.20). Specifically, if $\alpha = \beta$ then (1.17) is not necessarily equal to (1.18), however (1.19) would be equivalent to (1.20). Due to their historic significance and interpretability, most investigators may be more familiar with the principles of the parameters of (1.17) and (1.18). However (1.19) and (1.20) have more favorable theoretical properties. Therefore, care should be taken in the use of either estimator and the audience it is intended for.

1.4.3 Implementation

The HyPhy package (HyPhy) is a likelihood-based analysis platform for phylogenetic sequence data [Kosakovsky Pond et al., 2005]. All estimation was conducted using the HyPhy package. All constituent and composite parameters in the previous two Sections 1.4.1 and 1.4.2 were estimated via ML and and the composite parameters due to the invariance property. See Appendix C for details.

1.5 Model Selection

A "good" substitution model will efficiently describe the underlying forces that led to the evolution of the genetic sequence for a phylogenetic tree. The evaluation of whether a model is considered "good" is a function of the likelihood that the parameterization led to the observed data and the dimensionality of the parameterization.

General challenges of model selection include the issue of overfitting versus underfitting the observed data. The effort to obtain a precise representation of the true process that results in the data must be balanced with the dimensionality of the chosen model. Too few parameters to describe the underlying process generally result in biased estimates while too many parameters generally result in smaller bias but higher variance (discussed further in Chapters 2 and 3). Model averaging has been advocated as a means of controlling the bias and variability of the model selection process [Kullback and Leibler, 1951, Levins, 1966, Anderson et al., 2001, Burnham and Anderson, 2010].

Many statistical and scientific methods have been developed to deal with these issues. Generally, the approach of taking a census of all possible models for the selection process is either not practical or possible. Anderson et al. [2001] and Burnham and Anderson [2010] advocate the use of candidate models, drawn from a subset of those possible, that is based on relevant scientific hypotheses.

Given candidate sets of models, the selection process evaluates the "fit" of each. Here,

the "fit" of a model is the characteristic that describes how well the model approximates the process that resulted in what has been observed. Therefore, the manner and strategy used in the assessments of "fit" is the process of model selection. This section will detail several methods of model selection that are common in phylogenetics.

1.5.1 Hierarchical Likelihood Ratio Tests

A well-known, and early phylogenetic model selection method is the hierarchical likelihood ratio test (hLRT). The most popular implementation of this method in molecular evolution is MODELTEST [Posada and Crandall, 1998]. The basic paradigm for hLRT testing begins with two models (either the most complicated or simplest) and builds a top-down decision tree with the bifurcating results based on each pairwise likelihood ratio test (LRT). This process continues like a decision tree until it results with a particular model. The LRT at each stage in the tree (or hierarchy) is performed using the test statistic $LRT = 2[log(L_1) - log(L_2)]$. P-values are then derived based on $LRT \sim \chi^2_{p_1-p_2}$, where p_1 and p_2 are the number of free parameters in the more complicated and less complicated models, respectively. In the nucleotide case, the method often begins with a comparison of the simplest model that analyzes the difference between equivalent stationary frequencies versus otherwise. This translates to a LRT on JC69 and F81. If the winner is F81, the method will hold the assumption on the stationary frequencies per F81 and test against the next more complicated model (HKY85). However, if JC69 is the winner, the assumption on equivalent stationary frequencies will be held and the next test will compare K80. This will continue until the following step does not result in an improvement in model fit [Posada and Buckley, 2004].

While such methods have become widely used, they suffer from a variety of problems. One issue is that the multiplicity error rate compounds as the hLRT moves through the decision tree. By the time a model is selected at the end of the tree, the overall hypothesis test is no longer size α. While there are many methods of correcting for multiplicity error rates (i.e., Holm-Bonferroni, Scheffe, etc.), the hLRT procedure uses dependent tests that violate assumptions

for many of the correction methods and therefore complicates their use. Additionally, Posada and Buckley [2004] show that the starting model used for the tree can affect the final chosen model, and Sanderson and Kim [2000] point out that hLRTs can altogether miss the optimal model. For example, it is possible that the unspecified model of Tavare [1986] could lose to TN93 by the LRT test statistic, and then TN93 loose to HKY85 while the unspecified model would be able to beat HKY85 if tested together.

While it is possible that the nucleotide model space can be reasonably traversed with the careful application of hLRTs, the ability to traverse any reasonable subset of the codon model space is impractical due its size. Finally, Posada himself discourages use of hLRTs, and by extension, his own MODELTEST procedure in favor of more robust methods [Posada and Buckley, 2004].

1.5.2 Bayesian Markov Chain Monte Carlo

Multiple methods of parameter estimation and hypothesis testing with Bayesian methods have been developed for phylogenetic inferences. Various software packages such as MrBayes and BEAST (Bayesian Evolutionary Analysis by Sampling Trees) have been developed to implement certain MCMC algorithms. These estimate the posterior distribution of the substitution parameters given the observed sequence data [Suchard et al., 2001, Huelsenbeck and Ronquist, 2001, Drummond and Rambaut, 2007, Ronquiet et al., 2012]. Different packages have various focuses; for example, BEAST focuses on estimation of calibrated phylogenies in order to apply a time-scale to the statistical model of estimation [Drummond and Rambaut, 2007].

However, the basic model selection algorithm for these software packages may be interpreted as a weighting scheme that is based on the estimated posterior odds of selecting one model over another under a given set of prior distribution constraints (i.e., Bayes factors). This dissertation does not provide a detailed examination of the use of such methods; it refers the reader to a wealth of currently published analyses [Douady et al., 2003, Alfaro and Huelsenbeck, 2006, Huelsenbeck and Dyer, 2004, Nylander et al., 2004]

1.5.3 Information Loss

Assessments of model fit using information-loss estimators have become popular for model selection [Posada and Buckley, 2004, Alfaro and Huelsenbeck, 2006]. In 1974 Akaike developed an unbiased estimate for the expected, relative Kullback-Leibler (K-L) mean information number $(I(f,g))$ [Akaike, 1974, Sugiura, 1978]:

$$I(h,q) = \int_{\Re} h(x) log\big(\frac{h(x)}{q(x|\underline{\theta})}\big) dx. \tag{1.21}$$

The K-L number represents the amount of information lost by estimating the truth (represented with function, h) using a given estimation method (represented by q). $\underline{\theta}$ is a p-space vector of the parameters evaluated by q.

Akaike's estimate of the K-L number (known generally as the Akaike Information Criteria or AIC) is

$$\text{AIC} = 2p - 2ln[L(\underline{\theta}|data)]. \tag{1.22}$$

Note that the estimation of AIC substitutes $L(\underline{\theta}|data)$ with the iterative maximized likelihood following the work of Felsenstein [1981]. The value of 2p is essentially a penalty on the dimensionality of a given model (i.e., parsimonious models are favored relative to the penalty).

In application, AIC translates to an estimate of the amount of information lost while modeling genetic evolution. Fittingly, the smallest value of AIC from a group of candidate models would indicate the "best" performing one. Akaike referred to the AIC for this "best" model as the Minimum Information Theoretic Criterion Estimate (MAICE).

The assumptions underlying the use of AIC are the following

- $\hat{\theta}$ follows standard large sample asymptotic properties (i.e., $n >> p$)
- the models being compared using AIC must be drawn from the same underlying process

or data

- the distribution of the model parameter estimates is multivariate normal

In order to relax this constraint of large sample asymptotics Sugiura [1978] proposed an adjustment that effectively changed the scaling factor on the parsimony penalty of Akaike's original estimate

$$\mathrm{AIC}_{\mathrm{corr.}} = -2logL(\underline{\theta}|data) + 2p\big(\frac{K}{K-p-1}\big). \tag{1.23}$$

Here K represents the sample size, or in this case the number of codons or nucleotides in the sequence alignment.

Posada and Buckley [2004] discuss the appropriate application of information-loss estimates to be relative comparisons with the best model from among the candidate substitution models. They advocate the use of relative MAICE scores over the use of raw AIC values

$$\Delta\mathrm{AIC}_m = \mathrm{AIC}_m - \mathrm{MAICE}$$

The Bayesian Information Criteria (BIC) is also commonly used in assessing model selection and is essentially the Bayesian analogue to the information-loss method of Akaike.

$$\mathrm{BIC} = -2 * log(L) + p * log(K)$$

While AIC is theoretically based on the K-L mean information number, BIC is based on the Bayes factor and posterior probability. This results in an estimate of the log Bayes factor [Kass and Wasserman, 1994].

$$\text{Bayes Factor} = \frac{P(\text{data}|\text{model}_{m_1})}{P(\text{data}|\text{model}_{m_2})}$$
$$\text{Posterior Probability} = \frac{P(\text{data}|\text{model}_{m_1}) * P(\text{model}_{m_1})}{\sum_r P(\text{data}|\text{model}_{m_r}) * P(\text{model}_{m_r})}$$

While BIC may have theoretical differences with AIC, the application and interpretations of the two are analogous. Both methods and their variants enable multiple, simultaneous comparisons of models across a set of candidates regardless of the parameterization, or p. Also, these methods do not suffer from multiplicity as related to traditional statistical hypothesis testing, do not require independence across the models compared, and may therefore compare models both within and outside of hierarchies.

1.5.4 Genetic Algorithms

A codon model has P_{cod} possible substitution rates and these are clustered into p values. For the MG94 model, the $P_{cod} = 263$ rates are clustered into $p = 2$ possibilities for the synonymous and nonsynonymous changes. As described earlier, the number of possible rate clusterings can be 4032 but typically include a much smaller subset (263) in order to account for the constraints of single nucleotide substitutions and reversibility of the Markov process.

For computational purposes, the P_{cod} possible rates are encoded as a vector of integers such that each position uniquely corresponds to the clustering of a particular nonzero upper-diagonal substitution rate from R. The coding vector is length P_{cod} with each entry being an integer from 0 to $p - 1$. The constraints on the coding vectors are the same as those listed in Section 1.4.3

The coding vectors representing models are implemented with a genetic algorithm (GA) in order to identify optimal models over the substitution space. Selection and substitution of the set of models (or children) for each population of the GA, with size M, are created using the following algorithm:

- The initial population is seeded with a discrete uniform distribution of integers representing the substitution rate clusters for each model (also referred to as "children"). This initial population is then updated to equivalent representations that follow the constraints in Section 1.4.3. The population of vectors is also checked for any duplicates and these are removed and re-seeded until no duplicates exist.

- Follow-on populations are created with the following criteria:
  - The most fit individual from a population is retained for the following generation with probability one.
  - r_{mut} of the children in a given GA population are randomly selected for substitution. Of this subset of children, uniform random substitutions over the p clusterings are applied to each of the N_{sub} values of each child with probability r_{site}.
  - The remaining children of a follow-on GA population $[M(1 - r_{mut})]$ are randomly selected in pairs to create models via crossover for the next population. This is repeated until the number of models matches the designated population size. Details of the random selection of parents and the crossover process are described later.
- The process of follow-on populations is repeated until the best-fit model (child) does not change for a pre-specified number of consecutive populations. Once this criteria is met, the GA is terminated.

Model fitness is evaluated for models traversed by the GA using methods specified in Section 1.5.3 such as AIC and corrected AIC.

The selection of "parents" to create model children for subsequent populations is done using percentile scores that are based on the fitness assessments. For example, the random selection of a given "parent" using corrected AIC is $P(\text{selection}) = 1 - [(\text{AIC}_m - \text{MAICE})/\text{range}^*]$ where $range^*$ is the maximum AIC over the m models in the population minus the MAICE on a given population.

Crossover selection for each set of parent models is done using a random number coin toss to select the given element (substitution clustering).

Due to the computational burden of using GA runs to explore the codon model space, several constraints have been implemented to focus the search on particular subsets. Consistent with the methods described in the following Section 1.5.5, the GA runs set all S_{ij} values equivalent, essentially using the parameterization for synonymous codon changes under MG94 [Delport et

al., 2010b] (setting them all equal). The values of O_{ij} are then explored under a fixed cluster size, p, set prior to the run. For computational convenience, each GA run used a fixed value of p instead of allowing p to change (i.e., "dimension jump"). Extensive analyses were conducted that show convergence problems with "dimension jumping" GAs (results not shown).

The traverse of a GA was chosen as the method of obtaining candidate sets of models for averaging. This was chosen over other methods such as hLRT and Bayesian MCMC sampling due to the GA's relatively broad search range. hLRTs have a narrow pre-specified scope and Bayesian MCMC directly samples from the posterior distribution; however, despite the search range, GAs do not have a fully tractable set of statistical properties regarding convergence. Regardless, the mutation characteristic of the GA provides a mechanism of searching multiple local optima. It is this benefit, providing a broad search criteria, that is the justification for using this particular strategy of candidate model selection for codon substitution models.

1.5.5 CodonTest

Delport et al. [2010b] developed a selection procedure for reversible codon substitution models (CodonTest). Analogous with this dissertation, CodonTest is based on the implementation of Genetic Algorithms and information loss.

CodonTest explored the codon model space using a single rate parameter for synonymous substitutions and concentrated on fitting the nonsynonymous parameters. CodonTest's model selection method is analogous to forward selection. For iterative step one, the GA is constrained by $p_{\text{non}_1} = 2$ ($K = 3$) for the nonsynonymous rates. If the "best fit" model outperforms $p_{\text{non}_0} = 1$ ($K = 2$ or MG94), the GA moves to the next iterative step, setting $p_{\text{non}_2} = p_{\text{non}_1} + 1$. The iterative steps continue to repeat until the "best-fit" model at step X (p_{non_X}) does not outperform the previous model ($p_{\text{non}_{(X-1)}}$) by a given threshold.

Information loss used to assess model fit for CodonTest is a modified BIC estimator

$$\text{mBIC} = -2log(L) + 2p * log(K) \tag{1.24}$$

Note that this modified BIC increases the "penalty" associated with the dimensionality of a given model above that of the standard BIC function that only uses a penalty function of $2p$.

The most obvious drawback to CodonTest is the use of forward selection. This model selection method is well known to increase the risk of "key-hole" model solutions with poor characteristics such as inflated Type I error and upward-bias for estimates of the mean square error (MSE) [Blanchet et al., 2008]. The justification for using this method is that there are computational problems implementing GAs that can adaptively change the number of clusterings within the same run. Huelsenbeck et al. [2004] discusses these challenges while providing an analogous "reversible-jump" Markov Chain Monte Carlo (MCMC) for the nucleotide case. However, the proposed MCMC method does not efficiently translate to the GA case for codon models. We will not be exploring the MCMC proposed by Huelsenbeck on nucleotide models in favor of estimation and model averaging on censuses, both simulated and empirical.

Published results using CodonTest include the best fit model from the set of GA runs over the X iterative steps on p, a credible set resulting from the GA's traverse through the reversible codon model space, model averaged estimates considering the credible set, and estimation of a scale shift on the nonsynonymous rates of R.

Analysis of simulated and empirical results showed that the CodonTest method controlled overfitting, had relatively high precision under sequence conditions that include reasonable divergence and length, and generally outperformed the standard, published modeling methods with respect to information loss.

The model averaging in use with the results of CodonTest are of interest to this dissertation. The methods and weights for averaged estimates in use by CodonTest follow with those described by Posada and Buckley [2004] and Burnham and Anderson [2010]. Further details of the methods of model averaging are discussed in 1.5.6. However, this dissertation further investigates the methods of averaging on the GA's traverse through the set of reversible codon substitution models. While only the credible set was considered by CodonTest and weighted as a function of the mBIC estimates, this dissertation will assess the limits of what set of models

may be of interest and will also investigate methods of weighting models, to include both unique and non-unique representations of visited models by the GA. Full details are discussed in 1.5.6 as well as Chapters 2 and 3.

1.5.6 Model Averaging

Model Averaging has the advantage of directly incorporating estimates that account for the between and within-model variability instead of only within-model variability that single model methods account for. Additionally, model average methods easily implement AIC or BIC-based weights from among candidate sets and therefore scale selection risk across the domain of candidate models under consideration, as opposed to putting all decision risk into a single model selected [Buckland et al., 1997]. For the codon case, this dissertation analyzes candidate sets of models selected from the traverse of a genetic algorithm (GA) through the set of possible models. As the resulting set of models will not be independently distributed and not necessarily a simple random sample, additional analyses will be included to monitor the behavior of estimates resulting from these candidate sets of models [Yates, 2008].

Many standard model selection methods assume that the selected model (m) was analyzed *a priori*, and therefore $m = h$. As noted by Claeskens and Hjort [2008], historically these methods were developed from necessity that was based on computational and theoretical constraints. However, this assumption may not be practical as illustrated in a famous quote by George Box: "Essentially, all models are wrong, but some are useful" [Box and Draper, 1987]. A possible definition of a "useful" model is one with total sampling variance $\varphi_m = V(\varphi_h) + V(\varphi_{||m-h||})$ where $V(\varphi_h) >> V(\varphi_{||m-h||})$. Here $V(\varphi_h)$ and $V(\varphi_{||m-h||})$ represent the within and between-model variance, respectively [Hjort and Claeskens, 2003].

The sampling variance of model selection methods that assume $m = h$ effectively set $V(\varphi_{||m-h||}) = 0$ (or the uncertainty of the model selection process is conditioned out of the calculation). This produces estimates of the variance that may be biased, hypothesis tests that may not be size α, and confidence intervals that may be smaller than the nominal coverage

[Burnham and Anderson, 2010]. An exception to this issue is found in the Bayesian model averaging process where the uncertainty of the final selected model may adequately be taken into account with the posterior distribution [Hjort and Claeskens, 2003].

The model averaging process of this dissertation avoids these issues by assuming $m \neq h$, and weighting the average as a function of the expected K-L mean information number (equation 1.21).

The model averaged estimate is

$$\hat{\bar{\varphi}} = \sum_{m=1}^{M} w_m \hat{\varphi}_m \tag{1.25}$$

where M is the total number of models, w_m is the weight for the individual model estimates, $\sum_{m=1}^{M} w_m = 1$, and $\hat{\varphi}_m$ is the estimate of φ for the given model, m. For this dissertation, the weights (w_m) use the information loss estimators of AIC and BIC according to Buckland et al. [1997]

$$w_m = \frac{exp(-\Delta AIC_m/2)}{\sum_{i=1}^{M} exp(-\Delta AIC_{m_i}/2)} \tag{1.26}$$

Note that the application of these weights in a Bayesian methodology (by substituting BIC for AIC) results in an approximation of the posterior probability that model m is correct [Sugiura, 1978, Hjort and Claeskens, 2003, Burnham and Anderson, 2010].

The sampling variance of the averaged estimator is the weighted average of the MSE of φ_m where $\text{MSE}_{\varphi_m} = E[(\varphi_m - \bar{\varphi})^2 | q_m] = V(\varphi_m | q_m) + (\varphi_m - \bar{\varphi})^2$, $\bar{\varphi} = (1/m)\sum_{m=1}^{M} \varphi_m$ [Burnham and Anderson, 2010]. Therefore, the sampling variance of the model averaged estimate is

$$\begin{aligned} \text{V}(\hat{\bar{\varphi}}) &= \left[\sum_{m=1}^{M} w_m \Big(\text{V}(\hat{\varphi}_m | q_m) + (\varphi_m - \bar{\varphi})^2 \Big)^{1/2} \right]^2 \\ \text{and } \hat{\text{V}}(\hat{\bar{\varphi}}) &= \left[\sum_{m=1}^{M} \hat{w}_m \Big(\hat{\text{V}}(\hat{\varphi}_m | q_m) + (\hat{\varphi}_m - \hat{\bar{\varphi}})^2 \Big)^{1/2} \right]^2 \end{aligned} \tag{1.27}$$

The underlying assumption of equation 1.27 is that the M models are randomly selected. However, to generalize this for any set of M models, randomly selected or not, Burnham and Anderson [2010] provide the following

$$\begin{aligned}
&\mathrm{V}(\hat{\bar{\varphi}}) = \sum_{m=1}^{M}(w_m)^2 E\Big[(\hat{\varphi}_m - \bar{\varphi})^2 | q_m)\Big] + \sum_{m_i \neq m_j}^{M}\sum^{M} w_{m_i} w_{m_j}\Big[E(\hat{\varphi}_{m_i} - \bar{\varphi})(\hat{\varphi}_{m_j} - \bar{\varphi})|q_{m_i} q_{m_j}\Big] \\
&\text{where } E[(\hat{\varphi}_{m_i} - \bar{\varphi})^2 | q_{m_i}] = V(\hat{\varphi}_{m_i}|q_{m_i}) + (\varphi_i - \bar{\varphi})^2 \\
&\text{and } E\Big[(\hat{\varphi}_{m_i} - \bar{\varphi})(\hat{\varphi}_{m_j} - \bar{\varphi})|q_{m_i} q_{m_j}\Big] = \rho_{m_i m_j}\sqrt{E[(\hat{\varphi}_{m_i} - \bar{\varphi})^2|q_{m_i}]E[(\hat{\varphi}_{m_j} - \bar{\varphi})^2|q_{m_j}]}
\end{aligned} \tag{1.28}$$

Given $\rho_{m_i m_j} = \rho$, equation 1.28 reduces to

$$\begin{aligned}
\mathrm{V}(\hat{\bar{\varphi}}) =& (1-\rho)\Bigg[\sum_{m=1}^{M}(w_m)^2\Big[(V(\hat{\varphi}_m|q_m) + (\varphi_m - \bar{\varphi})^2\Big]\Bigg] \\
&+ \rho\Bigg[\sum_{m=1}^{M} w_m\Big(V(\hat{\varphi}_m|q_m) + (\varphi_m - \bar{\varphi})^2\Big)^{1/2}\Bigg]^2
\end{aligned} \tag{1.29}$$

Given a set of M models that are randomly selected, the unconditional variance is provided by equation 1.27. For a set of models that are not randomly selected, equation 1.28 still reduces to 1.27 given the assumption that $\rho_{m_i m_j} = \rho = 1$ [Burnham and Anderson, 2010].

Hjort and Claeskens [2003] derived the limit distribution of a general class of model averaged estimators and applied this to the case of weights given by Buckland et al. [1997] (equation 1.26). The resulting distribution is a mixture of normals with defined parameters. In order to describe these results the notation and the structure of the distribution are discussed below.

Given a model averaged estimator, $\hat{\bar{\varphi}}$, the misspecification is defined as

$$\begin{aligned}
q_m(y) &= q_m(y, \theta_0, \gamma_0 + \delta/\sqrt{n}) \\
&\text{where } \delta_1, \delta_2, ...\delta_q
\end{aligned} \tag{1.30}$$

δ represents the deviation of model m from the truth on the basis generated by q. Hjort and

Claeskens [2003] use this misspecification function to define deviations for a given model based on γ in a neighborhood of γ_0. The resulting distribution of $\sqrt{n}(\hat{\varphi}_m - \varphi_h) \sim \sum_{\forall m} w_m \Lambda$ where

$$\Lambda = \frac{\partial \varphi_m}{\partial \theta} J_{00}^{-1} H + \omega^t \{\delta - \hat{\delta}(D)\} \tag{1.31}$$

$$\text{where } H \sim N_p(0, J_{00})$$

Note that J_{00}^{-1} is from the Jacobian on the variance of the score function, $\frac{U(\varphi)}{V(\varphi)}$, where U and V are the partial derivatives on θ of the log of equation 1.31. However, U represents the model specified parameters and V is based on the misspecification defined by δ. Therefore J is defined as $\begin{pmatrix} J_{00} & J_{01} \\ J_{10} & J_{11} \end{pmatrix}$ and the inverse as $J^{-1} = \begin{pmatrix} J^{00} & J^{01} \\ J^{10} & J^{11} \end{pmatrix}$.

Hjort and Claeskens [2003] derived the means square error as $\tau_0^2 + E(\omega^t \hat{\delta} - \omega^t \delta)^2 = \omega^t E\{\hat{\delta}(B) - \delta\}\{\hat{\delta}(B) - \delta\}^t \omega$. $W = J^{10} H + J^{11} L = T(L - J_{10} J_{00}^{-1} H)$. $B_m = \hat{\delta} = \sqrt{n}(\hat{\gamma} - \gamma_0) \sim B = \delta + W \sim N_q(\delta, T)$ and $T = J^{11} = (J_{11} - J_{10} J_{00}^{-1} J_{01})^{-1}$. The mean and variance of the distribution are

$$\text{mean: } \omega^t \{\delta - E_{\hat{\delta}}(B)\} \tag{1.32}$$

$$\text{variance: } \tau_0^2 + \omega^t V_{\hat{\delta}}(B) \omega \tag{1.33}$$

where $\tau_0^2 = \left(\frac{\partial \varphi}{\partial \theta}\right)^2 J_{00}^{-1} \frac{\partial \varphi}{\partial \theta}$ and $\omega = J_{10} J_{00}^{-1} \frac{\partial \varphi}{\partial \theta} - \frac{\partial \varphi}{\partial \gamma}$.

Extensive simulation results are included in Chapters 2 and 3 to detail the performance of bias and variance for these model averaged estimates with respect to standard named models for phylogenetic inference.

1.6 Discussion

Substitution modeling in the field of molecular evolution has unique challenges. Given the discrete nature of clustering possible rates to define possible models, selection procedures have been developed with decision trees (hLRT), Genetic Algorithms, and fit statistics based on

information loss. Since candidate models are dependent on the same data with estimates of interest well defined across the full census, averaged estimates are easily implemented with well-established weights that are based on information loss. These averaged solutions provide desirable properties that account for the across-model variability in model selection as opposed to only accounting for within-model variability per historic, single model selection solutions.

However, given the challenges of selecting candidate sets of models from either a comprehensive or limited subset of models in the codon model space, this dissertation explores the use of GAs. The use of GAs is intended to be a pseudo-selection procedure that results in candidate subsets of models that are based on model fit. The analysis of the properties of model averaged bias and variability for these estimates is the primary focus of this dissertation. The goal is to show that this results in robust estimates of substitution rates that may be implemented in real-world data and facilitate a better understanding of molecular evolution.

Chapter 2

Nucleotide Analyses

This chapter presents our analyses of model averaged estimates of the transition-transversion ratio with respect to the census of reversible models. Model averaging was conducted for a variety of different simulated datasets. The various averaged estimators were then compared with each other and across the set of named models from Section 1.2.3. Resulting analyses were consistent with the use of candidate subsets of models that are based on biological considerations. Furthermore, the results provided a large range of conditions that, accompanied with comprehensive analyses of the full model space, resulted in a broad set of research on the expected performance of named models and averaged estimators. We showed that these averaged estimators are robust across a broad spectrum of conditions, and that the census of reversible models have an interesting bimodal distribution of variance. Furthermore, the use of information-loss provides robust estimators even in conditions using incorrect subsets of models for the averaged estimates. Finally, the body of work provides a proof-of-concept for the extension of the averaged estimators to the codon model space in Chapter 3.

2.1 Data Simulation Settings

The data simulations and analyses that we produced were run on the North Carolina State University Bioinformatics Research Center (BRC) Beowulf cluster running Ubuntu and using

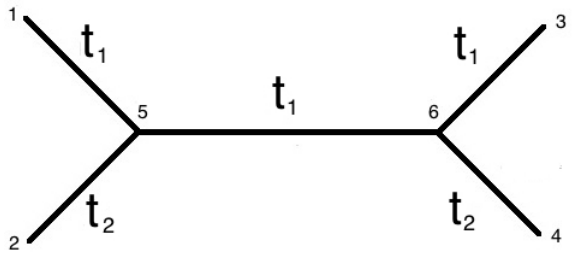
Intel Xeon E5-4627 v2 3.3 GHz and E5-4620 2.2 GHz processors with memory performance of 1866 MHz and 1333 MHz, respectively. Limited computations and data analyses were also run using OSX with a 2.3 GHz Intel Core i7-3615QM quad-core processor with DDR3 memory performance at 1600 MHz.

Our nucleotide analyses were performed on simulated and empirical data. For each alignment a complete census was conducted on the reversible substitution models. All data simulations consisted of 1000 replicates with a phylogenetic tree of four taxa. The tree structure that we used was the same as figures 1.4 and 1.5 [or notationally, ((Taxa 1, Taxa 2), Taxa 3, Taxa 4)]. As a result of assumption 5 and Section 1.2, all likelihoods are conditioned on the phylogenetic trees used for the data simulations. Therefore, the structure of the sequence alignment is assumed known and using a four taxa alignment.

The goal of the various versions of the simulated data that we produced was to provide a comprehensive exploration of the performance for each possible substitution model and the various model averaged estimators. The data simulation settings were varied for sequence length, total sequence divergence, branch lengths, and true substitution parameter values. We varied the substitution parameters according select, named models (HKY85, TN93, and GTR). Within the simulations based on these models, various values of κ were set via the individual substitution parameters and stationary distribution. The stationary distribution used for all simulations was unbalanced and consisted of the following: $\pi_A = .4$, $\pi_C = .3$, $\pi_G = .2$, and $\pi_T = .1$.

We set the branch length structure for the unrooted phylogenetic tree to scale with the total expected substitutions in the same manner as the four-taxon simulation work of Huelsenbeck and Hillis [1993]. Huelsenbeck and Hillis' research was based on two distinct branch lengths for a four taxa phylogenetic tree that we implemented as scaling factors for the branch lengths (times t_1 and t_2 per figure 2.1).

Figure 2.1: Huelsenbeck and Hillis [1993] Unrooted Four-Taxa Phylogenetic Tree



Based on this structure, we conducted simulations over three separate ratios of ν_1 to ν_2. Within each of these separate structures, 18 different sets of branch lengths (sequence divergence) were fit to assess estimator performance as sequence divergence changes per table 2.1. Additionally, to assess estimator performance as the sequence divergence increased to a point of unstable estimation, we included an extended scale of sequence divergence by increasing the total time (t) for Case I. The extended cases are shown in table 2.2.

Table 2.1: Sequence Divergence

Case I		Case II		Case III		Total Time (t)
ν_1	ν_2	ν_1	ν_2	ν_1	ν_2	
0.27	0.60	0.53	0.23	0.13	0.80	2
0.40	0.90	0.80	0.35	0.20	1.20	3
0.80	1.80	1.60	0.70	0.40	2.40	6
1.20	2.70	2.50	1.10	0.60	3.60	9
1.60	3.60	3.15	1.40	0.80	4.80	12
2.67	6.00	5.60	2.50	1.34	8.00	20

Table 2.2: Sequence Divergence for Extended Values of Case I

Case I		Total Time (t)
ν_1	ν_2	
3.33	7.50	25
4.00	9.00	30
6.00	13.50	45
8.00	18.00	60

For each permutation of simulation settings, 1000 replicated datasets were generated and for each, estimates of κ were calculated on the census of 203 reversible models, the best fit model, and model averaged estimators for each of the three sets of candidate models:

1. the full set of 203 reversible models
2. models exclusively clustering transitions and transversions separately ("Ts/Tv models")
3. the models that cluster at least one transition and one transversion substitution in the same cluster (the complement models to [2] or "non-Ts/Tv models").

Data Simulation Process

All data was simulated using the R Foundation for Statistical Computing Platform (R Platform) [R Core Team, 2013]. All programming code associated with the R Platform are covered by the General Public License (GPL) or Lesser General Public License (LGPL). Data was simulated with the following six-step process:

1. Nucleotide sites were independently generated for the root node of the phylogeny using the stationary distribution specified earlier. The nucleotide at site k, y^k_{root}, was mapped using $u^k_{temp} \overset{ind.}{\sim} U(0,1)$ for $k = 1, 2, ...K$ such that if $u^k_{root} \leq \pi_A$ then $y^k = A$; else if $u^k_{temp} \leq \pi_A + \pi_C$ then $y^k = C$; else if $u^k_{temp} \leq \pi_A + \pi_C + \pi_G$ then $y^k = G$; else $y^k = T$. Note that in this case, K is the sequence length setting for the simulated data.

2. The substitution matrix, R, was constructed using the stationary distribution and six substitution parameters (θ_p for $p = 0, 1, 2, 3, 4, 5$):

$$R = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} * & \theta_0\pi_C & \theta_1\pi_G & \theta_2\pi_T \\ \theta_0\pi_A & * & \theta_3\pi_G & \theta_4\pi_T \\ \theta_1\pi_A & \theta_3\pi_C & * & \theta_5\pi_T \\ \theta_2\pi_A & \theta_4\pi_C & \theta_5\pi_G & * \end{pmatrix} \end{array}$$

The diagonal elements ($*$) were defined as the negative sum of off-diagonal elements, and therefore each row summed to zero. This matrix was then multiplied by the associated branch length, ν_b, of the rooted tree. Note that ν_2 from the unrooted tree in figure 2.1 is the sum of ν_4 and ν_6 from the rooted tree in figure 1.1.

3. Transition probabilities were recovered using the matrix exponentiation approximation function, $expm(\nu_b R)$, from the Matrix package for the R Platform (i.e., $P = e^{\nu_b * R}$) [Bates and Maechler, 2015]. The default method of Higham [2008] was used in the $expm(\nu_b R)$ function to approximate a Taylor Series Expansion per Section 1.2.3.

4. The transition matrix, P, was then used to generate a non-root node of the alignment. The node(s) closest to the root were produced first and built out toward the leaf nodes sequentially. For example, given the root sequence from (1), the sequence for node 6 from figure 1.1 was based on the transition probabilities, $P = e^{\nu_6 * R}$. Therefore, for some k, if $y^k_{root} = A$ implies that for $u_{temp} \sim U(0, 1)$, if $u_{temp} \leq P_{A,C}$ then $y^k_6 = C$; else if $u_{temp} \leq P_{A,C} + P_{A,G}$ then $y^k_6 = G$; else if $u_{temp} \leq P_{A,C} + P_{A,G} + P_{A,T}$ then $y^k_6 = T$; else $y^k_6 = y^k_{root}$.

5. Steps (3) and (4) were repeated for all sites and nodes of the rooted phylogeny. In the case of figure 1.1, sequences for nodes 1-6 were created.

6. The sequence alignment was output for use with only the leaf node sequences (nodes 1-4).

In addition to the two-dimensional distances from the true value of κ, estimates were compared in terms of L2 Norms for each of the substitution models ($L_{2_m} = ||\hat{\underline{\theta}}_m - \underline{\theta}||; m = 1, 2, ..., 203$). In the case of the model averages, each of the substitution estimates were calculated using the same model weights applied to each of the individual parameter estimates. Variance was also analyzed for this dissertation, and included histogram and boxplots stratified for both biological and non-biologically-based estimates.

The model averaged estimate considering all 203 reversible nucleotide models provided a baseline on the performance of the other averaged estimators. Due to the role of biological characteristics in modeling, we expected the model average considering the 30 transitions and transversion models to perform best.

2.2 Simulation Results

2.2.1 Transition/Transversion Ratios

Analyses on the census of reversible nucleotide models were conducted per table 2.3. The highlighted models represent the 30 that exclusively cluster transitions and transversions separately ("Ts/Tv Models"). Also note that the dot plots in this section follow the ordering of individual models per table 2.3. This implies that for these plots, the simplest models occur earliest followed by the more complicated models. Therefore, the sets begin with F81 and complete with GTR.

Table 2.3: Reversible Nucleotide Models

No.	A to C	A to G	A to T	C to G	C to T	G to T
1 (F81)	θ_0	θ_0	θ_0	θ_0	θ_0	θ_0
2	θ_0	θ_0	θ_0	θ_0	θ_0	θ_1
3	θ_0	θ_0	θ_0	θ_0	θ_1	θ_0
4	θ_0	θ_0	θ_0	θ_0	θ_1	θ_1
5	θ_0	θ_0	θ_0	θ_1	θ_0	θ_0
6	θ_0	θ_0	θ_0	θ_1	θ_0	θ_1
7	θ_0	θ_0	θ_0	θ_1	θ_1	θ_0
8	θ_0	θ_0	θ_0	θ_1	θ_1	θ_1
9	θ_0	θ_0	θ_1	θ_0	θ_0	θ_0
10	θ_0	θ_0	θ_1	θ_0	θ_0	θ_1
11	θ_0	θ_0	θ_1	θ_0	θ_1	θ_0
12	θ_0	θ_0	θ_1	θ_0	θ_1	θ_1
13	θ_0	θ_0	θ_1	θ_1	θ_0	θ_0
14	θ_0	θ_0	θ_1	θ_1	θ_0	θ_1
15	θ_0	θ_0	θ_1	θ_1	θ_1	θ_0
16	θ_0	θ_0	θ_1	θ_1	θ_1	θ_1
17	θ_0	θ_1	θ_0	θ_0	θ_0	θ_0
18	θ_0	θ_1	θ_0	θ_0	θ_0	θ_1
19 (HKY85)	θ_0	θ_1	θ_0	θ_0	θ_1	θ_0
20	θ_0	θ_1	θ_0	θ_0	θ_1	θ_1
21	θ_0	θ_1	θ_0	θ_1	θ_0	θ_0
22	θ_0	θ_1	θ_0	θ_1	θ_0	θ_1
23	θ_0	θ_1	θ_0	θ_1	θ_1	θ_0
24	θ_0	θ_1	θ_0	θ_1	θ_1	θ_1
25	θ_0	θ_1	θ_1	θ_0	θ_0	θ_0
26	θ_0	θ_1	θ_1	θ_0	θ_0	θ_1
27	θ_0	θ_1	θ_1	θ_0	θ_1	θ_0
28	θ_0	θ_1	θ_1	θ_0	θ_1	θ_1
29	θ_0	θ_1	θ_1	θ_1	θ_0	θ_0
30	θ_0	θ_1	θ_1	θ_1	θ_0	θ_1
31	θ_0	θ_1	θ_1	θ_1	θ_1	θ_0
32	θ_0	θ_1	θ_1	θ_1	θ_1	θ_1
33	θ_0	θ_0	θ_0	θ_0	θ_1	θ_2
34	θ_0	θ_0	θ_0	θ_1	θ_0	θ_2
35	θ_0	θ_0	θ_0	θ_1	θ_1	θ_2
36	θ_0	θ_0	θ_0	θ_1	θ_2	θ_0
37	θ_0	θ_0	θ_0	θ_1	θ_2	θ_1
38	θ_0	θ_0	θ_0	θ_1	θ_2	θ_2
39	θ_0	θ_0	θ_1	θ_0	θ_0	θ_2
40	θ_0	θ_0	θ_1	θ_0	θ_1	θ_2
41	θ_0	θ_0	θ_1	θ_0	θ_2	θ_0
42	θ_0	θ_0	θ_1	θ_0	θ_2	θ_1
43	θ_0	θ_0	θ_1	θ_0	θ_2	θ_2
44	θ_0	θ_0	θ_1	θ_1	θ_0	θ_2
45	θ_0	θ_0	θ_1	θ_1	θ_1	θ_2
46	θ_0	θ_0	θ_1	θ_1	θ_2	θ_0
47	θ_0	θ_0	θ_1	θ_1	θ_2	θ_1
48	θ_0	θ_0	θ_1	θ_1	θ_2	θ_2
49	θ_0	θ_0	θ_1	θ_2	θ_0	θ_0
50	θ_0	θ_0	θ_1	θ_2	θ_0	θ_1
51	θ_0	θ_0	θ_1	θ_2	θ_0	θ_2
52	θ_0	θ_0	θ_1	θ_2	θ_1	θ_0
53	θ_0	θ_0	θ_1	θ_2	θ_1	θ_1
54	θ_0	θ_0	θ_1	θ_2	θ_1	θ_2
55	θ_0	θ_0	θ_1	θ_2	θ_2	θ_0
56	θ_0	θ_0	θ_1	θ_2	θ_2	θ_1
57	θ_0	θ_0	θ_1	θ_2	θ_2	θ_2
58	θ_0	θ_1	θ_0	θ_0	θ_0	θ_2
59	θ_0	θ_1	θ_0	θ_0	θ_1	θ_2
60 (TN93)	θ_0	θ_1	θ_0	θ_0	θ_2	θ_0
61	θ_0	θ_1	θ_0	θ_0	θ_2	θ_1
62	θ_0	θ_1	θ_0	θ_0	θ_2	θ_2
63	θ_0	θ_1	θ_0	θ_1	θ_0	θ_2
64	θ_0	θ_1	θ_0	θ_1	θ_1	θ_2
65	θ_0	θ_1	θ_0	θ_1	θ_2	θ_0
66	θ_0	θ_1	θ_0	θ_1	θ_2	θ_1
67	θ_0	θ_1	θ_0	θ_1	θ_2	θ_2
68	θ_0	θ_1	θ_0	θ_2	θ_0	θ_0
69	θ_0	θ_1	θ_0	θ_2	θ_0	θ_1
70	θ_0	θ_1	θ_0	θ_2	θ_0	θ_2
71	θ_0	θ_1	θ_0	θ_2	θ_1	θ_0
72	θ_0	θ_1	θ_0	θ_2	θ_1	θ_1
73	θ_0	θ_1	θ_0	θ_2	θ_1	θ_2
74	θ_0	θ_1	θ_0	θ_2	θ_2	θ_0
75	θ_0	θ_1	θ_0	θ_2	θ_2	θ_1
76	θ_0	θ_1	θ_0	θ_2	θ_2	θ_2
77	θ_0	θ_1	θ_1	θ_0	θ_0	θ_2
78	θ_0	θ_1	θ_1	θ_0	θ_1	θ_2
79	θ_0	θ_1	θ_1	θ_0	θ_2	θ_0
80	θ_0	θ_1	θ_1	θ_0	θ_2	θ_1
81	θ_0	θ_1	θ_1	θ_0	θ_2	θ_2
82	θ_0	θ_1	θ_1	θ_1	θ_0	θ_2
83	θ_0	θ_1	θ_1	θ_1	θ_1	θ_2
84	θ_0	θ_1	θ_1	θ_1	θ_2	θ_0
85	θ_0	θ_1	θ_1	θ_1	θ_2	θ_1
86	θ_0	θ_1	θ_1	θ_1	θ_2	θ_2
87	θ_0	θ_1	θ_1	θ_2	θ_0	θ_0
88	θ_0	θ_1	θ_1	θ_2	θ_0	θ_1
89	θ_0	θ_1	θ_1	θ_2	θ_0	θ_2
90	θ_0	θ_1	θ_1	θ_2	θ_1	θ_0
91	θ_0	θ_1	θ_1	θ_2	θ_1	θ_1
92	θ_0	θ_1	θ_1	θ_2	θ_1	θ_2
93	θ_0	θ_1	θ_1	θ_2	θ_2	θ_0
94	θ_0	θ_1	θ_1	θ_2	θ_2	θ_1
95	θ_0	θ_1	θ_1	θ_2	θ_2	θ_2
96	θ_0	θ_1	θ_2	θ_0	θ_0	θ_0
97	θ_0	θ_1	θ_2	θ_0	θ_0	θ_1
98	θ_0	θ_1	θ_2	θ_0	θ_0	θ_2
99	θ_0	θ_1	θ_2	θ_0	θ_1	θ_0
100	θ_0	θ_1	θ_2	θ_0	θ_1	θ_1
101	θ_0	θ_1	θ_2	θ_0	θ_1	θ_2
102	θ_0	θ_1	θ_2	θ_0	θ_2	θ_0
103	θ_0	θ_1	θ_2	θ_0	θ_2	θ_1
104	θ_0	θ_1	θ_2	θ_0	θ_2	θ_2
105	θ_0	θ_1	θ_2	θ_1	θ_0	θ_0
106	θ_0	θ_1	θ_2	θ_1	θ_0	θ_1
107	θ_0	θ_1	θ_2	θ_1	θ_0	θ_2
108	θ_0	θ_1	θ_2	θ_1	θ_1	θ_0
109	θ_0	θ_1	θ_2	θ_1	θ_1	θ_1
110	θ_0	θ_1	θ_2	θ_1	θ_1	θ_2
111	θ_0	θ_1	θ_2	θ_1	θ_2	θ_0
112	θ_0	θ_1	θ_2	θ_1	θ_2	θ_1
113	θ_0	θ_1	θ_2	θ_1	θ_2	θ_2
114	θ_0	θ_1	θ_2	θ_2	θ_0	θ_0
115	θ_0	θ_1	θ_2	θ_2	θ_0	θ_1
116	θ_0	θ_1	θ_2	θ_2	θ_0	θ_2
117	θ_0	θ_1	θ_2	θ_2	θ_1	θ_0
118	θ_0	θ_1	θ_2	θ_2	θ_1	θ_1
119	θ_0	θ_1	θ_2	θ_2	θ_1	θ_2
120	θ_0	θ_1	θ_2	θ_2	θ_2	θ_0
121	θ_0	θ_1	θ_2	θ_2	θ_2	θ_1
122	θ_0	θ_1	θ_2	θ_2	θ_2	θ_2
123	θ_0	θ_0	θ_0	θ_1	θ_2	θ_3
124	θ_0	θ_0	θ_1	θ_0	θ_2	θ_3
125	θ_0	θ_0	θ_1	θ_1	θ_2	θ_3
126	θ_0	θ_0	θ_1	θ_2	θ_0	θ_3
127	θ_0	θ_0	θ_1	θ_2	θ_1	θ_3
128	θ_0	θ_0	θ_1	θ_2	θ_2	θ_3
129	θ_0	θ_0	θ_1	θ_2	θ_3	θ_0
130	θ_0	θ_0	θ_1	θ_2	θ_3	θ_1
131	θ_0	θ_0	θ_1	θ_2	θ_3	θ_2
132	θ_0	θ_0	θ_1	θ_2	θ_3	θ_3
133	θ_0	θ_1	θ_0	θ_0	θ_2	θ_3
134	θ_0	θ_1	θ_0	θ_1	θ_2	θ_3
135	θ_0	θ_1	θ_0	θ_2	θ_0	θ_3
136	θ_0	θ_1	θ_0	θ_2	θ_1	θ_3
137	θ_0	θ_1	θ_0	θ_2	θ_2	θ_3
138	θ_0	θ_1	θ_0	θ_2	θ_3	θ_0
139	θ_0	θ_1	θ_0	θ_2	θ_3	θ_1
140	θ_0	θ_1	θ_0	θ_2	θ_3	θ_2
141	θ_0	θ_1	θ_0	θ_2	θ_3	θ_3
142	θ_0	θ_1	θ_1	θ_0	θ_2	θ_3
143	θ_0	θ_1	θ_1	θ_1	θ_2	θ_3
144	θ_0	θ_1	θ_1	θ_2	θ_0	θ_3
145	θ_0	θ_1	θ_1	θ_2	θ_1	θ_3
146	θ_0	θ_1	θ_1	θ_2	θ_2	θ_3
147	θ_0	θ_1	θ_1	θ_2	θ_3	θ_0
148	θ_0	θ_1	θ_1	θ_2	θ_3	θ_1
149	θ_0	θ_1	θ_1	θ_2	θ_3	θ_2
150	θ_0	θ_1	θ_1	θ_2	θ_3	θ_3
151	θ_0	θ_1	θ_2	θ_0	θ_0	θ_3
152	θ_0	θ_1	θ_2	θ_0	θ_1	θ_3
153	θ_0	θ_1	θ_2	θ_0	θ_2	θ_3
154	θ_0	θ_1	θ_2	θ_0	θ_3	θ_0
155	θ_0	θ_1	θ_2	θ_0	θ_3	θ_1
156	θ_0	θ_1	θ_2	θ_0	θ_3	θ_2
157	θ_0	θ_1	θ_2	θ_0	θ_3	θ_3
158	θ_0	θ_1	θ_2	θ_1	θ_0	θ_3
159	θ_0	θ_1	θ_2	θ_1	θ_1	θ_3
160	θ_0	θ_1	θ_2	θ_1	θ_2	θ_3
161	θ_0	θ_1	θ_2	θ_1	θ_3	θ_0
162	θ_0	θ_1	θ_2	θ_1	θ_3	θ_1
163	θ_0	θ_1	θ_2	θ_1	θ_3	θ_2
164	θ_0	θ_1	θ_2	θ_1	θ_3	θ_3
165	θ_0	θ_1	θ_2	θ_2	θ_0	θ_3
166	θ_0	θ_1	θ_2	θ_2	θ_1	θ_3
167	θ_0	θ_1	θ_2	θ_2	θ_2	θ_3
168	θ_0	θ_1	θ_2	θ_2	θ_3	θ_0
169	θ_0	θ_1	θ_2	θ_2	θ_3	θ_1
170	θ_0	θ_1	θ_2	θ_2	θ_3	θ_2
171	θ_0	θ_1	θ_2	θ_2	θ_3	θ_3
172	θ_0	θ_1	θ_2	θ_3	θ_0	θ_0
173	θ_0	θ_1	θ_2	θ_3	θ_0	θ_1
174	θ_0	θ_1	θ_2	θ_3	θ_0	θ_2
175	θ_0	θ_1	θ_2	θ_3	θ_0	θ_3
176	θ_0	θ_1	θ_2	θ_3	θ_1	θ_0
177	θ_0	θ_1	θ_2	θ_3	θ_1	θ_1
178	θ_0	θ_1	θ_2	θ_3	θ_1	θ_2
179	θ_0	θ_1	θ_2	θ_3	θ_1	θ_3
180	θ_0	θ_1	θ_2	θ_3	θ_2	θ_0
181	θ_0	θ_1	θ_2	θ_3	θ_2	θ_1
182	θ_0	θ_1	θ_2	θ_3	θ_2	θ_2
183	θ_0	θ_1	θ_2	θ_3	θ_2	θ_3
184	θ_0	θ_1	θ_2	θ_3	θ_3	θ_0
185	θ_0	θ_1	θ_2	θ_3	θ_3	θ_1
186	θ_0	θ_1	θ_2	θ_3	θ_3	θ_2
187	θ_0	θ_1	θ_2	θ_3	θ_3	θ_3
188	θ_0	θ_0	θ_1	θ_2	θ_3	θ_4
189	θ_0	θ_1	θ_0	θ_2	θ_3	θ_4
190	θ_0	θ_1	θ_2	θ_0	θ_3	θ_4
191	θ_0	θ_1	θ_2	θ_3	θ_0	θ_4
192	θ_0	θ_1	θ_2	θ_3	θ_4	θ_0
193	θ_0	θ_1	θ_1	θ_2	θ_3	θ_4
194	θ_0	θ_1	θ_2	θ_1	θ_3	θ_4
195	θ_0	θ_1	θ_2	θ_3	θ_1	θ_4
196	θ_0	θ_1	θ_2	θ_3	θ_4	θ_1
197	θ_0	θ_1	θ_2	θ_2	θ_3	θ_4
198	θ_0	θ_1	θ_2	θ_3	θ_2	θ_4
199	θ_0	θ_1	θ_2	θ_3	θ_4	θ_2
200	θ_0	θ_1	θ_2	θ_3	θ_3	θ_4
201	θ_0	θ_1	θ_2	θ_3	θ_4	θ_3
202	θ_0	θ_1	θ_2	θ_3	θ_4	θ_4
203 (GTR)	θ_0	θ_1	θ_2	θ_3	θ_4	θ_5

Highlighted models represent Ts/Tv models defined in Section 3.1;

The figures in this section include dotplots that illustrate the $\hat{\kappa}$ estimates for each of the 203 reversible models with each point in each figure representing 1000 replicate datasets being used to estimate that value. The red points show the 30 models that exclusively parameterize the transitions and transversions separately ("Ts/Tv models" that are highlighted in yellow in table 2.3). We colored the named models separately with the F81 using purple, HKY85 orange, TN93 blue, and GTR brown. The model averaged estimates are included on the far right of the figures with the best fit model in yellow, the model average over all models in dark khaki, the average over the "Ts/Tv models" in green, and the average over the "non-Ts/Tv models" in maroon. Finally, the true value of κ used to generate the data simulations is the dotted black line.

Corresponding boxplots for this section are included in Appendix A and illustrate the results that are summarized in tabular form within this section. These boxplots include the medians as the solid black line through the interquartile range with means represented by the red dots. Estimates are described on the x-axis.

HKY85 Data Analyses

Table 2.4 lists the simulation settings that are based on the HKY85 model used to produce the corresponding analyses in this section. Note that the analyses listed here include Case I from tables 2.1 and 2.2. The Case I analyses were run on sequence lengths of 250 (figures A.1 to A.3), 500 (figures 2.2 and A.4 to A.6), 1000 (figures 2.3 and A.7 to A.9), and 3000 (figures A.10 to A.12) nucleotide sites. The complete set of figures for all analyses is located in Appendix A.

Table 2.4: HKY85 Simulation Settings

Relative Substitution Rates										
θ_{AC}	θ_{AG}	θ_{AT}	θ_{CG}	θ_{CT}	θ_{GT}	E(Ts/Tv)	t	E(Ts)	E(Tv)	E(To)
0.07	0.16	0.07	0.07	0.16	0.07	2.095	2	0.070	0.067	0.138
							3	0.106	0.101	0.206
							6	0.211	0.202	0.413
							9	0.317	0.302	0.619
							12	0.422	0.403	0.826
							20	0.704	0.672	1.376
							25	0.880	0.840	1.720
							30	1.056	1.008	2.064
							45	1.584	1.512	3.096
							60	2.112	2.016	4.128

θ_{ij} = the instantaneous substitution rate of nucleotide i to j; t = total time;
Tv = transversion; To = total substitutions (substitutions per site)

Table 2.5: Estimated Ts/Tv Comparisons for HKY85 Data with E(To) = 0.138 to 1.376

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
2.095	0.138	250	2.260	2.266	2.257	2.096	1.922	2.258	1.741
		500	2.166	2.168	2.165	2.084	1.950	2.165	1.728
		1000	2.140	2.141	2.139	2.107	2.043	2.139	1.787
		3000	2.097	2.098	2.097	2.094	2.088	2.097	1.807
	0.206	250	2.195	2.200	2.193	2.081	1.925	2.193	1.729
		500	2.155	2.158	2.155	2.105	2.008	2.154	1.765
		1000	2.115	2.117	2.114	2.098	2.060	2.114	1.792
		3000	2.099	2.100	2.098	2.098	2.095	2.099	1.827
	0.413	250	2.161	2.166	2.160	2.099	1.990	2.160	1.770
		500	2.138	2.14	2.137	2.116	2.071	2.137	1.807
		1000	2.106	2.107	2.106	2.101	2.089	2.106	1.820
		3000	2.095	2.095	2.094	2.095	2.095	2.095	1.839
	0.619	250	2.131	2.138	2.133	2.083	2.007	2.132	1.788
		500	2.116	2.119	2.115	2.104	2.072	2.115	1.818
		1000	2.121	2.123	2.121	2.119	2.113	2.121	1.845
		3000	2.094	2.094	2.093	2.094	2.094	2.094	1.848
	0.826	250	2.114	2.122	2.116	2.080	2.010	2.114	1.79
		500	2.126	2.130	2.126	2.116	2.096	2.126	1.837
		1000	2.106	2.108	2.106	2.105	2.101	2.106	1.845
		3000	2.095	2.096	2.095	2.095	2.095	2.095	1.858
	1.376	250	2.132	2.144	2.138	2.101	2.052	2.134	1.835
		500	2.119	2.126	2.123	2.112	2.097	2.121	1.863
		1000	2.110	2.112	2.111	2.110	2.107	2.111	1.869
		3000	2.099	2.100	2.099	2.099	2.099	2.099	1.879

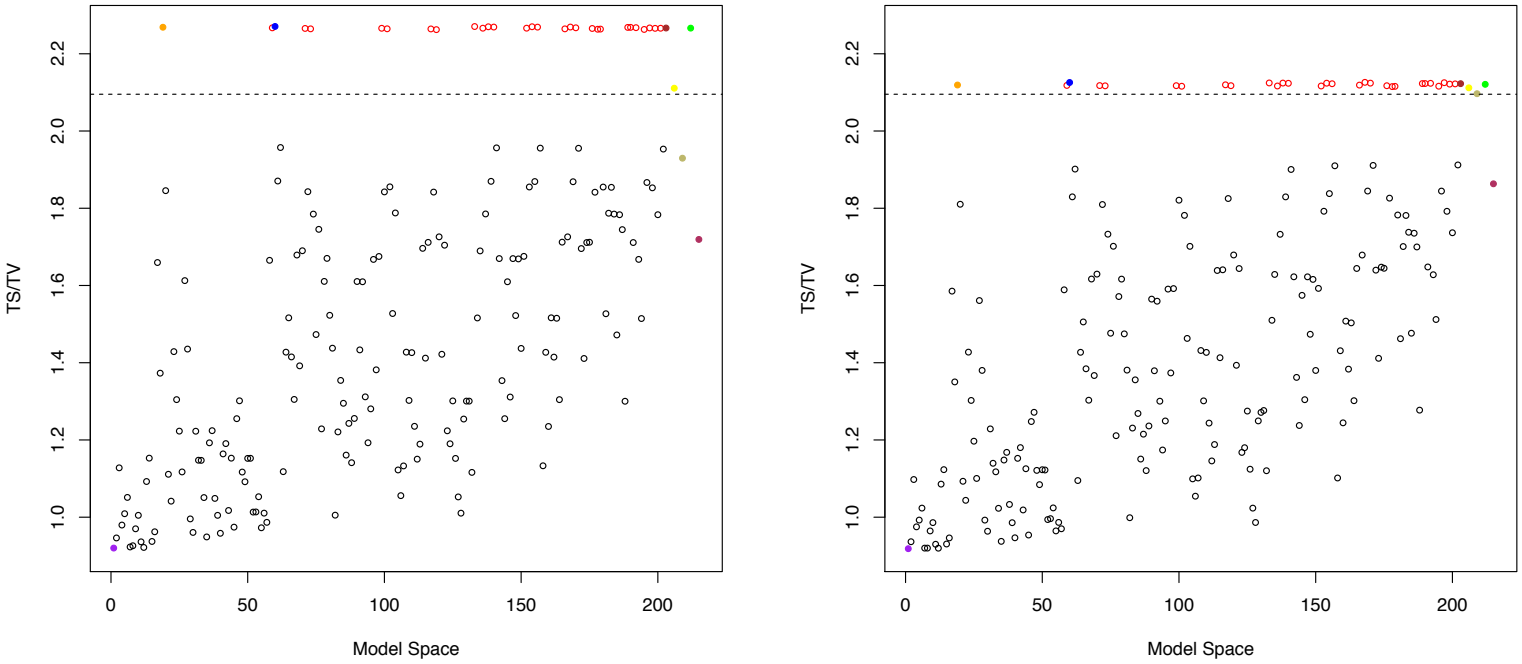
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Note: Substitution rates are defined as $\theta_0 = 0.07$, $\theta_1 = 0.16$, $\theta_2 = 0.07$, $\theta_3 = 0.07$, $\theta_4 = 0.16$, $\theta_5 = 0.07$

Table 2.6: Estimated Ts/Tv Comparisons for HKY85 Data with E(To) = 1.72 to 4.128

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
2.095	1.720	250	2.130	2.145	2.137	2.103	2.048	2.133	1.841
		500	2.119	2.128	2.125	2.113	2.098	2.122	1.873
		1000	2.110	2.114	2.112	2.110	2.107	2.110	1.883
		3000	2.095	2.096	2.096	2.095	2.095	2.095	1.888
	2.064	250	2.137	2.159	2.152	2.112	2.665	2.443	2.797
		500	2.112	2.122	2.120	2.104	2.086	2.115	1.881
		1000	2.104	2.109	2.108	2.104	2.101	2.106	1.893
		3000	2.099	2.100	2.100	2.099	2.099	2.099	1.902
	3.096	250	2.409	3.593	311.754	19.783	126.485	88.377	192.04
		500	2.142	2.287	47.542	2.152	21.772	6.899	83.890
		1000	2.127	2.142	21.419	2.133	3.310	2.313	25.730
		3000	1.332	2.110	2.111	2.107	2.107	2.107	2.146
	4.128	250	6.750	10.280	783.961	118.915	498.700	538.929	499.705
		500	3.155	5.844	900.778	77.269	458.614	363.821	631.087
		1000	2.216	3.004	406.190	23.480	219.796	126.456	546.863
		3000	2.118	2.153	90.800	2.129	12.310	5.430	336.678

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Note: Substitution rates are defined as $\theta_0 = 0.07$, $\theta_1 = 0.16$, $\theta_2 = 0.07$, $\theta_3 = 0.07$, $\theta_4 = 0.16$, $\theta_5 = 0.07$

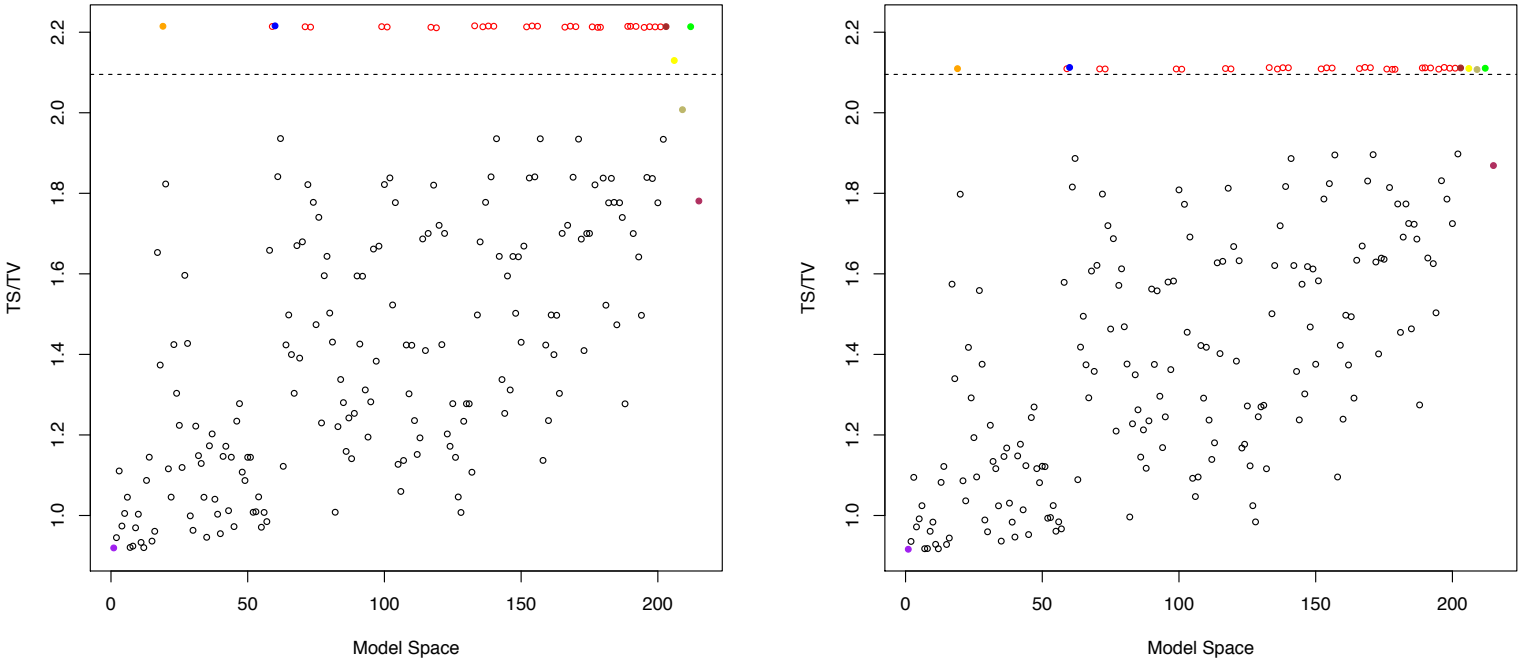
Figure 2.2: Ts/Tv Estimates on HKY85 Data with Sequence Length 500



E(To) = 0.069 **1.38**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure 2.3: Ts/Tv Estimates on HKY85 Data with Sequence Length 1000



E(To) = 0.069 **1.38**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Due to the design of using HKY85 to simulate the data, the most obvious finding in these results is the separation of biologically significant models for the estimation of κ from the non-biologically significant models (i.e., the red versus block dots in the plots). Analyses on the $AIC_{corr.}$ fitness scores show a similar clustering pattern (results not shown).

The figures show that the Ts/Tv models follow with the structure of the simulated data and the non-Ts/Tv models (black) estimate κ with a clear bias (downward). Additionally, the averaged estimators performed as expected with the average over the Ts/Tv models performing in-line with the estimates of the individual Ts/Tv models. The other averaged estimators performed consistent with the nature of the values being used.

Note that the average over the non-Ts/Tv models weight the $\hat{\kappa}$ results in the subset significantly higher than those that do so poorly (i.e., the estimate tends toward the true value of κ). Therefore, although the non-Ts/Tv models are estimating κ poorly (as seen in the black dots from the figures), the values of $\hat{\kappa}$ from this subset of models that are closest to the true value of κ are being weighted significantly higher than the other models through the information-loss weights.

