3-27-2016

Killing Two Birds with One Stone: The Concurrent Development of the Novel Alignment Free Tree Building Method, Scrawkov-Phy, and the Extensible Phyloinformatics Utility, EMU-Phy.

J. Nick Fisk
jnf3769@rit.edu

Killing Two Birds with One Stone: The Concurrent Development of the Novel Alignment Free Tree Building Method, Scrawkov-Phy, and the Extensible Phyloinformatics Utility, EMU-Phy.

❖

J. Nick Fisk

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Bioinformatics

Thomas H. Gosnell School of Life Sciences
College of Science
Rochester Institute of Technology
Rochester, NY
3-27-2016

❖

Abstract:

Many components of phylogenetic inference belong to the most computationally challenging and complex domain of problems. To further escalate the challenge, the genomics revolution has exponentially increased the amount of data available for analysis. This, combined with the foundational nature of phylogenetic analysis, has prompted the development of novel methods for managing and analyzing phylogenomic data, as well as improving or intelligently utilizing current ones. In this study, a novel alignment tree building algorithm using Quasi-Hidden Markov Models (QHMMs), Scrawkov-Phy, is introduced. Additionally, exploratory work in the design and implementation of an extensible phyloinformatics tool, EMU-Phy, is described. Lastly, features of the best-practice tools are inspected and provisionally incorporated into Scrawkov-Phy to evaluate the algorithm's suitability for said features.

This study shows that Scrawkov-Phy, as utilized through EMU-Phy, captures phylogenetic signal and reconstructs reasonable phylogenies without the need for multiple-sequence alignment or high-order statistical models. There are numerous additions to both Scrawkov-Phy and EMU-Phy which would improve their efficacy and the results of the provisional study shows that such additions are compatible.

Table of Contents

Rochester Institute of Technology
Thomas H. Gosnell School of Life Sciences
Bioinformatics Program

To: Head, Thomas H. Gosnell School of Life Sciences

The undersigned state that Jeffrey Fisk, a candidate for the Master of Science degree in Bioinformatics, has submitted his thesis and has satisfactorily defended it.

This completes the requirements for the Master of Science degree in Bioinformatics at Rochester Institute of Technology.

Thesis committee members:

Name	**Date**
Larry J. Buckley, Ph.D. Thesis Advisor	
Michael V. Osier, Ph.D.	
Gregory A. Babbitt, Ph.D.	

Michael V. Osier, Ph.D.
Director of Bioinformatics MS Program

475-4392 (voice)
mvoscl@rit.edu

Acknowledgements:

Firstly, I'd like to thank my father whose sacrifices for his family and his country have enabled my aspirations. His encouragement means the world to me. I also wish to thank the rest of my family for being there for me whenever I stuttered or faltered. I love you all dearly.

I also want to thank Dr. Larry Buckley, my advisor, for his continued support not just in the research presented here, but in guiding my academic success since my arrival to RIT. He has allowed and encouraged to operate with autonomy while always being excited to offer advice whenever I sought it, be it on Pokémon or Phylogeny. I truly appreciate his taking me as a research student despite his numerous other responsibilities.

I want to thank my other committee members, Dr. Michael Osier and Dr. Gregory Babbitt, whose perspectives I have continually synthesized and from whom I have learned a great deal. I respect them immensely and appreciate their continued support.

I wish to thank the GSOLS support staff whose tireless effort made my other responsibilities all too easy, allowing for the swift completion of this project. Thank you Gabby, Michelle, Jenny, and Allison for your hard work. Bioprep4Life!

I would also like to extend a thank you to Drs. Andre Hudson and Gary Skuse, who gave me great advice concerning graduate school during the final stages of the present work. I hope they continue to mentor and guide students for years to come.

I would like to thank my roommates and dear friends for your support throughout the years. Alex, Mike, and Zach, I literally could not have gone to school without you. I know the time for us to finally go our separate ways approaches, but I have faith that we will all find what we want out of life. Buen viaje!

I would like to thank Coaches Scott Stever and Jason Bovenzi for their understanding as I pursued my graduate degree alongside a demanding role on an athletics team. Your and the team's support and encouragement, both on and off the mat, inspired me to continue working to my goals every day and granted me the confidence and ability to do so.

In addition, I'd like to thank my dear friends back in New Mexico. Chase, Sterling, Tyler, Bruno, Elise, Natasha, and many others. Thanks for supporting me in my college career.

I'd also like to extend a special thanks to the Bioinformatics Journal Club at RIT. You all are friends and colleagues who I will never forget. Thank you for your support during troubled times, both personal and professional. And thank you for being a wall off which I bounced ideas for the duration of this project. Especially to Amanda, Mike, Cam, Connor, Alex, and Shariq—you have shown me immeasurable kindness that I will not soon forget.

Introduction:

It is generally agreed by biologists that phylogenetic relationships can be foundational in the pursuit of many other fields of biology. Continued research and understanding of the evolutionary relationships between organisms and genes contributes to advancements in a plethora of biological fields. Drug design, proteomics, comparative genetics, epidemiology, immunology, biogeography, developmental biology, and, naturally, evolutionary biology all benefit from analysis of phylogenetic relationships. As a result of the foundational nature of phylogenetic systematics, the field itself has evolved markedly since its conception.

In its early stages, phylogenetics started as a series of hypothesizes founded in thought experiments and philosophy[1–3], such as parsimony[4,5]. Today, it revolves largely around the application of mathematical models to molecular data, such as DNA sequence[6–8]. Indeed, the advent and development of high throughput sequencing techniques, also known as next generation sequencing (NGS) technologies, have completely revolutionized biology and its subdisciplines.[9] While early methodologies relied heavily on morphological character traits[10], evolutionary biology and systematics have been adapted to include molecular data and the use of molecular data is now standard in those fields when such data is available[11]. Use of molecular data, however, requires careful selection of segments of DNA to be considered for the estimation of phylogenetic relationships. As data becomes increasingly abundant, more information is capable of being incorporated into phylogenies. Due to the computational complexity of nearly every process involved in tree-building, phylogenetic methods have had to continually upgrade and change to account for the new sea of molecular data available. Some of the adaptations the field has seen include: the optimization of current approaches, alignment-free algorithms[12] that promise whole genome phylogenies, improved supertree methods, etc. With ever expanding data and workflows available to more diverse set of users, phyloinformtics has emerged as a way to organize, incorporate, and synthesize the vast resources relevant to phylogenetic analysis. While the term isn't used consistently in its definition and scope, a theme consistent to so-called phyloinformatic utilities is the organization of phylogenetically relevant data with emphasis on end-user ease.

Aims:

This thesis project has three distinct aims. The first aim is the construction of an extensible phyloinformatic utility focused on tree-building that contains a number of features that accommodate users of varying computational literacy. These features will be based on the critique of other phyloinformatic utilities, as well as a vignette based survey of beta-testers of the utility itself. The second aim of this thesis project is the design and characterization of a novel alignment-free tree-building algorithm, Scrawkov-Phy, that serves as the default behavior of the phyloinformatic utility. The last aim of this project is to further develop Scrawkov-Phy by working to incorporate features common among the successful and modern phylogenetic analysis tools found in the literature.

Chapter 1

❖

Scrawkov-Phy: A Multi-State Quasi-Hidden Markov (QHMM) Approach for Constructing Alignment Free Gene and Species Phylogenies

J. Nick Fisk

Background:

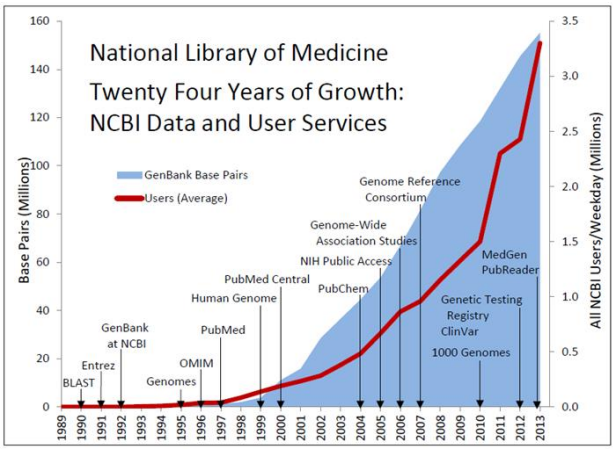
Necessity of Phylogenetic Methods

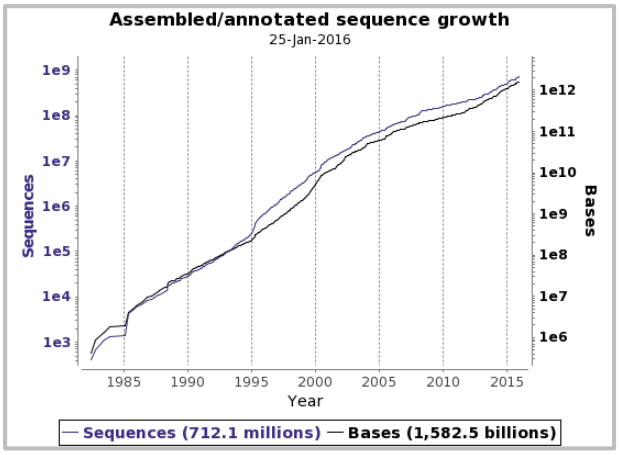
Many DNA sequence datasets are often invested in for reasons other than phylogeny. Indeed, much whole genome scale data is collected with the expectation of application beyond use in phylogenetic analysis. Largely, this is due to the complex and difficult problem with the computational alignment of genomic sequences[13]. Moreover, many methods developed to present day perform phylogenetic reconstruction using only a minuscule fraction (namely the homologous exons) of the genome. Due to these issues, as well as the relative cost, genome scale phylogenies have been difficult to construct and characterize. However, with ever-prominent advancements in high-throughput sequencing technologies, molecular sequence data will undoubtedly become less scarce and projects featuring an ever-increasing number of organisms will more feasibly be able to incorporate such data, further compounding this challenge. Critically, a large number of other comparative biological studies require the construction of a phylogeny as a foundational step. Indeed, phylogenetic relationships represent fundamental hypotheses in many areas of the biological sciences including evolutionary biology. This combination of phylogenetics as a keystone element in biological studies, rapidly expanding sources of molecular data, and the limitations of both computing resources and of the algorithms themselves ensures the continued need for the development and characterization of phylogenetic methodologies.

Mo' data, mo' problems

Big data is a term generally applied across many disciplines. It generally refers to extremely large datasets that, when analyzed carefully, can elucidate non-obvious trends or associations[14]. Regardless of the context in which it is discussed, the analysis of big data presents unique challenges in the sheer volume of often heterogeneous data used in studies thereof[15]. Traditional techniques or classic software used in the field may no longer feasibly be applied. Often, these reasons are computationally founded: There is simply not enough memory or time available to complete the task. Other times, it is logistically difficult to ascertain the types of analysis to perform to produce useful results[16]. It may also be statistically difficult to

determine the significance of a result or how to use the result in a meaningful way (knowledge discovery)[17]. As a result of these difficulties, novel methods are often needed to meet the challenges presented by big data and obtain relevant and meaningful results. DNA sequence data and the availability and analysis thereof undoubtedly falls under the larger umbrella of big data, generally speaking.[18] The availability of data, both locally and through services like the National Center for Biotechnology Information (NCBI) or the European Bioinformatics Institute (EBI), allows even small labs to perform projects for which seminal approaches would be insufficient. Indeed, as seen in Figures 1A, 1B, and 1C (below) the number of sequences in these databases has increased exponentially and the doubling time of the EBI database is roughly linear. Phylogenetics embodies many of these challenges as traditional approaches require algorithms which fall in some of the most problematic classes of computational complexity.





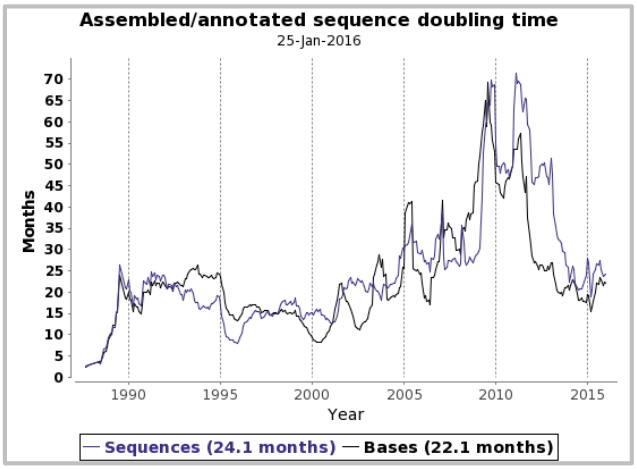


Figure 1: The deposition of sequence data in Genbank over time with historical events as an overlay is shown in Figure 1A (top left)[19]. The deposition of sequences in the EBI's European Nucleotide Archive (ENA) follows a similar pattern[20], shown in a log scale graph in Figure 1B (top right). The ENA doubling time, however, is only weakly increasing at a linear rate and shows indications of being periodic as presented in Figure 1C (bottom). Biological data is becoming exponentially more abundant.

There has been much discussion on how to most effectively use big data, both in science on other disciplines. Generally, it is agreed that such data, especially in the sciences, should not be used without careful consideration and scrubbing of the data[21]. The exploration of big data and all it has to offer, however, is vital to determining its place in science.

Computational Complexity, Computational Size, and Phylogenetics

In computer science, problems and algorithms are often categorized by their complexity by number of computations required to solve them[22]. While there are any number of measures of computational complexity, the discussion of which is beyond the scope of the present, they all typically refer to the runtime of an algorithm as a function of inputs. Best-case, worst-case, and average-performance metrics are often used and, depending on the limiting behavior of their solutions, problems are categorized into different classes, either by time or space[22]. The 'polynomial time' or 'P' class of problems are the set of computational problems formally considered easy as there exist at least one algorithm that completes the decision problem in polynomial time. The 'nondeterministic polynomial' or 'NP' class of problems is a broader class of decision problems that include the 'P' class of problems and allow that an algorithm derive a solution in a non-deterministic manner but still be verified by a deterministic algorithm.[23] The NP-complete class of problems include problems for which there is no proof against a polynomial time solution, but for which none are known. They are defined by their ability to solve or 'complete' all other NP problems in polynomial time if a polynomial time solution exists for it. To rephrase, if any NP-complete problem were to be solved, all NP problems could then be considered P problems, since it would provide a means of achieving polynomial time for every problem in NP. Although there are many types of problems, P, NP, and NP-complete refer only to decision problems. Since this system of classification is semantically simple and widely used, another class of problems was defined to make comparison to the aforementioned classes more straightforward. This class, called NP-hard, are any problems, whether they are decision problems or not, which are at least as hard as any problem in NP, including those that are NP-

complete. Simply put, problems that are categorized as NP-complete or NP-hard are so computationally difficult that efficient and optimal algorithms for solving them are very unlikely to exist. As a result, the size of inputs used in such algorithms is extremely limited, requiring heuristic, or suboptimal, methods to complete in reasonable time or with reasonable computational space.[23] Critically, many algorithms used in phylogenetic analysis fall into the classes of NP-complete or NP-hard.

There is no shortage of literature on the computational complexity of phylogenetic methods[24–27]. William H.E. Day, as well as many others, have shown the complexity of many algorithms used in phylogenetic analysis. Foulds and Graham showed that the Steiner Problem in Phylogeny (SPP) as it pertains to biology is an NP-complete problem[28]. The Steiner tree problem deals with minimizing the total length or weight of junctions between any set of objects. There are specific instances of Steiner-tree problem that have polynomial time solutions due to limitations in the domain in which the problem is being solved. Unfortunately, no such solution is available for phylogenetics and, thus, the SPP is also NP-complete, requiring all but the smallest phylogenies to be resolved using heuristic methods. The problem's classification persists whether parsimony, dissimilarity matrices, or many other methods are used. Thus, suboptimal heuristics, often using dynamic programming, are typically used to combat the extreme combinatorics of the problems.[29] As a result, most phylogenetic tree searches are non-exhaustive. Moreover, most phylogeny recovering algorithms require a multiple sequence alignments to perform. These algorithms are affected both by the number of sequences, as well as the length of those sequences. The precise and optimal way of computing an alignment between sequences has a computational complexity of $O(L^N)$, where N is the number of sequences being aligned and L is the length of the sequences.[29] Naturally, this absurdly exponential problem makes true alignment of sequences prohibitive for all but the smallest of datasets. Progressive multiple sequence alignment, a dynamic heuristic to the problem, reduces the complexity to a more reasonable figure. The development of more refined heuristics has popularized a number of heuristic multiple sequence alignment algorithms as sequencing data has become more abundant. Version 6 of MAFFT[30] has a $O(NL^2)+O(N^3)$ time complexity. MUSCLE[31], generally, has a time complexity of $O(N^2L+NL^2)$, though some parameters will cause it to run with an additional $O(N^3L)$. It is important to note that these are worst case

measures of run time and each program has cases in which they perform much better. For instance, MAFFT complexity drops to $O(NL^2)$ when comparing similar sequences. Despite these advancements in heuristic methods, many consider exponential run time to be insufficient. Indeed, for large numbers of long sequences, such as those on the scale of whole genomes, these methods still fall short of feasible. In addition, while not discussed at length here, memory space utilization also becomes a factor making both RAM resources and runtime considerations as investigators advance projects. While not the sole aim of improving phylogenetic analysis and alignment methodologies, whole genome phylogenies have a number of additional challenges that make traditional methods difficult to employ. For instance, when performing whole genome phylogeny, the genomes must all first be assembled. Then, traditionally, all true orthologous regions need to be identified and ordered before alignment and there are significant regions of unalignable (non-homologous) DNA (i.e. indels). Alignment is then subject to the same considerations previously mentioned, as is tree-building, along with additional considerations about the substitution rate etc. Alternative methodologies, including those which do not require multiple sequence alignment, have continuously been developed to address the challenges of the emerging field of phylogenomics.[32]

Phylogenetics, Phylogenomics, and the Traditional Methods

The main distinction between phylogenetics and phylogenomics is scale. Phylogenomics lays at the union between evolutionary biology and genomic studies. There have been numerous methods developed for performing phylogenetic analysis and, as the field as called for more ways to handle genome-scale data, these methods have improved and evolved themselves to meet the challenge. To fully understand the limitations of the original implementation of these methods, as well as to understand why improvements need be made for phylogenomics applications, first briefly surveying the traditional methods is necessary.

Parsimony approaches are perhaps one of the oldest approaches. It follows the philosophy of Occam's razor in which we shear away all unnecessary steps or events. In the specific instance of phylogeny, it implies that the tree that requires the fewest number of evolutionary events to occur is the most likely to be correct. It is, algorithmically, easy to determine the number of steps in a tree and evaluate a given tree on parsimony (that is, to score

it). However, there is no method for the generation of trees via this criteria, simply evaluation. A tree search must be performed. Exhaustive tree searches are computationally infeasible, so heuristics are often employed. Even when the most parsimonious tree is recovered, it is not guaranteed to be the correct tree, however, and such an approach performs poorly in certain conditions[33]. Typically, it will underestimate branch lengths. Naturally, a shortcoming invites alternative methods, such as the use of distance.

Alternatives to parsimony are the methods which may use distance to algorithmically construct a tree, instead of solely evaluating a tree for its goodness. Typically, distances are obtained using aligned genetic information by measuring dissimilarities. From here, there are a number of algorithms that can reconstruct a tree, such as UPGMA and Neighbor-Joining[34]. Some methods have optimality criteria, which means that they not only allow for the algorithmic construction of trees, but also the evaluation of the goodness of those trees. There are shortcomings in this method, however. As many methods rely on pairwise generation and clustering, information about individual characters is lost during the reduction of character stats to a distance[35]. Statistical methods may address some of these shortcomings.

Maximum likelihood trees are those produced under the statistical technique of the same name. Generally, the algorithm looks at the statistical likelihood of a particular tree, evaluating the probability of the data given the tree[36]. In this way, it is similar to maximum parsimony as it evaluates many trees, choosing the one that is most likely given its substitution model, rather than algorithmically constructing one. It also inherits same problem, where the number of possible trees exponentially increase with increasing numbers of taxa. There are a number of heuristics that are applied to mitigate this issue, a common one being splitting the tree into subtrees and finding the probability in this manner[37]. Bayesian approaches bear many similarities to maximum likelihood, as they both operate on statistical principles. However, Bayesian approaches assume a probability distribution and then use posterior probabilities to evaluate the tree as its optimality criteria, usually via simulation[38]. Bayesian approaches tend to be robust against simple stochastic variation, but, since they assume a model a priori, are a point of contention in the field[39].

Alignment Free Methods:

As the name suggests, alignment free methodologies in phylogeny are techniques that can produce trees without the need to perform multiple sequence alignment. Such techniques are based on any number of statistical, computational, and biological principles. K-mer frequencies are often incorporated into profiles to measure informational entropy or feature frequency, which allows for statistical inference. While exact implementation of k-mer methods are variable, many of them incorporate Markov models to account for noise unrelated to the evolution of the region of interest. Such models allow for a numeric probability of the result to be obtained which has utility in evaluating the efficacy of the approach as well as individually generated trees[40]. Other general approaches correlate the information of regions of the sequence, measures string distance (such as Levenstein or Lempel-Ziv compress distances), or weighed graph searches. Regardless of the algorithm used, there are limitations to alignment free methods and critics of the approach often cite a lack of obvious or direct biological meaning underlying the method, beyond the nebulous claim of "information content". Indeed, one of the strongest arguments for the traditional methods that utilize multiple sequence alignment is that it allows for the sufficiently reasonable conclusion that aligned sites (e.g. SNPs, loci, etc.) are orthologous in nature, meaning that the sites have a shared evolutionary history. While convergent evolution and other common plagues of phylogenetic analysis may obfuscate this evolutionary history, there are likelihood and other statistical methods traditionally used that characterize this risk and the results of these methods can be more straightforward in their interpretation (due in part to their seminal nature) compared to newer, less robustly tested alignment free methods. As one might expect, the traditional methods and alignment free methods each have their own respective strengths and weaknesses. Chan and Ragan[32] performed an extensive comparative analysis of alignment based and alignment free methods. They contrasted generalizations of the two approaches verbosely. Alignment free methods do not and cannot assume any sort of contiguity of homologous regions, whereas this is the basis of alignment based methods. Alignment based methods are presently well characterized and their limitations fairly well described in the literature, whereas the limits of alignment free methods are still being discovered and debated. The pairwise comparison of whole sequences in the alignment based methods can increase runtime, whereas the alignment free methods often use subsequences which improve runtime at the cost of memory. Generally,

alignment free methods are less dependent of evolutionary models and more robust to biological stochastic effects. The huerisitc nature of MSA means that alignment scores alone are not particularly telling of homology, whereas the exact and deterministic methods often applied in alignment free methods are more directly comparable. In essence, alignment based methods are well characterized and come with a guarantee of homologous site comparison at the cost of efficiency. Alignment free methods are less described in biological context, but are resilient to some pitfalls in alignment based methods, especially that of runtime. Neither outright guarantee the correct tree, so the exploration of both methods is justified.

It is apparent that alignment free methods offer solutions to many qualms in phylogenetics, both biologically and computationally. The runtimes for these methods are almost universally better and often scale better than the traditional methods. The alignment free methods offer solutions to heterogeneous data that emerges as sequences approach whole genome size caused by chromosomal rearrangements and duplications[32]. These methods are also more resilient to randomness in the observed mutations (i.e. stochastic variations). Alignment is oft inaccurate for distantly related sequences, which introduces error before the tree search or construction even begins[41,42]. Moreover, the alignment-free methods tend to rely less on the assumption and parameterization of a molecular clock, which is a subject of active debate in the literature. Despite these strengths, the abstraction of the model from the biological system in conjunction with the ill-explored limits of these methods make further research into alignment free methods needed before widespread application is advisable. However the potential of alignment-free methods to improve throughput, solve long standing biological questions, and more fully utilize the big data emerging in biology make investigation into and development of such methods of value, even inspiring 'Alignathons'[43]. Such was the motivation behind the development of Scrawkov-Phy.

Gene Trees, Species Trees, and Incomplete Lineage Sorting:

Simply stated, gene trees are phylogenetic representations of how homologous (most often orthologous) genes have evolved over time. Interestingly, it is not an uncommon fallacy for gene trees to be misinterpreted as being entirely representative of the evolution of a species. In

reality, gene trees represent just that—the inferred evolutionary relationship among particular genetic segments. In reality, the evolution of the species can be distinct from the evolution of any one particular gene, though obviously related. This view of species evolution has the underlying and widely accepted assumption that there are different evolutionary forces and, thus, history for different genes shared among species. The rate of mutation has been shown to vary throughout the genomes of taxa from all over the tree of life, likely due to the differing selective pressures on these regions. Other events, such as population bottlenecks or gene duplication, can influence the evolution of genes with a degree of independence from each other. Additionally, stochastic effects, which are well described in populations and genomes, contribute to the differencing evolutionary histories of genes within a clade. The construction of species trees, then, focuses on intelligently utilizing constituent gene data to infer a phylogeny of the species of interest. Naturally, there are cases where the true species tree is not represented in any or all of the constituent gene trees. One reason for this situation is incomplete lineage sorting.

If one considers the tracing of the genealogy of multiple genes back to a common ancestor, incomplete lineage sorting can be thought of as the failure of these genes to merge within the species (before the speciation event during traceback, which is afterwards temporally). This results in one or more discordant gene trees, which confounds phylogenetic inference. While there are many other problems in phylogeny, such as homoplasy[44], incomplete lineage sorting is an excellent example of an issue that both plagues and drives the search for robust methods for the construction of species trees.

One approach for the construction species trees from constituent gene trees commonly used today is the use of the coalescent. The coalescent is a time reversible statistical theory used to model stochastic processes. Historically, it has been used in population genetics to model the stochastic processes involved in evolutionary drift[45]. However, it has also been applied to the creation of species trees via the assumption of an underlying gene genealogy in molecular data[46,47]. This is the underlying philosophy in the BEAST[48] suite of algorithms, which avoids MSA by coalescing trees into a single topology. Alternatively, supertree[49] and supermatrix[50] (superalignment), and simultaneous-analysis[51] methods will also yield species trees by combining data, whether it be at the tree or score level[52]. For exceptionally large trees, concatenating methods are considerably more feasible than the coalescent[53]. Of course, while the

aforementioned are typical and well described methods, there exist a plethora of alternative methods for constructing species trees from gene trees, including one implemented in Scrawkov-Phy.

Principles behind Scrawkov-Phy:

Scrawkov-Phy, presented here, is a novel alignment free method of generating both gene and species trees. The spiritual successor of the original Scrawkov, which attempts a Hidden Markov Model approach to taxonomic identification of species from cytochrome b sequences with principles similar to the popular program HMMER[54], Scrawkov-Phy is named because the data used in early development was exclusively that of birds. The aim of Scrawkov-Phy is to take many of the aforementioned advantages offered by alignment free algorithms while keeping the reasoning more firmly rooted in tangible biological principles and phenomenon. The principles of the components of the algorithm are described below, while a more specific iteration through the algorithm can be found in the Methods section.

QHMMs, Earthquakes, and Phylogeny: Markov models are a class of timeless statistical models that deal in stochastic processes[55,56]. There are different flavors of Markov models depending on the observability of the state, as well as autonomy of the system being modelled[57]. Use of Markov models has become ubiquitous[58–65]. Common applications of Markov models scale all scientific disciplines and have great utility in modelling human decision making. In the biological sciences, the range of utility of Markov models run the gambit from Markov Chain Monte Carlo (MCMC) simulations in population genetics[66] to the seminal gene finding algorithms[67]. When a Markov model has states that are at least partially hidden (i.e. not directly observable), it is classically designated as a special case of the Markov model called a Hidden Markov Model (HMM). Application of Bayes' rule to Markov models falling into this definition allows for the determination of the probability of any sequence of hidden states given observable states. Traditionally, HMMs require the determination of transition probabilities, both hidden and observable, to be measured or calculated by empirical means. Except for a few experimental conditions, phylogenetic inference is not directly observable due to evolution working over deep time. While there are alignment free phylogenetic methods that utilize traditional HMMs,

Scrawkov-Phy, in the interest of more wholly adhering to the biology, does not use HMM in the traditional definition[68]. Instead, it draws inspiration from Wu's work on the cluster analysis of earthquake catalogs.[69] In the aforementioned work, quasi-hidden Markov models (QHMMs) are described and utilized to characterize, in terms of the mother-and-kids model and the domino model, the propagation and clustering of aftershocks (children) as a function of the first shock (mother). While the comparison of the problems of earthquake propagation and phylogenetic inference may not immediately seem logical, the connection becomes clearer when the temporal limitations, non-independence, and clustering aspects of each problem are considered. A challenge in earthquake predictions is that the main shock is defined by its intensity, not by its temporal occurrence (with foreshocks occurring before and aftershocks occurring after). That is, the main shock is defined by the magnitude of other shocks, which are not wholly independent of each other due to a common origin or neighborhood. As previously mentioned, construction of a phylogeny is inference due to the general inability to observe evolution occur. The evolution of each taxa is not wholly independent of each other and the branch lengths derived in a phylogeny are dependent on the other taxa being considered. Lastly, as tree building is a specific case of clustering, the two problems are somewhat semantically related. Minimally, the application of a QHMM using an iteratively constructed training set in a way inspired by this work merits exploration and may more closely represent the underlying biology than some other alignment free methodologies.

GC Content: One of the measures that Scrawkov-Phy uses to construct phylogeny is GC content. GC content is simply the measure of the amount of C or G base pairs in a given region of DNA. It has been shown to differ significantly, even among closely related taxa.[70] The creation or degradation of isochores have been shown to evolutionarily driven and have been correlated with taxa life history and cytology[71]. It has been suggested that GC content played a role in the evolution of biota during large scale ecological changes.[72] As GC content is a feature of the genome, and all subsets thereof, different areas of the genome may have evolved different GC contents at different points temporally. While the tendency of GC content to vary significantly makes its use in capturing phylogenetic signal difficult, there is evidence to suggest its utility in small genomes, especially that of bacteria[73]. Due to its ease of measurement and this potential evolutionary significance, GC content was incorporated into Scrawkov-Phy. It is

important to note that for particular ranges of taxonomy (namely, higher level organisms) this measure's usefulness is limited. However, in these instances, the weight of GC-content will be minimalized by application of a hill-climbing algorithm.

Di/Tri-nucleotide Frequency: The frequency profiles describing which di/tri-nucleotides occur is a measure often used even in traditional alignment-based methods such as MEGA. Indeed, there has been work on a broad number of orgasms using not only di- and tri-nucleotide frequency, but also with oligomers of lengths upwards of 10 bases.[74] Moreover, these k-mer frequency methods, while limited in the amount of signal they provide, retain that signal in differing statistical models. The resilience of information contained in di- and tri-nucleotide frequencies to a range of statistical models, their ease of computation, and precedent in the literature, as well as well-regarded programs, make this feature a prime candidate for inclusion in Scrawkov-Phy.

Codon Bias: Codon usage bias refers to the preferential usage of a particular codon among synonymous codons within coding DNA. Codon bias has been observed in taxa at all levels of life. Additionally, codon bias profiles vary even among more closely related OTUs and have been shown to be under, minimally, weak selective pressure. The third nucleotide position, or the wobble position, is the source of much of the redundancy in the genetic code. The conservation of the use of particular codons is explained by a number of biological systems. Synonymous codons have been shown to affect both DNA and RNA polymer structure and flexibility.[75] Thus, maintenance of the macromolecular properties, rather than simply the protein coding information[76], are supported by selection over deep evolutionary times[77]. mRNA secondary structure is influenced by the composition of the codons and third position nucleotide distribution largely impacts this even over shorter periods of evolutionary time, as evidenced in the Drosophilids.[78] These functional constraints are observed among many groups of organisms, both prokaryotes and eukaryotes. Due to the prevalence of codon bias, its persistence even in deep evolutionary time[79], and its subjugation to evolutionary forces[79,80], codon usage bias, as a measure, has the biological context and ease of computation necessary for incorporation into an alignment free phylogeny algorithm and was included in Scrawkov-Phy.

Trinucleotide Transition Probabilities: Historically, the probability of observing a codon given a preceding codon is widely used in gene finding programs. Indeed, many courses that

teach bioinformatics algorithms or genomics tend to use this seminal approach as an exercise. However, even modern gene finders incorporate codon transition probabilities in their algorithm. For instance, GENSCAN[81] utilizes a 'three-periodic' 5th order Markov model of coding regions to build their model for predicting eukaryotic genes. For GENSCAN, the authors shied away from the use of the term 'codon', and instead referred to them in term of their frame (i.e. three periodic, as aforementioned). Likewise, the term trinucleotide will be used here to mean sets of three nucleotides without the connotations associated with codons as Scrawkov-Phy does not assume that all sequences used with it contain only exons. Triplets of nucleotides, the basis of codons, are fundamental to information flow as per the Central Dogma. However, they may also contain structural information in the form of DNA and RNA structure, stability, and flexibility through a summation of local biophysical properties. As such, regardless of if the sequence is purely exon in nature, the incorporation of trinucleotide transition information into the algorithmic determination of phylogeny may be of merit and has basis in both the literature and biology.

Dinucleotide Transition Probabilities: Much like the trinucleotide case described above, dinucleotide transition probabilities likely guard structural information as the result of aggregate local effects. Evidence of this is shown in work showing that structural RNAs have lower folding energy than random RNAs, even if the dinucleotide composition is the same. The implications of this finding are, minimally, two-fold. Firstly, in the context of information theory, entropy is roughly related to the amount of information derivable from the probability distribution of a series of events. The lower-folding energetics of the non-random RNAs is indicative that a signal containing information may be present due to the non-random distribution of dinucleotides. Secondly, due to the lower folding energy and biological essentialness of structural RNAs, it is likely that the system is under selective pressure and that dinucleotide frequencies may provide an evolutionary signal from which a phylogeny may be constructed. Chaos game representations of DNA sequences often have patterns that are explainable in terms of dinucleotides and trinucleotides[82], which may support the presence of a useable and biologically relevant signal. While the exact nature of the biological significance is not fully described in both DNA and RNA contexts, there are biophysical works that support the claim that dinucleotide distribution is functionally conserved[83] and, thus, a measure of this was

incorporated into Scrawkov-Phy. Notably, single nucleotide transition probabilities and distributions were excluded from the algorithmic construction of phylogeny as they are historically noisier and contain less local information than either dinucleotide or trinucleotide features.

Hill-climbing algorithms: Due to the need to have many genomic features incorporated into Scrawkov-Phy, a way to weigh the different features in terms of importance was needed. A rough hill climbing algorithm was employed on a well resolved tree to tune the parameters to reconstruct said tree, acting much like a seed tree in other popular methods. As a beneficial side effect, the non-QHMM features need not be converted to log space to match the QHMM features. Hill climbing algorithms, fundamentally, score a population of solutions to a problem and change the parameters of the solutions to maximize the output of the scoring function. As a rudimentary form of machine learning, hill-climbing algorithms have a variety of strengths and weaknesses. As heuristics, they are not guaranteed to find the best solution and may fall into local maxima solutions to a problem. However, as heuristics, they are also more likely to finish in reasonable computational time. Ultimately, a hill-climbing approach was chosen to weigh parameters in Scrawkov-Phy.

A Greedy Approach to Species Trees: As previously discussed, the computational construction of species trees from molecular data is a complex problem with no clear solution. All methods currently have their merits and pitfalls. Perhaps somewhat oversimplified, the coalescent can be difficult to compute, especially with limited statistical expertise, and supertrees lack the mathematical rigor provided by the coalescent or similar methods[84]. Due to Scrawkov-Phy's aim as an alignment free method is to reduce runtime, the coalescent was not used in interest of improving runtime. Likewise, a concatenating method was not used. Instead, Scawkov-Phy favors an alternative heuristic that scales all the genes to the minimum and maximum semi-log scores used as a surrogate distance and obtains the one with the maximum probability by means of selecting locally minimum scores. The result is, in essence, a greedy algorithm for constructing species trees from gene trees.

Methods:

Components of Scrawkov-Phy

Scrawkov-Phy was written entirely in the Java language and requires Java 7 or later to run properly. Scrawkov-Phy is composed of three classes, described in Table 1, below. The source code, as well as Javadoc, are available under a GNU GPL v.3 license at https://github.com/nickjfisk/Scrawkov_PHY, by email at nick.j.fisk@gmail.com, or in the supplemental figure section of the present.

Table 1: The Java classes of Scrawkov-Phy

Java Class	Description	Used in which process
Bird	The Bird class is the programmatic representation of a single taxa to be included in the phylogenetic tree. The class stores information about the DNA sequence and taxa from which it was derived and defines methods that act on the sequence in the algorithmic determination of phylogeny.	Scoring of parameters used to create the composite index from which the tree is constructed
Node	Used in tree building after all scoring, normalizing, and composition aspects completed	Tree-building from composite index
Scrawkov-Phy (main)	Contains the main functional elements of the program. It creates the main data structures that keep all the information about and contained in instances of the Bird class organized. It calls on methods defined in both Node and Bird to construct a Newick formatted phylogenetic tree.	Integration, normalization, and maximizing of the scores. Construction of main data structures General program flow
Fisk 2016		

Algorithmic Composition of Scrawkov-Phy:

Algorithmically, Scrawkov-Phy is fairly simple and straightforward. The algorithm is detailed in text in the subsequent text and visually in Figure 2. First, the base algorithm is described, though it can be adjusted with command line parameters.

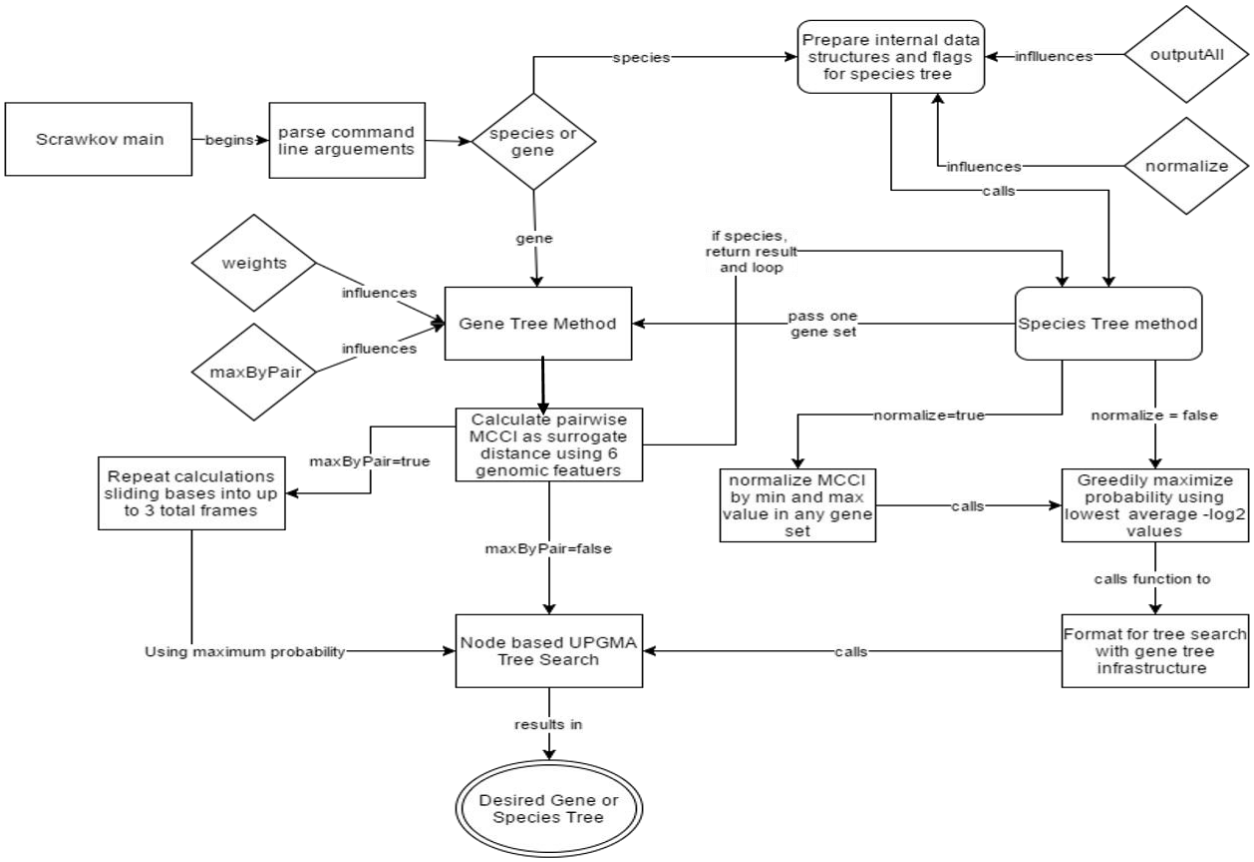


Figure 2: Schematic drawing of the programmatic flow of Scrawkov-Phy. Notably, aspects of code are reused between species and gene trees, including the final generation of the Newick formatted tree. Diamonds represent command line arguments that alter the flow of execution or the calculation of particular elements. Rectangles represent methods called to perform particular functions.

Calculation of GC Content: GC content of each sequence was scored by simply obtaining the proportion of the bases that were either Guanine or Cytosine.

Calculation of Trinucleotide Transition Frequencies: Trinucleotide transition frequencies were measured by crawling down the DNA sequence in three nucleotide partitions sliding three positions, similar to how codons are interpreted by cellular machinery. At each partition, the bases in the upcoming partition are observed and recorded relative to the times they were preceded by the current partition. The partition then slides. At the end of the crawl, each count is divided by the total number of partitions observed to obtain a frequency

Calculation of Dinucleotide Transition Frequencies: Calculation of the dinucleotide transition probabilities were performed in the same manner as the trinucleotide frequencies, but in partitions of two, rather than three.

Measurement of Codon Bias: Codon bias was measured by crawling down the DNA sequence in partitions of three nucleotides and translating the trinucleotide sequence into the appropriate amino acid representation. For each amino acid translated, a count of the codon of origin was incremented. The frequency at which a particular codon was used for a particular amino acid was obtained by dividing the count of the times that codon was used by the total number of times that amino acid was encoded.

Pairwise Comparison of Taxa GC Content: The pairwise score for GC content is simply the absolute value of the difference in proportion between a pair of sequences.

Pairwise Comparison of Taxa Trinucleotide Transition Probability: The observed trinucleotide frequencies are incorporated into a second order QHMM with an iteratively constructed training set, as described in the discussion of QHMMs above. The log-base-two probabilities of the emission of a sequence given the assumption that the two sequences being compared share the most common ancestor is determined and treated as the score for this feature.

Pairwise Comparison of Taxa Dinucleotide Transition Probability: The comparison of dinucleotide transition probability is performed in the same manner as the trinucleotide case described above. Notably, however, a separate QHMM is used for each.

Pairwise Comparison of Taxa Codon Bias: The comparison of Codon Bias is achieved by the summation of the absolute value of the difference of proportions of codon usage frequency across all possible amino acids.

Maximization of Scores from Initial Reading Frame: Notably, alignment free methods suffer from not knowing whether two sequences begin in the reading frame or stay in the same reading frame. For the present algorithm, this is a particularly detrimental limitation of the approach. Thus, to account for this, the algorithm runs its feature search several times with the sequences being assumed in different frames and the frame resulting best (here assumed to be

most parsimonious) is saved and used in the next steps of the algorithm. Under this assumption, the result is the assurance of the maximum, if unknown, orthologous sites being comparted and that two sequences falling out of the same reading frame still provides signal on which the algorithm operates.

Obtaining the Markov Chain Composite Index: The surrogate distance metric used in Scrawkov-Phy is a composite score in which all measures are weighted and then incorporated into a single score, here called the Markov Chain Composite Index (MCCI). The calculation of the score is simply the summation of all the feature scores after the application of the weight.

Obtaining the Parameter Weights: Weights for the parameters were obtained via a simple hill-climbing algorithm. Sequences of a simple and well-defined training were evaluated for their MCCI. The parameters were then weighed with random numbers and a tree constructed. The tree was checked for correct topology. If the topology of the tree was correct, then the weights were kept. After ten correct tree topologies were obtained, the weights were averaged to obtain the value in the current model. The tree used in the hill climbing algorithm is shown in Figure 3 below.

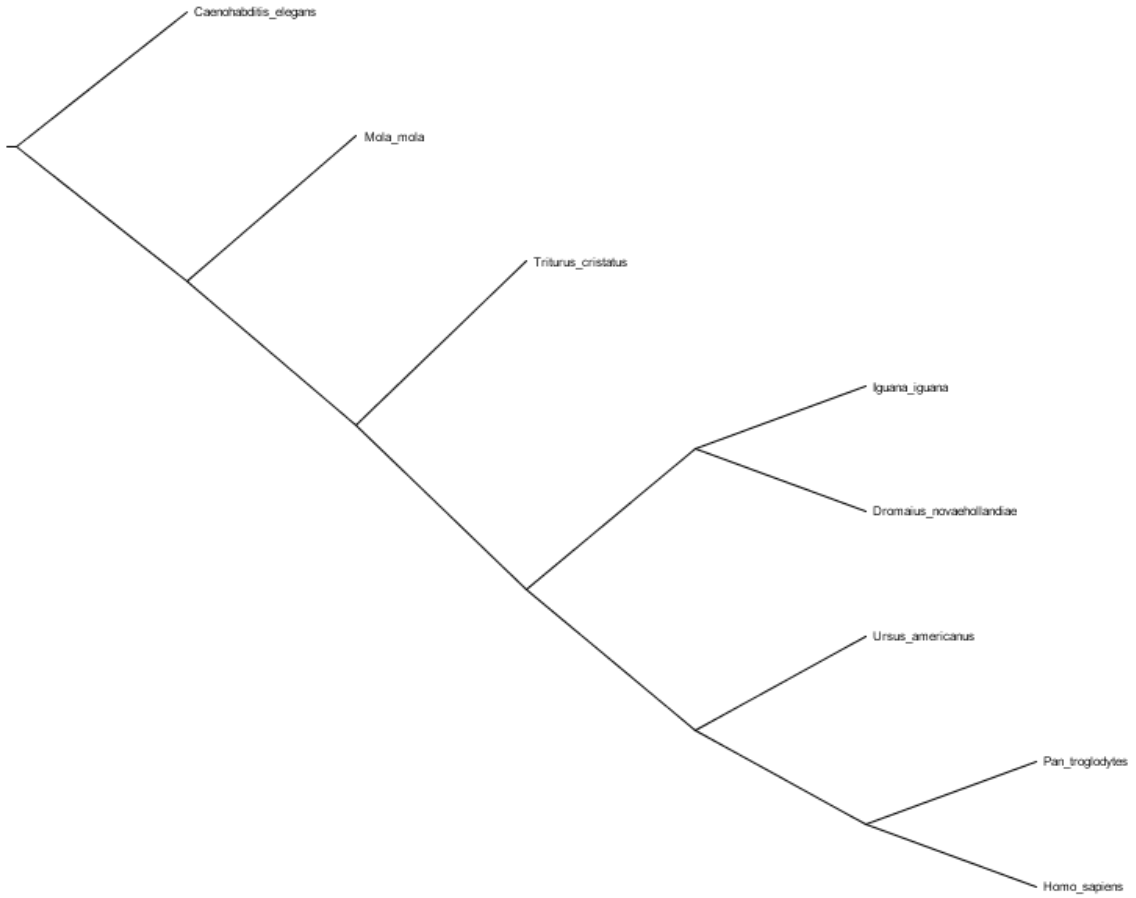


Figure 3: The tree whose topology was used to train the hill-climbing algorithm. The species represented are fairly widely divergent and no microbial or viral taxa are represented.

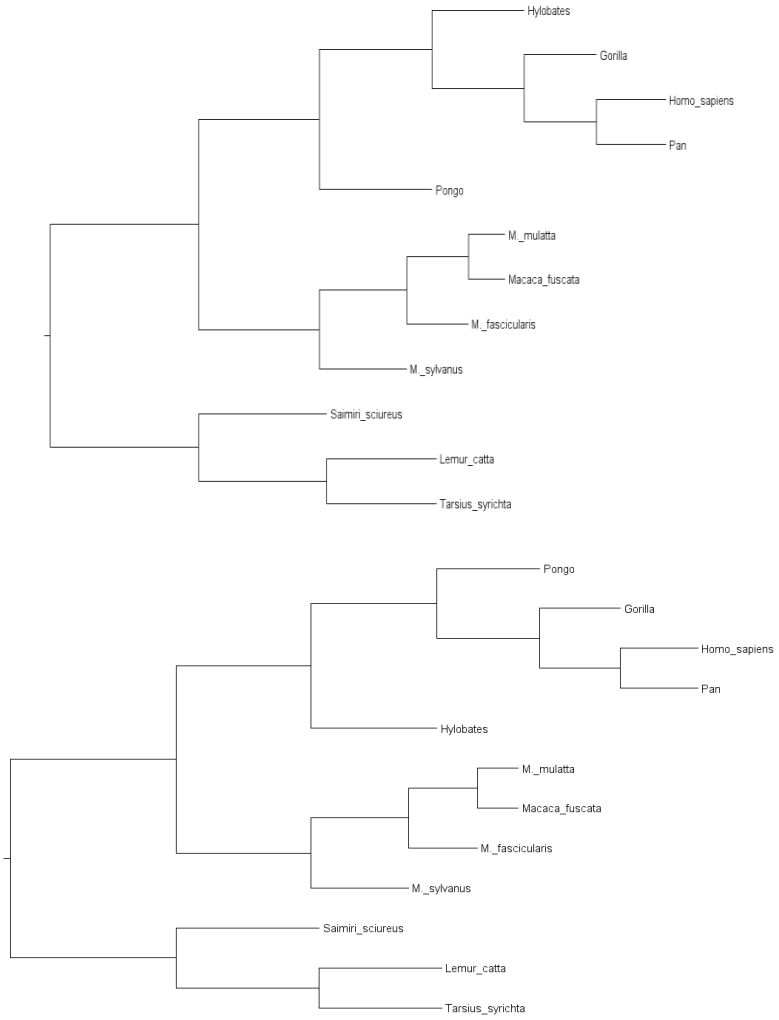
Constructing the Gene Tree: Gene trees are constructed via a simple UPGMA approach where MCCI is used as a surrogate for distance. Ties, which are incredibly unlikely to occur in non-contrived datasets, are broken arbitrarily.

Constructing a Species Tree from Gene Trees: If a species tree is desired, the MCCI scores from which each gene tree is derived are held until all gene trees are computed. The default behavior, which can be disabled via a command line parameter, is to then scale the all the scores to the gene with the greatest mathematical range in the scores between any taxa. Ultimately, the average score between each pair of taxa over all genes is obtained and used to build the species tree. Notably, not all taxa need have data for each gene and the number of genes available is factored into the determination of the average MCCI.

Results:

A variety of trees were produced using Scrawkov-Phy. Accompanying each tree is either a tree constructed via accepted methods or a tree from a paper from which the data was presented. Unless otherwise noted, the programs were run with default parameters. Trees were visualized in FigTree[85] unless otherwise noted.

Primate NADH Dehydrogenase Gene Trees



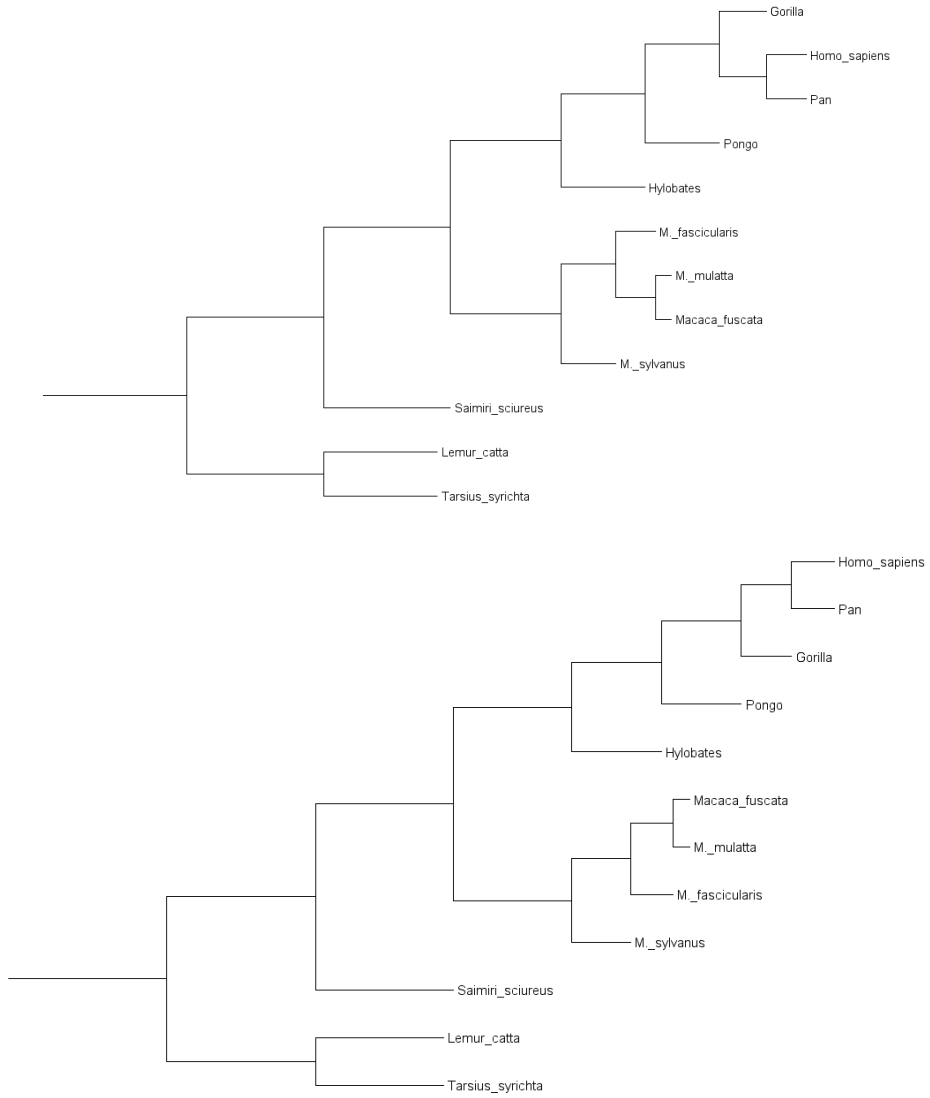
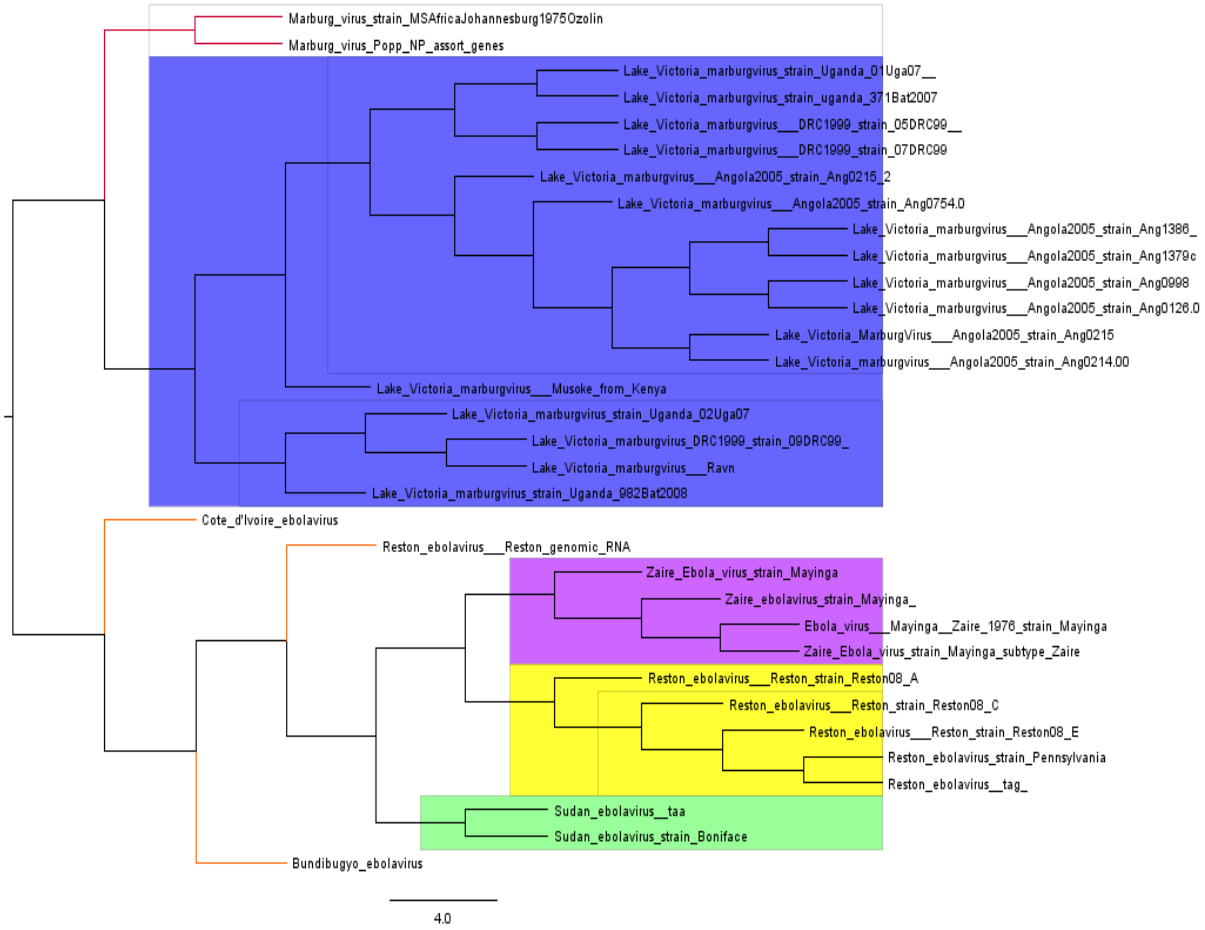


Figure 4: The resultant gene tree from the NADH dehydrogenase mitochondrial gene from selected primate taxa. From top to bottom, the trees were generated using the following methods respectively: an outdated version of Scrawkov-Phy, the current version of Scrawkov-Phy through the EMU-Phy interface, a tree generated from a T-coffee MSA fed to ClustalW2[86,87], and a tree generated from a ClustalOmega MSA fed to ClustalW2. All trees were generated using UPGMA methods. Notably, while the current version of Scrawkov-Phy recovers the accepted gene tree topology, the older version of Scrawkov-Phy implemented without the QHMM mistakenly inverts the placement of *Hylobates* and *Pongo*.

Figure 5: Radial trees of a 181 taxa primate species tree constructed with Scrawkov-Phy by two methods. The top tree is with the normalization parameter on and the bottom tree with it off. Shrews were used to root the trees. The more traditional cladogram view of the trees can be found in supplemental figure SF1.

Filovirus Tree



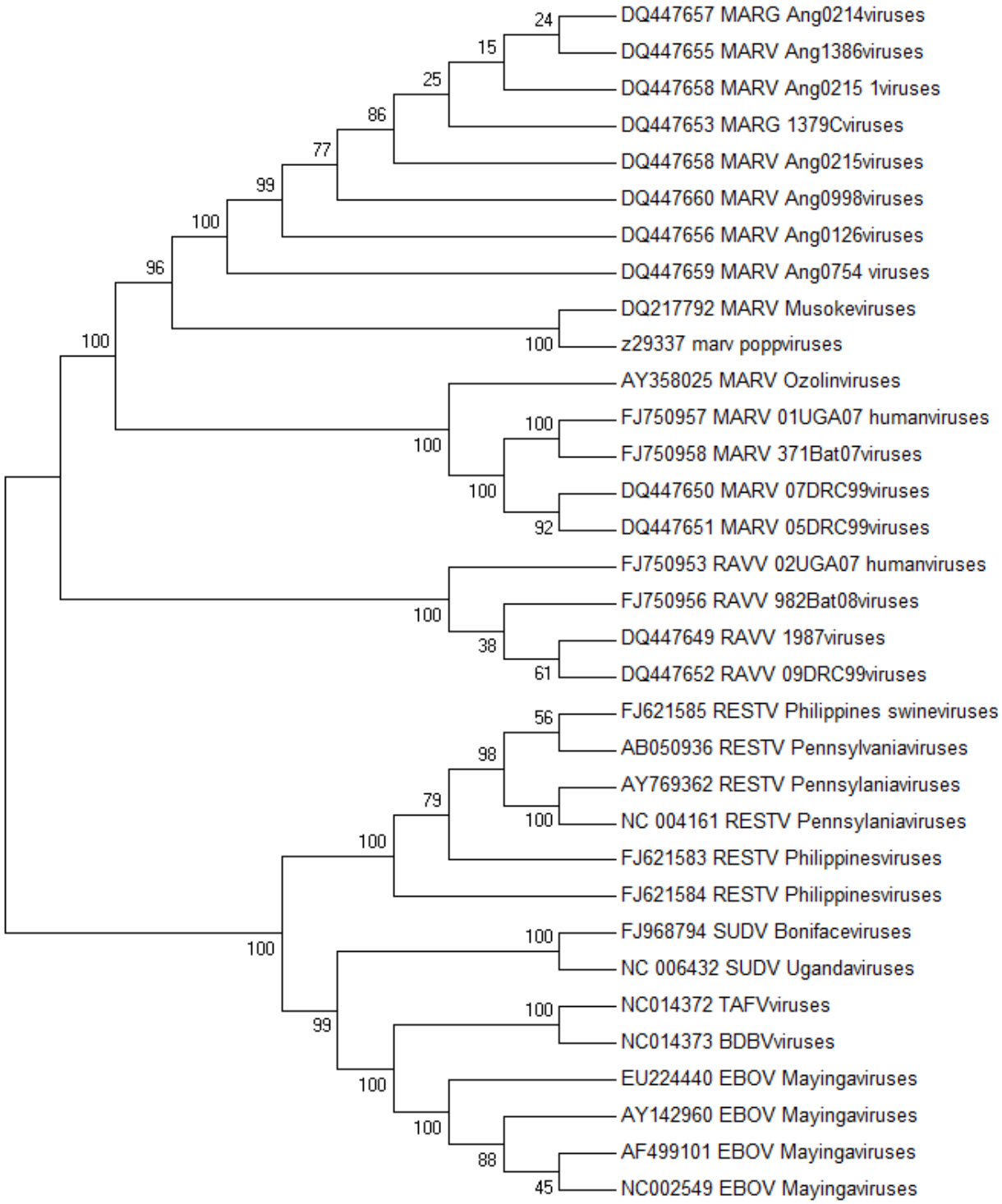


Figure 6: The phylogeny of the filoviruses was constructed by two methods using the whole genome of 33 taxa. Scrawkov-Phy (top) was run with the default parameters and completed the task in under a minute. A Maximum Likelihood consensus tree (bottom) was constructed in MEGA[88]. The consensus is the result of 100 bootstrapping iterations. Alignment was done using

the muscle algorithm in CodonCode Aligner. The genomic data was obtained from GenBank using the compiled list presented in work by Barrette et al.[89] Highlighted regions in the tree generated by Scrawkov-Phy represent groupings which were not in conflict with the tree recovered by Barrette et al. The orange highlighted branches represent conflicts that had a bootstrap value of below 80, but above 70. The red highlighted branches represent a conflict of above 80 in the tree. Notably, only one pairing falls into this category and still falls well within the larger Marburgvirus clade. It is essential to note that the trees are not comparable beyond a surface level since the Scrawkov-Phy tree was generated using UPGMA, which will almost assuredly yield an inferior tree. Incongruence observed may be an artifact of the use of UPGMA.

Passerine Cytochrome b Tree

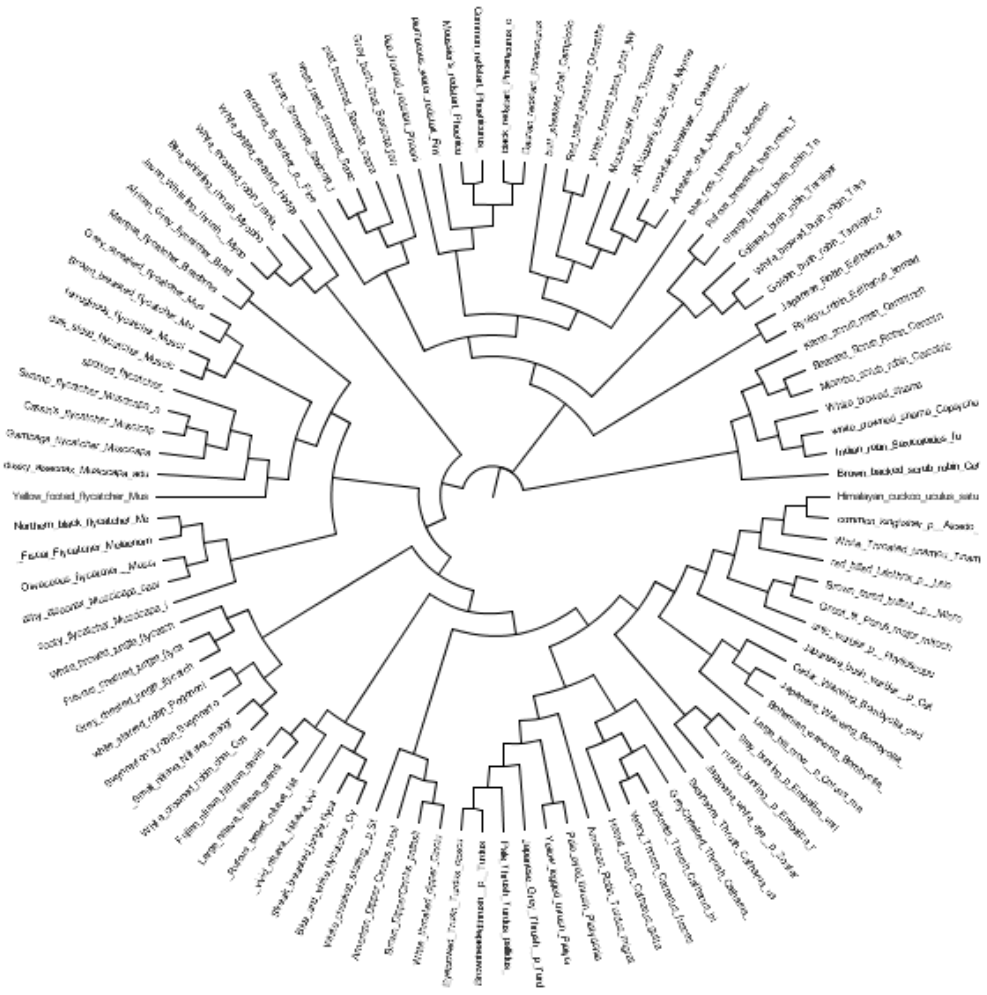


Figure 7: The radial tree of assorted passerine cytochrome b sequences. The top tree is the tree resolved by Scrawkov-Phy while the bottom tree was constructed from a ClustalOmega MSA fed to ClustalW2. The traditional cladogram view of the trees are available in Supplemental Figures SF3 and SF4.

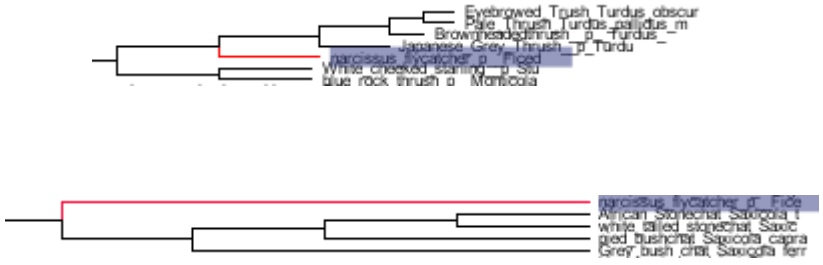
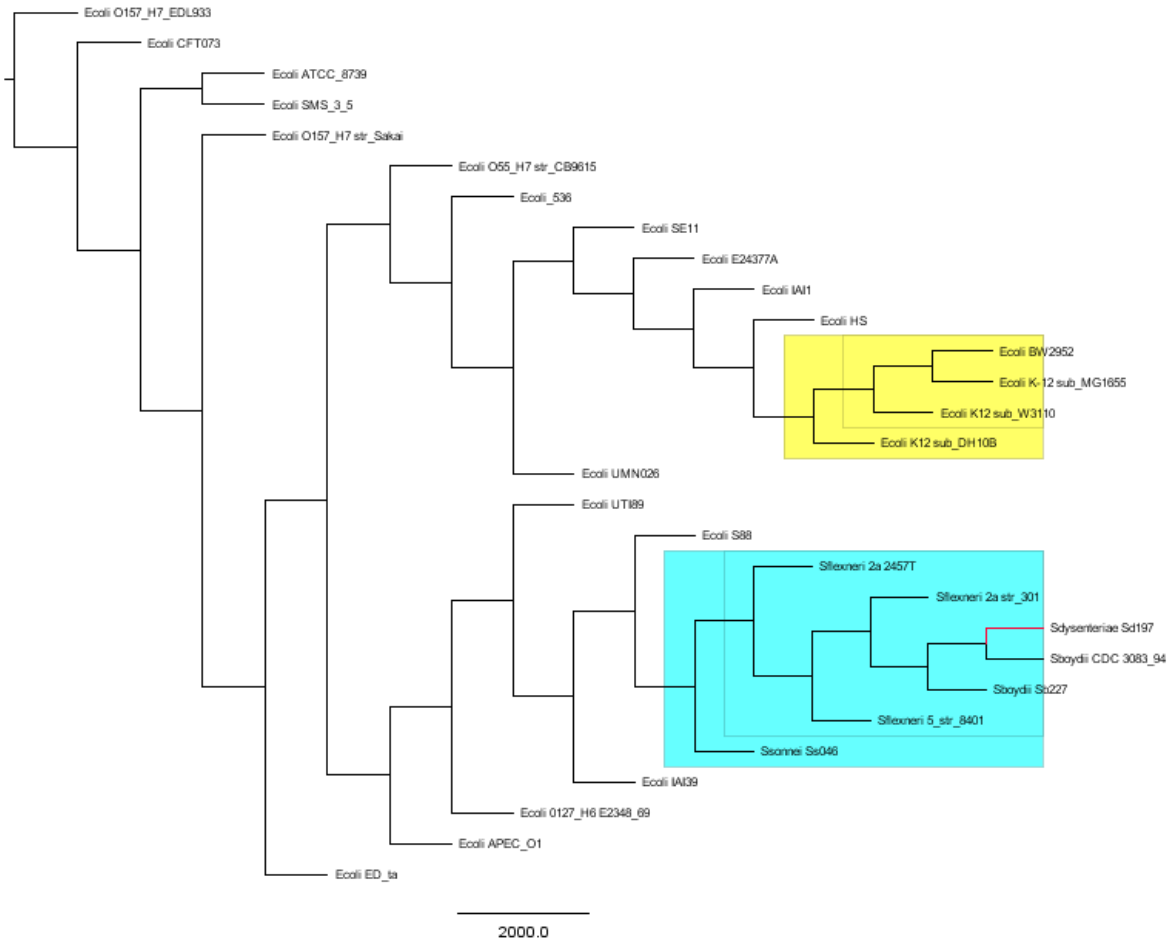


Figure 8: The Scrawkov-Phy tree (top) placement of the narcissus flycatcher (red) replicates the same paraphyly noted by Sangster et al,[90] though its placement does not agree with the tree

generated by ClustalOmega and ClustalW2. Without a follow-up bootstrap analysis further comparison is impossible, as the conflicting nodes cannot be verified as well-resolved in either analysis.

Whole Genome E.coli Trees



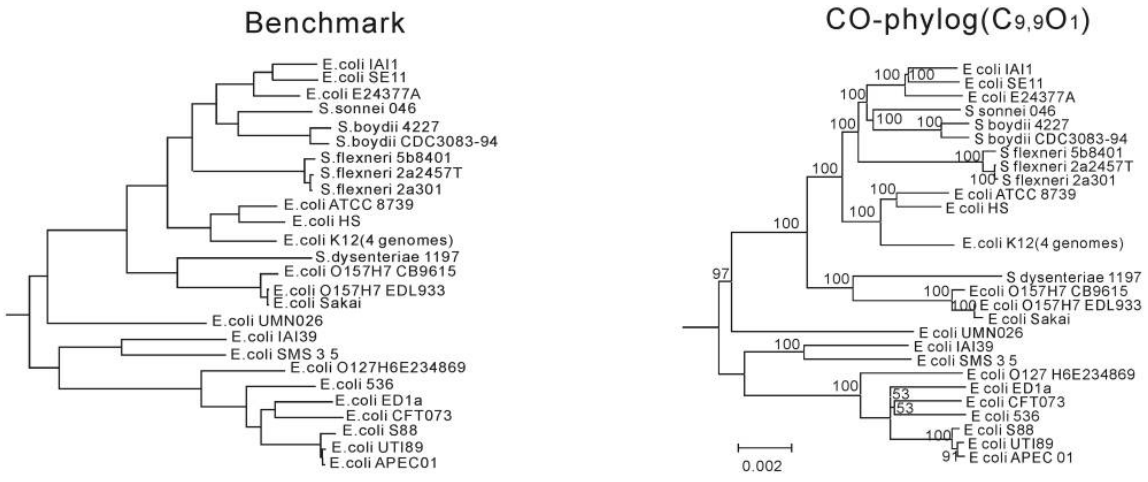


Figure 9: Scrawkov-Phy (top) was used to reconstruct the phylogeny of 29 whole *E.coli* and *Shigella* genomes. This dataset was used to benchmark another alignment free method, Co-Phylog (bottom right), comparing it to the assembled and aligned genome tree (bottom left).[91]

Discussion:

Primate Trees: The gene tree of the primate NADH dehydrogenase shown in Figure 4 demonstrates the efficacy of Scrawkov-Phy at resolving a small-single gene dataset. It also highlights that the algorithm was improved iteratively over time. The most current version of the algorithm recovers the accepted topology for the gene, despite the early version of the algorithm generating a tree the conflicts with the placement of *Pongo* and *Hylobates*. This initial success was a key proof of concept in the development of the algorithm, inspiring the construction of a primate species tree from the data provided by Perelman et al[92].

The species tree, shown in Figure 5, contains 181 taxa represented across 52 loci. The topology recovered was not in complete agreement with the study from which the study came. However, both the normalized and un-normalized runs of the algorithm resolved the core families well. Many groups, such as the *Eulemars* and *Callithrix* clades, clustered perfectly. Additionally, the shrew outgroup used to root the published tree was also naturally placed as an outgroup in Scrawkov-Phy. However, there are a few cases of paraphyly observed. *Cacajao calvus*, for instance, is not monophyletic in either the normalized or un-normalized attempts by Scrawkov-Phy. The two taxa did not cluster with each other or with the *Chiropotes* as they do in the published tree. Interestingly, the result for the un-normalized iteration of Scrawkov-Phy seemingly conflicted less with the published tree than did the normalized iteration. This is likely due to the large number of genes used in the study, with varying amounts of relevant evolutionary signal. Indeed, the authors of the original paper state that these loci account for nearly 90% of the diversity of the taxa used in the study, each with, presumably, differing amounts of phylogenetic signal. Thus, since it is well-reasoned that the genes may contain vastly different amount of signal, the normalization of scores was unnecessary. The runtime for the gene tree in Figure 4 was essentially instant. The species tree had a run-time of roughly half an hour. Example gene trees from which the species tree is derived can be seen in Supplemental Figure 6. It is, again, important to note that since there is no bootstrapping analysis presented, the conflicts and agreement cannot be confirmed.

Filovirus Whole Genome Gene Tree: The filovirus tree, shown in Figure 6, resolved surprisingly well. It barely conflicts with the publication tree and all but one of the areas were in

areas with less than 90% bootstrapping confidence. This result is particularly surprising because the hill-climbing algorithm was not tuned to viral or microbial sequences. Ultimately, the result may be due to the level relatedness between the viral sequences used in the study. It has been shown that phylogenetic signal is more difficult to resolve at the both extremes of dissimilarity. The sweet spot is between highly conserved, highly similar sequences and the so-called twilight zone of poorly conserved, highly dissimilar sequences. It is possible that, though the viruses used in the study are closely related, the high viral mutation rate put the sequence in the sweet spot for detection of phylogenetic signal. Of course, without a rigorous test for branch support, such as bootstrapping or jackknifing, the confidence in the tree is speculative at best. Despite this, the undeniable topology similarity lends, minimally, support in the potential of the algorithm.

Of the viral samples that were in at least partial conflict with the publication tree, two of them belonged to outbreaks which were not widespread geographically. They belonged to independent outbreaks, which may partially explain the difficulty in resolving them in the same way as the publication tree. Additionally, while the topology doesn't match, the 2 Marburg sequences that are in conflict (greater that 90% bootstrap) with the publication tree are only one "speciation event" or layer deep in the tree from being in concordance with the published tree. The same holds with the singular Restonvirus that disagrees to some extent (greater than 70% bootstrap). These are arguably near misses. Tuning the weights using a hill-climbing algorithm with a viral tree may improve the ability of Scrawkov-Phy to even better reconstruct the accepted phylogeny.

Bacterial Whole Genome Tree: Unlike the filoviruses, Scrawkov-Phy did not recover something very close to the benchmarking tree. However, the *Shigella* strains all clustered together (highlighted blue in Figure 9, above), which is a promising result. The *Shigella dysemteriae* branch did not cluster with the other *Shigella* strains in the accepted tree nor in the tree recovered by Co-Phylog, though it was in the case of Scrawkov-Phy. The literature supports this placement of *Shigella dysemteriae* outside of the *Shigella* cluster[93], so the placement of the taxon by Scrawkov-Phy is aberrant. Indeed, the dataset was originally selected in the Co-Phylog to test the robustness of their method to this result. While the debate on the taxonomic status of *Shigella* species is an interesting and impactful topic, further detail is beyond the scope of the present. Essentially, Scrawkov-Phy is not as robust as Co-Phylog, but clusters many things well.

For instance, all 4 K12 E. coli strains used clustered together, which was a consistent with the results of the original paper. There were a number of topological differences and the resultant tree was not nearly as high quality as the one obtained by Co-Phylog. However, it is notable that Scrawkov-Phy completed the analysis in just over a day of run time on a machine with 12GB of RAM, despite 29 whole genomes being utilized in the study. Further work would be needed to truly benchmark the memory usage of Scrawkov-Phy, but these initial results are promising. If the tree used to weigh the feature weights were constructed using some or exclusively microbial taxa, the tree might resolve better. As it stands now, however, the tree recovered by Scrawkov-Phy should be scrutinized before any biological interpretation be made from it. Reconstructing the tree using a more eloquent tree building method, such as neighbor joining, may result the recovery of a better topology.

Paraphyly in the Passeriformes cytochrome b tree: The taxonomy of avian species, especially the songbirds, is and has been historically, an area of focus for ornithologists. This effort has been particularly difficult as there is notable incongruence between the seminal character-based trees and those derived from molecular data. Furthermore, even within molecular data, paraphyly and polytomy are common issues among many bird studies. For instance, Sangster et al noted that there was paraphyly in their classification of Passeriformes according to cytochrome b and nuclear gene evidence. While there are systemic reasons that different genes may have different trees[94], their work implied that the paraphyly was inherent to the dataset rather than a limitation of the methods used. Thus, Scrawkov-Phy was run on the dataset, with additional taxa taken from NCBI, to discover if the same paraphylies were observed. Ultimately, many of the same limitations of the dataset were seen in Scrawkov-Phy.

In both Scrawkov-Phy and the alignment-based tree, there were cases of paraphyly. For instance, the narcissus flycatcher did not cluster with the other flycatchers, as seen in Figure 8. Interestingly, the traditional tree placed the hermit thrush very closely with other the thrushes, whereas Scrawkov-Phy resulted in a paraphyly for this taxon. However, work that came after the study from which the data was obtained has shown unusual paraphyly in the hermit thrush across multiple loci. Ultimately, the placement of the hermit thrush must remain speculative as neither Scrawkov-Phy nor the publication tree provided a bootstrapping confidence for the branches.

This is particularly problematic as there are a number of disparities between the trees, though clustering at the family level is fairly similar.

Limitations and Future Work: Needless to say, the algorithm used in Scrawkov-Phy is imperfect. While it aims to stay more firmly rooted in biology and less in statistics than its predecessor alignment-free methods, many of the principles on which they are based are not universally accepted. The algorithm itself is highly experimental and, despite the success of the algorithm in reconstructing trees and known phenomenon in its namesake clade, there is still much uncharacterized about Scrawkov-Phy. Indeed, there are a number of shortcomings in the algorithm in its current form.

Currently, the weight parameters are only constructed based on tree topology; branch length is not factored into the hill climbing algorithm. To improve Scrawkov-Phy's performance, the parameters should be weighed either in a similar manner as presented here incorporating such an addition or a more verbose statistical method should be employed, such as maximum likelihood. Since the weighted parameters are not determined at runtime (though they are alterable as command line arguments), the additional runtime would be negligible as it would only need to be computed once.

Another consideration concerning the efficacy and future development of Scrawkov-Phy is that, in the form presented here, it does not take into account any features on the strand not explicitly provided by the user. Similarly, codon bias is incorporated into the model despite having no guarantee of being in a coding region or in frame. While there are optional subroutines that alleviate the concern of frame, they remain heuristics that do not truly compare all orthologous sites. Computationally, the algorithm could receive a marked speed-up and reduction in memory usage if matrices were used for some of the underlying data structures.

UPGMA has been shown to be an inferior tree building method when comparted to almost any other established method[95]. The major flaw of UPGMA is that it assumes equal branch length and distance, which is biologically unfounded. This can systemically result not only in incorrect branch lengths, but incorrect placement and grouping of taxa. While the ease of implementation made it ideal for developing Scrawkov-Phy, it severely limits the conclusions that can be drawn from resultant trees, even though obtained MCCIs may work reasonably well

as distance measures. Use of an external tree building program or implementation of a better method is necessary to improve the ability to characterize Scrawkov-Phy and derive biological significance from its results.