Regarding the best-fit model estimate, it tends to perform well across sequence divergence and sequence length and empirically outperforms the averaged estimators.

Figures 2.2 and 2.3 show the Ts/Tv models are over-estimating κ for small branch lengths, with this apparent bias decreasing inversely with the branch lengths increasing. Additionally, the full set of figures found in Appendix A (A.1 through A.12) illustrate the over-estimation of κ found in the shorter sequences with this decreasing for increases of sequence length and divergence. Note, however, that the extended sequence lengths of E(To) = 1.72 to 4.128 generally show the estimators becoming unstable and are therefore printed separately on different scales (figures A.2, A.5, A.8, and A.11)

Table 2.5 and the various boxplots of Appendix A summarize the results from the dotplots for the model averaged estimates and named models. These are included to provide added precision for the results and quantify the comparisons between the different estimates of κ.

These HKY85 simulation results illustrate that the named models of Section 1.2.3 have relatively good performance in the point estimation of κ. Consistent with noted characteristics of Chapter 1, this reliability is due to the biological considerations of these models that are a result by simulated design as well as the structure of the mathematical estimation of κ (Section 1.4.1). While the grouping of $\hat{\kappa}$ for biological model results was expected, we continued to explore various simulated nucleotide substitution settings in order to better understand more complicated situations.

TN93 Data Analyses

Table 2.7 lists the TN93 simulation settings that were used to produce the analyses in this section. Similar to the analyses of the previous section, those included below utilize Case I from table 2.1. The analyses were run on sequence lengths of 250 (figures A.13 to A.15), 500 (figures 2.4 and A.16 to A.18), 1000 (figures A.19 to A.21), and 3000 (figures 2.5 and A.22 to A.24) nucleotide sites.

Table 2.7: TN93 Simulation Settings

Relative Substitution Rates										
θ_{AC}	θ_{AG}	θ_{AT}	θ_{CG}	θ_{CT}	θ_{GT}	E(Ts/Tv)	t	E(Ts)	E(Tv)	E(To)
0.075	0.083	0.075	0.075	0.25	0.075	1.57	2	0.057	0.072	0.129
							3	0.085	0.108	0.193
							6	0.170	0.216	0.386
							9	0.255	0.324	0.579
							12	0.340	0.432	0.772
							20	0.566	0.720	1.286
							25	0.707	0.900	1.607
							30	0.849	1.080	1.928
							45	1.273	1.620	2.893
							60	1.698	2.160	3.858

θ_{ij} = the instantaneous substitution rate of nucleotide i to j; t = total time; Ts = transition;
Tv = transversion; To = total substitutions (substitutions per site)

Table 2.8: Estimated Ts/Tv Comparisons for TN93 Data

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
1.572	0.129	250	1.640	1.659	1.653	1.558	1.468	1.653	1.413
		500	1.600	1.544	1.614	1.563	1.507	1.614	1.472
		1000	1.575	1.589	1.589	1.563	1.541	1.589	1.521
		3000	1.569	1.582	1.582	1.562	1.552	1.582	1.536
	0.193	250	1.599	1.625	1.620	1.543	1.473	1.619	1.432
		500	1.570	1.592	1.590	1.548	1.514	1.590	1.489
		1000	1.567	1.588	1.586	1.558	1.545	1.587	1.526
		3000	1.555	1.575	1.575	1.558	1.546	1.575	1.531
	0.386	250	1.569	1.618	1.614	1.579	1.534	1.614	1.505
		500	1.548	1.593	1.590	1.571	1.551	1.591	1.533
		1000	1.538	1.579	1.577	1.559	1.547	1.578	1.532
		3000	1.538	1.578	1.577	1.560	1.550	1.577	1.533
	0.579	250	1.546	1.615	1.611	1.578	1.546	1.611	1.524
		500	1.515	1.577	1.575	1.556	1.543	1.576	1.527
		1000	1.524	1.585	1.583	1.566	1.553	1.584	1.536
		3000	1.516	1.574	1.574	1.559	1.549	1.574	1.532
	0.772	250	1.532	1.623	1.619	1.588	1.566	1.619	1.545
		500	1.501	1.584	1.581	1.565	1.552	1.582	1.538
		1000	1.504	1.582	1.581	1.564	1.553	1.581	1.538
		3000	1.499	1.576	1.576	1.562	1.552	1.576	1.537
	1.286	250	1.464	2.072	1.606	1.589	1.560	1.606	1.544
		500	1.454	1.586	1.584	1.566	1.555	1.585	1.541
		1000	1.457	1.585	1.583	1.567	1.558	1.584	1.544
		3000	1.454	1.576	1.576	1.563	1.554	1.576	1.538

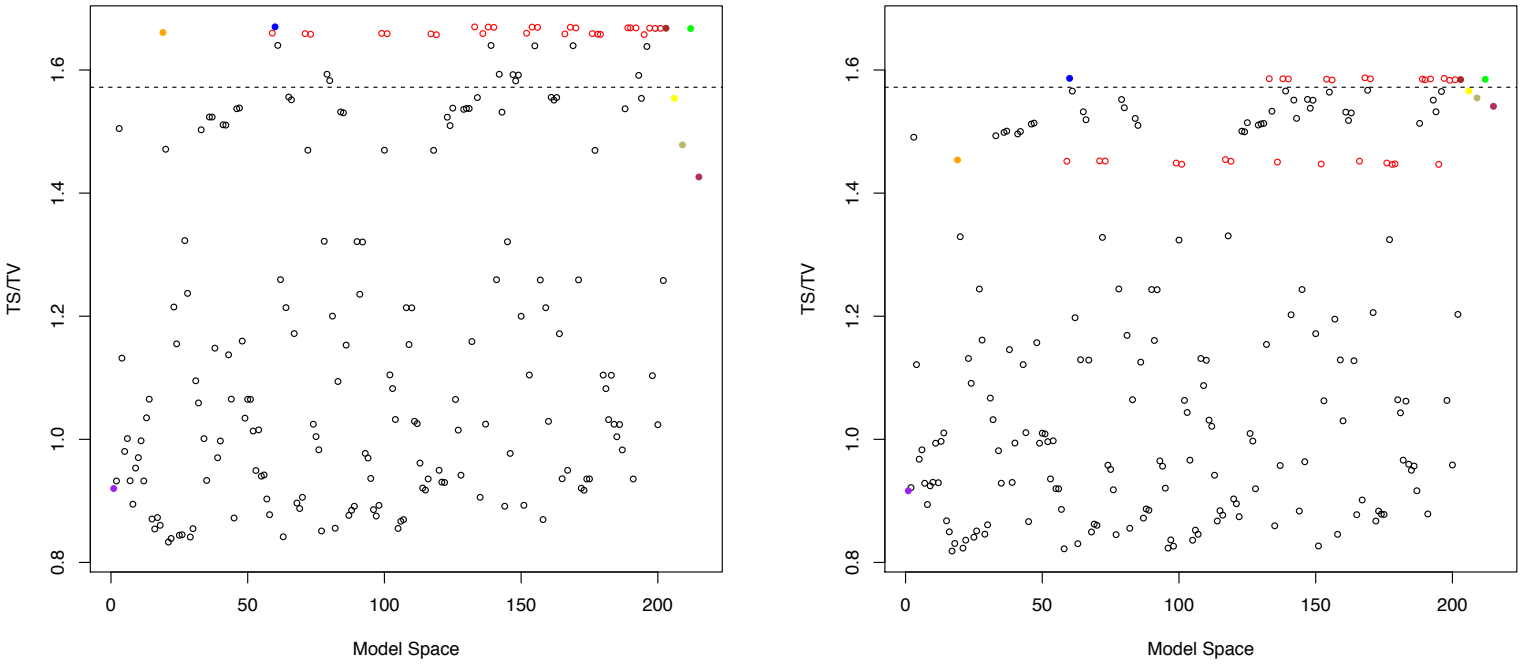
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Note: Substitution rates are defined as $\theta_0 = 0.075$, $\theta_1 = 0.0831$, $\theta_2 = 0.075$, $\theta_3 = 0.075$, $\theta_4 = 0.25$, $\theta_5 = 0.075$

Table 2.9: Estimated Ts/Tv Comparisons for TN93 Data for E(To) = 1.607 to 3.858

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
1.572	1.607	250	1.441	1.614	12.559	1.580	2.220	2.782	2.130
		500	1.438	1.599	1.596	1.578	1.566	1.597	1.553
		1000	1.429	1.579	1.577	1.560	1.551	1.577	1.537
		3000	1.428	1.576	1.576	1.565	1.555	1.576	1.540
	1.928	250	1.434	1.668	105.477	3.704	11.997	16.600	11.174
		500	1.408	1.592	47.354	1.567	2.263	2.527	2.203
		1000	1.411	1.588	5.036	1.568	1.562	1.589	1.548
		3000	1.405	1.578	1.578	1.566	1.556	1.578	1.543
	2.893	250	1.467	3.840	129.311	40.287	228.830	211.114	235.386
		500	1.375	2.491	109.357	6.187	170.103	143.729	180.332
		1000	1.361	1.657	53.055	4.842	72.485	55.714	79.393
		3000	1.355	1.585	5.446	1.572	3.935	2.989	4.989
	3.858	250	4.883	7.097	310.113	77.405	360.222	409.764	350.514
		500	1.934	5.596	290.708	62.357	435.149	410.048	446.034
		1000	1.331	3.825	347.384	67.949	444.633	379.363	466.217
		3000	1.336	1.982	360.741	27.833	259.064	200.611	284.632

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Note: Substitution rates are defined as $\theta_0 = 0.075$, $\theta_1 = 0.0831$, $\theta_2 = 0.075$, $\theta_3 = 0.075$, $\theta_4 = 0.25$, $\theta_5 = 0.075$

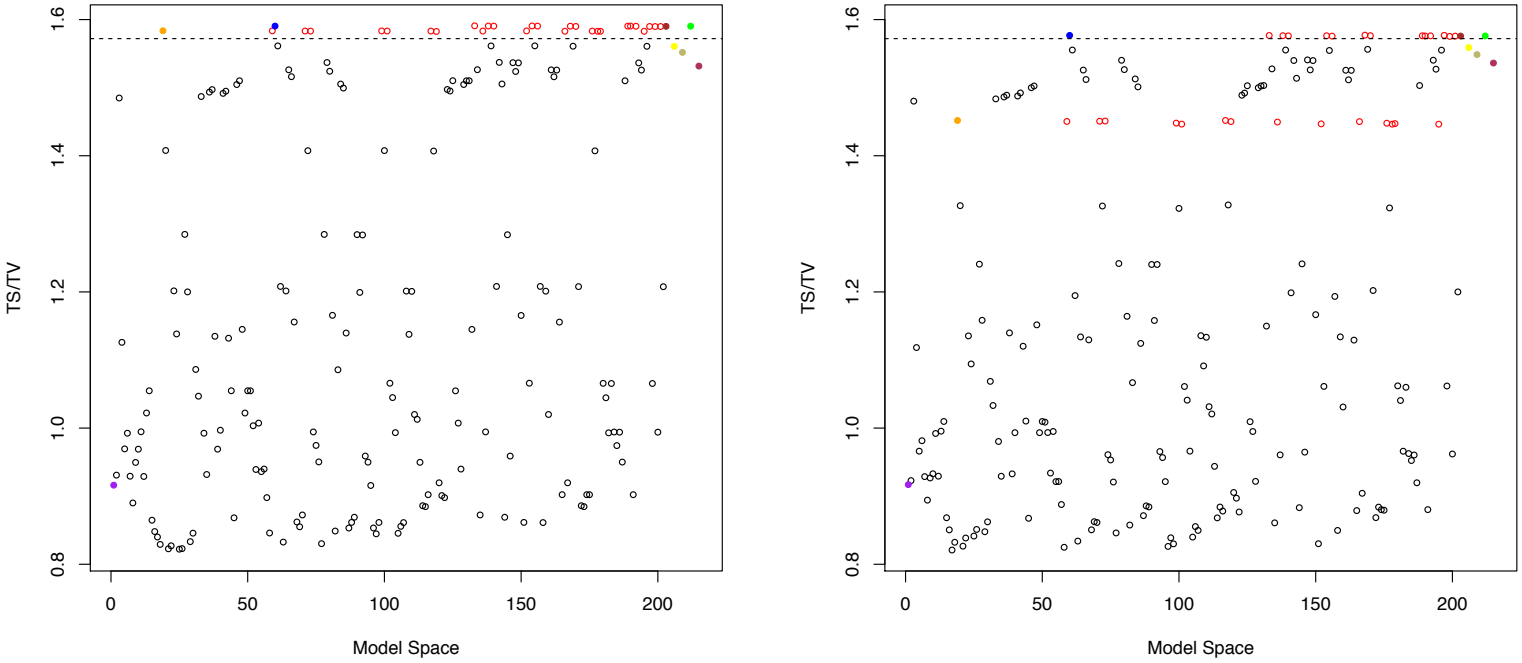
Figure 2.4: Ts/Tv Estimates on TN93 Data with Sequence Length 500



E(To) = 0.064 **1.290**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure 2.5: Ts/Tv Estimates on TN93 Data with Sequence Length 3000



E(To) = 0.064 **1.290**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figures A.13 A.15 as well as those illustrating the extended sequence divergence scale (E[To]= 1.607 to 3.858) in A.14, A.17, A.20, and A.23 show issues in the scale of the estimates due to higher sequence divergence resulting in outliers that skewed the shared scale (see E[To]= 1.29) as well as those without the shared scale in the extended divergence figures.

The simulated data was generated with one of the transition-related substitution parameters ($\theta_1 = 0.0831$) being similar to the transversion-related substitution parameters ($\theta_0 = \theta_2 = \theta_3 = \theta_5 = 0.075$), and the other transition-related substitution parameter being much larger ($\theta_4 = 0.25$). This difference was designed to confound a subset of the biologically-based Ts/Tv models with with the complement to that subset. Figures A.13, A.16, A.19, and A.22 are consistent with this design choice as they illustrate the Ts/Tv models performing relatively in-line with the true $\kappa = 1.289$. However, as the branch lengths increase, the Ts/Tv models appear to separate into two subsets with one converging toward the truth and the other underestimating κ (also noted in figures 2.4 and 2.5).

The behavior of the Ts/Tv models is consistent with models that set $\theta_1 = \theta_4$ underestimating the numerator of κ because θ_1 is less than θ_2 by three orders of magnitude. Therefore, the Ts/Tv models with $\theta_1 \neq \theta_2$ are the identified models in-line with the true value of κ. This is most obvious in the the plot of E(S)=1.290 with the HKY85 (orange) model biased due to the differences in the numerator values of $\hat{\kappa}$, and TN93 (blue) falling in the set of models that distinctly estimate θ_1 and θ_4 and results in a more robust estimate of κ.

Other models confounding the numerator and denominator values of $\hat{\kappa}$ have a relative spread across a range of zero to κ. Additionally, similar with the empirical appearance of two subsets within the Ts/Tv models, several distinct subsets within the non-Ts/Tv models are apparent as the divergence between sequences increases.

Note that in this simulation, all of the model average estimators perform relatively well and converge to the truth as the branch lengths increase. Particularly noteworthy is the convergence performance of the non-Ts/Tv model averaged estimate as the sequence divergence increases. Similar to the observed behavior in the HKY85 simulated data, this implies that

models estimating κ closer to the truth are being weighted much higher in-terms of model fit.

This simulation illustrates how nucleotide substitution models may become confounded between ones that are biologically significant and ones that are not. While all of the model averaged estimates are relatively robust to misspecification (i.e., non-Ts/Tv model averaged estimate), the Ts/Tv model averaged estimates remain the best performing.

GTR Data Analyses

Table 2.10 shows the simulation settings used to generate the data according to the GTR model. Similar to all of the other results in this section, the analyses listed here use the divergence settings from Case I of table 2.1. The analyses are based on data with sequence lengths of 250 (figures A.25 to A.27), 500 (figures 2.6 and A.28 to A.30), 1000 (figures A.31 to A.33), and 3000 (figures 2.7 and A.34 to A.36) nucleotide sites.

Table 2.10: GTR Simulation Settings

Relative Substitution Rates										
θ_{AC}	θ_{AG}	θ_{AT}	θ_{CG}	θ_{CT}	θ_{GT}	E(Ts/Tv)	t	E(Ts)	E(Tv)	E(To)
0.07	0.08	0.14	0.05	0.195	0.1	1.29	2	0.049	0.076	0.125
							3	0.074	0.114	0.188
							6	0.147	0.228	0.375
							9	0.221	0.342	0.563
							12	0.294	0.456	0.750
							20	0.490	0.760	1.250
							25	0.613	0.950	1.563
							30	0.735	1.140	1.875
							45	1.103	1.710	2.813
							60	1.470	2.280	3.750

θ_{ij} = the instantaneous substitution rate of nucleotide i to j; t = time; Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)

Table 2.11: Estimated Ts/Tv Comparisons for GTR Data

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
1.289	0.125	250	1.340	1.353	1.343	1.273	1.231	1.342	1.208
		500	1.321	1.333	1.327	1.262	1.242	1.326	1.227
		1000	1.309	1.320	1.315	1.264	1.248	1.315	1.237
		3000	1.293	1.303	1.298	1.261	1.246	1.298	1.232
	0.188	250	1.356	1.376	1.365	1.303	1.263	1.364	1.242
		500	1.316	1.332	1.323	1.262	1.247	1.324	1.234
		1000	1.284	1.300	1.292	1.242	1.232	1.292	1.222
		3000	1.286	1.301	1.293	1.275	1.256	1.294	1.242
	0.375	250	1.281	1.317	1.298	1.232	1.224	1.298	1.212
		500	1.295	1.327	1.311	1.258	1.247	1.313	1.236
		1000	1.272	1.304	1.288	1.252	1.236	1.290	1.225
		3000	1.275	1.304	1.289	1.279	1.268	1.290	1.256
	0.563	250	1.274	1.327	1.300	1.248	1.231	1.301	1.220
		500	1.278	1.324	1.301	1.253	1.240	1.303	1.228
		1000	1.271	1.316	1.294	1.264	1.248	1.296	1.233
		3000	1.265	1.309	1.288	1.281	1.273	1.288	1.262
	0.750	250	1.273	1.336	1.305	1.252	1.238	1.306	1.227
		500	1.270	1.331	1.302	1.261	1.244	1.305	1.232
		1000	1.263	1.322	1.293	1.267	1.249	1.295	1.235
		3000	1.262	1.321	1.293	1.290	1.282	1.294	1.272
	1.250	250	1.254	1.365	1.312	1.251	1.241	1.316	1.229
		500	1.252	1.355	1.307	1.264	1.250	1.312	1.235
		1000	1.245	1.342	1.297	1.272	1.257	1.301	1.241
		3000	1.243	1.338	1.293	1.292	1.284	1.294	1.275

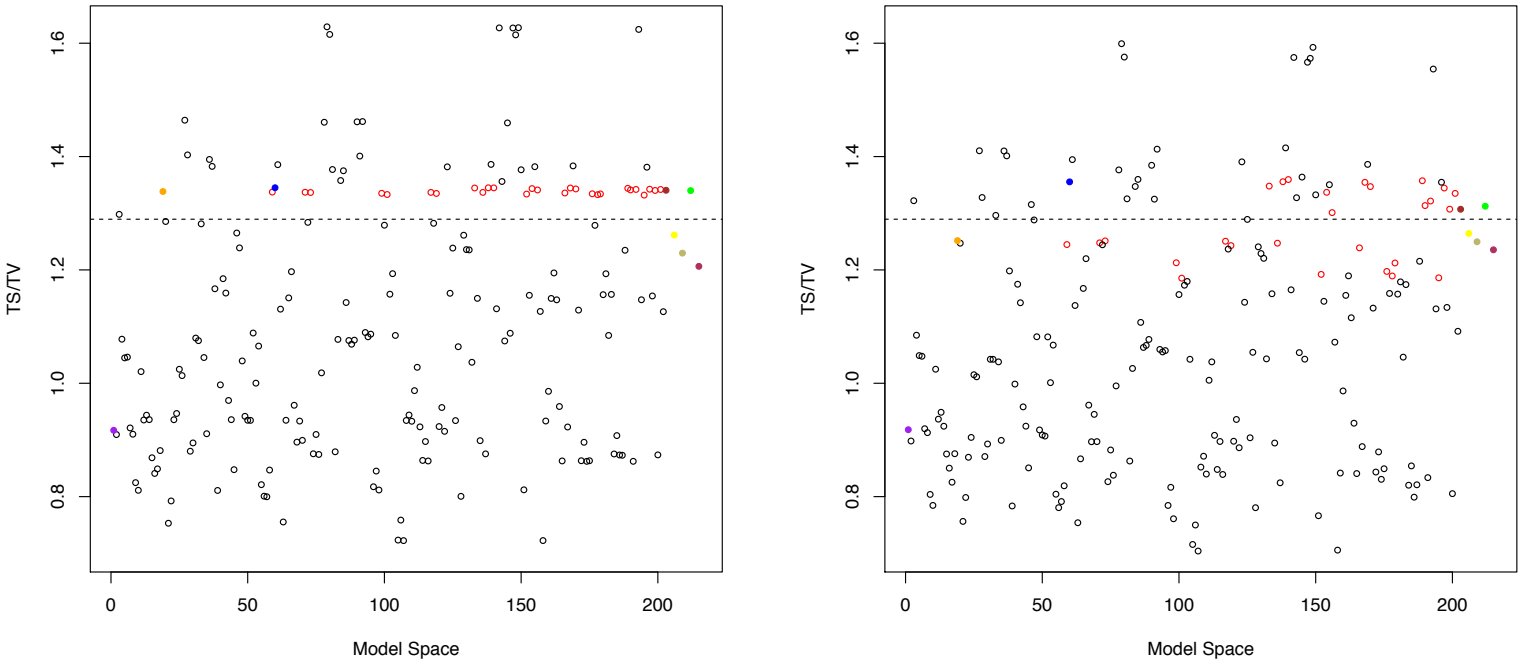
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$. Substitution rates are defined as $\theta_0 = 0.07$, $\theta_1 = 0.08$, $\theta_2 = 0.14$, $\theta_3 = 0.05$, $\theta_4 = 0.195$, $\theta_5 = 0.1$

Table 2.12: Estimated Ts/Tv Comparisons for GTR Data for E(To) = 1.289 to 3.75

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
1.289	1.563	250	1.246	1.382	7.509	1.259	1.287	1.404	1.266
		500	1.238	1.364	1.304	1.255	1.243	1.312	1.229
		1000	1.229	1.350	1.293	1.268	1.252	1.299	1.236
		3000	1.229	1.347	1.290	1.287	1.280	1.292	1.269
	1.875	250	1.241	1.428	51.550	1.284	5.171	6.514	5.040
		500	1.228	1.377	18.200	1.257	1.288	1.464	1.259
		1000	1.227	1.374	1.306	1.277	1.260	1.314	1.243
		3000	1.214	1.356	1.290	1.285	1.277	1.293	1.266
	2.813	250	1.216	2.361	40.296	20.914	56.747	67.819	56.105
		500	1.203	1.623	7.656	1.288	14.453	15.654	14.557
		1000	1.195	1.437	1.326	1.277	1.519	1.646	1.512
		3000	1.188	1.384	1.297	1.286	1.273	1.306	1.254
	3.750	250	2.984	6.589	211.613	62.952	188.495	250.194	182.216
		500	1.270	4.026	158.056	41.199	134.599	170.431	131.387
		1000	1.172	2.210	70.388	8.008	44.360	49.840	44.913
		3000	1.166	1.523	8.537	1.309	1.579	1.618	1.62

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Substitution rates are defined as $\theta_0 = 0.07$, $\theta_1 = 0.08$, $\theta_2 = 0.14$, $\theta_3 = 0.05$, $\theta_4 = 0.195$, $\theta_5 = 0.1$

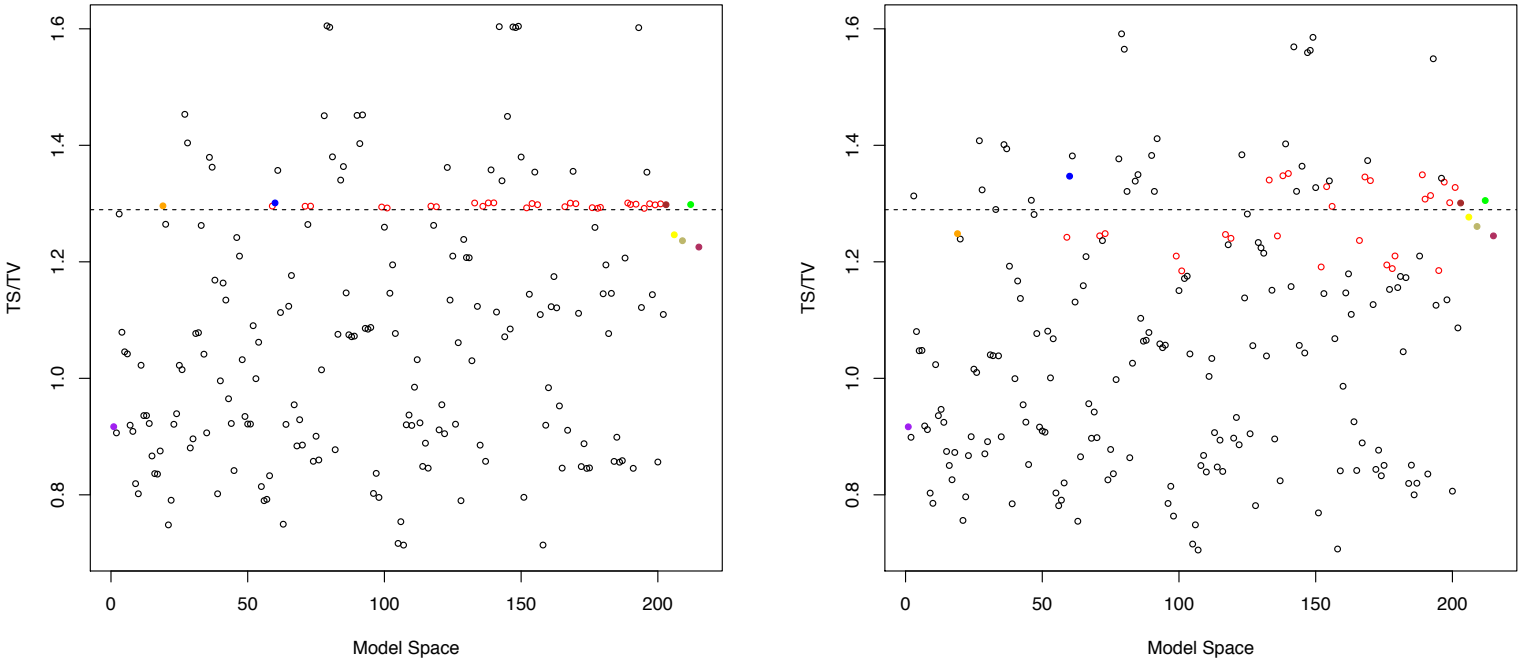
Figure 2.6: Ts/Tv Estimates on GTR Data with Sequence Length 500



E(To) = 0.063 **1.25**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure 2.7: Ts/Tv Estimates on GTR Data with Sequence Length 3000



Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

As with figures A.13 and A.15, the GTR simulations experienced similar issues (figure A.25) due to outlier estimates occurring in sequences with higher relative divergence. These issues resulted in a skewed scale that is shared across the figures (e.g. E[To]= 1.25). However, unlike the TN93 case, figure A.27 was not affected as the outliers in figure A.25 occurred in substitution models that did not directly impact the named model estimates in the boxplots. Additionally, the estimates on the extended scale of sequence divergence (figures A.26, A.29, A.32, and A.35) show the consistent issues with sequence divergence discussed for the HKY85 and TN93 cases.

Performance of the GTR estimates falls in-line with what was identified in the simpler data simulations displayed earlier. Consistent with the design of the simulations, the overall distribution of κ estimates was not as empirically clustered for the Ts/Tv and non-Ts/Tv models as in those earlier cases. However, the Ts/Tv model averaged estimates performed in a robust manner and were empirically better than the others with respect to the distance of the point estimates from the true value of κ.

The behavior of the κ estimates was informative regarding overall performance across the space of reversible models. The clustering of the 30 Ts/Tv models was particularly important as this subset of models was robust to the different changes in simulation settings and provides evidence that the proof of concept in clustering synonymous and non-synonymous models separately in Chapter 3 may add efficiency to the model selection procedure.

Based on the $\hat{\kappa}$ values estimated across sequence divergence, phylogenetic trees, substitution parameters, and sequence length, we noted that the κ estimates closest to the true value of κ were weighted highest through information loss. This characteristic was best displayed in the non-Ts/Tv model averaged estimates relative to the domain of κ values across that subset of models. With regard to distance from the true value of κ, for data simulated based on the biological characteristics of transition and transversion substitutions, the Ts/Tv model averaged estimator appeared to empirically perform the best across the variations of the substitution parameters, underlying model generating the data, sequence divergence, and sequence length. Note that although Cases II and III from table 2.1 are not shown here, performance in those

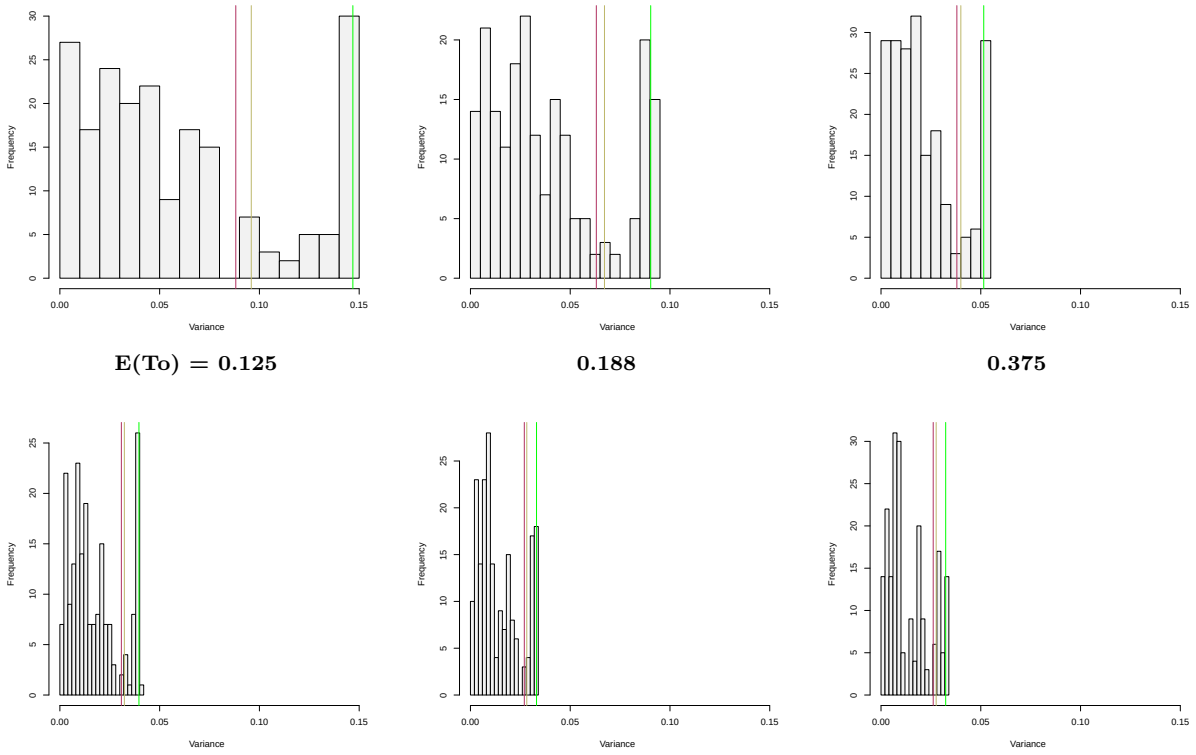
cases is analogous to the Case I results.

2.2.2 Variance - Transition/Transversion Ratios

This section provides empirical analyses for the variance of the different model estimates across the different data simulation settings described in Section 2.2.1 and tables 2.4, 2.7, and 2.10. Most results are located in the appendix, split by the named models the data was simulated under: HKY85 (figures A.37 to A.52), TN93 (figures A.53 to **??**), and GTR (figures **??** to **??**). Each section displays variance estimates in histograms with a shared scale as well as the extended sequence divergence ones that do not share the same scale. Corresponding boxplots are included to illustrate the clustering of the models in the bimodal distribution that we observed throughout.

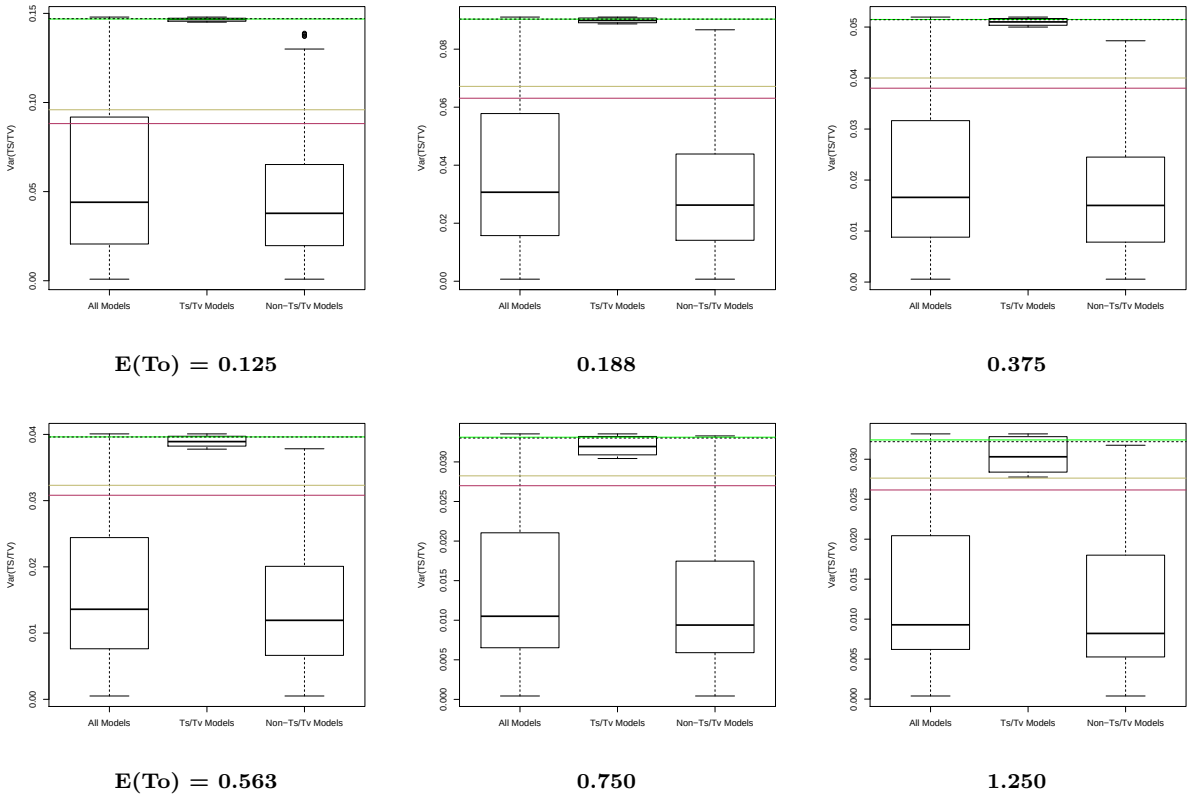
Each of the histograms show a discreet summary of the distribution of estimates with three vertical lines across the figure representing the variance for the three primary model averaged estimators according to equation 1.27. The colors of the lines correspond to those shown in the preceding section with the dark khaki representing the overall model averaged estimate, the green for the Ts/Tv model averaged estimate, and maroon for the non-Ts/Tv model averaged estimate. The primary purpose of the boxplots is to show additional information on the distribution of the variance estimates, broken out by the Ts/Tv and non-Ts/Tv models. Finally, note that the results based on the extended scale of sequence divergence per Section 3.1 do not use a shared scale due to the instability of the estimates for various settings of sequence divergence and sequence length.

Figure 2.8: Variance Histograms on GTR Data with Sequence Length 500



To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure 2.9: Variance Boxplots on GTR Data with Sequence Length 500 and E(To) = 0.125 to 1.250



Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

The two figures included in this section (2.8 and 2.9) illustrate the consistent message we noticed in the results of our analysis on variance estimates. The variance estimates tend to empirically cluster between the Ts/Tv models and non-Ts/Tv models. Essentially, the distribution of $V(\hat{\kappa})$ is bimodal and also clustered with the variance corresponding to the Ts/Tv models being significantly greater than that of the non-Ts/Tv models (see figures 2.9, A.39, A.43, A.47, A.51, A.55, **??**, **??**, **??**, **??**, **??**, and **??**).

This result suggests there may be a tradeoff between consistency and variability, with the Ts/Tv model estimates converging robustly to true values of κ but also having the highest variance. Further research may benefit the identification of shrinkage estimates through weighting schemes that incorporate the higher bias/lower variance of the non-Ts/Tv models in order to reduce the overall mean square error.

2.2.3 Euclidean Distance Comparisons

The following section displays tabular summaries of Euclidean distance (L2 Norm) based on the difference in the vector of substitution parameter estimates and the true values of $\theta_{AC}, \theta_{AG}, \theta_{AT}, \theta_{CG}, \theta_{CT}, and \theta_{GT}$.

Our analyses were run with the sequence divergence specified in Case I of table 2.1 and the extended sequence divergence of table 2.2. The simulation settings are those that have already been noted in tables 2.4, 2.7, and 2.10. The results from the HKY85 data is shown in tables 2.13 2.14, TN93 is summarized in tables 2.15 and 2.16, and GTR in tables 2.17 and 2.18.

Table 2.13: Estimated L2 Norm Comparisons for HKY85 Data for E(To) = 0.138 to 1.376

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
2.095	0.138	250	3.695	3.694	3.693	3.696	3.568	3.564	3.570
		500	3.694	3.694	3.694	3.695	3.567	3.564	3.569
		1000	3.694	3.694	3.694	3.695	3.565	3.564	3.568
		3000	3.694	3.694	3.694	3.694	3.564	3.564	3.567
	0.206	250	3.639	3.637	3.636	3.639	3.514	3.509	3.518
		500	3.641	3.640	3.640	3.641	3.515	3.512	3.519
		1000	3.642	3.641	3.642	3.642	3.514	3.513	3.518
		3000	3.642	3.641	3.641	3.642	3.513	3.513	3.517
	0.413	250	3.478	3.476	3.475	3.478	3.359	3.352	3.367
		500	3.481	3.479	3.479	3.480	3.359	3.356	3.367
		1000	3.481	3.480	3.480	3.481	3.358	3.358	3.366
		3000	3.482	3.481	3.481	3.481	3.359	3.359	3.366
	0.619	250	3.319	3.315	3.314	3.319	3.204	3.197	3.215
		500	3.321	3.319	3.318	3.319	3.203	3.201	3.214
		1000	3.320	3.319	3.319	3.319	3.202	3.201	3.212
		3000	3.322	3.321	3.321	3.321	3.204	3.204	3.214
	0.826	250	3.157	3.152	3.150	3.155	3.047	3.039	3.061
		500	3.158	3.155	3.155	3.156	3.045	3.044	3.060
		1000	3.163	3.161	3.161	3.161	3.049	3.049	3.063
		3000	3.162	3.162	3.162	3.162	3.050	3.050	3.065
	1.376	250	2.719	2.711	2.709	2.716	2.618	2.609	2.641
		500	2.728	2.724	2.724	2.726	2.626	2.625	2.650
		1000	2.732	2.730	2.730	2.731	2.633	2.633	2.656
		3000	2.735	2.734	2.734	2.734	2.637	2.637	2.662

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Note: Substitution rates are defined as $\theta_0 = 0.07$, $\theta_1 = 0.16$, $\theta_2 = 0.07$, $\theta_3 = 0.07$, $\theta_4 = 0.16$, $\theta_5 = 0.07$

Table 2.14: Estimated L2 Norm Comparisons for HKY85 Data for E(To) = 1.72 to 4.128

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
2.095	1.720	250	2.445	2.145	2.436	2.445	2.350	2.338	2.379
		500	2.460	2.457	2.458	2.461	2.369	2.367	2.399
		1000	3.0530	2.2517	2.517	2.517	2.436	2.435	2.436
		3000	2.468	2.467	2.468	2.468	2.380	2.380	2.415
	2.064	250	2.137	2.159	2.179	2.187	1.729	1.837	1.64
		500	2.187	2.183	2.188	2.189	2.099	2.097	2.133
		1000	2.199	2.197	2.199	2.199	2.116	2.116	2.152
		3000	2.203	2.202	2.202	2.203	2.123	2.123	2.168
	3.096	250	1.577	2.935	349.844	29.450	121.663	82.744	184.436
		500	1.358	1.482	51.934	1.432	17.887	3.732	73.455
		1000	1.376	1.380	21.931	1.402	1.094	1.202	21.832
		3000	1.395	1.395	1.403	1.399	1.339	1.339	1.339
	4.128	250	9.760	39.950	3443.386	245.384	1347.672	1484.72	1376.77
		500	2.052	9.677	1298.069	134.432	604.506	469.910	822.495
		1000	0.687	1.852	528.195	35.741	273.78	154.195	670.918
		3000	2.118	0.649	117.63	0.673	12.657	3.910	415.626

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Note: Substitution rates are defined as $\theta_0 = 0.07$, $\theta_1 = 0.16$, $\theta_2 = 0.07$, $\theta_3 = 0.07$, $\theta_4 = 0.16$, $\theta_5 = 0.07$

Table 2.15: Estimated L2 Norm Comparisons for TN93 Data for E(To) = 0.129 to 1.286

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
1.572	0.129	250	3.956	3.919	3.918	3.923	3.807	3.803	3.807
		500	2.956	3.920	3.920	3.921	3.803	3.801	3.803
		1000	3.956	3.920	3.920	3.920	3.799	3.799	3.799
		3000	3.956	3.920	3.920	3.920	3.798	3.798	3.798
	0.193	250	3.914	3.858	3.858	3.863	3.749	3.746	3.750
		500	3.914	3.859	3.859	3.860	3.743	3.742	3.743
		1000	3.914	3.861	3.860	3.860	3.741	3.741	3.928
		3000	3.914	3.859	3.859	3.859	3.739	3.739	3.739
	0.386	250	3.786	3.666	3.665	3.668	3.560	3.556	3.561
		500	3.787	3.673	3.673	3.673	3.560	3.559	3.560
		1000	3.788	3.676	3.676	3.676	3.562	3.562	3.562
		3000	3.818	3.731	3.731	3.730	3.617	3.618	3.617
	0.579	250	3.662	3.481	3.480	3.483	3.379	3.375	3.380
		500	3.665	3.493	3.493	3.493	3.384	3.384	3.384
		1000	3.665	3.493	3.493	3.493	3.384	3.384	3.384
		3000	3.665	3.496	3.496	3.496	3.387	3.387	3.387
	0.772	250	3.537	3.286	3.285	3.288	3.188	3.183	3.188
		500	3.543	3.305	3.303	3.304	3.200	3.199	3.200
		1000	3.544	3.310	3.310	3.310	3.206	3.206	3.206
		3000	3.545	3.314	3.313	3.313	3.210	3.210	3.210
	1.286	250	3.227	3.199	3.782	2.785	2.696	2.690	2.697
		500	3.233	2.810	2.810	2.810	2.810	2.719	2.718
		1000	3.234	2.815	2.814	2.814	2.724	2.724	2.724
		3000	3.235	2.827	2.827	2.827	2.739	2.739	2.738

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Note: Substitution rates are defined as $\theta_0 = 0.075$, $\theta_1 = 0.0831$, $\theta_2 = 0.075$, $\theta_3 = 0.075$, $\theta_4 = 0.25$, $\theta_5 = 0.075$

Table 2.16: Estimated L2 Norm Comparisons for TN93 Data for E(To) = 1.607 to 3.858

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
1.572	1.607	250	3.041	2.476	21.787	2.485	1.525	1.280	1.581
		500	3.049	2.499	2.500	2.501	2.414	2.412	2.414
		1000	3.050	2.517	2.517	2.517	2.436	2.435	2.436
		3000	2.697	2.474	2.475	2.475	2.395	2.392	2.397
	1.928	250	2.854	2.151	191.861	5.342	18.479	27.050	17.021
		500	2.871	2.196	81.612	2.202	1.220	1.119	1.258
		1000	2.871	2.201	7.752	2.202	2.124	2.122	2.124
		3000	2.875	2.213	2.213	2.213	2.142	2.142	2.142
	2.893	250	2.458	8.122	381.728	113.438	646.852	600.389	664.662
		500	2.379	3.557	314.237	14.734	465.339	392.450	493.002
		1000	2.389	1.380	143.814	10.356	196.129	149.717	215.029
		3000	2.396	1.292	12.523	1.297	5.652	2.975	8.538
	3.858	250	9.061	29.897	11520.600	335.368	1715.108	2301.456	1650.177
		500	2.951	19.019	1150.750	224.865	1649.825	1565.909	1689.059
		1000	2.014	9.520	1294.135	242.459	1638.466	1398.306	1717.015
		3000	2.017	1.914	1308.037	94.01	931.197	719.761	1022.474

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Note: Substitution rates are defined as $\theta_0 = 0.075$, $\theta_1 = 0.0831$, $\theta_2 = 0.075$, $\theta_3 = 0.075$, $\theta_4 = 0.25$, $\theta_5 = 0.075$

Table 2.17: Estimated L2 Norm Comparisons for GTR Data for E(To) = 0.125 to 1.250

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
1.289	0.125	250	3.994	3.974	3.961	3.965	3.849	3.848	3.849
		500	3.994	3.973	3.961	3.964	3.846	3.846	3.846
		1000	3.993	3.973	3.960	3.962	3.843	3.842	3.843
		3000	3.994	3.974	3.962	3.963	3.843	3.841	3.843
	0.188	250	3.952	3.919	3.900	3.906	3.792	3.792	3.792
		500	3.952	3.922	3.904	3.907	3.792	3.791	3.792
		1000	3.953	3.922	3.904	3.906	3.789	3.787	3.789
		3000	3.953	3.923	3.905	3.906	3.787	3.786	3.788
	0.375	250	3.830	3.765	3.725	3.733	3.627	3.626	3.628
		500	3.830	3.767	3.729	3.735	3.624	3.621	3.624
		1000	3.831	3.769	3.732	3.735	3.624	3.619	3.625
		3000	3.832	3.771	3.734	3.735	3.623	3.621	3.624
	0.563	250	3.711	3.610	3.549	3.561	3.460	3.457	3.460
		500	3.710	3.616	3.558	3.566	3.460	3.454	3.461
		1000	3.712	3.616	3.560	3.564	3.458	3.452	3.460
		3000	3.712	3.618	3.563	3.564	3.457	3.455	3.459
	0.750	250	3.590	3.460	3.381	3.394	3.300	3.296	3.301
		500	3.593	3.461	3.384	3.395	3.294	3.286	3.296
		1000	3.593	3.464	3.387	3.393	3.292	3.285	3.295
		3000	3.594	3.464	3.388	3.390	3.288	3.285	3.291
	1.250	250	3.281	3.038	2.900	2.926	2.840	2.828	2.842
		500	3.285	3.047	2.914	2.930	2.844	2.828	2.848
		1000	3.290	3.060	2.931	2.941	2.854	2.843	2.859
		3000	3.290	3.062	2.933	2.937	2.849	2.845	2.855

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Substitution rates are defined as $\theta_0 = 0.07$, $\theta_1 = 0.08$, $\theta_2 = 0.14$, $\theta_3 = 0.05$, $\theta_4 = 0.195$, $\theta_5 = 0.1$

Table 2.18: Estimated L2 Norm Comparisons for GTR Data for E(To) = 1.289 to 3.75

E(Ts/Tv)	E(To)	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
1.289	1.563	250	3.093	2.779	14.653	2.645	2.506	2.442	2.520
		500	3.103	2.798	2.627	2.650	2.570	2.548	2.576
		1000	3.107	2.809	2.644	2.657	2.579	2.564	2.585
		3000	3.108	2.813	2.647	2.652	2.574	2.567	2.581
	1.875	250	2.913	2.567	97.438	2.395	5.849	8.784	5.537
		500	2.924	2.549	34.625	2.370	2.221	2.040	2.250
		1000	2.929	2.555	2.347	2.366	2.293	2.271	2.301
		3000	2.933	2.562	2.358	2.366	2.295	2.286	2.303
	2.813	250	2.414	4.412	282.746	81.72	176.401	207.841	175.267
		500	2.421	2.415	61.536	1.702	37.636	41.356	37.896
		1000	2.432	1.921	1.507	1.554	0.926	0.981	0.911
		3000	2.441	1.859	1.501	1.528	1.459	1.436	1.474
	3.750	250	4.911	293.461	2634.84	265.294	918.041	1200.075	910.333
		500	2.107	10.755	954.934	206.994	548.822	683.746	539.370
		1000	2.006	4.212	389.709	26.402	167.56	188.739	169.741
		3000	2.024	1.740	27.438	1.073	0.851	0.922	1.01

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The stationary distribution is defined as $\pi_A = 0.4$, $\pi_C = 0.3$, $\pi_G = 0.2$, and $\pi_T = 0.1$.
Substitution rates are defined as $\theta_0 = 0.07$, $\theta_1 = 0.08$, $\theta_2 = 0.14$, $\theta_3 = 0.05$, $\theta_4 = 0.195$, $\theta_5 = 0.1$

Among the estimates, the model average over all models and only the Ts/Tv models perform similarly and result in the smallest distance from the truth. The non-Ts/Tv models are slightly larger followed by the best models and with the named models being the largest. Interestingly, the model averaged methods recover the vector values of the truth better than the true models used in the generation of the data itself.

Based on the summaries, the model averaged estimates' L2 Norms are uniformly smaller than the named and best models. Even the average over the non-Ts/Tv models outperformed the named models in terms of these distances. This implies that the averaged estimates are better able to recover the location in Euclidean space of the vector of substitution parameters. In addition to the direct use of these results, this may also be a useful tool in sensitivity analyses of model selection and estimation methods for future research.

2.2.4 Empirical Results

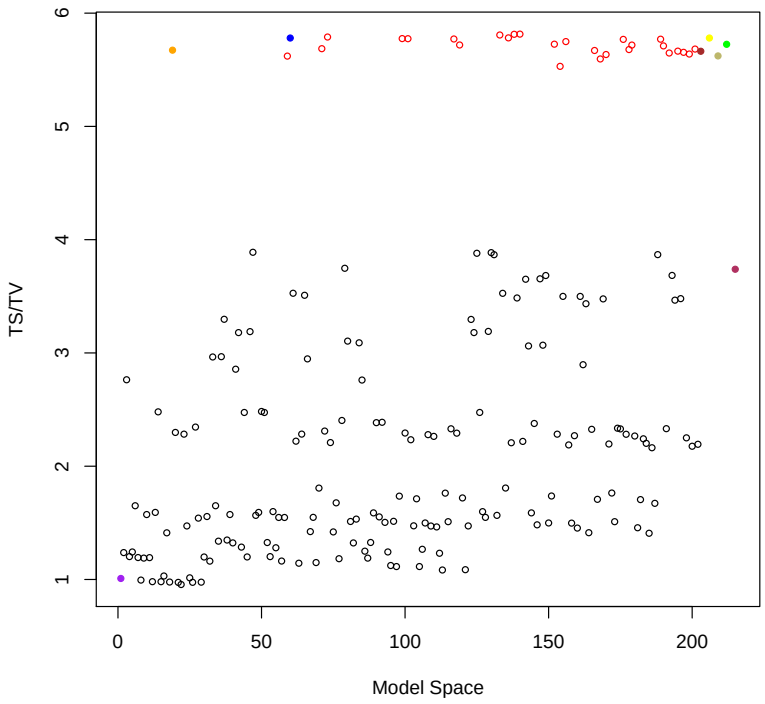
We conducted limited analyses on empirical data. The following figures and tables represent phylogenetic analyses for a GenBank Rat-Mouse sequence alignment (1942 sites at `http://www.ncbi.nlm.nih.gov/genbank/`), a publicly available human immunodeficiency virus (HIV) database that is part of the HyPhy software package (13 taxa on 273 sites at `http://hyphy.org/w/index.php/Main_Page`), and a primate alignment consisting of human, chimpanzee, gorilla, orangutan and gibbon that is publicly available through the Phylogenetic Analysis by Maximum Likelihood (PAML) software (895 sites at `http://abacus.gene.ucl.ac.uk/software/paml.html#download`) [Benson et al., 2011, Kosakovsky Pond et al., 2005, Yang, 2007].

Table 2.19: Empirical Analyses

Data	Sequence Length	HKY85	TN93	GTR	Best Fit	Ave. Model	Ave. Ts/Tv	Ave. Non-Ts/Tv
Rat/Mouse	1942	5.844	5.849	5.894	5.849	5.739	5.846	3.766
HIV	273	9.158	9.770	0.079	0.106	2.309	2.309	2.127
Primate	895	8.980	8.927	0.117	9.043	6.542	6.542	2.206

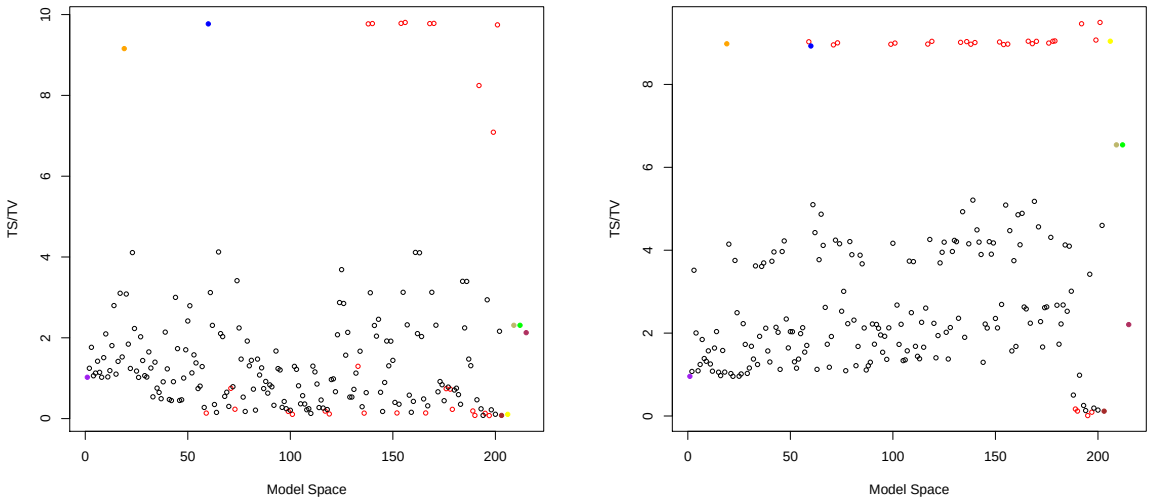
Ts = transition; Tv = transversion

Figure 2.10: Empirical Analysis: Rat/Mouse Data



Ts = transition; Tv = transversion
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure 2.11: Empirical Analyses



Ts = transition; Tv = transversion
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Unlike the data simulations, the model-based estimates for the HIV data have large variability. In this case, the GTR and best fit model estimates of κ are 0.079 and 0.106, respectively. However, the HKY85 and TN93 estimates are 9.158 and 9.770 while the model averaged estimates are 2.309 and 2.127 for the Ts/Tv models and non-Ts/Tv models, respectively. These relatively unstable results may be a byproduct of using global substitution parameters across the phylogeny and the resulting assumption of uniform rates across all branches of the tree.

In the case of the Rat/Mouse data, the issue of confounding non-uniform rates into global parameters is not an issue. This may contribute to the relatively well-behaved nature of the parameter estimates of the model space seen in figure 2.10. The distribution of $\hat{\kappa}$ values is similar to the empirical simulation results based on HKY85 data found in figures A.1, A.4, A.7, and A.10. The Ts/Tv model estimates are grouped together for $\hat{\kappa}$ values of 5.8 to 5.9.

Additionally, the change in the point estimate for the overall average models and the Ts/Tv model averages is less than two percent of the overall model averaged estimate. This implies that the model fit statistics are weighting the Ts/Tv models more than the non-Ts/Tv models. This results in only a two percent change in the overall estimate of κ for the 173 non-Ts/Tv models.

The estimates for the primate data behave similar to the Rat/Mouse data with the exception that the most complex models show high variability (also similar with the HIV data). Therefore, large differences exist between the named models in the primate and HIV data, with the HKY85 and TN93 κ estimates being 8.9 but the GTR and best fit models estimating κ with 0.117. The averaged models all estimate κ as 2.1 to 2.4. Similar to the HIV data, this relative instability may again be due to the confounded nature of assuming constant rates across the phylogenetic tree.

Based on these brief results, the understanding of the performance of substitution models is informative for the expected results from the relatively straightforward Rat/Mouse data that use the same assumptions of uniform substitution rates across the phylogeny, but the estimation methods should be updated for the case of more complete real-world data.

However, the key result from the empirical data is that the biological assumptions of the named models from Section 1.2.3 is valid and repeatable for relatively simple data with uniform substitution rates across sequence divergence. While adjustments may be necessary to implement the models with real world data, such as local estimates of κ, the basic framework of model averaging and its benefits will apply.

2.3 Discussion

Chapter 2 provides analyses of the complete set of substitution models under a variety of conditions and settings that include phylogenetic tree structure, sequence divergence, model-simulated data, and substitution parameters.

The conclusions of this chapter are three-fold:

- Ts/Tv model estimates performed consistently with the mathematical form of the κ estimate. While the design of the simulations was based on named models from the set of "Ts/Tv models", the estimates performed consistently across the full census and various model averaged estimates. This continued across a broad set of simulated conditions and settings.

- Model average estimates calculate $\hat{\kappa}$ well with respect to the named models and true values of κ. The estimates are robust to model misspecification, even under the use of candidate sets that do not contain models describing the underlying process resulting in the data (i.e., non-Ts/Tv model averaged estimates). While the estimates do not uniformly perform best, they consistently estimate κ well for all cases.

- The distribution of the variance for the model estimates is bimodal. Furthermore, this bimodal distribution consistently clustered with the "Ts/Tv models" having the highest variance and the "non-Ts/Tv models" with the lower variances.

The observations regarding the distribution of the variance of κ suggested additional research may benefit the development of shrinkage estimates that find a better performing compromise between the consistency of the "Ts/Tv models" and lower variance of the "non-Ts/Tv models". Possible changes to the weights used for the model averaging may result in better overall properties of Mean Square Error.

Finally, the weights based on information-loss are robust to model misspecification. This further justifies their use in the following chapter and in the use of these weights with real-world data. This is encouraging for the codon case with a large set of models to investigate such that for misspecified models in the traverse of the GA, the model weights may select models with estimates closest to the true value of ω.

The primary purpose of these analyses was to be able to use them as a proof-of-concept for the larger analyses of the codon case. As an added benefit, the comprehensive nature of the exploratory analyses of this section also provided a unique opportunity to study and better

understand the behavior of both consistency and variability across the full set of reversible nucleotide models as well as the various model averaged estimates. Both benefits of these analyses are further explored in the larger and more complex case of codon substitution models in the following chapter.

Chapter 3

Codon Analyses

This chapter presents the analyses of simulated data and model averaged estimates resulting from the traverse of genetic algorithms (GAs) through codon model space. The focus of estimation is on the nonsynonymous-to-synonymous ratio (ω) based on the methods discussed in Section 1.4.2.

Unlike nucleotide estimation in Chapter 2, our analyses of the codon case does not present estimator performance across the complete census of reversible substitution models. The constraints resulting from the enumeration of the model space (2.76×10^{386} per Section 1.2.4) and the computational burden of inverting a 61×61 matrix multiple times for each unrooted phylogenetic tree branch (as opposed to the nucleotide version's 4×4 matrix) resulted in the need to limit the scope of the analyses. Where the analyses of Chapter 2 focused on the comprehensive scope of thoroughly analyzing various simulated biological cases, the codon case focuses on the relative performance of the averaged estimates and the performance of the information-loss weights in order to identify well-performing estimators.

In this chapter, we illustrate that the model averaged estimators perform analogously to the proof-of-concept results discussed in Chapter 2. We also show that the model averaged estimators are generally favorable to standard named models. Finally, we provide methods of refining and better utilizing the information-loss weights for estimates of ω.

3.1 Data Simulation Settings

Similar to Chapter 2, the simulations and analyses that we produced were run with the North Carolina State University BRC Beowulf cluster (Intel Xeon E5-4627 v2 3.3 GHz and E5-4620 2.2 GHz with memory of 1866 and 1333 MHz, respectively). The Applecross/San Diego Alliance cluster was also used, consisting of 30 nodes with between seven and 14 central processing units (CPUs) each.

For each simulated data replicate, 11 independent GA runs were used for fixed values of C from two to 12. Our analyses include comparisons of the estimates of the nonsynonymous to synonymous ratio (ω) for MG94 and various model averaged estimates. The model averaged estimates were calculated for the following: 1) over every unique model the GA explored, 2) percentile-based subsets of the models that the GA traversed (i.e., those models with at least median fitness according to $\text{AIC}_{\text{corr.}}$ weights), and 3) exploratory assessments on the non-unique models traversed by the GA including duplicate models visited.

Our data simulations were produced with two different phylogenetic tree structures: a simplistic two leaf tree with a single rooted node and the four leaf structure from Huelsenbeck and Hillis [1993] (Section 3.1). For the two cases, the branch lengths dictated the total sequence divergence. For the two leaf tree, the unrooted branch length was designated as the unit interval. For the four leaf structure, per Section 3.1 we designated $\nu_1 = 1.60$ and $\nu_2 = 3.60$ (Case I) for a total branch length of 12.

Data Simulation Process

The substitution matrix that was used to create the transition probabilities for the simulated data was constructed under the following algorithm:

1. The synonymous substitution rate for the codon substitution matrix, θ_0, was set fixed.

2. A 4×3 matrix of nucleotide frequencies was created for the stationary distribution of codon frequencies such that $\mathcal{F}_c = \frac{\pi_{y1}\pi_{y2}\pi_{y3}}{1-\Pi_{stop}}, c = 1, 2, ..., 61$

3. An amino acid (AA) probability matrix (P^{AA}) was created using a reference set of amino acid frequencies. The reference set of frequencies was based on those reported by Jones et al. [1992]. This set was based on an empirical amino acid substitution matrix that used an evolutionary distance of one point accepted mutation (PAM) and scaled by a factor of 10^5. Data used for this was release 15.0 of the SWISS-PROT database that included 16,941 sequences [Bairoch, 1990], excluding those with less than 20 residues.

4. The 20×20 AA substitution matrix (R_{AA}) was approximated with the $logm(\cdot)$ function on the R Platform, as $R^{\text{AA}} = \frac{1}{t} * logm(P^{\text{AA}})$.

5. The elements of R_{AA} were clustered into $C = 6$ mutually exclusive sets (i.e., θ_{1-6}).

6. Single-change synonymous substitution positions in the codon substitution matrix were calculated as $r_{c_1c_2}^{\text{codon}} = \theta_0 \pi_{y_2}$ where π_{y_2} is the stationary probability of the target nucleotide whose change resulted in the substitution.

7. Single-change nonsynonymous substitution positions in the codon substitution matrix were calculated thusly: each position of the codon substitution matrix was mapped with the corresponding clustered amino acid substitution values ($\theta_i, i = 1, 2, ...6$) and scaled by the target nucleotide frequency such that $r_{c_1c_2}^{\text{codon}} = \theta_i \pi_{y_2}$.

8. Set the diagonals of R^{codon} as $r_{c_1c_1} = -\sum_j r_{c_1c_j}$, where $j = 1, 2, ...61$ and $c_j \neq c_1$.

Following the creation of the codon substitution matrix, the process of simulating the sequence alignment followed closely with the methods described in Section 3.1:

1. Codons were generated for the root node of the phylogeny using the discrete probabilities defined in the 4×3 matrix of nucleotide frequencies. Randomly generated stop codons were replaced.

2. The transition probabilities were calculated using R^{codon} for each branch of the rooted tree with the R Platform matrix exponentiation function ($exmp[\cdot]$) and equation 1.1 as $P^{\text{codon}} = exmp(\nu_b * R^{\text{codon}})$.