Horizontal gene transfer is notably excluded a priori from the present algorithm. Indeed, many general use algorithms do not factor horizontal gene transfer into their calculations, though there are plenty of known algorithms that do. These algorithms, however, tend to be optimized for bacterial sequences, rather than general use, as Scrawkov-Phy was intended. As previously stated, this exclusion may be responsible for disagreement seen in the whole genome *E.coli* tree presented in Figure 9.

Despite the status of the coalescent as the frontrunner for species tree construction, Scrawkov-Phy does not attempt to use it, favoring, instead, a greedy algorithm that maximizes probability stepwise. As such, the construction of species trees using the algorithm presented here is likely not as rigorous as the presently accepted methods. However, it is important to note that the construction of gene trees can be performed in Scrawkov-Phy and piped into a program implementing a rigorous coalescent method, such as *BEAST. Ultimately, the feature was not implemented here largely because the current implementations are of a quality far exceeding that achievable in the scope of this project.

Polytomy, which may be a natural result of a dataset, is avoided in this algorithm and no test for the significance of a branch (such as bootstrapping) currently exists within Scrawkov-Phy. Scrawkov-Phy operates on the assumption of bifurcation, which there may not be statistical evidence to show. Thus, branches with a particularly short length in Scrawkov-Phy should be examined critically before conclusions are drawn. Such branches should be verified with a method such as bootstrapping or collapsed into a polytomy.

The input which Scrawkov-Phy takes is relatively strict and it does not parse relevant information from the meta-data provided in many FASTA headers. Thus, a limitation is presented in that datasets must be manually proceeded by the user for readable output, especially for species trees. This is due to Scrawkov-Phy requiring that all taxa be named identically within each input gene file. Compared to other, comparable methods, this is a considerable shortcoming of the program and future work should prioritize effective parsing of the input. Minimally, the

information presented within each FASTA header can be more neatly presented. One way to accomplish this would to enable the output of nexus files, in addition to newick files.

GC content can be more effectively utilized in the algorithm than is currently implemented. Scrawkov-Phy is limited in to the global use of GC content. However, CpG islands themselves have been shown to be evolutionarily significant. In essence, only global measures of GC usage by taxa are used, whereas it is likely that incorporating local measures of GC usage could improve the efficacy of the algorithm's ability to recover meaningful phylogenies, especially in whole genome datasets or species tree construction.

Arguably, the use of both dinucleotide and trinucleotide frequencies in the algorithm could be considered redundant[96]. While both of these parameters have previously been shown to be significant biologically and are used extensively in well-known programs like those incorporated in MEGA, the signal is undoubtedly mixed. This is problematic due to the hill-climbing approach for weighing the features used in the determination of phylogeny. This method does not separate the effects of mixed signals particularly well, as compared to other machine learning methods. Thus, the evolutionary signal present in the dinucleotide and trinucleotide methods may be overrepresented in the final calculation of the MCCI. A more sophisticated genetic algorithm may alleviate this signal promiscuity[97], though there are a number of machine learning methods more suited to handle mixed signals[98].

A glaring issue that has yet to be addressed is that of k-mer homoplasy. Homoplasy, in a general sense, is similarity due to convergent evolution or reversion, rather than through ancestry. On a nucleotide scale, homoplasy is described as the independent acquisition of the same base at the same position over separate evolutionary lines. K-mer homoplasy refers to the situation that arises when identical k-mers are not derived from regions of the genome that are not homologous physically, evolutionarily, or functionally[12]. Fan et al describe the honeypot that Scrawkov-Phy falls into. Short k-mers are more likely to be resistant to multiple evolutionary events, sensing them each individually. However, shorter sequences are also more likely to occur due to chance and not be comparable as homologous in any sense. There exists, then, some equilibrium or balance in k-mer length used in alignment free methods. As previously described, Scrawkov-Phy essentially uses 2-mers and 3-mers to infer phylogeny (via the QHMM). In this respect, Scrawkov-Phy is likely to fall victim to inaccuracies due to k-mer homoplasy.

Another shortcoming of the method as it exists here is that the branch lengths are not directly comparable to trees constructed by traditional methodologies. The MCCI is presented as a semi-log aggregate score, which results in non-linear scaling of the branches. Additionally, since no formal substitution model is used, this method cannot be tuned to a molecular clock to resolve the branch lengths temporally. Thus, only limited information about the timeline of the evolution of the species can be recovered and the branch lengths are not well-suited for comparison to traditional methods. A means of improving the comparability of the algorithm is to obtain LOD scores for all of the parameters, not just those used in the QHMM.

Though not performed here, a standard in the field is to test tree building methods not only on real biological data, but on simulated data for which the true phylogeny is known[99]. While the model used to simulate the data would be non-trivial in the accuracy of the results of a method tuned to particular biological parameters, it is nonetheless a standard and future characterization of Scrawkov-Phy should include exploratory work with simulated sequences.

While Scrawkov-Phy uses particular biophysical principles to justify particular genomic features being used in the calculation of the MCCI, it does not use any proper biophysical measure as a feature. For instance, Shannon's entropy would be an easily computable measure that could be incorporated in the determination of MCCI. Additionally, while the chaos representation of RNAs was used to justify the use dinucleotide transition properties in the QHMM, the calculation of chaos was not calculated. While this calculation may slow the algorithm down, it would likely lead to cleaner signals than the surrogate presently used in the QHMM.

Runtime of the algorithm, as presented here, is by no means slow compared to the traditional methods. However, the algorithm essentially runs several times to determine the starting frame that results in the maximum score. However, due to there being several factors used in the computation of the MCCI, there could be checkpoints after the calculation of each of the respective components. These checkpoints could compare the current MCCI score to the best score. Since the algorithm aims to minimize MCCI, if the current frame's partial MCCI is greater than the current best MCCI, then computation can terminate for this frame, saving runtime and memory. The result would be an effective pruning of the search space not unlike that of branch and bound approaches. In this case, however, it is important to note that the order in which the

components are calculated is non-trivial. Optimally, the feature that is most heavily represented in the final MCCI would be computed first, thus maximizing the chance that a disparity between frames would become apparent in the fewest number of computational steps. The order of calculation in Scrawkov-Phy in its current form, however, is trivial and there is no in-place framework to change the order of calculations because the weights can be changed via the command line arguments. As a result, the order for the algorithm's calculations that are most optimal for pruning is not known. In future releases of Scrawkov-Phy, if such pruning is incorporated, it would be prudent to include a feature which allows the user to change the ordering of the calculation via command line arguments.

While Scrawkov-Phy did not fail to run on any system it was run on due to memory restraints, there are still a number of areas that could reduce the memory usage of the algorithm. Specifically, the sequences should not be stored after the features therein have been measured. For the generation of species trees, the algorithm holds onto more sequence information than is necessary. Instead, the algorithm might be better off having a command line argument that lets the user toggle the ability to read the sequences from ROM when they are needed, rather than holding them in RAM. This would allow for use of larger still datasets.

An additional area for consideration in this algorithm is the normalization of gene MCCI's to maximum and minimum scores in the computation of species trees. On one hand, the normalization of scores across differing genes seems intuitive. This allows for dissimilar data (long versus short genes, closely related versus distantly related taxa, etc.) to be compared without skewing the data towards outliers or oddities. On the other hand, however, this normalization assumes that each component gene contributes equally to the phylogenetic signal. This assumption is, of course, false as differing rates of mutations and selective pressures have been described in different genes in a broad spectrum of organisms verbosely. Thus, the ability to turn normalization, which is available as a command line parameter, is crucial. Unfortunately, it is difficult to programmatically assay the heterogeneity of the input data, which results in the end user needing to know their dataset and the role of the normalize argument well in order to effectively utilize this tool. Notably, this is likely a strength that alignment-based methods have over Scrawkov-Phy, due to the emphasis on the direct comparison on orthologous sites.

The parsing of standard FASTA formatted file headings would be essential in future work concerning Scrawkov-Phy. The comparison of the results of trees is significantly more difficult without conforming to the standard. However, due to NCBI announcing the retirement of the 'gi' identification system, a conscious effort was made to wait until the phasing out was completed.

Not all improvements to the algorithm need be borne of limitations presented here. For instance, MEGA uses differences in the trinucleotide and dinucleotide compositions as part of its algorithm, but does it in an apparently more sophisticated way using a matrix based approach. Biophysical or epigenetic profiles could conceivably be incorporated in future iterations of the algorithm. Regardless of the component or motivation for incorporation of addition features, it is important to note that there may be redundancy in the signals caused by such incorporation. This sort of redundancy in signals has been noted as a challenge in the feature selection subset problem in machine learning since it has become feasible to employ such methods. While there has been work describing automated selection of such features, the hill-climbing algorithm is an acceptable compromise for direct biological context and feature optimality. Needless to say, addition of additional features to the algorithmic determination of phylogeny should be considered carefully as to not loose biological context, not over represent a signal through redundancy, and to not fit the elephant with an overly parameter-rich model.

Despite its limitations, Scrawkov-Phy utilizes a promising algorithm that demonstrates that there is strength in using simple models in alignment-free phylogeny. While the algorithm's limitations have not been rigorously studied, the initial findings presented here suggest that alignment-free methods need not necessarily be based on high order statistical models from which biological context may be lost. Ultimately, it is up to the end user to intelligently utilize the tool and its options to maximize its efficacy.

❖

Chapter 2

❖

EMU-Phy: An **E**xtensible **M**anagement **U**tility for **Phy**loinformatics

J. Nick Fisk

Background:

Phyloinformatics:

Phylogenetic inference is playing an ever larger role in nearly all fields in biology. Naturally, the extent of this role varies, from integration into a full systems biology analysis of a phenomenon to the enrichment of a few taxa of interest. One thing is clear, however. There are an increasing number of methods available for performing phylogenetic inference. This, coupled with ever growing sources of molecular data, has facilitated sophisticated phylogenetic analysis. However, new challenges in assembling, organizing, and connecting the various software and datasets have arisen from this bounty of phylogenetic resources. Enter phyloinformatics. Phlyoinformatics, a term which has some ambiguity in its meaning, generally is a field whose goal is to streamline phylogenetic analysis computationally. Different phyloinformatics utilities approach this goal differently. Some phyloinformatic applications focus on data mining common biological databases for relevant information. Others aim to incorporate metadata in to enrich the standard analyses. A modified diagram from Roderic D.M. Page[100] details a potential phyloinformatics database design, in Figure 1, below.

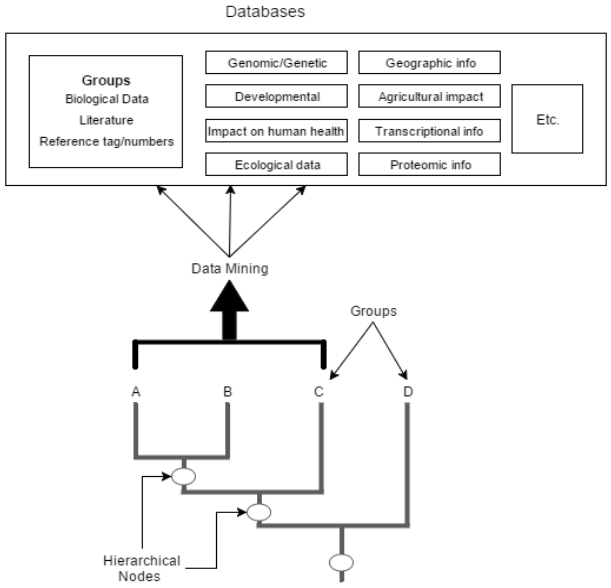


Figure 1: A schematic diagram depicting a hypothetical phyloinformatics database optimized for data mining and learning.

Regardless of the approach, the theme remains the same: Phyloinformatics consists of largely connecting or repurposing existing tools into pipelines to conduct meaningful analyses on large and dynamic data. The Extensible Management Utility for Phyloinformatics (EMU-Phy) presented here aims to provide a simple, but adaptable, phyloinformatics tool that focuses on organization and maintenance of phylogenetic oriented data with appeal to those comfortable with command line tools, as well as novices.

TreeBASE and Phyutility—Inspiration for EMU-Phy:

In the domain of phyloinformatics, TreeBASE[101] and Phyutility[102] are among the most well-known applications. They both approach the ultimate goals of phyloinformatics quite differently, placing different focus on different aspects of phyloinformatics. For instance, TreeBASE, at its base, operates very much like a standard, centralized database. It allows users to pull down data on phylogeny, character evolution, biogeography, and, of course, molecular datasets. However, the project has, over time, added a series of applications that allow for analysis of data in place. These applications facilitate or perform analyses such as method comparisons, supertree construction, and co-evolution based on data contained in the database. Thus, it can be argued that the functionality of TreeBASE was focused on application extensibility and user ease while providing a central database. While this makes TreeBASE an incredibly useful resource, it is not as straightforward when novel data is generated and a user wishes to analyze it. In scenarios such as this, Phyutility may be a more promising candidate.

Phyutility is another application that can be categorized as phyloinformatic. It focuses on providing a local, command line interface that connects a number of existing phylogenetic software. Additionally, it prides itself on the ability to manipulate tree topology (re-rooting multiple trees) and summarizing the variation of tree topology. Like many other programs, it has the ability to fetch data from the NCBI and can handle a variety of phylogenetic file formats, like the Nexus and newick formats. While its scope is limited, its limitations and purposes are well defined, making the program well respected and containing great utility. Ultimately, EMU-Phy draws philosophic inspiration from both Phyutility and TreeBASE.

EMU-Phy, like Phyutility and TreeBASE, has its own approach to phyloinformatics. It is considerably more lightweight than either Phyutility or TreeBASE. Similar to Phyutility, however, EMU-Phy is instantiated on individual machines, making it more of a tool for handling data a lab has generated, rather than datamining as TreeBASE allows for. However, like TreeBASE, EMU-Phy was designed for the addition of new modules. Thus, an interface allowing extensibility of the program was created. Perhaps most importantly, EMU-Phy differs from both Phyutility and TreeBASE in that the directory structure is purposefully simple. This allows for advanced users (i.e., those with programming ability) to use their own skills to customize their experience easily, while still facilitating ease of use for novice users (e.g. by making the locations of images and trees intuitive).

Software Design Principles:

Java is an exceptionally popular object oriented programming language developed and supported by Oracle. The Java language depends heavily on the Java Virtual Machine (JVM), which is an abstract computing machine that facilitates the design goals laid out at its conception. The famous five goals[103] of the Java language are for the language to be: "Simple, Object Oriented, and Familiar", "Robust and Secure", "Architecture Neutral and Portable", "High Performance", and "Interpreted, Threaded, and Dynamic". These design goals were achieved and incorporated into the language as it is known today. Due to this, a number of properties are inherent to Java and code written in it.

System Independence:

Java is considered a platform independent language. In layman's terms, it means that Java code written on a machine running a Windows OS will run on a machine running OSX without any alterations. This is in stark contrast to languages such as C or FORTRAN, which are platform dependent—code written and compiled on one machine is not guaranteed to run on another, even if the operating system is the same. Java accomplishes this via the Java Virtual Machine which acts as a pseudo-platform for which code to run on. It is a universal interface to the machine on which code is being executed.

This property of platform independence makes it invaluable for developers of applications which may be used by a wide variety of users running a wide variety of operating systems on a wide variety of machine architectures. Scientists, including biologists, fit this description well as science is carried out on the global scale. Indeed, the use of a programming language which promises platform independence ensures consistency in the analysis of scientific data—a tenant which is fundamental to the advancement of science. This makes Java a great candidate for biologically oriented software, including those used in phyloinformatics. System independence is not the only quality qualifying Java for the development of a phlyoinformatics application.

Extensibility:

Extensibility, simply put, is the ability of a program, software, or system to be improved upon via new function. A software system is extensible if significant extension of its scope or capabilities can be incorporated with little to no alteration of the base code. Extensibility is a software design principle. That is, for a system to be extensible, the code must be designed and implemented with this intent. Java makes the design of extensible code approachable with Java interfaces and abstract classes, which are made practical with the hierarchical class system in place in the Java language.[104]

Extensibility is another property invaluable to programming for the sciences. Fundamentally, scientific achievement is built upon previous findings, given those findings have been robustly tested and reviewed. Good code for scientific application, then, should follow the same principles. Extensible code would, ideally, be able to grow and adapt as relevant knowledge becomes known, thus minimizing the overhead of creating new code from scratch. This would aid the progression of research by allowing a community of developers to continue to work on a single project without the need to restart as science advances. This paradigm is even more important for databases and database management systems, as their entire purpose is the archival and retrieval of data over time. Needless to say, this applies to data relevant to phyloinformatics as well. This, combined with the other aforementioned properties, make Java an excellent candidate language for the development of EMU-Phy.

Programs Integrated:

In the iteration of EMU-Phy described here, 4 software pertaining to the construction of phylogenetic trees were considered for module/interface development. Scrawkov-Phy, which was developed concurrently with EMU-Phy, is an alignment free phylogeny algorithm capable of creating gene and species trees written in Java. While its efficacy is still being explored, the familiarity and system independence of the software made it ideal for integration into EMU-Phy. ClustalW, a well-regarded and simple alignment software, was also chosen to have a module constructed. PHYLIP, which is a series of programs containing methods for phylogenetic analysis, is also represented.[105] Specifically, dnadist, a program for calculating DNA distances using different substitution models, and neighbor, which creates a neighbor-joining tree given a matrix of DNA distances, were chosen. Lastly, the coalescent species tree building program, *BEAST, was also chosen to ensure that both gene and species trees could be constructed. *BEAST is a module of BEAST, which deals with the construction of phylogenetic trees from Bayesian models.

Methods:

Development: EMU-Phy was developed on a Windows 8 system using Java 7. The manipulation of the flat-file database maintained by EMU-Phy is entirely system independent. However, depending on the implementation of the interface, system independence cannot be assured because the software which is being called may not be available for all operating systems. The classes which comprise the base EMU-Phy package are shown in Table 1, below. The source code and Javadoc are available under a GNU GPL v.3 license at https://github.com/nickjfisk/EMU_PHY,by email at nick.j.fisk@gmail.com, or in the supplementary materials of the present. Notably, no protein methods were explored in the present work.

Table 1: Java classes and interfaces of EMU-Phy

Java Class	Description	Used in which process
geneTreeMethod (interface)	geneTreeMethod is a Java interface that, when implemented, uses the operating system to call the	Creation of gene trees using the method of choice
speciesTreeMethod (interface)		
alignmentMethod (interface)	Used in tree building after all scoring, normalizing, and composition aspects completed	Tree-building from composite index
EMUPhy (main)	Contains the main functional elements of the program. It creates the main data structures that keep all the information about and contained in instances of the Bird class organized. It calls on methods defined in both Node and Bird to construct a Newick formatted phylogenetic tree.	Integration, normalization, and maximizing of the scores. Construction of main data structures General program flow
Fisk 2016		

Testing: The base EMU-Phy data management system was tested on machines running Windows (7, 8, and 10), OSX (Mavericks), and Linux (Ubuntu and Fedora). Manipulation of the data was robustly tested for a reasonable number of datasets of varying scope and size. A list of commands available in the base EMU-Phy is shown below in Table 2.

Table 2: Selected commands available in the base EMU-Phy program.

Command	Options	Description
Help	N/A	Displays the help message. Within some interactive commands, help will also provide the options specific to that command.
Quit	N/A	Prompts the user to verify if they want to exit the program and, if so, exits.
Install	Examples	Initializes the filesystem for use. If examples are desired, a limited primate dataset is unpacked into the system.
Update	N/A	If the file structure differs from the internal record of the program, update will attempt to change the contents of the database to reflect the internal record.
Redo	N/A	Calls the analysis functions for a particular experiment, erasing previous entries if they exist.
Validate	N/A	If the file structure differs from the internal record of the program, validate will change the internal record to reflect the data that is actually present. This command must be run if outside scripting is used to place data.
Add	Gene, taxa, group, primer, taxonomy, done, cancel	Prompts the user to enter the appropriate information to create the selected option. Some options, such as genes, support being added from another location on the filesystem. The internal record will be updated without calling 'validate'
Show	Gene, taxa, group, primer, taxonomy, done, cancel	Displays a list of items in the database, depending on the option selected.
Delete	Gene, taxa, group, primer, taxonomy, done, cancel	Permanently removes the selected item, given an option. The internal record will be updated without calling 'validate'.
<shortcuts>	Many	Shortcut commands that allow the user to add or remove items without layers of interaction as required by the 'add' or 'remove' commands. See the documentation for specific commands.
Fisk 2016		

Testing of the modules still in development was performed on the systems for which the module was developed and reasonably available. A summary the datasets used for testing is shown in Table 3, below.

Table 3: Description of the datasets used in the testing of EMU-Phy

Dataset	Description	Used to Test	Notes
Primates	A 181 taxa dataset consisting of 52 loci each.	Scrawkov, ClustalW, PHYLIP, *BEAST	Often, subsets of this data were used, including single genes chosen arbitrarily.
Birds	A 50+ taxa dataset of cytochrome b sequences	Scrawkov-Phy, ClustalW, PHYLIP	Relatively short sequences
Filovirus	Whole viral genome of 33 taxa	Scrawkov-Phy	Sequences likely too long to be handled by ClustalW easily
E.coli/Shigella	Whole genome of 29 taxa	Scrawkov-Phy	Sequences likely too long to be handled by standard programs easily
Mammal	A 12 taxa dataset of NADH dehydrogenase sequences.	Scrawkov-Phy, ClustalW, PHYLIP	Small dataset good for prototyping.
Fisk 2016			

Results:

Base EMU-Phy: The results of testing the base program for EMU-Phy were as expected. Due to careful software design and use of system independent methods, EMU-Phy encountered no issues running internal commands as described. All internal commands were tested on multiple phylogenetic datasets. Importantly, the database filesystem also remained functional when acted upon by external scripts. Namely, the bash shell script to migrate the large primate data from chapter one of the present document into the filesystem was integrated flawlessly upon calling the 'update' command, as intended. A visual representation of the filesystem is presented in Figure 2, below. It is important to note that the flow of the initial input is not depicted for the sake of clarity and readability.

EMU-Phy Modules: The results of testing the modules on their intended systems were of mixed quality. In cases which the call to the module resulted in error, the base EMU-Phy system would often crash. A summary of the results of testing the modules is shown in Table 4.

Table 4: Summary of Module Testing

Method	Operation	OS	Result	Notes
Scrawkov-Phy	Gene Tree	Windows (7,8,10)	Success	Unconditional Success
Scrawkov-Phy	Species Tree	Windows (7,8,10)	Success	Unconditional Success
Scrawkov-Phy	Gene Tree	Linux (Ubuntu/Fedora)	Success	Unconditional Success
Scrawkov-Phy	Species Tree	Linux (Ubuntu/Fedora)	Success	Unconditional Success
Scrawkov-Phy	Gene Tree	OSX (Mavericks)	Success	Unconditional Success
Scrawkov-Phy	Species Tree	OSX (Mavericks)	Success	Unconditional Success
ClustalW	Alignment	Linux (Ubuntu, Fedora)	Success	Unconditional Success
PHYLIP	dnadist	Linux (Ubuntu, Fedora)	Conditional Success	Success conditional upon Perl installation
PHYLIP	neighbor	Linux (Ubuntu, Fedora)	Conditional Success	Success conditional upon Perl installation
PHYLIP	dnadist	Windows	Failure	Failure, even with installation of Strawberry Perl

PHYLIP	neighbor	Windows	Failure	Failure, even with installation of Strawberry Perl
BEAST	*BEAST (Species Tree)	OSX	Failure	Tutorials include BEAUti, limited tutorials on command line interface
Fisk 2016				

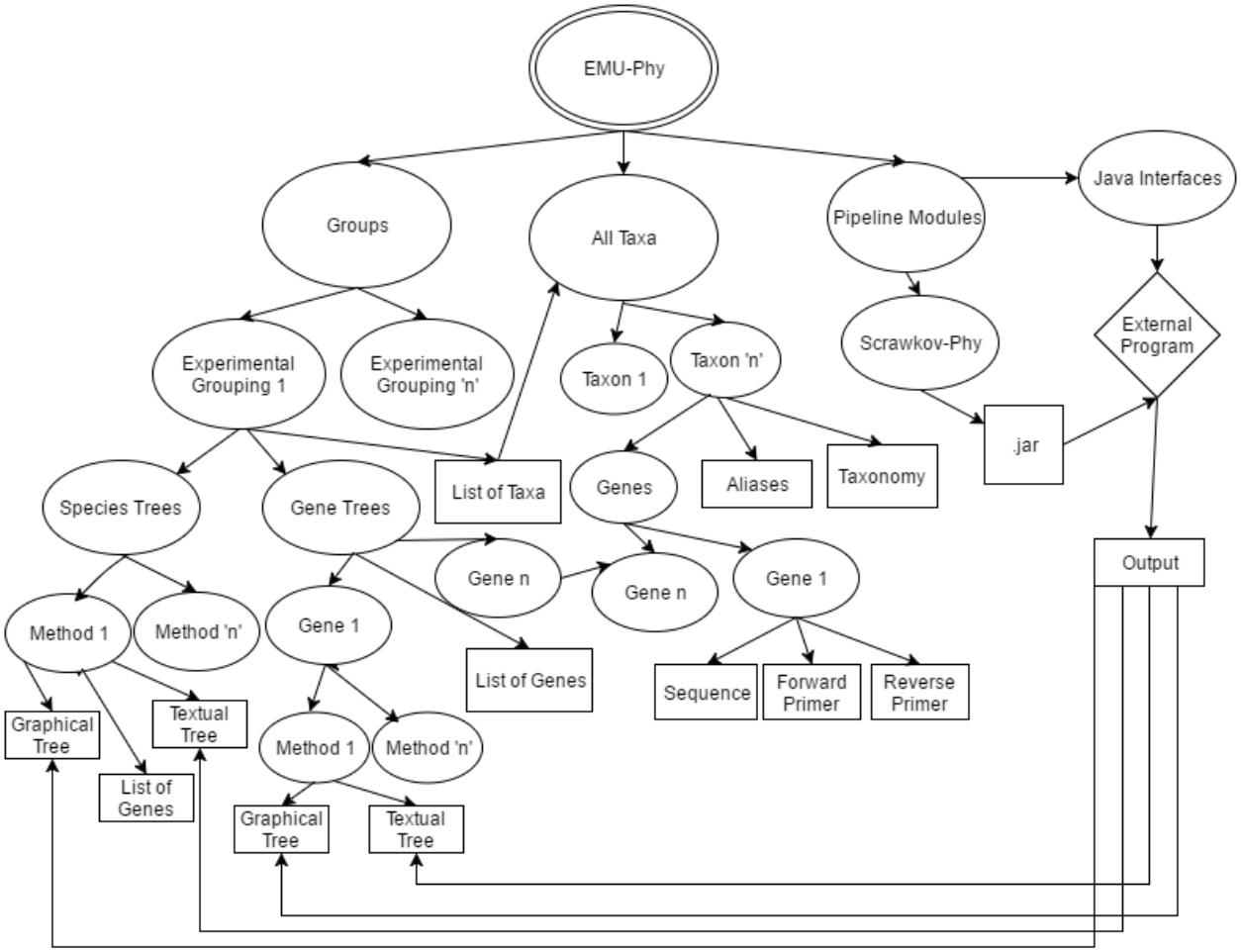


Figure 2: Schematic Diagram of the filesystem used in EMU-Phy. Circles represent directories and squares represent files. Arrows from circles all indicate subdirectories or links. Arrows from squares represent transfer or copying of data. The sole diamond represents system level interactions with external programs via Java system calls. For simplicity, the input connections are not shown.

Below are selected screenshots of the program being executed and interacted with. Associated above each screenshot are captions describing the process being depicted, as well as comments if anything of particular note, is shown.

The welcome prompt upon instantiation of EMU-Phy:

```
Welcome to EMU-Phy!
Please enter a command. Type 'help' for help and 'quit' to quit
```

Installation of the internal file system used by EMU-Phy. The optional example data is installed

```
Please enter a command. Type 'help' for help and 'quit' to quit
install
Installing EMU-Phy in working directory...!
C:\Users\Nick\workspace\ScrawQ
Examples desired? (Will insert example data in datasytem)
Valid options are 'y' or 'n' or 'cancel')
y
Installing EMU-Phy with examples...
Ready for next command      (enter 'quit' to exit)
```

The show command used in two different manners--First with the show and the option selected separately and secondly with the command truncated into one line.

```
Ready for next command      (enter 'quit' to exit)
show
Show what?
Options are: groups, taxa, genes, primers, taxonomy. Enter 'cancel' to cancel or 'done' to finish
taxa
Taxon are...

charmander
pikachu
squirtle
bulbasuar

Ready for next command      (enter 'quit' to exit)
show taxa
Show what?
Options are: groups, taxa, genes, primers, taxonomy. Enter 'cancel' to cancel or 'done' to finish
Taxon are...

charmander
pikachu
squirtle
bulbasuar
```

Addition of a new group to the database

```
Please enter a command. Type 'help' for help and 'quit' to quit
add
Add what?
Options are: group, taxa, gene, primer, taxonomy. Enter 'cancel' to cancel or 'done' to finish
group
Name of new group to add? (type 'cancel' to cancel
Digimon
adding group Digimon...
Digimon was added sucessfully!
Add what?
Options are: group, taxa, gene, primer, taxonomy. Enter 'cancel' to cancel or 'done' to finish
done
Ready for next command      (enter 'quit' to exit)
```

Displaying the groups in the database. Note the cancel command to terminate the addition of multiple groups

```
Please enter a command. Type 'help' for help and 'quit' to quit
show
Show what?
Options are: groups, taxa, genes, primers, taxonomy. Enter 'cancel' to cancel or 'done' to finish
groups
Groups are...

Pokemon
Digimon

Show what?
Options are: groups, taxa, genes, primers, taxonomy. Enter 'cancel' to cancel or 'done' to finish
cancel
Ready for next command      (enter 'quit' to exit)
```

Manual addition of a gene to a taxon via the direct method. Alternatively, a path to a FASTA formatted file would be accepted.

```
Ready for next command      (enter 'quit' to exit)
add
Add what?
Options are: group, taxa, gene, primer, taxonomy. Enter 'cancel' to cancel or 'done' to finish
gene
Name of taxa to add gene for? Type 'display' to see available taxa (type 'cancel' to cancel
charmander
What is the gene name?
Pikachurin
Using gene name: Pikachurin
DNA/RNA sequence required...
To supply a path to a file containing only this gene in FASTA format, enter 'path'...
To supply the sequence directly, enter 'direct'...
To cancel addition of gene, enter 'cancel'...
direct
What is the sequence?
ATGATGCATGACTTAGATACAATATATATAG
Gene addition sucessful!
```

Display of the genes available for a particular taxon. Note that the available taxa can be listed by the display command within the show command.

```
Please enter a command. Type 'help' for help and 'quit' to quit
show
Show what?
Options are: groups, taxa, genes, primers, taxonomy. Enter 'cancel' to cancel or 'done' to finish
genes
Name of taxa to display genes for? Type 'display' to show all available taxa (type 'cancel' to cancel
display
Displaying available taxa

charmander
pikachu
squirtle
bulbasuar

Name of taxa to display genes for? Type 'display' to show all available taxa (type 'cancel' to cancel
charmander
Displaying Genes for Taxa: charmander...
EMBER
SCRATCH
GROWL
LEER
```

Figure 3: A collection of sample internal commands available in EMU-Phy. For a full list of commands available, see the documentation available at the aforementioned GitHub or use run the program and enter ‘help’.

Discussion:

Ultimately, the base EMU-Phy program is complete and ready for release. It has particularly useful in repeating analyses as new data trickles in. However, the module components are lacking in both complete functionality and volume, severely limiting its utility. Naturally, Scrawkov-Phy worked easily as it was developed concurrently with EMU-Phy. Indeed, most of the tests performed in the development and characterization of Scrawkov-Phy were performed using EMU-Phy. Additionally, the simplicity of the ClustalW command line interface made the construction of a module for it relatively straightforward. Other modules, however, did not achieve nearly the desired level of functionality due to various obstacles.

EMU-Phy operates on the assumption that a user attempting to use a module has the underlying program installed on their machine. Thus, a logical approach that was utilized here was to use system calls from Java to run the programs and then redirect the output into the location it needed to be in the file system. However, PHYLIP, while command line based, still requires user interaction for proper function, even if the input and output files are provided explicitly. Thus, a simple operating system call from Java is insufficient for the module for dnadist and neighbor. However, utilization of a Perl script which passes in input to the program granted success in the Linux cases. However, Windows does not natively support Perl and, even when the popular Strawberry Perl is present on the machine, the script will not function properly and the call to PHYLIP fails. Thus, a Java based analog to this Perl script need be developed for the module to be fully realized. *BEAST is a beast of a program which is part of a larger suite of programs. While a large number of tutorials exist for the program, most of them are for tools as used through their BEAUti[106] GUI interface, which made the development of a module more difficult than anticipated.

There are a number of features that, if implemented, would greatly improve the efficacy and utility of EMU-Phy as a tool. Firstly, a vignette could be constructed that highlights the key features of using, and potentially developing for, EMU-Phy. The infrastructure for such a walkthrough was included, as the installation optionally includes example files on which such a vignette could be based. This will allow users to more effectively and quickly learn the basics of the tool, which is imperative if a widespread audience is to be reached. Furthermore, the program

could be given to testers with differing levels of command line experience. The program could then be iteratively improved based on the feedback of the testers.

While Scrawkov-Phy has potential as a tool, it is by no means a well-fleshed or universally accepted tool for phylogenetic analysis. An improvement upon EMU-Phy would be the bundling of additional software for this analysis along Scrawkov-Phy. Preferably, such a program would be relatively portable and generally well regarded. PHYLIP, in this case, fits the bill, as all it requires is a C compiler to install. Additionally, the most recent release of the PHYLIP source code was released under a license that would allow such a bundling to occur, in contrast to previous releases. The coupling of a more mature program, such as PHYLIP, with EMU-Phy would significantly enrich the tool.

A key functionality held by many phyloinformatic applications is the ability the interface with NCBI to retrieve sequences. The ability of a user to add genes or taxonomy using NCBI's nucleotide and taxonomy databases respectively would make the tool more diverse and useful in its input. Luckily, there is a Java ABI available for interfacing with Entrez that could likely be implemented in EMU-Phy with relative ease in the future.

Critically, EMU-Phy does not have any modules developed for it that allow for the use of protein sequences. Undoubtedly, if the tool is allowed to mature, implementations for protein based phylogeny methods would need to be realized. There are several candidates that may allow for protein trees in EMU-Phy, such as RAxML or PhyML.

Ultimately, though the base program is fully functional, the lack luster performance of modules presented here prevents EMU-Phy from proceeding out of development. However, the program itself reasonably achieves the scope and principles laid out in its software design goals. The infrastructure was designed with extensibility in mind. Additionally, the filesystem is simple enough for advanced users to script large datasets, such as the full primate dataset, without using the tool to add data, while retaining a simple user interface for the uninitiated. Largely, projects such as EMU-Phy are initiated as community projects, either with groups of like-minded individuals or scientific collaborators. If EMU-Phy is ever to grow and be used, the interfaces must be clearer, more robust, and follow an easily explainable API so that a community, rather than a single developer, can construct usable modules for it.

All and all, EMU-Phy provides a solid data management system that falls flat on the analytics aspect of phyloinformatics. It has potential for easing users into a command line environment and provides a number of features that would be appreciated by advanced and novice informaticians. EMU-Phy is an ambitious phyloinformatics project that will ultimately require community support if it is ever realistically going to catch on. However, the small scale implementation of such a tool is a useful exercise for those interested in phyloinformatics and, thus, EMU-Phy has additional potential as an educational tool or exercise.

❖

Chapter 3

❖

Work Toward the Development of Scrawkov-Phy into a Competitive Method for Phylogenetic Inference

J. Nick Fisk

Background:

Scrawkov-Phy: Scrawkov-Phy is an alignment free phylogeny tool that uses a second order Quasi-Hidden Markov model (QHMM) and six genomic features in-lieu of traditional comparison of orthologous sites. As discussed in the Chapter One of this document, Scrawkov-Phy has a number of improvements that needed to be made in order for the algorithm to be fully tested for robustness and potentially ascend to become a competitive or, minimally, useful tool. The foremost of the improvements that merit priority are the parallelization of the algorithm and the introduction of a non-parametric sampling methodology to allow for a metric of confidence for particular nodes, as well as more meaningful comparisons to other trees.

Parallelism and Scrawkov-Phy: Parallel computing is the truly concurrent execution of multiple commands on a computing system. With the exception of abstract cores like those used in hyperthreading, in general, each core of a processor may execute a single command at a time. These cores can be virtual, physical, or even part of a GPU. Parallel programming takes advantage of the existence of several cores in a system to execute code more quickly. The design of parallel programs requires careful consideration of the problem. The minimization of context switches, effective implementation of instruction lookaheads, and efficient use of cores are just a few examples of design aspects which merit attention in the creation of parallel programs. However, not every problem lends itself to parallel approaches. There are a number of characteristics a problem or algorithm should display if a parallel approach is to be considered. If the problem requires largely serial code to be executed often, the overhead of threading the operations or writing the code itself may not be worth the return. Parallel computing has become a necessity for handling big data, like that caused by the genomic revolution in biology. Many problems that are infeasible with serial computing suddenly become plausible or even easy with well-designed parallel implementations. Luckily, Scrawkov-Phy and phylogeny reconstruction itself lend themselves well to parallel solutions.

The idea of parallelizing code in phylogenetic reconstruction is far from novel. Any number of methods use parallel computing in at least some aspect of their algorithm. Version 2 of BEAST, which infers phylogenies based on Bayesian statistics, has become increasingly popular due to its highly parallelized approach resulting in phenomenal speedup in a robust methodology that would otherwise take large sums of time. Notably, the Bayesian estimation of trees is traditionally and almost exclusively carried out by Markov chain Monte Carlo (MCMC) methods, such as the Metropolis coupled MCMC [$(MC)^3$].[107] BEAST and MrBayes[108], then, take advantage of properties of the problem and their mathematical approach to the solution to implement impressive parallel solutions to phylogenetic inference. Many of these properties are shared by Scrawkov-Phy, which uses a QHMM and shares the Markov namesake with MCMCs.

A model or system demonstrates the Markov property if the probability distribution of future states are dependent only upon the state currently being observed. MCMCs, much like the coalescent, are popular in population genetics and were highly adaptable for phylogenetic analysis. Similarly, QHMMs display this Markov property of timelessness. This property is key for the parallelization of Scrawkov-Phy. Since the transition probabilities used in the QHMM do not rely on any previous information about prior states, the calculation of log probabilities can be threaded and solved in subsets before ultimately being incorporated into the Markov Chain Composite Index (MCCI) used in the phylogenetic reconstruction. Timelessness and independence of common computing resources make this calculation a desirable target for parallelization. There are other aspects of Scrawkov-Phy that lend themselves to parallelization as well.

The construction of species trees in Scrawkov-Phy is dependent upon the calculation of individual, constituent gene trees. However, the construction of these trees are independent of each other. Thus, the calculations leading to the construction of each individual gene tree can also be run in parallel. Though all trees would need to be calculated for the species tree to run, the necessary sub-problems (e.g. the gene trees) could be run in parallel, rather than serially. Moreover, 5 of the 6 features (the exception being GC content) lend themselves to being parallelized as well. The DNA sequence can be fragmented into pieces and the measured properties can be measured independently of other features. While breaking each of these 5 calculations into several sub-problems may cause unnecessary overhead in short sequences, it

may be invaluable in increasing the throughput of whole genome data. Furthermore, each of the 6 feature measurements are independent from each other in the current implementation. This suggests that, minimally, each of the 6 features can be measured concurrently, even if the measurements are not broken up for parallelization. The other improvement discussed in the present, non-parametric resampling, would also be a great target for parallelization as each subsample can be analyzed independently of each other.

Bootstrapping, Jackknifing, and Scrawkov-Phy: Non-parametric resampling methods are cornerstone methods for constructing and evaluating phylogenetic trees. While parametric methods exist, they have yet to have the support of the community and the comparative efficacy of each of the methods is a topic of active debate. Of the non-parametric resampling methods, bootstrapping and jackknifing are the well-established methods commonly employed for consensus tree generation. Bootstrapping and jackknifing are extremely similar in design and execution, though they vary in philosophical and statistical justifications. Simply put, the difference between the two is simply that bootstrapping methods allow for replacement while jackknifing methods do not. For the purpose of this paper, only bootstrapping will be considered, as it is more popular among competitive phylogenetic inference programs, though many of the same principle undoubtedly apply to both.

Bootstrapping, in the context of phylogenetic analysis, is used to test the reliability of a phylogenetic tree. Originally described by Felenstein[109], this bootstrap test has become widespread in its application and some software will even collapse groupings into polytomies if the bootstrap value is unsatisfactory. Bootstrapping is performed by sampling some length of nucleotides or amino acids from every sequence and performing the inference analysis with at these samplings some number of times. While the implementation of the sampling is method dependent (i.e. selection of values from a matrix, etc.), the result is typically the same: A large quantity of trees are generated from the sampling sequences. The topology of each of these trees is then compared to the topology of the original tree. A percent agreement of the topology of the original tree to all the sampled trees is calculated for each interior branch and displayed. While the exact percentage for significance is oft debated, the higher the percentage agreement, the more confidence can be had in that interior branch. Due to this method being one of few extant methods for tree comparison, having widespread implementation, and generally being positively

received, any method aiming to compete with the current methods must have nonparametric sampling as a requisite.