3. The transition matrix was used to generate each non-root node of the alignment.

4. Step (3) was repeated for all sites and nodes of the rooted phylogeny.

5. The leaf node sequences were output as the alignment for analysis and parameter estimation in HyPhy.

3.2 Simulation Results

3.2.1 Nonsynonymous/Synonymous Ratios

Our results of the estimation of ω across the various simulations are presented in this section with tabular summaries and figures. All results were summarized separately for the simulations based the two leaf tree versus the four leaf tree of Section 3.1.

Analyses of the two leaf tree include 10 replicates for each of the clustering possibilities displayed in this section. Typically, results are based on several replicates. However, the four leaf tree's results are based on only four, because of the requirement of approximately five times the computational resources (exponentiation of the 61×61 matrix five times versus one). Details of these challenges are discussed in further detail within Section 3.2.3.

The GAs used different convergence criteria depending on whether the data were simulated using a two leave or four leaf tree. This is a direct result of the computational constraints encountered above. GAs that were run on data simulated with a two leaf tree required 100 populations without a change of the best-fit child model in order to terminate. The four leaf simulation required half the number of populations without an improvement in order to obtain convergence.

Results based on the two leaf phylogenetic tree are summarized in table 3.1 and figures B.1 and B.2. The four leaf tree is represented in table 3.2 and figures B.3 and B.4. Figures B.1, B.2, B.3, and B.4 represent the true value of ω with the horizontal line across the plot field. Tables 3.1 and 3.2 use brackets to illustrate the distance of the summarized estimates from the truth.

Table 3.1: dN/dS Estimate Results for a Two Leaf Tree

		Mean [Dist.*]; Median [Dist.*]	
Nonsynonymous Clusters (C):	1	2	3
Winner:		0.338 [0.080]; 0.333 [0.085]	0.360 [0.058]; 0.362 [0.056]
Average:		0.337 [0.081]; 0.332 [0.085]	0.359 [0.059]; 0.360 [0.058]
Median:		0.328 [0.090]; 0.326 [0.091]	0.350 [0.068]; 0.359 [0.059]
MG94:	0.264 [0.154]; 0.270 [0.148]		
Nonsynonymous Cluster (C):	4	5	6
Winner:	0.375 [0.043]; 0.380 [0.038]	0.398 [0.020]; 0.421 [0.003]	0.412 [0.006]; 0.400 [0.018]
Average:	0.374 [0.044]; 0.377 [0.041]	0.395 [0.023]; 0.416 [0.001]	0.410 [0.008]; 0.401 [0.017]
Median:	0.356 [0.062]; 0.365 [0.053]	0.369 [0.048]; 0.370 [0.048]	0.387 [0.031]; 0.381 [0.037]
Nonsynonymous Cluster (C):	7	8	9
Winner:	0.441 [0.023]; 0.436 [0.018]	0.418 [0.000]; 0.414 [0.004]	0.829 [0.411]; 0.417 [0.001]
Average:	0.441 [0.023]; 0.435 [0.017]	0.416 [0.002]; 0.414 [0.003]	0.821 [0.404]; 0.414 [0.004]
Median:	0.422 [0.004]; 0.414 [0.004]	0.400 [0.017]; 0.405 [0.013]	0.655 [0.237]; 0.382 [0.036]
Nonsynonymous Cluster (C):	10	11	12
Winner:	2.108 [1.691]; 1.415 [0.997]	120.524 [120.106]; 0.869 [0.451]	76.440 [76.022]; 1.095 [0.677]
Average:	1.501 [1.083]; 1.043 [0.626]	118.047 [117.630]; 0.845 [0.427]	71.220 [70.802]; 1.382 [0.964]
Median:	1.214 [0.796]; 0.959 [0.542]	005.925 [005.507]; 0.870 [0.452]	39.762 [39.344]; 1.119 [0.701]

* Dist. represents the Euclidean norm with respect to the true value of ω.

Table 3.2: dN/dS Estimate Results for a Four Leaf Tree

		Mean [Dist.*]; Median [Dist.*]	
Nonsynonymous Clusters (C):	1	2	3
Winner:		0.233 [0.185]; 0.219 [0.199]	0.239 [0.179]; 0.249 [0.169]
Average:		0.232 [0.186]; 0.218 [0.200]	0.234 [0.183]; 0.242 [0.176]
Median:		0.160 [0.257]; 0.155 [0.263]	0.196 [0.222]; 0.209 [0.209]
MG94:	0.174 [0.244]; 0.169 [0.249]		
Nonsynonymous Cluster (C):	4	5	6
Winner:	0.300 [0.118]; 0.313 [0.105]	0.379 [0.039]; 0.335 [0.083]	0.395 [0.023]; 0.384 [0.034]
Average:	0.299 [0.118]; 0.311 [0.107]	0.366 [0.052]; 0.321 [0.097]	0.330 [0.088]; 0.331 [0.086]
Median:	0.242 [0.176]; 0.240 [0.178]	0.288 [0.130]; 0.270 [0.148]	0.285 [0.132]; 0.281 [0.137]
Nonsynonymous Cluster (C):	7	8	9
Winner:	0.666 [0.249]; 0.436 [0.018]	0.319 [0.099]; 0.340 [0.078]	0.469 [0.051]; 0.499 [0.081]
Average:	0.636 [0.218]; 0.435 [0.017]	0.351 [0.067]; 0.287 [0.131]	0.454 [0.036]; 0.432 [0.014]
Median:	0.370 [0.048]; 0.414 [0.004]	0.266 [0.152]; 0.270 [0.148]	0.317 [0.101]; 0.298 [0.120]
Nonsynonymous Cluster (C):	10	11	
Winner:	0.412 [0.006]; 0.384 [0.034]	0.419 [0.002]; 0.404 [0.014]	
Average:	0.520 [0.103]; 0.449 [0.032]	0.438 [0.020]; 0.461 [0.043]	
Median:	0.374 [0.044]; 0.481 [0.063]	0.303 [0.115]; 0.304 [0.114]	

* Dist. represents the Euclidean norm with respect to the true value of ω.

Figure 3.1: dN/dS Estimate Results for a Two Leaf Tree for Clusters of Size 2 to 7

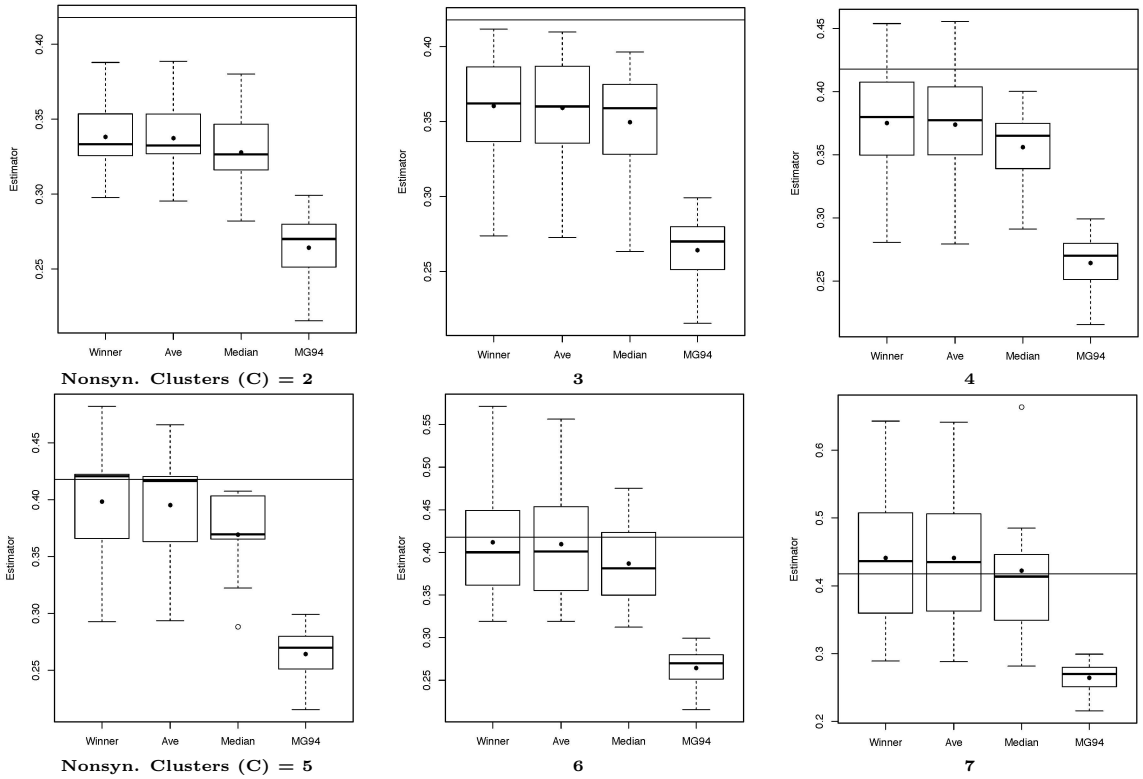


Figure 3.2: dN/dS Estimate Results for a Two Leaf Tree for Clusters of Size 8 to 12

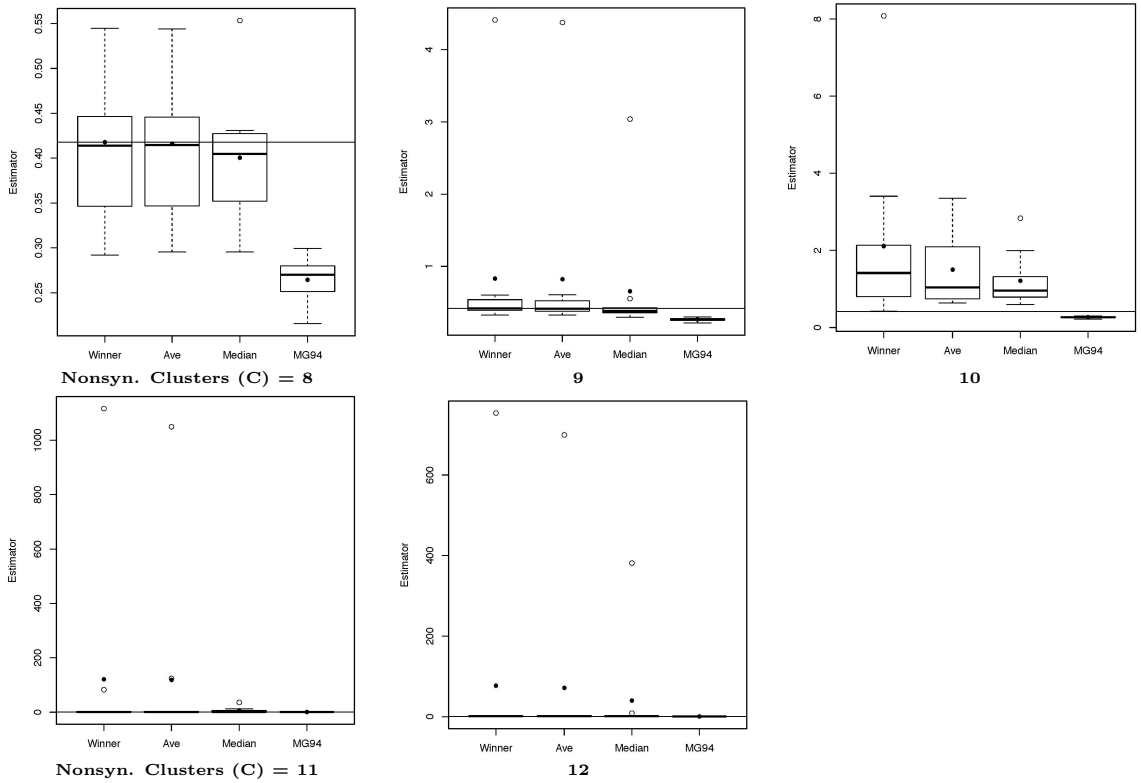


Figure 3.3: dN/dS Estimate Results for a Four Leaf Tree for Clusters of Size 2 to 7

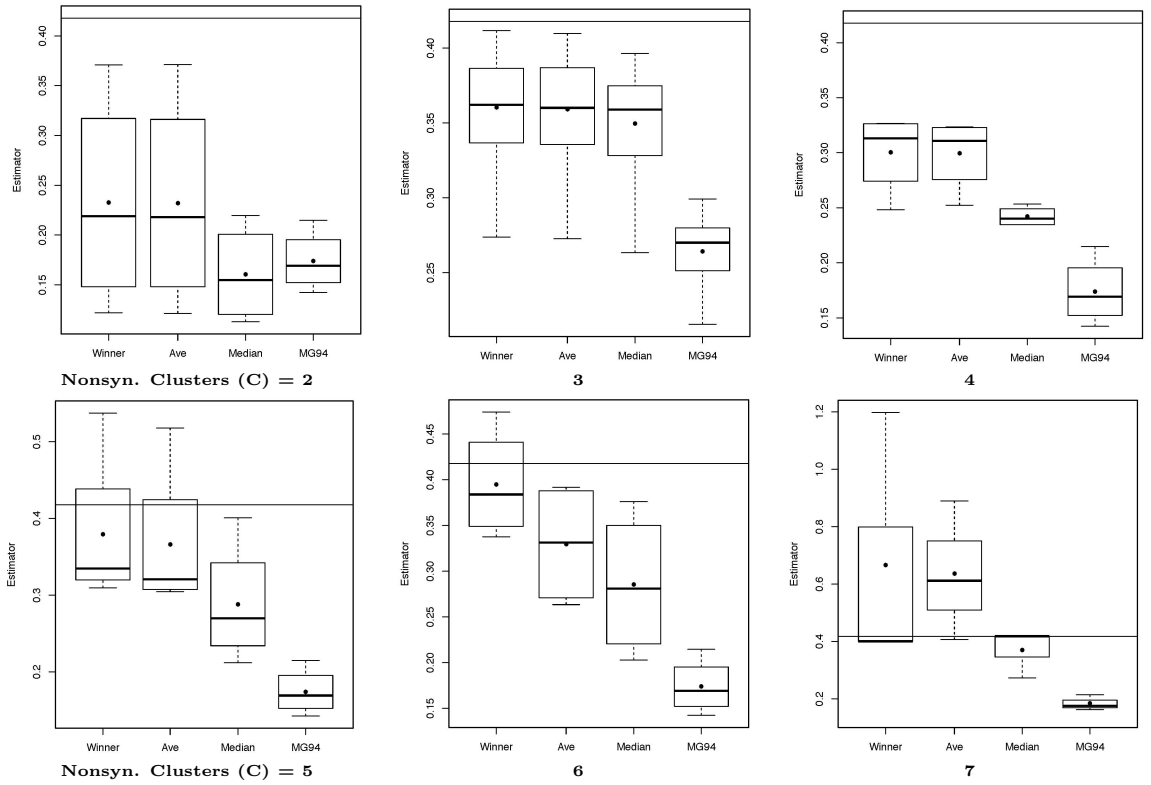
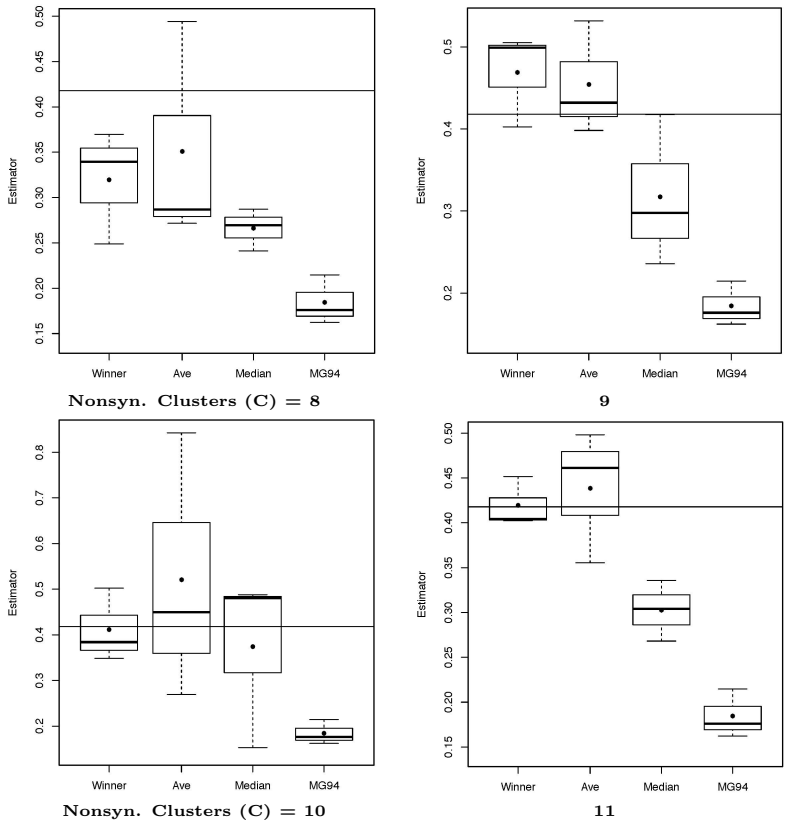


Figure 3.4: dN/dS Estimate Results for a Four Leaf Tree for Clusters of Size 8 to 11



The estimators of ω perform as expected based on the simulation settings and results discussed in Chapter 2. Note that per number 5 in Section 3.1, the true value of C was six for all data simulations. This is consistent with figures B.1 and B.2 that show estimates generally performing poorly in GAs with C being much smaller or larger than six. However, for cases of GAs that considered values of C close to the truth ($C = 6, 7,$ and 8), the mean and median of $\hat{\omega}$ for the model averaged and the best-fit models all estimated the true value of ω fairly well.

Our expectation was that differences between well performing estimators would be small given the following: 1) the two leave case is a simple data structure, and 2) our analysis of data in Chapter 2 resulted in the consideration of only models that did not share parameters between the nonsynonymous and synonymous substitutions. This holds true for figures B.1 and B.2 that shows estimates of ω as similar for the best-fit models and the model averaged estimators. For model averaging using only the top 50^{th} percentile of model fits in each population, the values appear to underestimate, "in the parlance of our times" Coen and Coen [1998], with respect to the best-fit and comprehensive sets of models. Our results strongly indicate that model averaging provides benefits over the named models, fitting the true values of omega more efficiently than other estimates. This is shown in figures B.1, B.2, B.3, and B.4 regarding any of the averaged estimates versus MG94.

The performance of the four tree analyses are disappointing; they do not appear to have a sufficient sample size. A larger set of samples is required to make more definitive conclusions for this analysis, this is shown in the performance of the different estimators with the sampling error across replicates dominating the observed variability. However, even with such a small number of replicates, the general observations of the two leaf case for the upper 50^{th} percentile and MG94 appear to be the same. Again, MG94 is the poorest performer for estimating the true value of ω.

Most importantly, the results shown here are consistent with those of Chapter 2 with respect to the performance of model averaging. Additionally, the analyses of Chapter 2 provide a framework for consideration of only those models that do not share parameters between the

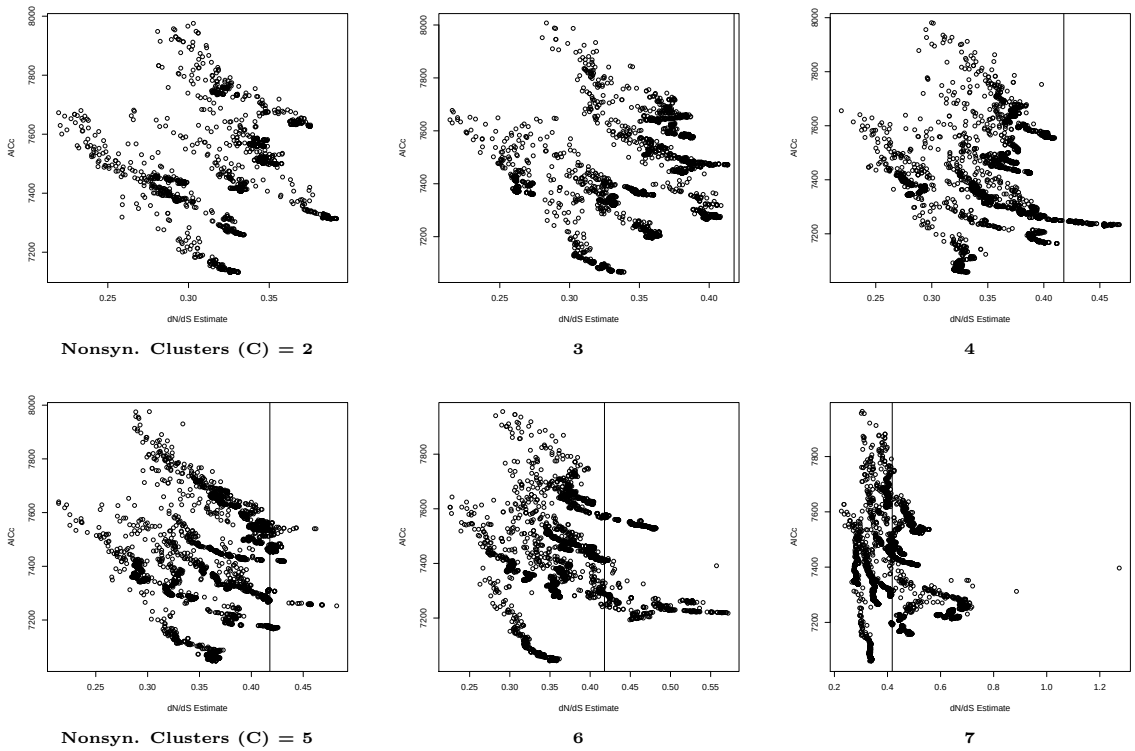
two types of substitutions. This assumption allows the GA to traverse a restricted subset of the possible substitution models and further focus its optimization within the framework of biologically significant models. This assumption is also key to avoiding convergence to local optima outside of biologically significant models, further improving model fits and by extension the model averages.

3.2.2 Information Loss by Nonsynonymous Clusterings

The estimation of information loss through modeling of sequence data is the backbone of our methodology; this is the basis of the GA's traverse through model space as well as the weighting scheme for model averaged estimation. In addition to $\text{AIC}_{\text{corr.}}$ (1.23), the implementation of mBIC (1.24) was studied for this dissertation. The particular function of mBIC was implemented on the basis of its use in CodonTest [Delport et al., 2010b]. However, there was little to no difference in either estimation of weights or resulting model averages between the two methodologies. Note these similarities between $\text{AIC}_{\text{corr.}}$ and mBIC in figures 3.5, 3.6, 3.7, and 3.8 for $\text{AIC}_{\text{corr.}}$ and B.7, B.8, B.11, and B.12 for mBIC.

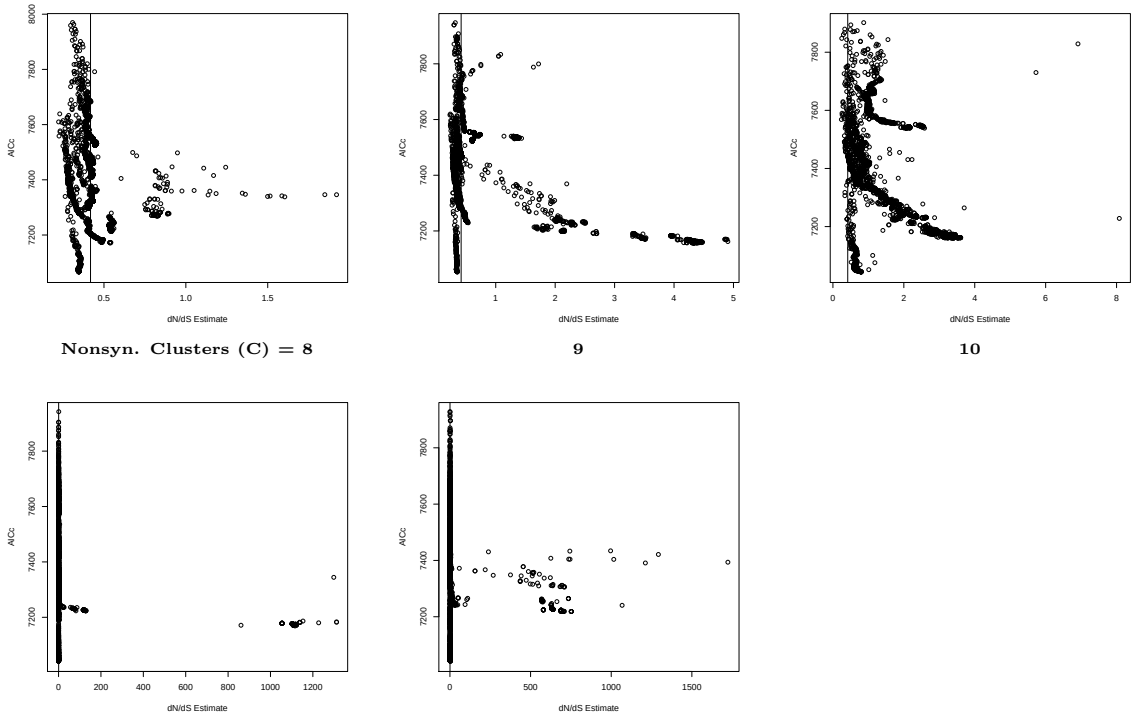
The following figures illustrate information loss estimation versus $\hat{\omega}$ for the best fitting models in each population of the GA's traverse. The vertical lines in the figures represent the true value of ω in the data simulated. Figures 3.5 and 3.6 correspond to the two leaf tree while figures 3.7 and 3.8 are based on the four leaf tree.

Figure 3.5: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree for Clusters of Size 2 to 7



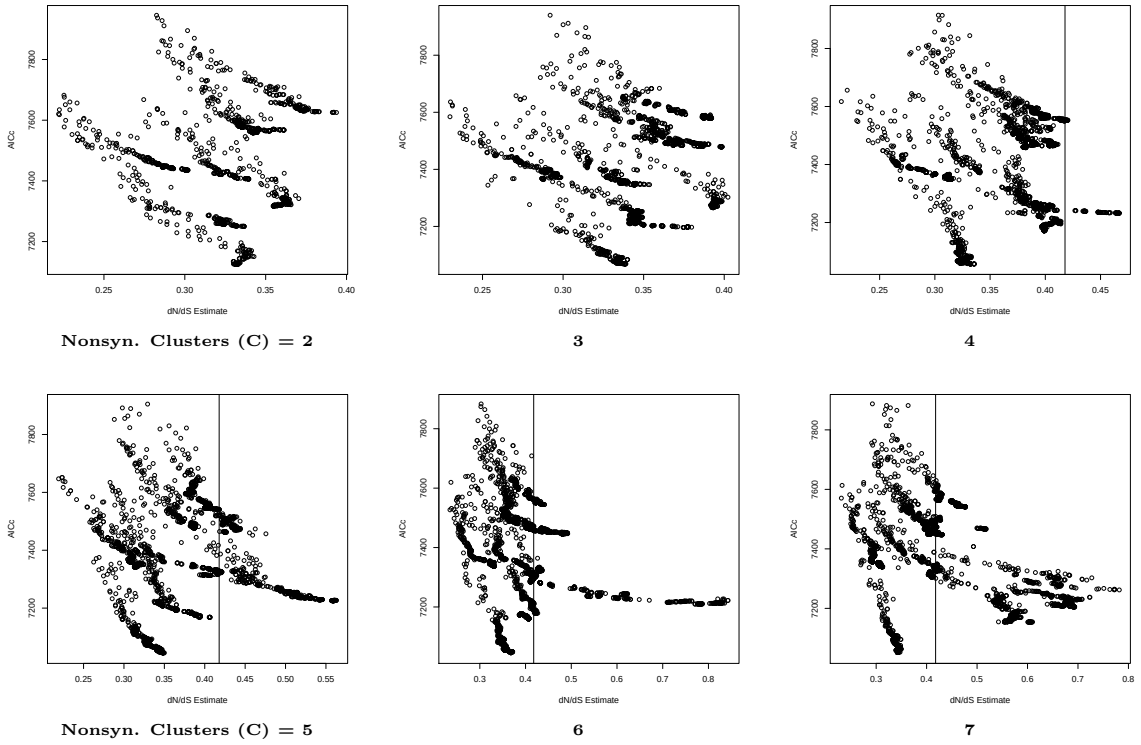
AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure 3.6: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree for Clusters of Size 8 to 12



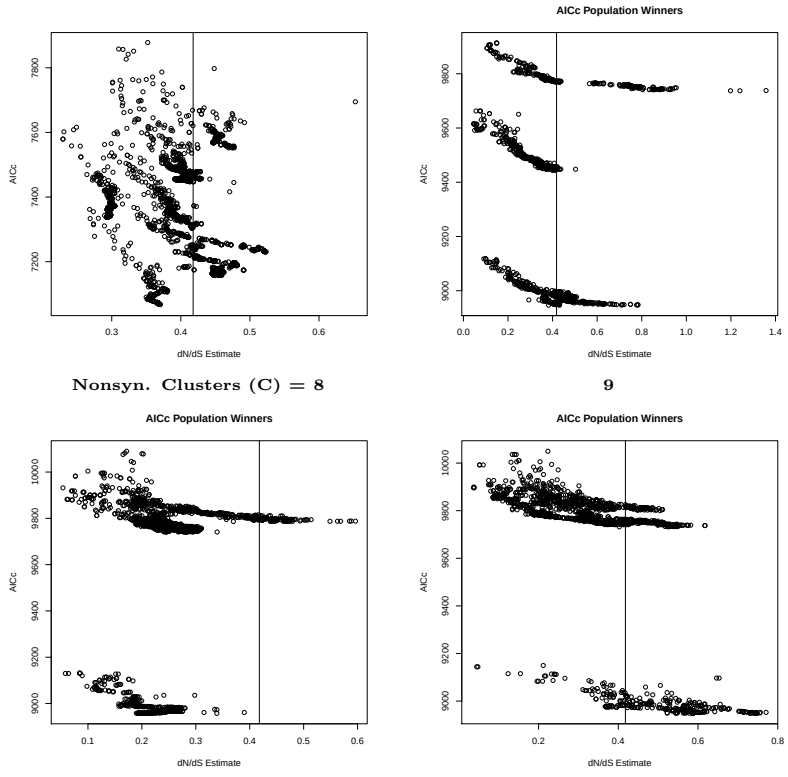
AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure 3.7: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree



AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure 3.8: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree



AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

In each of the figures, the various replicates can generally be seen in the clustering of the estimates from top left to bottom right with a grouping of data points toward the bottom right. This illustrates the convergence of the GA on a given replicate toward models with minimum information loss that also result in estimates of ω closest to the truth. Essentially, as the GAs traverse model space, the tendency of convergence toward the best fit models is robust across the various choices of clusterings (C) and replicates. Therefore, the weighting schemes and ability of the GAs to converge to the best solution remain reliable on the computationally challenging set of possible codon substitution models.

3.2.3 Genetic Algorithm Performance

Per Section 1.2.4, maximization of the likelihood for each model required the exponentiation of a 61×61 matrix for every likelihood evaluation. This along with the scope of the model space are the basis of the computational burden of codon substitution model estimation. Based on Section 1.3.1, this exponentiation must be carried out multiple times for each branch of the unrooted phylogenetic tree during the numeric optimization process. Therefore, while the two leaf case required multiple exponentiations per model, the four leaf case required at least an order of magnitude of five times as many (figure 2.1). Hence, the computational burden of model estimation is significantly higher for the four leaf tree and the reason why the analyses and creation of data replicates has been challenging.

To illustrate these issues, tables 3.3 and 3.4 summarize the number of models required for GA convergence by two versus four leaf cases as well as clustering sizes (C). Results are broken out by the total number of model instances that the GA was required to traverse prior to convergence and the number of unique models visited (excluding duplicate instances that models were visited by the GA). Note that the sample sizes per Section 3.2.1 were 10 for the two leaf case and four for the four leaf case.

Table 3.3: GA Performance - Two Leaf Tree

Nonsyn. Clusters (C) =	2		3		4	
	Unique	Total	Unique	Total	Unique	Total
Mean No. Models	32005.9	56510	61105.8	106520	77311.2	135110
Standard Dev.	7022.62	11677.94	14085.17	28464.47	22183.38	40557.63
Median No. Models	31506	55000	58858.5	98950	73488	129550
Min No. Models	20246	34300	39755	69400	49908	81700
Max No. Models	47049	77900	83216	150700	115060	203400

Nonsyn. Clusters (C) =	5		6		7	
	Unique	Total	Unique	Total	Unique	Total
Mean No. Models	81232.2	134020	79833.7	132420	104978.8	177790
Standard Dev.	28968.21	48243.08	29463.65	50877.04	25156.34	46883.74
Median No. Models	74949	124900	85742	144750	106620	180300
Min No. Models	34925	51600	35020	54100	63405	96200
Max No. Models	129558	207100	125406	214300	145226	259200

Nonsyn. Clusters (C) =	8		9		10	
	Unique	Total	Unique	Total	Unique	Total
Mean No. Models	88339.2	144380	73166.3	120750	82509.3	131570
Standard Dev.	41476.28	70605.95	26830.50	49107.67	33732.09	54015.45
Median No. Models	76112.5	123650	61076	99600	75344.5	121850
Min No. Models	42816	68400	42472	71000	37916	57100
Max No. Models	148240	246600	129251	226100	126633	202000

Nonsyn. Clusters (C) =	11		12	
	Unique	Total	Unique	Total
Mean No. Models	91012.9	150730	85046.2	137740
Standard Dev.	55340.64	97102.98	33710.20	57498.18
Median No. Models	84060.5	131500	74740.5	119500
Min No. Models	23676	38600	37248	60100
Max No. Models	191968	327300	140706	241000

Table 3.4: GA Performance - Four Leaf Tree

Nonsyn. Clusters (C) =	2		3		4	
	Unique	Total	Unique	Total	Unique	Total
Mean No. Models	17756.0	33340.0	39328.2	69360.0	49024.8	89260.0
Standard Dev.	10011.87	18839.67	28967.74	51823.00	32600.38	61838.56
Median No. Models	41400.0	22099.0	31894.0	57900.0	49704.0	85800.0
Min No. Models						
Max No. Models	23735	23735	74124	130400	90857	173300

Nonsyn. Clusters (C) =	5		6		7	
	Unique	Total	Unique	Total	Unique	Total
Mean No. Models	57879.2	103720.0	63630.0	103960.0	37461.8	45280.0
Standard Dev.	36484.25	66495.09	37310.40	60987.28	39924.57	42665.76
Median No. Models	62128.0	107400.0	77120.0	127900.0	39961.0	58200.0
Min No. Models						
Max No. Models	99476	179100	97009	156900	95356	84100

Nonsyn. Clusters (C) =	8		9		10	
	Unique	Total	Unique	Total	Unique	Total
Mean No. Models	47169.8	75700.0	38757.6	59700.0	28538.0	45700.0
Standard Dev.	43236.71	69350.20	35550.85	55315.64	26053.69	41793.96
Median No. Models	73777.0	118000.0	58924	86800.0	47021.0	72300.0
Min No. Models						
Max No. Models	84634	134500	67540	113500	47914	79300

Nonsyn. Clusters (C) =	11	
	Unique	Total
Mean No. Models	18610.4	29700.0
Standard Dev.	17856.60	28410.74
Median No. Models	22219.0	35800.0
Min No. Models		
Max No. Models	36970	58200

Tables 3.4 and 3.4 show the physical burden of running GAs through codon substitution model space. Convergence of GAs required as many as 260,000 model instances (maximum total models for the two leaf case with $C = 7$). Because of the prerequisite of the exponentiation of a large 61×61 matrix for each case, a cluster is required with extensive programming resources and time (see Appendix E) in order to access the robust estimation benefits of the model averaged estimation methods illustrated in Section 3.2.1.

This trade-off of robust estimation versus the computational challenges of traversing codon substitution model space must be considered when implementing the methods outlined in this dissertation for real-world data.

3.3 Discussion

The use of GAs and model averaged estimates in the molecular evolution of codons results in large computational challenges with the potential for significant benefits in the estimation of nonsynonymous-to-synonymous ratios (ω).

The analysis of information loss estimators show the ability of these weights to guide the traverse of the GA to convergence in best-fit. This observation and the results from Chapter 2 provide sufficient evidence of their use in both model selection and averaging given the estimates remained robust even to systemic misspecification. Additionally, examination of possible differences in corrected AIC versus modified BIC show little to no difference between their usage for both model selection and averaged estimation. $\text{AIC}_{\text{corr.}}$ was used in the analysis of $\hat{\omega}$ performance.

The strategy of partitioning the model space to exclude those sharing parameters between the nonsynonymous and synonymous codon substitutions is consistent with the methods of CodonTest [Delport et al., 2010b], justified with extensive analysis in Chapter 2, and results in much needed computational efficiencies for this problem. Despite the gains in computational efficiency, challenges with the convergence of GAs for codon substitution models are shown to be as much as hundreds of thousands of times greater than the traditional methods of *a priori*

designated models for codon evolutionary estimation. These issues are then shown to grow significantly as the complexity of the tree increases on even a small scale (i.e., a two versus four leaf phylogenetic tree structure results in five times the computational burden).

Despite the challenges of the methods described in this Chapter, the benefits in robust estimation of ω remain. Consideration of both the benefits and challenges of GA model selection for averaged estimators should be weighted in the decisions regarding usage of these methods.

Chapter 4

Conclusions

This dissertation analyzed model averaged estimates from the traverse of a GA through the set of reversible substitution models. Typical analyses are based on a single model that was selected based on prior information and/or the observed data. Our goal was to improve the performance of analyses by incorporating information from the models that were traversed during the model selection routine as opposed to a single selected model or *a priori* chosen one.

We used model averaging for the reversible, nucleotide substitution models across different evolutionary processes, divergence, sequence length, phylogeny structure, and parameter values. The benefits were particularly notable as limiting empirical bias in the transition-to-transversion estimates and recovering the individual parameter values. Additionally, we found estimation benefits even when deliberately selecting non-biologically significant models outside the class of models based on our the simulated data. In these cases, the information-loss weights of our averaged estimates produced transition-transversion ratios closest to the true values from among the set of models available. This illustrated the robust nature of our methodology to best recover the true values even among models that were deliberately chosen to be poor. Finally, the empirical analysis of variance for the set of reversible models and model averaged estimates showed an interesting bimodal distribution based on the biological significance and complexity of the models analyzed. This suggested potential benefits that may be found by using shrinkage

estimates in order to improve the MSE of averaged estimators by incorporating non-biologically significant models into the estimates.

Potential future research may be the exploration of shrinkage estimates to leverage the observed bimodal distribution of variance from this research. The most challenging part of this may be in the weighting of estimates with "good" variance properties. This is because the results of the information-loss weights currently show that models poorly estimating the true, underlying value tend to be weighted significantly less and these estimates are those in the lower mode of the variance distribution. Therefore, the current information-loss methods will tend to down-weight those models with small variance and therefore limit the shrinkage effect. Based on these results, a different weighting scheme may be needed to result in an effective shrinkage estimate.

Following our exploration of the set of nucleotide substitution models, we extended our methods to the codon case. We developed a genetic algorithm with the use of information loss estimates as weights and simulated the data based on amino acid frequencies observed in the SWISS-PROT database. The traverse of the GAs, together with the weighted averages were then used as a model selection method for estimation of the nonsynonymous-to-synonymous ratio for codon substitutions. Consistent with the results of the nucleotide substitution model estimates, our model averages for the codon case show better performance in point estimation than named models. However, challenges exist due to the computational burden of likelihood maximization for codon models, and the magnitude of models required for convergence of the genetic algorithm. This currently results in limitations of the scope of the analysis detailed in this dissertation.

Future research may aid in producing further empirical evidence on the benefits of model averaging for codon substitutions with the use of GAs. Additionally, further research may also provide more computationally efficient methods.

The application of the model average methods of this dissertation may be currently applied within the phylogenetic evolutionary field as an alternative to traditional, *a priori* named mod-

els, hLRTs, and information loss-based selection. However, benefits of these methods may be extended beyond the field of molecular evolution. For example, in the field of health economics and cost effectiveness, Markov processes have been used with increasing frequency to analyze disease state changes over deductive reasoning with respect to decision trees [Sonnenberg and Beck, 1993]. The use of genetic algorithms and model averaging may be a novel benefit to analyze healthcare resource utilization on the model space generated by these disease state changes.

In general, model averaging show two-fold benefit. It accounts for the variability associated with the use of different models and recovers the true values efficiently with respect to the broad set of empirical analyses contained within this dissertation. The author of this dissertation intends to continue the extension of these benefits to the pharmaceutical industry and health economic fields.

References

Abascal, Federico, Zardoya, Rafael, and Posada, David. *ProtTest: Selection of Best-Fit Models of Protein Evolution.* Bioinformatics. 21(9): 2104-2105, 2005.

Akaike, Hirotugu. *A New Look at the Statistical Model Identification.* IEE Transactions on Automatic Control. 19(6): 716-723, 1974.

Alfaro Michael E., and Huelsenbeck John P. , *Comparative Performance of Bayesian and AIC-based Measures of Phylogenetic Model Uncertainty.* Systemic Biology. 55(1): 89-96, 2006.

Anderson, David R., Burnham, Kenneth P., Gould, William R., and Cherry, Steve. *Concerns About Finding Effects That Are Actually Spurious.* Wildlife Society Bulletin. 29(1): 311-316, 2001.

Anisimova, Maria, Bielawski, Joseph P., and Yang, Ziheng. *Accuracy and power of the likelihood ratio test in detecting adaptive molecular evolution.* Molecular Biology and Evolution. 18(8): 1585-1592, 2001.

Bairoch, Amos. *PC/Gene: a protein and nucleic acid sequence analysis micro-computer package, PROSITE: a dictionary of sites and patterns in proteins and SWISS-PROT: a protein sequence data bank.* Ph.D. thesis, University of Geneva. 1990.

Bates, Douglas and Maechler, Martin. matrix: Sparse and Dense Matrix Classes and Methods. R Package version 1.2-2. 2015.

Benson, David A., Karsch-Mizrachi, Ilene, Clark, Karen, Lipman, David J., Ostell, James, and Sayers, Eric A. *GenBank* Nucleic Acids Research. 1-6. 2011.

Blanchet, F. Guillaume, Legendre, Pierre, and Borcard, Daniel. *Forward Selection of Explanatory Variables* Ecology. 89(9): 2623-2632, 2008.

Box, George EP, and Normal R. Draper. *Empirical Model-Building and Response Surfaces.* John Wiley and Sons, 1987.

Buckland, Stephen T., Burnham, Kenneth P., and Augustin, Nicole H. *Model Selection: An Integral Part of Inference.* Biometrics. 53(2): 603-618, 1997.

Burnham, Kenneth P., and Anderson, David R. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach,* 2^{nd} *edition.* Springer-Verlag New York, Inc., 2010.

Camin, Joseph H., and Sokal, Robert R. *A method for deducing branching sequences in phylogeny.* Evolution. 311-326, 1965.

Cavalli-Sforza, Luigi L., and Edwards, A. F. W. *Phylogenetic Analysis. Models and Estimation Procedures* American Journal of Human Genetics. 19(3 Pt 1): 233-257, 1967

Chang, J T. *Full Reconstruction of Markov Models on Evolutionary Trees: Identifiability and Consistency.* Mathematical Biosciences. 137: 51-73, 1996.

Champion, Matthew. "Re: How many atoms make up the universe?" http://www.madsci.org/posts/archives/oct98/905633072.As.r.html. 1998.

Claeskens, Gerda, and Hjort, Nils Lid. *Model selection and model averaging. Vol. 330.* Cambridge: Cambridge University Press, 2008.

The Big Lebowski. Dir. Ethan Coen and Joel Coen. Perf. Jeff Bridges, John Goodman, and Julianne Moore. Gramercy, 1998. DVD

Crandall, Keith A. (ed.) *The Evolution of HIV* Baltimore, The Johns Hopkins University Press, 1999.

Dayhoff, Margaret O., and Eck, Raimund. *A model of evolutionary change in proteins. In:* Dayhoff, M and Eck, R (ed.) Atlas of protein sequence and structure. 196768. Washington DC: National Biomedical Research Foundation. p. 3341, 1968.

Dayhoff, Margaret O., and Schwartz, Robert M. *A model of evolutionary change in proteins. In:* Dayhoff M (ed.) Atlas of protein sequence and structure. Vol. 5 (suppl 3). Washington DC: National Biomedical Research Foundation. p. 345-352, 1978.

Delport, Wayne, Scheffler, Konrad, Gravenor, Mike B., Muse, Spencer V., and Kosakovsky Pond, Sergei. *Benchmarking Multi-Rate Codon Models* PLoS One. 5(7): e11587, 2010.

Delport, Wayne, Scheffler, Konrad, Botha, Gordon, Gravenor, Mike B., Muse, Spencer V., and Kosakovsky Pond, Sergei L. *CodonTest: Modeling Amino Acid Substitution Preferences in Coding Sequences.* PLoS Computational Biology. 6(8): e1000885, 2010.

Douady, Christophe J., Delsuc, Frederic, Boucher, Yan, Doolittle, Ford W., and Douzery, Emmanuel J. P. *Comparison of Bayesian and Maximum Likelihood Bootstrap Measures of Phylogenetic Reliability.* Molecular Biology and Evolution. 20(2): 248-254, 2003.

Drummond, Alexei, J., and Rambaut, Andrew. *BEAST: Bayesian Evolutionary Analysis by Sampling Trees.* BMC Evolutionary Biology 7(1): 214, 2007.

Edwards, A. W. F. *The Origin and Early Development of the Method of Minimum Evolution for the Reconstruction of Phylogenetic Trees.* Systematic Biology. 45(1): 79-91, 1996.

Edwards, A. W. F, and Cavalli-Sforza, L. L. *The reconstruction of evolution. (Abstr.)* Heredity 18:553; Ann. Hum. Genet. 27:104-105, 1963b.

Farris, James S. *Methods for computing Wagner trees.* Systematic Biology. 19(1): 83-92, 1970.

Farris, James S. *On the use of the parsimony criterion for inferring evolutionary trees.* Systematic Zoology. 22: 250-256, 1973.

Farris, James S. *Phylogenetic analysis under Dollo's Law.* Systematic Biology. 26(1): 77-88, 1977.

Ferris, Stephen D., Portnoy, Stephen L., and Whitt, Gregory S. *The Roles of Speciation and Divergence Time in the Loss of Duplicate Gene Expression* Theoretical Population Biology. 15(1): 114-139, 1979.

Felsenstein, Joseph. *Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters.* Systematic Biology. 22(3): 240-249, 1973.

Felsenstein, Joseph. *Cases in which parsimony or compatibility methods will be positively misleading.* Systematic Biology. 27(4): 401-410. 1978.

Felsenstein, Joseph. *Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach.* Journal of Molecular Evolution. 17: 368-376, 1981.

Felsenstein, Joseph. *Inferring Phylogenies.* Sinauer Associates, Inc, Massachusetts, 2004.

Forsberg, Roald, and Christiansen, Freddy B. *A Codon-Based Model of Host-Specific Selection in Parasites, with an Application to the Influenza A Virus.* Molecular Biology and Evolution. 20(8): 1252-1259, 2003.

Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning,* 1^{st} *edition.* Addison-Wesley Professional, 1989.

Goldman, David E., and Yang, Ziheng. *A Codon-based Model of Nucleotide Substitution for Protein-coding DNA Sequences.* Molecular Biology and Evolution. 11(5): 725-736, 1994.

Graur, Dan, and Li, Wen-Hsiung. *Fundamentals of Molecular Evolution,* 2^{nd} *edition.* Sinauer Associates, Inc., Sunderland, 2008.

Griffiths, Anthony J.F., Miller, Jeffrey H., Suzuki, David T., Lewontin, Richard C., and Gelbart, William M. *An Introduction to Genetic Analysis* W.H. Freeman and Company, New York, 2000.

Hasegawa, Masami, Kishino, Hirohisa, and Yano, Taka-aki. *Dating of the Human-Ape Splitting by a Molecular Clock of Mitochondrial DNA.* Journal of Molecular Evolution. 22: 160-174, 1985.

Higham, Nicholas J. *Functions of Matricies: Theory and Computation.* Society for industrial and Applied Mathematics, Philadelphia, 2008.

Hjort, Nils Lid, and Claeskens, Gerda. *Frequentist Model Average Estimators* Journal of the American Statistical Association. 98(464): 879-899, 2003.

Holmquist, Richard. *Theoretical Foundations for a Quantitative Approach to Paleogenetics, Part 1: DNA* Journal of Molecular Evolution. 1: 115-133, 1972.

Huelsenbeck, John P., and Hillis, David M. *Success of Phylogenetic Methods in the Four-Taxon Case* Systematic Biology. 42(3): 247-264, 1993.

Huelsenbeck, John P., and Ronquist, Fredrik. *MRBAYES: Bayesian inference of phylogenetic trees.* Bioinformatics. 17(8): 754-755, 2001.

Huelsenbeck, John P., and Dyer, Kelly A. *Bayesian Estimation of Positively Selected Sites.* Journal of Molecular Evolution. 58: 661-672, 2004.

Huelsenbeck, John P., Larget, Bret, and Alfaro, Michael E. *Bayesian Phylogenetic Model Selection using Reversible Jump Markov Chain Monte Carlo.* Molecular Biology and Evolution. 21(6): 1123-1133, 2004.

Huelsenbeck, John P., Jain, Sonia, Frost, Simon W. D., and Kosakovsky Pond, Sergei L. *A Dirichlet Process Model for Detecting Positive Selection in Protein-Coding DNA Sequences.* Proceedings of the National Academy of Sciences of the United States of America. 103(16): 6263-6268, 2006.

Ina, Yasuo. *Estimation of the Transition/Transversion Ratio.* Journal of Molecular Evolution. 46: 521-533, 1998.

Jones, David T., Taylor, William R., and Thornton, Janet M., *The Rapid Generation of Mutation Data Matrices from Protein Sequences.* Bioinformatics. 8(3): 275-282, 1992.

Jukes, Thomas H., and Cantor, Charles R. *Evolution of Protein Molecules. in* H.M. Munrow (ed.) Mammalian Protein Metabolisms. Academic Press, New York. p. 21-132, 1969.

Kaplan, Norman, and Langley, Charles H. *A New Estimate of Sequence Divergence of Mitochondrial DNA Using Restriction Endonuclease Mappings.* Journal of Molecular Evolution. 13(4): 295-304, 1979.

Kashyap, R. L., and Subas, S. *Statistical Estimation of Parameters in a Phylogenetic Tree Using a Dynamic Model of the Substitution Process* Journal of Theoretical Biology. 47(1): 75-101, 1974.

Kass, Robert E., and Wasserman, Larry. *A Reference Bayesian Test for Nested Hypotheses and Its Relationship to the Schwarz Criterion.* Journal of the American Statistical Association. 90(431): 928-934, 1995.

Kimura, Motoo *A Simple Method for Estimating Evolutionary Rates of Base Substitutions Through Comparative Studies of Nucleotide Sequences.* Journal of Molecular Evolution. 16: 111-120, 1980.

Kosakovsky Pond, Sergei L. *Modeling Evolution of Protein Coding DNA Sequences.* Department of Mathematics, University of Arizona, 2003.

Kosakovsky Pond, Sergei L., and Frost, Simon D.W. *A Genetic Algorithm Approach to Detecting Lineage-Specific Variation in Selection Pressure.* Molecular Biology and Evolution. 22(3): 478-485, 2004.

Kosakovsky Pond, Sergei L., and Muse, Spencer V. *Site-to-Site Variation of Synonymous Substitution Rates.* Molecular Biology and Evolution. 22(12): 2375-2385, 2005.

Kosakovsky Pond, Sergei L., Frost, Simon D.W., and Muse, Spencer V. *HyPhy: Hypothesis Testing Using Phylogenies.* Bioinformatics. 21(5): 676-679, 2005.

Kosakovsky Pond, Sergei L., Posada, David, Gravenor, Michael B., Woeld, Christopher H., and Frost, SimonD. W. *GARD: A Genetic Algorithm for Recombination Detection.* Bioinformatics. 22(24): 3096-3098, 2006.

Kosakovsky Pond, Sergei L., Mannino, Frank V., Gravenor, Michael B., Muse, Spencer V., and Frost, Simon D.W. *Evolutionary Model Selection with Genetic Algorithm: A Case Study Using Stem RNA.* Molecular Biology and Evolution. 24(1): 159-170, 2007.

Kosiol, Carolin, Holmes, Ian, and Goldman, Nick. *An Empirical Codon Model for Protein Sequence Evolution.* Molecular Biology and Evolution. 24(7): 1464-1479, 2007.

Kullback, Solomon and Leibler, Richard A. *On Information and Sufficiency* Annals of Mathematical Statistics. 22: 79-86, 1951.

Lartillot, Nicolas and Philippe, Herve. *Computing Bayes Factors Using Thermodynamic integration.* Systematic Biology. 55(2): 195-207, 2006.

Levins, Richard. *The Strategy of Model Building in Population Biology.* American Scientist. 54(4): 421-431, 1966.

Li, Wen-Hsiung and Gouy, Manolo. *Statistical Methods for Testing Molecular Phylogenies.* Phylogenetic Analysis of DNA Sequences (M. M. Miyamoto and J. Cracraft, Ed.) Oxford University Press, New York. 1991.

Li, Wen-Hsiung, Wu, Chung-I, and Luo, Chi-Cheng. *A New Method for Estimating Synonymous and Nonsynonymous Rates of Nucleotide Substitution Considering the Relative Likelihood of Nucleotide and Codon Changes.* Molecular Biology and Evolution. 2(2): 150-174, 1985.

Luzon, Maria Victoria, Barreiro, E., Yeguas, E., and Joan-Arinyo, R. *GA and CHC. Two Evolutionary Algorithms to Solve the Root Identification Problem in Geometric Constraint Solving.* ICCS: 4th International Conference. 3039: 139-146, 2004.

Moler, Cleve, and Van Loan, Charles. *Nineteen Dubious Ways to Compute the Exponential of a Matrix.* SIAM Review. 20(4): 801-836, 1978.

Moler, Cleve, and Van Loan, Charles. *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later.* SIAM review. 45(1): 3-49, 2003.

Muse, Spencer V., and Gaut, Brandon S. *A Likelihood Approach for Comparing Synonymous and Nonsynonymous Nucleotide Substitution Rates, with Application to the Chloroplast Genome.* Molecular Biology and Evolution. 11(5): 715-724, 1994.

Muse, Spencer V. *Evolutionary Analyses of DNA Sequences Subject to Constraints on Secondary Structure.* Genetics. 139: 1429-1439, 1995.

Muse, Spencer V. *Estimating Synonymous and Nonsynonymous Substitution Rates.* Molecular Biology and Evolution. 13(1): 105-114, 1996.

Muse, Spencer, V. and Weird, B. S. *Testing for Equality of Evolutionary Rates.* Genetics. 132: 269-276, 1992.

Miller, Forrest R., Neill, James W., and Sherfey, Brian W. *Implementation of a Maximin Power Clustering Criteria to Select Near Replicates for Regression Lack-of-Fit Tests.* Journal of the American Statistical Association. 94(446): 610-620, 1999.

Miyata, Takashi, and Yasunaga, Teruo. *Molecular Evolution of mRNA: A Method for Estimating Evolutionary Rates of Synonymous and Amino Acid Substitutions from Homologous Nucleotide Sequences and Its Application* Journal of Molecular Evolution. 16: 23-36, 1980.

Nei, Masatoshi, and Gojobori, Takashi. *Simple Methods for Estimating the Numbers of Synonymous and Nonsynonymous Nucleotide Substitutions* Molecular Biology and Evolution. 3(5): 418-426, 1986.

Nei, Masatoshi. *Molecular Evolutionary Genetics.* Columbia Univ. Press, New York, 1987.

Neyman, J. *Statistical decision- theory and related topics.* 1-27 *in* Gupta S S, Yackel J (eds.) Academic Press, New York. 1971.

Nielsen, Rasmus, and Yang, Ziheng. *Likelihood Models for Detecting Positively Selected Amino Acid Sites and Applications to the HIV-1 Envelope Gene.* Genetics. 148(3); 929-936, 1998

Nielsen, Rasmus. ed. *Statistics for Biology and Health,* 2^{nd} *edition.* New York, Singer, 2005.

Nylander, Johan A., Ronquist, fredrik, Huelsenbeck, John P., and Nieves-Aldrey, Jose Luis. *Bayesian Phylogenetic Analysis of Combined Data.* Systematic Biology. 53(1): 45-67, 2004.

Pol, Diego. *Empirical problems of the hierarchical likelihood ratio test for model selection.* Systematic Biology. 53(6): 949-962, 2004.

Posada, David, and Crandall, Keith A. *MODELTEST: Testing the Model of DNA Substitution.* Bioinformatics. 14(9): 817-818, 1998.

Posada, David, and Crandall, Keith A. *Selecting the Best-Fit Model of Nucleotide Substitution.* Systematic Biology. 50(4): 580-601, 2001.

Posada, David and Buckley, Thomas R. *Model Selection and Model Averaging in Phylogenetics: Advantages of Akaike Information Criterion and Bayesian Approaches Over Likelihood Ratio Tests.* Systematic Biology. 53(5): 793-808, 2004.

Posada, David. *jModelTest: Phylogenetic Model Averaging.* Molecular Biology and Evolution. 25(7): 1253-1256, 2008.

R Core Team. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. 2013. URL http://www.R-project.org/.

Ren, Fengrong, Tanaka, Hiroshi, and Yang, Ziheng. *An Empirical Examination of the Utility of Codon-Substitution Models in Phylogeny Reconstruction.* Systematic Biology. 54(5): 808-818, 2005.

Roch, Sebastien. *A Short Proof that Phylogenetic Tree Reconstruction by Maximum Likelihood is Hard.* Transactions on Computational Biology and Bioinformatics. 3(1): 92-94, 2006.

Rodrigue, Nicolas, Philippe, Herv Philippe, and Lartillot, Nicolas. *Uniformization for Sampling Realizations of Markov Processes: Applications to Bayesian Implementations of Codon Substitution Models.* Bioinformatics. 24(1): 56-62, 2008.

Rodrigue, Nicolas, Lartillot, Nicolas, and Philippe, Herve. *Bayesian Comparisons of Codon Substitution Models.* Genetics. 180: 1579-1591, 2008.

Rogers, James S. *On the Consistency of Maximum Likelihood Estimation of Phylogenetic Trees from Nucleotide Sequences.* Systematic Biology. 46: 354-357, 1997.

Ronquist, Fredrik, Teslenko, Maxim, van der Mark, Paul, Ayres, Daniel L., Darling, Aaron, Hohna, Sebastian, Larget, Bret, Liu, Liang, Suchard, Mark A., and Huelsenbeck, John. P. *MrBayes 3.2: Efficient Bayesian Phylogenetic Inference and Model Choice Across a Large Model Space.* Systematic Biology. 61(3): 539-542, 2012.

Saitou, Naruya and Nei, Masatoshi. *The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees.* Journal of Molecular Evolution. 27: 261-273, 1988.

Sanderson, Michael J., and Kim, Junhyong. *Parametric Phylogenetics?* Systemic Biology. 49(4):817-829.

Schneider, Adrian, Cannarozzi, Gina M., and Gonnet, Gaston H. *Empirical Codon Substitution Matrix.* BMC Bioinformatics. 6(134): 1-7, 2005.

Shimodaira, Hidetoshi. *Assessing the Error Probability of the Model Selection Test.* Annals of the Institute of Statistical Mathematics. 49(3): 395-410, 1997.

Shimodaira, Hidetoshi. *An Application of Multiple Comparison Techniques to Model Selection.* Annals of the Institute of Statistical Mathematics. 50(1): 1-13, 1998.

Sonnenberg, Frank A., and Beck, Robert. *Markov Models in Medical Decision Making: A Practical Guide.* Medical Decision Making 13: 322-339, 1993.

Suchard, Marc A., Weiss, Robert E., and Sinsheimer, Janet S. *Bayesian Selection of Continuous-Time Markov Chain Evolutionary Models.* Molecular Biology and Evolution. 18(6): 1001-1013, 2001.

Sugiura, Nariaki. *Further Analysts of the Data by Akaike's Information Criterion and the Finite Corrections.* Communications in Statistics - Theory and Methods. 7: 13-26, 1978.

Swoford, David L., Waddell, Peter J., Huelsenbeck, John P., Foster, Peter G., Lewis, Paul O., and Rogers, James S. *Bias in Phylogenetic Estimation and Its Relevance to the Choice Between Parsimony and Likelihood Methods.* Systematic Biology. 50(4): 525-539, 2001.

Tamura, Koichiro, and Nei, Masatoshi. *Estimation of the Number of Nucleotide Substitutions in the Control Region of Mitochondrial DNA in Humans and Chimpanzees* Molecular Biology and Evolution. 10: 512-526, 1993.

Tavare, Simon. *Some Probabilistic and Statistical Problems in the Analysis of DNA Sequences.* Pages 57-86 *in* Some Mathematical Questions in Biology - DNA Sequence Analysis (R. M. Miura, Ed.). American Mathematics Society. Providence, RI.

Uzzell, Thomas, and Corbin, Kendal W. *Fitting Discrete Probability Distributions to Evolutionary Events.* Science. 172(3988): 1089-1096, 1971.

Vuong, Quang H. *Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses.* Econometrica. 57(2): 307-333, 1989.

Wald, Abraham. *Note on the Consistency of the Maximum Likelihood Estimate.* Annals of Mathematical Statistics. 20: 595-601, 1949.

Yang, Ziheng. *Maximum Likelihood Estimation of Phylogeny from DNA Sequences when Substitution Rates Differ Over Sites.* Molecular Biology and Evolution. 10: 1396-1401, 1993.

Yang, Ziheng. *Estimating the Pattern of Nucleotide Substitution.* Journal of Molecular Evolution. 39: 105-111, 1994.

Yang, Ziheng. *Statistical Properties of the Maximum Likelihood Method of Phylogenetic Estimation and Comparisons with Distance Matrix Methods.* Systematic Biology. 43(3): 329-342, 1994.

Yang, Ziheng. *PAML: A Program Package for Phylogenetic Analysis by Maximum Likelihood.* Computer Applications in the Biosciences 13(5): 555-556, 1997.

Yang, Ziheng, and Nielsen, Rasmus. *Estimating Synonymous and Nonsynonymous Substitution Rates Under Realistic Evolutionary Models.* Molecular Biology and Evolution 17(1): 32-43, 2000.

Yang, Ziheng, Nielsen, Rasmus, Goldman, Nick, and Pedersen, Anne-Mette Krabbe. *Codon-Substitution Models for Heterogeneous Selection Pressure at Amino Acid Sites.* Genetics. 155: 431-449, 2000.

Yang, Ziheng, and Swanson, Willie J. *Codon-substitution models to detect adaptive evolution that account for heterogeneous selective pressures among site classes.* Molecular Biology and Evolution. 19(1): 49-57, 2002.

Yang, Ziheng, and Nielsen, Rasmus. *Codon-substitution models for detecting molecular adaptation at individual sites along specific lineages.* Molecular Biology and Evolution. 19(6): 908-917, 2002.

Yang, Ziheng. *Computational Molecular Evolution, Vol. 21.* Oxford, Oxford University Press, 2006.

Yang, Ziheng. *PAML 4: Phylogenetic Analysis by Maximum Likelihood.* Molecular Biology and Evolution. 24(8): 1586-1591, 2007.

Yates, Daniel S., Moore, David S., and Starnes, Daren S. *The Practice of Statistics* 3rd Ed. Freeman, 2008.

Appendices

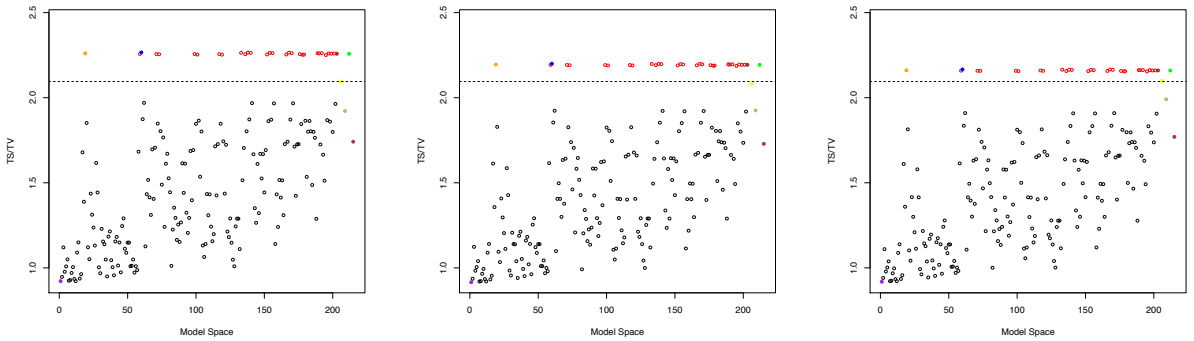
Appendix A

Nucleotide Simulation Results

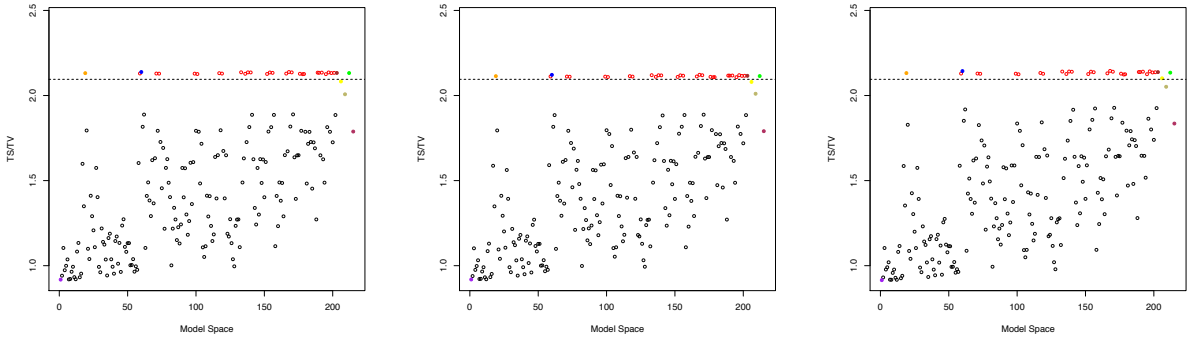
A.1 Transition/Transversion Estimates

A.1.1 HKY85 Data Analyses

Figure A.1: Ts/Tv Estimates on HKY85 Data with Sequence Length 250 and E(To) = 0.138 to 1.376



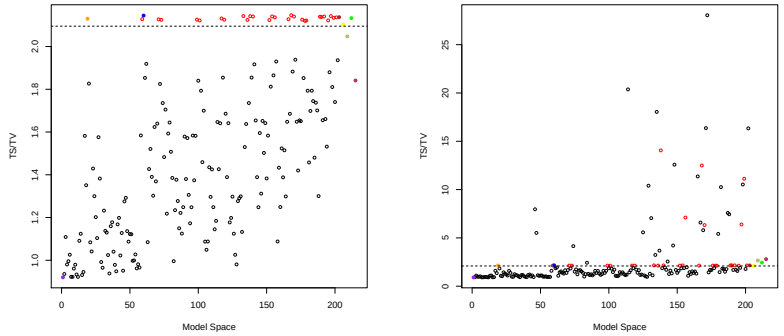
E(To) = 0.138 **0.206** **0.413**



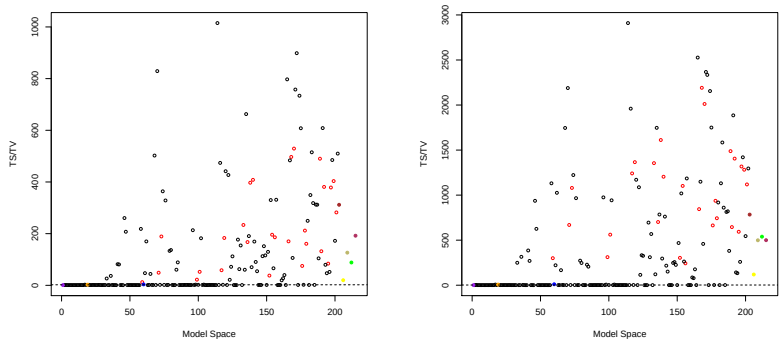
E(To) = 0.619 **0.826** **1.376**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.2: Ts/Tv Estimates on HKY85 Data with Sequence Length 250 and E(To) = 1.72 to 4.128



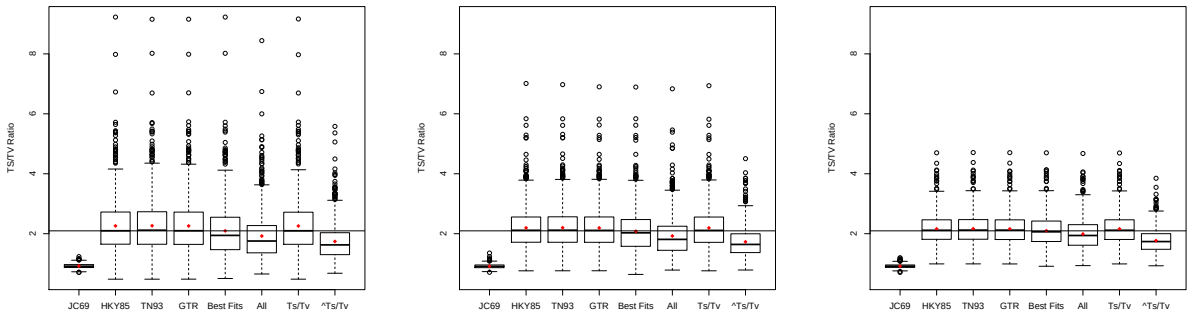
E(To) = 1.720 **2.064**



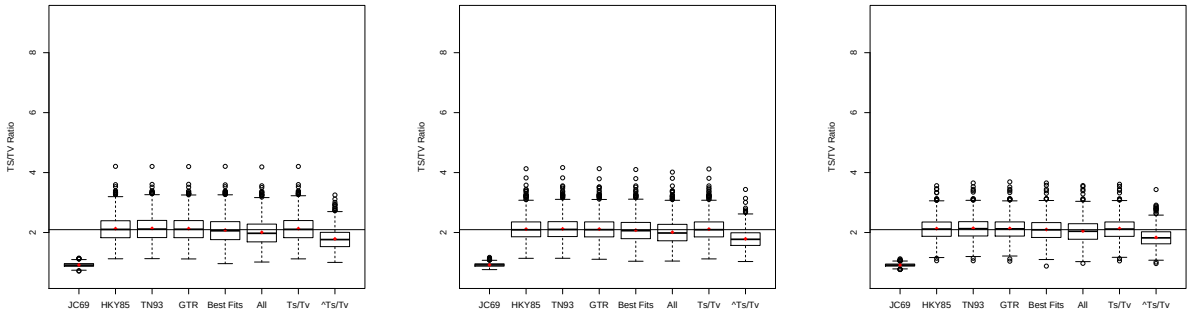
E(To) = 3.096 **4.128**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.3: Ts/Tv Box Plots on HKY85 Data with Sequence Length 250 and E(To) = 0.069 to 1.376



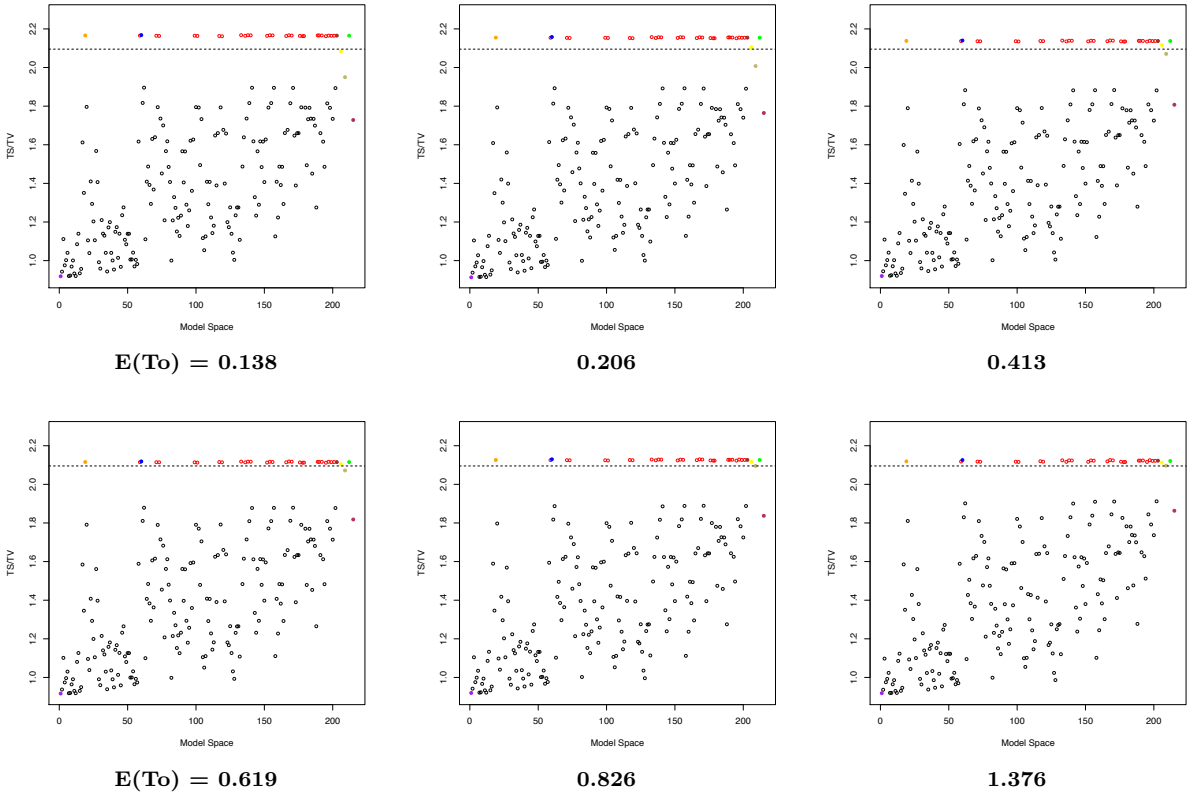
E(To) = 0.138 **0.206** **0.413**



E(To) = 0.619 **0.826** **1.38**

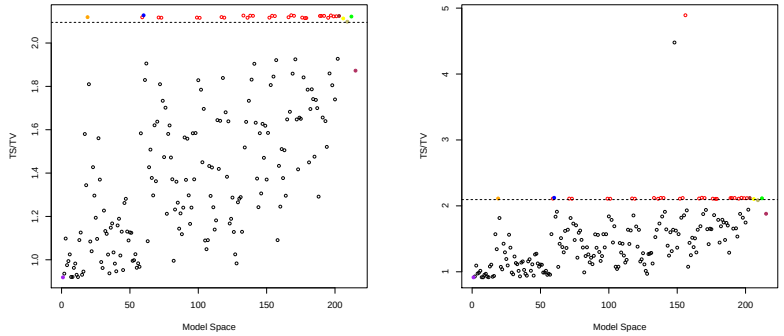
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.4: Ts/Tv Estimates on HKY85 Data with Sequence Length 500 and E(To) = 0.138 to 1.376

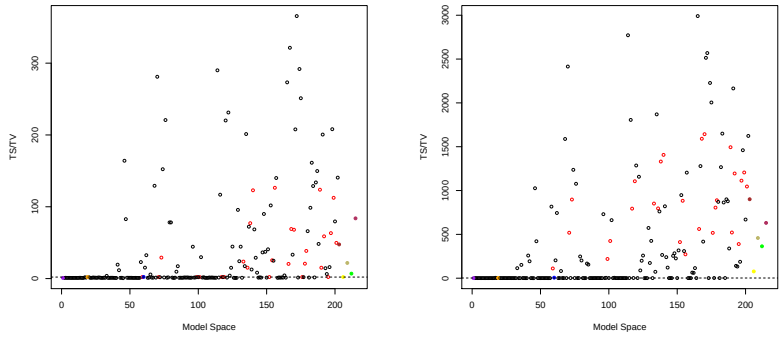


Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.5: Ts/Tv Estimates on HKY85 Data with Sequence Length 500 and E(To) = 1.72 to 4.128



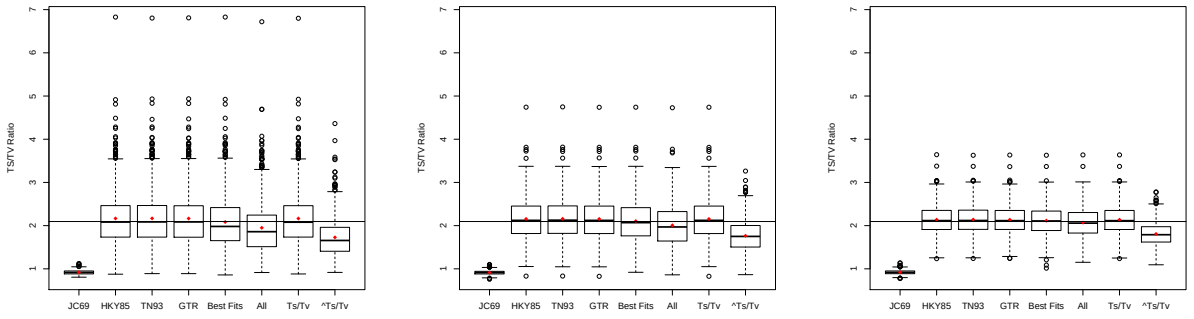
E(To) = 1.720 **2.064**



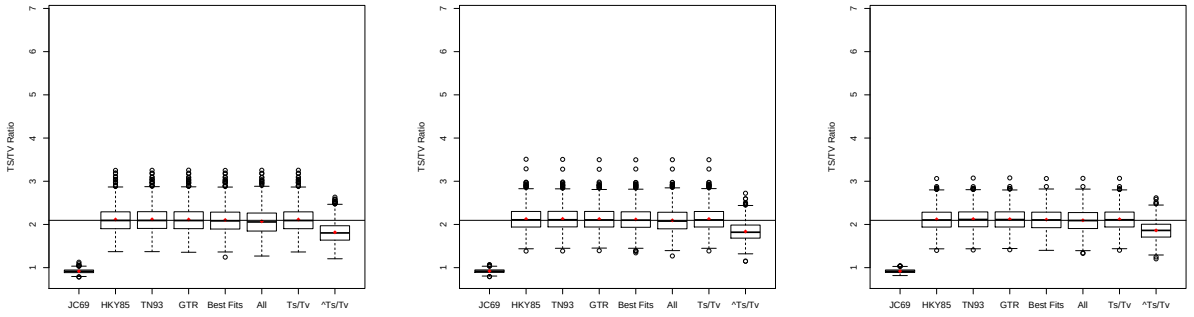
E(To) = 3.096 **4.128**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.6: Ts/Tv Box Plots on HKY85 Data with Sequence Length 500 and E(To) = 0.069 to 1.376



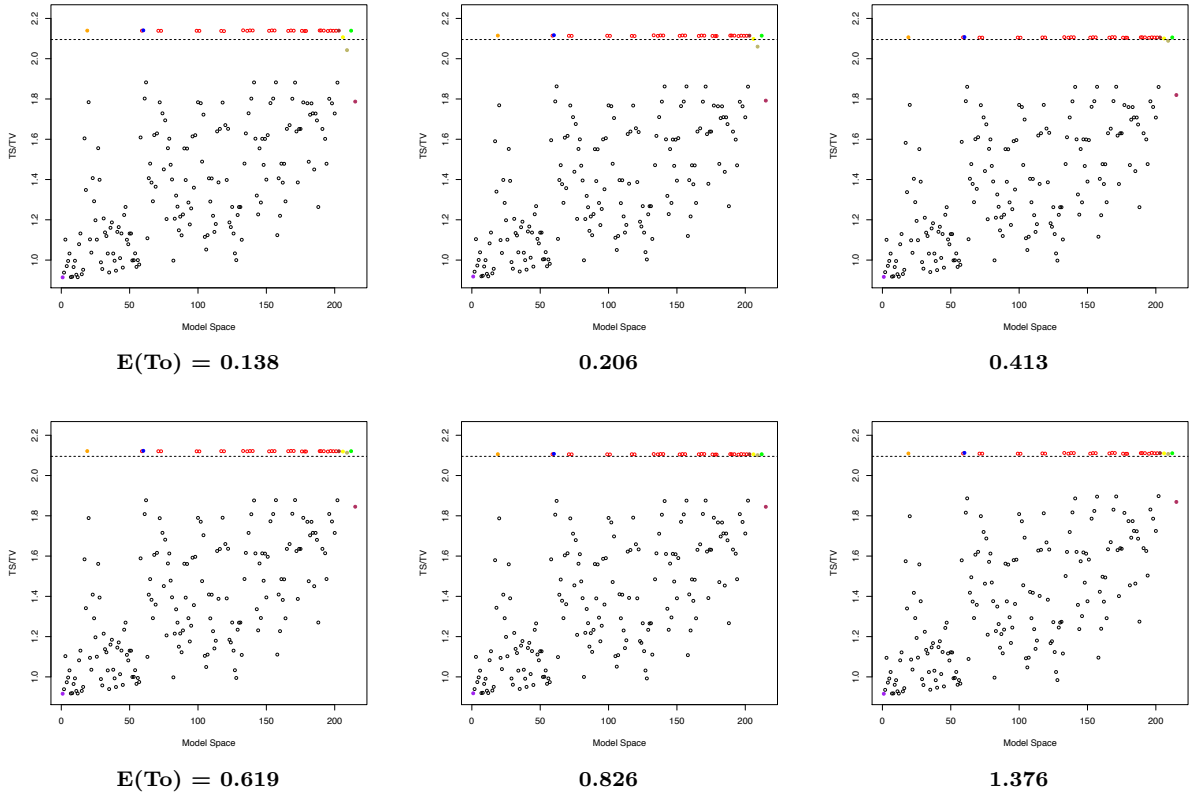
E(To) = 0.138 **0.206** **0.413**



E(To) = 0.619 **0.826** **1.376**

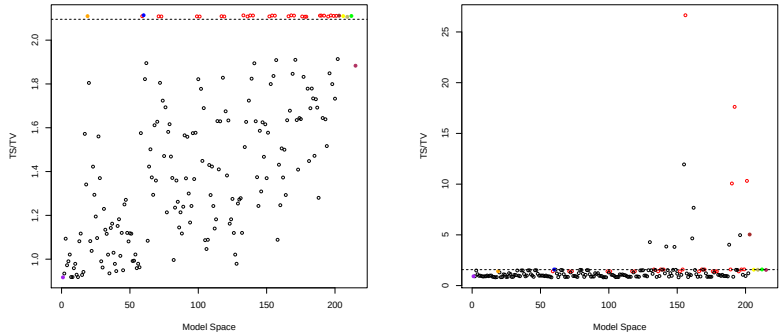
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.7: Ts/Tv Estimates on HKY85 Data with Sequence Length 1000 and E(To) = 0.138 to 1.376

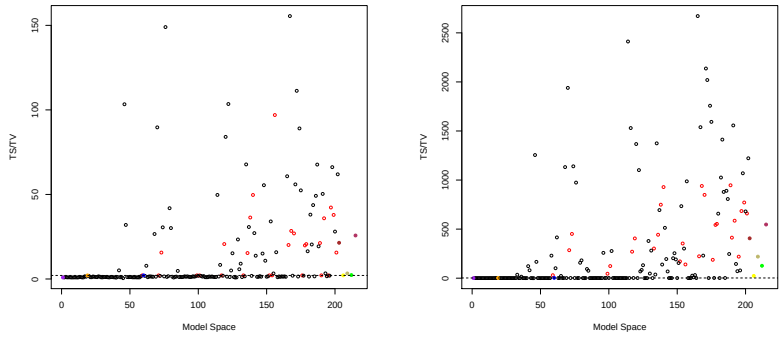


Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.8: Ts/Tv Estimates on HKY85 Data with Sequence Length 1000 and E(To) = 1.72 to 4.128



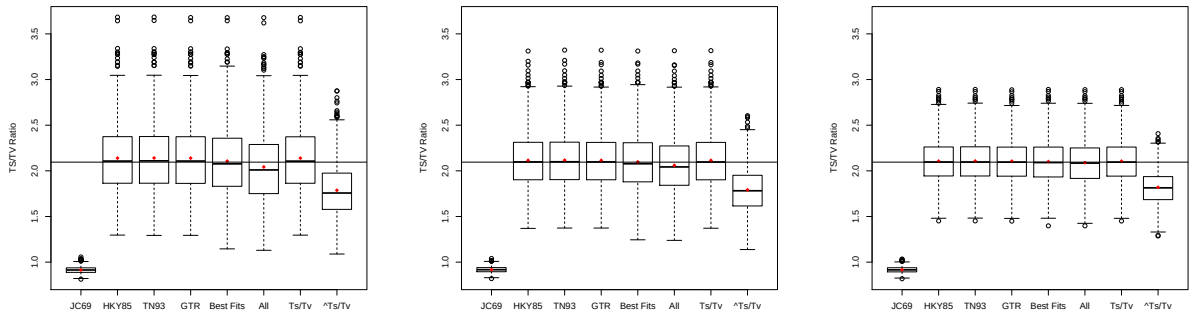
E(To) = 1.720 **2.064**



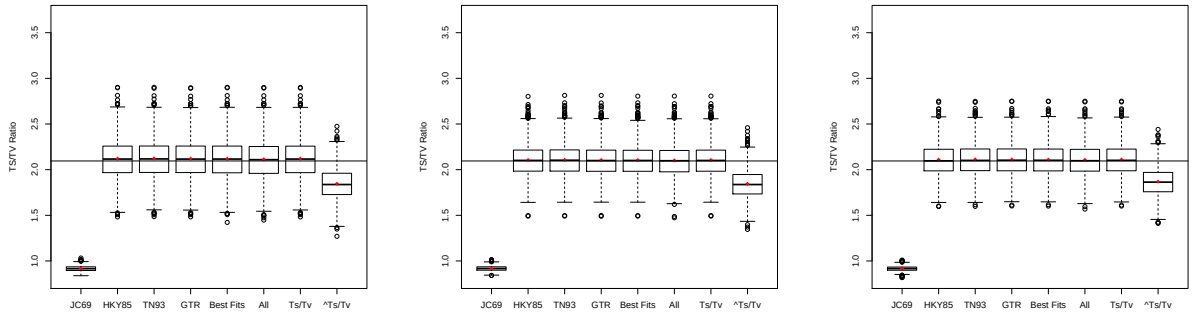
E(To) = 3.096 **4.128**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.9: Ts/Tv Box Plots on HKY85 Data with Sequence Length 1000 and E(To) = 0.069 to 1.376



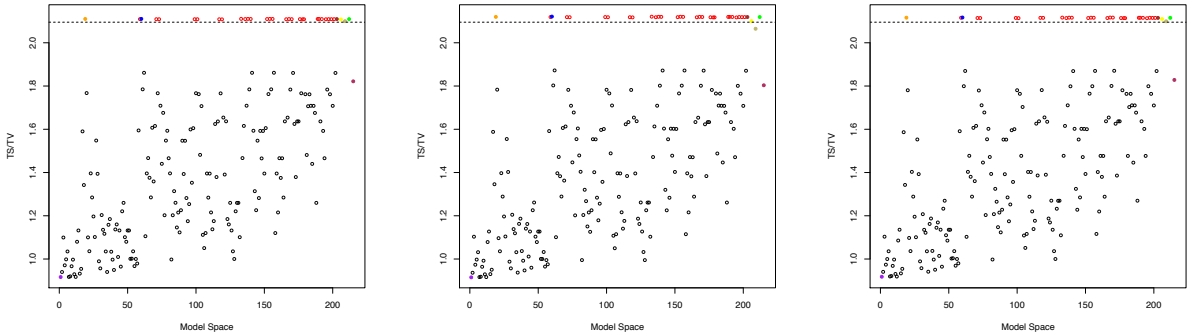
E(To) = 0.138 **0.206** **0.413**



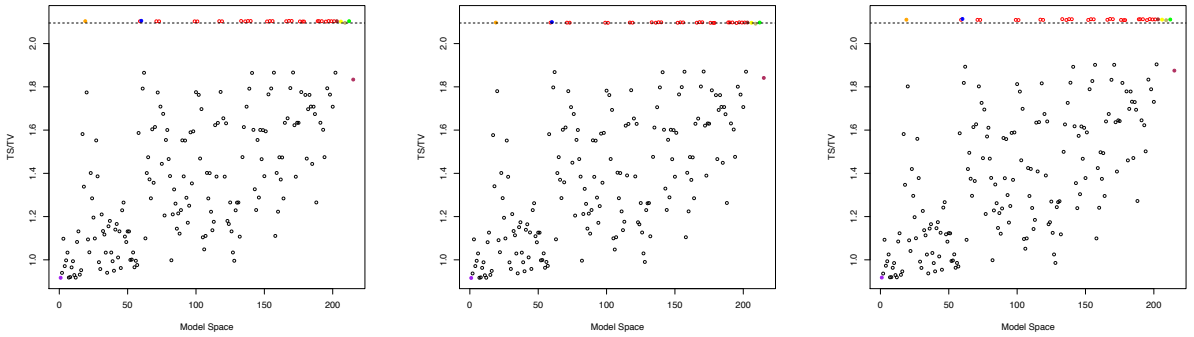
E(To) = 0.619 **0.826** **1.376**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.10: Ts/Tv Estimates on HKY85 Data with Sequence Length 3000 and E(To) = 0.138 to 1.376



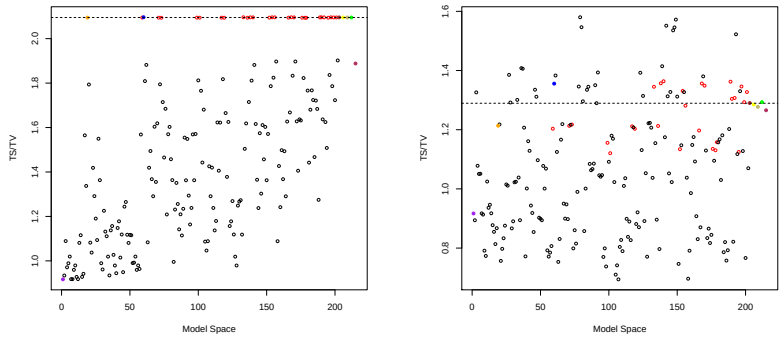
E(To) = 0.138 **0.206** **0.413**



E(To) = 0.619 **0.826** **1.376**

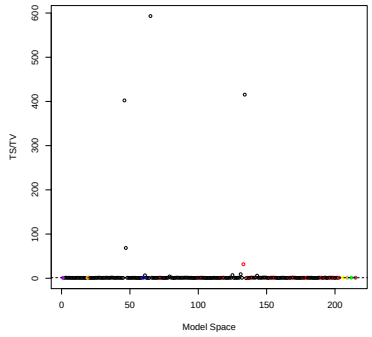
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.11: Ts/Tv Estimates on HKY85 Data with Sequence Length 3000 and E(To) = 1.72 to 4.128

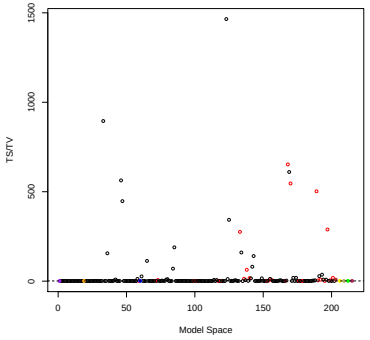


E(To) = 1.720

2.064



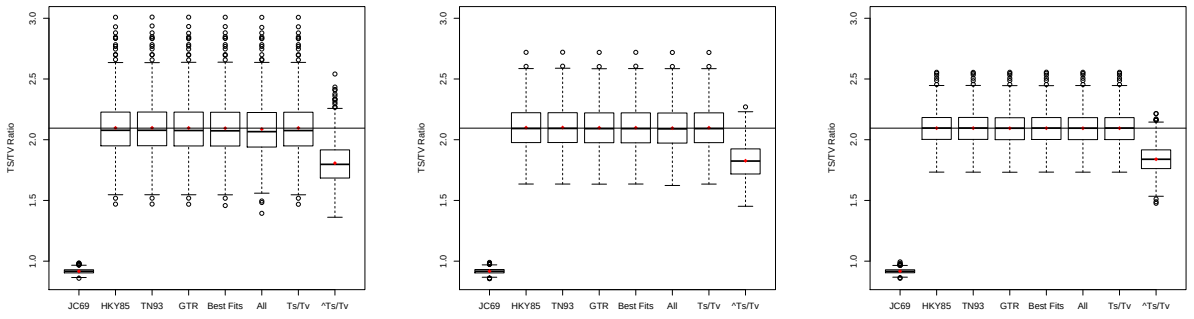
E(To) = 3.096



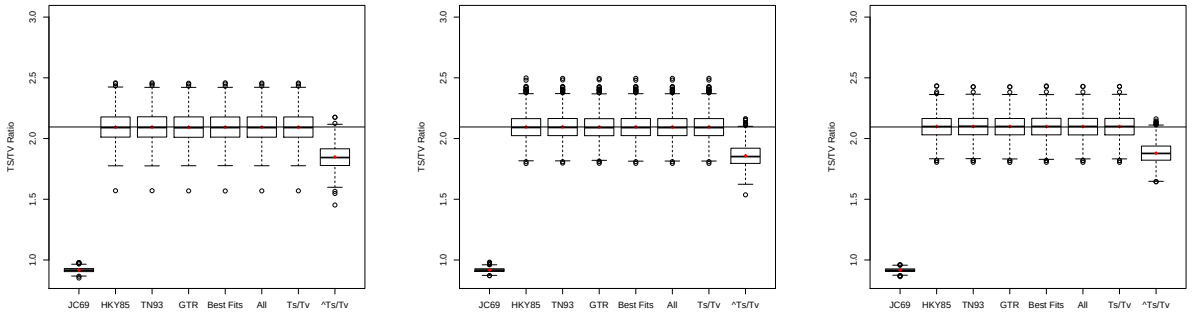
4.128

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.12: Ts/Tv Box Plots on HKY85 Data with Sequence Length 3000 and E(To) = 0.138 to 1.376



E(To) = 0.138 **0.206** **0.413**

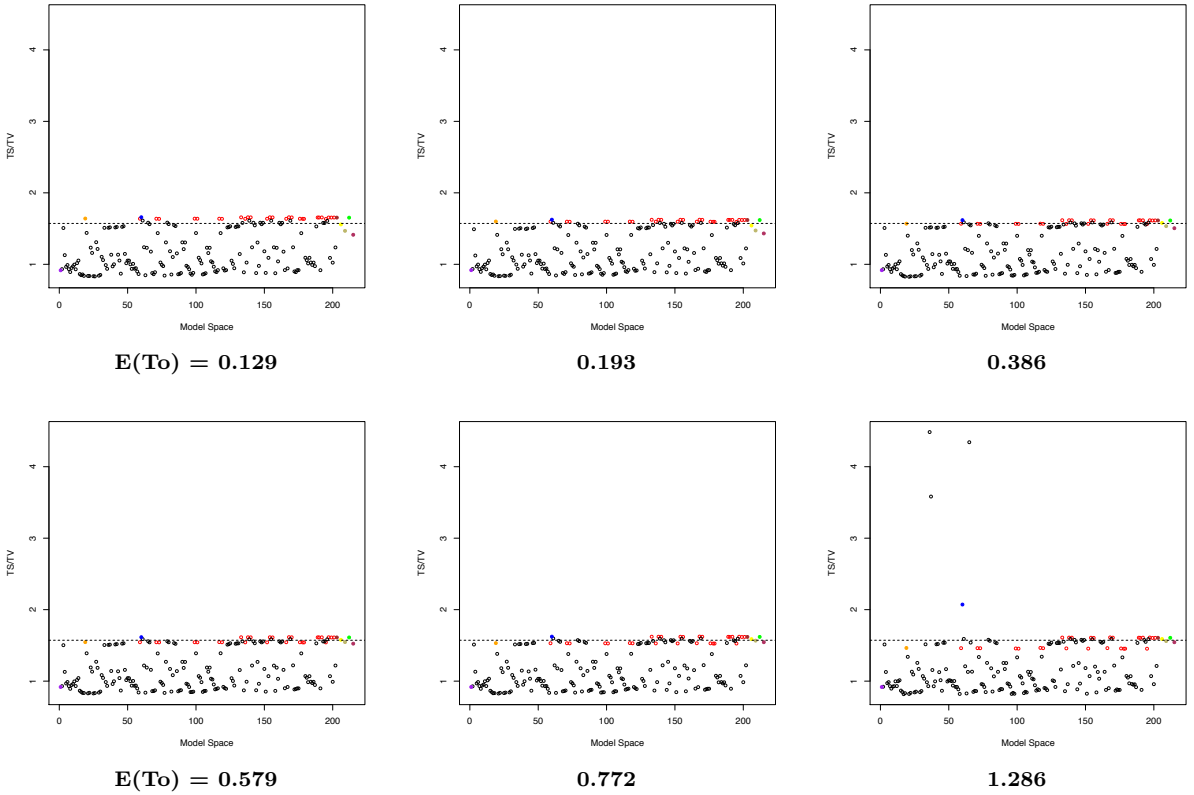


E(To) = 0.619 **0.826** **1.376**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

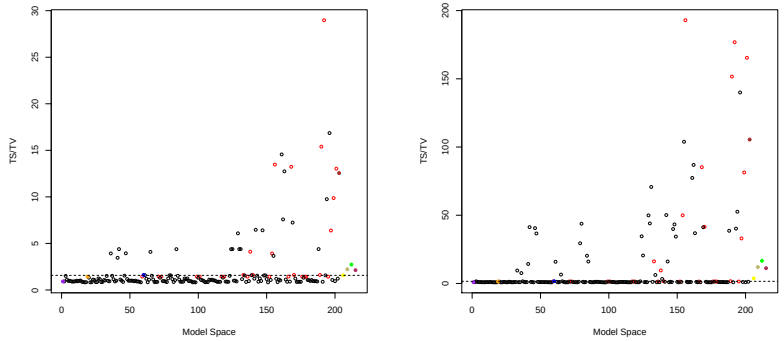
A.1.2 TN93 Data Analyses

Figure A.13: Ts/Tv Estimates on TN93 Data with Sequence Length 250 and E(To) = 0.129 to 1.286



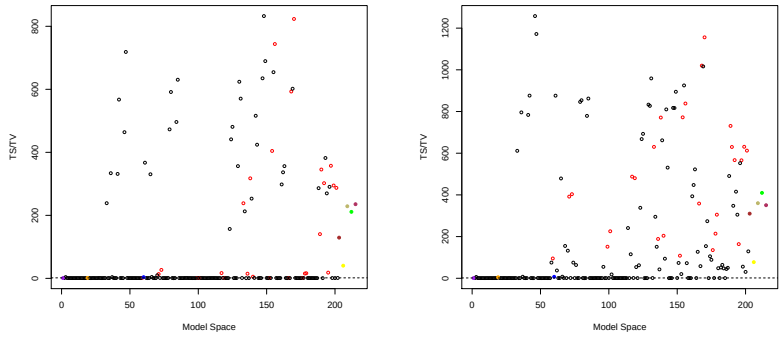
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.14: Ts/Tv Estimates on TN93 Data with Sequence Length 250 and E(To) = 1.607 to 3.858



E(To) = 1.607

1.928

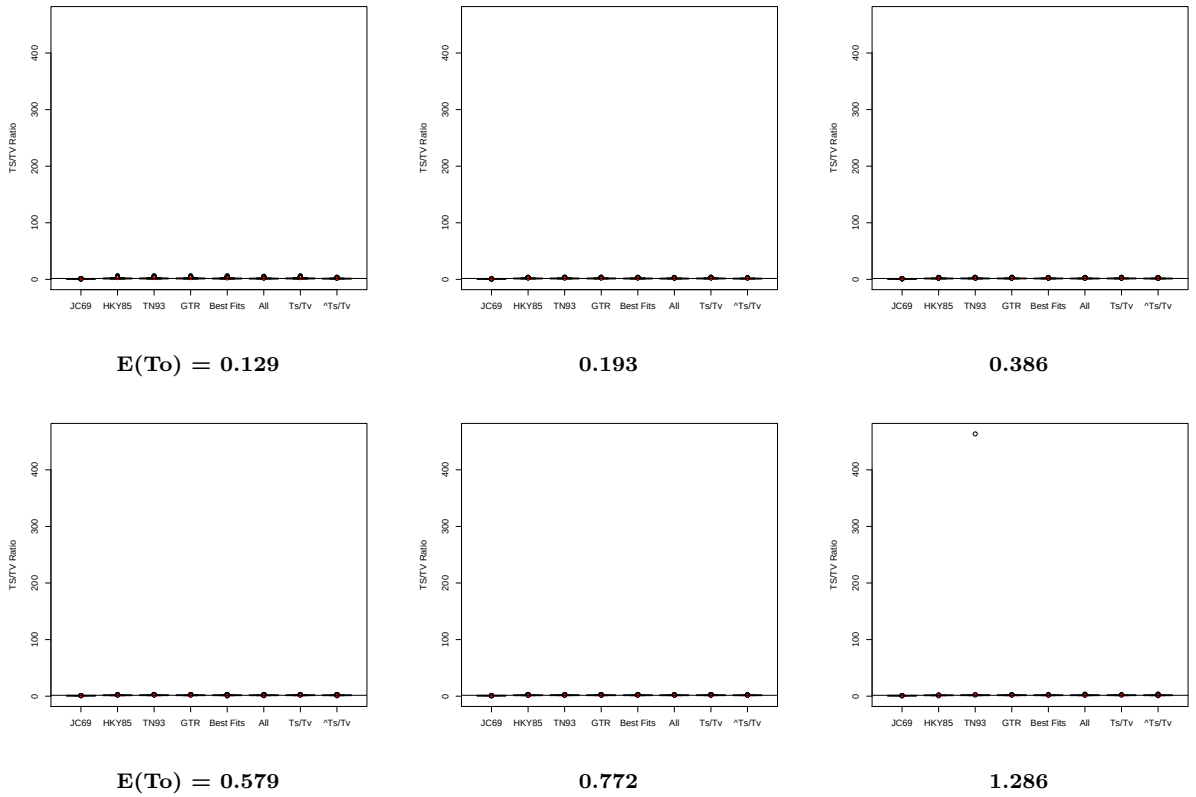


E(To) = 2.893

3.858

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.15: Ts/Tv Box Plots on TN93 Data with Sequence Length 250 and E(To) = 0.129 to 1.286

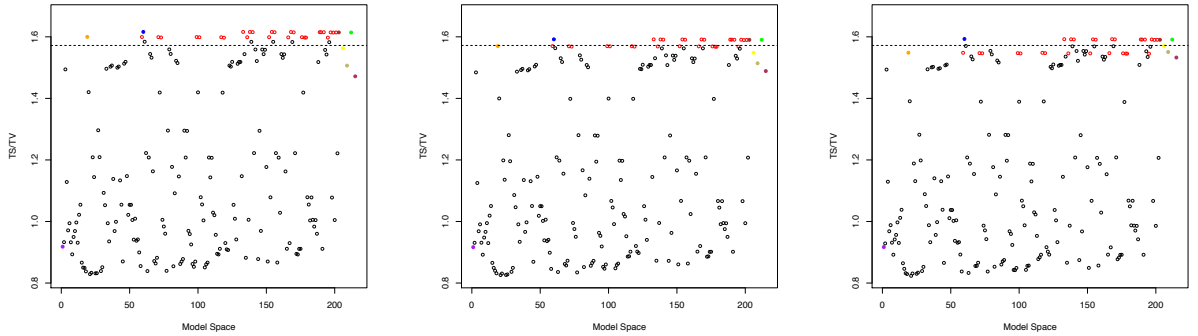


E(To) = 0.129 **0.193** **0.386**

E(To) = 0.579 **0.772** **1.286**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.16: Ts/Tv Estimates on TN93 Data with Sequence Length 500 and E(To) = 0.129 to 1.286

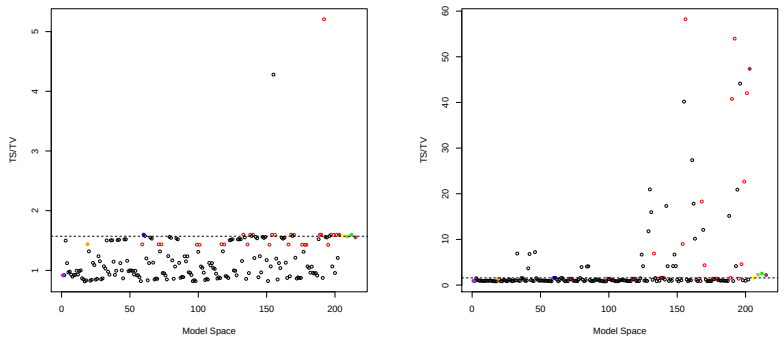


E(To) = 0.129 **0.193** **0.386**

E(To) = 0.579 **0.772** **1.286**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.17: Ts/Tv Estimates on TN93 Data with Sequence Length 500 and E(To) = 1.607 to 3.858

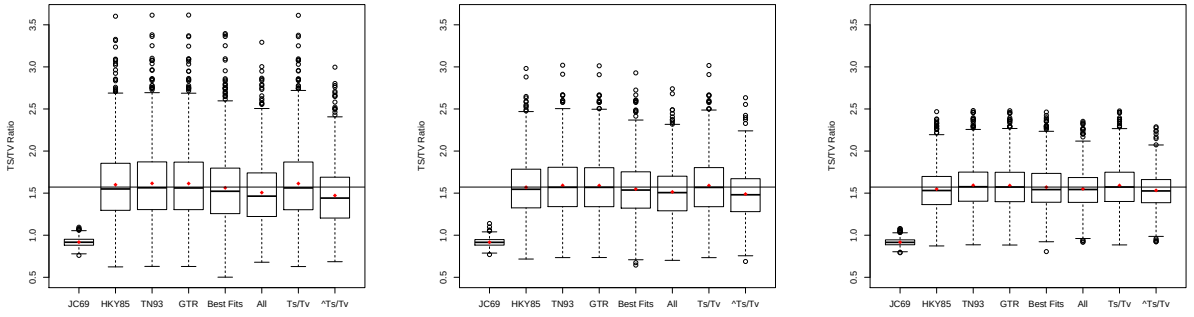


E(To) = 1.607 **1.928**

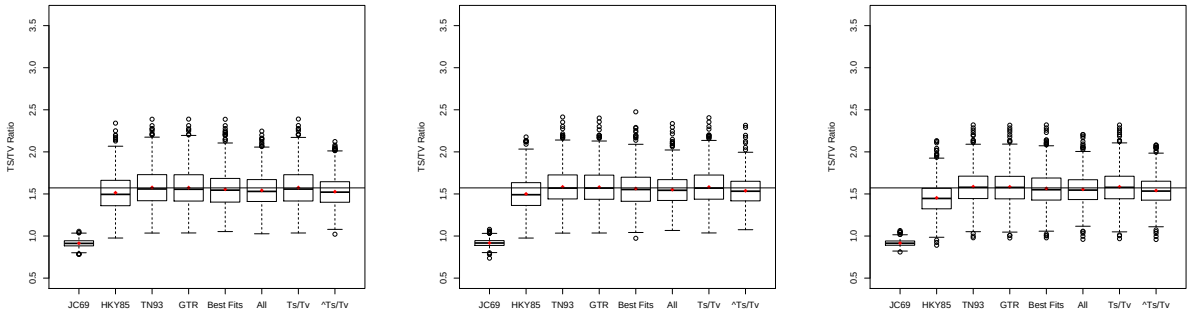
E(To) = 2.893 **3.858**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.18: Ts/Tv Box Plots on TN93 Data with Sequence Length 500 and E(To) = 0.129 to 1.286



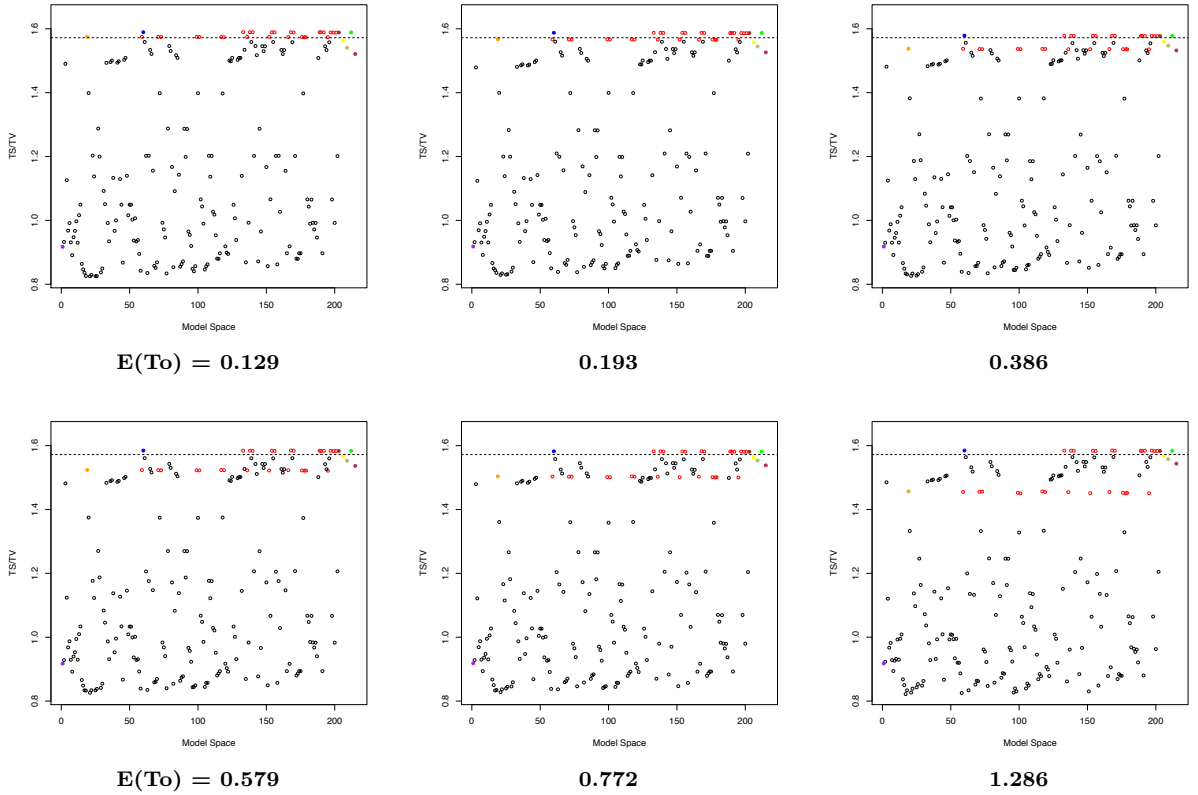
E(To) = 0.129 **0.193** **0.386**



E(To) = 0.579 **0.772** **1.286**

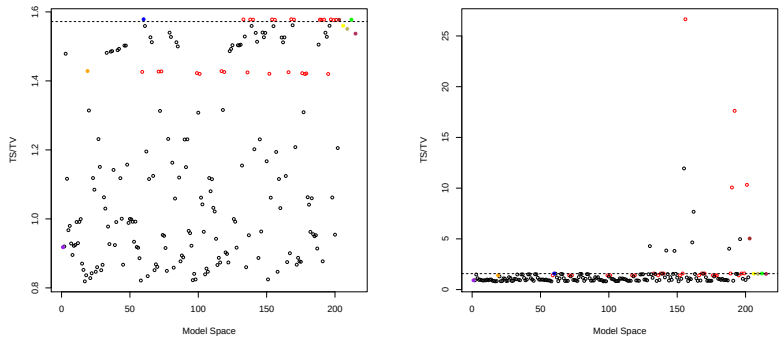
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.19: Ts/Tv Estimates on TN93 Data with Sequence Length 1000 and E(To) = 0.129 to 0.286



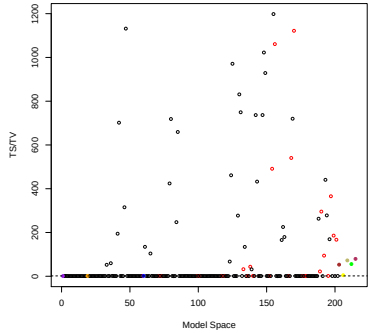
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.20: Ts/Tv Estimates on TN93 Data with Sequence Length 1000 and E(To) = 1.607 to 3.858

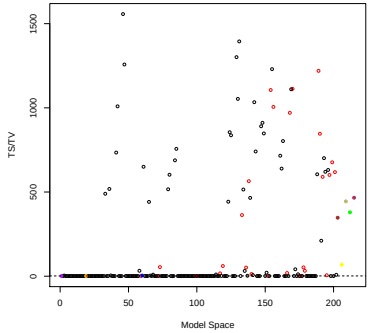


E(To) = 1.607

1.928



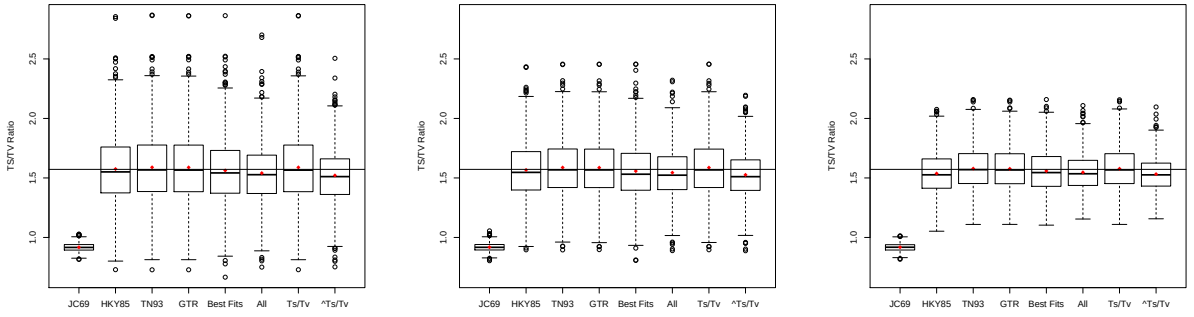
E(To) = 2.893



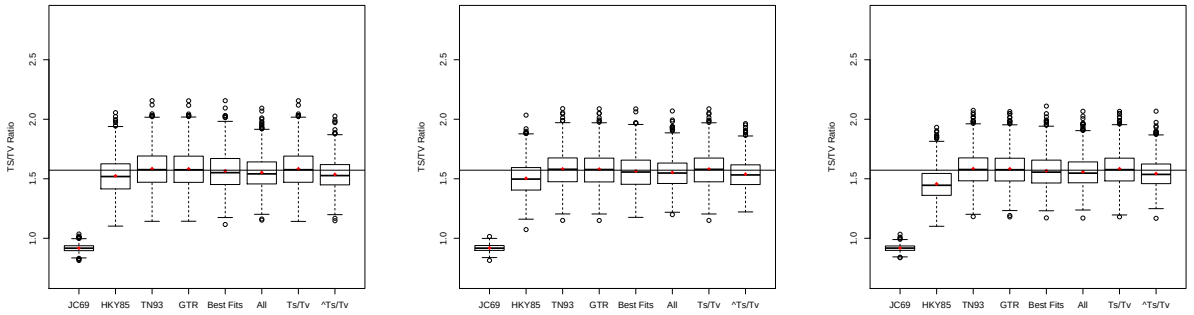
3.858

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.21: Ts/Tv Box Plots on TN93 Data with Sequence Length 1000 and E(To) = 0.129 to 1.286



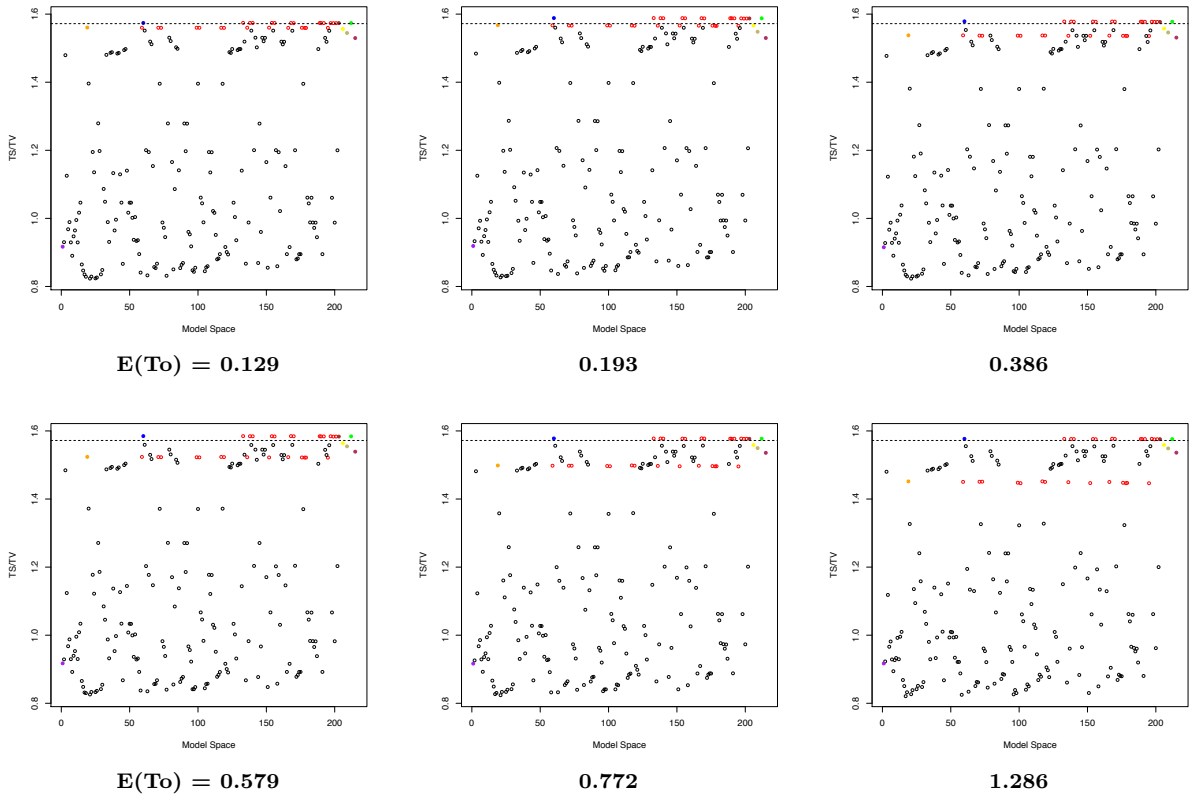
E(To) = 0.129 **0.193** **0.386**



E(To) = 0.579 **0.772** **1.286**

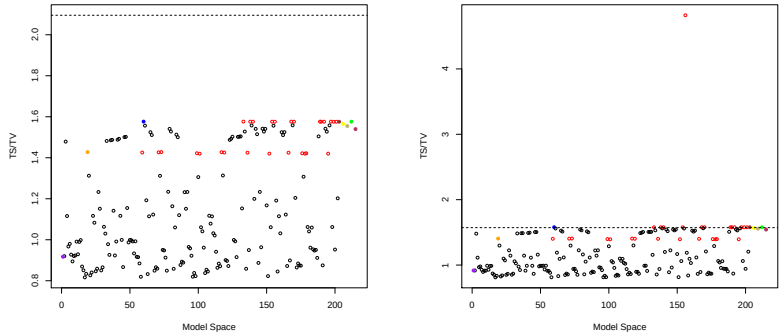
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.22: Ts/Tv Estimates on TN93 Data with Sequence Length 3000 and E(To) = 0.129 to 1.286

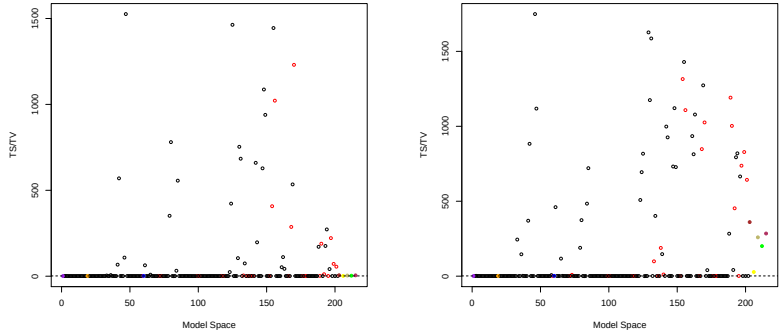


Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.23: Ts/Tv Estimates on TN93 Data with Sequence Length 3000 and E(To) = 1.607 to 3.858



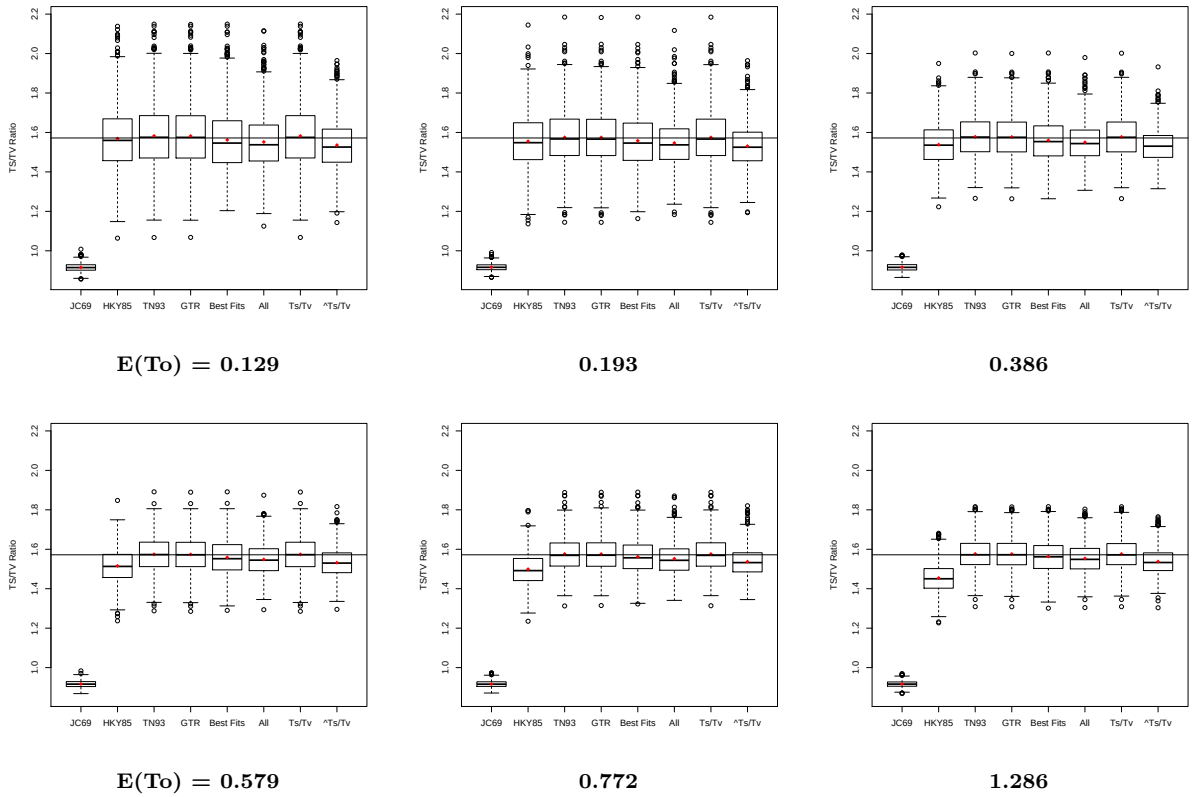
E(To) = 1.607 **1.928**



E(To) = 2.893 **3.858**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.24: Ts/Tv Box Plots on TN93 Data with Sequence Length 3000 and E(To) = 0.129 to 1.286



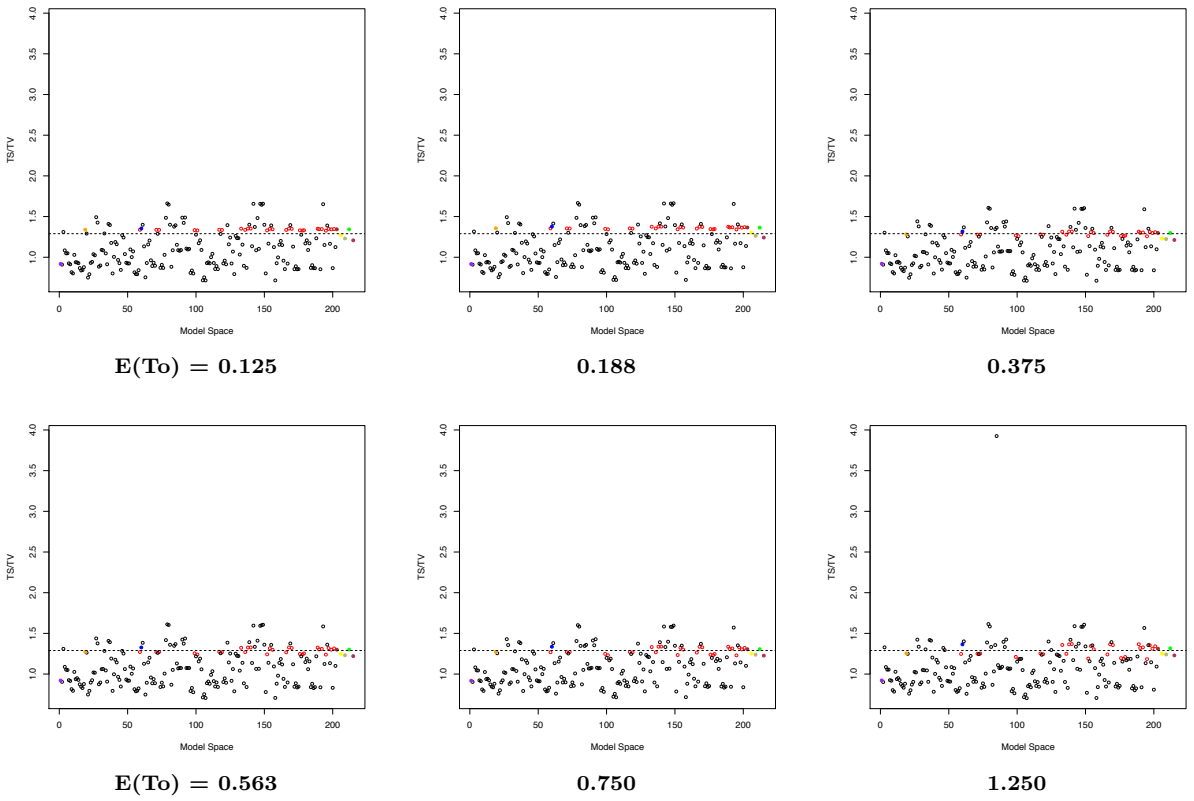
E(To) = 0.129 **0.193** **0.386**

E(To) = 0.579 **0.772** **1.286**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

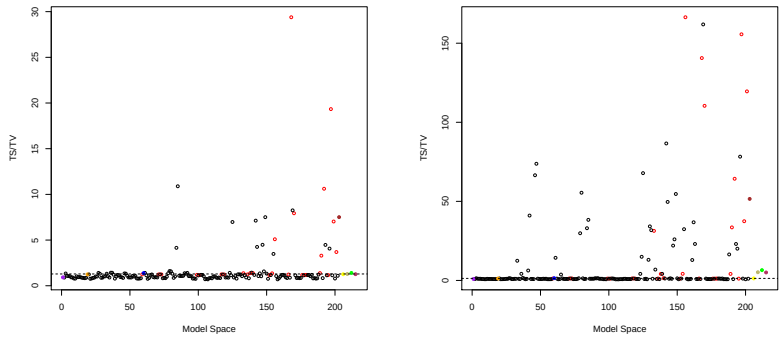
A.1.3 GTR Data Analyses

Figure A.25: Ts/Tv Estimates on GTR Data with Sequence Length 250 and E(To) = 0.125 to 1.250



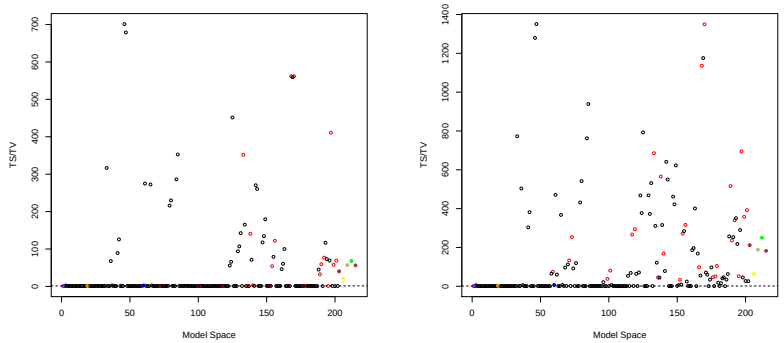
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.26: Ts/Tv Estimates on TN93 Data with Sequence Length 3000 and E(To) = 1.563 to 3.750