Challenges in Bootstrapping and Parallelization: As has been verbosely established, if a method is to become competitive or widely applied in the field of phylogenetics as it exists today, the said method must be parallelizable and either contain a method or support a method that allows for tree evaluation through non-parametric sampling. However, there are notable challenges that come with incorporating these features into Scrawkov-Phy.

As previously stated, typically the most effective parallel programs are those who have been implemented with parallel design principles ab initio. The adaptation of existing programs to support parallelization often requires an entirely new development period as much of the code or subroutines are not of a framework well suited for parallelization. One strategy for the implementation of parallel code in an existing software is to implement parallel computation where it is naturally facilitated already within the framework of the code. Then, when a new-full blown release of the program is released, a proper parallel implementation can be included. However, the speedup gained by such an approach is limited and does not scale as well. Instead, it reduces the overhead of creating new code, creating a compromise between developing time and run time. Scrawkov-Phy was not designed with a parallel implementation in mind. Thus, the internal data structures may limit the parallel implementation, despite its mathematical model being so highly conducive to it. The lack of a parallel framework is not the only challenge that Scrawkov-Phy faces if it is to become more competitive.

Bootstrapping was implemented and described over 30 years ago and has held as a method for traditional tree-building since[109]. The approach is robust and has been shown to be relevant to phylogenetic inference, despite critiques[110]. However, it is largely unexplored if this approach would hold the same significance when applied to alignment free systems which do not rely on the direct comparison of orthologous sites. The underlying biological principles are different, though the true signal each method attempts to measure should be the same. The discussion of the statistical relevance of parametric versus non-parametric methods and their efficacy when used on alignment free approaches is beyond the scope of the present. However, it is important to note that the branches with high support by bootstrapping Scrawkov-Phy do not necessarily imply the correctness of the recovered tree, nor does low-branch support necessarily

disqualify the tree. Until the mathematical models are more fully explored and linked to the biology, limited conclusions should be drawn. However, the need to evaluate the efficacy of bootstrapping on alignment free models necessitates the implementation thereof—a method that does not exist cannot be tested.

Other Improvements to Scrawkov-Phy In addition to paralleization and bootstrapping, there are a number of other improvements to Scrawkov-Phy that would vastly improve its utility, many of which are discussed in Chapter One of the present. The ability to write the MCCI as a PHYLIP formatted distance matrix would allow for more sophisticated tree-building methods to be used, subverting the issues caused by UPGMA. Likewise, incorporating the basic Neighor-Joining[34] method in Scrawkov-Phy will allow for better characterization of the method and better facilitate its comparison to other methods. Lastly, the option to alter the k-mer size used in the QHMM to a larger length to better account for k-mer homoplasy is introduced.

Methods:

Due to the design aim of Scrawkov-Phy being system independent and completely self-contained, development of the parallel version of Scrawkov-Phy was continued in Java. To enhance the rate in which a parallel version could be developed, an external library is used. The Parallel Java 2 library, written by Alan Kaminsky[111], is an entirely Java library that facilitates the development of parallel code in Java and is available under a GNU GPL v.3 license. It is middleware that significantly reduces the overhead involved in parallel programming as it is available in Java. Likewise, the bootstrapping tree functionality is being designed as a method within the main Scrawkov-Phy Java file, taking advantage of the in-place infrastructure to generate the trees. Both functionalities are partially completed.

The neighbor-joining tree method implemented was tested with the small primate NADH dataset and with the larger, whole genome filovirus dataset. The ability to use k-mers of any length were tested using the whole genome filovirus dataset. K-mers used were of size 12 and 20. These trees were all constructed using the aforementioned neighbor-joining implementation.

Results:

Parallelization: The computation of the codon bias feature as determined by Scrawkov-Phy was completed. The function was tested and the improved speedup of the function was tested iteratively. The function was run with 4 cores on a single node on the RIT Research Computing Cluster. The results are shown graphically in Figure 1, below.

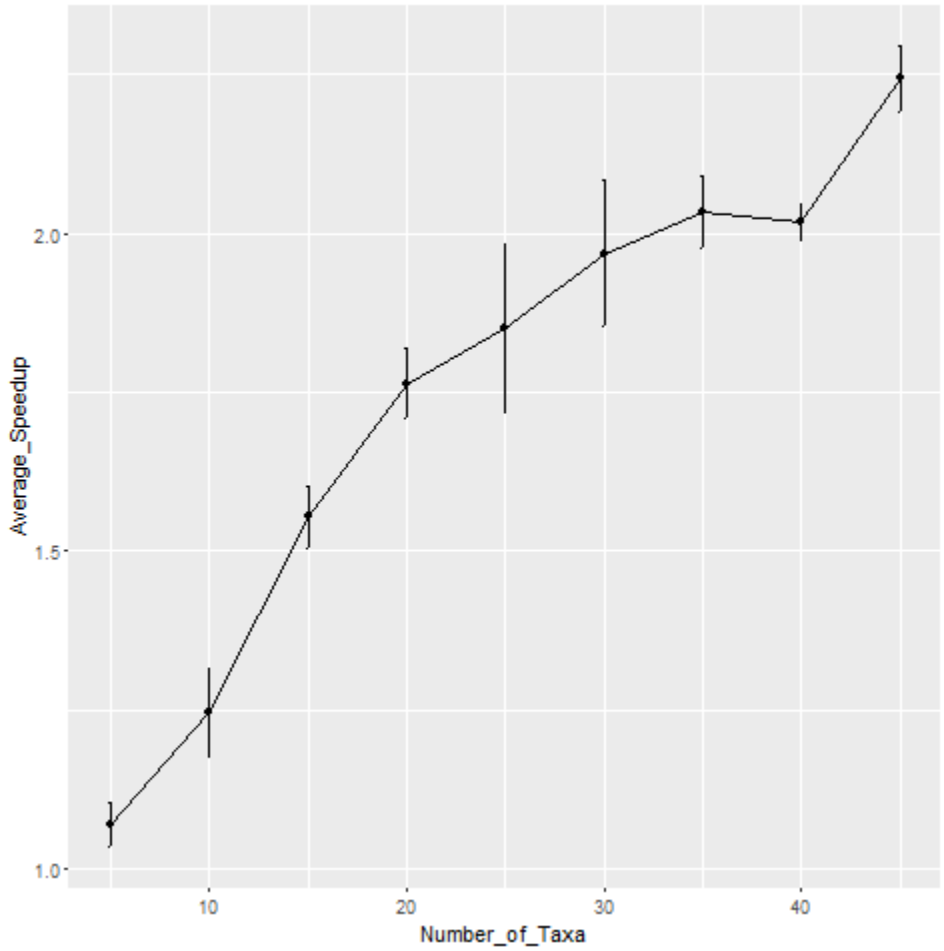


Figure 1: The average speedup observed in the codon bias computation as a function of increasing number of taxa. All computation was performed on the BRCA2 gene sequences from the primate dataset. 30 iterations were performed.

In addition to the parallelization of the codon bias computation, the creation of species trees was parallelized by running the constituent gene tree calls equally among available cores. The species

tree method was run with 4 cores on a single node on the RIT Research Computing Cluster. The results are shown graphically in Figure 2, below.

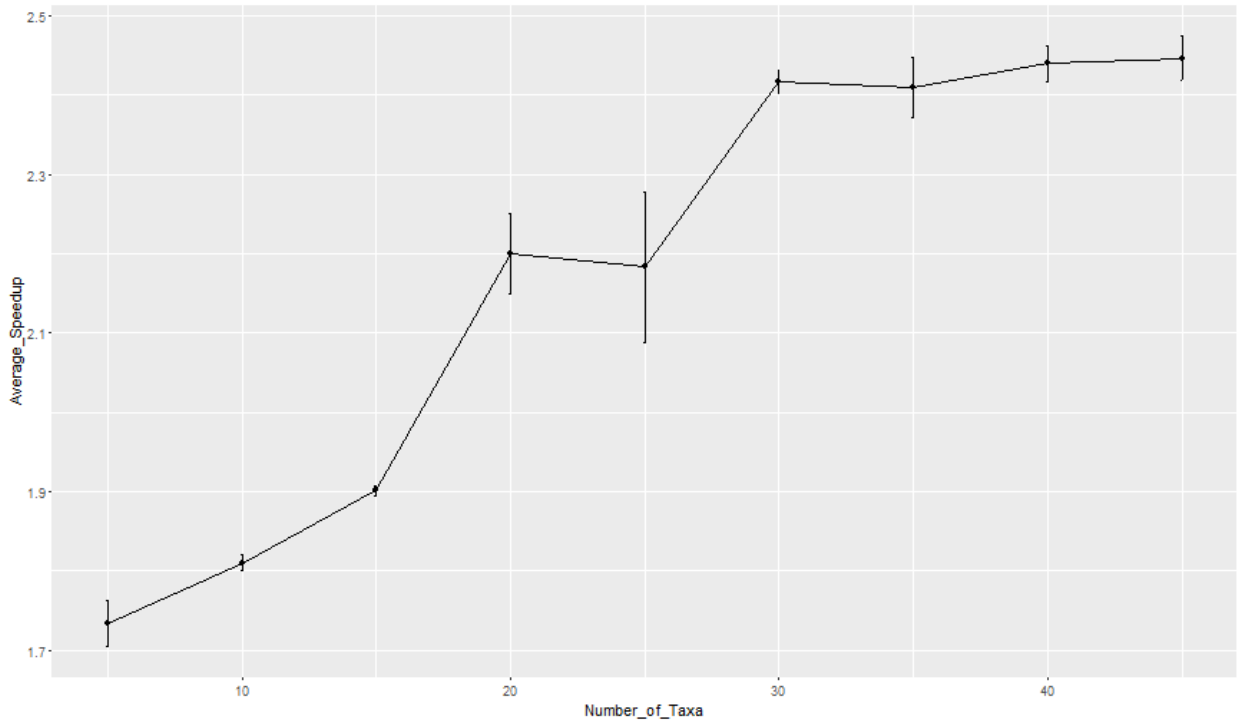


Figure 2: The average speed-up of the construction of a species tree over an increasing number of taxa. The same 4 genes from the aforementioned primate study were used in each iteration: ABCA, BRCA2, CNR1, and MBD5. 30 iterations were performed.

Bootstrapping: Scrawkov-Phy now has the ability to create bootstrapping trees from gene tree datasets. It uses the infrastructure in place, as well as some new command line arguments to do so. The generation of the data subsets is performed as a call to the 'gene' functionality with an additional argument:

```
java ScrawkovPHY gene nameOfInput.fasta -o treeOutputFilename.nwk --bootSeq true –numBoots 100
```

The above command will generate a gene tree with the data to the desired output filename. Additionally, with the bootSeq argument, a bootstrapping dataset is created and saved to a temporary directory. The numBoots argument allows control of the number of iterations for

bootstrapping. To complete the creation of the bootstrapping trees, a second command must be executed, this time using the 'species' functionality:

```
java ScrawkovPHY species --folder tmp --outputAll true --bootSeq true
```

The above command will generate gene trees for each of the bootstrapping samples contained in the tmp directory. The outputAll argument, which is also available in the regular function of the species tree command, indicates to the program that all constituent gene trees of the species tree should be saved to file. In the case of this species tree command, bootSeq instructs the program to not construct a final species tree, as it would have little practical application here, though the functionality is preserved. The result are a large number of bootstrapped trees ready for comparison to the tree that was also generated in the first command.

The programmatic determination of the percent bootstrapping support is still in development. Newick formatted files may contain information about bootstrapping confidence and are also the proper format needed for such a comparison. However, the internal data structure used in the construction of the constituent trees may provide a faster alternative than reading the files in and parsing them back out. A sample set of 10 bootstrapping trees for the NADH dehydrogenase gene in a subset of primates was constructed and manually examined for efficacy. 9 of the 10 bootstrapped trees matched the topology of the gene tree precisely. The other tree conflicted only in the resolution of humans, chimps, and gorillas. It inverted the placement of gorillas and chimps.

Neighbor-Joining and Variable Length K-mer: The neighbor joining tree generated by Scrawkov-Phy for the small primate NADH gene is shown below in Figure 3.

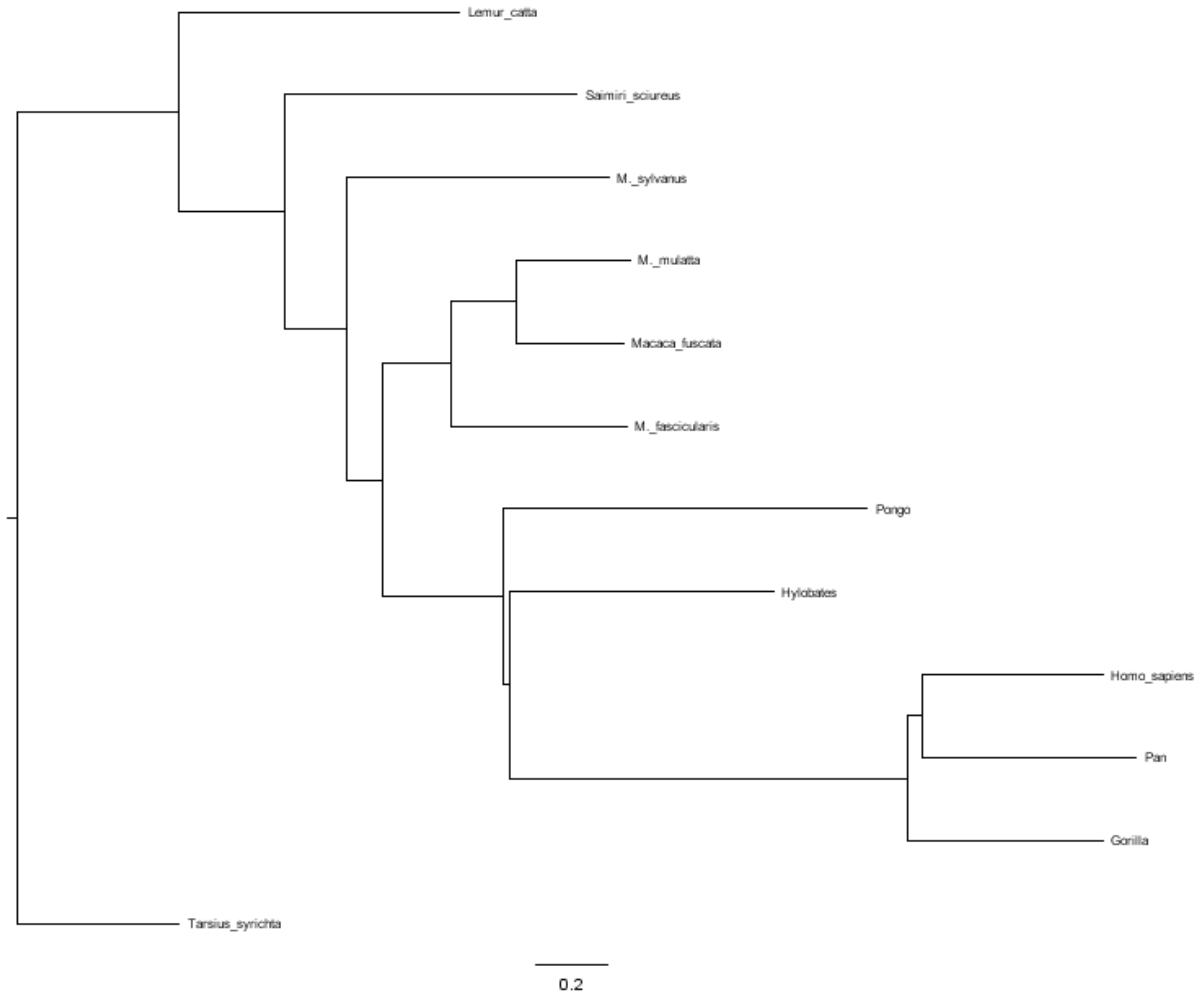
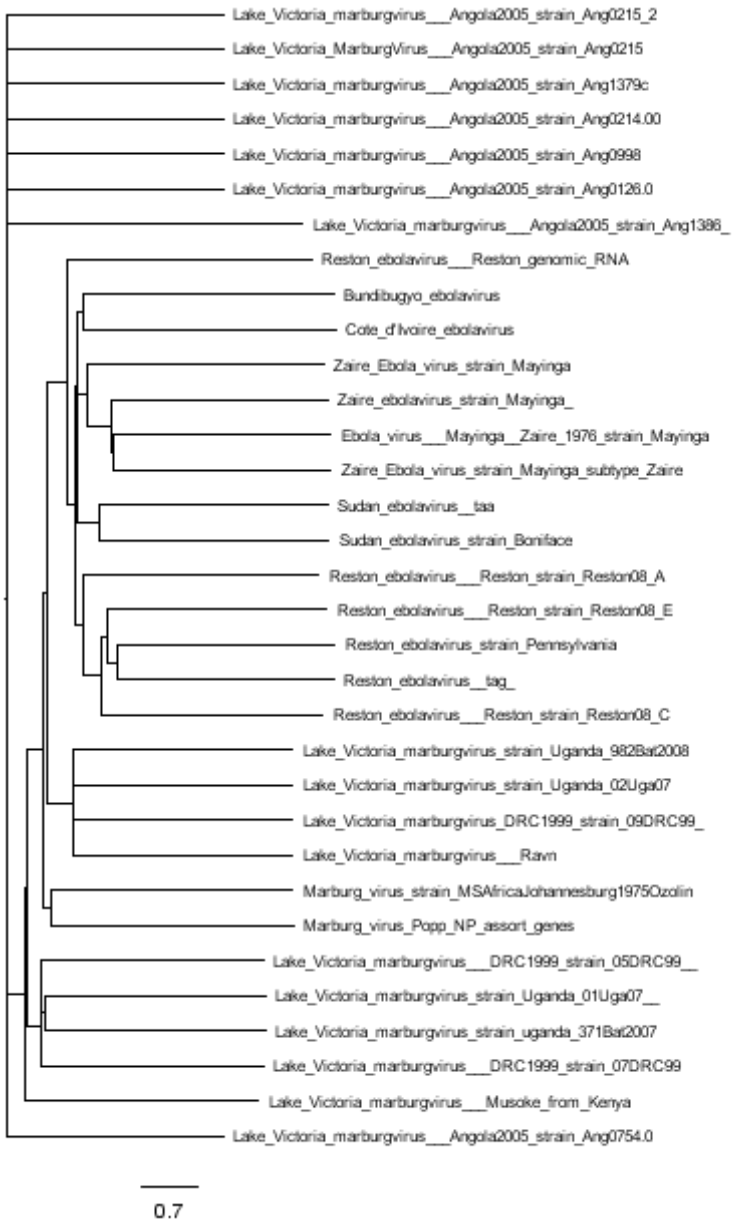


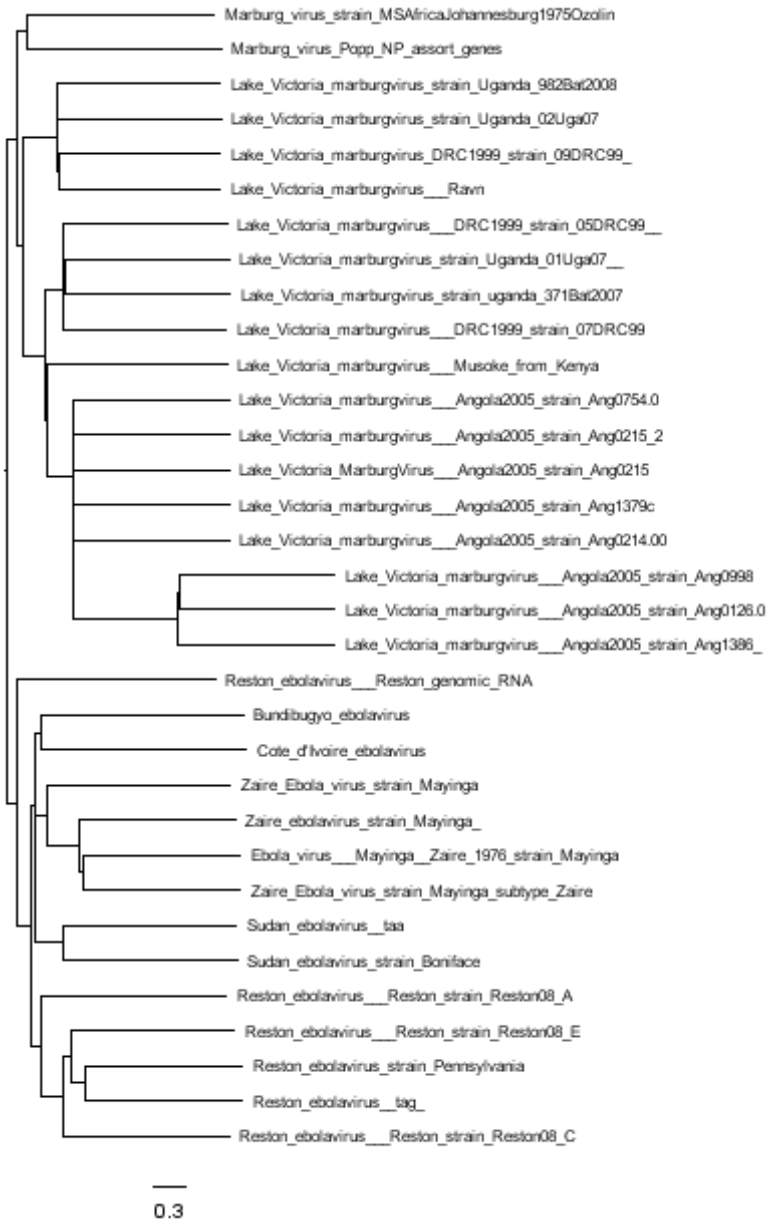
Figure 3: The result of applying the in-place neighbor joining algorithm to a small NADH dataset in select primates. Note that the tree topology is the same as that obtained in Chapter 1, but the branches are now of a variable length.

Additionally, the program can now write out the distance to a PHYLIP formatted distance matrix file, either mirrored or just the diagonal, as shown in Figure 4.

Figure 4: Scrawkov-Phy is capable of generating and writing PHYLIP style distance formatted matrices, be it fully mirrored (top) or along the diagonal (bottom). The file was capable of being read and used by PHYLIP's neighbor program. The flag '—diagMat' controls which is outputted.

To test the k-mer method, which was implemented to address the possible issue of homoplasy, a neighbor joining tree was produced for the original method, the method using a 12-mer, and using a 20-mer. All three trees are visualized in Figure XX below.


Lake_Victoria_marburgvirus___Angola2005_strain_Ang0215_2
Lake_Victoria_MarburgVirus___Angola2005_strain_Ang0215
Lake_Victoria_marburgvirus___Angola2005_strain_Ang1379c
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0214.00
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0998
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0126.0
Lake_Victoria_marburgvirus___Angola2005_strain_Ang1386_
Reston_ebolavirus___Reston_genomic_RNA
Bundibugyo_ebolavirus
Cote_d'Ivoire_ebolavirus
Zaire_Ebola_virus_strain_Mayinga
Zaire_ebolavirus_strain_Mayinga_
Ebola_virus___Mayinga__Zaire_1976_strain_Mayinga
Zaire_Ebola_virus_strain_Mayinga_subtype_Zaire
Sudan_ebolavirus__tas
Sudan_ebolavirus_strain_Boniface
Reston_ebolavirus___Reston_strain_Reston08_A
Reston_ebolavirus___Reston_strain_Reston08_E
Reston_ebolavirus_strain_Pennsylvania
Reston_ebolavirus__tag_
Reston_ebolavirus___Reston_strain_Reston08_C
Lake_Victoria_marburgvirus_strain_Uganda_982Bat2008
Lake_Victoria_marburgvirus_strain_Uganda_02Uga07
Lake_Victoria_marburgvirus_DRC1999_strain_09DRC99_
Lake_Victoria_marburgvirus___Ravn
Marburg_virus_strain_MSAfricaJohannesburg1975Ozolin
Marburg_virus_Popp_NP_assort_genes
Lake_Victoria_marburgvirus___DRC1999_strain_05DRC99__
Lake_Victoria_marburgvirus_strain_Uganda_01Uga07__
Lake_Victoria_marburgvirus_strain_uganda_371Bat2007
Lake_Victoria_marburgvirus___DRC1999_strain_07DRC99
Lake_Victoria_marburgvirus___Musoke_from_Kenya
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0754.0
0.7


Marburg_virus_strain_MSAfricaJohannesburg1975Ozolin
Marburg_virus_Popp_NP_assort_genes
Lake_Victoria_marburgvirus_strain_Uganda_982Bat2008
Lake_Victoria_marburgvirus_strain_Uganda_02Uga07
Lake_Victoria_marburgvirus_DRC1999_strain_09DRC99_
Lake_Victoria_marburgvirus___Ravn
Lake_Victoria_marburgvirus___DRC1999_strain_05DRC99__
Lake_Victoria_marburgvirus_strain_Uganda_01Uga07__
Lake_Victoria_marburgvirus_strain_uganda_371Bat2007
Lake_Victoria_marburgvirus___DRC1999_strain_07DRC99
Lake_Victoria_marburgvirus___Musoke_from_Kenya
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0754.0
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0215_2
Lake_Victoria_MarburgVirus___Angola2005_strain_Ang0215
Lake_Victoria_marburgvirus___Angola2005_strain_Ang1379c
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0214.00
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0998
Lake_Victoria_marburgvirus___Angola2005_strain_Ang0126.0
Lake_Victoria_marburgvirus___Angola2005_strain_Ang1386_
Reston_ebolavirus___Reston_genomic_RNA
Bundibugyo_ebolavirus
Cote_d'Ivoire_ebolavirus
Zaire_Ebola_virus_strain_Mayinga
Zaire_ebolavirus_strain_Mayinga_
Ebola_virus___Mayinga__Zaire_1976_strain_Mayinga
Zaire_Ebola_virus_strain_Mayinga_subtype_Zaire
Sudan_ebolavirus__taa
Sudan_ebolavirus_strain_Boniface
Reston_ebolavirus___Reston_strain_Reston08_A
Reston_ebolavirus___Reston_strain_Reston08_E
Reston_ebolavirus_strain_Pennsylvania
Reston_ebolavirus__tag_
Reston_ebolavirus___Reston_strain_Reston08_C
0.3

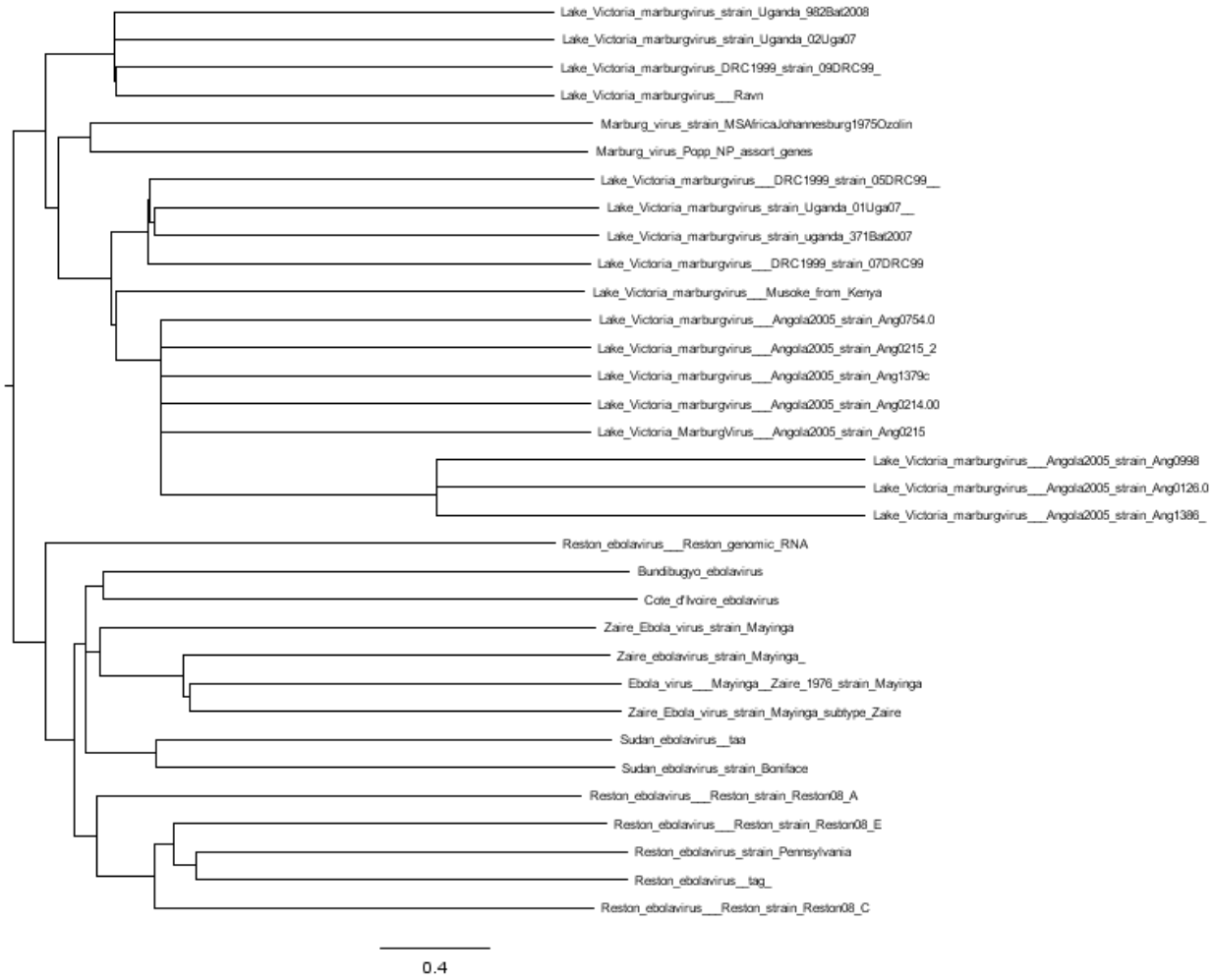


Figure 5: Three NJ trees reconstructed from whole Filovirus genomes. The original Scrawkov-Phy (top) did not recover quite as nice as a topology as the 12-mer (middle) and 20-mer (bottom) methods. As described by Fan et al[12], there appears to be a delicate balance in the selection of K-mer size, as the 12mer seemingly recovers a better tree than the 20-mer method. Of course, this conjecture needs validation by bootstrapping.

Discussion:

The continued development of Scrawkov-Phy is underway and tangible progress has been made on both the evaluation of tree quality via non-parametric sampling and in the parallelization of the code. There are still a number of principles that must be described in order to consider the method robust, especially concerning the efficacy of using sampling to evaluate tree quality. On one hand, the bootstrap is a standard used widely in phylogenetics. On the other hand, QHMMs are a basic form of machine learning which construct better and better models the more data is present. Individual genes can be short and the further generation of sequence subsets may severely limit the algorithm's ability to learn. Using bootstrapping may be more appropriate with large, genome size data so that features may still be properly characterized. While principles such as k-fold validation exist in machine learning, further research will be necessary to determine if and when approaches should be used and if they would be comparable to the bootstrap as it is used in other applications.

The result of the prototyped bootstrapping testing set was interesting. In this case, the gene tree constructed with Scrawkov-Phy (which is also supported by other methods, as shown in Chapter 1 of the present document) was in almost total agreement with its bootstrapping trees. The placement of Pan versus Gorilla with humans is not uncommon among particular datasets, which made the resolution of the primate tree difficult for some time. This near-miss is notable because it has been previously described and has biological relevance.

The parallelization of the codon bias method resulted in a method that was softly parallel. The return on cores was not one to one, even as the number of taxa increased. This is an expected result, as the method was not written with parallel design principles in mind. Additional cores may improve the return, but it is notable that the number of taxa steadily and significantly improved the speedup, perhaps implying that parallelization may be best utilized for datasets with large number of taxa. The species tree method, on the other hand, did not return on the number of cores as well as was expected. Since each gene tree is constructed independently and due to the relative simplicity of the method used to construct the final species tree from gene trees, the speedup was expected to be closer to the number of cores used. This method had a better speedup than the codon bias function, but it was still not nearly as scalable to cores as is

desirable. Interestingly, the same trend was observed in the species tree and codon bias methods, where the number of taxa results in a better overall speedup, again implying that parallelization may best suited to trees with large numbers of taxa.

In the continued development of the bootstrapping function, an opportunity to repurpose old code has been identified. In essence, the hill-climbing algorithm (described in chapter one of the present) compares the topology of generated trees to well resolved tree to generate parameter weights for the algorithmic determination of phylogeny. This comparison of topology is crucial for the next step in implementing a bootstrapping method and would require only minor alterations to achieve the desired result. Likewise, jackknifing can be implemented simply by changing the way the current method subsamples sequences.

All in all, the prototype methods for parallelism and bootstrapping were modestly successful. This modest success is an indicator that development of the methods should continue. If the example parallelization had been arduous to develop or had minimal return, that result would have been an indicator to either abandon the prospect or rework the framework to enable parallelization. Likewise, if there was not a way to easily utilize the existing functions available in Scrawkov-Phy, the bootstrapping tree generation may have been better off as a separate program. Similarly, if the programmatic evaluation of the bootstrapping trees does not fit within the framework provided by Scrawkov-Phy, it may be more prudent to include the function as a separate program to best conform to best practices in code design. It may also be wise to narrow the taxonomic focus of the algorithm, perhaps focusing more on prokaryotes, as the demand and approach may better apply.

Neighbor-joining was successfully implemented and incorporated into Scrawkov-Phy. This incorporation was necessary as the UPGMA method is generally not accepted as being a rigourous tree-building method by many due to its reliance on mid-point rooting. Such rooting assumes ultrametric evolution, which is not a biologically rigourous assumption and yields sub-par and inconsistent results[112].While neighbor-joining has its faults, it is a standard for distance approaches to phylogenetic reconstruction. Though algorithm can recover the same topology, as shown in the primate dataset. Notably, implementing the NJ also allows for the collapse of nodes into a polytomy when distance is short and no bootstrapping test has been performed occurred. It is interesting to note that, for the filovirus data set, the default Scrawkov-Phy did not produce a

high-quality NJ-tree. In that instance, the UPGMA tree displayed in Chapter 1 recovered a topology more closely representing the natural history of the family. This is remedied, however, by increasing the k-mer length. The 12-mer method recovered correct clustering for all the families—a notable improvement from the UPGMA tree in Chapter 1.

There are still many things to improve upon if Scrawkov-Phy if it is ever to be a competitive tool for phylogenetic analysis, but the modest successes presented here show that such improvements are possible. Neighbor joining and the ability to use any length k-mer were completed in their entirety and incorporated into the program. The provisional success of the bootstrapping and parallel components suggest that further improvements of this nature will be compatible with the algorithm and that there is merit in performing such improvements.

Closing Remarks

This work ties together, at some level, many facets of the field of phylogenetics as it exists today. Within it are attempts at the creation of new methods and data management techniques, but also exploration of the traditional methods and the biology on which they are based. Many of the methods used in phylogenetic analysis today were described decades ago and are still used today, with computational improvements. The introduction of big data has led to paradigm shifts and novel methods for managing phylogenetic data and analyzing it when traditional methods fail. In these ways, phylogenetics is both a very old field and a very new one. There are and will continue to be disagreements in the field about proper methods, the limitations of inference, and the interpretation of results. However, this is also a field that contains a set of some of the most unique challenges in addressing some of the most interesting questions about an unobservable past. While phylogenetic inference is just that—inference—the foundational nature of the field, the questions it poses, and its application in such a wide range of disciplines promise that the field will continue to mature, modernize, and motivate for generations.

References:

1. Shanahan, T. Phylogenetic inertia and Darwin's higher law. *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences* **42,** 60–68 (2011).
2. Velasco, J. D. Philosophy and Phylogenetics. *Philosophy Compass* **8,** 990–998 (2013).
3. Kluge, A. G. Explanation and Falsification in Phylogenetic Inference: Exercises in Popperian Philosophy. *Acta Biotheor* **57,** 171–186 (2009).
4. Gilmour, J. S. L. Review of Evolution: The Modern Synthesis. *Philosophy* **19,** 166–170 (1944).
5. Platnick, N. I. Philosophy and the Transformation of Cladistics. *Syst Biol* **28,** 537–546 (1979).
6. Emerson, B. C., Alvarado-Serrano, D. F. & Hickerson, M. J. Model misspecification confounds the estimation of rates and exaggerates their time dependency. *Mol Ecol* **24,** 6013–6020 (2015).
7. Sullivan, J. & Joyce, P. Model Selection in Phylogenetics. *Annual Review of Ecology, Evolution, and Systematics* **36,** 445–466 (2005).
8. Baele, G., Lemey, P. & Suchard, M. A. Genealogical working distributions for Bayesian model testing with phylogenetic uncertainty. *Syst Biol* syv083 (2015). doi:10.1093/sysbio/syv083
9. Metzker, M. L. Sequencing technologies [mdash] the next generation. *Nat Rev Genet* **11,** 31–46 (2010).
10. Stevens, P. F. Homology and Phylogeny: Morphology and Systematics. *Systematic Botany* **9,** 395–409 (1984).
11. Garland, T., Harvey, P. H. & Ives, A. R. Procedures for the Analysis of Comparative Data Using Phylogenetically Independent Contrasts. *Syst Biol* **41,** 18–32 (1992).
12. Fan, H., Ives, A. R., Surget-Groba, Y. & Cannon, C. H. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC Genomics* **16,** 522 (2015).

13. Jantschi, L., Bolboaca, S. D. & Sestras, R. E. Hard Problems in Gene Sequence Analysis: Classical Approaches and Suitability of Genetic Algorithms. *Biotechnology & Biotechnological Equipment* **23,** 1275–1280 (2009).
14. Dabhade, K. R. Big Data An Overview. *International Journal of Technology Enhancements and Emerging Engineering Research* **3,** 255–257 (2014).
15. Wu, X., Zhu, X., Wu, G. Q. & Ding, W. Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering* **26,** 97–107 (2014).
16. Labrinidis, A. & Jagadish, H. V. Challenges and Opportunities with Big Data. *Proc. VLDB Endow.* **5,** 2032–2033 (2012).
17. Begoli, E. & Horey, J. Design Principles for Effective Knowledge Discovery from Big Data. in *2012 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA) and European Conference on Software Architecture (ECSA)* 215–218 (2012). doi:10.1109/WICSA-ECSA.212.32
18. Marx, V. Biology: The big challenges of big data. *Nature* **498,** 255–260 (2013).
19. Congressional Justification FY2015. Available at: https://www.nlm.nih.gov/about/2015CJ.html. (Accessed: 29th March 2016)
20. Statistics < About the European Nucleotide Archive < European Nucleotide Archive < EMBL-EBI. Available at: http://www.ebi.ac.uk/ena/about/statistics. (Accessed: 29th March 2016)
21. Big Data: Issues and Challenges Moving Forward. Available at: http://www.academia.edu/10967116/Big_Data_Issues_and_Challenges_Moving_Forward. (Accessed: 25th April 2016)
22. Cook, S. A. An Overview of Computational Complexity. *Commun. ACM* **26,** 400–408 (1983).
23. Papadimitriou, C. H. in *Encyclopedia of Computer Science* 260–265 (John Wiley and Sons Ltd.).
24. Day, W. H. E. Computational complexity of inferring phylogenies from dissimilarity matrices. *Bltn Mathcal Biology* **49,** 461–467 (1987).

25. Day, W. H. E. Computationally difficult parsimony problems in phylogenetic systematics. *Journal of Theoretical Biology* **103,** 429–438 (1983).
26. Day, W. H. E. & Sankoff, D. Computational Complexity of Inferring Phylogenies by Compatibility. *Syst Biol* **35,** 224–229 (1986).
27. Day, W. H. E., Johnson, D. S. & Sankoff, D. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences* **81,** 33–42 (1986).
28. Foulds, L. R. & Graham, R. L. The steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics* **3,** 43–49 (1982).
29. Wang, L. & Jiang, T. On the Complexity of Multiple Sequence Alignment. *Journal of Computational Biology* **1,** 337–348 (1994).
30. Katoh, K. & Toh, H. Recent developments in the MAFFT multiple sequence alignment program. *Brief Bioinform* **9,** 286–298 (2008).
31. Edgar, R. C. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics* **5,** 113 (2004).
32. Chan, C. X. & Ragan, M. A. Next-generation phylogenomics. *Biology Direct* **8,** 3 (2013).
33. Felsenstein, J. Cases in which Parsimony or Compatibility Methods will be Positively Misleading. *Syst Biol* **27,** 401–410 (1978).
34. Saitou, N. & Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol* **4,** 406–425 (1987).
35. Bruno, W. J., Socci, N. D. & Halpern, A. L. Weighted Neighbor Joining: A Likelihood-Based Approach to Distance-Based Phylogeny Reconstruction. *Mol Biol Evol* **17,** 189–197 (2000).
36. Yang, Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: Approximate methods. *J Mol Evol* **39,** 306–314 (1994).