E(To) = 1.563

1.875

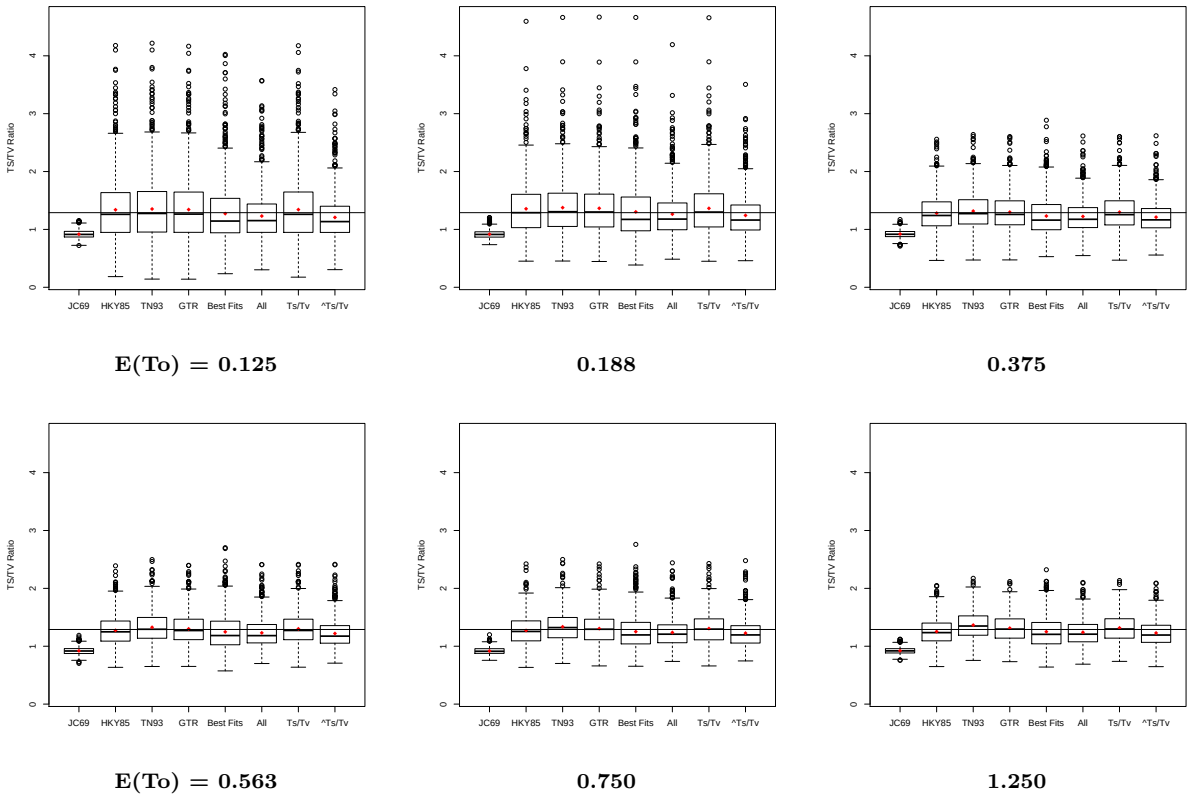


E(To) = 2.813

3.750

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.27: Ts/Tv Box Plots on GTR Data with Sequence Length 250 and E(To) = 0.125 to 1.250

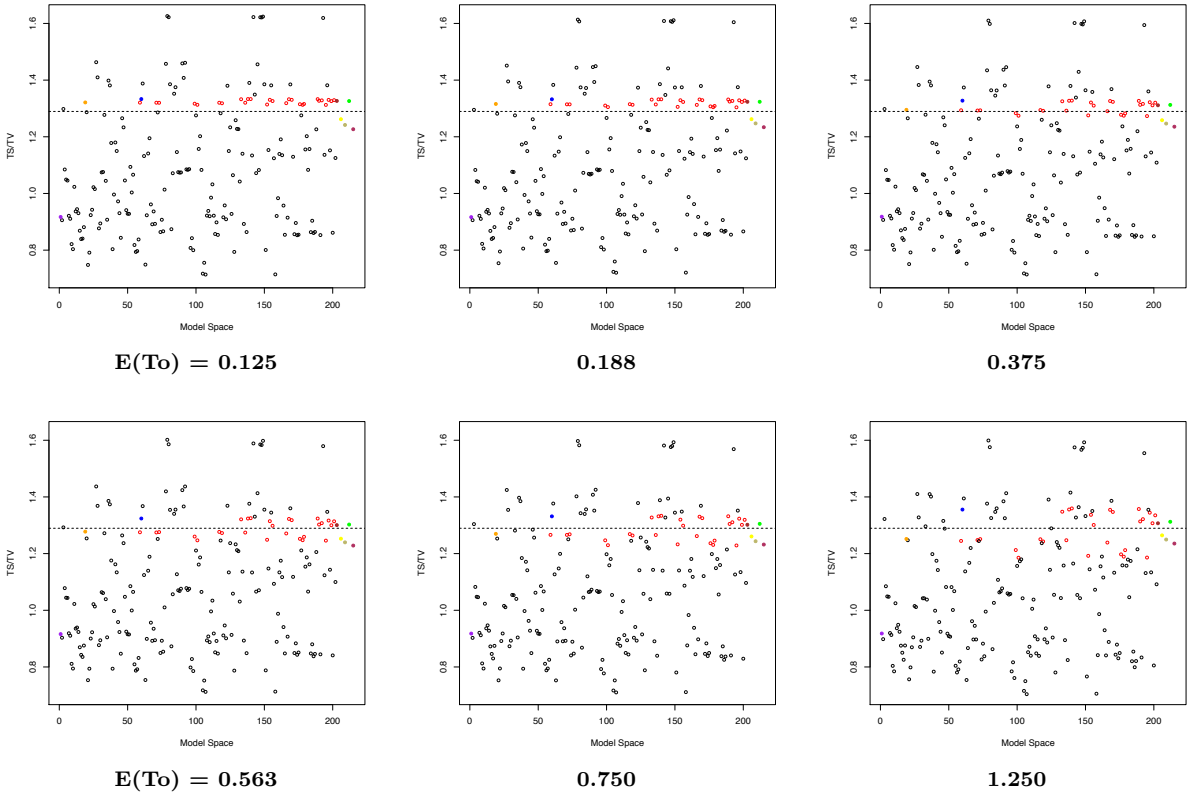


E(To) = 0.125 **0.188** **0.375**

E(To) = 0.563 **0.750** **1.250**

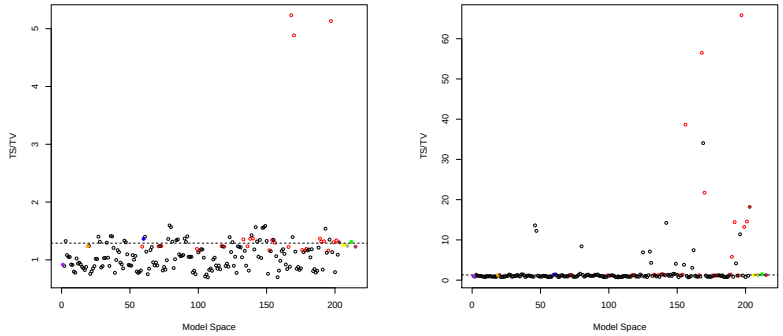
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.28: Ts/Tv Estimates on GTR Data with Sequence Length 500 and E(To) = 0.125 to 1.250



Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.29: Ts/Tv Estimates on GTR Data with Sequence Length 3000 and E(To) = 1.563 to 3.750

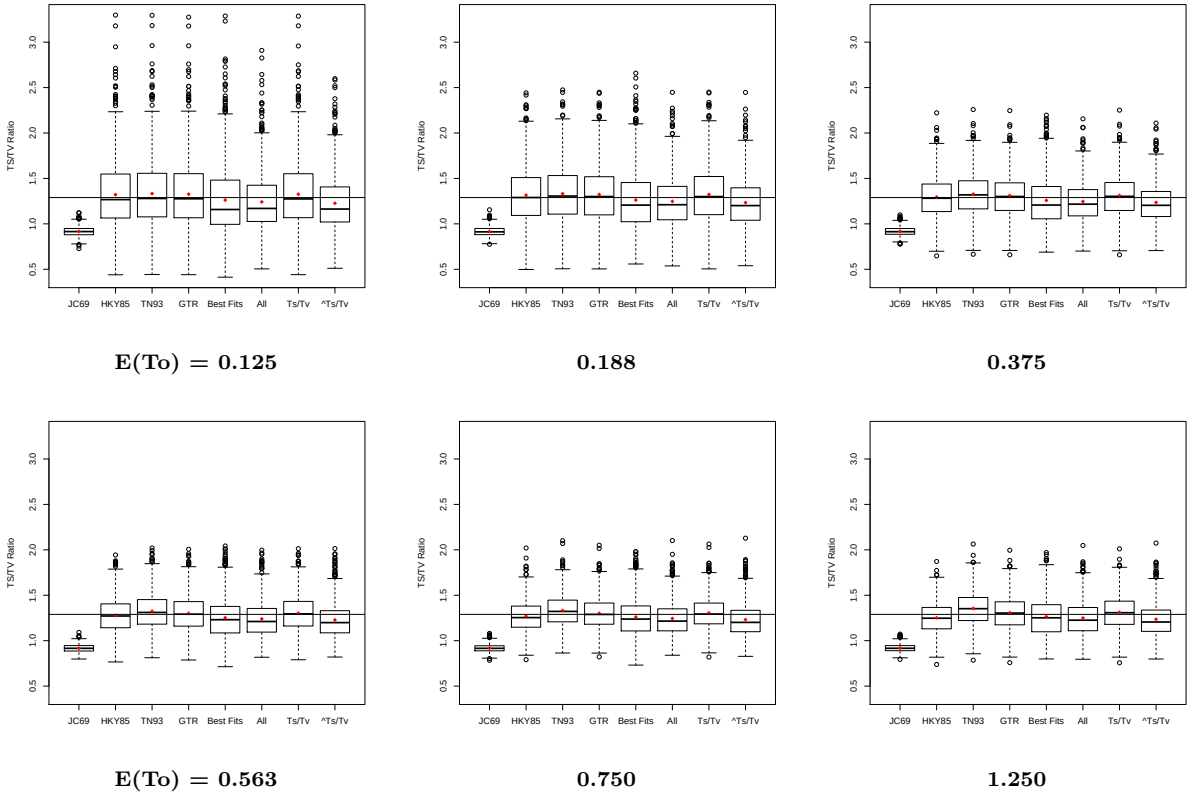


E(To) = 1.563 **1.875**

E(To) = 2.813 **3.750**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.30: Ts/Tv Box Plots on GTR Data with Sequence Length 500 and E(To) = 0.125 to 1.250

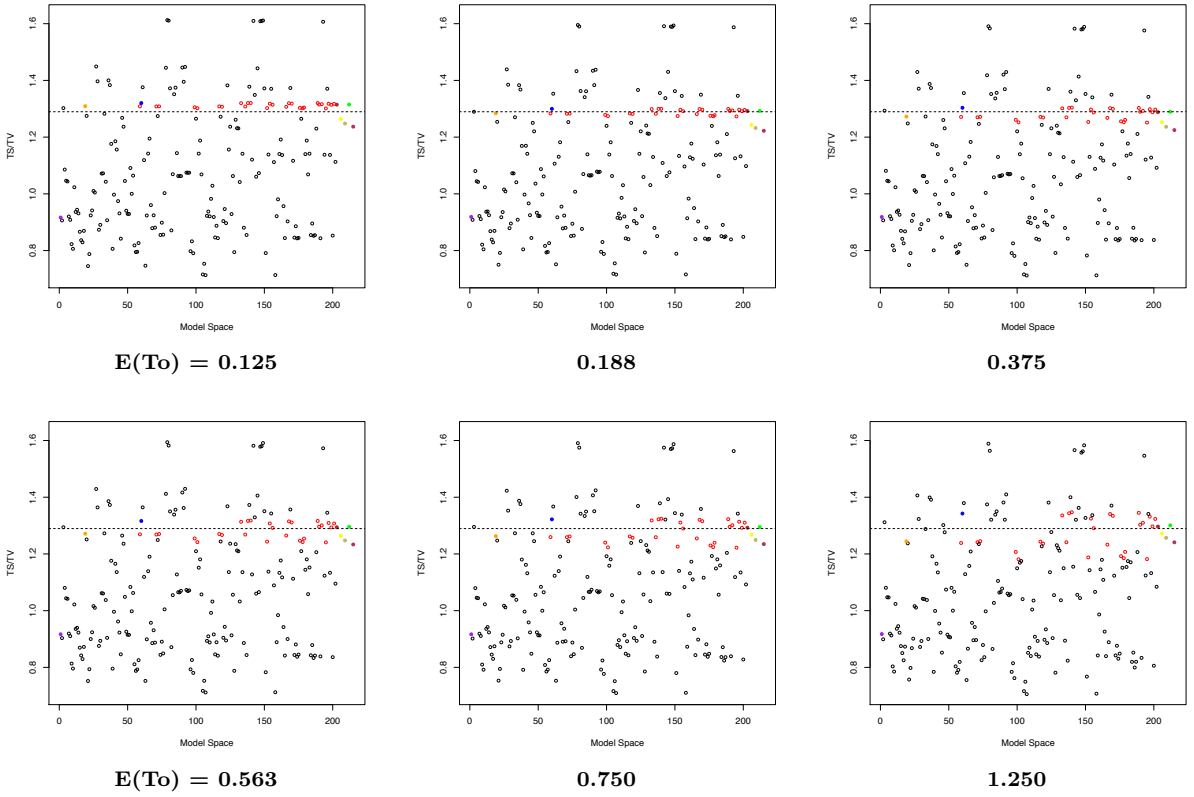


E(To) = 0.125 **0.188** **0.375**

E(To) = 0.563 **0.750** **1.250**

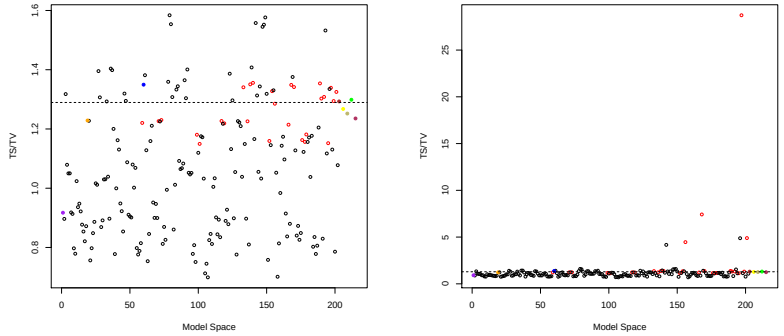
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.31: Ts/Tv Estimates on GTR Data with Sequence Length 1000 and E(To) = 0.125 to 1.250



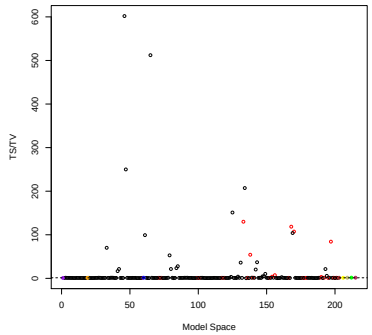
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.32: Ts/Tv Estimates on GTR Data with Sequence Length 3000 and E(To) = 1.563 to 3.750

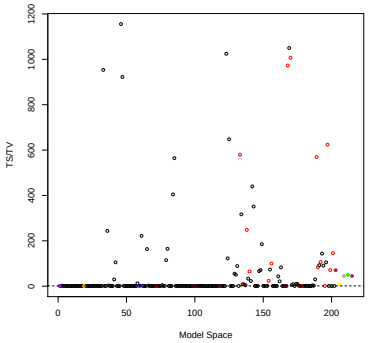


E(To) = 1.563

1.875



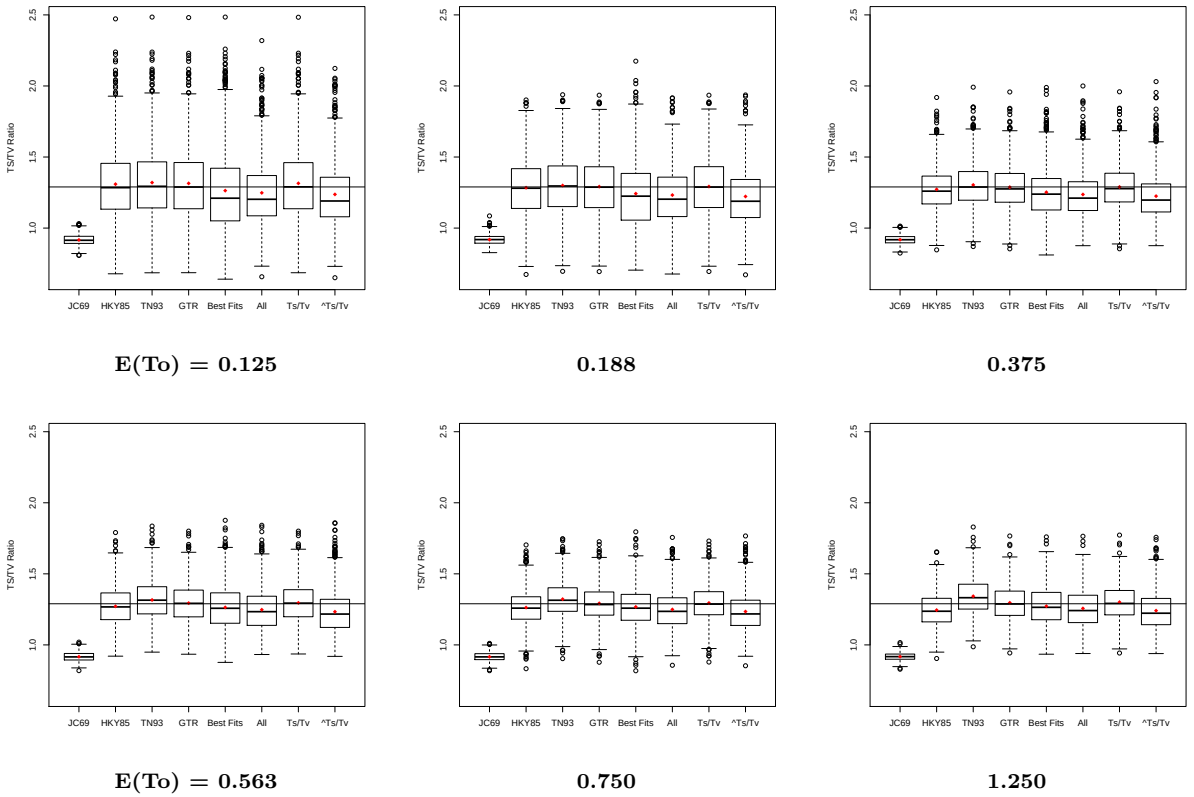
E(To) = 2.813



3.750

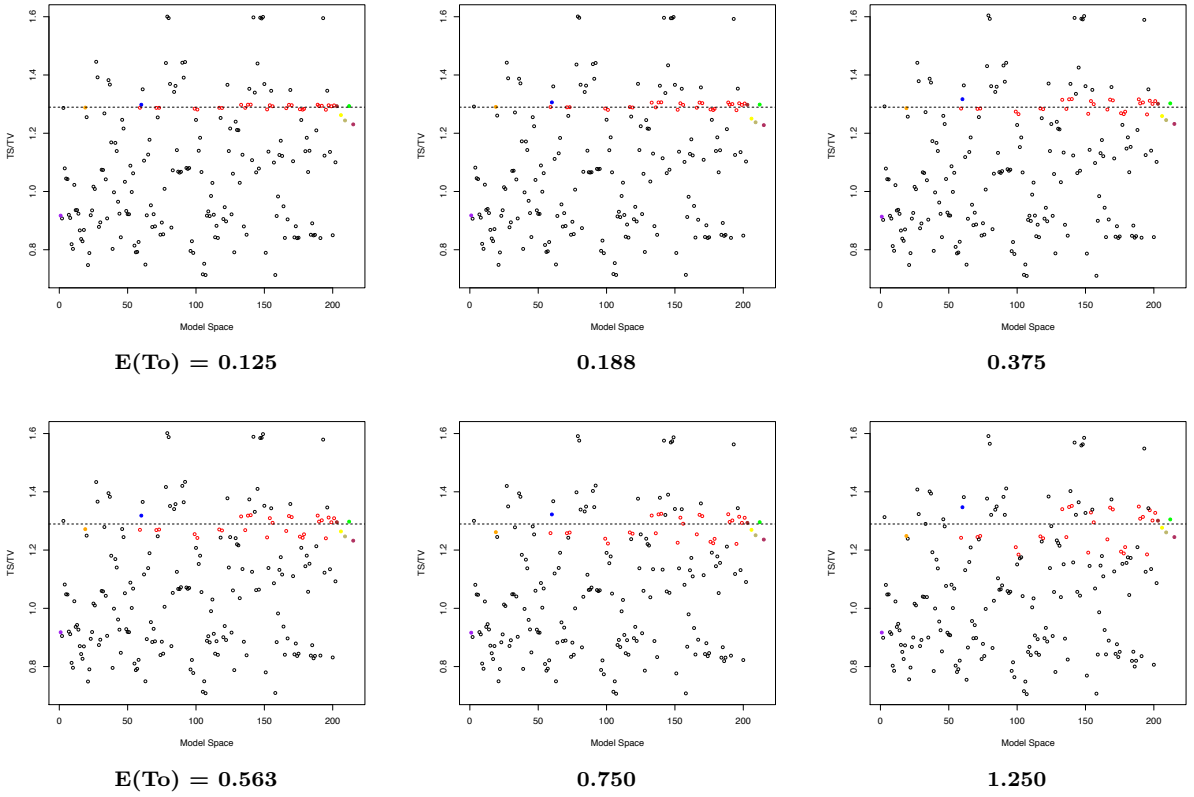
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.33: Ts/Tv Box Plots on GTR Data with Sequence Length 1000 and E(To) = 0.125 to 1.250



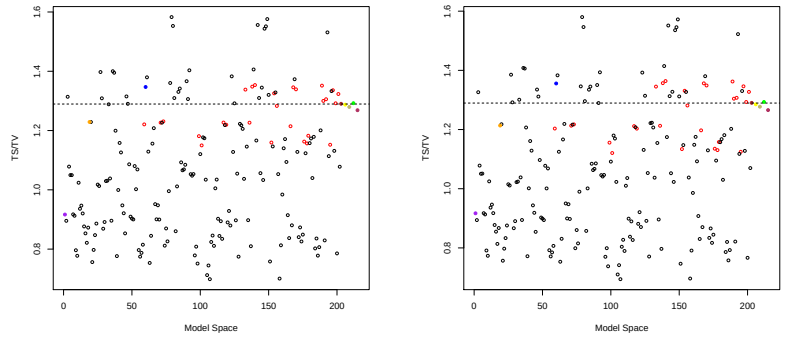
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.34: Ts/Tv Estimates on GTR Data with Sequence Length 3000 and E(To) = 0.125 to 1.250

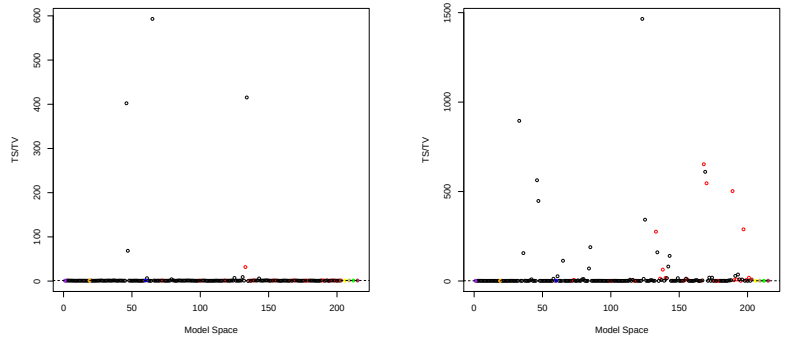


Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.35: Ts/Tv Estimates on GTR Data with Sequence Length 3000 with E(To) = 1.563 to 3.750



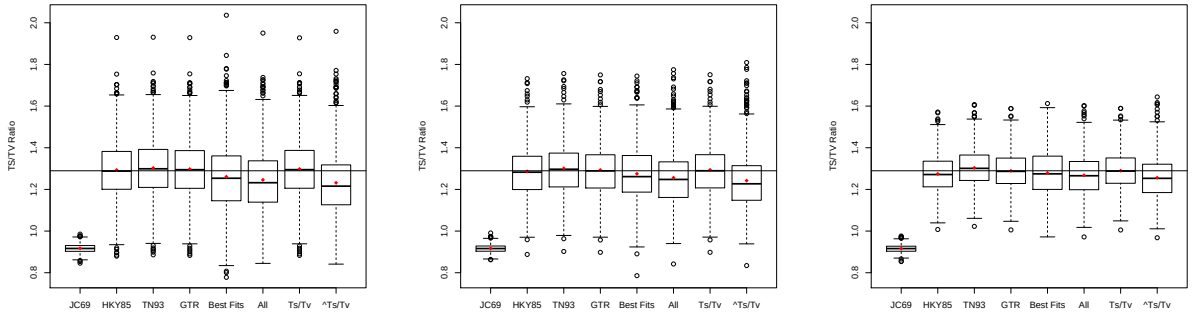
E(To) = 1.563 **1.875**



E(To) = 2.813 **3.750**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The solid black line across the figure represents the true value of Ts/TV used to simulate the data.
Note: Within the boxes, red dots represent the mean and the solid black line represents the median.

Figure A.36: Ts/Tv Box Plots on GTR Data with Sequence Length 3000 and E(To) = 0.125 to 1.250



E(To) = 0.125 **0.188** **0.375**

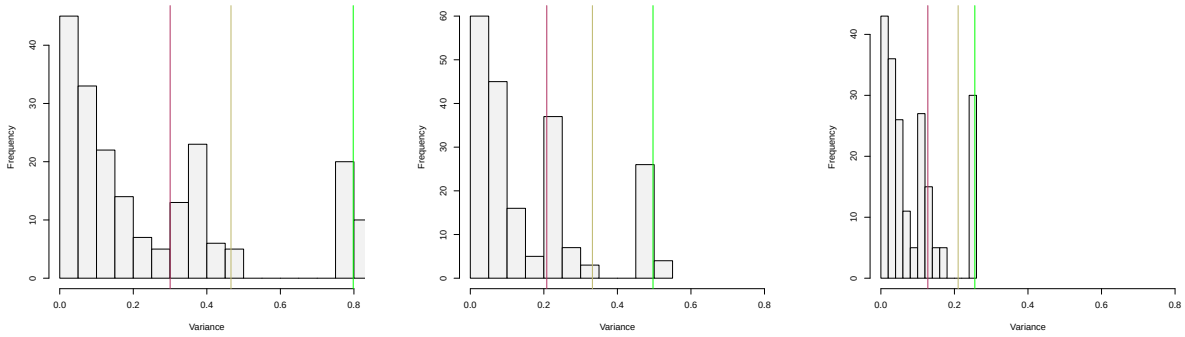
E(To) = 0.563 **0.750** **1.250**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Note: The dotted line represents the true value of Ts/TV used to simulate the data.
Note: Red = Ts/TV Models; Black = non-Ts/Tv Models; Purple = F81; Orange = HKY85; Blue = TN93; Brown = GTR; Yellow = Best Fit; Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

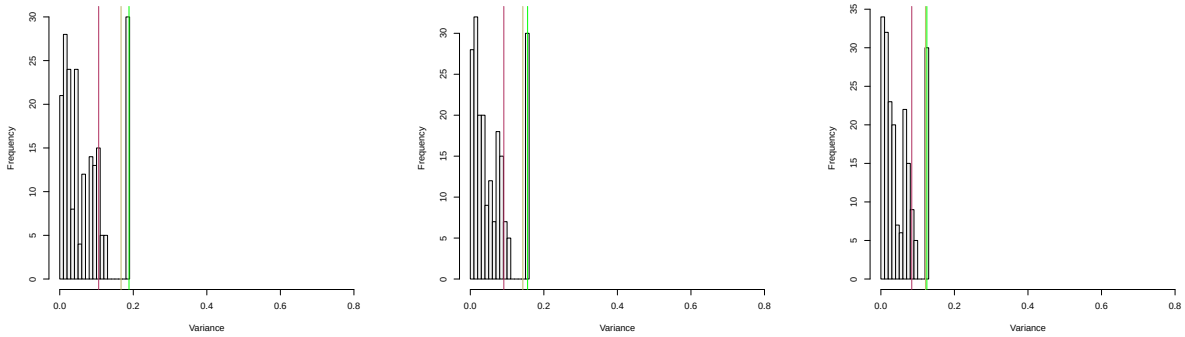
A.2 Variance of Transition/Transversion Estimates

A.2.1 HKY85 Variance Analyses

Figure A.37: Variance Histograms on HKY85 Data with Sequence Length 250



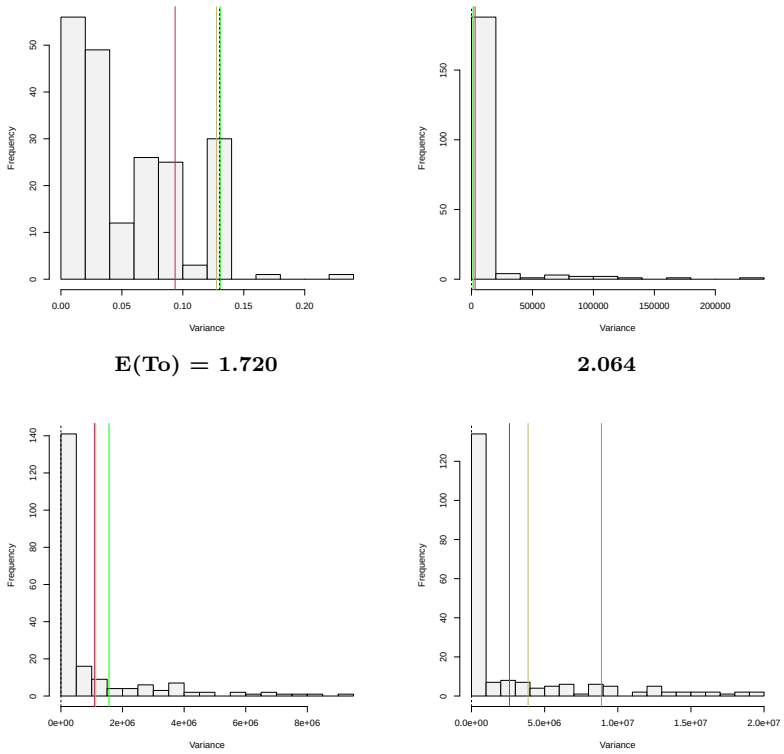
E(To) = 0.138 **0.206** **0.413**



E(To) = 0.619 **0.826** **1.376**

To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.38: Variance Histograms on HKY85 Data with Sequence Length 250 for E(To) = 1.72 to 4.128

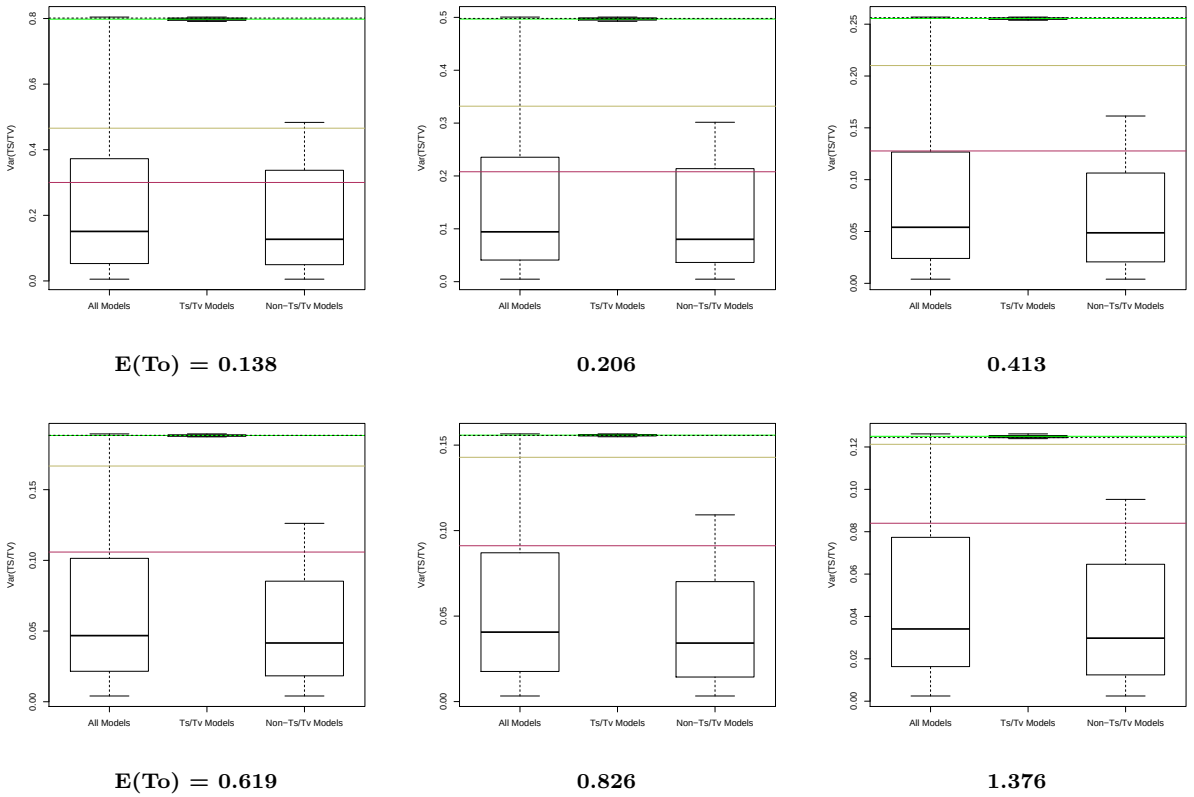


E(To) = 1.720 **2.064**

E(To) = 3.096 **4.128**

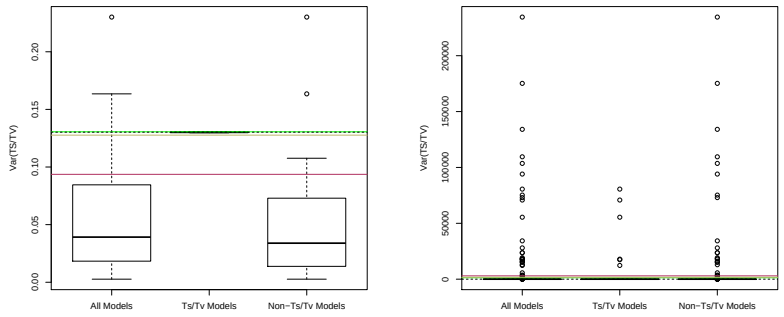
To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.39: Variance Boxplots on HKY85 Data with Sequence Length 250 for E(To) = 0.138 to 1.376



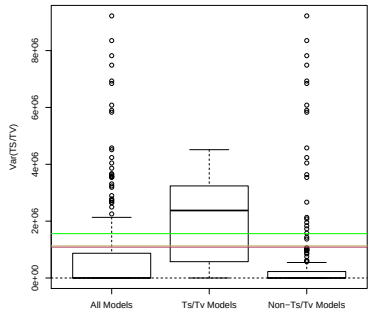
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

Figure A.40: Variance Boxplots on HKY85 Data with Sequence Length 250 and E(To) = 1.720 to 4.128

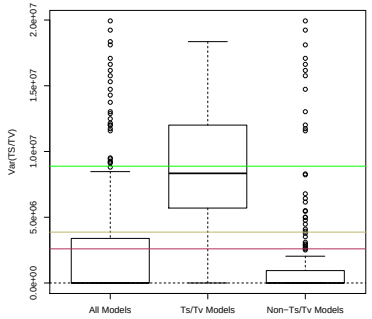


E(To) = 1.720

2.064



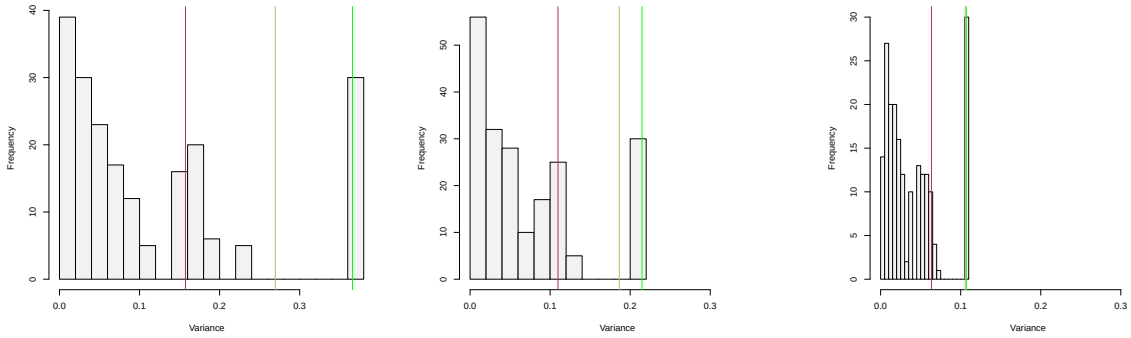
E(To) = 3.096



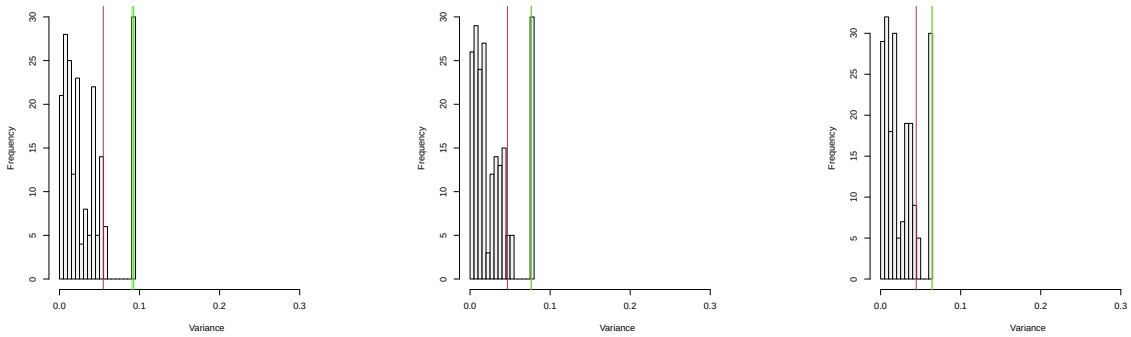
4.128

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

Figure A.41: Variance Histograms on HKY85 Data with Sequence Length 500 with E(To) = 0.138 to 1.376



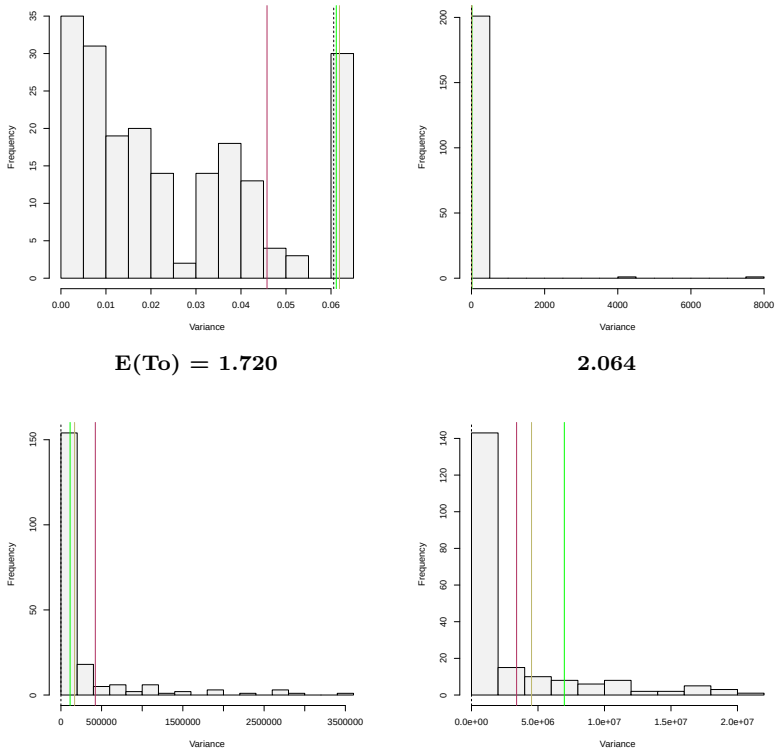
E(To) = 0.138 **0.206** **0.413**



E(To) = 0.619 **0.826** **1.376**

To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.42: Variance Histograms on HKY85 Data with Sequence Length 500 for E(To) = 1.72 to 4.128

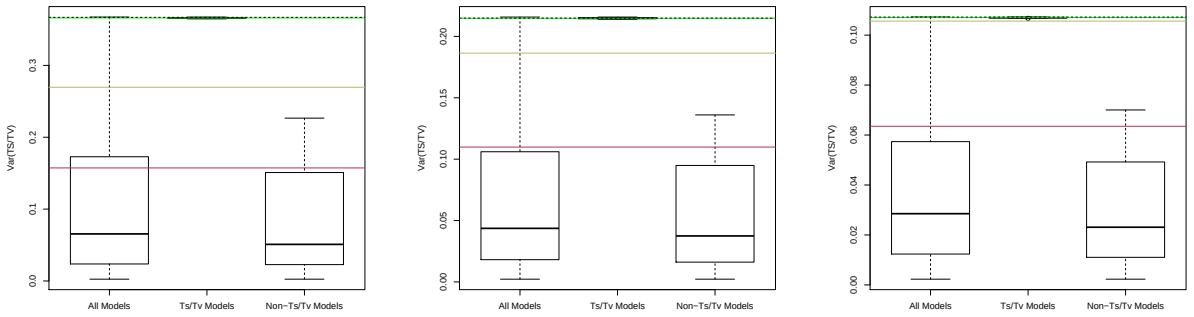


E(To) = 1.720 **2.064**

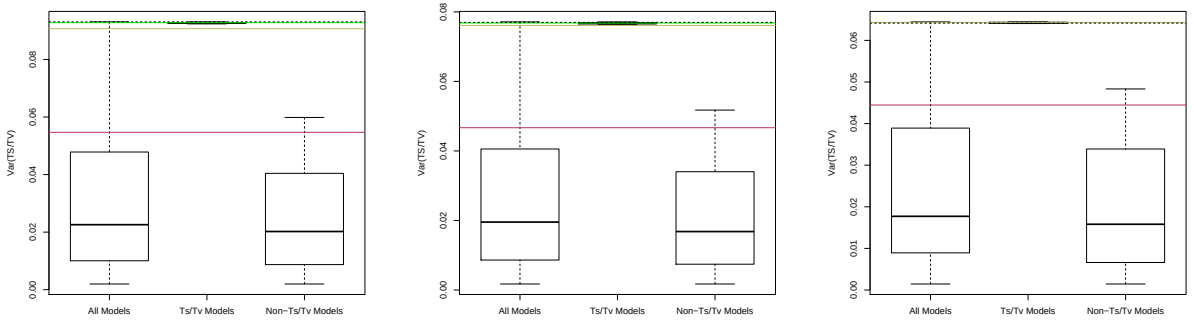
E(To) = 3.096 **4.128**

To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.43: Variance Boxplots on HKY85 Data with Sequence Length 500 with E(To) = 0.138 to 1.376



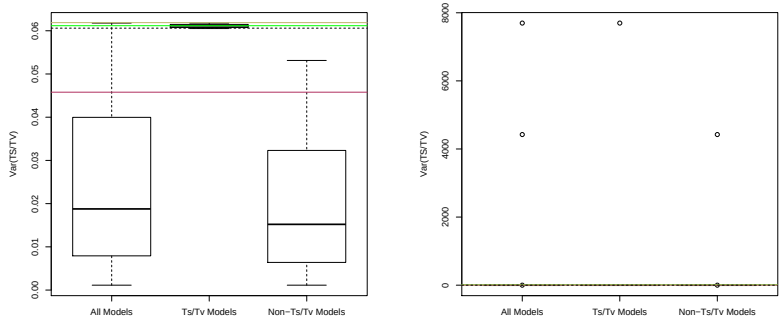
E(To) = 0.138 **0.206** **0.413**



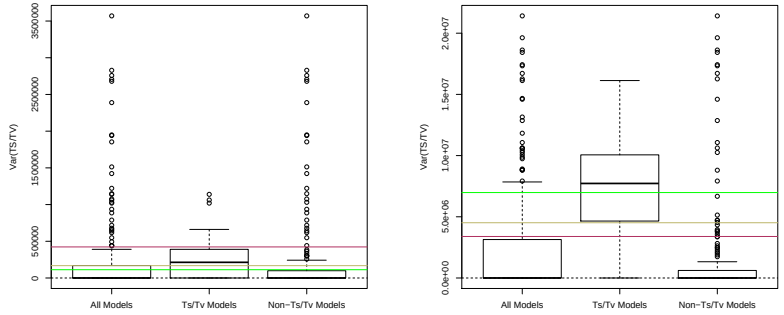
E(To) = 0.619 **0.826** **1.376**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

Figure A.44: Variance Boxplots on HKY85 Data with Sequence Length 500 and E(To) = 1.720 to 4.128



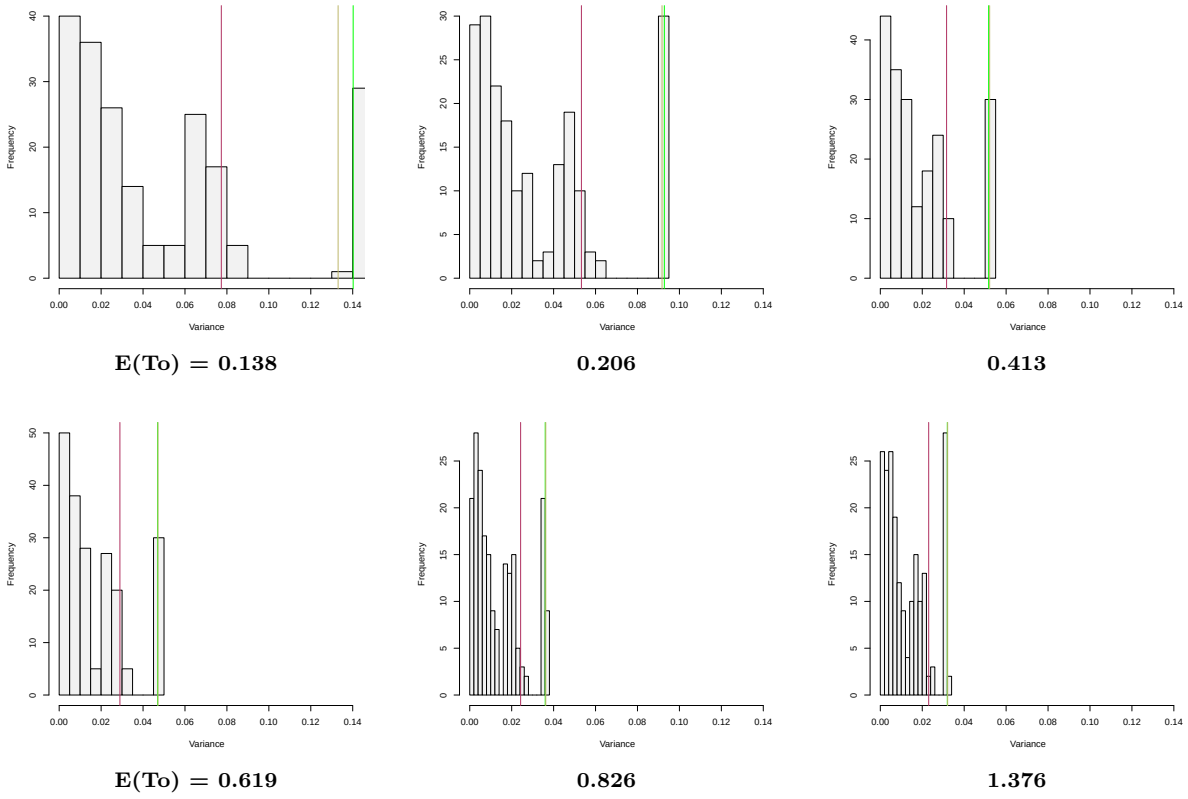
E(To) = 1.720 **2.064**



E(To) = 3.096 **4.128**

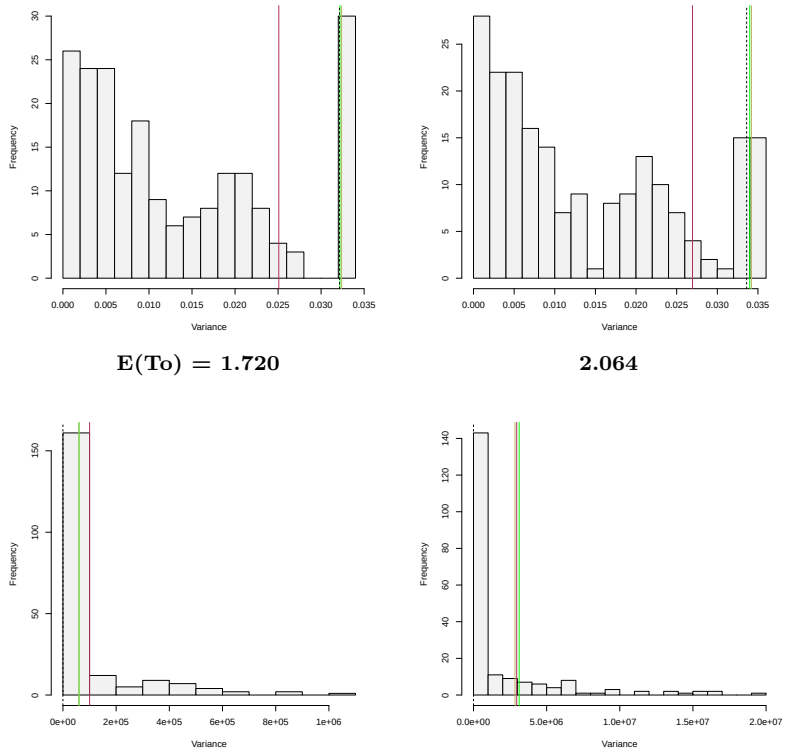
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

Figure A.45: Variance Histograms on HKY85 Data with Sequence Length 1000 with E(To) = 0.138 to 1.376



To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.46: Variance Histograms on HKY85 Data with Sequence Length 1000 for E(To) = 1.72 to 4.128

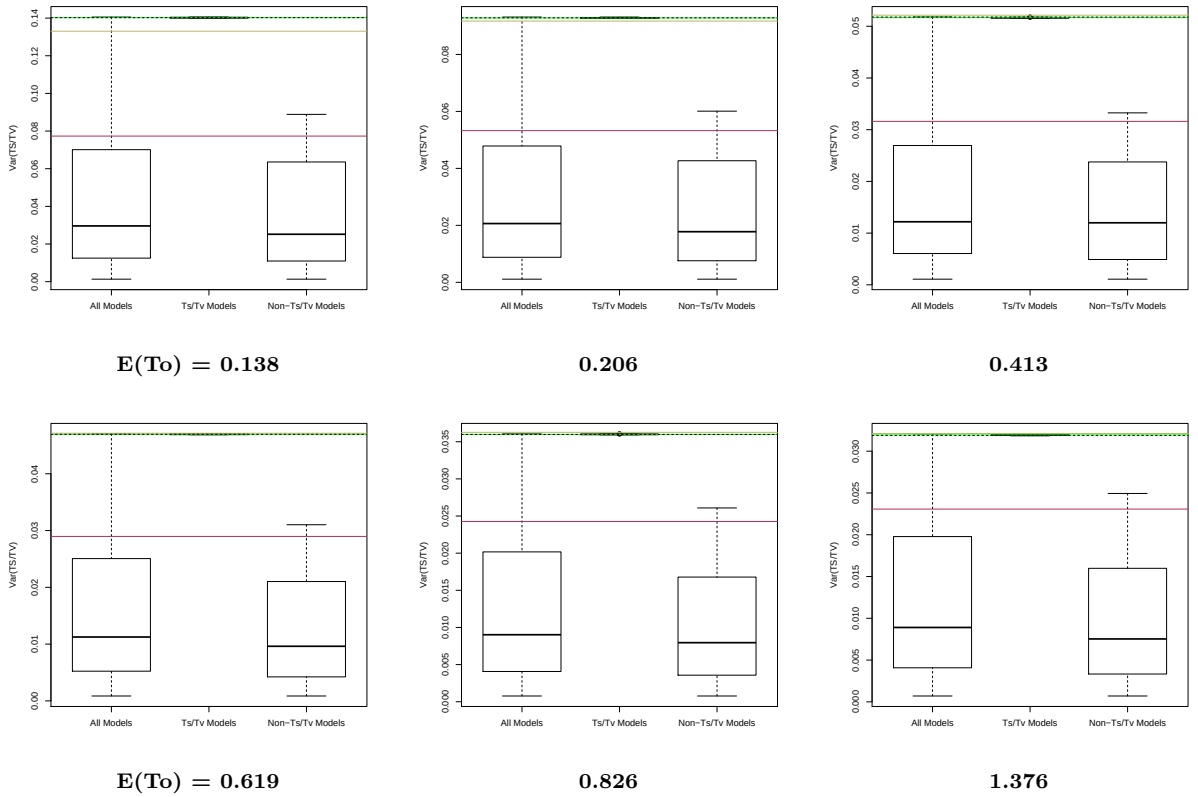


E(To) = 1.720 **2.064**

E(To) = 3.096 **4.128**

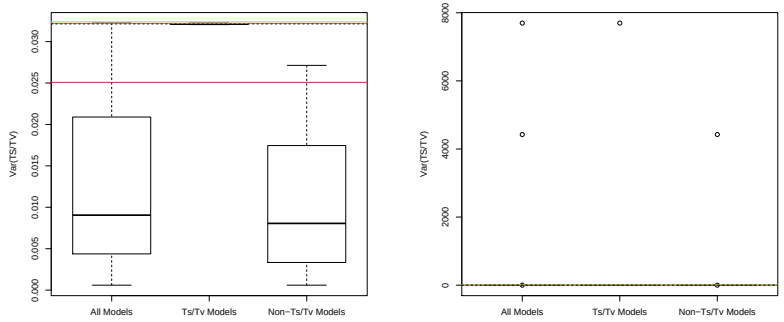
To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.47: Variance Boxplots on HKY85 Data with Sequence Length 1000 with E(To) = 0.138 to 1.376



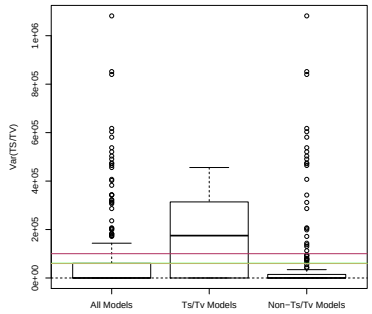
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

Figure A.48: Variance Boxplots on HKY85 Data with Sequence Length 1000 and E(To) = 1.720 to 4.128

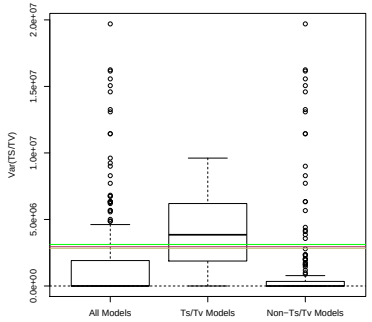


E(To) = 1.720

2.064



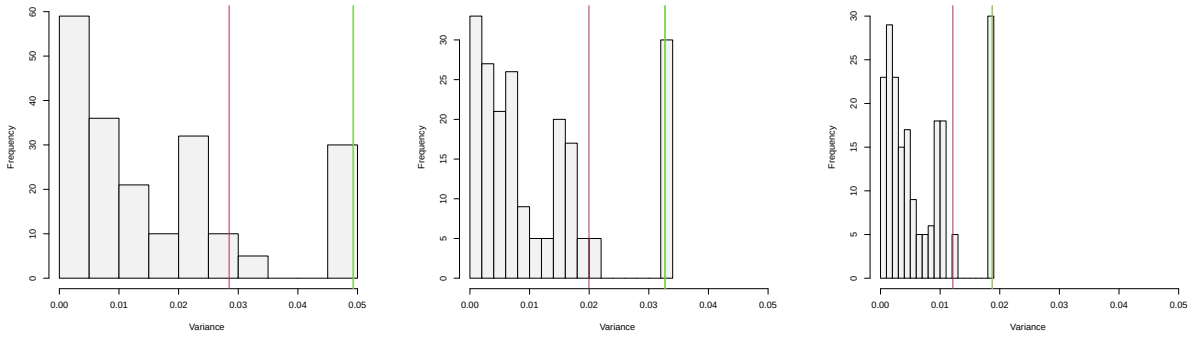
E(To) = 3.096



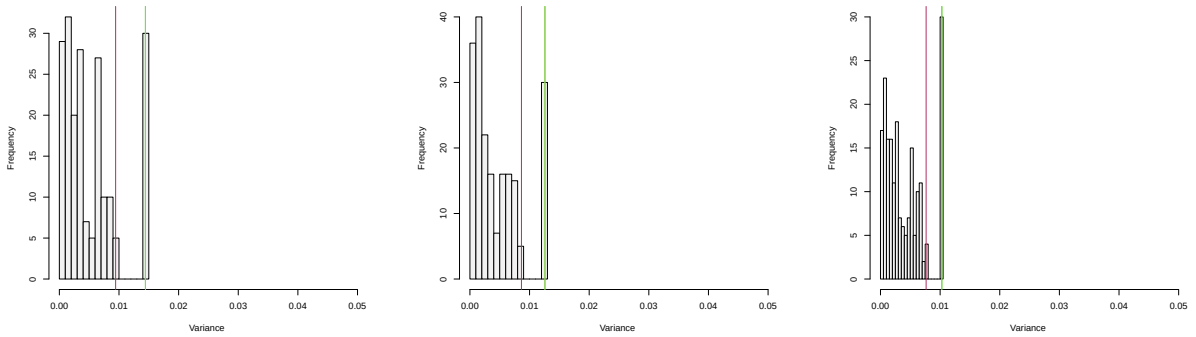
4.128

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

Figure A.49: Variance Histograms on HKY85 Data with Sequence Length 3000



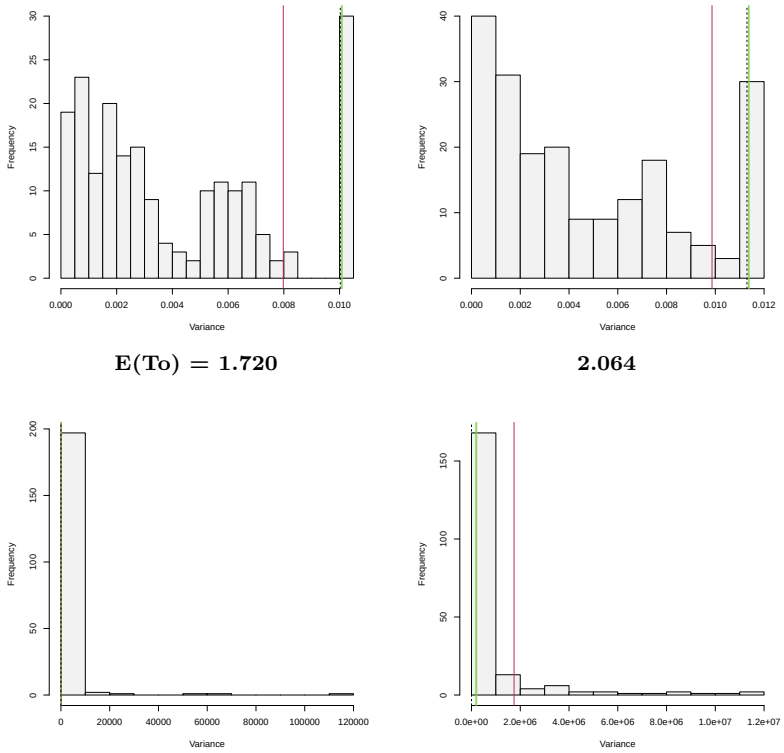
E(To) = 0.138 **0.206** **0.413**



E(To) = 0.619 **0.826** **1.376**

To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.50: Variance Histograms on HKY85 Data with Sequence Length 3000 for E(To) = 1.72 to 4.128

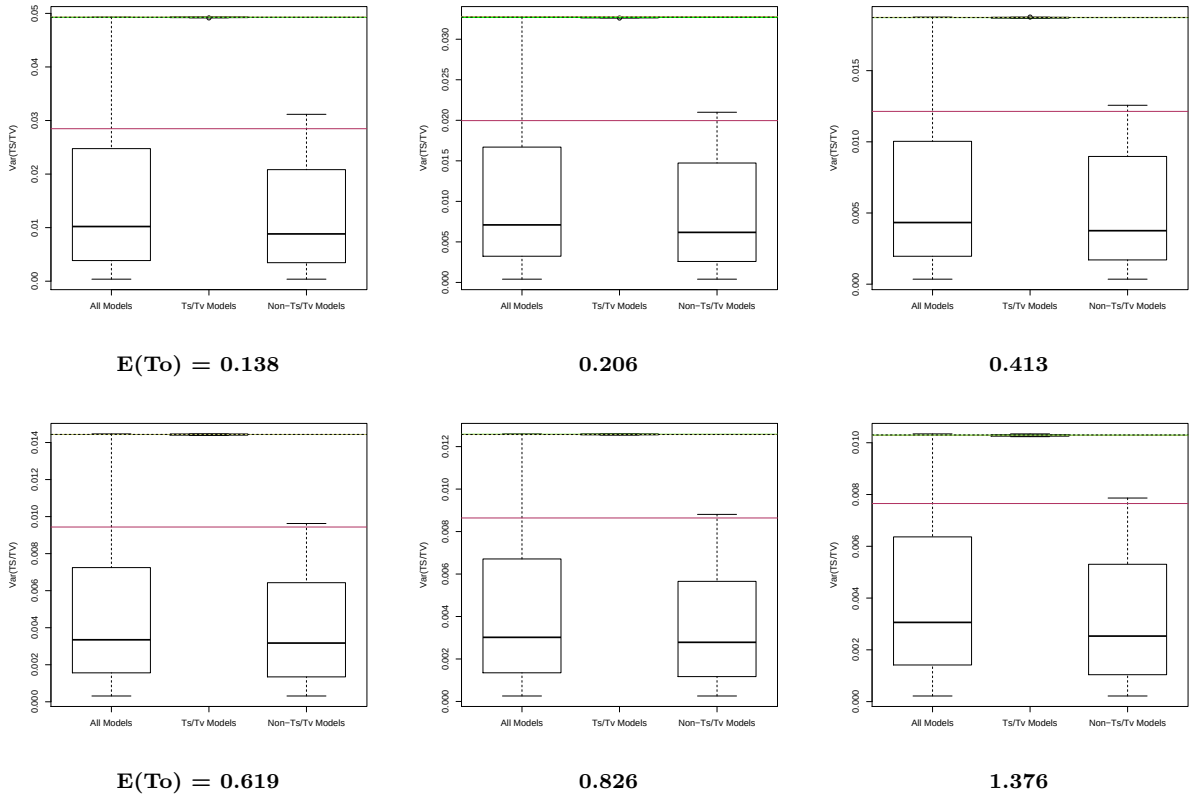


E(To) = 1.720 **2.064**

E(To) = 3.096 **4.128**

To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.51: Variance Boxplots on HKY85 Data with Sequence Length 3000 with E(To) = 0.138 to 1.376