37. Guindon, S. & Gascuel, O. A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood. *Syst Biol* **52,** 696–704 (2003).

38. Yang, Z. & Rannala, B. Bayesian phylogenetic inference using DNA sequences: a Markov Chain Monte Carlo Method. *Mol Biol Evol* **14,** 717–724 (1997).

39. Huelsenbeck, J. P., Ronquist, F., Nielsen, R. & Bollback, J. P. Bayesian Inference of Phylogeny and Its Impact on Evolutionary Biology. *Science* **294,** 2310–2314 (2001).

40. Susana Vinga, J. A. Alignment-free sequence comparison - A review. *Bioinformatics (Oxford, England)* **19,** 513–23 (2003).

41. Thompson, J. D., Higgins, D. G. & Gibson, T. J. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.* **22,** 4673–4680 (1994).

42. Prlić, A., Domingues, F. S. & Sippl, M. J. Structure-derived substitution matrices for alignment of distantly related sequences. *Protein Eng.* **13,** 545–550 (2000).

43. Earl, D. *et al.* Alignathon: A competitive assessment of whole genome alignment methods. *Genome Res.* gr.174920.114 (2014). doi:10.1101/gr.174920.114

44. B.A, E. R. L. II.—On the use of the term homology in modern zoology, and the distinction between homogenetic and homoplastic agreements. *Annals and Magazine of Natural History* **6,** 34–43 (1870).

45. Drummond, A. J., Rambaut, A., Shapiro, B. & Pybus, O. G. Bayesian Coalescent Inference of Past Population Dynamics from Molecular Sequences. *Mol Biol Evol* **22,** 1185–1192 (2005).

46. Liu, L., Yu, L., Kubatko, L., Pearl, D. K. & Edwards, S. V. Coalescent methods for estimating phylogenetic trees. *Molecular Phylogenetics and Evolution* **53,** 320–328 (2009).

47. Crandall, K. A. & Templeton, A. R. Empirical tests of some predictions from coalescent theory with applications to intraspecific phylogeny reconstruction. *Genetics* **134,** 959–969 (1993).

48. Drummond, A. J. & Rambaut, A. BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology* **7,** 214 (2007).

49. Bininda-Emonds, O. R. P. The evolution of supertrees. *Trends in Ecology & Evolution* **19,** 315–322 (2004).

50. de Queiroz, A. & Gatesy, J. The supermatrix approach to systematics. *Trends in Ecology & Evolution* **22,** 34–41 (2007).

51. Kluge, A. G. A Concern for Evidence and a Phylogenetic Hypothesis of Relationships among Epicrates (Boidae, Serpentes). *Syst Biol* **38,** 7–25 (1989).

52. Huelsenbeck, J. P., Bull, J. J. & Cunningham, C. W. Combining data in phylogenetic analysis. *Trends in Ecology & Evolution* **11,** 152–158 (1996).

53. Gatesy, J. & Springer, M. S. Phylogenetic analysis at deep timescales: Unreliable gene trees, bypassed hidden support, and the coalescence/concatalescence conundrum. *Molecular Phylogenetics and Evolution* **80,** 231–266 (2014).

54. Eddy, S. {HMMER: Profile hidden Markov models for biological sequence analysis}. (2001).

55. Blumenthal, R. M. An Extended Markov Property. *Transactions of the American Mathematical Society* **85,** 52–72 (1957).

56. Fine, S., Singer, Y. & Tishby, N. The Hierarchical Hidden Markov Model: Analysis and Applications. *Mach. Learn.* **32,** 41–62 (1998).

57. Kaelbling, L. P., Littman, M. L. & Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* **101,** 99–134 (1998).

58. Collier, N., Nobata, C. & Tsujii, J. Extracting the Names of Genes and Gene Products with a Hidden Markov Model. in *Proceedings of the 18th Conference on Computational Linguistics - Volume 1* 201–207 (Association for Computational Linguistics, 2000). doi:10.3115/990820.990850

59. Felsenstein, J. & Churchill, G. A. A Hidden Markov Model approach to variation among sites in rate of evolution. *Mol Biol Evol* **13,** 93–104 (1996).

60. Eddy, S. R. Profile hidden Markov models. *Bioinformatics* **14,** 755–763 (1998).

61. Krogh, A., Brown, M., Mian, I. S., Sjölander, K. & Haussler, D. Hidden Markov Models in Computational Biology. *Journal of Molecular Biology* **235,** 1501–1531 (1994).

62. Eddy, S. R. Hidden Markov models. *Current Opinion in Structural Biology* **6,** 361–365 (1996).

63. Krogh, A., Larsson, B., von Heijne, G. & Sonnhammer, E. L. L. Predicting transmembrane protein topology with a hidden markov model: application to complete genomes1. *Journal of Molecular Biology* **305,** 567–580 (2001).

64. Lusseau, D. Effects of Tour Boats on the Behavior of Bottlenose Dolphins: Using Markov Chains to Model Anthropogenic Impacts. *Conservation Biology* **17,** 1785–1793 (2003).

65. Grundy, W. N., Bailey, T. L., Elkan, C. P. & Baker, M. E. meta-MEME: Motif-based hidden Markov models of protein families. *Comput Appl Biosci* **13,** 397–406 (1997).

66. Hey, J. & Nielsen, R. Integration within the Felsenstein equation for improved Markov chain Monte Carlo methods in population genetics. *PNAS* **104,** 2785–2790 (2007).

67. Salzberg, S. L., Pertea, M., Delcher, A. L., Gardner, M. J. & Tettelin, H. Interpolated Markov Models for Eukaryotic Gene Finding. *Genomics* **59,** 24–31 (1999).

68. Huang, Q., Ge, R., Kakade, S. & Dahleh, M. Minimal Realization Problems for Hidden Markov Models. *IEEE Transactions on Signal Processing* **64,** 1896–1904 (2016).

69. Wu, Z. Quasi-hidden Markov model and its applications in cluster analysis of earthquake catalogs. *J. Geophys. Res.* **116,** B12316 (2011).

70. Šmarda, P. *et al.* Ecological and evolutionary significance of genomic GC content diversity in monocots. *PNAS* **111,** E4096–E4102 (2014).

71. Costantini, M., Cammarano, R. & Bernardi, G. The evolution of isochore patterns in vertebrate genomes. *BMC Genomics* **10,** 146 (2009).

72. Romiguier, J., Ranwez, V., Douzery, E. J. P. & Galtier, N. Contrasting GC-content dynamics across 33 mammalian genomes: Relationship with life-history traits and chromosome sizes. *Genome Res* **20,** 1001–1009 (2010).

73. Lassalle, F. *et al.* GC-Content Evolution in Bacterial Genomes: The Biased Gene Conversion Hypothesis Expands. *PLOS Genet* **11,** e1004941 (2015).

74. Takahashi, M., Kryukov, K. & Saitou, N. Estimation of bacterial species phylogeny through oligonucleotide frequency distances. *Genomics* **93,** 525–533 (2009).

75. Babbitt, G. A., Alawad, M. A., Schulze, K. V. & Hudson, A. O. Synonymous codon bias and functional constraint on GC3-related DNA backbone dynamics in the prokaryotic nucleoid. *Nucl. Acids Res.* **42,** 10915–10926 (2014).

76. Yang, Z., Nielsen, R., Goldman, N. & Pedersen, A.-M. K. Codon-Substitution Models for Heterogeneous Selection Pressure at Amino Acid Sites. *Genetics* **155,** 431–449 (2000).

77. Plotkin, J. B. & Kudla, G. Synonymous but not the same: the causes and consequences of codon bias. *Nat Rev Genet* **12,** 32–42 (2011).

78. Carlini, D. B., Chen, Y. & Stephan, W. The Relationship Between Third-Codon Position Nucleotide Content, Codon Bias, mRNA Secondary Structure and Gene Expression in the Drosophilid Alcohol Dehydrogenase Genes Adh and Adhr. *Genetics* **159,** 623–633 (2001).

79. Hershberg, R. & Petrov, D. A. Selection on Codon Bias. *Annual Review of Genetics* **42,** 287–299 (2008).

80. Sharp, P. M., Emery, L. R. & Zeng, K. Forces that influence the evolution of codon bias. *Philosophical Transactions of the Royal Society of London B: Biological Sciences* **365,** 1203–1212 (2010).

81. Burge, C. & Karlin, S. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.* **268,** 78–94 (1997).

82. Goldman, N. Nucleotide, dinucleotide and trinucleotide frequencies explain patterns observed in chaos game representations of DNA sequences. *Nucl. Acids Res.* **21,** 2487–2491 (1993).

83. Clote, P., Ferré, F., Kranakis, E. & Krizanc, D. Structural RNA has lower folding energy than random RNA of the same dinucleotide frequency. *RNA* **11,** 578–591 (2005).

84. Smith, S. A., Beaulieu, J. M. & Donoghue, M. J. Mega-phylogeny approach for comparative biology: an alternative to supertree and supermatrix approaches. *BMC Evol Biol* **9,** 37 (2009).

85. Rambaut, A. FigTree, version 1.3. 1. *Computer program distributed by the author, website: http://treebioedacuk/software/figtree/[accessed January 4, 2011]* (2009).

86. Sievers, F. *et al.* Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology* **7,** 539–539 (2014).

87. Larkin, M. A. *et al.* Clustal W and Clustal X version 2.0. *Bioinformatics* **23,** 2947–2948 (2007).

88. Kumar, S., Nei, M., Dudley, J. & Tamura, K. MEGA: A biologist-centric software for evolutionary analysis of DNA and protein sequences. *Brief Bioinform* **9,** 299–306 (2008).

89. Barrette, R. W., Xu, L., Rowland, J. M. & McIntosh, M. T. Current perspectives on the phylogeny of Filoviridae. *Infect. Genet. Evol.* **11,** 1514–1519 (2011).

90. Sangster, G., Alström, P., Forsmark, E. & Olsson, U. Multi-locus phylogenetic analysis of Old World chats and flycatchers reveals extensive paraphyly at family, subfamily and genus level (Aves: Muscicapidae). *Molecular Phylogenetics and Evolution* **57,** 380–392 (2010).

91. Yi, H. & Jin, L. Co-phylog: an assembly-free phylogenomic approach for closely related organisms. *Nucleic Acids Res.* **41,** e75 (2013).

92. Perelman, P. *et al.* A molecular phylogeny of living primates. *PLoS Genet.* **7,** e1001342 (2011).

93. Sims, G. E. & Kim, S.-H. Whole-genome phylogeny of Escherichia coli/Shigella group by feature frequency profiles (FFPs). *PNAS* **108,** 8329–8334 (2011).

94. Maddison, W. P. Gene Trees in Species Trees. *Syst Biol* **46,** 523–536 (1997).

95. Huelsenbeck, J. P. Performance of Phylogenetic Methods in Simulation. *Syst Biol* **44,** 17–48 (1995).

96. Kohavi, R. & John, G. H. Wrappers for feature subset selection. *Artificial Intelligence* **97,** 273–324 (1997).

97. Mitchell, M. & Holland, J. When Will a Genetic Algorithm Outperform Hill-Climbing? *Computer Science Faculty Publications and Presentations* (1993).

98. Iswandy, K. & Koenig, A. Towards Effective Unbiased Automated Feature Selection. in *Proceedings of the Sixth International Conference on Hybrid Intelligent Systems* 29– (IEEE Computer Society, 2006). doi:10.1109/HIS.2006.72

99. Kuhner, M. K. & Felsenstein, J. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol. Biol. Evol.* **11,** 459–468 (1994).

100. Page, R. D. M. Phyloinformatics: Toward a Phylogenetic Database. *Data Mining in Bioinformatics* 219–241 (2004). doi:10.1007/1-84628-059-1_10

101. Piel, W. H., Sanderson, M. J. & Donoghue, M. J. The small-world dynamics of tree networks and data mining in phyloinformatics. *Bioinformatics* **19,** 1162–1168 (2003).

102. Smith, S. A. & Dunn, C. W. Phyutility: a phyloinformatics tool for trees, alignments and molecular data. *Bioinformatics* **24,** 715–716 (2008).

103. The Java Language Environment. Available at: http://www.oracle.com/technetwork/java/intro-141325.html. (Accessed: 30th March 2016)

104. Creating Extensible Applications (The Java™ Tutorials > The Extension Mechanism > Creating and Using Extensions). Available at: https://docs.oracle.com/javase/tutorial/ext/basics/spi.html. (Accessed: 30th March 2016)

105. Retief, J. in *Bioinformatics Methods and Protocols* (eds. Misener, S. & Krawetz, S.) 243–258 (Humana Press, 1999).

106. Drummond, A. J., Suchard, M. A., Xie, D. & Rambaut, A. Bayesian Phylogenetics with BEAUti and the BEAST 1.7. *Mol Biol Evol* **29,** 1969–1973 (2012).

107. Geyer, C. J. & Thompson, E. A. Annealing Markov Chain Monte Carlo with Applications to Ancestral Inference. *Journal of the American Statistical Association* **90,** 909–920 (1995).

108. Huelsenbeck, J. P. & Ronquist, F. MRBAYES: Bayesian inference of phylogenetic trees. *Bioinformatics* **17,** 754–755 (2001).

109. Felsenstein, J. Confidence Limits on Phylogenies: An Approach Using the Bootstrap. *Evolution* **39,** 783–791 (1985).

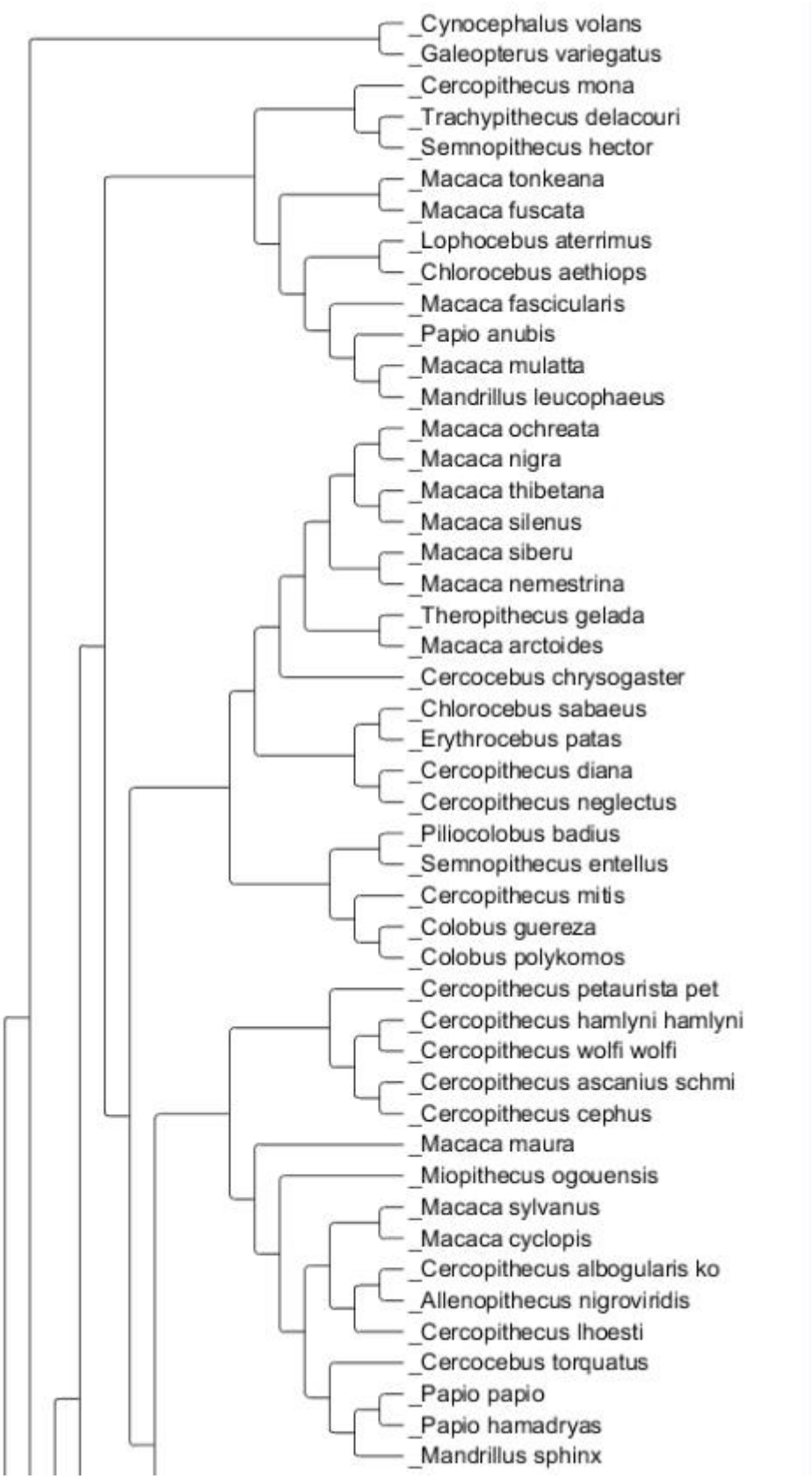
110. Felsenstein, J. & Kishino, H. Is There Something Wrong with the Bootstrap on Phylogenies? A Reply to Hillis and Bull. *Systematic Biology* **42,** 193–200 (1993).

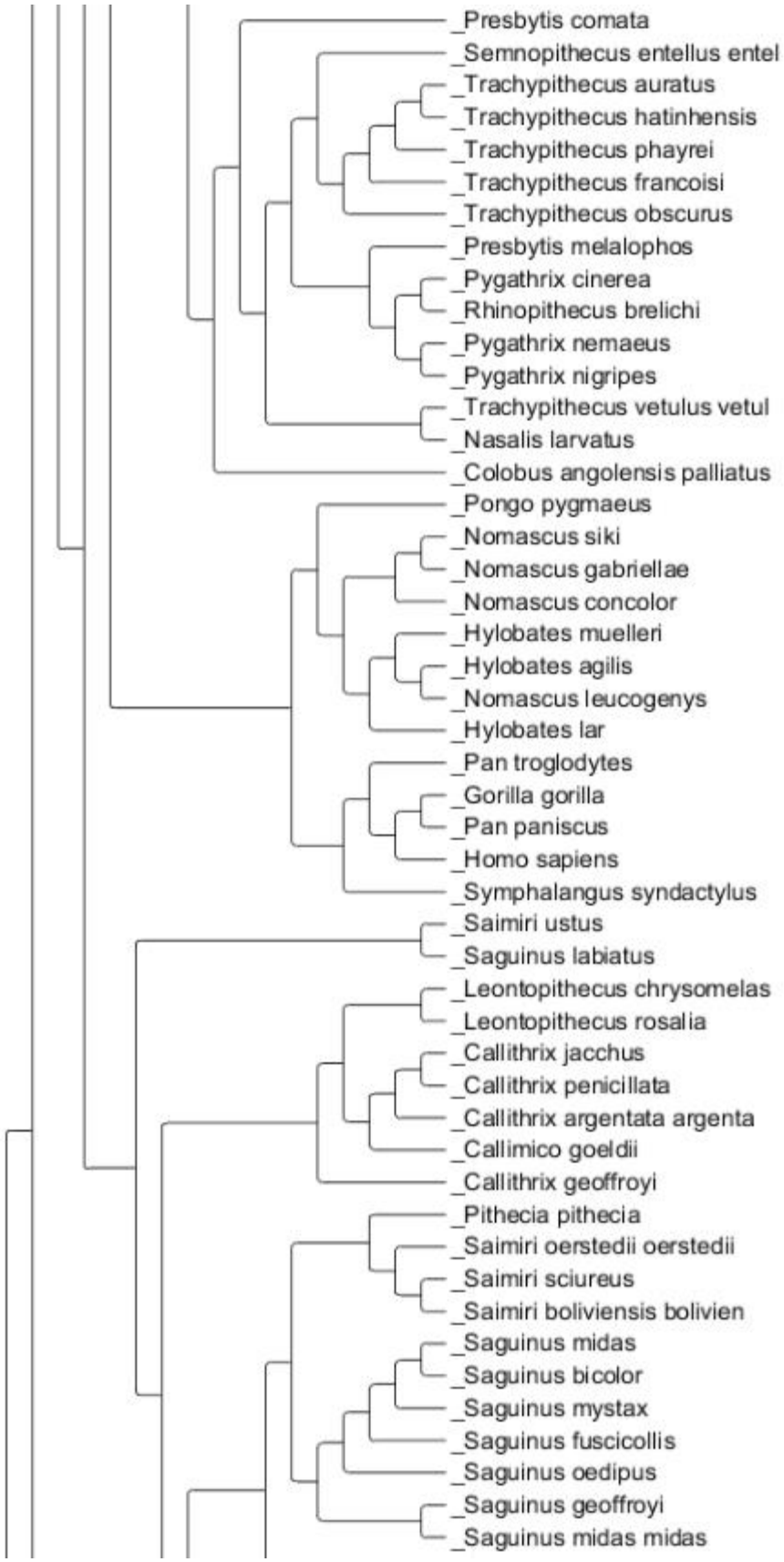
111. Kaminsky, A. Parallel Java: A Unified API for Shared Memory and Cluster Parallel Programming in 100% Java. *Proceedings - 21st International Parallel and Distributed Processing Symposium, IPDPS 2007; Abstracts and CD-ROM* 1–8 (2007). doi:10.1109/IPDPS.2007.370421

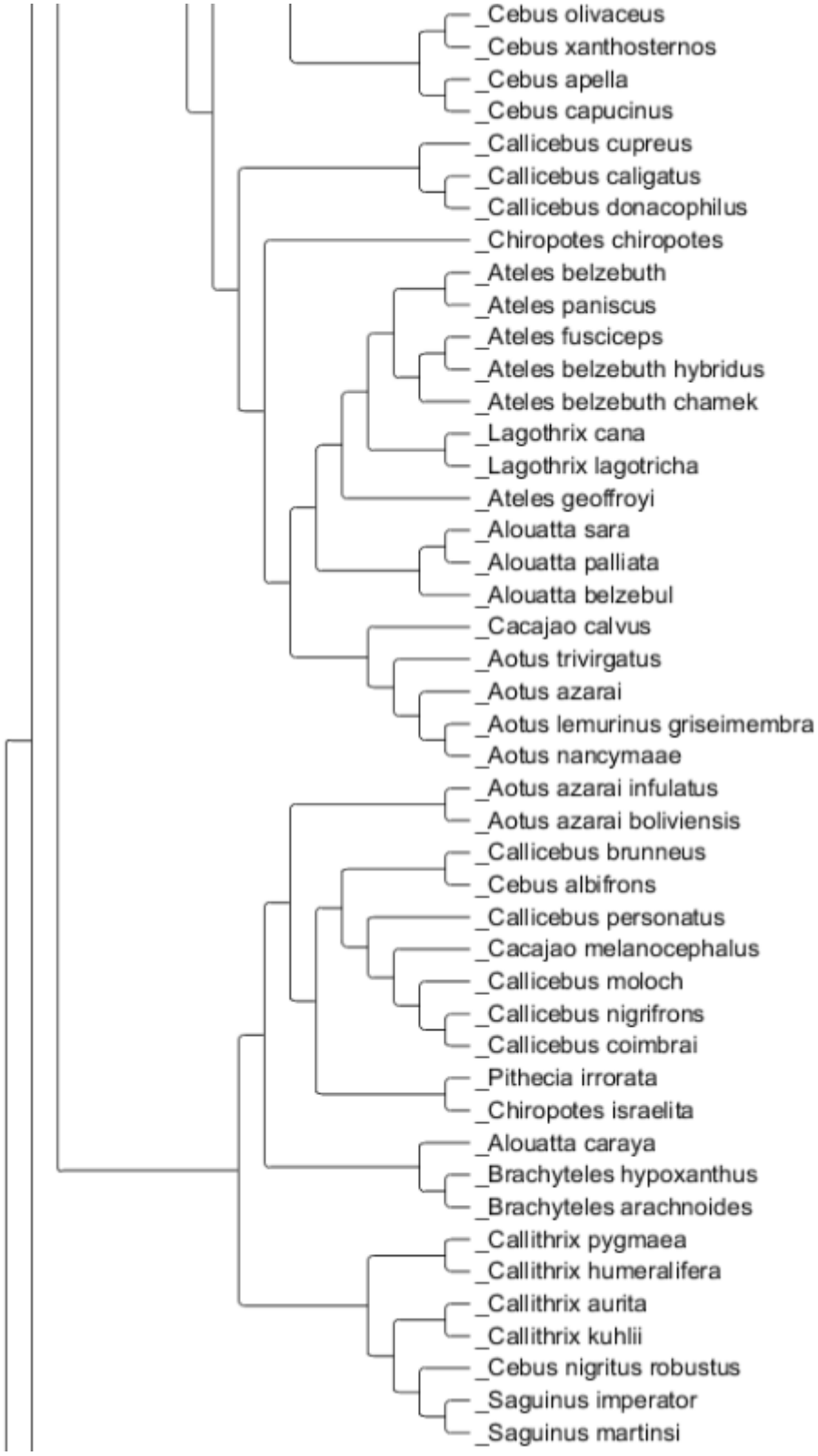
112. Hillis, D. M., Huelsenbeck, J. P. & Cunningham, C. W. Application and Accuracy of Molecular Phylogenies. *Science* **264,** 671–677 (1994).

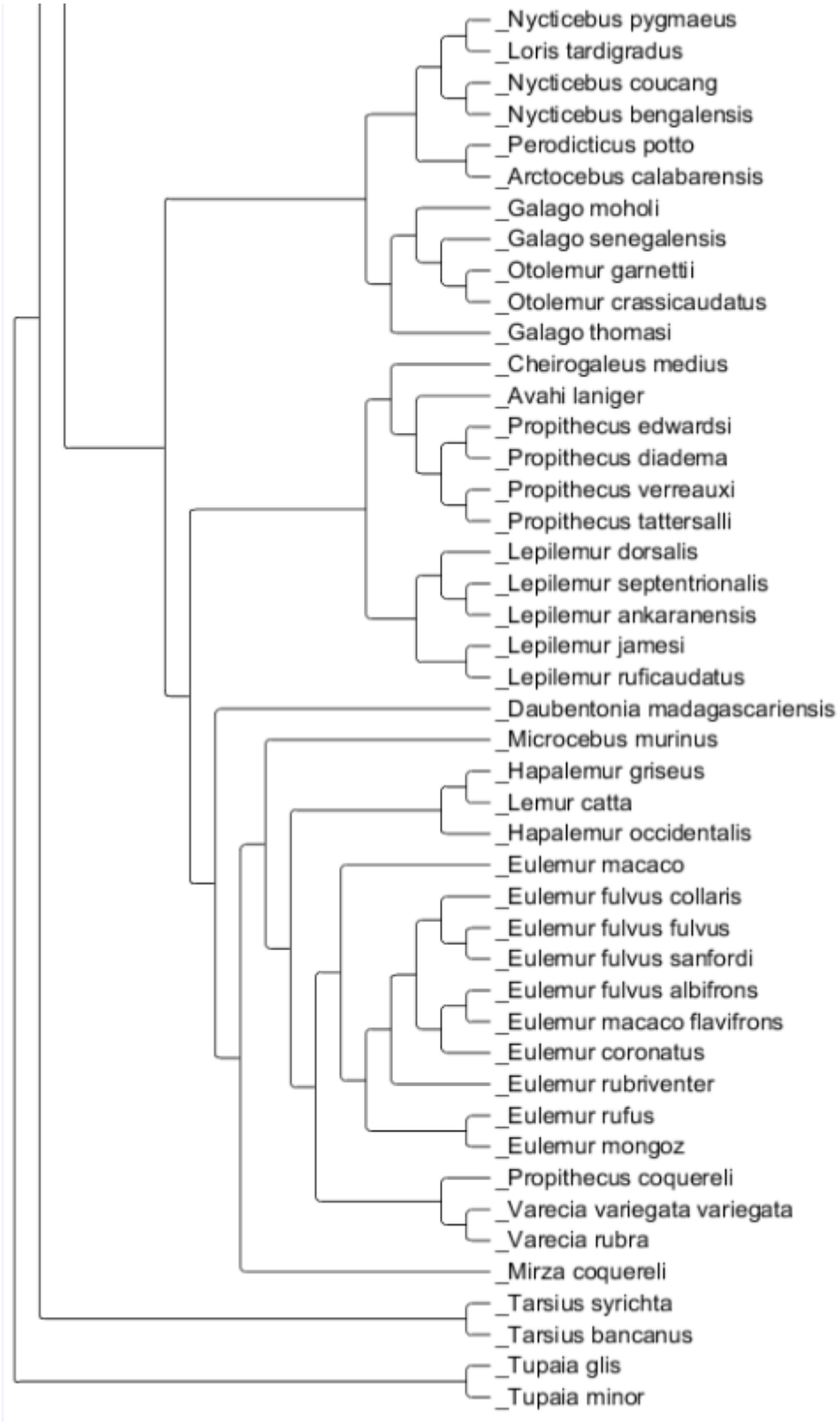
Supplemental Material

SF1: The species tree by Scrawkov-Phy of the 181 primate species over 52 loci in a traditional cladogram view. A full size pdf or png file is available request. Each of the subsequent 4 pages contains a piece of the figure, taken from top to bottom.

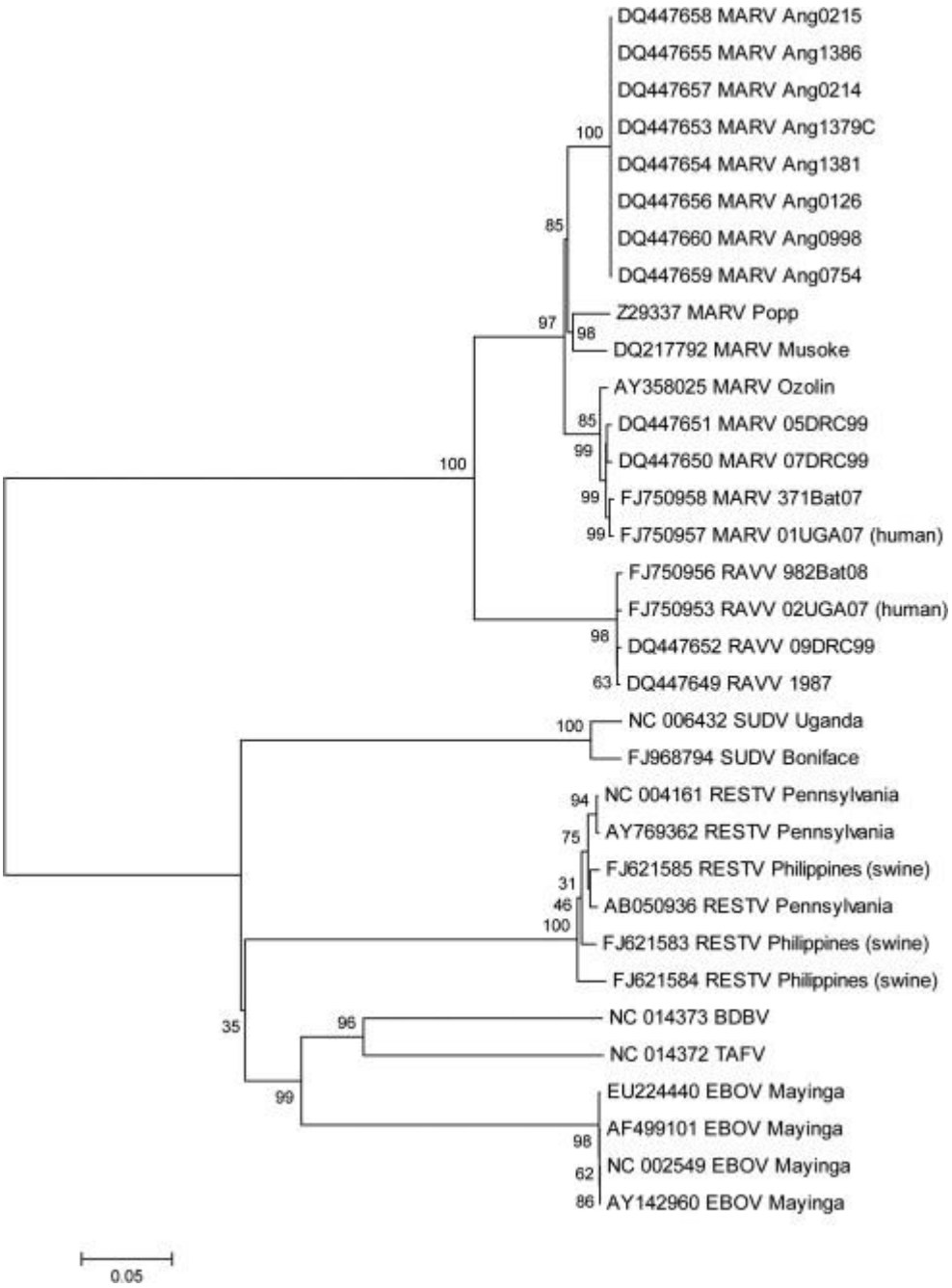

_Cynocephalus volans
_Galeopterus variegatus
_Cercopithecus mona
_Trachypithecus delacouri
_Semnopithecus hector
_Macaca tonkeana
_Macaca fuscata
_Lophocebus aterrimus
_Chlorocebus aethiops
_Macaca fascicularis
_Papio anubis
_Macaca mulatta
_Mandrillus leucophaeus
_Macaca ochreata
_Macaca nigra
_Macaca thibetana
_Macaca silenus
_Macaca siberu
_Macaca nemestrina
_Theropithecus gelada
_Macaca arctoides
_Cercocebus chrysogaster
_Chlorocebus sabaeus
_Erythrocebus patas
_Cercopithecus diana
_Cercopithecus neglectus
_Piliocolobus badius
_Semnopithecus entellus
_Cercopithecus mitis
_Colobus guereza
_Colobus polykomos
_Cercopithecus petaurista pet
_Cercopithecus hamlyni hamlyni
_Cercopithecus wolfi wolfi
_Cercopithecus ascanius schmi
_Cercopithecus cephus
_Macaca maura
_Miopithecus ogouensis
_Macaca sylvanus
_Macaca cyclopis
_Cercopithecus albogularis ko
_Allenopithecus nigroviridis
_Cercopithecus lhoesti
_Cercocebus torquatus
_Papio papio
_Papio hamadryas
_Mandrillus sphinx


_Presbytis comata
_Semnopithecus entellus entel
_Trachypithecus auratus
_Trachypithecus hatinhensis
_Trachypithecus phayrei
_Trachypithecus francoisi
_Trachypithecus obscurus
_Presbytis melalophos
_Pygathrix cinerea
_Rhinopithecus brelichi
_Pygathrix nemaeus
_Pygathrix nigripes
_Trachypithecus vetulus vetul
_Nasalis larvatus
_Colobus angolensis palliatus
_Pongo pygmaeus
_Nomascus siki
_Nomascus gabriellae
_Nomascus concolor
_Hylobates muelleri
_Hylobates agilis
_Nomascus leucogenys
_Hylobates lar
_Pan troglodytes
_Gorilla gorilla
_Pan paniscus
_Homo sapiens
_Symphalangus syndactylus
_Saimiri ustus
_Saguinus labiatus
_Leontopithecus chrysomelas
_Leontopithecus rosalia
_Callithrix jacchus
_Callithrix penicillata
_Callithrix argentata argenta
_Callimico goeldii
_Callithrix geoffroyi
_Pithecia pithecia
_Saimiri oerstedii oerstedii
_Saimiri sciureus
_Saimiri boliviensis bolivien
_Saguinus midas
_Saguinus bicolor
_Saguinus mystax
_Saguinus fuscicollis
_Saguinus oedipus
_Saguinus geoffroyi
_Saguinus midas midas

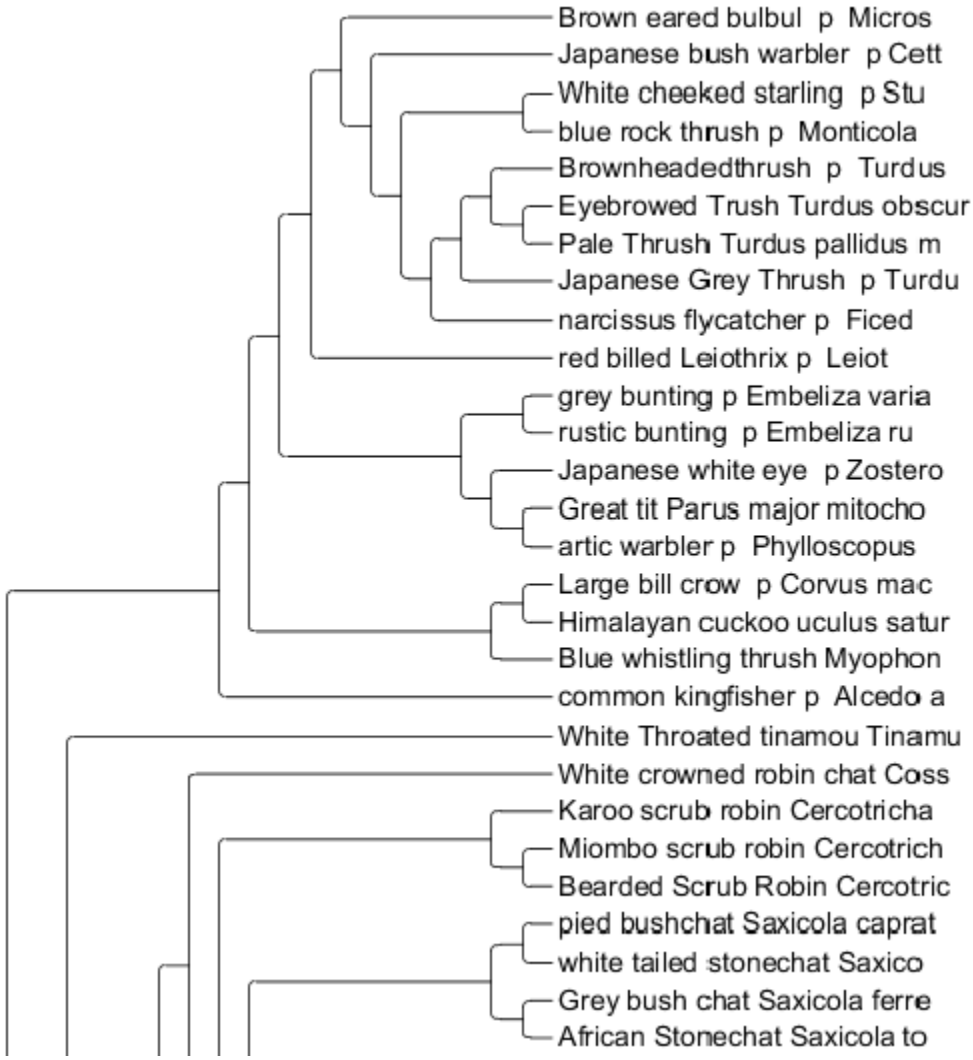

_Cebus olivaceus
_Cebus xanthosternos
_Cebus apella
_Cebus capucinus
_Callicebus cupreus
_Callicebus caligatus
_Callicebus donacophilus
_Chiropotes chiropotes
_Ateles belzebuth
_Ateles paniscus
_Ateles fusciceps
_Ateles belzebuth hybridus
_Ateles belzebuth chamek
_Lagothrix cana
_Lagothrix lagotricha
_Ateles geoffroyi
_Alouatta sara
_Alouatta palliata
_Alouatta belzebul
_Cacajao calvus
_Aotus trivirgatus
_Aotus azarai
_Aotus lemurinus griseimembra
_Aotus nancymaae
_Aotus azarai infulatus
_Aotus azarai boliviensis
_Callicebus brunneus
_Cebus albifrons
_Callicebus personatus
_Cacajao melanocephalus
_Callicebus moloch
_Callicebus nigrifrons
_Callicebus coimbrai
_Pithecia irrorata
_Chiropotes israelita
_Alouatta caraya
_Brachyteles hypoxanthus
_Brachyteles arachnoides
_Callithrix pygmaea
_Callithrix humeralifera
_Callithrix aurita
_Callithrix kuhlii
_Cebus nigritus robustus
_Saguinus imperator
_Saguinus martinsi