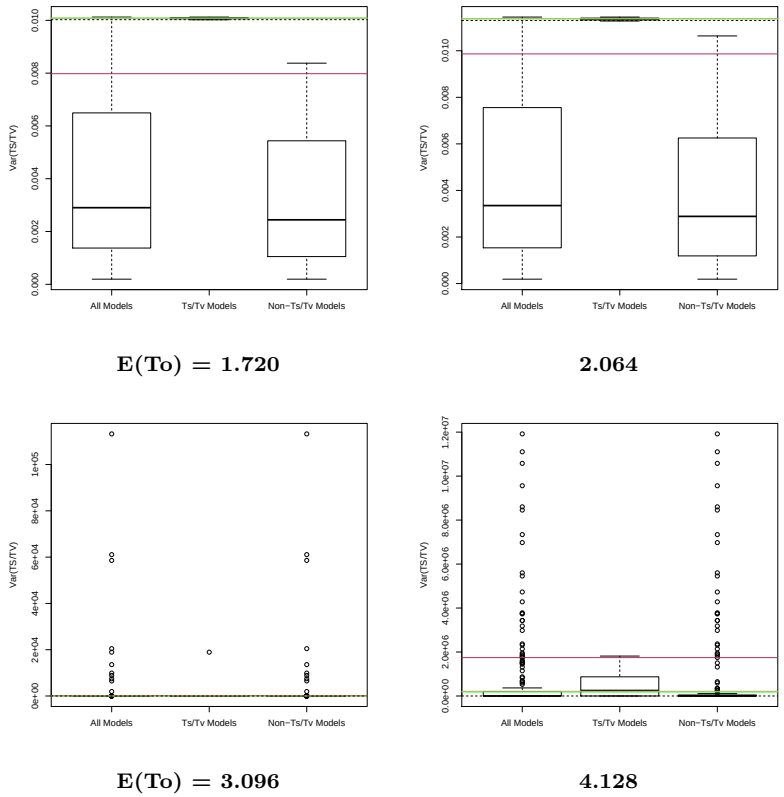


E(To) = 0.138 **0.206** **0.413**

E(To) = 0.619 **0.826** **1.376**

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

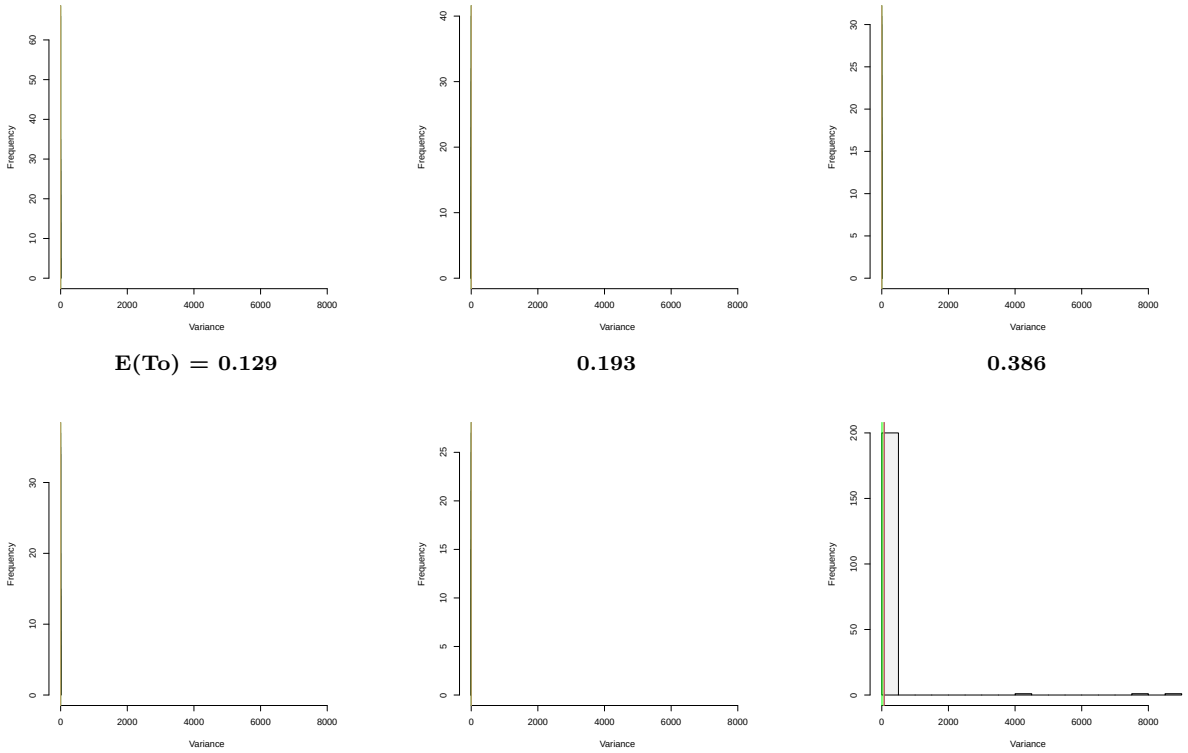
Figure A.52: Variance Boxplots on HKY85 Data with Sequence Length 3000 and E(To) = 1.720 to 4.128



Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

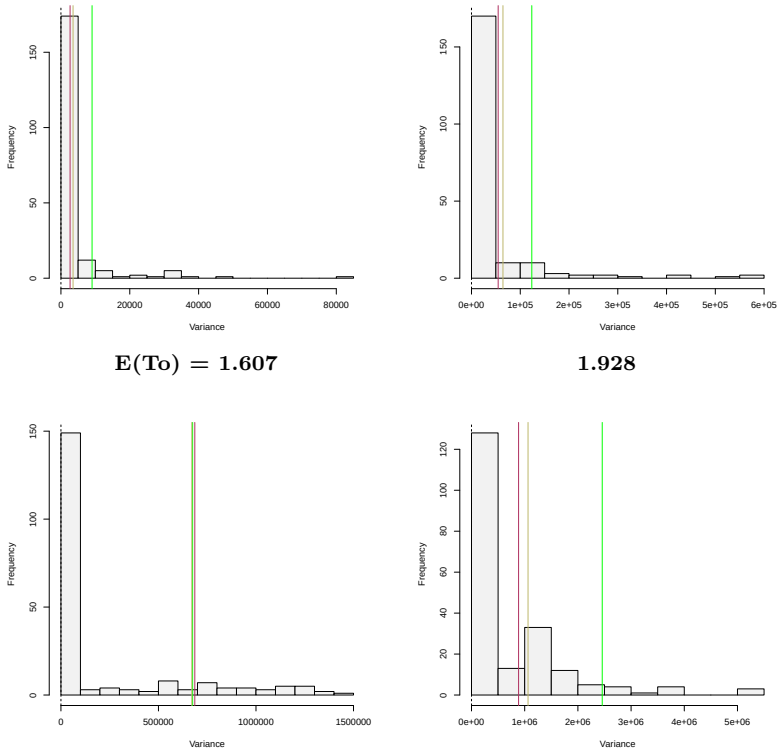
A.2.2 TN93 Variance Analyses

Figure A.53: Variance Histograms on TN93 Data with Sequence Length 250 and E(To) = 0.129 to 1.286



To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.54: Variance Histograms on HKY85 Data with Sequence Length 250 for E(To) = 1.607 to 3.858

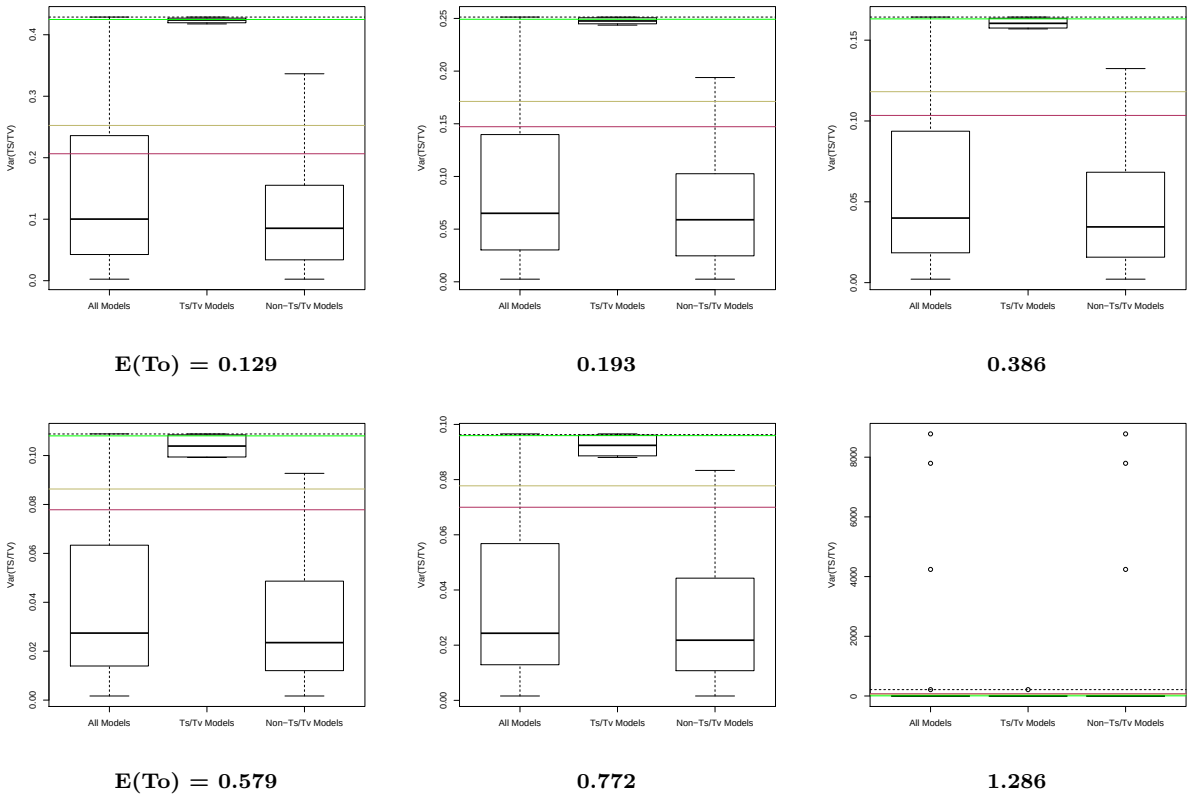


E(To) = 1.607 **1.928**

E(To) = 2.893 **3.858**

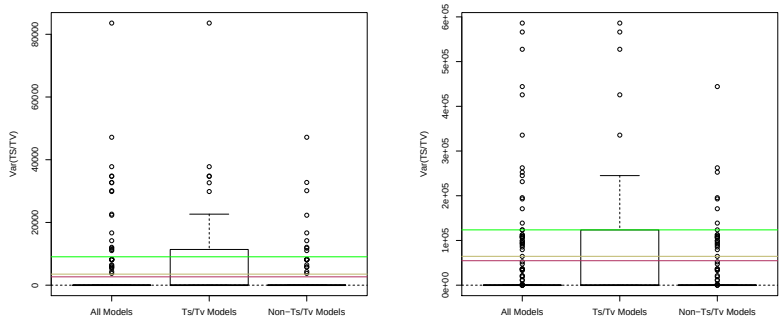
To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average

Figure A.55: Variance Boxplots on TN93 Data with Sequence Length 250 and E(To) = 0.129 to 1.286



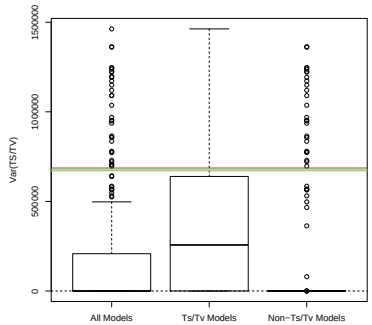
Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

Figure A.56: Variance Boxplots on TN93 Data with Sequence Length 250 and E(To) = 1.607 to 3.858

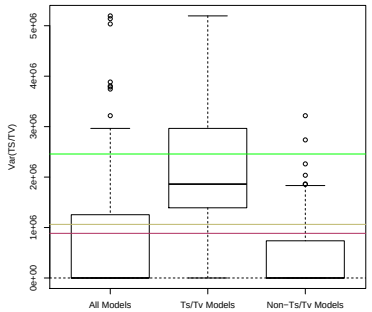


E(To) = 1.607

1.928



E(To) = 2.893



3.858

Ts = transition; Tv = transversion; To = total substitutions (substitutions per site)
Dark Khaki = Average for All Models; Green = Ts/Tv Model Average; Maroon = Non-Ts/Tv Model Average
Within the boxes, the solid black line represents the median.

Appendix B

Codon Simulation Results

B.1 Nonsynonymous/Synonymous Ratios

Figure B.1: dN/dS Estimate Results for a Two Leaf Tree for Clusters of Size 2 to 7

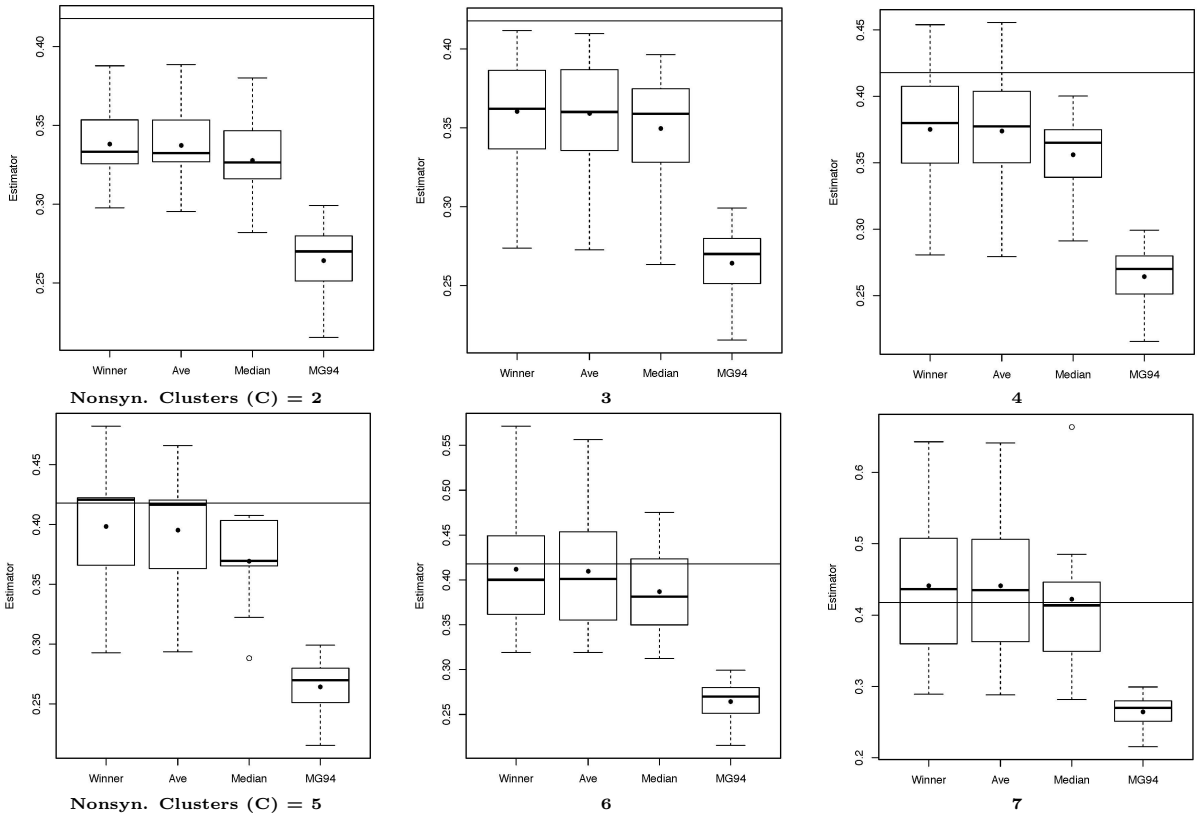


Figure B.2: dN/dS Estimate Results for a Two Leaf Tree for Clusters of Size 8 to 12

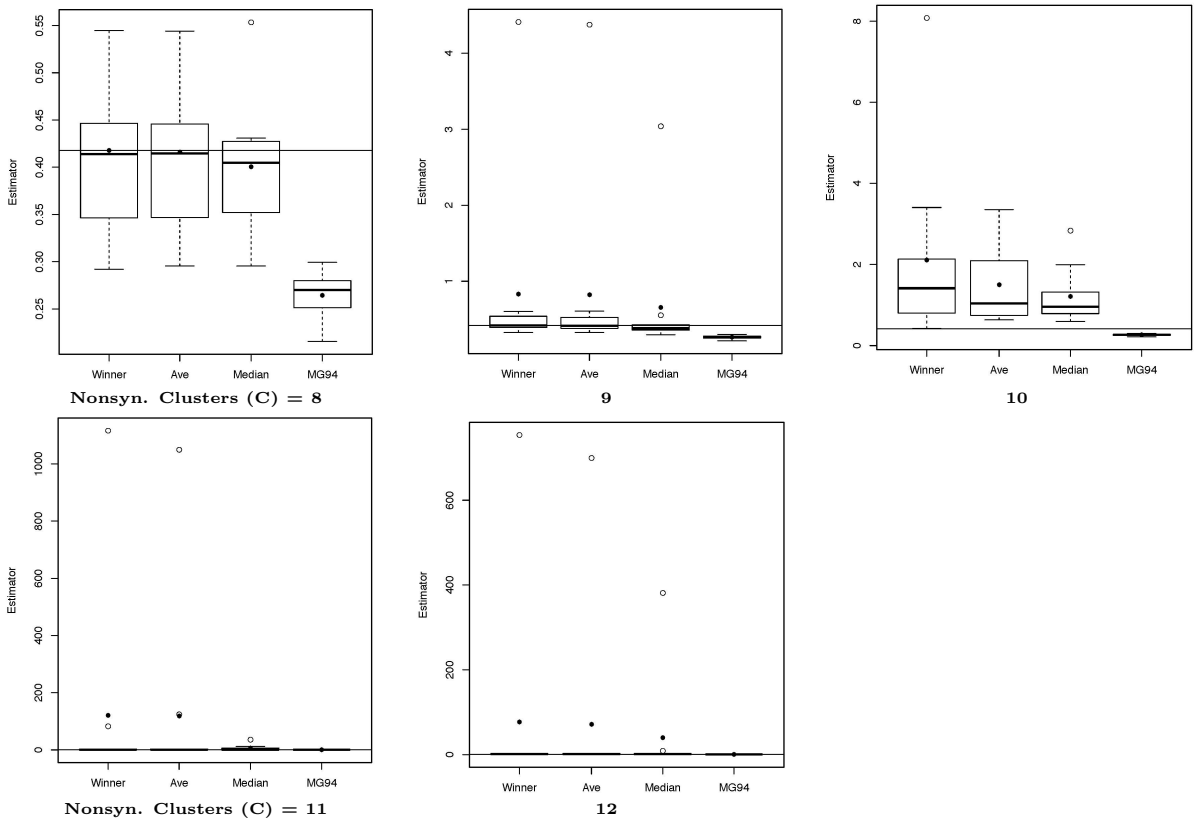


Figure B.3: dN/dS Estimate Results for a Four Leaf Tree for Clusters of Size 2 to 7

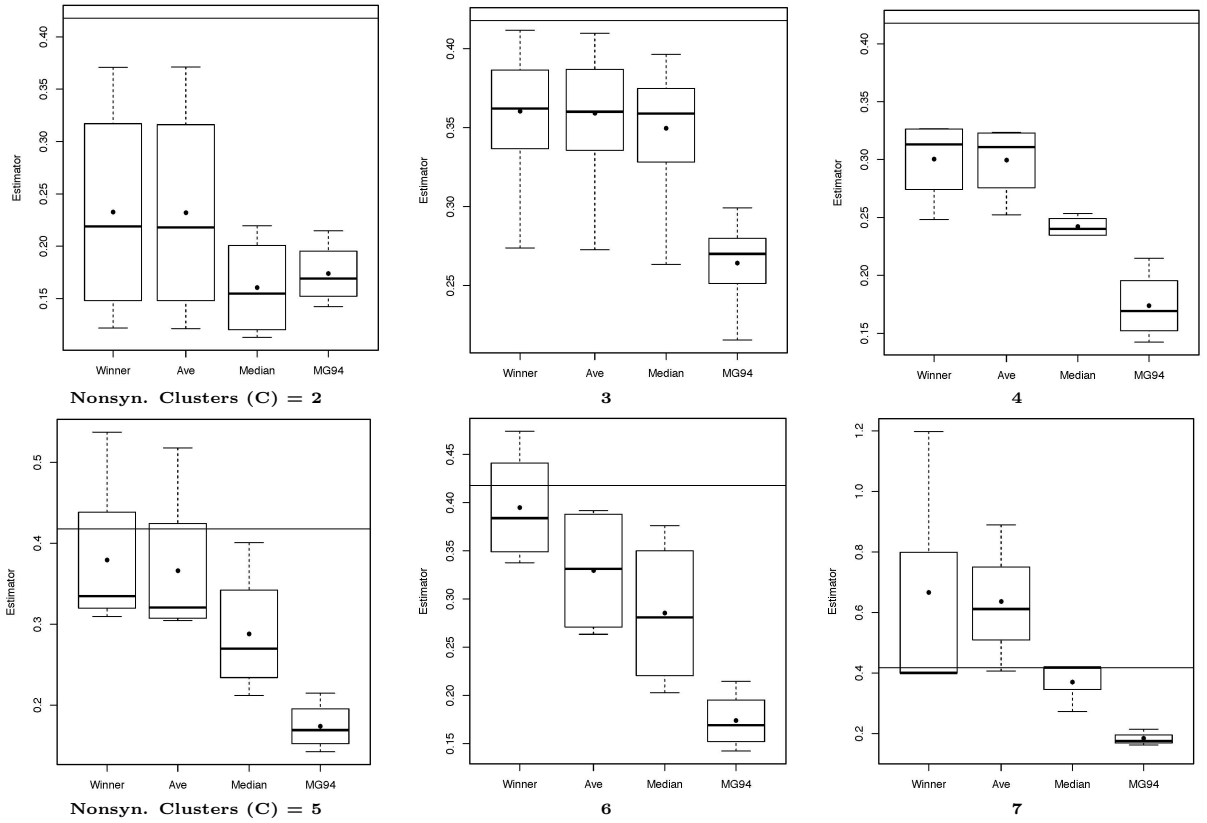
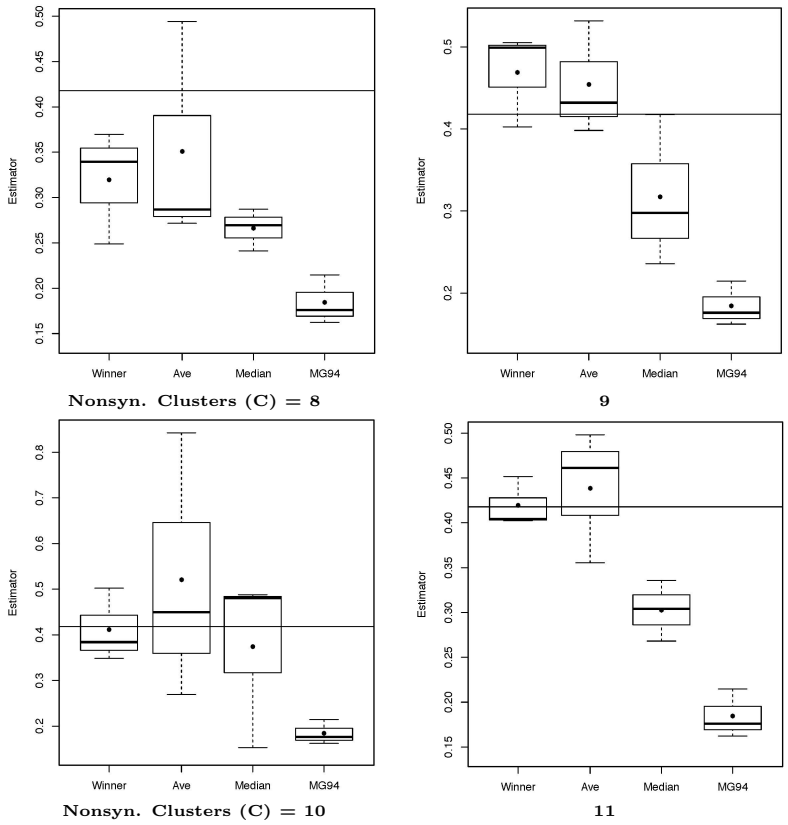
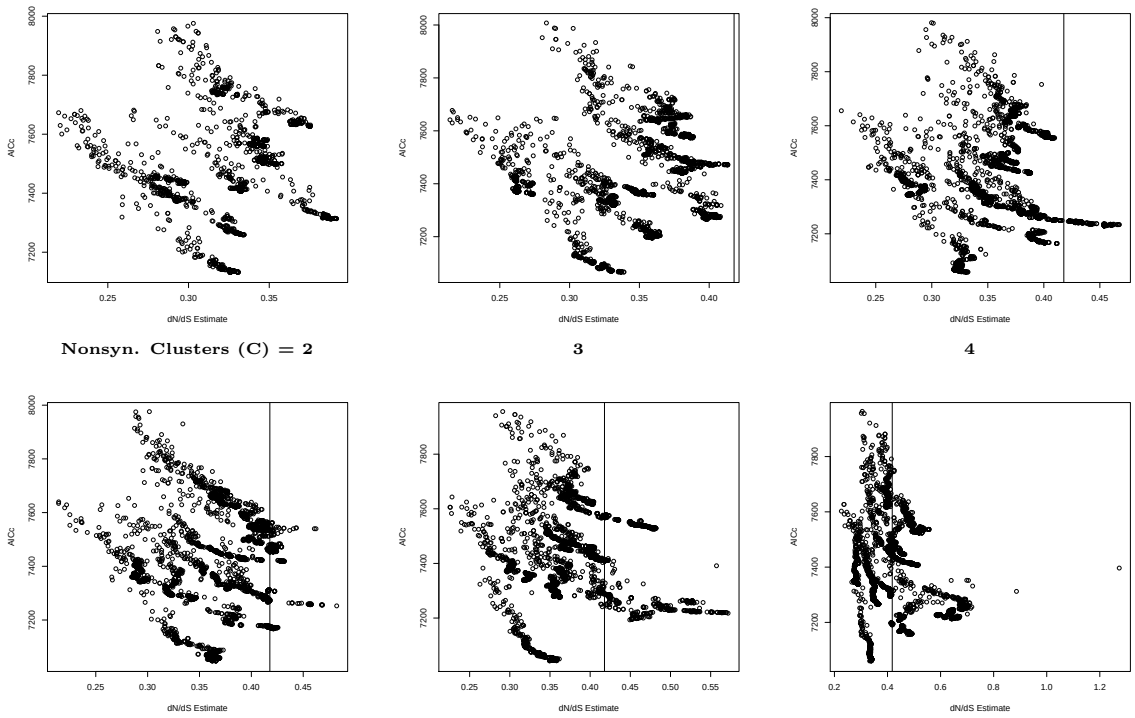


Figure B.4: dN/dS Estimate Results for a Four Leaf Tree for Clusters of Size 8 to 11



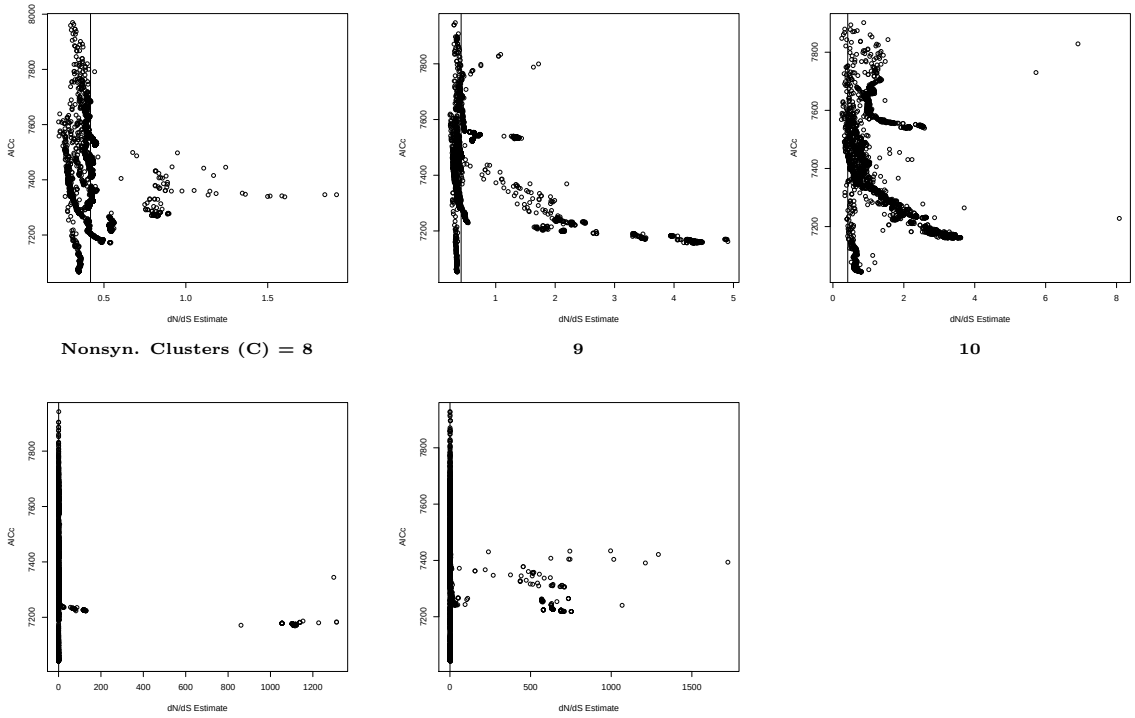
B.2 Information Loss by Nonsynonymous Clusterings

Figure B.5: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree for Clusters of Size 2 to 7



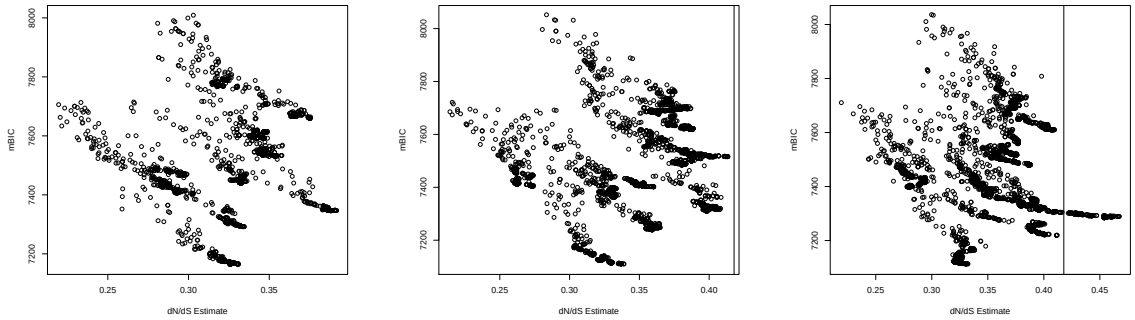
AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure B.6: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree for Clusters of Size 8 to 12

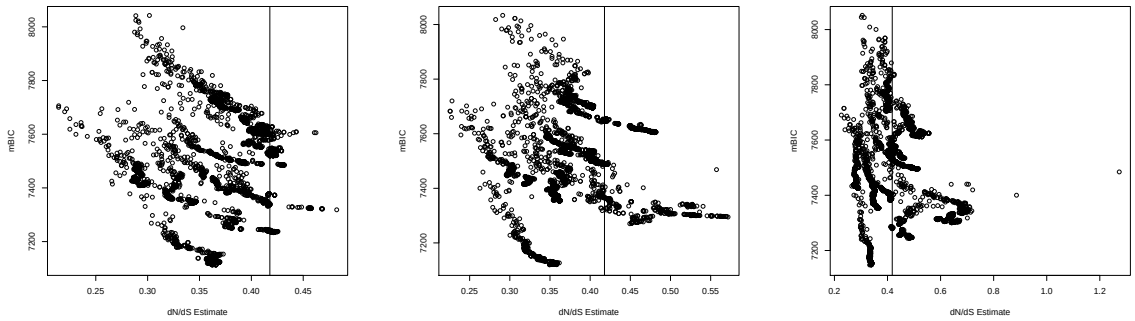


AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure B.7: Population Best-Fit mBIC by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree



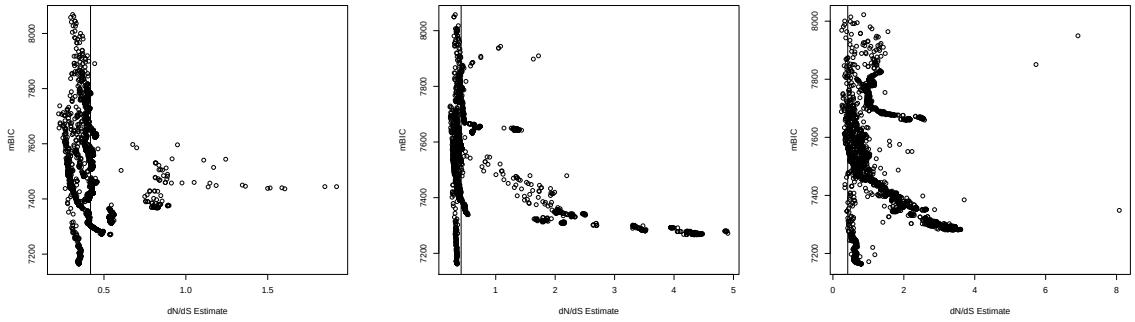
Nonsyn. Clusters (C) = 2 **3** **4**



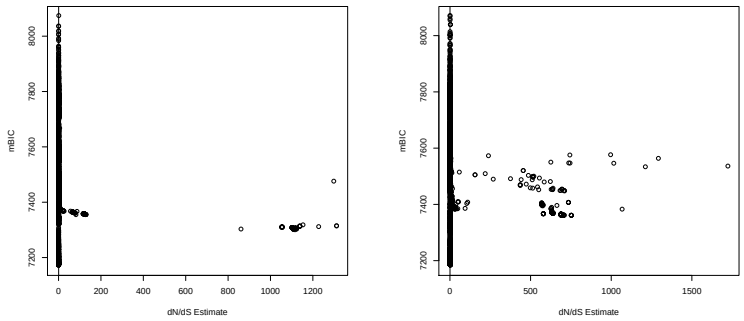
Nonsyn. Clusters (C) = 5 **6** **7**

mBIC = modified Bayesian Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure B.8: Population Best-Fit mBIC by dN/dS Estimates and Non-Synonymous Clusters - Two Leaf Tree



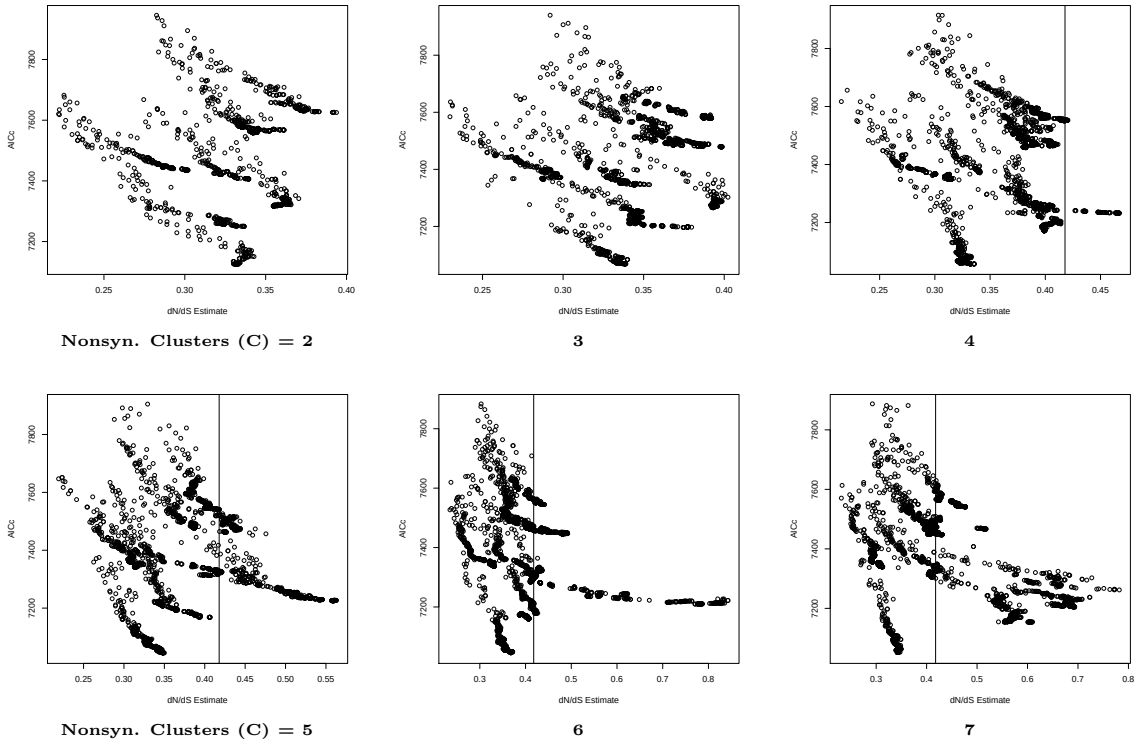
Nonsyn. Clusters (C) = 8 **9** **10**



Nonsyn. Clusters (C) = 11 **12**

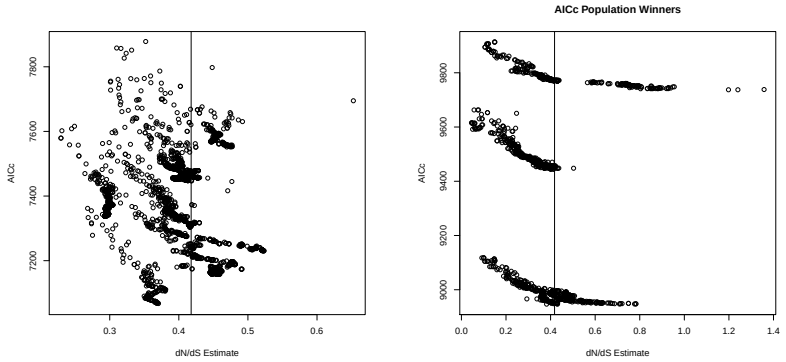
mBIC = modified Bayesian Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure B.9: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree



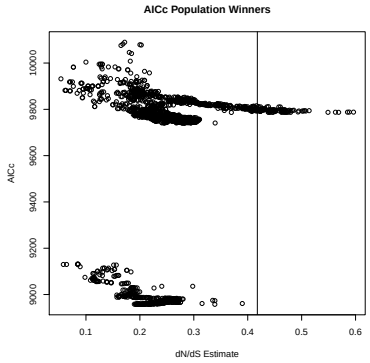
AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure B.10: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree

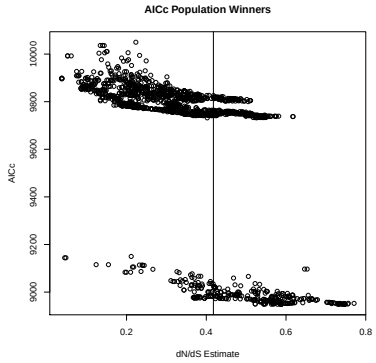


Nonsyn. Clusters (C) = 8

9



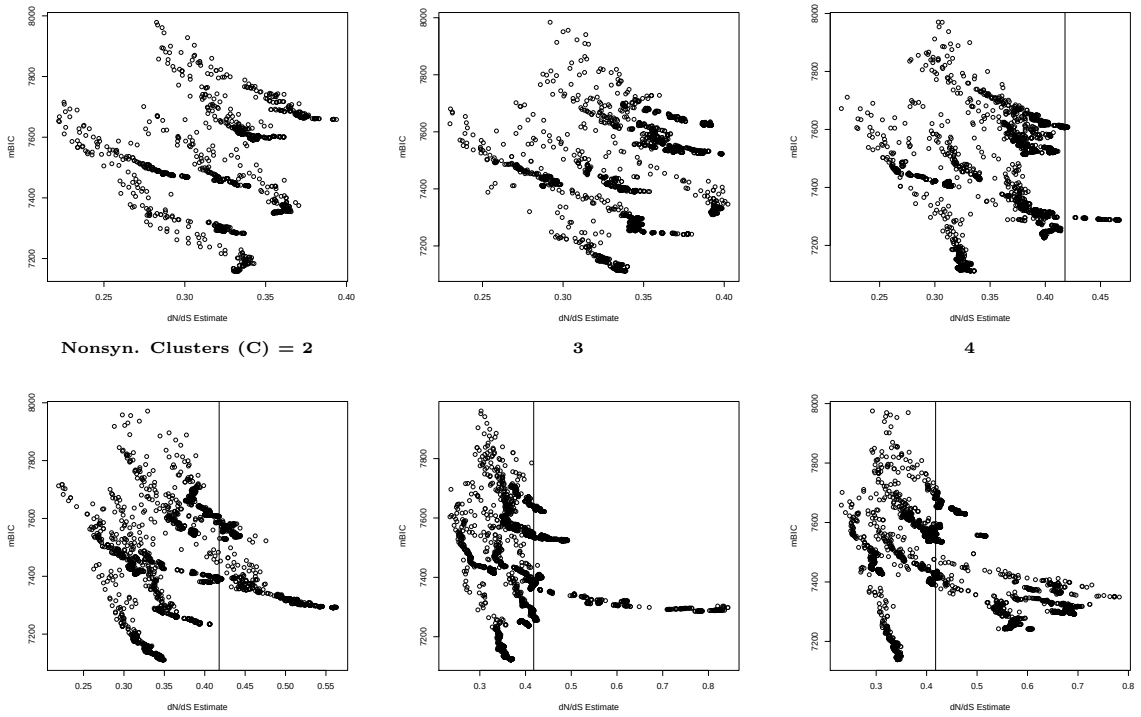
Nonsyn. Clusters (C) = 10



11

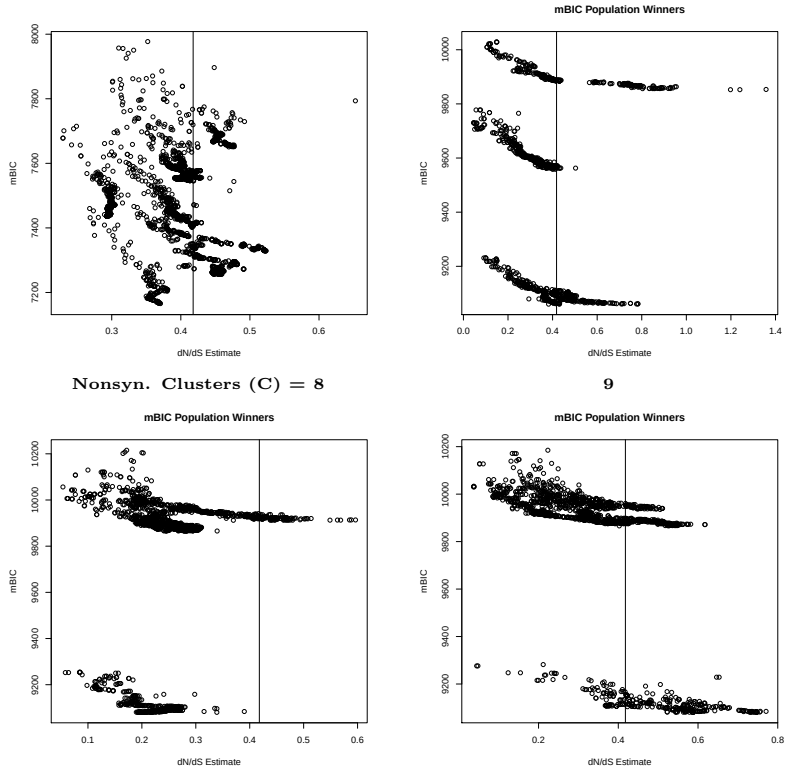
AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure B.11: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree



AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Figure B.12: Population Best-Fit $AIC_{corr.}$ by dN/dS Estimates and Non-Synonymous Clusters - Four Leaf Tree



AICc = corrected Akaike Information Criteria; dN = nonsynonymous; dS = synonymous
Note: The solid black line represents the true value of dN/dS in the simulated data.

Appendix C

HyPhy

For this report, likelihood and model parameter estimates are performed using the HyPhy system. This includes stationary frequencies and the substitution parameters for both nucleotide and codon models.

HyPhy substitution rates are parameterized relative to the first ordered parameter in the model, such as $(\widehat{t\theta}_0, \frac{\hat{\theta}_1}{\theta_0}, \frac{\hat{\theta}_2}{\theta_0}, \frac{\hat{\theta}_3}{\theta_0}, \frac{\hat{\theta}_4}{\theta_0}, \frac{\hat{\theta}_5}{\theta_0})$ where $\widehat{t\theta}_0 = \sum_{b=1}^{B} \hat{\nu}_b$ and B is the number of branches in the unrooted version of the phylogenetic tree. Given the GTR model, the transition-to-transversion ratio (κ) and branch lengths are parameterized and estimated in HyPhy as follows:

$$\begin{aligned}
\hat{E}(\text{Ts}) =& 2\widehat{t\theta}_0 * \{\frac{\hat{\theta}_1}{\theta_0}(\hat{\pi}_A\hat{\pi}_G) + \frac{\hat{\theta}_4}{\theta_0}(\hat{\pi}_C\hat{\pi}_T)\} \\
\hat{E}(\text{Tv}) =& 2\widehat{t\theta}_0 * \{(\hat{\pi}_A\hat{\pi}_C) + \frac{\hat{\theta}_2}{\theta_0}(\hat{\pi}_A\hat{\pi}_T) + \frac{\hat{\theta}_3}{\theta_0}(\hat{\pi}_C\hat{\pi}_G) + \frac{\hat{\theta}_5}{\theta_0}(\hat{\pi}_G\hat{\pi}_T)\} \\
\text{where } \widehat{t\theta_0} =& (\sum_{i=b}^{5} \nu_b) * \{\hat{\pi}_A * (\hat{\theta}_0\hat{\pi}_C + \hat{\theta}_1\hat{\pi}_G + \hat{\theta}_2\hat{\pi}_T) + \hat{\pi}_C * (\hat{\theta}_0\hat{\pi}_A + \hat{\theta}_3\hat{\pi}_G + \hat{\theta}_4\hat{\pi}_T) + \\
& \hat{\pi}_G * (\hat{\theta}_1\hat{\pi}_A + \hat{\theta}_3\hat{\pi}_C + \hat{\theta}_5\hat{\pi}_T) + \hat{\pi}_T * (\hat{\theta}_2\hat{\pi}_A + \hat{\theta}_4\hat{\pi}_C + \hat{\theta}_5\hat{\pi}_G)\} \qquad \text{(C.1)}
\end{aligned}$$

For the nucleotide case, substitution parameter estimates from HyPhy are encoded into vectors of length 6 with the first parameter scaled on the branch lengths (per C.1). However, for our analyses the codon substitution matrix was decoded into substitution parameter vectors of length 263 with the first 67 positions indicating the upper diagonal synonymous changes and the remaining for the nonsynonymous elements. The order of the vector elements represent the positions within the substitution matrix that begin at the top left and move from left to right and top to bottom.

Similar to the nucleotide case, the parameter estimates for codon models were calculated as

the ratio of rates with respect to the first ordered substitution parameter in the coding vector. Estimates were then calculated with respect to (1.17), (1.18), (1.19), and (1.19) where $t\alpha$ and $t\beta$ are estimated as

$$\widehat{t\alpha} = \widehat{t\theta}_0 \times \sum_{i=1}^{66} \frac{\hat{\theta}_i}{\widehat{t\theta}_0} \hat{\pi}_{y_i}$$
$$\widehat{t\beta} = \widehat{t\theta}_0 \times \sum_{i=67}^{262} \frac{\hat{\theta}_i}{\widehat{t\theta}_0} \hat{\pi}_{y_i}$$

with substitution estimates parameterized as $\widehat{t\theta}_0, \frac{\hat{\theta}_1}{\widehat{t\theta}_0}, \frac{\hat{\theta}_2}{\widehat{t\theta}_0}, ..., \frac{\hat{\theta}_{263}}{\widehat{t\theta}_0}$.

Estimation of stationary frequencies is straightforward. HyPhy does this by simply calculating the incidence of each of the four states across the alignment and averaging on the length of the alignment. This is done using the HarvestFrequencies function.

Substitution model parameters are calculated as the *arg max* on the iterated maximized likelihood via Felsenstein [1981]. For computational purposes, nucleotide substitution parameters are encoded using 1×6 vectors that describe the clustering of the upper-diagonal elements of the substitution matrix (i.e. θ_{0-5}).

Each element of the vector codes a clustering for the reversible nucleotide substitutions. For example, the JC69 and F81 models are coded as $(0, 0, 0, 0, 0, 0)$ since all substitution parameters are assumed equal in these models. K80 and HKY85 are coded as $(0, 1, 0, 0, 1, 0)$ since the $A \leftrightarrow G$ and $C \leftrightarrow T$ transition substitutions are assumed equal and the other substitutions (transversions) are assumed equal. The other named models, TN93 and GTR are coded as $(0, 1, 0, 0, 2, 0)$, and $(0, 1, 2, 3, 4, 5)$, respectively. Coded vectors are then decoded to 4×4 substitution model matrices and entered into the LikelihoodFunction and Optimize functions of the HyPhy system's iterative, maximized likelihood process.

Given the TN93 model that is coded by $(0, 1, 0, 0, 2, 0)$, the decoded substitution matrix for analysis by HyPhy corresponds to $0 = \theta_0$, $1 = \theta_1$, and $2 = \theta_2$ and implies that $(0, 1, 0, 0, 2, 0) = (\theta_0, \theta_1, \theta_0, \theta_0, \theta_2, \theta_0)$. This corresponds to $A \leftrightarrow C = \theta_0$, $A \leftrightarrow G = \theta_0$, $A \leftrightarrow T = \theta_0$, $C \leftrightarrow G = \theta_0$, $C \leftrightarrow T = \theta_0$, and $G \leftrightarrow T = \theta_0$ such that

$$R^{\text{TN93}} = \begin{pmatrix} * & \theta_0\pi_C & \theta_1\pi_G & \theta_0\pi_T \\ \theta_0\pi_A & * & \theta_0\pi_G & \theta_2\pi_T \\ \theta_1\pi_A & \theta_0\pi_C & * & \theta_0\pi_T \\ \theta_0\pi_A & \theta_2\pi_C & \theta_0\pi_G & * \end{pmatrix}$$

In order to uniquely specify each possible substitution model with vector codings, Kosakovsky Pond et al. [2007] cite two rules used to rule out cases of multiple vectors coding the same sub-

stitution models such as $[0,0,0,0,0] = [1,1,1,1,1]$ or $[0,1,0,0,1,0] = [1,3,1,1,3,1]$.

1. Parameters corresponding to the number of rate clusterings for a given model ($p = 1,2,3,...N$) must each be identified in the coding vector at least one time. For example, if $N = 3$, the values of $\theta_{p-1} = \{\theta_0, \theta_1, \theta_2\}$ must each be represented at least one time in the coding vector. For example, if this condition does not hold, consider $N = 3$ such that a coding vector is $(0,1,1,0,3,3)$, however, the same model can also be coded with $(0,1,1,0,2,2)$. However, under rule no. 1, vector $(0,1,1,0,3,3)$ is not permissible.

2. The first occurrence of each possible clustering must be represented in sequence. For example, if $N = 3$, then the rate parameter, θ_0, will precede the first occurrence of the following parameter, θ_1, and the first occurrence of θ_2 will not proceed the first occurrence of θ_1. For instance, (0,0,1,2,1) follows this rule, but (0,2,1,0,1) does not since the same model can be coded with (0,1,2,0,2).‘‘‘

Appendix D

Nucleotide Modeling Programs

D.1 SimoData.R

```
#program to simulate data
#Current Version generates according to the GTR model

#primary function generating and outputing the data
simo0<-function(theta1,theta2,theta3,theta4,theta5,theta6,p1,p2,p3,p4,
sequence_length,p1a,p2b,p4d,p4dz)
{
source(paste("./utilities/translate.R",sep=""))
source(paste("./utilities/generatetaxa.R",sep=""))

library(Matrix)
pi<-c(p1,p2,p3,1-p1-p2-p3)

z1<--theta1*pi[2]-theta2*pi[3]-theta3*pi[4]
z2<--theta1*pi[1]-theta4*pi[3]-theta5*pi[4]
z3<--theta2*pi[1]-theta4*pi[2]-theta6*pi[4]
z4<--theta3*pi[1]-theta5*pi[2]-theta6*pi[3]

x<-runif(sequence_length)
r<-rep(0,length(x))
```

```
for (i in 1:length(x))
{
if(x[i]<pi[1]){r[i]<-1} else
if(x[i]<pi[1]+pi[2]){r[i]<-2} else
if(x[i]<pi[1]+pi[2]+pi[3]){r[i]<-3} else
{r[i]<-4}
}

R<-matrix(c(z1, theta1*pi[2], theta2*pi[3], theta3*pi[4],
theta1*pi[1], z2 , theta4*pi[3], theta5*pi[4],
theta2*pi[1], theta4*pi[2], z3, theta6*pi[4],
theta3*pi[1], theta5*pi[2], theta6*pi[3], z4),nrow=4,byrow=TRUE)

Pa<-expm(R*p1a) #brA
Pb<-expm(R*p2b) #brB
Pd<-expm(R*p4d) #brAa
Pdz<-expm(R*p4dz) #brAb

d<-generate_taxa(indata=r,Ptemp=Pd) #brAa
z1<-generate_taxa(indata=r,Ptemp=Pdz)
c<-generate_taxa(indata=z1,Ptemp=Pb) #brB
z2<-generate_taxa(indata=z1,Ptemp=Pa)
b<-generate_taxa(indata=z2,Ptemp=Pb) #brB
a<-generate_taxa(indata=z2,Ptemp=Pa) #brA

a.char<-translate(a)
b.char<-translate(b)
c.char<-translate(c)
d.char<-translate(d)

cat("#Wiget1",file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n")
cat(a.char,file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",
append=TRUE)
cat("#Wiget2",file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",
append=TRUE)
cat(b.char,file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",
```

```
append=TRUE)
cat("#Wiget3",file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",
append=TRUE)
cat(c.char,file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",
append=TRUE)
cat("#Wiget4",file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",
append=TRUE)
cat(d.char,file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",
append=TRUE)

#print the tree on the file for use with the HyPhy GUI
cat("\n","((Wiget1,Wiget2),Wiget3,Wiget4);",file=paste("./HyPhy_Data/simoz",
bigreps,".txt",sep=""),"\n",append=TRUE)
# cat("\n","(Wiget1,Wiget2);",file=paste("./HyPhy_Data/simoz",bigreps,
".txt",sep=""),"\n",append=TRUE)
}
```

D.1.1 Utility Function: translate.R

```
#utility function tranlating the numeric vector values into character output
translate<-function(input)
{
translation<-rep('z',length(input))
for(k in 1:length(input))
{
if(input[k]==1)
{translation[k]<-'a'}
if(input[k]==2)
{translation[k]<-'c'}
if(input[k]==3)
{translation[k]<-'g'}
if(input[k]==4)
{translation[k]<-'t'}
}
return(translation)
}
```

D.1.2 Utility Function: generatetaxa.R

```
generate_taxa<-function(indata,Ptemp)
{
Pt<-Ptemp
temp<-runif(length(indata))
outdata<-rep(0,length(indata))
for(j in 1:length(indata))
{
if (indata[j]==1){
if(temp[j]<=Pt[1,1]){outdata[j]<-indata[j]} else
if(temp[j]<=Pt[1,1]+Pt[1,2]){outdata[j]<-2} else
if (temp[j]<=Pt[1,1]+Pt[1,2]+Pt[1,3]){outdata[j]<-3} else
{outdata[j]<-4}} else
if(indata[j]==2){
if(temp[j]<=Pt[2,2]){outdata[j]<-indata[j]} else
if(temp[j]<=Pt[2,2]+Pt[2,1]){outdata[j]<-1} else
if (temp[j]<=Pt[2,2]+Pt[2,1]+Pt[2,3]){outdata[j]<-3} else
{outdata[j]<-4}} else
if(indata[j]==3){
if(temp[j]<=Pt[3,3]){outdata[j]<-indata[j]} else
if(temp[j]<=Pt[3,3]+Pt[3,1]){outdata[j]<-1} else
if (temp[j]<=Pt[3,3]+Pt[3,1]+Pt[3,2]){outdata[j]<-2} else
{outdata[j]<-4}} else
if(indata[j]==4){
if(temp[j]<=Pt[4,4]){outdata[j]<-indata[j]} else
if(temp[j]<=Pt[4,4]+Pt[4,1]){outdata[j]<-1} else
if (temp[j]<=Pt[4,4]+Pt[4,1]+Pt[4,2]){outdata[j]<-2} else
{outdata[j]<-3}}
}
return(outdata)
}
```

D.2 NucModelSpace.h

```
#include("utilities\pop.h");
#include("utilities\MatrixMapNuc.h");
```

```
#include("utilities\sum.h");

popmaker(1);

function nuc_modelspace(prior)
{
datafile="HyPhy_Data\simoz";
dataformat=".txt";
indatafile=datafile+bigreps+dataformat;
DataSet myData = ReadDataFile (indatafile);

DataSetFilter nucFilter = CreateFilter (myData, 1);
HarvestFrequencies (obsFreqs, nucFilter, 1, 1, 0);

length=1;
rows = Rows(prior);
transition_transversion={rows,19};
model_parameters = {rows,6};

global R1; global R2; global R3; global R4; global R5;

for(zz=0; zz<length; zz=zz+1)
{

for(z=0; z<rows; z=z+1)
{
REPLACE_TREE_STRUCTURE = 1;
row = {1,Columns(prior)};
CMatrix = {4,4};
theta0v={1,5};
/*************************************************************************/
/**Model Likelihoods for individulas in the population **/
for(y=0; y<Columns(prior); y=y+1)
{
row[0][y]=prior[z][y];
}
```

```
R1=0; R2=0; R3=0; R4=0; R5=0; R6=0;
MatrixMap_nuc(row); /*output is CMatrix*/
Model eval = (CMatrix , obsFreqs);
Tree myTree = ((Wiget1,Wiget2),Wiget3,Wiget4);

LikelihoodFunction theLikFun = (nucFilter , myTree,
LIKELIHOOD_FUNCTION_OUTPUT=1);

Optimize(MLEs, theLikFun);

fprintf(stdout,"MLEs=",MLEs,"\n");

AIC = -2*MLEs[1][0] + (2*MLEs[1][2]*myData.sites)/(myData.sites - MLEs[1][2]
- 1);

theta0v={1,5};
for (ct=0; ct<5; ct=ct+1) {
theta0v[0][ct]=MLEs[0][MLEs[1][1]-1-ct];
}

fprintf(stdout,"theta0v=",theta0v,"\n");
theta0=sum(theta0v)/Columns(theta0v);

transition_transversion[z][0]=z+1;
transition_transversion[z][1]=AIC;

/*store the clustering parameterizations for each model*/
for(_hh=0;_hh<Columns(prior);_hh=_hh+1)
{
transition_transversion[z][_hh+7] = row[0][_hh];
}

/**save the model parameter values from the MLEs object**/
for(_i=0; _i<Columns(prior);_i=_i+1)
{
```

```
if(prior[z][_i]==0) {model_parameters[z][_i]=theta0;}
else
{
for(_ii=0; _ii<MLEs[1][2];_ii=_ii+1)
{
if(prior[z][_i] == _ii+1)
{
model_parameters[z][_i] = MLEs[0][_ii]*theta0;
}
}
}
}

/*store the parameter estimates for each model*/
for(_hhh=0;_hhh<Columns(prior);_hhh=_hhh+1)
{
transition_transversion[z][_hhh+13] = model_parameters[z][_hhh];
}

/*fprintf(stdout,"Model_Parameters",z,": ",model_parameters,"\n");*/

/**store the numerator and denominator of the transition/transversion
ratio**/

/*the transition piece*/
transition_transversion[z][3]= 2*(
  (model_parameters[z][1]*(obsFreqs[0][0]*obsFreqs[2][0]))
  +(model_parameters[z][4]*(obsFreqs[1][0]*obsFreqs[3][0]))
  );

/*the transversion piece*/
transition_transversion[z][4]= 2*(
  (model_parameters[z][0]*(obsFreqs[0][0]*obsFreqs[1][0]))
  +(model_parameters[z][2]*(obsFreqs[0][0]*obsFreqs[3][0]))
  +(model_parameters[z][3]*(obsFreqs[1][0]*obsFreqs[2][0]))
  +(model_parameters[z][5]*(obsFreqs[2][0]*obsFreqs[3][0]))
```

```
  );

transition_transversion[z][2]=2*transition_transversion[z][3]/
transition_transversion[z][4];
}
extension=".txt";
file="data\_TsTv";
outfile=file+bigreps+extension;
fprintf(outfile,transition_transversion,"\n");
}

return rows;
}
```

D.2.1 Utility Function: pop.h

```
function popmaker(test)
{
pop =
{{1, 1, 1, 1, 1, 1}
{1, 1, 1, 1, 1, 2}
{1, 1, 1, 1, 2, 1}
{1, 1, 1, 1, 2, 2}
{1, 1, 1, 2, 1, 1}
{1, 1, 1, 2, 1, 2}
{1, 1, 1, 2, 2, 1}
{1, 1, 1, 2, 2, 2}
{1, 1, 2, 1, 1, 1}
{1, 1, 2, 1, 1, 2}
{1, 1, 2, 1, 2, 1}
{1, 1, 2, 1, 2, 2}
{1, 1, 2, 2, 1, 1}
{1, 1, 2, 2, 1, 2}
{1, 1, 2, 2, 2, 1}
{1, 1, 2, 2, 2, 2}
{1, 2, 1, 1, 1, 1}
{1, 2, 1, 1, 1, 2}
```

```
{1, 2, 1, 1, 2, 1}
{1, 2, 1, 1, 2, 2}
{1, 2, 1, 2, 1, 1}
{1, 2, 1, 2, 1, 2}
{1, 2, 1, 2, 2, 1}
{1, 2, 1, 2, 2, 2}
{1, 2, 2, 1, 1, 1}
{1, 2, 2, 1, 1, 2}
{1, 2, 2, 1, 2, 1}
{1, 2, 2, 1, 2, 2}
{1, 2, 2, 2, 1, 1}
{1, 2, 2, 2, 1, 2}
{1, 2, 2, 2, 2, 1}
{1, 2, 2, 2, 2, 2}
{1, 1, 1, 1, 2, 3}
{1, 1, 1, 2, 1, 3}
{1, 1, 1, 2, 2, 3}
{1, 1, 1, 2, 3, 1}
{1, 1, 1, 2, 3, 2}
{1, 1, 1, 2, 3, 3}
{1, 1, 2, 1, 1, 3}
{1, 1, 2, 1, 2, 3}
{1, 1, 2, 1, 3, 1}
{1, 1, 2, 1, 3, 2}
{1, 1, 2, 1, 3, 3}
{1, 1, 2, 2, 1, 3}
{1, 1, 2, 2, 2, 3}
{1, 1, 2, 2, 3, 1}
{1, 1, 2, 2, 3, 2}
{1, 1, 2, 2, 3, 3}
{1, 1, 2, 3, 1, 1}
{1, 1, 2, 3, 1, 2}
{1, 1, 2, 3, 1, 3}
{1, 1, 2, 3, 2, 1}
{1, 1, 2, 3, 2, 2}
{1, 1, 2, 3, 2, 3}
```

```
{1, 1, 2, 3, 3, 1}
{1, 1, 2, 3, 3, 2}
{1, 1, 2, 3, 3, 3}
{1, 2, 1, 1, 1, 3}
{1, 2, 1, 1, 2, 3}
{1, 2, 1, 1, 3, 1}
{1, 2, 1, 1, 3, 2}
{1, 2, 1, 1, 3, 3}
{1, 2, 1, 2, 1, 3}
{1, 2, 1, 2, 2, 3}
{1, 2, 1, 2, 3, 1}
{1, 2, 1, 2, 3, 2}
{1, 2, 1, 2, 3, 3}
{1, 2, 1, 3, 1, 1}
{1, 2, 1, 3, 1, 2}
{1, 2, 1, 3, 1, 3}
{1, 2, 1, 3, 2, 1}
{1, 2, 1, 3, 2, 2}
{1, 2, 1, 3, 2, 3}
{1, 2, 1, 3, 3, 1}
{1, 2, 1, 3, 3, 2}
{1, 2, 1, 3, 3, 3}
{1, 2, 2, 1, 1, 3}
{1, 2, 2, 1, 2, 3}
{1, 2, 2, 1, 3, 1}
{1, 2, 2, 1, 3, 2}
{1, 2, 2, 1, 3, 3}
{1, 2, 2, 2, 1, 3}
{1, 2, 2, 2, 2, 3}
{1, 2, 2, 2, 3, 1}
{1, 2, 2, 2, 3, 2}
{1, 2, 2, 2, 3, 3}
{1, 2, 2, 3, 1, 1}
{1, 2, 2, 3, 1, 2}
{1, 2, 2, 3, 1, 3}
{1, 2, 2, 3, 2, 1}
```

```
{1, 2, 2, 3, 2, 2}
{1, 2, 2, 3, 2, 3}
{1, 2, 2, 3, 3, 1}
{1, 2, 2, 3, 3, 2}
{1, 2, 2, 3, 3, 3}
{1, 2, 3, 1, 1, 1}
{1, 2, 3, 1, 1, 2}
{1, 2, 3, 1, 1, 3}
{1, 2, 3, 1, 2, 1}
{1, 2, 3, 1, 2, 2}
{1, 2, 3, 1, 2, 3}
{1, 2, 3, 1, 3, 1}
{1, 2, 3, 1, 3, 2}
{1, 2, 3, 1, 3, 3}
{1, 2, 3, 2, 1, 1}
{1, 2, 3, 2, 1, 2}
{1, 2, 3, 2, 1, 3}
{1, 2, 3, 2, 2, 1}
{1, 2, 3, 2, 2, 2}
{1, 2, 3, 2, 2, 3}
{1, 2, 3, 2, 3, 1}
{1, 2, 3, 2, 3, 2}
{1, 2, 3, 2, 3, 3}
{1, 2, 3, 3, 1, 1}
{1, 2, 3, 3, 1, 2}
{1, 2, 3, 3, 1, 3}
{1, 2, 3, 3, 2, 1}
{1, 2, 3, 3, 2, 2}
{1, 2, 3, 3, 2, 3}
{1, 2, 3, 3, 3, 1}
{1, 2, 3, 3, 3, 2}
{1, 2, 3, 3, 3, 3}
{1, 1, 1, 2, 3, 4}
{1, 1, 2, 1, 3, 4}
{1, 1, 2, 2, 3, 4}
{1, 1, 2, 3, 1, 4}
```

```
{1, 1, 2, 3, 2, 4}
{1, 1, 2, 3, 3, 4}
{1, 1, 2, 3, 4, 1}
{1, 1, 2, 3, 4, 2}
{1, 1, 2, 3, 4, 3}
{1, 1, 2, 3, 4, 4}
{1, 2, 1, 1, 3, 4}
{1, 2, 1, 2, 3, 4}
{1, 2, 1, 3, 1, 4}
{1, 2, 1, 3, 2, 4}
{1, 2, 1, 3, 3, 4}
{1, 2, 1, 3, 4, 1}
{1, 2, 1, 3, 4, 2}
{1, 2, 1, 3, 4, 3}
{1, 2, 1, 3, 4, 4}
{1, 2, 2, 1, 3, 4}
{1, 2, 2, 2, 3, 4}
{1, 2, 2, 3, 1, 4}
{1, 2, 2, 3, 2, 4}
{1, 2, 2, 3, 3, 4}
{1, 2, 2, 3, 4, 1}
{1, 2, 2, 3, 4, 2}
{1, 2, 2, 3, 4, 3}
{1, 2, 2, 3, 4, 4}
{1, 2, 3, 1, 1, 4}
{1, 2, 3, 1, 2, 4}
{1, 2, 3, 1, 3, 4}
{1, 2, 3, 1, 4, 1}
{1, 2, 3, 1, 4, 2}
{1, 2, 3, 1, 4, 3}
{1, 2, 3, 1, 4, 4}
{1, 2, 3, 2, 1, 4}
{1, 2, 3, 2, 2, 4}
{1, 2, 3, 2, 3, 4}
{1, 2, 3, 2, 4, 1}
{1, 2, 3, 2, 4, 2}
```

```
{1, 2, 3, 2, 4, 3}
{1, 2, 3, 2, 4, 4}
{1, 2, 3, 3, 1, 4}
{1, 2, 3, 3, 2, 4}
{1, 2, 3, 3, 3, 4}
{1, 2, 3, 3, 4, 1}
{1, 2, 3, 3, 4, 2}
{1, 2, 3, 3, 4, 3}
{1, 2, 3, 3, 4, 4}
{1, 2, 3, 4, 1, 1}
{1, 2, 3, 4, 1, 2}
{1, 2, 3, 4, 1, 3}
{1, 2, 3, 4, 1, 4}
{1, 2, 3, 4, 2, 1}
{1, 2, 3, 4, 2, 2}
{1, 2, 3, 4, 2, 3}
{1, 2, 3, 4, 2, 4}
{1, 2, 3, 4, 3, 1}
{1, 2, 3, 4, 3, 2}
{1, 2, 3, 4, 3, 3}
{1, 2, 3, 4, 3, 4}
{1, 2, 3, 4, 4, 1}
{1, 2, 3, 4, 4, 2}
{1, 2, 3, 4, 4, 3}
{1, 2, 3, 4, 4, 4}
{1, 1, 2, 3, 4, 5}
{1, 2, 1, 3, 4, 5}
{1, 2, 3, 1, 4, 5}
{1, 2, 3, 4, 1, 5}
{1, 2, 3, 4, 5, 1}
{1, 2, 2, 3, 4, 5}
{1, 2, 3, 2, 4, 5}
{1, 2, 3, 4, 2, 5}
{1, 2, 3, 4, 5, 2}
{1, 2, 3, 3, 4, 5}
{1, 2, 3, 4, 3, 5}
```

```
{1, 2, 3, 4, 5, 3}
{1, 2, 3, 4, 4, 5}
{1, 2, 3, 4, 5, 4}
{1, 2, 3, 4, 5, 5}
{1, 2, 3, 4, 5, 6}};


for(i=0;i<Rows(pop);i=i+1)
{
for(j=0;j<Columns(pop);j=j+1)
{
pop[i][j]=pop[i][j]-1;
}
}
return pop;
}
```

D.2.2 Utility Function: MatrixMapNuc.h

```
/*the argument 'cprior' is a row vector specifying the clustering, (i.e.
{{0,1,2,3,4,5}};  The resulting CMatrix is then read into the Mode
eval=(Cmatrix,XXXXX);*/

function MatrixMapNuc(cprior)
{
CMatrix={4,4};

for(aa=0;aa<Columns(cprior);aa=aa+1)
{
pm = cprior[0][aa];
if(aa==0){i=0; j=1;}
if(aa==1){i=0; j=2;}
if(aa==2){i=0; j=3;}
if(aa==3){i=1; j=2;}
if(aa==4){i=1; j=3;}
if(aa==5){i=2; j=3;}
```

```
if(pm==0){CMatrix[i][j]:=t;   CMatrix[j][i]:=t;   }
if(pm==1){CMatrix[i][j]:=t*R1;CMatrix[j][i]:=t*R1;}
if(pm==2){CMatrix[i][j]:=t*R2;CMatrix[j][i]:=t*R2;}
if(pm==3){CMatrix[i][j]:=t*R3;CMatrix[j][i]:=t*R3;}
if(pm==4){CMatrix[i][j]:=t*R4;CMatrix[j][i]:=t*R4;}
if(pm==5){CMatrix[i][j]:=t*R5;CMatrix[j][i]:=t*R5;}
}
return CMatrix;
}
```

D.2.3 Utility Function: sum.h

```
function sum(value)
{
for(j=0; j<Columns(value); j=j+1)
{
if (j==0)
{
sum_val = value[0][j];
}
if (j>0)
{
sum_val = sum_val + value[0][j];
}
}
return sum_val;
}
```

D.3 nuc_table.h

```
###graphing and descriptive statistics work

nuc_table<-function(theta1,theta2,theta3,theta4,theta5,theta6,p1,p2,p3,p4,reps,
brA,brB,brAa,brAb)
{
simo.loops<-reps
```

```
thetaAC<-theta1
thetaAG<-theta2
thetaAT<-theta3
thetaCG<-theta4
thetaCT<-theta5
thetaGT<-theta6
true.pi<-c(p1,p2,p3,p4)

R1.true<-thetaAG/thetaAC
R2.true<-thetaAT/thetaAC
R3.true<-thetaCG/thetaAC
R4.true<-thetaCT/thetaAC
R5.true<-thetaGT/thetaAC
R.truth<-c(1,R1.true,R2.true,R3.true,R4.true,R5.true)

branch_vector<-c(brA,brA,brB,brB,brAa,brAb)
true.ts<-2*(sum(branch_vector)/(length(branch_vector)-1))*(thetaAG*
(true.pi[1]*true.pi[3])+thetaCT*(true.pi[2]*true.pi[4]))
true.tv<-2*(sum(branch_vector)/(length(branch_vector)-1))*(thetaAC*
(true.pi[1]*true.pi[2])+thetaAT*(true.pi[1]*true.pi[4])+thetaCG*
(true.pi[2]*true.pi[3])+thetaGT*(true.pi[3]*true.pi[4]))

true.tstv<-2*true.ts/true.tv

##expected number of substitutions
thetaAA<-1-thetaAC-thetaAG-thetaAT
thetaCC<-1-thetaAC-thetaCG-thetaCT
thetaGG<-1-thetaAG-thetaCG-thetaGT
thetaTT<-1-thetaAT-thetaCT-thetaGT
ES<-true.pi[1]*thetaAA+true.pi[2]*thetaCC+true.pi[3]*thetaGG+true.pi[4]
*thetaTT

#########################################################################
pop<-scan(file=paste("./utilities/pop.txt",sep=""),what='character',sep=',')
pop_new<-chartr('}',' ',pop)
```

```
pop_new<-chartr('{',' ',pop_new)
pop_new<-as.numeric(pop_new)
pop_mat<-matrix(pop_new-1,ncol=6,byrow=TRUE)
pop_char<-rep(0,nrow(pop_mat))
for(z in 1:nrow(pop_mat))
{
pop_char[z]<-paste(pop_mat[z,1],pop_mat[z,2],pop_mat[z,3],pop_mat[z,4],
pop_mat[z,5],pop_mat[z,6],sep="")
}

best_fits<-matrix(rep(0,simo.loops*19),ncol=19)

for(z in 1:simo.loops)
{
data<-scan(file=paste("./data/_TsTv",z,".txt",sep=""),what='character',sep=',')
data_new<-chartr('}',' ',data)
data_new<-chartr('{',' ',data_new)
data_new<-as.numeric(data_new)
tstv_temp<-matrix(data_new[1:(length(data_new)-1)],ncol=19,byrow=TRUE)

#scaling for how this was put together
tstv_temp[,4]<-tstv_temp[,4]
tstv_temp[,5]<-tstv_temp[,5]

#L2norm stuff
R.est<-tstv_temp[,14:19]
L2dist_temp<-rep(0,nrow(R.est))
for(k in 1:nrow(R.est))
{
diffs<-c(R.truth[1]-R.est[k,1],R.truth[2]-R.est[k,2],
 R.truth[3]-R.est[k,3],R.truth[4]-R.est[k,4],
 R.truth[5]-R.est[k,5],R.truth[6]-R.est[k,6])
L2dist_temp[k]<-sqrt(sum(diffs^2))
}

if(z==1){
```

```
tstv<-tstv_temp[,3]
ts_<-tstv_temp[,4]
tv_<-tstv_temp[,5]
aicc<-tstv_temp[,2]
delta<-tstv_temp[,2] - min(tstv_temp[,2])
aicn<-exp(-0.5*delta)/sum(exp(-0.5*delta))
L2norm<-L2dist_temp
R1<-tstv_temp[,15]
R2<-tstv_temp[,16]
R3<-tstv_temp[,17]
R4<-tstv_temp[,18]
R5<-tstv_temp[,19]
}
if(z>1){
tstv<-cbind(tstv,tstv_temp[,3])
ts_<-cbind(ts_,tstv_temp[,4])
tv_<-cbind(tv_,tstv_temp[,5])
delta<-tstv_temp[,2] - min(tstv_temp[,2])
aicc<-cbind(aicc,tstv_temp[,2])
aicn<-cbind(aicn,exp(-0.5*delta)/sum(exp(-0.5*delta)))
L2norm<-cbind(L2norm,L2dist_temp)
R1<-cbind(R1,tstv_temp[,15])
R2<-cbind(R2,tstv_temp[,16])
R3<-cbind(R3,tstv_temp[,17])
R4<-cbind(R4,tstv_temp[,18])
R5<-cbind(R5,tstv_temp[,19])
}

best_temp<-matrix(tstv_temp[tstv_temp[,2]==min(tstv_temp[,2]),],ncol=19,
byrow=FALSE)
if(nrow(best_temp)==1){best_fits[z,]<-best_temp}
if(nrow(best_temp)>1){
sortord<-abs(best_temp[,3]-true.tstv)
best_temp<-cbind(best_temp,sortord)
best_temp1<-best_temp[order(best_temp[,ncol(best_temp)]),]
best_fits[z,]<-best_temp1[1,1:(ncol(best_temp1)-1)]
```

```
}
}

tstv_id<-rep(0,nrow(tstv_temp))
for(i in 1:nrow(tstv_temp))
{
if(tstv_temp[i,9]!=tstv_temp[i,8] && tstv_temp[i,9]!=tstv_temp[i,10] &&
tstv_temp[i,9]!=tstv_temp[i,11] && tstv_temp[i,9]!=tstv_temp[i,13] &&
tstv_temp[i,12]!=tstv_temp[i,8] && tstv_temp[i,12]!=tstv_temp[i,10] &&
tstv_temp[i,12]!=tstv_temp[i,11] && tstv_temp[i,12]!=tstv_temp[i,13]){
tstv_id[i]<-1
}
else{
tstv_id[i]<-0
}
}

##overall weights and ave
weight.est<-matrix(rep(0,ncol(tstv)*3),ncol=3)
for(j in 1:ncol(tstv))
{
weight_temp<-rep(0,nrow(tstv))
weight_temp_ts<-rep(0,nrow(tstv))
weight_temp_tv<-rep(0,nrow(tstv))
for(i in 1:nrow(tstv))
{
weight_temp[i]<-aicn[i,j]*tstv[i,j]
weight_temp_ts[i]<-aicn[i,j]*ts_[i,j]
weight_temp_tv[i]<-aicn[i,j]*tv_[i,j]

weight.est[j,1]<-sum(weight_temp)
weight.est[j,2]<-sum(weight_temp_ts)
weight.est[j,3]<-sum(weight_temp_tv)
}
}
```

```
##overall weights and ave
weight.est.temp<-rep(0,ncol(tstv))
R1.weight.est<-weight.est.temp
R2.weight.est<-weight.est.temp
R3.weight.est<-weight.est.temp
R4.weight.est<-weight.est.temp
R5.weight.est<-weight.est.temp
for(j in 1:ncol(tstv))
{
weight_temp<-rep(0,nrow(tstv))
R1.weight_temp<-weight_temp
R2.weight_temp<-weight_temp
R3.weight_temp<-weight_temp
R4.weight_temp<-weight_temp
R5.weight_temp<-weight_temp
for(i in 1:nrow(tstv))
{
weight_temp[i]<-aicn[i,j]*tstv[i,j]
weight.est[j]<-sum(weight_temp)
R1.weight_temp[i]<-aicn[i,j]*R1[i,j]
R1.weight.est[j]<-sum(R1.weight_temp)
R2.weight_temp[i]<-aicn[i,j]*R2[i,j]
R2.weight.est[j]<-sum(R2.weight_temp)
R3.weight_temp[i]<-aicn[i,j]*R3[i,j]
R3.weight.est[j]<-sum(R3.weight_temp)
R4.weight_temp[i]<-aicn[i,j]*R4[i,j]
R4.weight.est[j]<-sum(R4.weight_temp)
R5.weight_temp[i]<-aicn[i,j]*R5[i,j]
R5.weight.est[j]<-sum(R5.weight_temp)
}
}

##tstv weights and ave
aic.bio.temp<-aicc[tstv_id==1,]
for(j in 1:ncol(aic.bio.temp))
{
```

```
delta<-aic.bio.temp[,j] - min(aic.bio.temp[,j])
if(j==1){
aicn.wt<-exp(-0.5*delta)/sum(exp(-0.5*delta))
}
if(j>1){
aicn.wt<-cbind(aicn.wt,exp(-0.5*delta)/sum(exp(-0.5*delta)))
}
}
##use the weights to create the tstv averages
tstv.bio<-tstv[tstv_id==1,]
R1.bio<-R1[tstv_id==1,]
R2.bio<-R2[tstv_id==1,]
R3.bio<-R3[tstv_id==1,]
R4.bio<-R4[tstv_id==1,]
R5.bio<-R5[tstv_id==1,]
bio.wt<-rep(0,ncol(tstv.bio))
R1.bio.wt<-bio.wt
R2.bio.wt<-bio.wt
R3.bio.wt<-bio.wt
R4.bio.wt<-bio.wt
R5.bio.wt<-bio.wt

tstv.bio.ts<-ts_[tstv_id==1,]
tstv.bio.tv<-tv_[tstv_id==1,]
bio.wt<-matrix(rep(0,ncol(tstv.bio)*3),ncol=3)
for(j in 1:ncol(tstv.bio))
{
bio.wt.temp<-rep(0,nrow(tstv.bio))
R1.bio.wt.temp<-bio.wt.temp
R2.bio.wt.temp<-bio.wt.temp
R3.bio.wt.temp<-bio.wt.temp
R4.bio.wt.temp<-bio.wt.temp
R5.bio.wt.temp<-bio.wt.temp
bio.wt.temp.ts<-rep(0,nrow(tstv.bio))
bio.wt.temp.tv<-rep(0,nrow(tstv.bio))
for(i in 1:nrow(tstv.bio))
```

```
{
bio.wt.temp[i]<-aicn.wt[i,j]*tstv.bio[i,j]
bio.wt.temp.ts[i]<-aicn.wt[i,j]*tstv.bio.ts[i,j]
bio.wt.temp.tv[i]<-aicn.wt[i,j]*tstv.bio.tv[i,j]
bio.wt[j,1]<-sum(bio.wt.temp)
bio.wt[j,2]<-sum(bio.wt.temp.ts)
bio.wt[j,3]<-sum(bio.wt.temp.tv)
R1.bio.wt.temp[i]<-aicn.wt[i,j]*R1.bio[i,j]
R1.bio.wt[j]<-sum(R1.bio.wt.temp)
R2.bio.wt.temp[i]<-aicn.wt[i,j]*R2.bio[i,j]
R2.bio.wt[j]<-sum(R2.bio.wt.temp)
R3.bio.wt.temp[i]<-aicn.wt[i,j]*R3.bio[i,j]
R3.bio.wt[j]<-sum(R3.bio.wt.temp)
R4.bio.wt.temp[i]<-aicn.wt[i,j]*R4.bio[i,j]
R4.bio.wt[j]<-sum(R4.bio.wt.temp)
R5.bio.wt.temp[i]<-aicn.wt[i,j]*R5.bio[i,j]
R5.bio.wt[j]<-sum(R5.bio.wt.temp)

}
}

##non-tstv weights and ave
aic.nbio.temp<-aicc[tstv_id==0,]
for(j in 1:ncol(aic.nbio.temp))
{
delta<-aic.nbio.temp[,j] - min(aic.nbio.temp)
if(j==1){
aicnn.wt<-exp(-0.5*delta)/sum(exp(-0.5*delta))
}
if(j>1){
aicnn.wt<-cbind(aicnn.wt,exp(-0.5*delta)/sum(exp(-0.5*delta)))
}
}
##use the weights to create the non-tstv averages
ntstv.bio<-tstv[tstv_id==0,]
R1.nbio<-R1[tstv_id==0,]
```

```
R2.nbio<-R2[tstv_id==0,]
R3.nbio<-R3[tstv_id==0,]
R4.nbio<-R4[tstv_id==0,]
R5.nbio<-R5[tstv_id==0,]
nbio.wt<-rep(0,ncol(ntstv.bio))
R1.nbio.wt<-nbio.wt
R2.nbio.wt<-nbio.wt
R3.nbio.wt<-nbio.wt
R4.nbio.wt<-nbio.wt
R5.nbio.wt<-nbio.wt
ntstv.bio.ts<-ts_[tstv_id==0,]
ntstv.bio.tv<-tv_[tstv_id==0,]
nbio.wt<-matrix(rep(0,ncol(ntstv.bio)*3),ncol=3)
for(j in 1:ncol(ntstv.bio))
{
nbio.wt.temp<-rep(0,nrow(ntstv.bio))
R1.nbio.wt.temp<-nbio.wt.temp
R2.nbio.wt.temp<-nbio.wt.temp
R3.nbio.wt.temp<-nbio.wt.temp
R4.nbio.wt.temp<-nbio.wt.temp
R5.nbio.wt.temp<-nbio.wt.temp
nbio.wt.temp.ts<-rep(0,nrow(ntstv.bio))
nbio.wt.temp.tv<-rep(0,nrow(ntstv.bio))
for(i in 1:nrow(ntstv.bio))
{
nbio.wt.temp[i]<-aicnn.wt[i,j]*ntstv.bio[i,j]
nbio.wt.temp.ts[i]<-aicnn.wt[i,j]*ntstv.bio.ts[i,j]
nbio.wt.temp.tv[i]<-aicnn.wt[i,j]*ntstv.bio.tv[i,j]
nbio.wt[j,1]<-sum(nbio.wt.temp)
nbio.wt[j,2]<-sum(nbio.wt.temp.ts)
nbio.wt[j,3]<-sum(nbio.wt.temp.tv)
R1.nbio.wt.temp[i]<-aicnn.wt[i,j]*R1.nbio[i,j]
R1.nbio.wt[j]<-sum(R1.nbio.wt.temp)
R2.nbio.wt.temp[i]<-aicnn.wt[i,j]*R2.nbio[i,j]
R2.nbio.wt[j]<-sum(R2.nbio.wt.temp)
R3.nbio.wt.temp[i]<-aicnn.wt[i,j]*R3.nbio[i,j]
```

```
R3.nbio.wt[j]<-sum(R3.nbio.wt.temp)
R4.nbio.wt.temp[i]<-aicnn.wt[i,j]*R4.nbio[i,j]
R4.nbio.wt[j]<-sum(R4.nbio.wt.temp)
R5.nbio.wt.temp[i]<-aicnn.wt[i,j]*R5.nbio[i,j]
R5.nbio.wt[j]<-sum(R5.nbio.wt.temp)

}
}

#best fit L2 norm stuff
L2.best.fit_temp<-rep(0,nrow(best_fits))
for(i in 1:nrow(best_fits))
{
diff<-best_fits[i,14:19] - R.truth
L2.best.fit_temp[i]<-sqrt(sum(diff^2))
}


#put it all together
best.fit<-matrix(cbind(204,mean(best_fits[,3],na.rm=TRUE),mean(best_fits[,4]
,na.rm=TRUE),mean(best_fits[,5],na.rm=TRUE)),ncol=4)
wt.est<-matrix(cbind(205,mean(weight.est[,1],na.rm=TRUE),mean(weight.est[,2]
,na.rm=TRUE),mean(weight.est[,3],na.rm=TRUE)),ncol=4)
wt.tstv<-matrix(cbind(206,mean(bio.wt[,1],na.rm=TRUE),mean(bio.wt[,2],
na.rm=TRUE),mean(bio.wt[,3],na.rm=TRUE)),ncol=4)
wt.ntstv<-matrix(cbind(207,mean(nbio.wt[,1],na.rm=TRUE),mean(nbio.wt[,2],
na.rm=TRUE),mean(nbio.wt[,3],na.rm=TRUE)),ncol=4)
med.est<-matrix(cbind(seq(1,nrow(tstv)),apply(tstv,1,median),
apply(ts_,1,median),apply(tv_,1,median)),ncol=4)
ave.est<-matrix(cbind(seq(1,nrow(tstv)),apply(tstv,1,mean),apply(ts_,1,mean)
,apply(tv_,1,mean)),ncol=4)
sd.est<-matrix(cbind(seq(1,nrow(tstv)),apply(tstv,1,sd),apply(ts_,1,sd),
apply(tv_,1,sd)),ncol=4)
L2.mean<-matrix(apply(L2norm,1,mean),nrow=nrow(tstv))
L2.sd<-matrix(apply(L2norm,1,sd)/sqrt(simo.loops),nrow=nrow(tstv))
R.wt.est<-c(1,mean(R1.weight.est),mean(R2.weight.est),mean(R3.weight.est),
```

```
mean(R4.weight.est),mean(R5.weight.est))
R.wt.tstv<-c(1,mean(R1.bio.wt),mean(R2.bio.wt),mean(R3.bio.wt),
mean(R4.bio.wt),mean(R5.bio.wt))
R.wt.ntstv<-c(1,mean(R1.nbio.wt),mean(R2.nbio.wt),mean(R3.nbio.wt),
mean(R4.nbio.wt),mean(R5.nbio.wt))

L2.wt.est<-sqrt(sum((R.wt.est-R.truth)^2))
L2.wt.tstv<-sqrt(sum((R.wt.tstv-R.truth)^2))
L2.wt.ntstv<-sqrt(sum((R.wt.ntstv-R.truth)^2))

tstvnmm<-matrix(cbind(tstv_id,ave.est[,1:4]),ncol=5)
tstvm<-matrix(tstvnmm[tstvnmm[,1]==1],ncol=5)
tstvnm<-matrix(tstvnmm[tstvnmm[,1]==0],ncol=5)
nontstv<-tstvnm
tstvnm_all<-tstvnm
tstvm_all<-tstvm
tstvnm<-tstvnm[tstvnm[,2]!=1,]
tstvm<-tstvm[tstvm[,2]!=203,]
tstvm<-tstvm[tstvm[,2]!=19,]
tstvm<-tstvm[tstvm[,2]!=60,]

jc69<-matrix(tstvnmm[tstvnmm[,2]==1],ncol=5)
k80<-matrix(tstvnmm[tstvnmm[,2]==19],ncol=5)
tn93<-matrix(tstvnmm[tstvnmm[,2]==60],ncol=5)
gtr<-matrix(tstvnmm[tstvnmm[,2]==203],ncol=5)

allmod<-matrix(tstvnmm[,3],nrow=nrow(tstvnmm))
allmod[,ncol(allmod)]<-allmod[,ncol(allmod)]/sqrt(simo.loops)
allmod_and<-rbind(
cbind(mean(best_fits[,3]),sd(best_fits[,3])/sqrt(simo.loops),
mean(L2.best.fit_temp)),
cbind(mean(weight.est[,1]),sd(weight.est[,1])/sqrt(simo.loops),L2.wt.est),
cbind(mean(bio.wt[,1]),sd(bio.wt[,1])/sqrt(simo.loops),L2.wt.tstv),
cbind(mean(nbio.wt[,1]),sd(nbio.wt[,1])/sqrt(simo.loops),L2.wt.ntstv))
allmod_temp<-cbind(ave.est[,2],sd.est[,2]/sqrt(simo.loops),L2.mean)
allmod_and<-rbind(allmod_and,allmod_temp)
```

```
allmod_and<-cbind(round(allmod_and[,1],3),round(allmod_and[,2],4),
round(allmod_and[,3],3))

pop_charplus<-rbind("Truth","Best Fit","Ave Model","Ts/Tv Ave",
"Non-Ts/Tv Ave",matrix(pop_char,ncol=1))
tableframe<-cbind(pop_charplus,rbind(cbind(round(true.tstv,digits=3),"NA",
"0"),allmod_and))

xlimul<-203
xliml<-1

#############convergence plots
ave.mov<-matrix(rep(0,ncol(tstv)*nrow(tstv)),ncol=ncol(tstv))
wt.est.mov<-matrix(rep(0,ncol(tstv)),ncol=ncol(tstv))
wt.bio.mov<-matrix(rep(0,ncol(tstv)),ncol=ncol(tstv))
best.fit.mov<-matrix(rep(0,ncol(tstv)),ncol=ncol(tstv))
wt.nbio.mov<-matrix(rep(0,ncol(tstv)),ncol=ncol(tstv))
for(i in 1:nrow(ave.mov))
{
for(j in 1:ncol(ave.mov))
{
ave.mov[i,j]<-mean(tstv[i,1:j])
wt.est.mov[1,j]<-mean(weight.est[1:j])
wt.bio.mov[1,j]<-mean(bio.wt[1:j])
best.fit.mov[1,j]<-mean(best_fits[1:j,3])
wt.nbio.mov[1,j]<-mean(nbio.wt[1:j])
}
}
#ave.mov<-cbind(seq(1,nrow(ave.mov)),ave.mov)
jc69.mov<-ave.mov[1,]
k80.mov<-ave.mov[19,]
tn93.mov<-ave.mov[60,]
gtr.mov<-ave.mov[203,]

library(hwriter)
data.table<-data.frame(tableframe)
```

```
cols<-c("Model","Ts/Tv Estimate","Standard Error","L2 Norm")
colnames(data.table)<-cols
outfile<-"_Nuc_Table"
extensiont<-".html"
p=openPage(paste(outfile,extensiont,sep=""))
hwrite('Nucleotide Rev-Space Table', p)
hwrite(data.table,style='padding:5px', p)
closePage(p)

pdf(file=paste("_Mov_Ave.pdf",sep=""))

#Convergence Plot
ylimll<-min(k80.mov[200:length(k80.mov)],tn93.mov[200:length(tn93.mov)],
gtr.mov[200:length(gtr.mov)],true.tstv,wt.est.mov[,200:ncol(wt.est.mov)],
wt.bio.mov[,200:ncol(wt.bio.mov)],best.fit.mov[,200:ncol(best.fit.mov)],
wt.nbio.mov[200:ncol(wt.nbio.mov)])
ylimul<-max(k80.mov[200:length(k80.mov)],tn93.mov[200:length(tn93.mov)],
gtr.mov[200:length(gtr.mov)],true.tstv,wt.est.mov[,200:ncol(wt.est.mov)],
wt.bio.mov[,200:ncol(wt.bio.mov)],best.fit.mov[,200:ncol(best.fit.mov)],
wt.nbio.mov[200:ncol(wt.nbio.mov)])
plot(seq(1,ncol(ave.mov)),jc69.mov,cex=0,lty=1,
xlab='Replicates',ylab='TS/TV',ylim=c(ylimll,ylimul))
lines(seq(1,ncol(ave.mov)),k80.mov,cex=0,col='orange')
lines(seq(1,ncol(ave.mov)),tn93.mov,cex=0,col='blue')
lines(seq(1,ncol(ave.mov)),gtr.mov,cex=0,col='brown')
lines(seq(1,ncol(ave.mov)),best.fit.mov,col='yellow')
lines(seq(1,ncol(ave.mov)),wt.est.mov,col='darkkhaki')
lines(seq(1,ncol(ave.mov)),wt.bio.mov,col='green')
lines(seq(1,ncol(ave.mov)),wt.nbio.mov,col='maroon')

abline(a=true.tstv,b=0,lty=2,col='black')

dev.off()
}
```

Appendix E

Codon Modeling Programs

E.1 SimoData.R

```
#############################################################################
source(paste("./utilities/translate.R",sep=""))
source(paste("./utilities/transl_nuc_codon.R",sep=""))
source(paste("./utilities/generate_taxa.R",sep=""))
source(paste("./utilities/amino_test_cpu.R",sep=""))
source(paste("./utilities/s_cnt.txt",sep=""))
source(paste("./utilities/n_cnt.txt",sep=""))

###initial data setup
#codon frequencies
vector_c<-matrix(rep(1/4,12),ncol=3)
vector_c[,1]<-c(.4,.3,.2,.1)
vector_c[,2]<-c(.1,.2,.3,.4)
vector_c[,3]<-c(.2,.4,.1,.3)

#create the 1x61 vector of codon frequencies from the nucleotide, base
#frequency
index<-1
vector_temp<-matrix(rep(0,4*64),ncol=4)
vector<-matrix(rep(0,4*61),ncol=4)
Cfreq_vector_temp<-matrix(rep(0,3*61),ncol=3)
Cfreq_vector<-rep(0,61)

for(i in 1:4)
{
```

```
for(j in 1:4)
{
for(k in 1:4)
{
vector_temp[index,]<-c(index,i,j,k)
index<-index+1
}
}
}
index<-1
#map the 64x64 matrix nucleotide positions for a 61x61 model
#[universal genetic code]
for(z in 1:64)
{
if(z!=49 && z!=51 && z!=57)
{
vector[index,1]<-index
vector[index,2:4]<-vector_temp[z,2:4]
index<-index+1
}
}

Cfreq_vector_temp<-matrix(rep(0,3*61),ncol=3)
codon_c<-rep(0,61)
TAA<-vector_c[4,1]*vector_c[1,2]*vector_c[1,3]
TAG<-vector_c[4,1]*vector_c[1,2]*vector_c[3,3]
TGA<-vector_c[4,1]*vector_c[3,2]*vector_c[1,3]
stop<-1-(TAA+TAG+TGA)
for(z in 1:61)
{
for(zz in 1:ncol(Cfreq_vector_temp))
{
vectorcol<-zz+1
Cfreq_vector_temp[z,zz]<-vector_c[vector[z,vectorcol],zz]
}
codon_c[z]<-(Cfreq_vector_temp[z,1]*Cfreq_vector_temp[z,2]*
Cfreq_vector_temp[z,3])/stop
}

#synonymous and non-synonymous changes map
all<-scan(file=paste("./utilities/_dNdS_Matrix.txt",sep=""),what= 'character',sep=',')
```

```
all_new<-chartr('}',' ',all)
all_new<-chartr('{',' ',all_new)
cat(all_new,file=paste("./utilities/_dNdS_Matrix_tempz.txt",sep=""))
data<-scan(file=paste("./utilities/_dNdS_Matrix_tempz.txt",sep=""),what=double(0))
data.all<-matrix(data,ncol=61,byrow=TRUE)

#amino acid frequency and codon to amino coding map
aminofreqs<-matrix(scan(file=paste("./utilities/amino_sub_matrix.csv",sep=""),
sep=','),ncol=20,byrow=TRUE)
Ramino_maps<-matrix(scan(file=paste("./utilities/amino_codon_map.csv",sep=""),sep=',',
what=double(0)),byrow=TRUE,nrow=8)

Rmatrix<-function(freq,alpha_,matrix,amino_vals,amino_bins,amino_maps_)
{
index<-1
tempoutdata<-matrix
vector_temp<-matrix(rep(0,4*64),ncol=4)
vector<-matrix(rep(0,4*61),ncol=4)
for(i in 1:4)
{
for(j in 1:4)
{
for(k in 1:4)
{
vector_temp[index,]<-c(index,i,j,k)
index<-index+1
}
}
}
index<-1
for(z in 1:64)
{
if(z!=49 && z!=51 && z!=57)
{
vector[index,1]<-index
vector[index,2:4]<-vector_temp[z,2:4]
index<-index+1
}
}

outdata<-matrix(rep(0,61*61),ncol=61)
```

```
for(a in 1:61)
{
for(b in 1:61)
{
if(matrix[a,b]!=0)
{
temp<-vector[a,2:4]==vector[b,2:4]
for(c in 1:3)
{
if(temp[c]==0){
base_pos<-c+1
cval<-c
break
}
}

if(matrix[a,b]==1){
outdata[a,b]<-alpha_*freq[vector[b,base_pos],cval]
}

if(matrix[a,b]==2){
temp_amino<-rep(0,2)
temp_hyphy<-rep(0,2)
for(bz in 1:ncol(amino_maps_))
{
for(az in 3:nrow(amino_maps_))
{
if(amino_maps_[az,bz]==a)
{
temp_hyphy[1]<-amino_maps_[2,bz]
temp_amino[1]<-amino_maps_[1,bz]}
if(amino_maps_[az,bz]==b)
{
temp_hyphy[2]<-amino_maps_[2,bz]
temp_amino[2]<-amino_maps_[1,bz]}
if(sum(temp_hyphy==c(0,0))==0){break}
}
if(sum(temp_hyphy==c(0,0))==0){break}
}
outdata[a,b]<-amino_vals[amino_bins[temp_amino[1],temp_amino[2]]]*
freq[vector[b,base_pos],cval]
```

```
}
}
}
}
return(list(vector=vector,outdata=outdata))
}

#primary function generating and outputing the data
simo<-function(alpha,base_freqs,codon_freqs,indata,sequence_length,amino_freq,
amino_maps)
{
library(Matrix)
#get the amino acid substitution matrix (R)
Ramino_all<-amino_test(20,amino_freq)
Ramino_REV<-Ramino_all$aminoR
Ramino_bins<-Ramino_all$binR
Ramino_vals<-100*Ramino_all$Rvals
Ramino_prob<-Ramino_all$aminoP
amino_maps_<-amino_maps

Rmat<-Rmatrix(base_freqs,alpha,indata,Ramino_vals,Ramino_bins,amino_maps)
R<-Rmat$outdata
vector<-Rmat$vector

for(k in 1:ncol(R))
{
if(k==1){R[k,k]<- -1*sum(R[k,(k+1):ncol(R)])
}else{
if(k==ncol(R)){
R[k,k]<--1*sum(R[k,1:(k-1)])
}else{
R[k,k]<--1*(sum(R[k,1:(k-1)])+sum(R[k,(k+1):ncol(R)]))
}
}
}
x<-rep(0,3*sequence_length)
for(v in 0:sequence_length)
{
if(v==sequence_length)
{
for(cc in 1:nrow(base_freqs))
```

```
{
if(runif(1)<sum(base_freqs[cc:cc,ci+1]))
{
x[3*v]<-cc
break
}
}
if(x[3*v-2]==3 && x[3*v-1]==0 && x[3*v]==0){
v<-v-1}
if(x[3*v-2]==3 && x[3*v-1]==0 && x[3*v]==2){
v<-v-1}
if(x[3*v-2]==3 && x[3*v-1]==2 && x[3*v]==0){
v<-v-1}
}
if(v<sequence_length)
{
for(ci in 0:2)
{
for(cc in 1:nrow(base_freqs))
{
if(runif(1)<sum(base_freqs[1:cc,ci+1]))
{
x[3*v+ci]<-cc
break
}
}
}
if(x[3*v+ci-2]==3 && x[3*v+ci-1]==0 && x[3*v+ci]==0){
v<-v-1}
if(x[3*v+ci-2]==3 && x[3*v+ci-1]==0 && x[3*v+ci]==2){
v<-v-1}
if(x[3*v+ci-2]==3 && x[3*v+ci-1]==2 && x[3*v+ci]==0){
v<-v-1}
}
}
r<-x+1
rcod<-transl_nuc_codon(vector,r)

P05<-expm(.5*R)
a<-generate_taxa(indata=rcod,Pt=P05)
b<-generate_taxa(indata=rcod,Pt=P05)
```

```
a.char<-translate(a)
b.char<-translate(b)

cat("#Wiget1",file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n")
cat(a.char,file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",append=TRUE)
cat("#Wiget2",file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",append=TRUE)
cat(b.char,file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),"\n",append=TRUE)

#print the tree on the file for use with the HyPhy GUI
cat("\n","(Wiget1,Wiget2);",file=paste("./HyPhy_Data/simoz",bigreps,".txt",sep=""),
"\n",append=TRUE)
}
```

E.1.1 Utility Function: translate.R

```
#program to simulate data dNdS codon data using the universal genetic code
#utility function translating the numeric vector values into character output
#stop codons are excluded
#this is exclusively for the universal genetic code.
translate<-function(input)
{
translation<-rep('z',length(input))
for(k in 1:length(input))
{
if(input[k]==1) {translation[k]<-'a a a'}
if(input[k]==2) {translation[k]<-'a a c'}
if(input[k]==3) {translation[k]<-'a a g'}
if(input[k]==4) {translation[k]<-'a a t'}
if(input[k]==5) {translation[k]<-'a c a'}
if(input[k]==6) {translation[k]<-'a c c'}
if(input[k]==7) {translation[k]<-'a c g'}
if(input[k]==8) {translation[k]<-'a c t'}
if(input[k]==9) {translation[k]<-'a g a'}
if(input[k]==10) {translation[k]<-'a g c'}
if(input[k]==11) {translation[k]<-'a g g'}
if(input[k]==12) {translation[k]<-'a g t'}
if(input[k]==13) {translation[k]<-'a t a'}
if(input[k]==14) {translation[k]<-'a t c'}
if(input[k]==15) {translation[k]<-'a t g'}
if(input[k]==16) {translation[k]<-'a t t'}
if(input[k]==17) {translation[k]<-'c a a'}
```

```
if(input[k]==18) {translation[k]<-'c a c'}
if(input[k]==19) {translation[k]<-'c a g'}
if(input[k]==20) {translation[k]<-'c a t'}
if(input[k]==21) {translation[k]<-'c c a'}
if(input[k]==22) {translation[k]<-'c c c'}
if(input[k]==23) {translation[k]<-'c c g'}
if(input[k]==24) {translation[k]<-'c c t'}
if(input[k]==25) {translation[k]<-'c g a'}
if(input[k]==26) {translation[k]<-'c g c'}
if(input[k]==27) {translation[k]<-'c g g'}
if(input[k]==28) {translation[k]<-'c g t'}
if(input[k]==29) {translation[k]<-'c t a'}
if(input[k]==30) {translation[k]<-'c t c'}
if(input[k]==31) {translation[k]<-'c t g'}
if(input[k]==32) {translation[k]<-'c t t'}
if(input[k]==33) {translation[k]<-'g a a'}
if(input[k]==34) {translation[k]<-'g a c'}
if(input[k]==35) {translation[k]<-'g a g'}
if(input[k]==36) {translation[k]<-'g a t'}
if(input[k]==37) {translation[k]<-'g c a'}
if(input[k]==38) {translation[k]<-'g c c'}
if(input[k]==39) {translation[k]<-'g c g'}
if(input[k]==40) {translation[k]<-'g c t'}
if(input[k]==41) {translation[k]<-'g g a'}
if(input[k]==42) {translation[k]<-'g g c'}
if(input[k]==43) {translation[k]<-'g g g'}
if(input[k]==44) {translation[k]<-'g g t'}
if(input[k]==45) {translation[k]<-'g t a'}
if(input[k]==46) {translation[k]<-'g t c'}
if(input[k]==47) {translation[k]<-'g t g'}
if(input[k]==48) {translation[k]<-'g t t'}
#if(input[k]== {translation[k]<-'t a a'}  #Stop codon
if(input[k]==49) {translation[k]<-'t a c'}
#if(input[k]==) {translation[k]<-'t a g'}  #Stop codon
if(input[k]==50) {translation[k]<-'t a t'}
if(input[k]==51) {translation[k]<-'t c a'}
if(input[k]==52) {translation[k]<-'t c c'}
if(input[k]==53) {translation[k]<-'t c g'}
if(input[k]==54) {translation[k]<-'t c t'}
#if(input[k]== {translation[k]<-'t g a'}  #Stop codon
if(input[k]==55) {translation[k]<-'t g c'}
```

```
if(input[k]==56) {translation[k]<-'t g g'}
if(input[k]==57) {translation[k]<-'t g t'}
if(input[k]==58) {translation[k]<-'t t a'}
if(input[k]==59) {translation[k]<-'t t c'}
if(input[k]==60) {translation[k]<-'t t g'}
if(input[k]==61) {translation[k]<-'t t t'}
}
return(translation)
}
```

E.1.2 Utility Function: transl_nuc_codon.R

```
transl_nuc_codon<-function(codon_ref,sequence)
{
outsequence<-rep(0,length(sequence)/3)
size<-(length(sequence)/3)
for(i in 1:size)
{
for(j in 1:nrow(codon_ref))
{
row1<-i*3-2
row25<-i*3-1
row2<-i*3
if(sequence[row1]==codon_ref[j,2] && sequence[row25]==codon_ref[j,3] &&
sequence[row2]==codon_ref[j,4] &&
(sequence[row1]!=4 || sequence[row25]!=1 || sequence[row2]!=1) &&
(sequence[row1]!=4 || sequence[row25]!=1 || sequence[row2]!=3) &&
(sequence[row1]!=4 || sequence[row25]!=3 || sequence[row2]!=1)
)
{
outsequence[i]<-codon_ref[j,1]
break
}
}
}
length<-1
for(k in 1:length(outsequence))
{
if(outsequence[k]!=0) {length<-length+1}
}
length<-length-1
```

```
outsequence_nonmissing<-rep(0,length)
cnt<-1
for(l in 1:length(outsequence))
{
if(outsequence[l]!=0)
{
outsequence_nonmissing[cnt]<-outsequence[l]
cnt<-cnt+1
}
}
return(outsequence_nonmissing)
}
```

E.1.3 Utility Function: generate_taxa.R

```
generate_taxa<-function(indata,Pt)
{
temp<-runif(length(indata))
outdata<-rep(0,length(indata))
for(j in 1:length(indata))
{
row<-indata[j]
for(z in 1:ncol(Pt))
{
if(temp[j]<=Pt[row,row]){outdata[j]<-indata[j]
break} else{
if(z<row){
if(temp[j]<=Pt[row,row]+sum(Pt[row,1:z])){outdata[j]<-z
break}}
if(z>=row){
if(temp[j]<=sum(Pt[row,1:z])){outdata[j]<-z
break}}
}
}
}
return(outdata)
}
```

E.1.4 Utility Function: amino_test_cpu.R

```
#create the amino acid substitution matrix
amino_test<-function(col_vals,matrixvals)
```

```
{
aminoP<-matrix(rep(0,col_vals^2),ncol=col_vals)
for(i in 1:col_vals)
{
den<-sum(matrixvals[i,])
for(j in 1:col_vals)
{
aminoP[i,j]<-matrixvals[i,j]/den
}
}
library(expm)
aminoRtemp<-logm(aminoP)/.5
aminoR<-matrix(rep(0,col_vals^2),ncol=col_vals)
for(i in 1:col_vals)
{
for(j in 1:col_vals)
{
if(i != j)
{
if(abs(aminoRtemp[i,j])<=1e-12){
aminoR[i,j]<-0
}
if(abs(aminoRtemp[i,j])>1e-12){
aminoR[i,j]<-aminoRtemp[i,j]
}
}
}
}
for(i in 1:col_vals)
{
if(i==1){
aminoR[i,i]<--1*sum(aminoR[i,2:col_vals])
}
else{
minus<-i-1
plus<-i+1
if(i < col_vals){
aminoR[i,i]<--1*(sum(aminoR[i,1:minus])+sum(aminoR[i,plus:col_vals]))
}
if(i == col_vals){
aminoR[i,i]<--1*(sum(aminoR[i,1:col_vals]))
```

```
}
}
}
binR<-matrix(rep(0,col_vals*col_vals),ncol=col_vals)
for(i in 1:col_vals)
{
for(j in 1:col_vals)
{
if(aminoR[i,j]<5e-5) {
binR[i,j]<-1}
if(aminoR[i,j]>=5e-5 && aminoR[i,j]<1e-4){
binR[i,j]<-2}
if(aminoR[i,j]>=1e-4 && aminoR[i,j]<5e-4){
binR[i,j]<-3}
if(aminoR[i,j]>=5e-4 && aminoR[i,j]<1e-3){
binR[i,j]<-4}
if(aminoR[i,j]>=1e-3 && aminoR[i,j]<5e-3){
binR[i,j]<-5}
if(aminoR[i,j]>=5e-3){
binR[i,j]<-6}
}
}
bin1<-0; bin2<-0; bin3<-0; bin4<-0; bin5<-0; bin6<-0
for(i in 1:col_vals)
{
for(j in 1:col_vals)
{
if(binR[i,j]==1){
if(length(bin1)==1 && bin1==0){
bin1<-aminoR[i,j]} else{
bin1<-append(bin1,aminoR[i,j])
}
}
if(binR[i,j]==2){
if(length(bin1)==1 && bin2==0){
bin2<-aminoR[i,j]} else{
bin2<-append(bin2,aminoR[i,j])
}
}
if(binR[i,j]==3)
{
```

```
if(length(bin3)==1 && bin3==0){
bin3<-aminoR[i,j]} else{
bin3<-append(bin3,aminoR[i,j])
}
}
if(binR[i,j]==4)
{
if(length(bin4)==1 && bin4==0){
bin4<-aminoR[i,j]} else{
bin4<-append(bin4,aminoR[i,j])
}
}
if(binR[i,j]==5)
{
if(length(bin5)==1 && bin5==0){
bin5<-aminoR[i,j]} else{
bin5<-append(bin5,aminoR[i,j])
}
}
if(binR[i,j]==6)
{
if(length(bin6)==1 && bin6==0){
bin6<-aminoR[i,j]} else{
bin6<-append(bin6,aminoR[i,j])
}
}
}
}
Rvals<-rep(0,6)
Rvals[1]<-(1/length(bin1))*sum(bin1)
Rvals[2]<-(1/length(bin2))*sum(bin2)
Rvals[3]<-(1/length(bin3))*sum(bin3)
Rvals[4]<-(1/length(bin4))*sum(bin4)
Rvals[5]<-(1/length(bin5))*sum(bin5)
Rvals[6]<-(1/length(bin6))*sum(bin6)
return(list(aminoR=aminoR,binR=binR,Rvals=Rvals,aminoP=aminoP))
}
```

E.1.5 Utility Vector: s_cnt.txt

```
#synonymous change counts
```

```
s_cnt=
c(1,1,1,1,3,3,3,3,2,1,2,1,2,2,0,2,1,1,1,1,3,3,3,3,4,3,4,3,4,3,4,3,1,1,1,1,3,3,3,3,3,3,
3,3,3,3,3,3,1,1,3,3,3,3,1,0,1,2,1,2,1)
```

E.1.6 Utility Vector: n_cnt.txt

```
#nonsynonymous change counts
n_cnt=
c(7,8,7,8,6,6,6,6,6,8,7,8,7,7,9,7,7,8,7,8,6,6,6,6,4,6,5,6,5,6,5,6,7,8,7,8,6,6,6,6,5,6,
6,6,6,6,6,6,6,6,4,6,5,6,7,7,7,5,8,6,8)
```

E.2 GA_HYPHY_codon.h

```
#include "utilities\sum.h";
#include "utilities\CreateTemplates.h";
#include "utilities\MatrixMap_dNdS_marker.h";
#include "utilities\MatrixMap_codon_EFM.h";
#include "utilities\IsChildViable.h";
#include "utilities\MakeStringCanonical.h";
#include "utilities\ModelGeneticCode.h";
#include "utilities\one_step_dNdS.h";
#include "utilities\SimpleCodonFreq.h";
#include "utilities\One_Step_Cnts_Matrix.h";
#include "utilities\SumNucCnts.h";
#include "utilities\dNdS_est_utility.h";
#include "utilities\EFM_assign.h";

/*************************************************************************************
************************  FIRST_POPULATION  ******************************************
*****Function to produce the first set of values for use by the GA********************
*************************************************************************************/

function first_population(samples, length, ClassCount)
{
datafile="HyPhy_Data\simoz";
dataformat=".txt";
indatafile=datafile+bigreps+dataformat;
DataSet myData = ReadDataFile (indatafile);
DataSetFilter codonFilter = CreateFilter (myData, 3,"","","TAA,TAG,TGA");
HarvestFrequencies (obsFreqs_temp, codonFilter, 3, 1, 1);
```

```
SumNuc = SumNucCnts(obsFreqs_temp);
CFreqs = SimpleCodonFreq(obsFreqs_temp);
one_step=CreateTemplates(0);
synonymous=CreateTemplates(3);
nonsynonymous=CreateTemplates(4);
Template_dNdS=one_step_dNdS(61,one_step,synonymous,nonsynonymous);

/*create a 2 row matrix map for the GA individuals to map their parameter values
to the associated codon freqs*/
dS=0;
dN=0;
for(i=0;i<Rows(Template_dNdS);i=i+1)
{
if(Template_dNdS[i][0]==1){
dS=dS+1;}
if(Template_dNdS[i][0]==2){
dN=dN+1;}
}

marker_row={1,(dS/2)+(dN/2)};
for(i=0;i<Columns(marker_row);i=i+1)
{
if(i<(dS/2)){
marker_row[0][i]=1; }
else {
marker_row[0][i]=2; }
}
codon_freq_map = MatrixMap_dNdS_marker(Template_dNdS,marker_row,61);
One_Step_Cnts_Matrix(CMatrix_marker);
dNdS_est_utility(ModelGeneticCode,CFreqs,s_cnt,n_cnt);
EFM_assign(obsFreqs_temp,CMatrix_marker,ModelGeneticCode);

next_child = {1,length};
model = next_child;
check_duplicate=next_child;
next_pop = {samples, length};
index = 0;
for(_j=0; _j< samples; _j=_j+1)
{
for(z=0; z< length; z=z+1)
{
```

```
next_child[0][z]=Random(0, ClassCount)$1;
}
aChild = MakeStringCanonical(next_child, ClassCount);
for(zz=0;zz<length;zz=zz+1)
{
next_pop[_j][zz] = aChild[0][zz];
}

if(_j>0)
{
viability = IsChildViable(aChild,_j,next_pop);

while(viability == 0)
{
for(z=0; z< length; z=z+1)
{
next_child[0][z]=Random(0, (_jj+1))$1;
}
aChild = MakeStringCanonical(next_child,  ClassCount);
for(zz=0;zz<length;zz=zz+1)
{
next_pop[index][zz] = aChild[0][zz];
}
viability = IsChildViable(aChild,_j,next_pop);
}
}
else
{
for(_z=0; _z<length; _z=_z+1)
{
next_pop[index][_z]=aChild[0][_z];
}
}
index = index+1;
}
return model;
return check_duplicate;
return next_pop;
}

/*****************************************************************************
```

```
************************  EVALUATE  ********************************************
******* Likelihood shell (evaluation of population) program ************************
*******************************************************************************/
function evaluate(prior, mut, para,check_dup)
{
/*initialize storage and parameters*/
population = prior;
x = {Rows(prior),1};
columns_prior = Columns(prior)+67;
row = {1,columns_prior};
model={1,Columns(prior)};
model_best = {1,Columns(prior)+1};
trans_temp = {Rows(prior),2};
dNdS_models = {Rows(prior),15+(2*columns_prior)};
model_parameters = {Rows(prior),columns_prior};
p = x;
marker_ = x;
AIC = x;

for(z=0; z<Rows(prior); z=z+1)
{
REPLACE_TREE_STRUCTURE = 1;
/**Model Likelihoods for individulas in the population **/
/*Synonymous Substitution in a single cluster currently*/
for(y_=0; y_<263; y_=y_+1)
{
if(y_<67){row[0][y_]=0;}
}
/*NonSynonymous portion of the model - clustered via populations*/
for(y=0; y<Columns(prior); y=y+1)
{
row[0][y+67]=prior[z][y]+1;
}
global R1; global R2; global R3; global R4; global R5; global R6; global R7;
global R8; global R9; global R10; global R11; global R12; global R13; global R14;
global R15; global R16; global R17; global R18;global R19; global R20; global R21;
global R22; global R23; global R24; global R25; global R26; global R27; global R28;
global R29; global R30; global R31; global R32; global R33; global R34; global R35;
global R36; global R37; global R38; global R39; global R40;

R1=.1; R2=.1; R3=.1; R4=.1; R5=.1; R6=.1; R7=.1; R8=.1; R9=.1; R10=.1; R11=.1; R12=.1;
```

```
R13=.1; R14=.1; R15=.1; R16=.1; R17=.1; R18=.1; R19=.1; R20=.1; R21=.1; R22=.1;
R23=.1; R24=.1; R25=.1; R26=.1; R27=.1; R28=.1; R29=.1; R30=.1; R31=.1; R32=.1;
R33=.1; R34=.1; R35=.1; R36=.1; R37=.1; R38=.1; R39=.1; R40=.1;

MatrixMap_codon_EFM(Template_dNdS,row,61,EFM); /*output is CMatrix*/
Model eval = (CMatrix , CFreqs, 0);
Tree myTree = (Wiget1,Wiget2);
LikelihoodFunction theLikFun = (codonFilter , myTree);
Optimize (MLEs, theLikFun);

/*AIC[z][0] = -2*MLEs[1][0] + (2*MLEs[1][1])*Log(codonFilter.sites); mBIC*/
AIC[z][0] = -2*MLEs[1][0] + (2*MLEs[1][1]*codonFilter.sites)/
(codonFilter.sites - MLEs[1][1] - 1); /*AICc*/

dS_parametervalue=MLEs[1][1]-1;
dS_parameter=MLEs[0][dS_parametervalue];
dNdS_models[z][7]=MLEs[1][0];
dNdS_models[z][6]=AIC[z][0];
dNdS_models[z][3]=MLEs[1][1];
dNdS_models[z][8]=codonFilter.sites;
/*********************************************************************/
for(_hh=0;_hh<columns_prior;_hh=_hh+1)
{
dNdS_models[z][_hh+15] = row[0][_hh];
}
/**save the model parameter values from the MLEs object**/
for(_i=0; _i<columns_prior;_i=_i+1)
{
if(row[0][_i]==0) {model_parameters[z][_i]=1;}
else
{
for(_ii=0; _ii<MLEs[1][2];_ii=_ii+1){
if(row[0][_i] == _ii+1){
model_parameters[z][_i] = MLEs[0][_ii];
}
}
}
}

/*store the parameter estimates for each model*/
for(_hhh=0;_hhh<columns_prior;_hhh=_hhh+1)
```

```
{
dNdS_models[z][_hhh+columns_prior+15] = model_parameters[z][_hhh];
}

/**store the numerator and denominator of the dS/dN ratio estimates by Muse 96**/
/*the numerator dS (synonymous) piece*/
dS_vals = {1,(dS/2)};
sS_vals = {1,(dS/2)};
for(_ds=0;_ds<dS/2;_ds=_ds+1)
{
freq1 = codon_freq_map[0][_ds];
freq2 = codon_freq_map[1][_ds];
/*numerator of the estimate*/
dS_vals[0][_ds]=dS_parameter*(EFM[freq1][freq2]*CFreqs[freq1][0]+EFM[freq2][freq1]*
CFreqs[freq2][0]);
sS_vals[0][_ds]=dS_parameter*(EFM[freq1][freq2]*EFM[freq2][freq1]);
}
sumdS = sum(dS_vals,2);

/*the numerator dN (nonsynonymous) piece*/
dN_vals = {1,(dN/2)};
sN_vals = {1,(dN/2)};
for(_dn=0;_dn<dN/2;_dn=_dn+1)
{
freq1 = codon_freq_map[0][_dn+(dS/2)];
freq2 = codon_freq_map[1][_dn+(dS/2)];
/*numerator of the estimate*/
dN_vals[0][_dn]=(model_parameters[z][_dn+(dS/2)]*dS_parameter)*(EFM[freq1][freq2]*
CFreqs[freq1][0]+EFM[freq2] [freq1]*CFreqs[freq2][0]);
sN_vals[0][_dn]=(model_parameters[z][_dn+(dS/2)]*dS_parameter)*((EFM[freq1][freq2]*
EFM[freq2][freq1]));
}
sumdN = sum(dN_vals,2);

dNdS_models[z][4]=sumdS/den_dS_vals;
dNdS_models[z][5]=sumdN/den_dN_vals;
dNdS_models[z][2]=dNdS_models[z][5]/dNdS_models[z][4];
dNdS_models[z][9]=sum(sS_vals,2);
dNdS_models[z][10]=sum(sN_vals,2);
dNdS_models[z][11]=dNdS_models[z][10]/dNdS_models[z][9];
```

```
/*************************************************************************/
if (z==0)
{
max = AIC[z][0];
min = AIC[z][0];
for (q=0; q<Coluns(prior); q=q+1){
model[0][q] = prior[z][q];
}
for (qq=0; qq<Columns(prior);qq=qq+1){
model_best[0][qq+1]=prior[z][qq];
}
model_best[0][0]=counter;
}
if (z>0)
{
if (max < AIC[z][0]){
max = AIC[z][0];
}
if (min > AIC[z][0])
{
min = AIC[z][0];
for (q=0; q<Columns(prior); q=q+1){
model[0][q] = prior[z][q];
}
for (qq=0; qq<Columns(prior);qq=qq+1){
model_best[0][qq+1]=prior[z][qq];
}
model_best[0][0]=counter;
}
}
}
range = max - min;

/**Calculation of percentiles (use '1-' for AIC - smaller is better)**/
for(k=0; k<Rows(prior); k=k+1)
{
p[k][0] = 1-((AIC[k][0]-min)/range);
/**selection of individuals by the probability associated with their percentiles**/
if (Random(0,1) <= p[k][0]){
marker_[k][0] = 1;
}
```

```
else{
marker_[k][0] = 0;
}
/*storage of estimates and weights - column 1: population, 2: percentile, 3: ratio */
/*note: trans_temp[x][0]=dS & trans_temp[x][1]=dN*/
dNdS_models[k][0]=counter;
dNdS_models[k][1]=p[k][0];
}
outputdata="_dNdS_models";
outputdataformat=".txt";
outfile=outputdata+bigreps+smallreps+outputdataformat;
fprintf(outfile,dNdS_models,"\n");

/** stoping criteria code (either update the best model accross iterations or increase
the counter**/
if (min < best || counter==1)
{
if(counter==1){
best = min;
count_ = 1;
}
if(min<best)
{
flag_duplicate=0;
for(l=0;l<Columns(prior);l=l+1){
flag_duplicate = (model[0][l]!=check_dup[0][l]) + flag_duplicate;
}
if(flag_duplicate!=0){
best = min;
count_ = 1;
}
else{
count_=count_+1;
}
}
}
else{
count_ = count_ + 1;
}
return model;
return model_best;
```

```
return population;
return p;
return best;
return count_;
}

/*****************************************************************************
**********************NEXT_POPULATION*****************************************
****** Function to generate children out of the population ********************
*****************************************************************************/
function next_population(pop,resample,ClassCount,ref,aic_,mut)
{
next_pop = {resample,Columns(pop)};
aChild = {1,Columns(pop)};
check_duplicate=aChild;
next_child = aChild;
ref_pop = {.2*resample,1};

tenp = .1*resample;
tenp2 = tenp + .1*resample;
/*main portion of the next_pop generation*/
for(_l=0; _l<resample; _l=_l+1)
{
/*keep the best fitting individual*/
if(_l==0)
{
tag=0;
for(_lg=0; _lg<Rows(pop) && tag==0; _lg=_lg+1)
{
if(ref[_lg][0]==1)
{
/**************  check **********************/
tag=1;
ref_pop[0][0]=_lg;
for(_lgg=0; _lgg<Columns(pop);_lgg=_lgg+1)
{
next_pop[0][_lgg] = pop[_lg][_lgg];
check_duplicate[0][_lgg]=pop[_lg][_lgg];
}
break;
}
```

```
}
}

/*keep a random allocation of 10% of the individuals*/
if (_l>0 && _l<tenp)
{
test_=0;*/
/*ensure a duplicate individual is not selected*/
while (test_ == 0)
{
test_=1;
val1 = Random(1,Rows(pop))$1;
for(j=0;j<_l;j=j+1)
{
if(ref_pop[j][0] == val1)
{
test_=0;
break;
}
}
}
for(_lgg=0; _lgg<Columns(pop);_lgg=_lgg+1){
next_pop[_l][_lgg] = pop[val1][_lgg];
}
ref_pop[_l][0]=val1;
}

/*mutate 10% (or more accurately, the percentage specified above (see &tenp and
&tenp2)) of the individuals*/
if (_l>=tenp && _l<tenp2)
/*if (_l>0 && _l<=tenp)*/
{
test_=0;
/*ensure a duplicate individual is not selected*/
while (test_ == 0)
{
test_=1;
val1 = Random(1,Rows(pop))$1;
for(j=0;j<_l;j=j+1)
{
if(ref_pop[j][0] == val1)
```

```
{
test_=0;
break;
}
}
}

/***location of mutation control***/
for(_lgg=0; _lgg<Columns(pop);_lgg=_lgg+1)
{
temp = Random(0,1);
if (temp < mut){
next_child[0][_lgg] =Random(0,ClassCount)$1;
}
else{
next_child[0][_lgg] = pop[val1][_lgg];
}

}
aChild = MakeStringCanonical(next_child, ClassCount);

for(zz=0;zz<Columns(pop);zz=zz+1){
next_pop[_l][zz] = aChild[0][zz];
}
viability = IsChildViable(aChild,_l,next_pop);
while(viability == 0)
{
for(_lgg; _lgg<Columns(pop);_lgg=_lgg+1)
{
if (Random(0,1) < mut)
{
aChild[0][_lgg] =Random(0,ClassCount)$1;
}
else
{
aChild[0][_lgg] = pop[val1][_lgg];
}
}
aChild = MakeStringCanonical(next_child, ClassCount);
viability = IsChildViable(aChild,_lg,next_pop);
if(viability != 0)
```

```
{
for(zz=0;zz<Columns(pop);zz=zz+1)
{
next_pop[_l][zz] = aChild[0][zz];
}
}
}
ref_pop[_l][0]=val1;
}

/*crossover the remaining 80% of the individuals (or more accurately, the reminder not
specified from the &tenp and &tenp1 vars above) for the next population*/
if(_l>=tenp2)
/*if(_l>tenp)*/
{
test_flag=0;
val1 = Random(0,Rows(pop))$1;
val2 = Random(0,Rows(pop))$1;
vala = Random(0,1);
/*duplicate selection is allowed (duplicated with best fit or mutated individuals) but
selection is percentile ranked here*/
while(val2==val1 || vala>=ref[val1][0] || vala>=ref[val2][0])
{
while(val2==val1)
{
val1 = Random(0,Rows(pop))$1;
val2 = Random(0,Rows(pop))$1;
}
while (vala>=ref[val2][0])
{
val2=Random(0,Rows(pop))$1;
vala=Random(0,1);
}
while (vala>=ref[val1][0])
{
val1=Random(0,Rows(pop))$1;
vala = Random(0,1);
}
}
/*fitness scores for the scaled fitness-based crossover*/
AICd={{aic_[val1][0]}{aic_[val2][0]}};
```

```
delta1=Exp((-1/2)*(aic_[val1][0]-Min(AICd,5)));
delta2=Exp((-1/2)*(aic_[val2][0]-Min(AICd,5)));
AICsum=delta1+delta2;

fit1 = delta1/AICsum;
fit2 = delta2/AICsum;
for (h=0; h<Columns(pop); h=h+1)
{
if( Random(0,1) < fit1){
next_child[0][h] = pop[val1][h];
}
else{
next_child[0][h] = pop[val2][h];
}
}
aChild = MakeStringCanonical(next_child, ClassCount);
for(zz=0;zz<Columns(pop);zz=zz+1){
next_pop[_l][zz] = aChild[0][zz];
}

viability = IsChildViable(aChild,_l,next_pop);
while(viability == 0)
{
for(z=0; z< Columns(pop); z=z+1) /*if canonical recombo child is a duplicate, generate
a random child*/
{
next_child[0][z]=Random(0, ClassCount)$1;
}
aChild = MakeStringCanonical(next_child, ClassCount);

viability = IsChildViable(aChild,_l,next_pop);
if(viability != 0)
{
for(zz=0;zz<Columns(pop);zz=zz+1){
next_pop[_l][zz] = aChild[0][zz];
}
}
}
}
return check_duplicate;
```

```
return next_pop;
}

/*****************************************************************************
******************************************************************************
******** linking all of the above functions together in a 'while' loop *************
******************************************************************************/
/*pop_size = initial population size for the GA*/
/*resample = population sizes for all following the initial population*/
/*model_length = number of substitution parameters being evaluated by the GA*/
/*numclusters = number of clusters being used for the GA run*/
/*termination = number of populations with no change in the best-fit for termination
of the GA*/
/*mut = mutation probability*/
/*cata and cata_like - dummy variables for possible changes*/
function GA(pop_size, resample, model_length, numclusters, termination, mut, cata,
cata_like)
{
counter=1;
count_=0;
/**This generates the initial population**/
first_population(pop_size, model_length , numclusters);

/**Evaluate the initial population (output is next_pop - has rows equal to the number
of selected from percentile selection mtd)**/
evaluate(next_pop, mut, numclusters,check_duplicate);
while (count_ < termination)
{
/** output is next_pop and is the number of children created from 'random selection'
of next_pop **/
next_population(population,resample,numclusters,p,AIC,mut);

counter=counter+1;

evaluate(next_pop, mut, numclusters,check_duplicate);
}
return next_pop;
}

/* Calculate the estimates for the model creating the simulated data*/
function MG94(parameters,para)
```

```
{
columns_prior = parameters+67;
model_parameters = {1,columns_prior};
true_model = {1,15};

row={1,263};
for(i=0; i<263; i=i+1)
{
if(i<67){row[0][i]=0;}
if(i>=67){row[0][i]=1;}
}
global R1;
R1=.1;

MatrixMap_codon_EFM(Template_dNdS,row,61,EFM); /*output is CMatrix*/
Model eval = (CMatrix , CFreqs, 0);
Tree myTree = (Wiget1,Wiget2);

LikelihoodFunction theLikFun = (codonFilter , myTree);
Optimize (MLEs, theLikFun);
/*AIC = -2*MLEs[1][0] + (2*MLEs[1][1])*Log(codonFilter.sites); mBIC*/
AIC = -2*MLEs[1][0] + (2*MLEs[1][1]*codonFilter.sites)/(codonFilter.sites - MLEs[1][1]
- 1); /*AICc*/
dS_parametervalue=MLEs[1][1]-1;
dS_parameter=MLEs[0][dS_parametervalue];
true_model[0][1] = AIC;
true_model[0][7] = MLEs[1][0];
true_model[0][0] = MLEs[0][0];
true_model[0][8] = codonFilter.sites;

/*************************************************************************/
/**save the model parameter values from the MLEs object**/
for(_i=0; _i<columns_prior;_i=_i+1)
{
if(row[0][_i]==0) {model_parameters[0][_i]=1;}
else
{
for(_ii=0; _ii<MLEs[1][2];_ii=_ii+1)
{
if(row[0][_i] == _ii+1)
{
```

```
model_parameters[0][_i] = MLEs[0][_ii];
}
}
}
}
/**store the numerator and denominator of the dS/dN ratio estimates by Muse 96**/
/*the numerator dS (synonymous) piece*/
dS_vals = {1,(dS/2)};
sS_vals = {1,(dS/2)};
for(_ds=0;_ds<dS/2;_ds=_ds+1)
{
freq1 = codon_freq_map[0][_ds];
freq2 = codon_freq_map[1][_ds];
/*numerator of the estimate*/
dS_vals[0][_ds]=dS_parameter*(EFM[freq1][freq2]*CFreqs[freq1][0]+EFM[freq2][freq1]*
CFreqs[freq2][0]);
sS_vals[0][_ds]=dS_parameter*(EFM[freq1][freq2]*EFM[freq2][freq1]);
}
sumdS = sum(dS_vals,2);
true_model[0][2] = den_dS_vals;
true_model[0][3] = sumdS/den_dS_vals;

/*the numerator dN (nonsynonymous) piece*/
dN_vals = {1,(dN/2)};
sN_vals = {1,(dN/2)};
for(_dn=0;_dn<dN/2;_dn=_dn+1)
{
freq1 = codon_freq_map[0][_dn+(dS/2)];
freq2 = codon_freq_map[1][_dn+(dS/2)];
/*numerator of the estimate*/
dN_vals[0][_dn]=(model_parameters[0][_dn+(dS/2)]*dS_parameter)*(EFM[freq1][freq2]*
CFreqs[freq1][0]+EFM[freq2][freq1]*CFreqs[freq2][0]);
sN_vals[0][_dn]=(model_parameters[z][_dn+(dS/2)]*dS_parameter)*((EFM[freq1][freq2]*
EFM[freq2][freq1]));
}

sumdN = sum(dN_vals,2);
true_model[0][4]=den_dN_vals;
true_model[0][5]=sumdN/den_dN_vals;
true_model[0][6]=true_model[0][5]/true_model[0][3];
true_model[z][9]=sum(sS_vals,2);
```

```
true_model[z][10]=sum(sN_vals,2);
true_model[z][11]=true_model[z][10]/true_model[z][9];

outputdata="_true_model";
outputdataformat=".txt";
outfile=outputdata+bigreps+outputdataformat;
fprintf(outfile,true_model,"\n");
return 0;
}
```