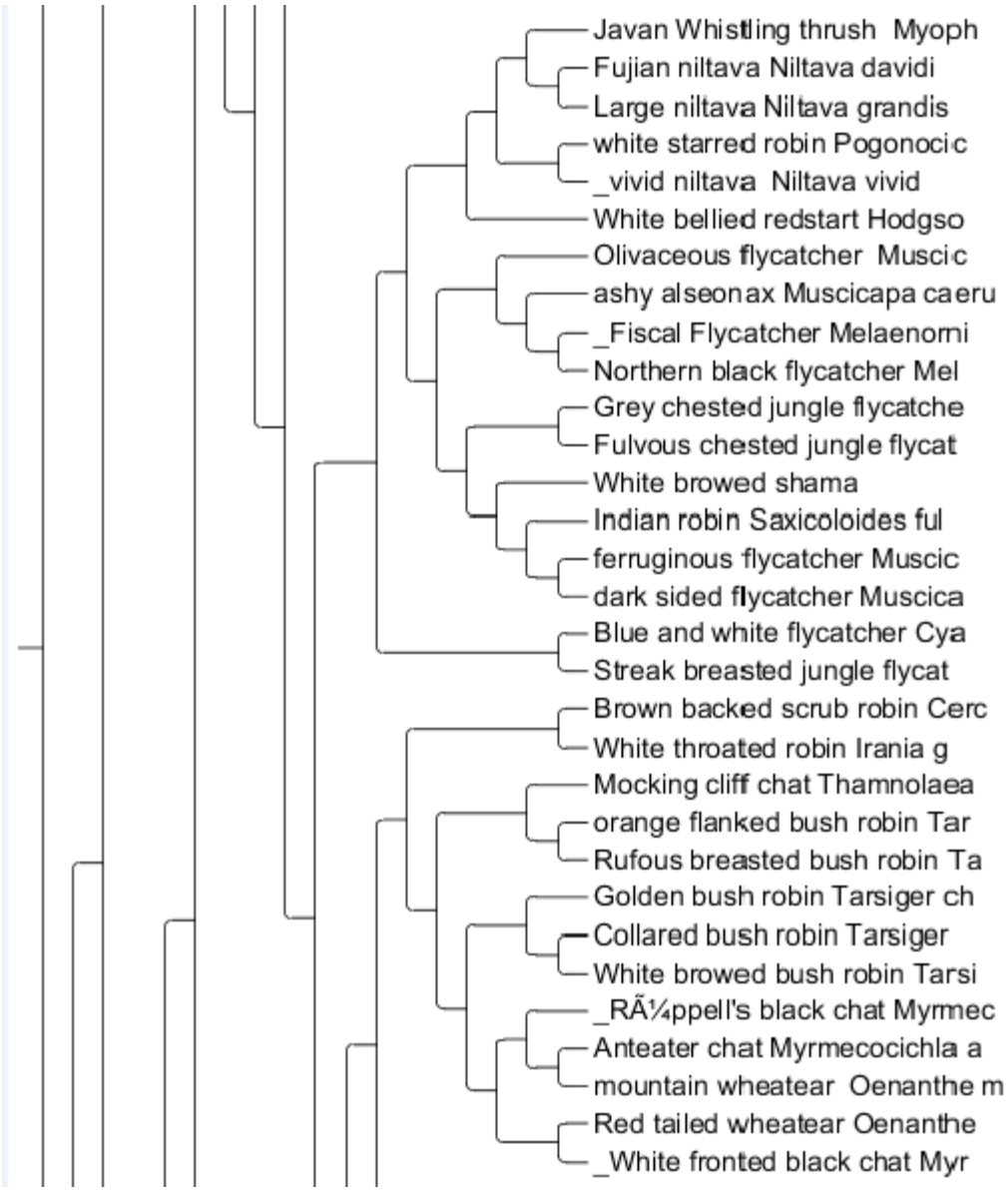

_Nycticebus pygmaeus
_Loris tardigradus
_Nycticebus coucang
_Nycticebus bengalensis
_Perodicticus potto
_Arctocebus calabarensis
_Galago moholi
_Galago senegalensis
_Otolemur garnettii
_Otolemur crassicaudatus
_Galago thomasi
_Cheirogaleus medius
_Avahi laniger
_Propithecus edwardsi
_Propithecus diadema
_Propithecus verreauxi
_Propithecus tattersalli
_Lepilemur dorsalis
_Lepilemur septentrionalis
_Lepilemur ankaranensis
_Lepilemur jamesi
_Lepilemur ruficaudatus
_Daubentonia madagascariensis
_Microcebus murinus
_Hapalemur griseus
_Lemur catta
_Hapalemur occidentalis
_Eulemur macaco
_Eulemur fulvus collaris
_Eulemur fulvus fulvus
_Eulemur fulvus sanfordi
_Eulemur fulvus albifrons
_Eulemur macaco flavifrons
_Eulemur coronatus
_Eulemur rubriventer
_Eulemur rufus
_Eulemur mongoz
_Propithecus coquereli
_Varecia variegata variegata
_Varecia rubra
_Mirza coquereli
_Tarsius syrichta
_Tarsius bancanus
_Tupaia glis
_Tupaia minor

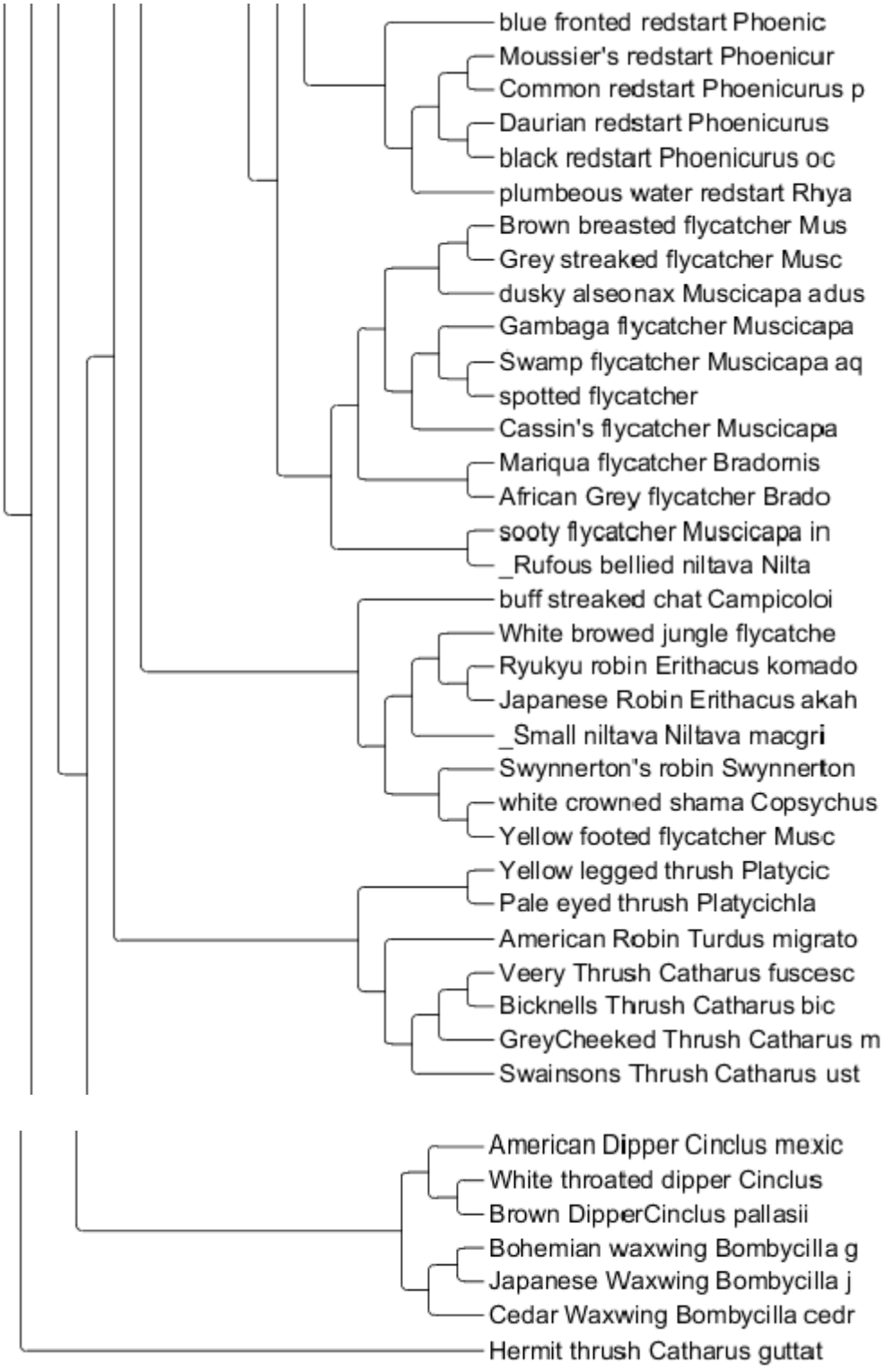
SF2: Filovirus whole genome tree from original study



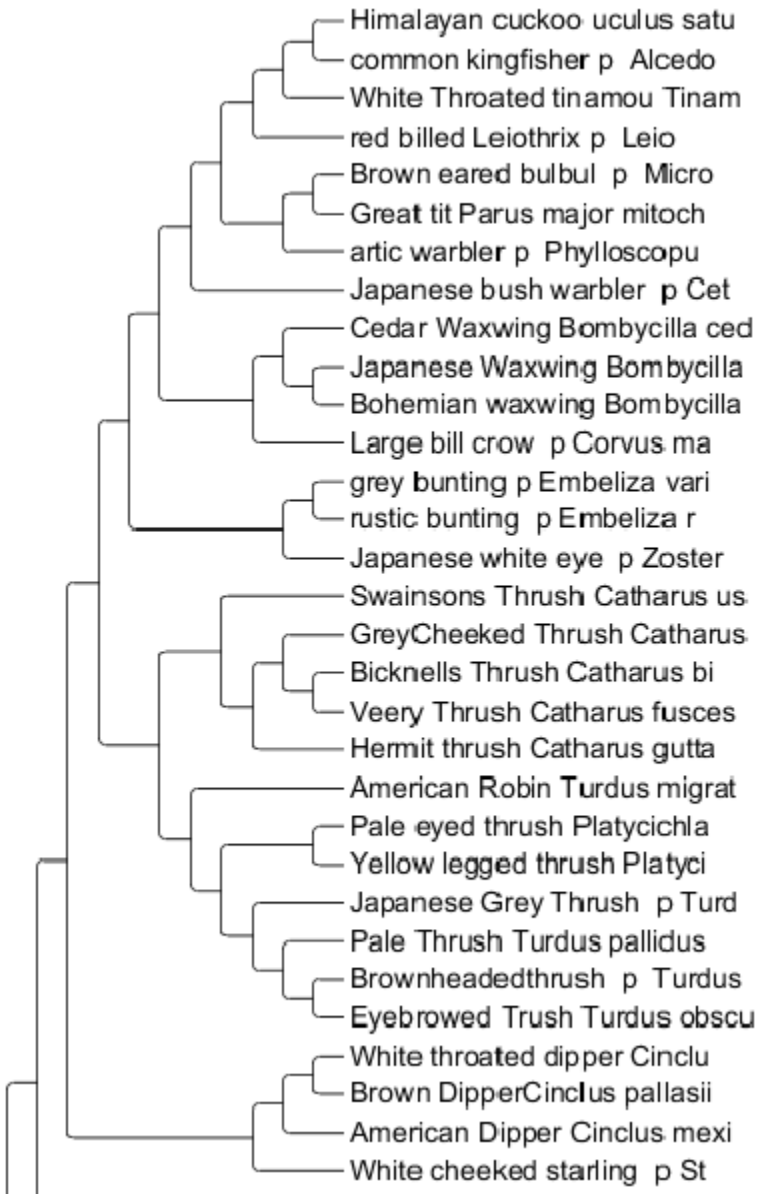
SF3: The cladogram view of the Passeriformes CYTB tree as constructed by Scrawkov-Phy. A full size png/pdf file is available upon request. This and the subsequent 2 pages contain the tree, top to bottom.

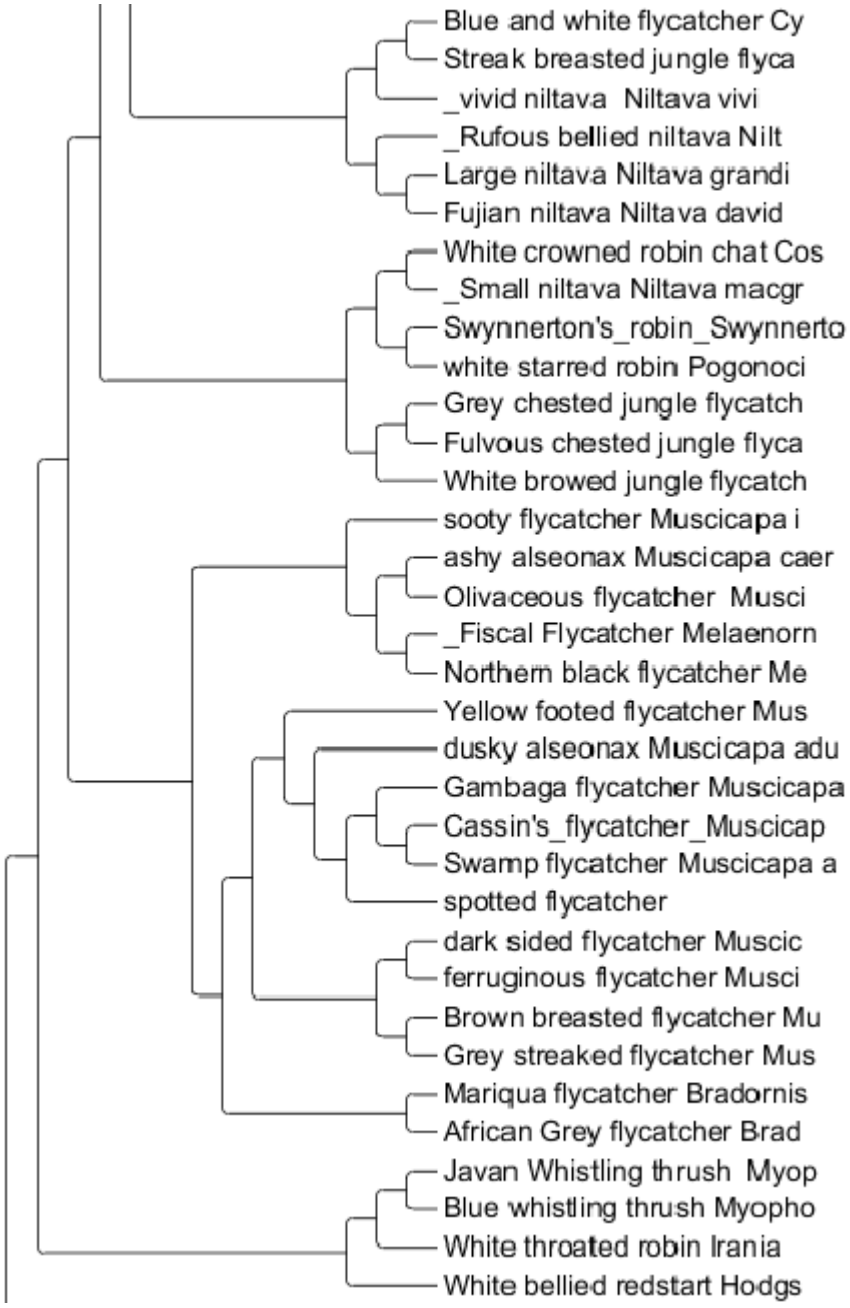


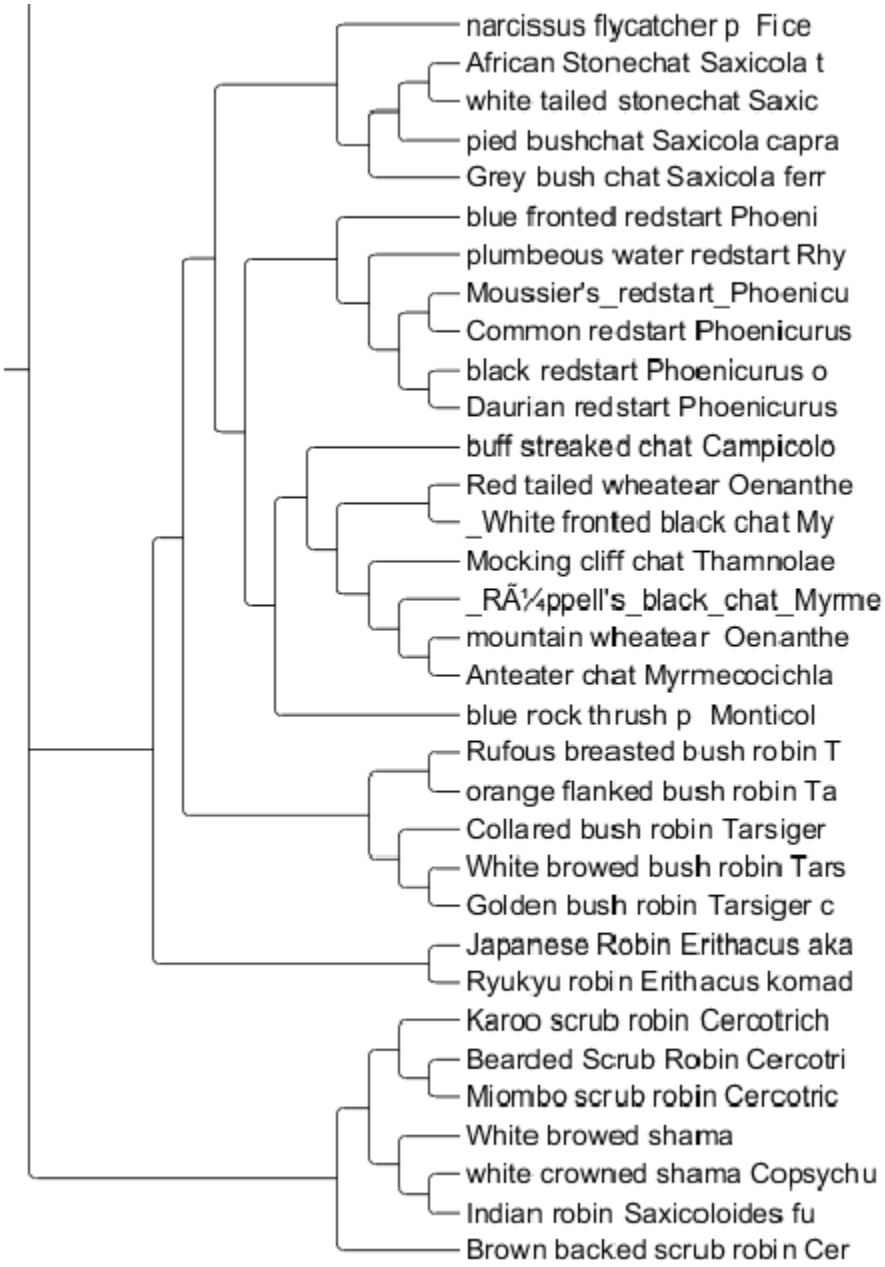

Javan Whistling thrush Myoph
Fujian niltava Niltava davidi
Large niltava Niltava grandis
white starred robin Pogonocic
_vivid niltava Niltava vivid
White bellied redstart Hodgso
Olivaceous flycatcher Muscic
ashy alseonax Muscicapa caeru
_Fiscal Flycatcher Melaenorni
Northern black flycatcher Mel
Grey chested jungle flycatche
Fulvous chested jungle flycat
White browed shama
Indian robin Saxicoloides ful
ferruginous flycatcher Muscic
dark sided flycatcher Muscica
Blue and white flycatcher Cya
Streak breasted jungle flycat
Brown backed scrub robin Cerc
White throated robin Irania g
Mocking cliff chat Thamnolaea
orange flanked bush robin Tar
Rufous breasted bush robin Ta
Golden bush robin Tarsiger ch
Collared bush robin Tarsiger
White browed bush robin Tarsi
_RÃ¼ppell's black chat Myrmec
Anteater chat Myrmecocichla a
mountain wheatear Oenanthe m
Red tailed wheatear Oenanthe
_White fronted black chat Myr


blue fronted redstart Phoenic
Moussier's redstart Phoenicur
Common redstart Phoenicurus p
Daurian redstart Phoenicurus
black redstart Phoenicurus oc
plumbeous water redstart Rhya
Brown breasted flycatcher Mus
Grey streaked flycatcher Musc
dusky alseonax Muscicapa adus
Gambaga flycatcher Muscicapa
Swamp flycatcher Muscicapa aq
spotted flycatcher
Cassin's flycatcher Muscicapa
Mariqua flycatcher Bradornis
African Grey flycatcher Brado
sooty flycatcher Muscicapa in
_Rufous bellied niltava Nilta
buff streaked chat Campicoloi
White browed jungle flycatche
Ryukyu robin Erithacus komado
Japanese Robin Erithacus akah
_Small niltava Niltava macgri
Swynnerton's robin Swynnerton
white crowned shama Copsychus
Yellow footed flycatcher Musc
Yellow legged thrush Platycic
Pale eyed thrush Platycichla
American Robin Turdus migrato
Veery Thrush Catharus fuscesc
Bicknells Thrush Catharus bic
GreyCheeked Thrush Catharus m
Swainsons Thrush Catharus ust
American Dipper Cinclus mexic
White throated dipper Cinclus
Brown DipperCinclus pallasii
Bohemian waxwing Bombycilla g
Japanese Waxwing Bombycilla j
Cedar Waxwing Bombycilla cedr
Hermit thrush Catharus guttat

SF4: The cladogram view of the Passeriformes CYTB tree as constructed by Clustal. A full size png/pdf file is available upon request. This and the subsequent 2 pages contain the tree, top to bottom.

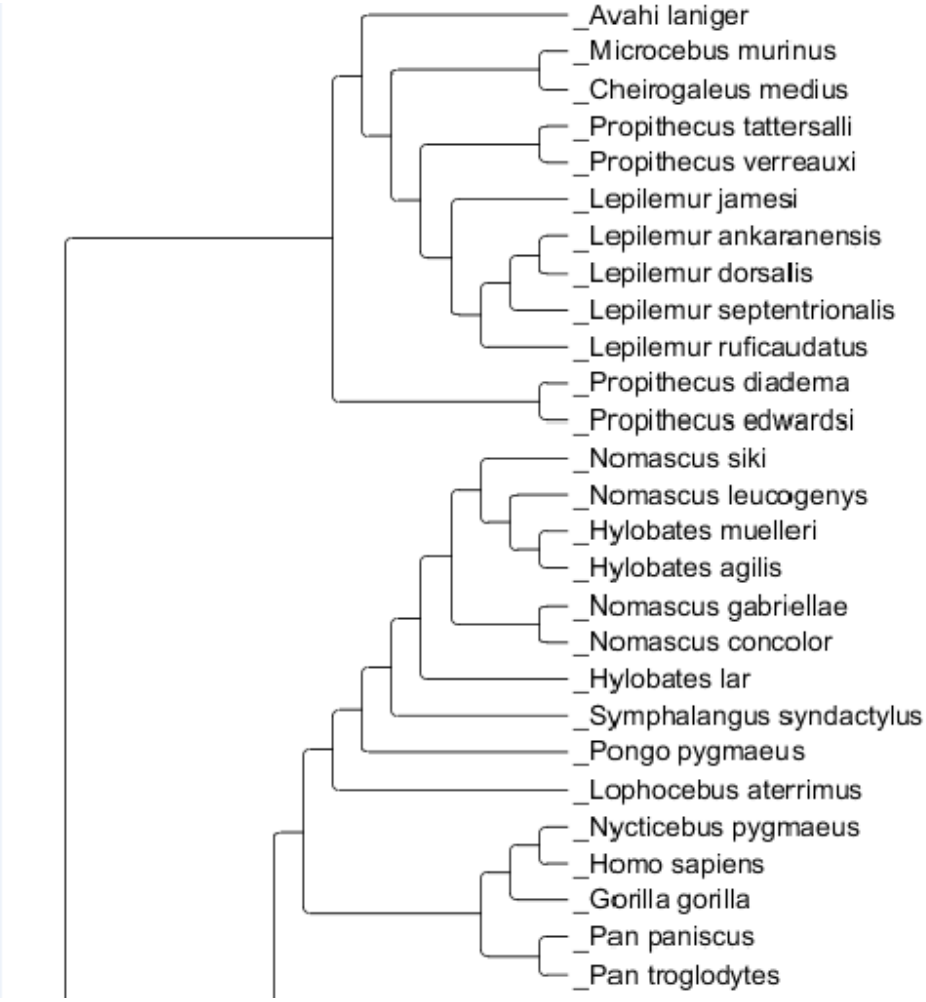


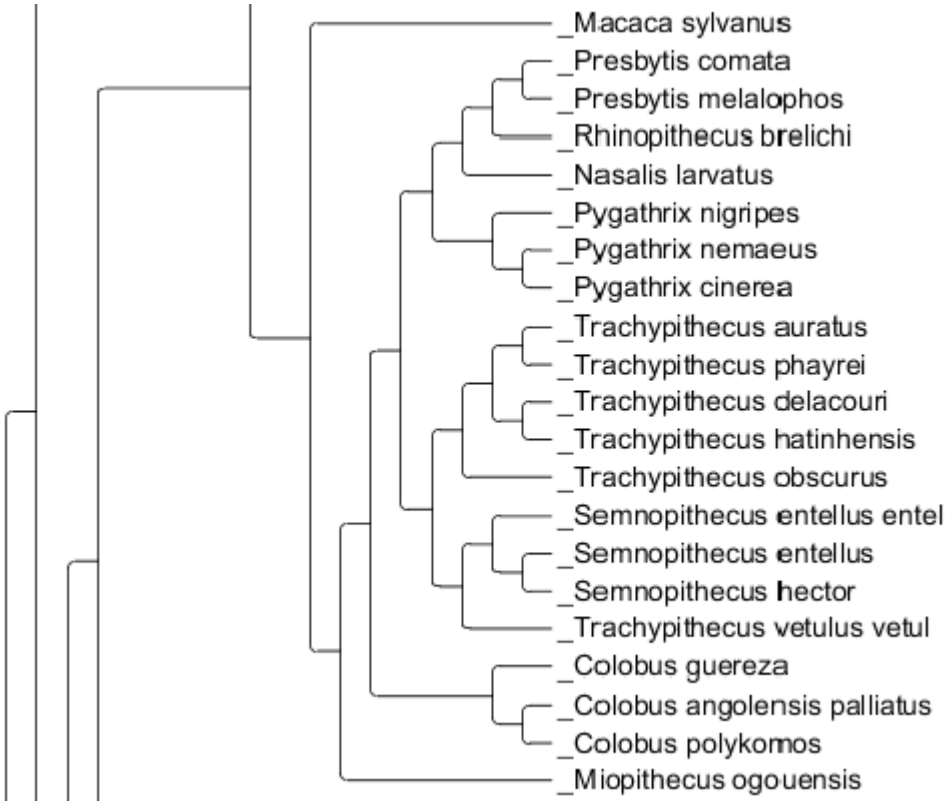

Blue and white flycatcher Cy
Streak breasted jungle flyca
_vivid niltava Niltava vivi
_Rufous bellied niltava Nilt
Large niltava Niltava grandi
Fujian niltava Niltava david
White crowned robin chat Cos
_Small niltava Niltava macgr
Swynnerton's_robin_Swynnerto
white starred robin Pogonoci
Grey chested jungle flycatch
Fulvous chested jungle flyca
White browed jungle flycatch
sooty flycatcher Muscicapa i
ashy alseonax Muscicapa caer
Olivaceous flycatcher Musci
_Fiscal Flycatcher Melaenorn
Northern black flycatcher Me
Yellow footed flycatcher Mus
dusky alseonax Muscicapa adu
Gambaga flycatcher Muscicapa
Cassin's_flycatcher_Muscicap
Swamp flycatcher Muscicapa a
spotted flycatcher
dark sided flycatcher Muscic
ferruginous flycatcher Musci
Brown breasted flycatcher Mu
Grey streaked flycatcher Mus
Mariqua flycatcher Bradornis
African Grey flycatcher Brad
Javan Whistling thrush Myop
Blue whistling thrush Myopho
White throated robin Irania
White bellied redstart Hodgs

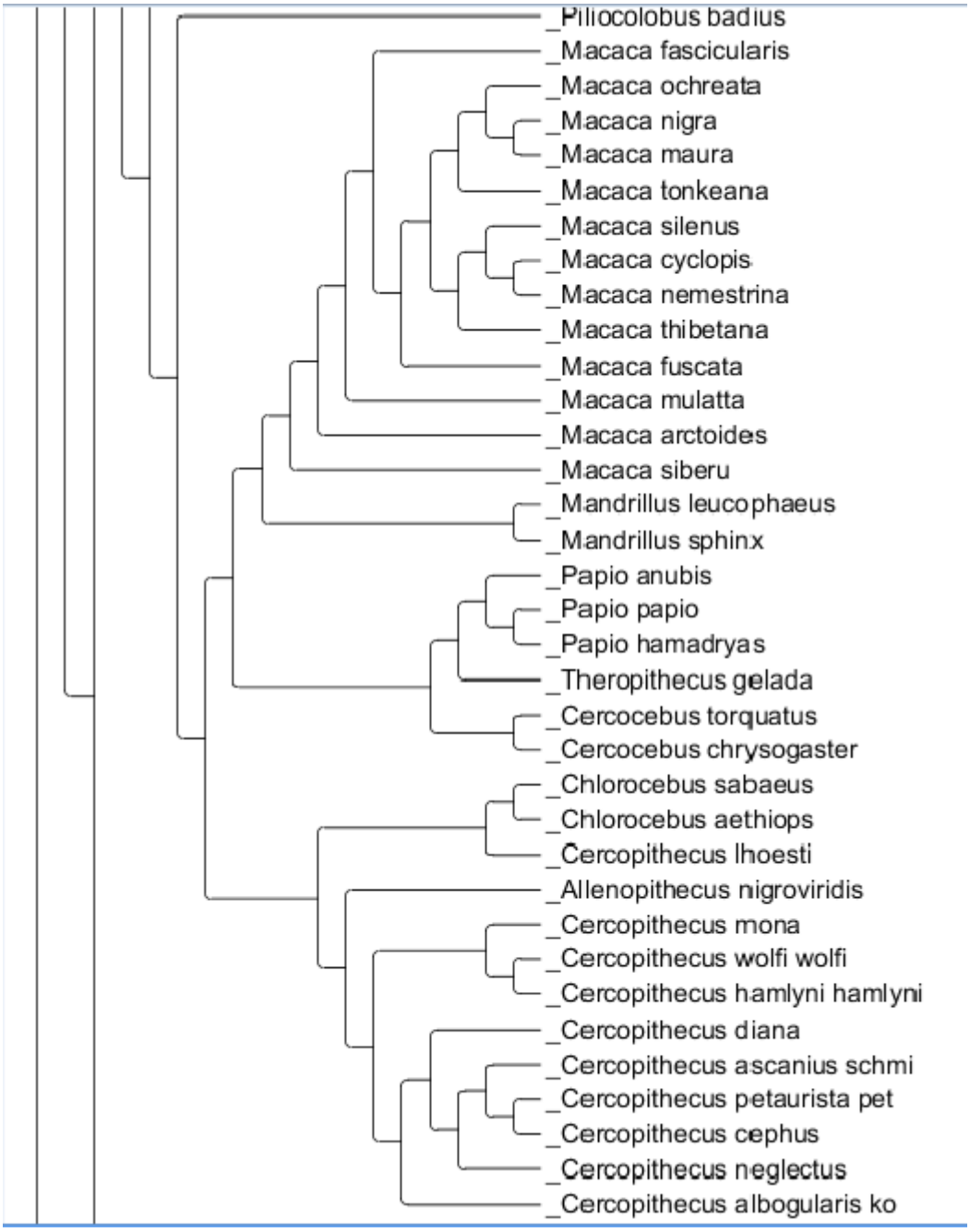


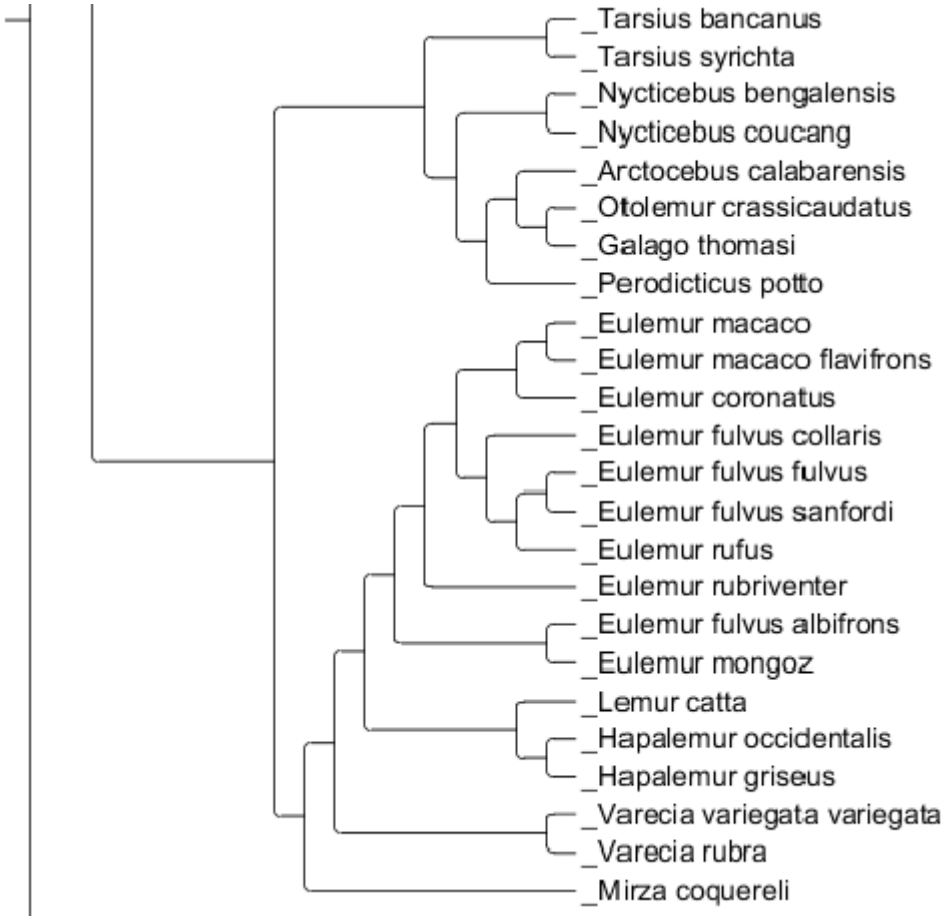
SF5: Example gene trees created *en route* to the primate species tree by Scrawkov-Phy.

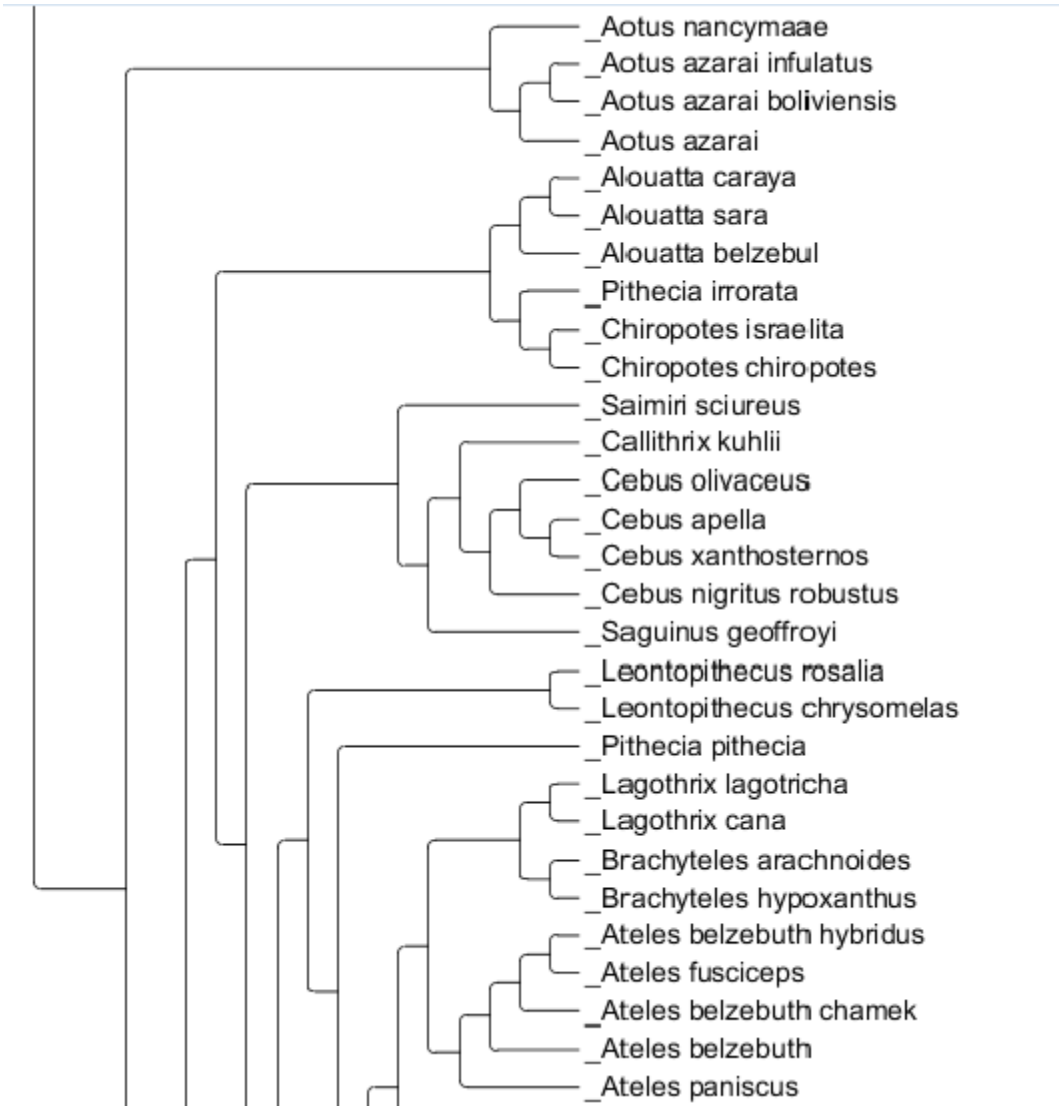
ABCA1

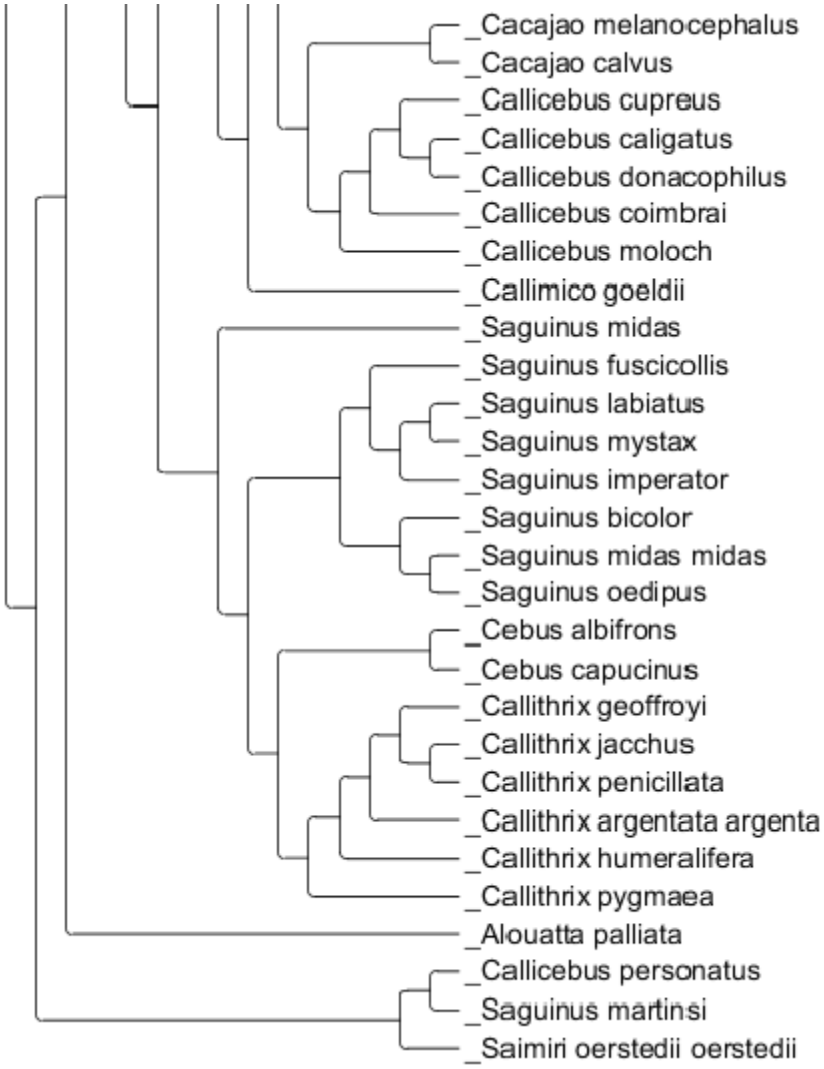

_Avahi laniger
_Microcebus murinus
_Cheirogaleus medius
_Propithecus tattersalli
_Propithecus verreauxi
_Lepilemur jamesi
_Lepilemur ankaranensis
_Lepilemur dorsalis
_Lepilemur septentrionalis
_Lepilemur ruficaudatus
_Propithecus diadema
_Propithecus edwardsi
_Nomascus siki
_Nomascus leucogenys
_Hylobates muelleri
_Hylobates agilis
_Nomascus gabriellae
_Nomascus concolor
_Hylobates lar
_Symphalangus syndactylus
_Pongo pygmaeus
_Lophocebus aterrimus
_Nycticebus pygmaeus
_Homo sapiens
_Gorilla gorilla
_Pan paniscus
_Pan troglodytes