E.2.1 Utility Function: CreateTemplates.h

```
function CreateTemplates (classID)
{
ModelGeneticCode = {
{14,13,14,13,
7,7,7,7,
19,5,19,5,
2,2,3,2,
12,11,12,11,
6,6,6,6,
19,19,19,19,
1,1,1,1,
16,15,16,15,
8,8,8,8,
20,20,20,20,
4,4,4,4,
10,9,10,9,
5,5,5,5,
10,17,18,17,
1,0,1,0 }
};
TemplateNames = {{"One-step","Two-step","Three-step","Synonymous","Non-synonymous",
"1-step Transitions","1-step Transversions"}};

ModelGeneticCodeSize = 61;
vShift = 0;
hShift = 0;
MatrixTemplate = {ModelGeneticCodeSize*ModelGeneticCodeSize,1};
matchCount = 0;
```

```
if (classID < 3)
{
for (h=0; h<63; h=h+1)
{
if (ModelGeneticCode[h]==10)
{
hShift = hShift+1;
continue;
}

vShift = hShift;

for (v=h+1; v<64; v=v+1)
{
if (ModelGeneticCode[v]==10)
{
vShift = vShift+1;
continue;
}

posCount = (h%4!=v%4)+(h$16!=v$16)+(h%16$4!=v%16$4)-1;

if (posCount == classID)
{
MatrixTemplate[matchCount] = (h-hShift)*ModelGeneticCodeSize + v-vShift;
matchCount = matchCount+1;
MatrixTemplate[matchCount] = (v-vShift)*ModelGeneticCodeSize + h-hShift;
matchCount = matchCount+1;
}
}
}
}
else
{
if (classID<5)
{
classID = 4-classID;
for (h=0; h<63; h=h+1)
{
if (ModelGeneticCode[h]==10)
{
```

```
hShift = hShift+1;
continue;
}
vShift = hShift;
for (v=h+1; v<64; v=v+1)
{
if (ModelGeneticCode[v]==10)
{
vShift = vShift+1;
continue;
}
if ((ModelGeneticCode[v]==ModelGeneticCode[h])==classID)
{
MatrixTemplate[matchCount] = (h-hShift)*ModelGeneticCodeSize + v-vShift;
matchCount = matchCount+1;
MatrixTemplate[matchCount] = (v-vShift)*ModelGeneticCodeSize + h-hShift;
matchCount = matchCount+1;
}
}
}
}
}
return MatrixTemplate;
}
```

E.2.2 Utility Function: MatrixMap_codon.h

```
/*********include and interpret the Templating function**********/
function MatrixMap_codon(Template,cprior,nonstop)
{
marker = 0;
pmarker = 0;
CMatrix={nonstop,nonstop};

for(_a=1;_a<3;_a=_a+1)
{
diff=0;
for(aa=0;aa<nonstop && diff==0;aa=aa+1)
{
set = (aa+1)*nonstop;
if (marker<nonstop*nonstop)
```

```
{
temp = Template[marker][1];
}
if (marker>=nonstop*nonstop)
{
temp = 0;
}
if (temp==0)
{
break;
}
while (temp < set && temp != 0 && diff==0)
{
set2 = Template[marker][1] - aa*nonstop;
pm = cprior[0][pmarker];

if(pm==0){CMatrix[aa][set2]:=t;
CMatrix[set2][aa]:=t;}
if(pm==1){CMatrix[aa][set2]:=t*R1;
CMatrix[set2][aa]:=t*R1;}
if(pm==2){CMatrix[aa][set2]:=t*R2;
CMatrix[set2][aa]:=t*R2;}
if(pm==3){CMatrix[aa][set2]:=t*R3;
CMatrix[set2][aa]:=t*R3;}
if(pm==4){CMatrix[aa][set2]:=t*R4;
CMatrix[set2][aa]:=t*R4;}
if(pm==5){CMatrix[aa][set2]:=t*R5;
CMatrix[set2][aa]:=t*R5;}
if(pm==6){CMatrix[aa][set2]:=t*R6;
CMatrix[set2][aa]:=t*R6;}
if(pm==7){CMatrix[aa][set2]:=t*R7;
CMatrix[set2][aa]:=t*R7;}
if(pm==8){CMatrix[aa][set2]:=t*R8;
CMatrix[set2][aa]:=t*R8;}
marker=marker+2;

last=temp;
temp = Template[marker][1];
if(temp>last){diff=0;}
else {diff=1;}
```

```
if(pmarker<262)
{
pmarker=pmarker+1;
}
}
}
}
return cprior;
return CMatrix;
}
```

E.2.3 Utility Function: MatrixMap_dNdS_marker.h

```
/*********include and interpret the Templating function**********/
function MatrixMap_dNdS_marker(Template,cprior,nonstop)
{
Template_marker={2,Columns(cprior)};
marker = 0;
pmarker = 0;
CMatrix_marker={nonstop,nonstop};

for(_a=1;_a<3;_a=_a+1)
{
diff=0;
for(aa=0;aa<nonstop && diff==0;aa=aa+1)
{
set = (aa+1)*nonstop;
if (marker<nonstop*nonstop)
{
temp = Template[marker][1];
}
if (marker>=nonstop*nonstop)
{
temp = 0;
}
if (temp==0)
{
break;
}
while (temp < set && temp != 0 && diff==0)
{
```

```
set2 = Template[marker][1] - aa*nonstop;
pm = cprior[0][pmarker];

if(pm==1){CMatrix_marker[aa][set2]=1;
CMatrix_marker[set2][aa]=1;}
if(pm==2){CMatrix_marker[aa][set2]=2;
CMatrix_marker[set2][aa]=2;}

Template_marker[0][pmarker]=set2;
Template_marker[1][pmarker]=aa;

marker=marker+2;

last=temp;

temp = Template[marker][1];
if(temp>last){diff=0;}
else {diff=1;}

if(pmarker<262)
{
pmarker=pmarker+1;
}
}
}
}

return Template_marker;
return CMatrix_marker;
}
```

E.2.4 Utility Function: MatrixMap_codon_EFM.h

```
/*********include and interpret the Templating function**********/
function MatrixMap_codon_EFM(Template,cprior,nonstop,EFM)
{
marker = 0;
pmarker = 0;
CMatrix={nonstop,nonstop};

for(_a=1;_a<3;_a=_a+1)
```

```
{
diff=0;
for(aa=0;aa<nonstop && diff==0;aa=aa+1)
{
set = (aa+1)*nonstop;
if (marker<nonstop*nonstop)
{
temp = Template[marker][1];
}
if (marker>=nonstop*nonstop)
{
temp = 0;
}
if (temp==0)
{
break;
}
while (temp < set && temp != 0 && diff==0)
{
set2 = Template[marker][1] - aa*nonstop;
pm = cprior[0][pmarker];

if(pm==0){CMatrix[aa][set2]:=t*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*EFM__[set2__][aa__];}
if(pm==1){CMatrix[aa][set2]:=t*R1*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R1*EFM__[set2__][aa__];}
if(pm==2){CMatrix[aa][set2]:=t*R2*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R2*EFM__[set2__][aa__];}
if(pm==3){CMatrix[aa][set2]:=t*R3*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R3*EFM__[set2__][aa__];}
if(pm==4){CMatrix[aa][set2]:=t*R4*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R4*EFM__[set2__][aa__];}
if(pm==5){CMatrix[aa][set2]:=t*R5*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R5*EFM__[set2__][aa__];}
if(pm==6){CMatrix[aa][set2]:=t*R6*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R6*EFM__[set2__][aa__];}
if(pm==7){CMatrix[aa][set2]:=t*R7*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R7*EFM__[set2__][aa__];}
if(pm==8){CMatrix[aa][set2]:=t*R8*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R8*EFM__[set2__][aa__];}
```

```
if(pm==9){CMatrix[aa][set2]:=t*R9*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R9*EFM__[set2__][aa__];}
if(pm==10){CMatrix[aa][set2]:=t*R10*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R10*EFM__[set2__][aa__];}
if(pm==11){CMatrix[aa][set2]:=t*R11*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R11*EFM__[set2__][aa__];}
if(pm==12){CMatrix[aa][set2]:=t*R12*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R12*EFM__[set2__][aa__];}
if(pm==13){CMatrix[aa][set2]:=t*R13*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R13*EFM__[set2__][aa__];}
if(pm==14){CMatrix[aa][set2]:=t*R14*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R14*EFM__[set2__][aa__];}
if(pm==15){CMatrix[aa][set2]:=t*R15*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R15*EFM__[set2__][aa__];}
if(pm==16){CMatrix[aa][set2]:=t*R16*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R16*EFM__[set2__][aa__];}
if(pm==17){CMatrix[aa][set2]:=t*R17*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R17*EFM__[set2__][aa__];}

if(pm==18){CMatrix[aa][set2]:=t*R18*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R18*EFM__[set2__][aa__];}
if(pm==19){CMatrix[aa][set2]:=t*R19*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R19*EFM__[set2__][aa__];}
if(pm==20){CMatrix[aa][set2]:=t*R20*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R20*EFM__[set2__][aa__];}
if(pm==21){CMatrix[aa][set2]:=t*R21*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R12*EFM__[set2__][aa__];}
if(pm==22){CMatrix[aa][set2]:=t*R22*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R22*EFM__[set2__][aa__];}
if(pm==23){CMatrix[aa][set2]:=t*R23*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R23*EFM__[set2__][aa__];}
if(pm==24){CMatrix[aa][set2]:=t*R24*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R24*EFM__[set2__][aa__];}
if(pm==25){CMatrix[aa][set2]:=t*R25*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R25*EFM__[set2__][aa__];}
if(pm==26){CMatrix[aa][set2]:=t*R26*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R26*EFM__[set2__][aa__];}

if(pm==27){CMatrix[aa][set2]:=t*R27*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R27*EFM__[set2__][aa__];}
if(pm==28){CMatrix[aa][set2]:=t*R28*EFM__[aa__][set2__];
```

```
CMatrix[set2][aa]:=t*R28*EFM__[set2__][aa__];}
if(pm==29){CMatrix[aa][set2]:=t*R29*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R29*EFM__[set2__][aa__];}
if(pm==30){CMatrix[aa][set2]:=t*R30*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R30*EFM__[set2__][aa__];}
if(pm==31){CMatrix[aa][set2]:=t*R31*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R31*EFM__[set2__][aa__];}
if(pm==32){CMatrix[aa][set2]:=t*R32*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R32*EFM__[set2__][aa__];}
if(pm==33){CMatrix[aa][set2]:=t*R33*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R33*EFM__[set2__][aa__];}
if(pm==34){CMatrix[aa][set2]:=t*R34*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R34*EFM__[set2__][aa__];}
if(pm==35){CMatrix[aa][set2]:=t*R35*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R35*EFM__[set2__][aa__];}

if(pm==36){CMatrix[aa][set2]:=t*R36*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R36*EFM__[set2__][aa__];}
if(pm==37){CMatrix[aa][set2]:=t*R37*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R37*EFM__[set2__][aa__];}
if(pm==38){CMatrix[aa][set2]:=t*R38*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R38*EFM__[set2__][aa__];}
if(pm==39){CMatrix[aa][set2]:=t*R39*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R39*EFM__[set2__][aa__];}
if(pm==40){CMatrix[aa][set2]:=t*R40*EFM__[aa__][set2__];
CMatrix[set2][aa]:=t*R40*EFM__[set2__][aa__];}

marker=marker+2;
last=temp;
temp = Template[marker][1];
if(temp>last){diff=0;}
else {diff=1;}

if(pmarker<262)
{
pmarker=pmarker+1;
}
}
}
}
```

```
return cprior;
return CMatrix;
}
```

E.2.5 Utility Function: IsChildViable.h

```
function IsChildViable (putativeChild, Index,population)
{
/**  evaluate the population for a match and stop at the first one (if it exists) **/
flag = 1;
for (_lr = 0; _lr < Index && flag != 0; _lr = _lr + 1)
{
flag = 0;
if(_lr < Index)
{
for (_lc = 0; _lc < Columns(population); _lc = _lc + 1)
{
flag = (putativeChild[0][_lc] != population[_lr][_lc]) + flag;
}
}
}

return flag;
}
```

E.2.6 Utility Function: MakeStringCanonical.h

```
/*update child vectors of the GA to follow uniqueness constraints*/
function MakeStringCanonical_new (randomModel)
{
classCount = ncol(randomModel);
startVectorDimension = ncol(randomModel);
compressedString = {classCount,1};
for (h=0; h<stateVectorDimension; h=h+1)
{
v = randomModel[h];
compressedString[v] = 1;
}
fprintf(stdout,"v=",v,"\n");
compressedString[0] = 0;
for (h=1; h<classCount; h=h+1)
{
```

```
compressedString[h] = compressedString[h]+compressedString[h-1];
}
for (h=0; h<stateVectorDimension; h=h+1)
{
v = randomModel[h];
v = compressedString[v];
randomModel[h] = v;
}
v = compressedString[classCount-1]+1;
if (v>1)
{
sortedOrder = {v,2};
for (h=0; h<v; h=h+1)
{
sortedOrder[h][0] = -1;
}
cc = 0;
for (h=0; h<stateVectorDimension; h=h+1)
{
hshift = randomModel[h];
if (sortedOrder[hshift][0] < 0)
{
sortedOrder[hshift][0] = cc;
sortedOrder[hshift][1] = hshift;
cc = cc+1;
}
}
sortedOrder = sortedOrder%1;
for (h=0; h<stateVectorDimension; h=h+1)
{
v = randomModel[h];
randomModel[h] = sortedOrder[v][0];
}
}
fprintf(stdout,"randomModel=",randomModel,"\n");
return randomModel;
}
```

E.2.7 Utility Function: ModelGeneticCode.h

```
ModelGeneticCode = {
```

```
{/*AAA*/14,/*AAC*/13,/*AAG*/14,/*AAT*/13,
/*ACA*/ 7,/*ACC*/ 7,/*ACG*/ 7,/*ACT*/  7,
/*AGA*/19,/*AGC*/ 5,/*AGG*/19,/*AGT*/  5,
/*ATA*/ 2,/*ATC*/ 2,/*ATG*/ 3,/*ATT*/  2,
/*CAA*/12,/*CAC*/11,/*CAG*/12,/*CAT*/ 11,
/*CCA*/ 6,/*CCC*/ 6,/*CCG*/ 6,/*CCT*/  6,
/*CGA*/19,/*CGC*/19,/*CGG*/19,/*CGT*/ 19,
/*CTA*/ 1,/*CTC*/ 1,/*CTG*/ 1,/*CTT*/  1,
/*GAA*/16,/*GAC*/15,/*GAG*/16,/*GAT*/ 15,
/*GCA*/ 8,/*GCC*/ 8,/*GCG*/ 8,/*GCT*/  8,
/*GGA*/20,/*GGC*/20,/*GGG*/20,/*GGT*/ 20,
/*GTA*/ 4,/*GTC*/ 4,/*GTG*/ 4,/*GTT*/  4,
/*TAA*/10,/*TAC*/ 9,/*TAG*/10,/*TAT*/  9,
/*TCA*/ 5,/*TCC*/ 5,/*TCG*/ 5,/*TCT*/  5,
/*TGA*/10,/*TGC*/17,/*TGG*/18,/*TGT*/ 17,
/*TTA*/ 1,/*TTC*/ 0,/*TTG*/ 1,/*TTT*/  0}
};

ModelGeneticCodeCnts = {
{/*AAA*/8,/*AAC*/9,/*AAG*/ 8,/*AAT*/ 9,
/*ACA*/ 9,/*ACC*/ 9,/*ACG*/ 9,/*ACT*/ 9,
/*AGA*/ 8,/*AGC*/ 9,/*AGG*/ 9,/*AGT*/ 9,
/*ATA*/ 9,/*ATC*/ 9,/*ATG*/ 9,/*ATT*/ 9,
/*CAA*/ 8,/*CAC*/ 9,/*CAG*/ 8,/*CAT*/ 9,
/*CCA*/ 9,/*CCC*/ 9,/*CCG*/ 9,/*CCT*/ 9,
/*CGA*/ 8,/*CGC*/ 9,/*CGG*/ 9,/*CGT*/ 9,
/*CTA*/ 9,/*CTC*/ 9,/*CTG*/ 9,/*CTT*/ 9,
/*GAA*/ 8,/*GAC*/ 9,/*GAG*/ 8,/*GAT*/ 9,
/*GCA*/ 9,/*GCC*/ 9,/*GCG*/ 9,/*GCT*/ 9,
/*GGA*/ 8,/*GGC*/ 9,/*GGG*/ 9,/*GGT*/ 9,
/*GTA*/ 9,/*GTC*/ 9,/*GTG*/ 9,/*GTT*/ 9,
/*TAA*/ 0,/*TAC*/ 7,/*TAG*/ 0,/*TAT*/ 7,
/*TCA*/ 9,/*TCC*/ 9,/*TCG*/ 9,/*TCT*/ 9,
/*TGA*/ 0,/*TGC*/ 9,/*TGG*/ 9,/*TGT*/ 9,
/*TTA*/ 9,/*TTC*/ 9,/*TTG*/ 9,/*TTT*/ 9 }
};
```

E.2.8 Utility Function: one_step_dNdS.h

```
function one_step_dNdS(nonstop,one,syn,nonsyn)
{
```

```
density=nonstop*nonstop;
MatrixTemplate_dNdS={density,2};
index=0;
for(x=0; x< density; x=x+1)
{
if(one[x][0]==0){break;}
else
{
for(z=0; z< density; z=z+1)
{
if(one[x][0]==syn[z][0])
{
MatrixTemplate_dNdS[index][0]=1;
MatrixTemplate_dNdS[index][1]=one[x][0];
index=index+1;
break;
}
}
}
}
for(_x=0; _x< density; _x=_x+1)
{
if(one[_x][0]==0){break;}
else
{
for(_z=0; _z< density; _z=_z+1)
{
if(one[_x][0]==nonsyn[_z][0])
{
MatrixTemplate_dNdS[index][0]=2;
MatrixTemplate_dNdS[index][1]=one[_x][0];
index=index+1;
break;
}
}
}
}
return MatrixTemplate_dNdS;
}
```

E.2.9 Utility Function: SimpleCodonFreq.h

```
function SimpleCodonFreq(obs)
{
CFreqs_temp = {4*4*4,1};
index=0;
for(i=0;i<4;i=i+1)
{
for(j=0;j<4;j=j+1)
{
for(k=0;k<4;k=k+1)
{
CFreqs_temp[index][0]=obs[i][0]*obs[j][1]*obs[k][2];
index=index+1;
}
}
}

/*determine length of freq vector for codons*/
#include "ModelGeneticCode.h";
sumval=0;
stop=0;
for(i=0;i<Columns(ModelGeneticCode);i=i+1)
{
if(ModelGeneticCode[0][i]==10)
{
sumval=sumval+1;
stop=stop+CFreqs_temp[0][i];
}
}

CFreqs_vv = {4*4*4-sumval,1};
index=0;
for(i=0;i<Columns(ModelGeneticCode);i=i+1)
{
if(ModelGeneticCode[0][i]!=10)
{
CFreqs_vv[index][0]=CFreqs_temp[i][0]/(1-stop);
index=index+1;
}
}
```

```
return CFreqs_vv;
}
```

E.2.10 Utility Function: One_Step_Cnts_Matrix.h

```
function One_Step_Cnts_Matrix(temp_matrix)
{
s_cnt={1,Columns(temp_matrix)};
for(i=0;i<Columns(temp_matrix);i=i+1)
{
for(j=0;j<Columns(temp_matrix);j=j+1)
{
if(temp_matrix[i][j]==1){s_cnt[0][i]=s_cnt[0][i]+1;}
}
}

n_cnt={1,Columns(temp_matrix)};
for(i=0;i<Columns(temp_matrix);i=i+1)
{
for(j=0;j<Columns(temp_matrix);j=j+1)
{
if(temp_matrix[i][j]==2){n_cnt[0][i]=n_cnt[0][i]+1;}
}
}

return s_cnt;
return n_cnt;
}
```

E.2.11 Utility Function: SumNucCnts.h

```
function SumNucCnts(freqs)
{
#include "ModelGeneticCode.h";
/* traverse for nuc freq sums */
_SumNucCnts={1,64};
index=0;
for (i=0;i<4;i=i+1)
{
for(j=0;j<4;j=j+1)
{
for(k=0;k<4;k=k+1)
```

```
{
_SumNucCnts[0][index]=freqs[i][0]+freqs[j][1]+freqs[k][2];
index=index+1;
}
}
}
return _SumNucCnts;
}
```

E.2.12 Utility Function: dNdS_est_utility.h

```
#include "One_Step_Cnts.h";
#include "SumNucCnts.h";
#include "ModelGeneticCode.h";
function dNdS_est_utility(code,freq,scnt,ncnt)
{
/*create the denominator constants for the dN dS estimates by Muse 96*/
/*dS section*/
den_dS={1,61};
index=0;
for(i=0;i<Columns(code);i=i+1)
{
if(code[0][i]!=10)
{
den_dS[0][index]=freq[index][0]*scnt[0][index];
index=index+1;
}
}
den_dS_vals=(1/3)*sum(den_dS,2);

/*dN section*/
den_dN={1,61};
index=0;
for(i=0;i<Columns(code);i=i+1)
{
if(code[0][i]!=10)
{
den_dN[0][index]=freq[index][0]*ncnt[0][index];
index=index+1;
}
}
```

```
den_dN_vals=(1/3)*sum(den_dN,2);
return den_dS_vals;
return den_dN_vals;
}
```

E.2.13 Utility Function: EFM_assign.h

```
/*Create an EFM for implementation of freqs into the model matrix with the Model
option of "mult_flag=0"*/
#include "ModelGeneticCode.h";
function EFM_assign(freq,marker,code)
{
RowID_temp = {Columns(code),Columns(freq)};
index=0;
for (i=0;i<Rows(freq);i=i+1) {
for (j=0; j<Rows(freq); j=j+1) {
for (k=0; k<Rows(freq); k=k+1) {
RowID_temp[index][0]=i;
RowID_temp[index][1]=j;
RowID_temp[index][2]=k;
index=index+1;
}
}
}

indexd=0;
for(i=0;i<Columns(code);i=i+1)
{
if (code[0][i]==10) {
indexd=indexd+1;
}
}
dim = Columns(code)-indexd;
RowID = {dim,3};
index=0;
for (i=0; i<Columns(code); i=i+1) {
if (code[0][i] != 10) {
RowID[index][0]=RowID_temp[i][0];
RowID[index][1]=RowID_temp[i][1];
RowID[index][2]=RowID_temp[i][2];
index=index+1;
```

```
}

}

EFM = {dim,dim};
for(i=0;i<Columns(marker);i=i+1)
{
for(j=0;j<Columns(marker);j=j+1)
{
if (marker[i][j] != 0) {
if (RowID[i][0] != RowID[j][0]) {
EFM[i][j]=freq[RowID[j][0]][0];
}
if (RowID[i][1] != RowID[j][1]) {
EFM[i][j]=freq[RowID[j][1]][1];
}
if (RowID[i][2] != RowID[j][2]) {
EFM[i][j]=freq[RowID[j][2]][2];
}
}
}
}
return EFM;
}
```

E.2.14 Utility Function: sum.h

```
function sum(value)
{
for(j=0; j<Columns(value); j=j+1)
{
if (j==0)
{
sum_val = value[0][j];
}
if (j>0)
{
sum_val = sum_val + value[0][j];
}
}
return sum_val;
```

```
}
```

E.3 Results_Data_loops.R

```
################################################################################
################################################################################
############ Programer: Sam Wilson
############ Purpose: Process the transition/transverion output dataset from HyPhy
################################################################################
################################################################################
#big = number of replicates run in the GA simulation
#small = maximum number of clusters analyzed by the GA simulation
#popn = population size for the GA simulation
dataloop <- function(big,small,popn)
{
#Set initial conditions
reps<-big
nodatafiles<-small
extension<-".txt"
popsize<-popn
loop<-nodatafiles
first<-83
last<-278
Columns<-196

############read-in the main data
for(j in 1:reps)
{
data.true<-scan(file=paste("./_true_model",j,extension,sep=""),what= 'character',
sep=',')
true_new<-chartr('}',' ',data.true)
true_new<-chartr('{',' ',true_new)
cat(true_new,file=paste("./_true_mod",j,extension,sep=""),append=TRUE)
}

for(i in 2:nodatafiles)
{
for(j in 1:reps)
{
data<-scan(file=paste("./_dNdS_models",j,i,extension,sep=""),what= 'character',
sep=',')
```

```
data_new<-chartr('}',' ',data)
data_new<-chartr('{',' ',data_new)
cat(data_new,file=paste("./_dNdS_modelz",i,extension,sep=""),append=TRUE)
}

trans<-matrix(scan(file=paste("./_dNdS_modelz",i,extension,sep="")),ncol=541,
byrow=TRUE)
models<-trans[,first:last]

trans12temp<-trans[,1:15]
nodupkey0<-trans[!duplicated(trans[,16:278]),]
nodupkey<-nodupkey0[,1:15]
truth12<-matrix(scan(file=paste("./_true_mod",j,extension,sep="")),ncol=15,byrow=TRUE)

cat(trans12temp,file=paste("./_trans",i,extension,sep=""))
cat(truth12,file=paste("./_truth",i,extension,sep=""))
cat(nodupkey,file=paste("./_nodupkey",i,extension,sep=""))
cat(models,file=paste("./_models",i,extension,sep=""))
}
}
```

E.4 Results_Data_Analysis.R

```
##############################################################################################
##############################################################################################
################### Programer: Sam Wilson
################### Purpose: Analysis of the GA simulation
##############################################################################################
##############################################################################################

Rmatrix<-function(freq,alpha_,matrix,amino_vals,amino_bins,amino_maps_)
{
index<-1
tempoutdata<-matrix
#setup the 64x64 matrix of nucleotide positions in the codons
vector_temp<-matrix(rep(0,4*64),ncol=4)
vector<-matrix(rep(0,4*61),ncol=4)
for(i in 1:4)
{
for(j in 1:4)
{
```

```
for(k in 1:4)
{
vector_temp[index,]<-c(index,i,j,k)
index<-index+1
}
}
}
index<-1
#map the 64x64 matrix nucleotide positions for a 61x61 model [universal genetic code]
for(z in 1:64)
{
if(z!=49 && z!=51 && z!=57)
{
vector[index,1]<-index
vector[index,2:4]<-vector_temp[z,2:4]
index<-index+1
}
}
outdata<-matrix(rep(0,61*61),ncol=61)
for(a in 1:61)
{
for(b in 1:61)
{
if(matrix[a,b]!=0)
{
temp<-vector[a,2:4]==vector[b,2:4]
for(c in 1:3)
{
if(temp[c]==0){
base_pos<-c+1
cval<-c
break
}
}
##synonymous substitution portion
if(matrix[a,b]==1){
outdata[a,b]<-alpha_*freq[vector[b,base_pos],cval]
}
##non-synonymous substitution portion
if(matrix[a,b]==2){
temp_amino<-rep(0,2)
```

```
temp_hyphy<-rep(0,2)
for(bz in 1:ncol(amino_maps_))
{
for(az in 3:nrow(amino_maps_))
{
if(amino_maps_[az,bz]==a)
{
temp_hyphy[1]<-amino_maps_[2,bz]
temp_amino[1]<-amino_maps_[1,bz]}
if(amino_maps_[az,bz]==b)
{
temp_hyphy[2]<-amino_maps_[2,bz]
temp_amino[2]<-amino_maps_[1,bz]}
if(sum(temp_hyphy==c(0,0))==0){break}
}
if(sum(temp_hyphy==c(0,0))==0){break}
}
outdata[a,b]<-amino_vals[amino_bins[temp_amino[1],temp_amino[2]]]*
freq[vector[b,base_pos],cval]
}
}
}
}
return(list(vector=vector,outdata=outdata))
}

####SET INITIAL CONDITIONS#################################################
alpha_<-.95
vector_c<-matrix(rep(1/4,12),ncol=3)
#create the 1x61 vector of codon frequencies from the nucleotide, base frequency
index<-1
vector_temp<-matrix(rep(0,4*64),ncol=4)
vector<-matrix(rep(0,4*61),ncol=4)
Cfreq_vector_temp<-matrix(rep(0,3*61),ncol=3)
Cfreq_vector<-rep(0,61)

for(i in 1:4)
{
for(j in 1:4)
{
for(k in 1:4)
```

```
{
vector_temp[index,]<-c(index,i,j,k)
index<-index+1
}
}
}
index<-1
#map the 64x64 matrix nucleotide positions for a 61x61 model [universal genetic code]
for(z in 1:64)
{
if(z!=49 && z!=51 && z!=57)
{
vector[index,1]<-index
vector[index,2:4]<-vector_temp[z,2:4]
index<-index+1
}
}

Cfreq_vector_temp<-matrix(rep(0,3*61),ncol=3)
codon_c<-rep(0,61)
TAA<-vector_c[4,1]*vector_c[1,2]*vector_c[1,3]
TAG<-vector_c[4,1]*vector_c[1,2]*vector_c[3,3]
TGA<-vector_c[4,1]*vector_c[3,2]*vector_c[1,3]
stop<-1-(TAA+TAG+TGA)
for(z in 1:61)
{
for(zz in 1:ncol(Cfreq_vector_temp))
{
vectorcol<-zz+1
Cfreq_vector_temp[z,zz]<-vector_c[vector[z,vectorcol],zz]
}
codon_c[z]<-(Cfreq_vector_temp[z,1]*Cfreq_vector_temp[z,2]*
Cfreq_vector_temp[z,3])/stop
}

#read in the utility programs for the analyses
all<-scan(file=paste("./utilities/_dNdS_Matrix.txt",sep=""),what= 'character',sep=',')
all_new<-chartr('}',' ',all)
all_new<-chartr('{',' ',all_new)
cat(all_new,file=paste("./utilities/_dNdS_Matrix_temp.txt",sep=""))
data<-scan(file=paste("./utilities/_dNdS_Matrix_temp.txt",sep=""),what=double(0))
```

```
data.all<-matrix(data,ncol=61,byrow=TRUE)

source(paste("./utilities/s_cnt.txt",sep=""))
source(paste("./utilities/n_cnt.txt",sep=""))
source(paste("./utilities/amino_test_cpu.R",sep=""))
aminofreqs<-matrix(scan(file=paste("./utilities/amino_sub_matrix.csv",sep=""),
sep=',',),ncol=20,byrow=TRUE)
Ramino_maps<-matrix(scan(file=paste("./utilities/amino_codon_map.csv",sep=""),sep=",",
what=double(0)),byrow=TRUE,nrow=8)
Ramino_all<-amino_test(20,aminofreqs)
Ramino_REV<-Ramino_all$aminoR
Ramino_bins<-Ramino_all$binR
Ramino_vals<-100*Ramino_all$Rvals
Ramino_prob<-Ramino_all$aminoP

matrixR<-Rmatrix(vector_c,alpha_,data.all,Ramino_vals,Ramino_bins,Ramino_maps)
trueR<-matrixR$outdata
source(paste("./utilities/dNdS_truth.R",sep=""))
truthvals<-dNdS_truth(trueR,data.all)
dN_muse<-truthvals$outdn96
dS_muse<-truthvals$outds96
truthiness<-dN_muse/dS_muse
beta<-dN_muse
alpha<-dS_muse
source(paste("./utilities/ExpS_multirate.R",sep=""))
ES.temp<-ExpS_multirate(codon_c,trueR,1)
ES<-as.numeric(ES.temp)
##########END OF INITIAL CONDITIONS#####################################################

resultloop <- function(big,small,popn)
{
bigreps<-big
smallreps<-small
extension<-".txt"
popsize<-popn
cluster.dim<-196
Columns.cluster<-cluster.dim+2

ga.ave3<-rep(0,bigreps); ga.ave3t<-ga.ave3; ga.ave4<-ga.ave3; ga.ave4t<-ga.ave3;
ga.ave.nodupkey3<-ga.ave3; ga.ave.nodupkey3t<-ga.ave3; ga.ave.nodupkey4<-ga.ave3;
ga.med.nodupkey3<-ga.ave3; ga.med.nodupkey3t<-ga.ave3; ga.med.nodupkey4<-ga.ave3;
```

```
ga.med.nodupkey4t<-ga.ave3;
no.stats<-5
ga.stats<-matrix(rep(0,2*no.stats),ncol=2); ga.stats.temp<-matrix(rep(0,2*bigreps),
ncol=2);
dn.ave3<-ga.ave3; dn.ave3t<-ga.ave3; dn.ave4<-ga.ave3; dn.ave.nodupkey3<-ga.ave3;
dn.ave.nodupkey3t<-ga.ave3; dn.med.nodupkey3<-ga.ave3; dn.med.nodupkey3t<-ga.ave3;
ds.ave3<-ga.ave3; ds.ave3t<-ga.ave3; ds.ave.nodupkey3<-ga.ave3;
ds.ave.nodupkey3t<-ga.ave3; ds.med.nodupkey3<-ga.ave3; ds.med.nodupkey3t<-ga.ave3;

ga.best<-matrix(rep(0,2*bigreps),ncol=2); ga.best.t<-ga.best; ds.best<-ga.best;
ds.best.t<-ga.best; dn.best<-ga.best; dn.best.t<-ga.best;
###read in the data from the data program
#iterate the number of replicates (bigreps)
for(i in 1:bigreps)
{
data.true<-scan(file=paste("./_true_model",i,extension,sep=""),what= 'character',
sep=',')
true_new<-chartr('}',' ',data.true)
true_new<-chartr('{',' ',true_new)
cat(true_new,file=paste("./_true12",extension,sep=""),append=TRUE)
}
temp<-as.numeric(scan(file=paste("./_true12.txt",sep=""),what='character',sep=''))
truth12<-matrix(temp,ncol=15,byrow=TRUE)
#iterate across the clustering sizes (smallreps)
for(ii in 2:smallreps)
{
temp<-as.numeric(scan(file=paste("./_trans",ii,extension,sep=""),what='character',
sep=' '))
trans12<-matrix(as.numeric(temp),ncol=15,byrow=FALSE)
temp<-as.numeric(scan(file=paste("./_nodupkey",ii,extension,sep=""),what='character',
sep=' '))
nodupkey12<-matrix(as.numeric(temp),ncol=15,byrow=FALSE)

val<-1
ga.id<-rep(0,nrow(trans12))
for(i in 1:nrow(trans12))
{
if(i==nrow(trans12)){
ga.id[i]<-val
break
}
```

```
else{
if(trans12[i,1]>trans12[i+1,1] && trans12[i+1,1]==1){
ga.id[i]<-val
val<-val+1
}
else{ga.id[i]<-val}
}
}
trans12<-cbind(trans12,ga.id)
val2<-1
ga.id.nodupkey<-rep(0,nrow(nodupkey12))
for(i in 1:nrow(nodupkey12))
{
if(i==nrow(nodupkey12)){
ga.id.nodupkey[i]<-val2
break
}
else{
if(nodupkey12[i,1]>nodupkey12[i+1,1] && nodupkey12[i+1,1]==1){
ga.id.nodupkey[i]<-val2
val2<-val2+1
}
else{ga.id.nodupkey[i]<-val2}
}
}
nodupkey12<-cbind(nodupkey12,ga.id.nodupkey)
for(z in 1:bigreps)
{
trans12temp<-subset(trans12,trans12[,ncol(trans12)]==z)
nodupkey<-subset(nodupkey12,nodupkey12[,ncol(nodupkey12)]==z)
ga.stats.temp[z,1]<-nrow(trans12temp)
#initialize best_vals for the GA run
best_vals<-matrix(rep(0,15*max(trans12temp[,1])),ncol=15)
index<-1
#record the best model AIC and dNdS estimates from each population
for(i in 1:nrow(trans12temp))
{
if(trans12temp[i,2]==1){
best_vals[index,1]<-trans12temp[i,1] #population index
best_vals[index,2]<-trans12temp[i,2] #probability weight
best_vals[index,3]<-trans12temp[i,7] #AIC
```

```
best_vals[index,4]<-trans12temp[i,3] #estimate1
best_vals[index,5]<-trans12temp[i,4] #estimate2
best_vals[index,6]<-trans12temp[i,6] #dN estimate
best_vals[index,7]<-trans12temp[i,5] #dS estimate
best_vals[index,12]<-trans12temp[i,12] #delta dN/dS
best_vals[index,10]<-trans12temp[i,10] #delta dS
best_vals[index,11]<-trans12temp[i,11] #delta dN
if (index==max(trans12temp[,1])){ break}
else {index<-index+1}
}
}
delta<-matrix(rep(0,(max(trans12temp[,1])+6)*nrow(trans12temp)),
nrow=nrow(trans12temp))
delta[,1:2]<-trans12temp[,1:2] #first 2 columns are population & prob indicators
delta[,3]<-trans12temp[,7] #col records the AIC for each child & each pop for GA run
#set the best model AIC from each population for all children of that population
for(i in 1:max(trans12temp[,1]))
{
temp1<-popsize*(i-1)+1
temp2<-popsize*(i-1)+popsize
delta[temp1:temp2,4]<-best_vals[i,3]
}
#calculate numerator within-population weights
delta[,5]<-exp(-.5*(delta[,3]-delta[,4]))
pop.delta<-matrix(rep(0,2*max(trans12temp[,1])),nrow=max(trans12temp[,1]))
pop.delta[,1]<-seq(1:nrow(pop.delta))
avs.temp<-matrix(rep(0,max(trans12temp[,1])),nrow=max(trans12temp[,1]))
#pull in each pop best model AIC and then calculate the denom for each weight
for(i in 1:max(trans12temp[,1]))
{
temp1<-popsize*(i-1)+1
temp2<-popsize*(i-1)+popsize
delta[temp1:temp2,4]<-best_vals[i,3]
pop.delta[i,2]<-sum(exp(-.5*(delta[temp1:temp2,3]-delta[temp1:temp2,4])))
delta[temp1:temp2,5]<-exp(-.5*(delta[temp1:temp2,3]-delta[temp1:temp2,4]))
#calculate each population-specific weight
delta[temp1:temp2,6]<-exp(-.5*(delta[temp1:temp2,3]-
delta[temp1:temp2,4]))/pop.delta[i,2]
#calculate each population-specific weighted-average
avs.temp[i,1]<-sum(trans12temp[temp1:temp2,3]*delta[temp1:temp2,6])
#calculate moving population weighted-averages
```

```
temp<-(1/i)*delta[1:temp2,6]
delta[1:temp2,6+i]<-(temp*exp(-.5*(delta[1:temp2,3]-
min(delta[1:temp2,3])))/sum(temp*exp(-.5*(delta[1:temp2,3]-min(delta[1:temp2,3]))))
}
#record ea dNdS est (trans12[,3:4]) and weighted est and basic AICc and weight info
pop.spec.weighted.estimates<-delta[,6]*trans12temp[,3]
ds.spec.weighted.estimates<-delta[,6]*trans12temp[,5]
dn.spec.weighted.estimates<-delta[,6]*trans12temp[,6]
delta<-cbind(delta,trans12temp[,3:4],pop.spec.weighted.estimates,trans12temp[,5:6])
#build the overall GA-weighted average
tau<-(1/max(trans12temp[,1]))*delta[,6]
bestAIC<-min(trans12temp[,7])
ga.ave.vals3<-trans12temp[,3]*((tau*exp(-.5*(trans12temp[,7]-bestAIC)))/
(sum(tau*exp(-.5*(trans12temp[,7]-bestAIC)))))
ds.ave.vals3<-trans12temp[,5]*((tau*exp(-.5*(trans12temp[,7]-bestAIC)))/
(sum(tau*exp(-.5*(trans12temp[,7]-bestAIC)))))
dn.ave.vals3<-trans12temp[,6]*((tau*exp(-.5*(trans12temp[,7]-bestAIC)))/
(sum(tau*exp(-.5*(trans12temp[,7]-bestAIC)))))
ga.ave.vals3t<-(67/196)*(trans12temp[,12])*((tau*exp(-.5*(trans12temp[,7]-bestAIC)))/
(sum(tau*exp(-.5*(trans12temp[,7]-bestAIC)))))
ds.ave.vals3t<-(1/67)*trans12temp[,10]*((tau*exp(-.5*(trans12temp[,7]-bestAIC)))/
(sum(tau*exp(-.5*(trans12temp[,7]-bestAIC)))))
dn.ave.vals3t<-(1/196)*trans12temp[,11]*((tau*exp(-.5*(trans12temp[,7]-bestAIC)))/
(sum(tau*exp(-.5*(trans12temp[,7]-bestAIC)))))

ga.ave3[z]<-sum(ga.ave.vals3)
ds.ave3[z]<-sum(ds.ave.vals3)
dn.ave3[z]<-sum(dn.ave.vals3)
ga.ave3t[z]<-sum(ga.ave.vals3t)
ds.ave3t[z]<-sum(ds.ave.vals3t)
dn.ave3t[z]<-sum(dn.ave.vals3t)

#build the overall simple/nodupkeyd GA-weighted average
ga.stats.temp[z,2]<-nrow(nodupkey)
num.ave<-exp(-0.5*(nodupkey[,7]-min(nodupkey[,7])))
sum.ave<-sum(num.ave)
weights<-num.ave/sum.ave
weights.ave.nodupkey<-cbind(sum.ave,weights,weights)
nodupkey1<-cbind(nodupkey,weights.ave.nodupkey)
ga.ave.nodupkey3[z]<-sum(nodupkey1[,3]*nodupkey1[,ncol(nodupkey1)])
ds.ave.nodupkey3[z]<-sum(nodupkey1[,5]*nodupkey1[,ncol(nodupkey1)])
```

```
dn.ave.nodupkey3[z]<-sum(nodupkey1[,6]*nodupkey1[,ncol(nodupkey1)])
ga.ave.nodupkey3t[z]<-sum(nodupkey1[,10]*nodupkey1[,ncol(nodupkey1)])
ds.ave.nodupkey3t[z]<-sum(nodupkey1[,11]*nodupkey1[,ncol(nodupkey1)])
dn.ave.nodupkey3t[z]<-sum(nodupkey1[,12]*nodupkey1[,ncol(nodupkey1)])

#build the overall simple/nodupkeyd upper median GA-weighted average
nodupkey.med<-matrix(rep(0,ncol(nodupkey)),nrow=1)
for(i in 1:nrow(nodupkey))
{
if(nodupkey[i,7]>=median(nodupkey[,7]))
{
if(nodupkey.med[1,1]==0){
nodupkey.med[1,]<-nodupkey[i,]
}
else {
nodupkey.med<-rbind(nodupkey.med,nodupkey[i,])
}
}
}
num.med<-exp(-0.5*(nodupkey.med[,7]-min(nodupkey.med[,7])))
sum.med<-sum(num.med)
weights<-num.med/sum.med
ga.med.nodupkey3[z]<-sum(nodupkey.med[,3]*weights)
ds.med.nodupkey3[z]<-sum(nodupkey.med[,5]*weights)
dn.med.nodupkey3[z]<-sum(nodupkey.med[,6]*weights)
ga.med.nodupkey3t[z]<-sum(nodupkey.med[,12]*weights)
ds.med.nodupkey3t[z]<-sum(nodupkey.med[,10]*weights)
dn.med.nodupkey3t[z]<-sum(nodupkey.med[,11]*weights)

#build the ga best model values
for(i in 1:nrow(best_vals))
{
if(best_vals[i,3]==min(best_vals[,3])){
ga.best[z,1]<-best_vals[i,4]
ga.best[z,2]<-best_vals[i,5]
dn.best[z,1]<-best_vals[i,6]
ds.best[z,1]<-best_vals[i,7]
ga.best.t[z,1]<-best_vals[i,12]
ga.best.t[z,2]<-best_vals[i,5]
ds.best.t[z,1]<-best_vals[i,10]
dn.best.t[z,2]<-best_vals[i,11]
```

```
break
}
}
print(z)
]
#build results matricies
ests3<-cbind(seq(1,bigreps),ga.best[,1],ga.ave3,ga.ave.nodupkey3,ga.med.nodupkey3,
truth12[,7])
dn.ests3<-cbind(seq(1,bigreps),dn.best[,1],dn.ave3,dn.ave.nodupkey3,dn.med.nodupkey3,
truth12[,6])
ds.ests3<-cbind(seq(1,bigreps),ds.best[,1],ds.ave3,ds.ave.nodupkey3,ds.med.nodupkey3,
truth12[,4])

ests3t<-cbind(seq(1,bigreps),ga.best.t[,1],ga.ave3t,ga.ave.nodupkey3t,
ga.med.nodupkey3t,truth12[,7])
dn.ests3t<-cbind(seq(1,bigreps),dn.best.t[,1],dn.ave3t,dn.ave.nodupkey3t,
dn.med.nodupkey3t,truth12[,6])
ds.ests3t<-cbind(seq(1,bigreps),ds.best.t[,1],ds.ave3t,ds.ave.nodupkey3t,
ds.med.nodupkey3t,truth12[,4])

#create the results labels
axis.label<-c('Rep','Winner','Dup Ave','Ave','Median','MG94')
colnames(ests3)<-axis.label;colnames(dn.ests3)<-axis.label;
colnames(ds.ests3)<-axis.label;
colnames(ests3t)<-axis.label;colnames(dn.ests3t)<-axis.label;
colnames(ds.ests3t)<-axis.label;

#plot results
if(truthiness!=0){
title.val0<-paste('dN/dS Estimates\n',ii,' Non-Synonomous Clusters\nE(S) = ',
round(ES,2),',dN/dS=',round(truthiness,5))
title.val1<-paste('dN Estimates\n',ii,' Non-Synonomous Clusters\n E(S) = ',
round(ES,2),', dN = ',round(beta,5))
title.val2<-paste('dS Estimates\n',ii,' Non-Synonomous Clusters\n E(S) = ',
round(ES,2),', dS = ',round(alpha,5))
}
if(truthiness==0){
title.val0<-paste('dN/dS Estimates by GA Population \n Beta/Alpha is Unknown')
title.val1<-paste('dN Estimates\n E(S) and Beta is Unknown')
title.val2<-paste('dN Estimates\n E(S) and Alpha is Unknown')
}
```

```
title.val3<-paste(title.val0,sep="")
title.val13<-paste(title.val1,sep="")
title.val23<-paste(title.val2,sep="")
source(file=paste("./utilities/EdNdS.R",sep=""))
source(file=paste("./utilities/ExpS_multirate.R",sep=""))

plot.est<-function(b,c,d,dataest1,dataest2,dataest3,dataest1t,dataest2t,dataest3t)
{
if(truthiness !=0){
ylimll<-min(truthiness,dataest1[,2:ncol(dataest1)],dataest1t[,2:ncol(dataest1)],
truth12[,7],na.rm=TRUE)
ylimul<-max(truthiness,dataest1[,2:ncol(dataest1)],dataest1t[,2:ncol(dataest1)],
truth12[,7],na.rm=TRUE)
}
if(truthiness ==0) {
ylimll<-min(dataest1[,2:ncol(dataest1)],dataest1t[,2:ncol(dataest1t)],truth12[,7])
ylimul<-max(dataest1[,2:ncol(dataest1)],dataest1t[,2:ncol(dataest1t)],truth12[,7])
}
#boxplot of dN/dS estimates from Muse '96
boxplot(dataest1[,2:ncol(dataest1)],ylab='Estimator',xlab="Muse '96 dN/dS Estimate",
ylim=c(ylimll,ylimul))
abline(a=truthiness,b=0,lty=1,col='black')
#boxplot of the theoretical delta dN/dS estimates from Muse '96
boxplot(dataest1t[,2:ncol(dataest1t)],ylab='Estimator',
xlab="Muse '96 Delta dN/dS Estimate",ylim=c(ylimll,ylimul))
abline(a=truthiness,b=0,lty=1,col='black')

##########dN plot
if(truthiness !=0){
ylimll<-min(beta,dataest2[,2:ncol(dataest2)],dataest2t[,2:ncol(dataest2t)],
truth12[,6],na.rm=TRUE)
ylimul<-max(beta,dataest2[,2:ncol(dataest2)],dataest2t[,2:ncol(dataest2t)],
truth12[,6],na.rm=TRUE)
}
if(truthiness ==0){
ylimll<-min(dataest2[,2:ncol(dataest2)],truth12[,6],na.rm=TRUE)
ylimul<-max(dataest2[,2:ncol(dataest2)],truth12[,6],na.rm=TRUE)
}
boxplot(dataest2[,2:ncol(dataest2)],main=c,ylab='Estimator',
xlab="Muse '96 dN Estimate",ylim=c(ylimll,ylimul))
abline(a=beta,b=0,lty=1,col='black')
```

```
boxplot(dataest2t[,2:ncol(dataest2t)],main=c,ylab='Estimator',
xlab="Muse '96 Delta dN Estimate",ylim=c(ylimll,ylimul))
abline(a=beta,b=0,lty=1,col='black')

#########dS plot
if(truthiness !=0){
ylimll<-min(alpha,dataest3[,2:ncol(dataest3)],dataest3t[,2:ncol(dataest3t)],
truth12[,4],na.rm=TRUE)
ylimul<-max(alpha,dataest3[,2:ncol(dataest3)],dataest3t[,2:ncol(dataest3t)],
truth12[,4],na.rm=TRUE)
}
if(truthiness ==0){
ylimll<-min(dataest3[,2:ncol(dataest3)],dataest3t[,2:ncol(dataest3t)],truth12[,4])
ylimul<-max(dataest3[,2:ncol(dataest3)],dataest3t[,2:ncol(dataest3t)],truth12[,4])
}
boxplot(dataest3[,2:ncol(dataest3)],main=d,ylab='Estimator',
xlab="Muse '96 dS Estimate",ylim=c(ylimll,ylimul))
abline(a=alpha,b=0,lty=1,col='black')
boxplot(dataest3t[,2:ncol(dataest3t)],main=d,ylab='Estimator',
xlab="Muse '96 Delta dS Estimate",ylim=c(ylimll,ylimul))
abline(a=alpha,b=0,lty=1,col='black')

dev.off()
}
outfile<-"./_GA_bigreps_plots"
extensionp<-".pdf"

val<-"_MG94est"
pdf(file=paste(outfile,ii,val,extensionp,sep=""))
plot.est(title.val3,title.val13,title.val23,ests3,dn.ests3,ds.ests3)

#### build descriptive statistics of the GA runs
totpops<-nrow(trans12)
totpops.nodupkey<-nrow(nodupkey)
#ga.stats[,1]<-bigreps
ga.stats[1,]<-round(apply(ga.stats.temp,2,mean),2)
ga.stats[2,]<-round(apply(ga.stats.temp,2,median),2)
ga.stats[3,]<-round(apply(ga.stats.temp,2,sd),2)
ga.stats[4,]<-round(apply(ga.stats.temp,2,min),2)
ga.stats[5,]<-round(apply(ga.stats.temp,2,max),2)
```

```
##put the final estimates in a table and output the html file
library(hwriter)
data.table<-data.frame(ga.stats)
rows<-c("Ave No. Models","Median No. Models","SD of Models","Min Models","Max Models")
cols<-c("Non-Unique","Unique")
rownames(data.table)<-rows
colnames(data.table)<-cols
outfile<-"./_GA_Statistics"
extensiont<-".html"
p=openPage(paste(outfile,ii,extensiont,sep=""))
hwrite('Simulation Statistics on the GA Runs', p)
hwrite(data.table,style='padding:5px', p)
closePage(p)

mean.val<-apply(ests3,2,mean); median.val<-apply(ests3,2,median);
se.val<-apply(ests3,2,sd)/sqrt(popn); sd.val<-apply(ests3,2,sd);
min.val<-apply(ests3,2,min); max.val<-apply(ests3,2,max); q1.val<-rep(0,ncol(ests3));
q3.val<-rep(0,ncol(ests3));
mean.valt<-apply(ests3t,2,mean); median.valt<-apply(ests3t,2,median);
se.valt<-apply(ests3t,2,sd)/sqrt(popn); sd.valt<-apply(ests3t,2,sd);
min.valt<-apply(ests3t,2,min); max.valt<-apply(ests3t,2,max);
q1.valt<-rep(0,ncol(ests3t)); q3.valt<-
rep(0,ncol(ests3t))

for(i in 1:ncol(ests3))
{
q1.val[i]<-quantile(ests3[,i],prob=0.25)
q3.val[i]<-quantile(ests3[,i],prob=0.75)
q1.valt[i]<-quantile(ests3t[,i],prob=0.25)
q3.valt[i]<-quantile(ests3t[,i],prob=0.75)
}
table0<-cbind(paste(round(mean.val,3),' (',round(abs(mean.val-truthiness),3),')',
sep=""),round(sd.val,4),paste(round(max.val,3),' (',
round(abs(max.val-truthiness),3),')',sep=""),paste(round(q3.val,3),' (',
round(abs(q3.val-truthiness),3),')',sep=""),paste(round(median.val,3),' (',
round(abs(median.val-truthiness),3),')',sep=""),paste(round(q1.val,3),' (',
round(abs(q1.val-truthiness),3),')',sep=""),paste(round(min.val,3),' (',
round(abs(min.val-truthiness),3),')',sep=""))
table0t<-cbind(paste(round(mean.valt,3),' (',round(abs(mean.valt-truthiness),3),')',
sep=""),round(sd.valt,4),paste(round(max.valt,3),' (',
round(abs(max.valt-truthiness),3),')',sep=""),paste(round(q3.valt,3),' (',
```

```
round(abs(q3.valt-truthiness),3),')',sep=""),paste(round(median.valt,3),' (',
round(abs(median.valt-truthiness),3),')',sep=""),paste(round(q1.valt,3),' (',
round(abs(q1.valt-truthiness),3),')',sep=""),paste(round(min.valt,3),' (',
round(abs(min.valt-truthiness),3),')',sep=""))

table<-table0[2:nrow(table0),]; tablet<-table0t[2:nrow(table0t),]
tablecolumns<-c('Mean','SD','Max','Q3','Median','Q1','Min')

table.frame<-data.frame(table)
table.framet<-data.frame(tablet)
outfile<-"./_GA_Est_Performance"
colnames(table.frame)<-tablecolumns; colnames(table.framet)<-tablecolumns;
p=openPage(paste(outfile,ii,extensiont,sep=""))
hwrite('GA Runs Estimate Performance: Values (Delta From the Truth)', p)
hwrite(table.frame,style='padding:5px', p)
closePage(p)
outfilet<-"./_GA_Est_Performance_Delta"
p=openPage(paste(outfilet,ii,extensiont,sep=""))
hwrite('GA Runs Estimate Performance: Values (Delta From the Truth)', p)
hwrite(table.framet,style='padding:5px', p)
closePage(p)

#########AIC and BIC
mBIC<-rep(0,nrow(trans12))
mBIC<--2*trans12[,8]+( (2*trans12[,4])*log(trans12[,9]))
AICc<-rep(0,nrow(trans12))
AICc<--2*trans12[,8]+( (2*trans12[,4]*trans12[,9])/(trans12[,9]-trans12[,4]-1))
trans12<-cbind(trans12,mBIC,AICc)

winners<-trans12[trans12[,2]==1,]
winners_nodup<-winners[!duplicated(winners[,7]),]

pdf(file=paste("./AICc_mBIC",ii,extensionp,sep=""))
plot(trans12[,3],AICc,main='AICc by Clustering Size',xlab='dN/dS Estimate',
ylab='AICc')
abline(v=truthiness,lty=1,col='black')
plot(trans12[,3],mBIC,main='mBIC by Clustering Size',xlab='dN/dS Estimate',
ylab='mBIC')
abline(v=truthiness,lty=1,col='black')
dev.off()
pdf(file=paste("./AICc_mBIC_delta",ii,extensionp,sep=""))
```

```
plot(trans12[,12],AICc,main='AICc by Clustering Size',xlab='dN/dS Estimate',
ylab='AICc')
abline(v=truthiness,lty=1,col='black')
plot(trans12[,12],mBIC,main='mBIC by Clustering Size',xlab='dN/dS Estimate',
ylab='mBIC')
abline(v=truthiness,lty=1,col='black')
dev.off()

pdf(file=paste("./AICc_mBIC_winners",ii,extensionp,sep=""))
plot(winners_nodup[,3],winners_nodup[,ncol(winners_nodup)],
main='AICc Population Winners',xlab='dN/dS Estimate',ylab='AICc')
abline(v=truthiness,lty=1,col='black')
plot(winners_nodup[,3],winners_nodup[,ncol(winners_nodup)-1],
main='mBIC Population Winners',xlab='dN/dS Estimate',ylab='mBIC')
abline(v=truthiness,lty=1,col='black')
pdf(file=paste("./AICc_mBIC_winners_delta",ii,extensionp,sep=""))
plot(winners_nodup[,12],winners_nodup[,ncol(winners_nodup)],
main='AICc Population Winners',xlab='dN/dS Estimate',ylab='AICc')
abline(v=truthiness,lty=1,col='black')
plot(winners_nodup[,12],winners_nodup[,ncol(winners_nodup)-1],
main='mBIC Population Winners',xlab='dN/dS Estimate',ylab='mBIC')
abline(v=truthiness,lty=1,col='black')
dev.off()
}
}
```