_Macaca sylvanus
_Presbytis comata
_Presbytis melalophos
_Rhinopithecus brelichi
_Nasalis larvatus
_Pygathrix nigripes
_Pygathrix nemaeus
_Pygathrix cinerea
_Trachypithecus auratus
_Trachypithecus phayrei
_Trachypithecus delacouri
_Trachypithecus hatinhensis
_Trachypithecus obscurus
_Semnopithecus entellus entel
_Semnopithecus entellus
_Semnopithecus hector
_Trachypithecus vetulus vetul
_Colobus guereza
_Colobus angolensis palliatus
_Colobus polykomos
_Miopithecus ogouensis

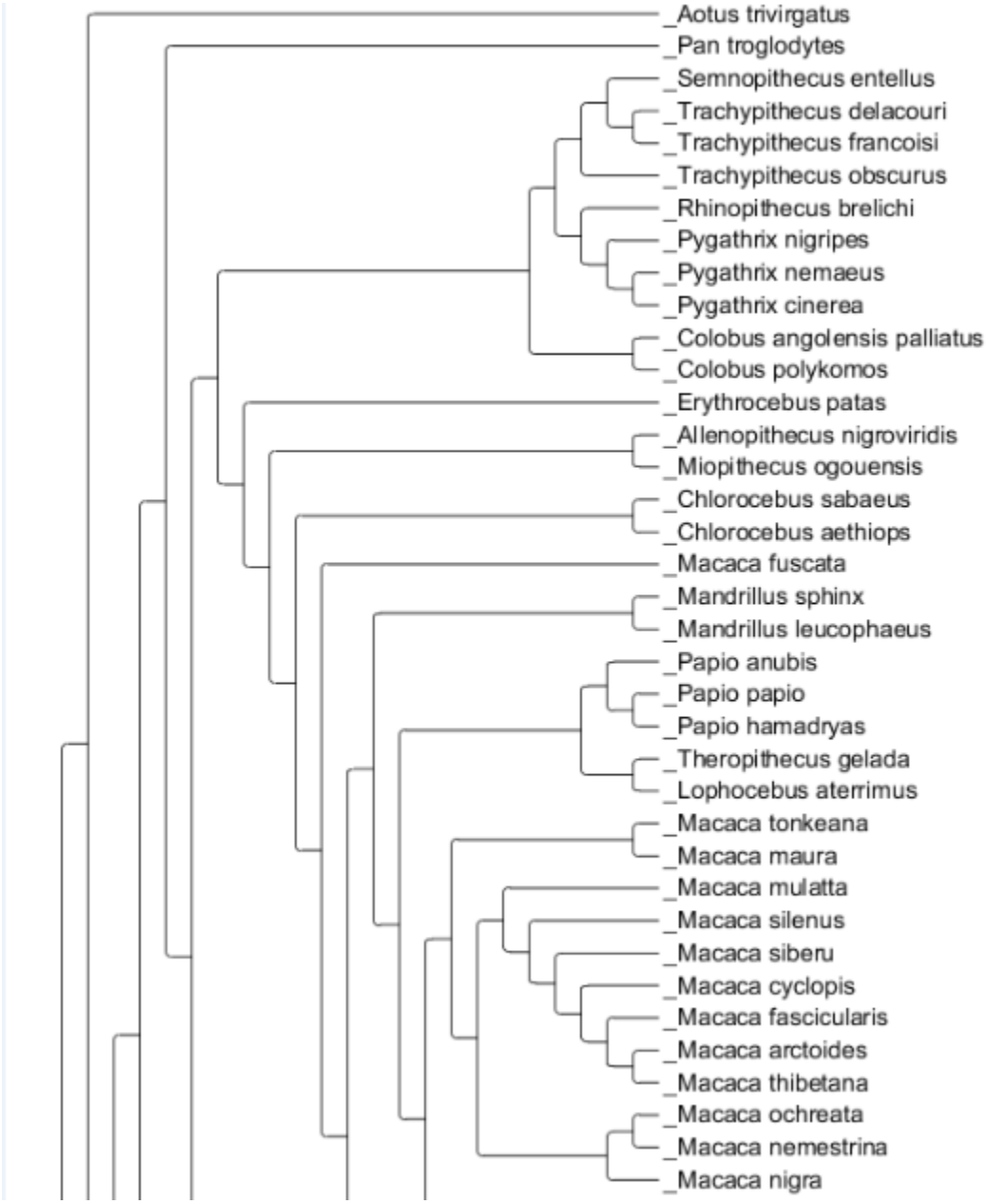

_Piliocolobus badius
_Macaca fascicularis
_Macaca ochreata
_Macaca nigra
_Macaca maura
_Macaca tonkeana
_Macaca silenus
_Macaca cyclopis
_Macaca nemestrina
_Macaca thibetana
_Macaca fuscata
_Macaca mulatta
_Macaca arctoides
_Macaca siberu
_Mandrillus leucophaeus
_Mandrillus sphinx
_Papio anubis
_Papio papio
_Papio hamadryas
_Theropithecus gelada
_Cercocebus torquatus
_Cercocebus chrysogaster
_Chlorocebus sabaeus
_Chlorocebus aethiops
_Cercopithecus lhoesti
_Allenopithecus nigroviridis
_Cercopithecus mona
_Cercopithecus wolfi wolfi
_Cercopithecus hamlyni hamlyni
_Cercopithecus diana
_Cercopithecus ascanius schmi
_Cercopithecus petaurista pet
_Cercopithecus cephus
_Cercopithecus neglectus
_Cercopithecus albogularis ko

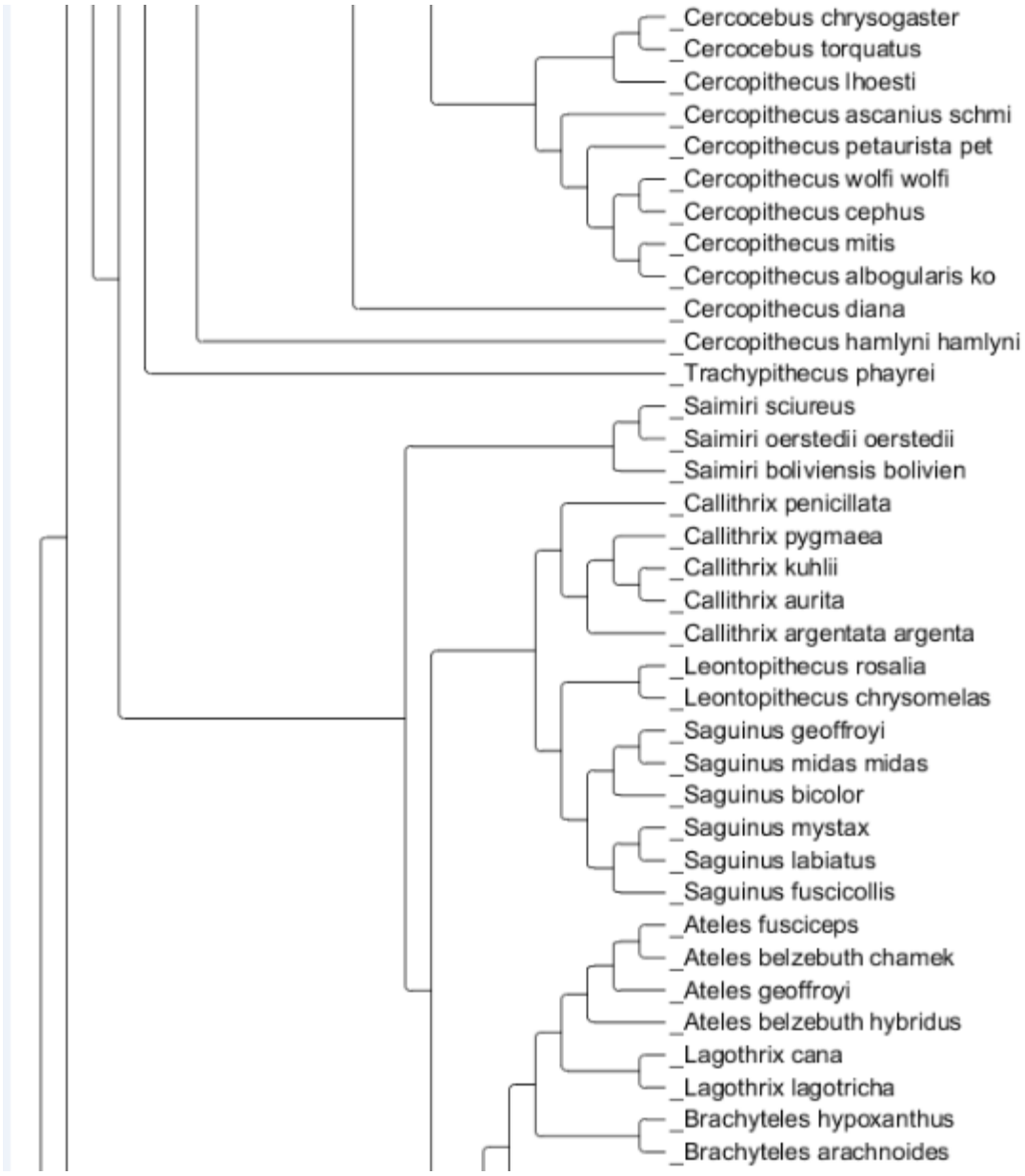

_Tarsius bancanus
_Tarsius syrichta
_Nycticebus bengalensis
_Nycticebus coucang
_Arctocebus calabarensis
_Otolemur crassicaudatus
_Galago thomasi
_Perodicticus potto
_Eulemur macaco
_Eulemur macaco flavifrons
_Eulemur coronatus
_Eulemur fulvus collaris
_Eulemur fulvus fulvus
_Eulemur fulvus sanfordi
_Eulemur rufus
_Eulemur rubriventer
_Eulemur fulvus albifrons
_Eulemur mongoz
_Lemur catta
_Hapalemur occidentalis
_Hapalemur griseus
_Varecia variegata variegata
_Varecia rubra
_Mirza coquereli

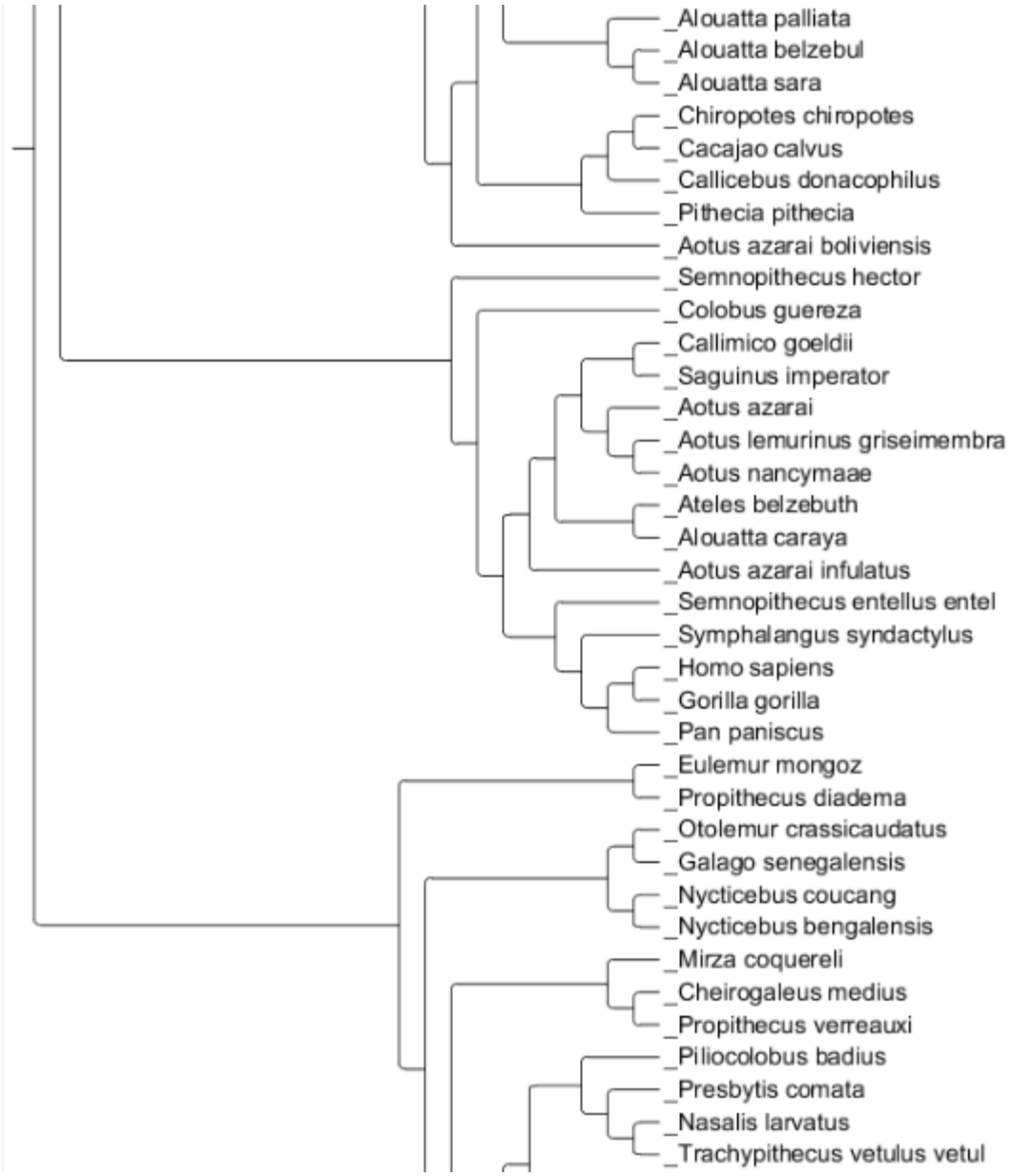

_Aotus nancymaae
_Aotus azarai infulatus
_Aotus azarai boliviensis
_Aotus azarai
_Alouatta caraya
_Alouatta sara
_Alouatta belzebul
_Pithecia irrorata
_Chiropotes israelita
_Chiropotes chiropotes
_Saimiri sciureus
_Callithrix kuhlii
_Cebus olivaceus
_Cebus apella
_Cebus xanthosternos
_Cebus nigritus robustus
_Saguinus geoffroyi
_Leontopithecus rosalia
_Leontopithecus chrysomelas
_Pithecia pithecia
_Lagothrix lagotricha
_Lagothrix cana
_Brachyteles arachnoides
_Brachyteles hypoxanthus
_Ateles belzebuth hybridus
_Ateles fusciceps
_Ateles belzebuth chamek
_Ateles belzebuth
_Ateles paniscus

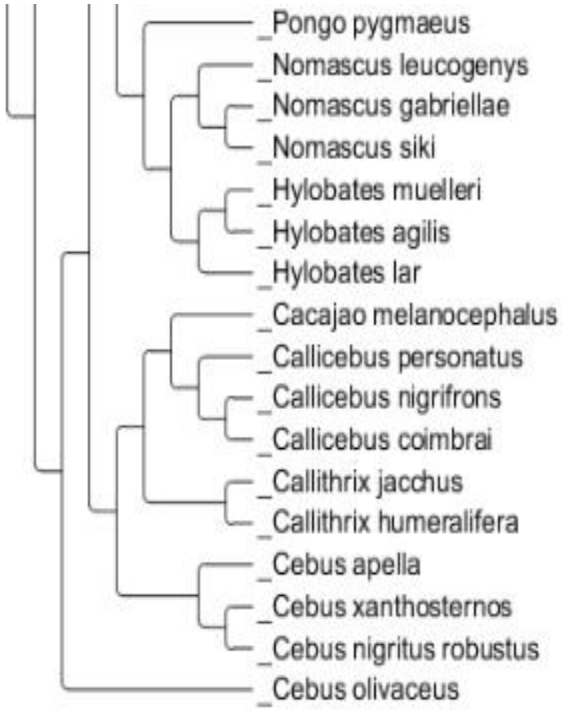

_Cacajao melanocephalus
_Cacajao calvus
_Callicebus cupreus
_Callicebus caligatus
_Callicebus donacophilus
_Callicebus coimbrai
_Callicebus moloch
_Callimico goeldii
_Saguinus midas
_Saguinus fuscicollis
_Saguinus labiatus
_Saguinus mystax
_Saguinus imperator
_Saguinus bicolor
_Saguinus midas midas
_Saguinus oedipus
_Cebus albifrons
_Cebus capucinus
_Callithrix geoffroyi
_Callithrix jacchus
_Callithrix penicillata
_Callithrix argentata argenta
_Callithrix humeralifera
_Callithrix pygmaea
_Alouatta palliata
_Callicebus personatus
_Saguinus martinsi
_Saimiri oerstedii oerstedii

BRCA2


_Aotus trivirgatus
_Pan troglodytes
_Semnopithecus entellus
_Trachypithecus delacouri
_Trachypithecus francoisi
_Trachypithecus obscurus
_Rhinopithecus brelichi
_Pygathrix nigripes
_Pygathrix nemaeus
_Pygathrix cinerea
_Colobus angolensis palliatus
_Colobus polykomos
_Erythrocebus patas
_Allenopithecus nigroviridis
_Miopithecus ogouensis
_Chlorocebus sabaeus
_Chlorocebus aethiops
_Macaca fuscata
_Mandrillus sphinx
_Mandrillus leucophaeus
_Papio anubis
_Papio papio
_Papio hamadryas
_Theropithecus gelada
_Lophocebus aterrimus
_Macaca tonkeana
_Macaca maura
_Macaca mulatta
_Macaca silenus
_Macaca siberu
_Macaca cyclopis
_Macaca fascicularis
_Macaca arctoides
_Macaca thibetana
_Macaca ochreata
_Macaca nemestrina
_Macaca nigra

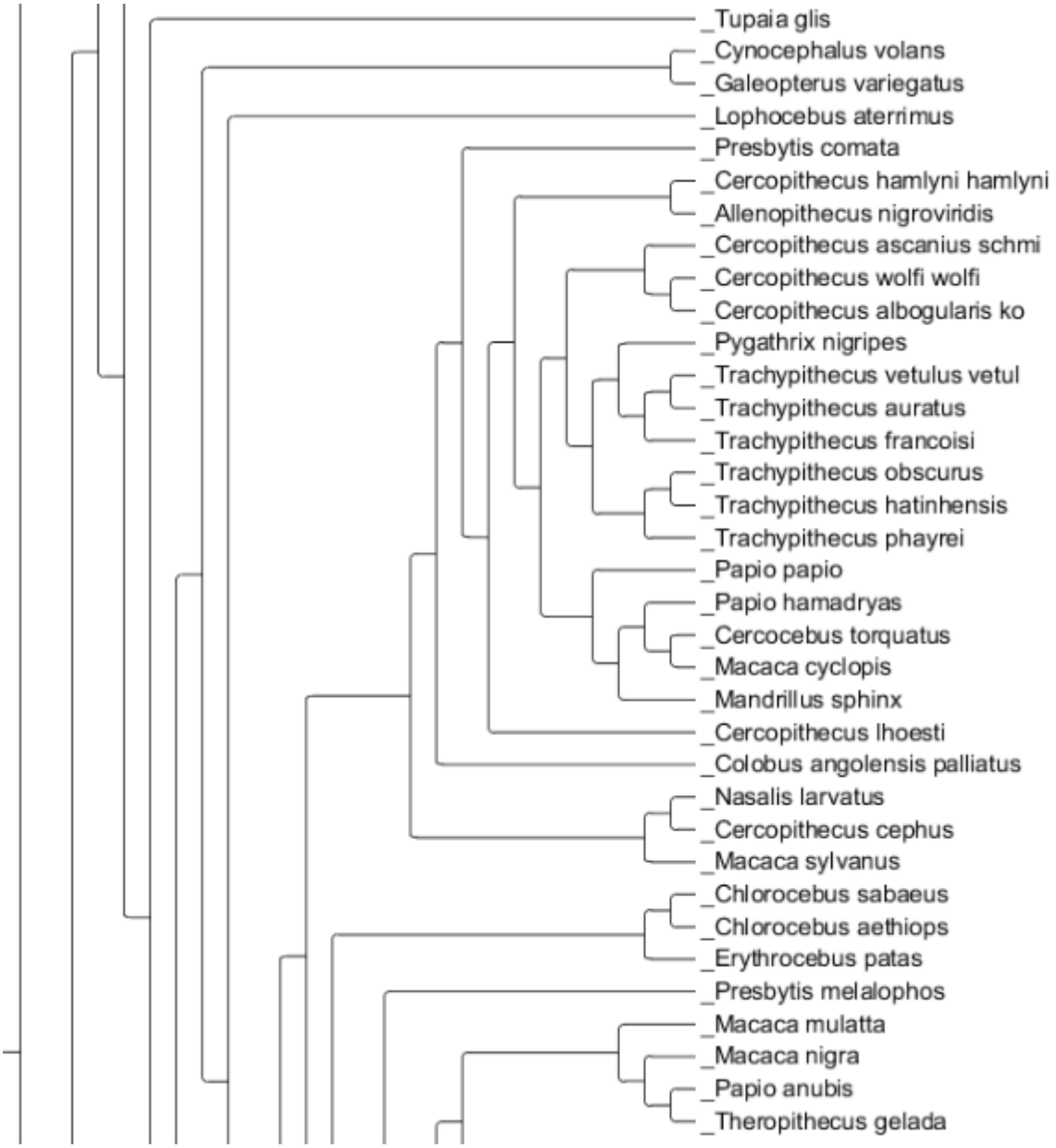

_Cercocebus chrysogaster
_Cercocebus torquatus
_Cercopithecus lhoesti
_Cercopithecus ascanius schmi
_Cercopithecus petaurista pet
_Cercopithecus wolfi wolfi
_Cercopithecus cephus
_Cercopithecus mitis
_Cercopithecus albogularis ko
_Cercopithecus diana
_Cercopithecus hamlyni hamlyni
_Trachypithecus phayrei
_Saimiri sciureus
_Saimiri oerstedii oerstedii
_Saimiri boliviensis bolivien
_Callithrix penicillata
_Callithrix pygmaea
_Callithrix kuhlii
_Callithrix aurita
_Callithrix argentata argenta
_Leontopithecus rosalia
_Leontopithecus chrysomelas
_Saguinus geoffroyi
_Saguinus midas midas
_Saguinus bicolor
_Saguinus mystax
_Saguinus labiatus
_Saguinus fuscicollis
_Ateles fusciceps
_Ateles belzebuth chamek
_Ateles geoffroyi
_Ateles belzebuth hybridus
_Lagothrix cana
_Lagothrix lagotricha
_Brachyteles hypoxanthus
_Brachyteles arachnoides

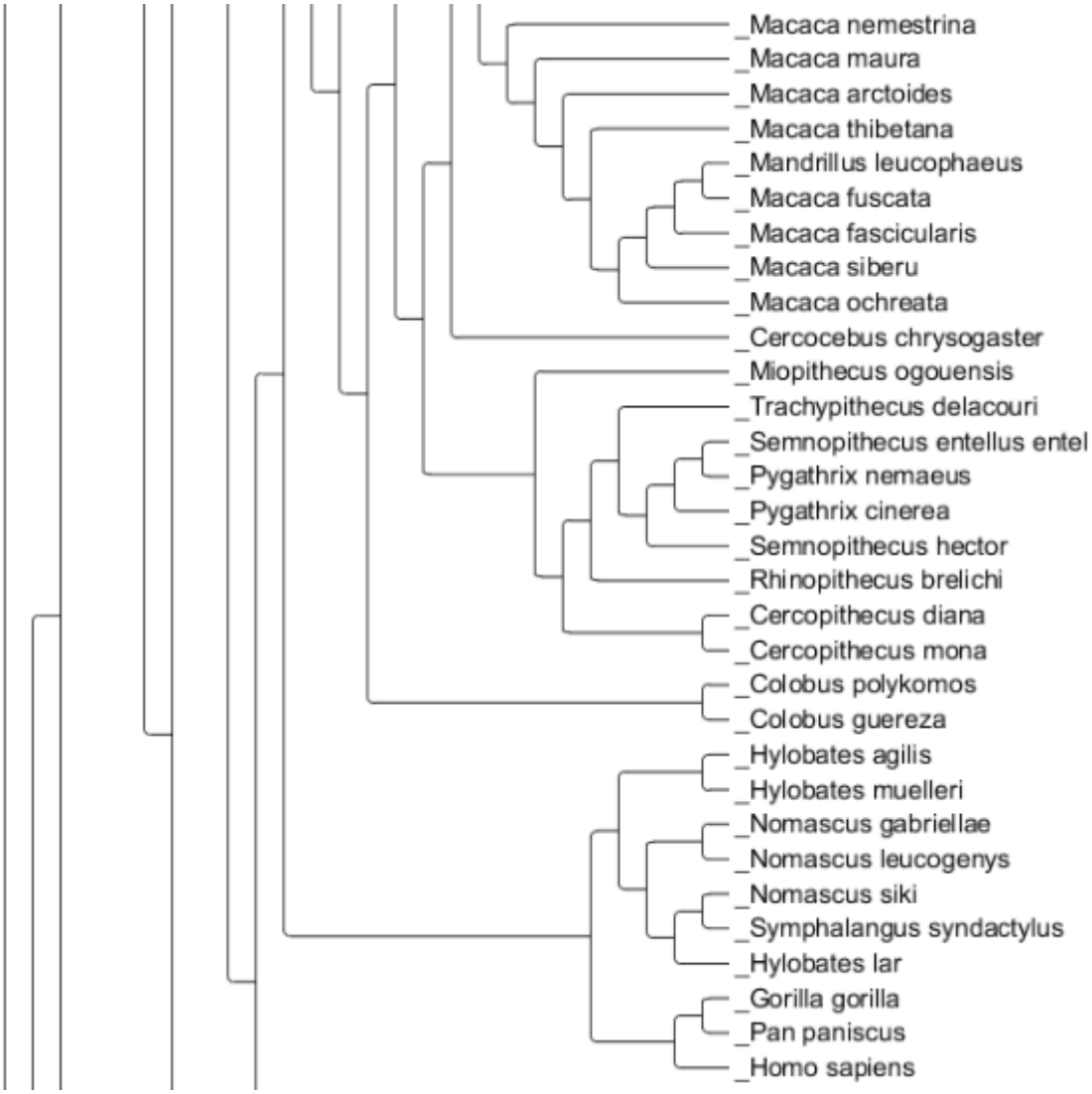

_Alouatta palliata
_Alouatta belzebul
_Alouatta sara
_Chiropotes chiropotes
_Cacajao calvus
_Callicebus donacophilus
_Pithecia pithecia
_Aotus azarai boliviensis
_Semnopithecus hector
_Colobus guereza
_Callimico goeldii
_Saguinus imperator
_Aotus azarai
_Aotus lemurinus griseimembra
_Aotus nancymaae
_Ateles belzebuth
_Alouatta caraya
_Aotus azarai infulatus
_Semnopithecus entellus entel
_Symphalangus syndactylus
_Homo sapiens
_Gorilla gorilla
_Pan paniscus
_Eulemur mongoz
_Propithecus diadema
_Otolemur crassicaudatus
_Galago senegalensis
_Nycticebus coucang
_Nycticebus bengalensis
_Mirza coquereli
_Cheirogaleus medius
_Propithecus verreauxi
_Piliocolobus badius
_Presbytis comata
_Nasalis larvatus
_Trachypithecus vetulus vetul

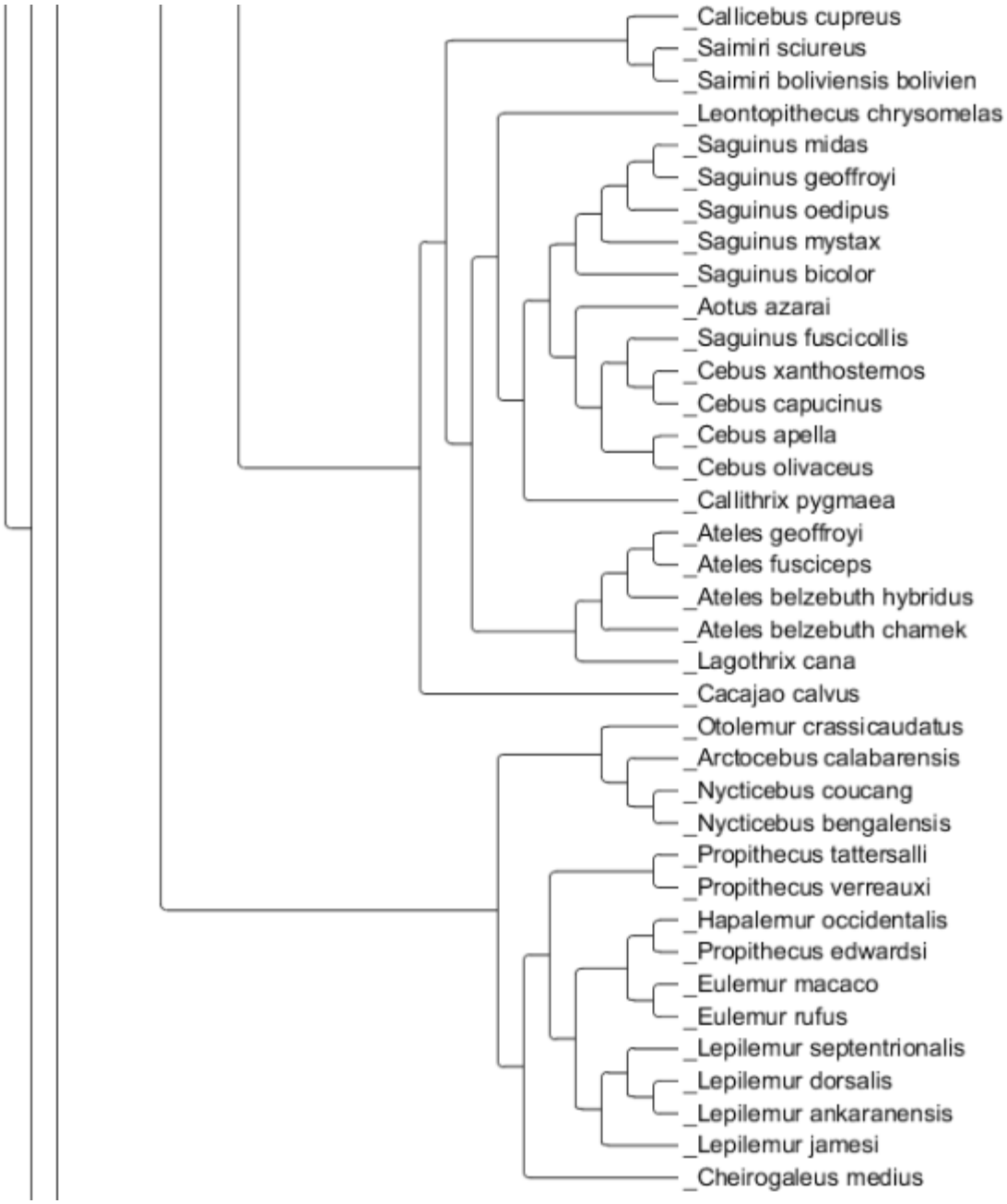

_Pongo pygmaeus
_Nomascus leucogenys
_Nomascus gabriellae
_Nomascus siki
_Hylobates muelleri
_Hylobates agilis
_Hylobates lar
_Cacajao melanocephalus
_Callicebus personatus
_Callicebus nigrifrons
_Callicebus coimbrai
_Callithrix jacchus
_Callithrix humeralifera
_Cebus apella
_Cebus xanthosternos
_Cebus nigritus robustus
_Cebus olivaceus

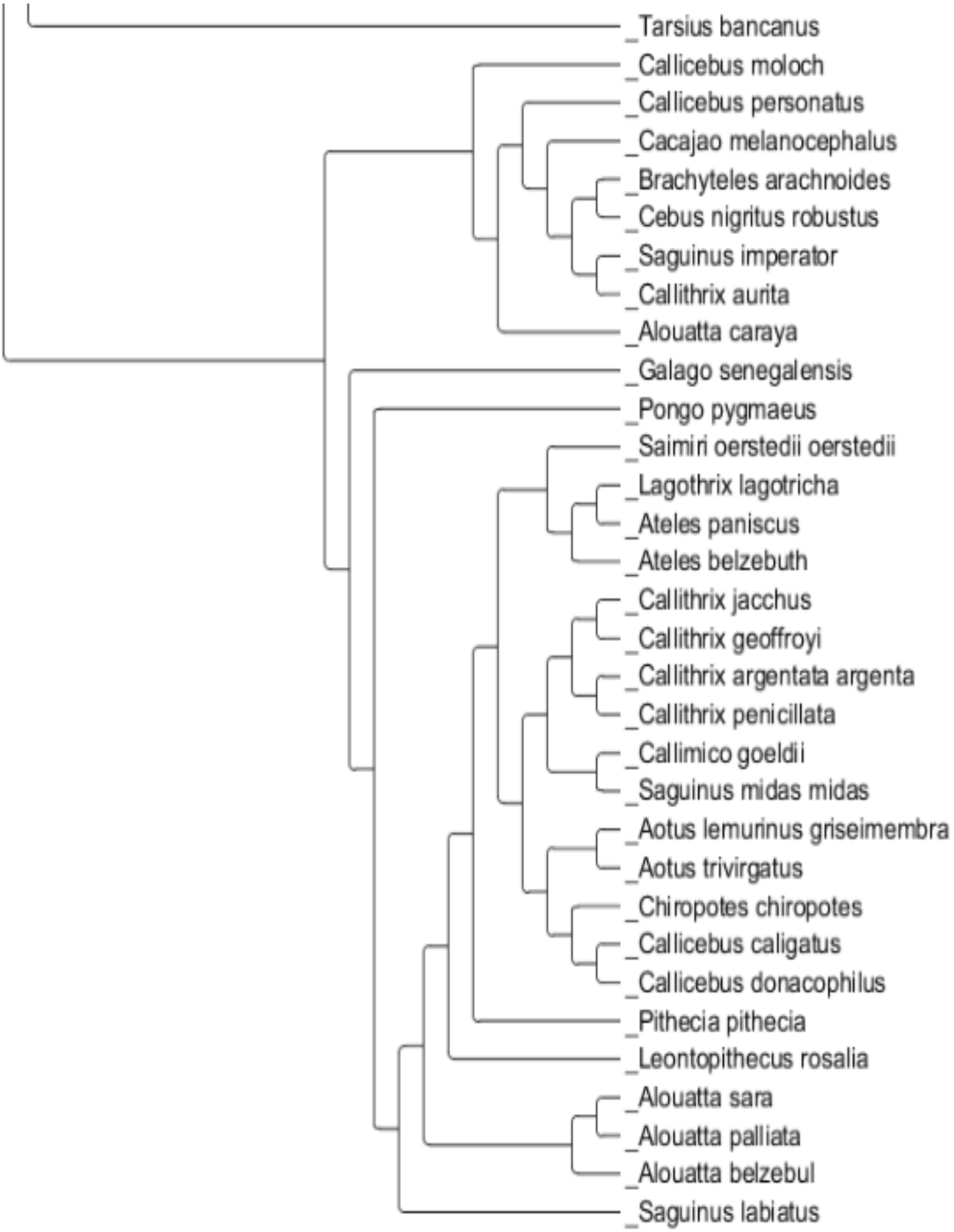
RAG2


_Daubentonia madagascariensis
_Microcebus murinus
_Propithecus coquereli
_Eulemur fulvus fulvus
_Eulemur fulvus albifrons
_Eulemur coronatus
_Galago moholi
_Chiropotes israelita
_Brachyteles hypoxanthus
_Aotus nancymaae
_Otolemur garnettii
_Nycticebus pygmaeus
_Loris tardigradus
_Piliocolobus badius
_Cercopithecus petaurista pet
_Pan troglodytes
_Nomascus concolor
_Semnopithecus entellus
_Macaca silenus
_Macaca tonkeana
_Propithecus diadema
_Avahi laniger
_Lemur catta
_Hapalemur griseus
_Eulemur mongoz
_Mirza coquereli
_Tupaia minor
_Tarsius syrichta
_Perodicticus potto
_Varecia variegata variegata
_Cebus albifrons
_Aotus azarai infulatus
_Callicebus nigrifrons
_Aotus azarai boliviensis
_Callithrix humeralifera
_Callithrix kuhlii
_Callicebus brunneus


_Tupaia glis
_Cynocephalus volans
_Galeopterus variegatus
_Lophocebus aterrimus
_Presbytis comata
_Cercopithecus hamlyni hamlyni
_Allenopithecus nigroviridis
_Cercopithecus ascanius schmi
_Cercopithecus wolfi wolfi
_Cercopithecus albogularis ko
_Pygathrix nigripes
_Trachypithecus vetulus vetul
_Trachypithecus auratus
_Trachypithecus francoisi
_Trachypithecus obscurus
_Trachypithecus hatinhensis
_Trachypithecus phayrei
_Papio papio
_Papio hamadryas
_Cercocebus torquatus
_Macaca cyclopis
_Mandrillus sphinx
_Cercopithecus lhoesti
_Colobus angolensis palliatus
_Nasalis larvatus
_Cercopithecus cephus
_Macaca sylvanus
_Chlorocebus sabaeus
_Chlorocebus aethiops
_Erythrocebus patas
_Presbytis melalophos
_Macaca mulatta
_Macaca nigra
_Papio anubis
_Theropithecus gelada


_Macaca nemestrina
_Macaca maura
_Macaca arctoides
_Macaca thibetana
_Mandrillus leucophaeus
_Macaca fuscata
_Macaca fascicularis
_Macaca siberu
_Macaca ochreata
_Cercocebus chrysogaster
_Miopithecus ogouensis
_Trachypithecus delacouri
_Semnopithecus entellus entel
_Pygathrix nemaeus
_Pygathrix cinerea
_Semnopithecus hector
_Rhinopithecus brelichi
_Cercopithecus diana
_Cercopithecus mona
_Colobus polykomos
_Colobus guereza
_Hylobates agilis
_Hylobates muelleri
_Nomascus gabriellae
_Nomascus leucogenys
_Nomascus siki
_Symphalangus syndactylus
_Hylobates lar
_Gorilla gorilla
_Pan paniscus
_Homo sapiens


_Callicebus cupreus
_Saimiri sciureus
_Saimiri boliviensis bolivien
_Leontopithecus chrysomelas
_Saguinus midas
_Saguinus geoffroyi
_Saguinus oedipus
_Saguinus mystax
_Saguinus bicolor
_Aotus azarai
_Saguinus fuscicollis
_Cebus xanthosternos
_Cebus capucinus
_Cebus apella
_Cebus olivaceus
_Callithrix pygmaea
_Ateles geoffroyi
_Ateles fusciceps
_Ateles belzebuth hybridus
_Ateles belzebuth chamek
_Lagothrix cana
_Cacajao calvus
_Otolemur crassicaudatus
_Arctocebus calabarensis
_Nycticebus coucang
_Nycticebus bengalensis
_Propithecus tattersalli
_Propithecus verreauxi
_Hapalemur occidentalis
_Propithecus edwardsi
_Eulemur macaco
_Eulemur rufus
_Lepilemur septentrionalis
_Lepilemur dorsalis
_Lepilemur ankaranensis
_Lepilemur jamesi
_Cheirogaleus medius


_Tarsius bancanus
_Callicebus moloch
_Callicebus personatus
_Cacajao melanocephalus
_Brachyteles arachnoides
_Cebus nigritus robustus
_Saguinus imperator
_Callithrix aurita
_Alouatta caraya
_Galago senegalensis
_Pongo pygmaeus
_Saimiri oerstedii oerstedii
_Lagothrix lagotricha
_Ateles paniscus
_Ateles belzebuth
_Callithrix jacchus
_Callithrix geoffroyi
_Callithrix argentata argenta
_Callithrix penicillata
_Callimico goeldii
_Saguinus midas midas
_Aotus lemurinus griseimembra
_Aotus trivirgatus
_Chiropotes chiropotes
_Callicebus caligatus
_Callicebus donacophilus
_Pithecia pithecia
_Leontopithecus rosalia
_Alouatta sara
_Alouatta palliata
_Alouatta belzebul
_Saguinus labiatus

Code used in Scrawkov-Phy

(all code is stylistically minimal to decrease print-size. Properly stylized code can be found at the github under the user nickjfisk)

```
///ScrawkovPHY.java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;
/**
 *
 * @author J. Nick Fisk
 * Main class for the alignment free phylogeny algorithm, Scrawkov-Phy.
 */
public class ScrawkovPHY{
        //all the data from one file (as separated by newlines)
        public static ArrayList<String> allData=new ArrayList<String>();
        //the names of the taxa in order of appearance (to conserve sequence, name order)
        public static ArrayList<String> finchInOrder=new ArrayList<String>();
        //sequences in name-sequencce order in the desired format
        public static ArrayList<String> formatedSeqList=new ArrayList<String>();
        //ID to bird object pairings. Basically ID sequence pairs.
        public static HashMap<String, Bird> birdMap=new HashMap<String, Bird>();
        //contains all of the standard codons.
        public static ArrayList<String> allCodons=new ArrayList<String>();
        //contains all of the standard dinucs
        public static ArrayList<String> allDinucs=new ArrayList<String>();
        //key for going from codons to AA
        public static HashMap<String,String> codonAAMap=new HashMap<String,String>();
        //goes from AA to a list of possible codons
        public static HashMap<String,ArrayList<String>> AAcodonMap=new HashMap<String,
ArrayList<String>>();
        //has the scores for one gene tree.
        public static HashMap<String,HashMap<String,Double>>initScores=new
HashMap<String,HashMap<String,Double>>();
        //public static HashMap<String,HashMap<Integer,HashMap<String,Double>>> QHMMTri=new
HashMap<String,HashMap<Integer,HashMap<String,Double>>>();
        //public static HashMap<String,HashMap<Integer,HashMap<String,Double>>> QHMMDi=new
HashMap<String,HashMap<Integer,HashMap<String,Double>>>();
        public static HashMap<String,HashMap<String, Double>> QHMMTri_final=new
HashMap<String,HashMap<String, Double>>();
        public static HashMap<String,HashMap<String, Double>> QHMMDi_final=new
HashMap<String,HashMap<String, Double>>();
        public static HashMap<String,HashMap<String,HashMap<String,Double>>> allScoresByGene= new
HashMap<String,HashMap<String,HashMap<String,Double>>>();

        public static int k=-1; //kmer size. If negative (default), two will be used.
        private static boolean doNJ=true; //NJ tree desired
        private static String njOut=""; //name of njOut file
        public static boolean maxByPair=true; //default behaviour should be true
        private static boolean normalize=true; //default should be true
        private static String geneOrSpecies="gene"; //default to gene
        private static ArrayList<String> fastaFiles=new ArrayList<String>(); //keep track of all the files
for species trees
        private static String outputFile=null; //UPGMA output
        private static String usage="Usage: java ScrawkovPHY <treetype> <inputFasta> <optionalParams>";
        private static double weightGC=10;
```

```
private static double weightBias=.01;
private static double weightTriFreq=1;
private static double weightDiFreq=1;
private static double weightTriQHMM=.00005;
private static double weightDiQHMM=.00005;
private static boolean outputAll=false;
private static boolean bootSeq=false;
private static String outBoot=null;
private static boolean diag=false;


//Globally available parameters
/**
 * Parses the command line arguments and sets globals accordingly
 * @param params are the command line arguments in a arraylist
 */
private static void parseParams(ArrayList<String> params){
        //ask the magic counch
        //"Nothing."
        //THE COUNCH HAS SPOKEN!
        if(params.get(0).equals("gene")||params.get(0).equals("Gene")){
                geneOrSpecies="gene";
                params.remove(0);
                fastaFiles.add(params.get(0)); //only one fasta file
                params.remove(0);
                File f = new File(fastaFiles.get(0));
                if(!f.exists()){
                        System.err.println("File does not exist");
                        System.err.println(usage);
                        System.exit(1);}
                if(f.isDirectory()){
                        System.err.println("Input file is actually directory");
                        System.err.println(usage);
                        System.exit(1);}}
        else if(params.get(0).equals("species")||params.get(0).equals("Species")){
                geneOrSpecies="species";
                params.remove(0);
                if(params.isEmpty()){
                        System.err.println("No input files provided");
                        System.err.println("Exiting...");
                        System.err.println(usage);
                        System.exit(1);}
                if(params.get(0).equals("--folder")){
                        File folder=new File(params.get(1));
                        File[] listOfFiles=folder.listFiles();
                        for(int i=0; i<listOfFiles.length;i++){
                                if(listOfFiles[i].isFile()){
                                        fastaFiles.add(listOfFiles[i].getAbsolutePath());}}
                        params.remove(0);
                        params.remove(0);}
                String curFile=params.get(0);
                while((curFile.charAt(0)=='-')==false){
                        fastaFiles.add(curFile);
                        params.remove(0);
                        curFile=params.get(0);}
                for(String i:fastaFiles){
                        File f = new File(i);
                        if(!f.exists()){
                                System.err.println("File does not exist");
                                System.err.println(usage);
                                System.exit(1);}
                        if(f.isDirectory()){
                                System.err.println("Input file is actually directory");
                                System.err.println(usage);
                                System.exit(1);}}}
        else if(params.get(0).equals("--help")||params.get(0).equals("-h")){
                printHelp();
                System.exit(0);}
```

```
		else{
			System.err.println("Invalid tree type. Valid values are 'gene' and 'species'");
			System.err.println(usage);
			System.exit(1);}
		while(!params.isEmpty()){
			String curFlag=params.get(0);
			params.remove(0);
			if(params.isEmpty()){
				System.err.println("Unpaired flag "+ curFlag);
				System.err.println("Exiting...");
				System.err.println(usage);
				System.exit(1);}
			String curValue=params.get(0);
			params.remove(0);
			switch(curFlag){
				case "--maxByPair":
					switch(curValue){
						case "T":
							maxByPair=true;
							break;
						case "t":
							maxByPair=true;
							break;
						case "true":
							maxByPair=true;
							break;
						case "True":
							maxByPair=true;
							break;
						case "TRUE":
							maxByPair=true;
							break;
						case "F":
							maxByPair=false;
							break;
						case "f":
							maxByPair=false;
							break;
						case "false":
							maxByPair=false;
							break;
						case "False":
							maxByPair=false;
							break;
						case "FALSE":
							maxByPair=false;
							break;
						default:
							System.err.println("Invalid value for
maxByPair");
							System.err.println("Exiting...");
							System.err.println(usage);
							System.exit(1);
							break;
					}
					break;
				case "--outputFile":
					outputFile=curValue;
					break;
				case "-o":
					print(curValue);
					outputFile=curValue;
					break;
				case "--kSize":
					k=Integer.parseInt(curValue);
					break;
				case "--doNJ":
					switch(curValue){
```

```
                        case "T":
                                doNJ=true;
                                break;
                        case "t":
                                doNJ=true;
                                break;
                        case "true":
                                doNJ=true;
                                break;
                        case "True":
                                doNJ=true;
                                break;
                        case "TRUE":
                                doNJ=true;
                                break;
                        case "F":
                                doNJ=false;
                                break;
                        case "f":
                                doNJ=false;
                                break;
                        case "false":
                                doNJ=false;
                                break;
                        case "False":
                                doNJ=false;
                                break;
                        case "FALSE":
                                doNJ=false;
                                break;
                        default:
                                System.err.println("Invalid value for doNJ");
                                System.err.println("Exiting...");
                                System.err.println(usage);
                                System.exit(1);
                                break;}
                        break;
                case "--diagMat":
                        switch(curValue){
                        case "T":
                                diag=true;
                                break;
                        case "t":
                                diag=true;
                                break;
                        case "true":
                                diag=true;
                                break;
                        case "True":
                                diag=true;
                                break;
                        case "TRUE":
                                diag=true;
                                break;
                        case "F":
                                diag=false;
                                break;
                        case "f":
                                diag=false;
                                break;
                        case "false":
                                diag=false;
                                break;
                        case "False":
                                diag=false;
                                break;
                        case "FALSE":
                                diag=false;
```

```
                        break;
                default:
                        System.err.println("Invalid value for diagMat");
                        System.err.println("Exiting...");
                        System.err.println(usage);
                        System.exit(1);
                        break;}
                break;
        case "--njOut":
                njOut=curValue;
                break;
        case "--bootSeq":
                switch(curValue){
                        case "T":
                                bootSeq=true;
                                break;
                        case "t":
                                bootSeq=true;
                                break;
                        case "true":
                                bootSeq=true;
                                break;
                        case "True":
                                bootSeq=true;
                                break;
                        case "TRUE":
                                bootSeq=true;
                                break;
                        case "F":
                                bootSeq=false;
                                break;
                        case "f":
                                bootSeq=false;
                                break;
                        case "false":
                                bootSeq=false;
                                break;
                        case "False":
                                bootSeq=false;
                                break;
                        case "FALSE":
                                bootSeq=false;
                                break;
                        default:
                                System.err.println("Invalid value for bootSeq");
                                System.err.println("Exiting...");
                                System.err.println(usage);
                                System.exit(1);
                                break;}
                        break;

        case "--outputAll":
                switch(curValue){
                        case "T":
                                outputAll=true;
                                break;
                        case "t":
                                outputAll=true;
                                break;
                        case "true":
                                outputAll=true;
                                break;
                        case "True":
                                outputAll=true;
                                break;
                        case "TRUE":
                                outputAll=true;
                                break;
```

```
                                        case "F":
                                                outputAll=false;
                                                break;
                                        case "f":
                                                outputAll=false;
                                                break;
                                        case "false":
                                                outputAll=false;
                                                break;
                                        case "False":
                                                outputAll=false;
                                                break;
                                        case "FALSE":
                                                outputAll=false;
                                                break;
                                        default:
                                                System.err.println("Invalid value for
outputAll");
                                                System.err.println("Exiting...");
                                                System.err.println(usage);
                                                System.exit(1);
                                                break;}
                                        break;
                                case "--normalize":
                                        switch(curValue){
                                        case "T":
                                                normalize=true;
                                                break;
                                        case "t":
                                                normalize=true;
                                                break;
                                        case "true":
                                                normalize=true;
                                                break;
                                        case "True":
                                                normalize=true;
                                                break;
                                        case "TRUE":
                                                normalize=true;
                                                break;
                                        case "F":
                                                normalize=false;
                                                break;
                                        case "f":
                                                normalize=false;
                                                break;
                                        case "false":
                                                normalize=false;
                                                break;
                                        case "False":
                                                normalize=false;
                                                break;
                                        case "FALSE":
                                                normalize=false;
                                                break;
                                        default:
                                                System.err.println("Invalid value for
normalize");
                                                System.err.println("Exiting...");
                                                System.err.println(usage);
                                                System.exit(1);
                                                break;}
                                        break;
                                        //--weightGC --weightCodon --weightTri --weightDi
                                case "--weightGC":
                                        if(isDouble(curValue)){
                                                weightGC=Double.parseDouble(curValue);}
                                        else{
```

```
                                        System.err.println("Invalid value for weightGC");
                                        System.err.println("Exiting...");
                                        System.err.println(usage);
                                        System.exit(1);}
                                break;
                        case "--weightBias":
                                if(isDouble(curValue)){
                                        weightBias=Double.parseDouble(curValue);}
                                else{
                                        System.err.println("Invalid value for weightBias");
                                        System.err.println("Exiting...");
                                        System.err.println(usage);
                                        System.exit(1);}
                                break;
                        case "--weightTriQHMM":
                                if(isDouble(curValue)){
                                        weightTriQHMM=Double.parseDouble(curValue);}
                                else{
                                        System.err.println("Invalid value for weightTriQHMM");
                                        System.err.println("Exiting...");
                                        System.err.println(usage);
                                        System.exit(1);}
                                break;
                        case "--weightDiQHMM":
                                if(isDouble(curValue)){
                                        weightDiQHMM=Double.parseDouble(curValue);}
                                else{
                                        System.err.println("Invalid value for weightDiQHMM");
                                        System.err.println("Exiting...");
                                        System.err.println(usage);
                                        System.exit(1);}
                                break;
                        case "--weightTriFreq":
                                if(isDouble(curValue)){
                                        weightTriFreq=Double.parseDouble(curValue);}
                                else{
                                        System.err.println("Invalid value for weightTriFreq");
                                        System.err.println("Exiting...");
                                        System.err.println(usage);
                                        System.exit(1);}
                                break;
                        case "--weightDiFreq":
                                if(isDouble(curValue)){
                                        weightDiFreq=Double.parseDouble(curValue);}
                                else{
                                        System.err.println("Invalid value for weightDiFreq");
                                        System.err.println("Exiting...");
                                        System.err.println(usage);
                                        System.exit(1);}
                                break;
                        case "--help":
                                printHelp();
                                System.exit(0);
                                break;
                        default:
                                System.err.println("Invalid flag");
                                System.err.println("Exiting...");
                                System.err.println(usage);
                                System.exit(1);
                                break;}}}

/**
 * The main function of the program, calling gene or species tree methods.
 * It also instantiates all of the reference data structures.
 * @param args command line arguments
 * @throws IOException
 */
public static void main(String[] args) throws IOException {
```

```
		//populate some internal data structures
		popAllCodons();
		popAllDinucs();
		popCodonMap();
		popAAtoCodonMap();
		//get the parameters into an arrayList
		ArrayList<String> params=new ArrayList<String>();
		int parCount=0;
		while(parCount<args.length){
			params.add(args[parCount]);
			parCount+=1;}
		//parse the params
		parseParams(params);
		if(geneOrSpecies.equals("gene")||geneOrSpecies.equals("Gene")){
			doGeneTree(params,fastaFiles.get(0));//get fasta for now
			System.exit(0);}
		else if(geneOrSpecies.equals("species")||geneOrSpecies.equals("Species")){
			doSpeciesTree(params);
			ArrayList<String> allGeneTrees=new ArrayList<String>(allScoresByGene.keySet());
			ArrayList<String> allOrgs=new ArrayList<String>();
			for(String i: allGeneTrees){
				allOrgs.addAll(allScoresByGene.get(i).keySet());}
			Set<String> allOrgsSet=new HashSet<String>( allOrgs);
			allOrgs=new ArrayList<String>(allOrgsSet);
			HashMap<String,HashMap<String,Double>>finScores=new
HashMap<String,HashMap<String,Double>>();
			HashMap<String,HashMap<String,Integer>> countMap=new
HashMap<String,HashMap<String,Integer>>();
			for(String i: allOrgs){
				HashMap<String,Double>tempu=new HashMap<String,Double>();
				HashMap<String,Integer>tempu2=new HashMap<String,Integer>();
				for(String j: allOrgs){
					if(i.equals(j)==false){
						tempu.put(j, 0.0);
						tempu2.put(j,0);}}
				finScores.put(i, tempu);
				countMap.put(i,tempu2);}
			//only valid for species tree construction
			if(normalize==true){
				double max=getMaxScore();
				double miniMax=Double.MIN_VALUE;
				for(String i: allScoresByGene.keySet()){
					for(String j: allScoresByGene.get(i).keySet()){
						for(String k: allScoresByGene.get(i).get(j).keySet()){
							if(miniMax<allScoresByGene.get(i).get(j).get(k)){
								//find the largest element in the map

	miniMax=allScoresByGene.get(i).get(j).get(k);}}}
					//normalize everything to be proportional
					for(String j: allScoresByGene.get(i).keySet()){
						for(String k: allScoresByGene.get(i).get(j).keySet()){
							allScoresByGene.get(i).get(j).put(k,
(allScoresByGene.get(i).get(j).get(k)*(max/miniMax)));}}}}
			//sum all the scores
			for(String j: allScoresByGene.keySet()){
				for(String q: allScoresByGene.get(j).keySet()){
					for(String h: allScoresByGene.get(j).get(q).keySet()){
						finScores.get(q).put(h,
finScores.get(q).get(h)+allScoresByGene.get(j).get(q).get(h));
						countMap.get(q).put(h, countMap.get(q).get(h)+1);}}}
			//normalize by number of times a species appears.
			ArrayList<String>allSpeciesOccurances=new ArrayList<String>();
			for(String i: allScoresByGene.keySet()){
				allSpeciesOccurances.addAll(allScoresByGene.get(i).keySet());}
			for(String i: countMap.keySet()){
				for(String j: countMap.get(i).keySet()){
					if(countMap.get(i).get(j)!=0){
```

```
                                        finScores.get(i).put(j,
(finScores.get(i).get(j)/countMap.get(i).get(j)));}}}
                        initScores=finScores;
                        //build the UPGMA Tree
                        HashMap<String,HashMap<String,Double>> wkMap=new
HashMap<String,HashMap<String,Double>>(initScores);
                        HashMap<Node,HashMap<Node,Double>>nodeDist=new
HashMap<Node,HashMap<Node,Double>>();
                        for(String s: initScores.keySet()){
                                Node sn=new Node(s);
                                nodeDist.put(sn, new HashMap<Node,Double>());
                                for(String r: initScores.keySet()){
                                        if(s.equals(r)==false){
                                                Node rn=new Node(r);
                                                nodeDist.get(sn).put(rn, initScores.get(s).get(r));}}}
                        ArrayList<String> allNames=new ArrayList<String>(wkMap.keySet());
                        Node curMinNode1=null;
                        Node curMinNode2=null;
                        double curMinScore=Double.MAX_VALUE;
                        HashMap<String,Boolean>inNode=new HashMap<String,Boolean>();
                        for(String i: initScores.keySet()){
                                inNode.put(i, false);}
                        ArrayList<Node>nodeList=new ArrayList<Node>();
                        while(nodeDist.keySet().size()>1){
                                curMinScore=Double.MAX_VALUE;
                                for(Node i: nodeDist.keySet()){
                                        for(Node ent:nodeDist.get(i).keySet()){
                                                if(nodeDist.get(i).get(ent)<curMinScore){
                                                        curMinNode1=i;
                                                        curMinNode2=ent;
                                                        curMinScore=nodeDist.get(i).get(ent);}}}
                                Node nw2=new Node((Node)curMinNode1,(Node)curMinNode2, curMinScore);
                                nodeDist.remove(curMinNode1);
                                nodeDist.remove(curMinNode2);
                                for(Node n: nodeDist.keySet()){
                                        if(nodeDist.get(n).containsKey(curMinNode1)){
                                                nodeDist.get(n).remove(curMinNode1);}
                                        if(nodeDist.get(n).containsKey(curMinNode2)){
                                                nodeDist.get(n).remove(curMinNode2);}}
                                nodeDist.put(nw2, new HashMap<Node,Double>());
                                for(Node n: nodeDist.keySet()){
                                        if(n.equals(nw2)==false){
                                                //calc dist and put
                                                double calcDist=Node.calcDistance(nw2, n);
                                                nodeDist.get(nw2).put(n, calcDist);
                                                nodeDist.get(n).put(nw2, calcDist);}}}
                        for(Node fin: nodeDist.keySet()){
                                print("Printing UPGMA Tree");
                                print(fin.newick+";");
                                if(outputFile!=null){
                                        PrintWriter out = new PrintWriter(outputFile+".newick");
                                        out.print(fin.newick+";");
                                        out.close();}}}}

        /**
         * Constructs a single gene tree from a single fasta file
         * @param params, the command line arguments as an arraylist
         * @param file, the fasta file
         * @throws IOException
         */
        public static void doGeneTree(ArrayList<String> params, String file) throws IOException{
                //initialize data structures
                ArrayList<String> allData=new ArrayList<String>();
                formatedSeqList=new ArrayList<String>();
                HashMap<String, Bird> birdMap=new HashMap<String, Bird>();
                initScores=new HashMap<String,HashMap<String,Double>>();
                FileReader fr = new FileReader(new File(file));
                BufferedReader br = new BufferedReader(fr);
```

```
                String line;
                //read in sequence
                while((line = br.readLine()) != null){
                    line = line.trim(); // remove leading and trailing whitespace
                    allData.add(line);}
                String holdMe = "";
                for (String element : allData) {
                        if (element.charAt(0) == '>') {
                                if (holdMe != "") {
                                        formatedSeqList.add(holdMe);
                                        holdMe = "";}
                                formatedSeqList.add(element);}
                        else {
                                holdMe += element;}}
                formatedSeqList.add(holdMe);
                fr.close();
                //make Bird Objects
                for(int i=0; i<formatedSeqList.size(); i++){
                        String id=formatedSeqList.get(i);
                        String seq=formatedSeqList.get(i+1);
                        Bird newbie=new Bird(id,seq);
                        birdMap.put(id,newbie);
                        i+=1;}
                if(bootSeq==true){
                        genBootstrapSeqs(true, 3, .85, birdMap);}
                //populate hidden markov model measures
                QHMMALL(birdMap);
                HashMap<String,ArrayList<String>>keepTrack=new HashMap<String,ArrayList<String>>();
                //score all
                for(String id: birdMap.keySet()){
                        for(String id2:birdMap.keySet()){
                                if(id.equals(id2)==false){
                                        if(keepTrack.containsKey(id)){
                                                if(keepTrack.get(id).contains(id2)==false){
                                                        Double
d=computeANDcombine(birdMap.get(id),birdMap.get(id2));
                                                        if(initScores.containsKey(id)==false){
                                                                initScores.put(id, new
HashMap<String,Double>());}
                                                        initScores.get(id).put(id2, d);}}
                                        else{
                                                keepTrack.put(id, new ArrayList<String>());
                                                keepTrack.get(id).add(id2);
                                                initScores.put(id, new HashMap<String, Double>());
                                                if(keepTrack.containsKey(id2)==false){
                                                        keepTrack.put(id2, new ArrayList<String>());
                                                        keepTrack.get(id2).add(id);
                                                        Double d=
computeANDcombine(birdMap.get(id),birdMap.get(id2));
                                                        initScores.get(id).put(id2, d);}}}}}
                for(String i: initScores.keySet()){
                        for(String j: initScores.keySet()){
                                if(i.equals(j)==false){
                                        if(initScores.get(i).get(j)==null){
                                                initScores.get(i).put(j, initScores.get(j).get(i));}}}}
                /////CLUSTERING AND TREE BUILDING NEXT!/////
                HashMap<String,HashMap<String,Double>> wkMap=new
HashMap<String,HashMap<String,Double>>(initScores);
                HashMap<Node,HashMap<Node,Double>>nodeDist=new HashMap<Node,HashMap<Node,Double>>();
                for(String s: initScores.keySet()){
                        Node sn=new Node(s);
                        nodeDist.put(sn, new HashMap<Node,Double>());
                        for(String r: initScores.keySet()){
                                if(s.equals(r)==false){
                                        Node rn=new Node(r);
                                        nodeDist.get(sn).put(rn, initScores.get(s).get(r));}}}
                toMatrix(nodeDist);
                Node curMinNode1=null;
```

```
                Node curMinNode2=null;
                double curMinScore=Double.MAX_VALUE;
                while(nodeDist.keySet().size()>1){
                        curMinScore=Double.MAX_VALUE;
                        for(Node i: nodeDist.keySet()){
                                for(Node ent:nodeDist.get(i).keySet()){
                                        if(nodeDist.get(i).get(ent)<curMinScore){
                                                curMinNode1=i;
                                                curMinNode2=ent;
                                                curMinScore=nodeDist.get(i).get(ent);}}}
                        Node nw2=new Node((Node)curMinNode1,(Node)curMinNode2, curMinScore);
                        nodeDist.remove(curMinNode1);
                        nodeDist.remove(curMinNode2);
                        for(Node n: nodeDist.keySet()){
                                if(nodeDist.get(n).containsKey(curMinNode1)){
                                        nodeDist.get(n).remove(curMinNode1);}
                                if(nodeDist.get(n).containsKey(curMinNode2)){
                                        nodeDist.get(n).remove(curMinNode2);}}
                        nodeDist.put(nw2, new HashMap<Node,Double>());
                        for(Node n: nodeDist.keySet()){
                                if(n.equals(nw2)==false){
                                        //calc dist and put
                                        double calcDist=Node.calcDistance(nw2, n);
                                        nodeDist.get(nw2).put(n, calcDist);
                                        nodeDist.get(n).put(nw2, calcDist);}}}
                for(Node fin: nodeDist.keySet()){
                        print("Printing UPGMA Tree");
                        print(fin.newick+";");
                        if(outputFile!=null){
                                if(geneOrSpecies.equals("gene")){
                                        PrintWriter out = new PrintWriter(outputFile);
                                        out.print(fin.newick+";");
                                        out.close();}}
                        if(outputAll==true){
                                if(geneOrSpecies.equals("species")){
                                        File f = new File("geneTreesForSpeciesTree");
                                        if (f.exists() && f.isDirectory()) {
                                           PrintWriter out=new PrintWriter(file+"gene_tree.newick");
                                           out.print(fin.newick);
                                           out.close();}
                                        else{
                                                f.mkdir();
                                                PrintWriter out=new
PrintWriter(f+file+"_gene_tree.newick");
                                                out.print(fin.newick);
                                                out.close();}}}}}
        /**
         * Formats a HashMap of Distances to a PHYLIP formatted distance matrix
         * @param nodeDist
         */
        private static void toMatrix(HashMap<Node, HashMap<Node, Double>> nodeDist) {
                ArrayList<Node> nodes=new ArrayList<Node>(nodeDist.keySet());
                int matSize=nodes.size();
                double [][] scoreMatrix=new double[matSize][matSize];
                ArrayList<String> stringsToWrite=new ArrayList<String>();
                ArrayList<String> namesList=new ArrayList<String>();
                for(Node n : nodes){
                        String temp=n.names.get(0).substring(1, n.names.get(0).length());
                        namesList.add(temp);
                        temp=temp+"          ";
                        if(temp.length()>10){
                                temp=temp.substring(0,10);}
                        stringsToWrite.add(temp+" ");}
                try {
                        PrintWriter write=new PrintWriter("distanceMatrix.txt","UTF-8");
                        write.println(" "+matSize);
                        //i is row and j is col
                        for(int i=0; i< nodes.size();i++){
```

```
                                Node n=nodes.get(i);
                                for(int j=0; j<nodes.size();j++){
                                        if(i==j){
                                                scoreMatrix[i][j]=0.0;}
                                        else{
                                                Node n2=nodes.get(j);
                                                scoreMatrix[i][j]=Node.calcDistance(n, n2);}}}
                        //diagonal formmat of phylip distance matrix
                        if(diag){
                                int count=matSize;
                                for(int i=0; i<matSize;i++){
                                        String tmpStr=stringsToWrite.get(i);
                                        for(int j=0+count;j<matSize;j++){
                                                if(j!=matSize-1){
                                                        tmpStr=tmpStr+scoreMatrix[i][j-count]+" ";}
                                                else{
                                                        tmpStr=tmpStr+scoreMatrix[i][j-count];}}
                                        count=count-1;
                                        write.println(tmpStr);}}
                        else{
                                for(int i=0;i<matSize;i++){
                                        String tempStr=stringsToWrite.get(i);
                                        for(int j=0;j<matSize;j++){
                                                if(j!=matSize-1){
                                                        tempStr=tempStr+scoreMatrix[i][j]+" ";}
                                                else{
                                                        write.println(tempStr+scoreMatrix[i][j]);}}}}
                        write.close();
                } catch (FileNotFoundException | UnsupportedEncodingException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();}
                if(doNJ==true){
                        doNJTree(scoreMatrix,nodeDist,matSize,namesList);}}
/**
 * Constructs a neighbor joining tree in newick format
 * @param scoreMatrix
 * @param nodeDist
 * @param matSize
 * @param namesList
 */
        public static void doNJTree(double[][] scoreMatrix, HashMap<Node, HashMap<Node, Double>>
nodeDist,int matSize,ArrayList<String> namesList){
                //i is row, j is column
                HashMap<NJNode,HashMap<NJNode,Double>> NJNodeMap=new
HashMap<NJNode,HashMap<NJNode,Double>>();
                ArrayList<NJNode> NJList=new ArrayList<NJNode>();
                ArrayList<ArrayList<Double>> M=new ArrayList<ArrayList<Double>>();
                ArrayList<ArrayList<Double>> Mp=new ArrayList<ArrayList<Double>>();
                for(int i=0; i<namesList.size(); i++){
                        NJNode tmp=new NJNode(namesList.get(i));
                        NJList.add(i,tmp);}
                for(int i=0; i<NJList.size();i++){
                        for(int j=0;j<NJList.size();j++){
                                if(i!=j){
                                        if(NJNodeMap.containsKey(NJList.get(i))==false){
                                                NJNodeMap.put(NJList.get(i), new
HashMap<NJNode,Double>());}

        if(NJNodeMap.get(NJList.get(i)).containsKey(NJList.get(j))==false){
                                                NJNodeMap.get(NJList.get(i)).put(NJList.get(j),
scoreMatrix[i][j]);}}}}
                while(NJNodeMap.keySet().size()>2){
                        HashMap<NJNode,HashMap<NJNode,Double>> Q=new
HashMap<NJNode,HashMap<NJNode,Double>>();
                        HashMap<NJNode,Double> T=new HashMap<NJNode,Double>();
                        //compute T, the sum total of rows
                        double totalRow=0;
                        for(NJNode i: NJNodeMap.keySet()){
```

```
                                for(NJNode j: NJNodeMap.keySet()){
                                        if(!i.equals(j)){
                                                totalRow+=NJNodeMap.get(i).get(j);}}
                                        T.put(i, totalRow);
                                        totalRow=0;}
                                NJNode curMinNode1=null;
                                NJNode curMinNode2=null;
                                double curMinScore=Double.POSITIVE_INFINITY;
                                //compute Q, the transformed matrix for neighbor joining minimal selection
                                for(NJNode i: NJNodeMap.keySet()){
                                        Q.put(i, new HashMap<NJNode,Double>());
                                        for(NJNode j:NJNodeMap.keySet()){
                                                if(!i.equals(j)){
                                                        if(Q.containsKey(j)==false){
                                                                Q.put(j, new HashMap<NJNode, Double>());}
                                                        double tmp=(((NJNodeMap.keySet().size()-
2)*NJNodeMap.get(i).get(j))-T.get(i)-T.get(j));
                                                        Q.get(i).put(j,tmp);
                                                        Q.get(j).put(i, tmp);
                                                        if(tmp<curMinScore){
                                                                if(!i.equals(j)){
                                                                        curMinNode1=i;
                                                                        curMinNode2=j;
                                                                        curMinScore=tmp;}}}}}
                                Q.clear();
                                //Find branch lengths
                                double deltaIJ=((T.get(curMinNode1)-T.get(curMinNode2))/(NJNodeMap.size()-2));
                                double LLi=0.5*(NJNodeMap.get(curMinNode1).get(curMinNode2)+deltaIJ);
                                double LLj=0.5*(NJNodeMap.get(curMinNode1).get(curMinNode2)-deltaIJ);
                                //make new node
                                NJNode tempuNode=new NJNode(curMinNode1,curMinNode2,LLi,LLj);
                                //compute new distances to other nodes and add to map.
                                double joinDist=0;
                                HashMap<NJNode, Double> joinMap=new HashMap<NJNode,Double>();
                                for(NJNode n: NJNodeMap.keySet()){
                                        if(!n.equals(curMinNode1)){
                                                if(!n.equals(curMinNode2)){

                joinDist=(NJNodeMap.get(n).get(curMinNode1)+NJNodeMap.get(n).get(curMinNode2)-
NJNodeMap.get(curMinNode1).get(curMinNode2))/2;
                                                        joinMap.put(n, joinDist);}}}
                                //remove the two old nodes...
                                NJNodeMap.remove(curMinNode1);
                                NJNodeMap.remove(curMinNode2);
                                for(NJNode n: NJNodeMap.keySet()){
                                        if(NJNodeMap.get(n).containsKey(curMinNode1)){
                                                NJNodeMap.get(n).remove(curMinNode1);}
                                        if(NJNodeMap.get(n).containsKey(curMinNode2)){
                                                NJNodeMap.get(n).remove(curMinNode2);}}
                                //add new nodes
                                for(NJNode n: NJNodeMap.keySet()){
                                        NJNodeMap.get(n).put(tempuNode, joinMap.get(n));}
                                NJNodeMap.put(tempuNode, joinMap);}
                        ArrayList<NJNode>tmp2=new ArrayList<NJNode>(NJNodeMap.keySet());
                        NJNode fin=tmp2.get(0);
                        NJNode fin2=tmp2.get(1);
                        print("Printing NJ Tree");
                        String newickNJTree="("+fin2.newick+":"+NJNodeMap.get(fin).get(fin2)+","+fin.newick+");";
                        print(newickNJTree);
                        if(njOut.equals("")==false){
                                PrintWriter write;
                                try {
                                        write = new PrintWriter(njOut,"UTF-8");
                                        write.write(newickNJTree);
                                        write.close();
                                } catch (FileNotFoundException | UnsupportedEncodingException e) {
                                        e.printStackTrace();}}
                        else{
```

```
                        PrintWriter write;
                        try {
                                write = new PrintWriter("NJ_Tree.nwk","UTF-8");
                                write.write(newickNJTree);
                                write.close();
                        } catch (FileNotFoundException | UnsupportedEncodingException e) {
                                e.printStackTrace();}}}

        //START SPECIES TREE CONSTRUCTION (resumed in main for adaptive scope reasons)////
        /**
         * Scores every gene tree needed for the construction of a species tree
         * by calling the doGeneTree method and storing the result in a global hashmap
         * @param params parameters in ArrayList that will affect the settings of the tree building
algorithm
         * @throws IOException
         */
        public static void doSpeciesTree(ArrayList<String>params) throws IOException{
                for(String name : fastaFiles){
                        doGeneTree(params,name);
                        allScoresByGene.put(name, initScores);}
                for(String o : allScoresByGene.keySet()){
                        HashMap<String, HashMap<String,Double>>temp=allScoresByGene.get(o);
                        print(temp.keySet());
                        print("");}}
        /**
         * Finds the largest score of any tree for any gene
         * Used only in species tree method with scaling enabled
         * @return a double of the highest score observed
         */
        public static double getMaxScore(){
                double curMax=Double.MIN_VALUE;
                for(String i: allScoresByGene.keySet()){
                        for(String j: allScoresByGene.get(i).keySet()){
                                for(String k: allScoresByGene.get(i).get(j).keySet()){
                                        if(allScoresByGene.get(i).get(j).get(k)>curMax){
                                                curMax=allScoresByGene.get(i).get(j).get(k);}}}}
                return curMax;}

        /**
         * Calculates the scores for each of the components of
         * the algorithm and combines them to get a composite score
         * @param bird1 taxa 1 to get the score between
         * @param bird2 taxa 2 to get the score between
         * @return the composite score
         */
        public static double computeANDcombine(Bird bird1, Bird bird2){
                return
((scoreBirdCodon(bird1,bird2)*weightTriFreq)+(scoreBirdDinuc(bird1,bird2)*weightDiFreq)+
                +(scoreGC(bird1,bird2)*weightGC)+(scoreCodonBias(bird1,bird2)*weightBias)+

        (QHMMTri_final.get(bird1.id).get(bird2.id)*weightTriQHMM)+(QHMMDi_final.get(bird1.id).get(bird2.id
)*weightDiQHMM));
        }

        /**
         * Computes the score for the difference in codon usage (codon bias)
         * between two taxa.
         * @param bird1 taxa 1
         * @param bird2 taxa 2
         * @return the score, stored as a double
         */
        public static double scoreCodonBias(Bird bird1, Bird bird2){
                double totalScore=0.0;
                double protScore=0.0;
                int numPotCodons=0;
                double scoreLogCheck=0.0;
                for(String aa1:AAcodonMap.keySet()){
                        protScore=0;
```

```
                        if(bird1.biasMap.containsKey(aa1)==false){
                                //bird one doesnt, but bird 2 does
                                if(bird2.biasMap.containsKey(aa1)==true){
                                        numPotCodons=bird2.biasMap.get(aa1).keySet().size();
                                        for(String cod:bird2.biasMap.get(aa1).keySet()){

        protScore+=Math.abs(log2(bird2.biasMap.get(aa1).get(cod)/numPotCodons));}
                                        totalScore+=protScore;}
                                //neither do
                                else{
                                        //pass, rather, add 0 to total.
                                }}
                        //bird1 has the aa1
                        else{
                                //bird 2 doesn't
                                if(bird2.biasMap.containsKey(aa1)==false){
                                        numPotCodons=bird1.biasMap.get(aa1).keySet().size();
                                        for(String cod:bird1.biasMap.get(aa1).keySet()){

        protScore+=Math.abs(log2(bird1.biasMap.get(aa1).get(cod)/numPotCodons));}
                                        totalScore+=protScore;}
                                //they both have aa of interest
                                else{
                                        Set<String> unionCodons=new HashSet<String>();
                                        unionCodons.addAll(bird1.biasMap.get(aa1).keySet());
                                        unionCodons.addAll(bird2.biasMap.get(aa1).keySet());
                                        numPotCodons=unionCodons.size();
                                        for(String cod:unionCodons){
                                                //bird 2 doesn't have the codon
                                                if(bird2.biasMap.get(aa1).containsKey(cod)==false){
                                                        //that means bird one must!

        protScore+=Math.abs(log2(bird1.biasMap.get(aa1).get(cod)/numPotCodons));}
                                                //bird two does have the codon
                                                else{
                                                        //check if bird one does...
                                                        //no?

        if(bird1.biasMap.get(aa1).containsKey(cod)==false){

        protScore+=Math.abs(log2(bird2.biasMap.get(aa1).get(cod)/numPotCodons));}
                                                        //yes, they both do
                                                        else{

        scoreLogCheck=Math.abs(bird1.biasMap.get(aa1).get(cod)-
bird2.biasMap.get(aa1).get(cod)/numPotCodons);
                                                                if(scoreLogCheck!=0){

        protScore+=Math.abs(log2(scoreLogCheck));}
                                                                else{
                                                                        //add zero
                                                                }}}}
                                        totalScore+=protScore;}}}
                totalScore=totalScore/20;
                return totalScore;}

        /**
         * Calculates the score for a difference in codon transition probabilities between
         * two taxa for a given gene.
         * @param bird1 taxa 1
         * @param bird2 taxa 2
         * @return score, stored as a double
         */
        public static double scoreBirdCodon(Bird bird1, Bird bird2){
                double bestDist=Double.MAX_VALUE;
                for(int i: bird1.codonFrequencies.keySet()){
                        double score1=0.0;
                        double score2=0.0;
```

```
                        double distScore=0.0;
                        for(String c1:allCodons){
                                for(String c2:allCodons){
                                        if(bird1.codonFrequencies.get(i).containsKey(c1)==false){
                                                score1=0.0;}
                                        else{

        if(bird1.codonFrequencies.get(i).get(c1).containsKey(c2)==false){
                                                        score1=0.0;}
                                                else{

        score1=bird1.codonFrequencies.get(i).get(c1).get(c2);}}
                                        if(bird2.codonFrequencies.get(i).containsKey(c1)==false){
                                                score2=0.0;}
                                        else{

        if(bird2.codonFrequencies.get(i).get(c1).containsKey(c2)==false){
                                                        score2=0.0;}
                                                else{

        score2=bird2.codonFrequencies.get(i).get(c1).get(c2);}}
                                        distScore+=Math.abs(score1-score2);}}
                        if(distScore<bestDist){
                                bestDist=distScore;}}
                return bestDist;}

        /**
         * Constructs and computes the scores for the QHMM for both Tri and Di nucleotides
         * The results are not returned--rather, stored as a feature in the inputted birdMap.
         * @param birdMap, the internal data structure used to calculate QHMM
         */
        public static void QHMMALL(HashMap<String, Bird> birdMap){
                HashMap<String,HashMap<Integer,HashMap<String,Double>>> QHMMTri=new
HashMap<String,HashMap<Integer,HashMap<String,Double>>>();
                HashMap<String,HashMap<Integer,HashMap<String,Double>>> QHMMDi=new
HashMap<String,HashMap<Integer,HashMap<String,Double>>>();
                int[] iterlist=new int[3];
                iterlist[0]=1;
                iterlist[1]=2;
                iterlist[2]=3;
                for(String name: birdMap.keySet()){
                        HashMap<Integer,HashMap<String,Double>> tempu=new
HashMap<Integer,HashMap<String,Double>>();
                        for(int i: iterlist){
                                HashMap<String,Double>tempu2=new HashMap<String,Double>();
                                        for(String name2: birdMap.keySet()){
                                                if(name.equals(name2)==false){
                                                        tempu2.put(name2, 0.0);}
                                        tempu.put(i, tempu2);}}
                        QHMMTri.put(name, tempu);}
                for(String name: QHMMTri.keySet()){
                        Bird tempuBird=birdMap.get(name);
                        String tseq=tempuBird.sequence;
                        for(int i: iterlist){
                                while(tseq.length()>5){
                                        String codon1=tseq.substring(0, 3);
                                        String codon2=tseq.substring(3, 6);
                                        for(String name2: birdMap.keySet()){
                                                if(!name2.equals(name)){

        if(birdMap.get(name2).codonFrequencies.get(i).containsKey(codon1)){

        if(birdMap.get(name2).codonFrequencies.get(i).get(codon1).containsKey(codon2)){

        QHMMTri.get(name).get(i).put(name2,
(QHMMTri.get(name).get(i).get(name2))+(Math.abs(log2(birdMap.get(name2).codonFrequencies.get(i).get(codon1
).get(codon2)))));}}}}
                                        tseq=tseq.substring(3);}
```

```
                                tseq=tempuBird.sequence;}
                        for(String name2: QHMMTri.keySet()){
                                if(name.equals(name2)==false){
                                        double score1=QHMMTri.get(name).get(1).get(name2);
                                        double score2=QHMMTri.get(name).get(2).get(name2);
                                        double score3=QHMMTri.get(name).get(3).get(name2);
                                        double bestScore=Math.min(Math.min(score1, score2), score3);
                                        if(QHMMTri_final.containsKey(name)==false){
                                                HashMap<String,Double> ntempu=new
HashMap<String,Double>();
                                                ntempu.put(name2, bestScore);
                                                QHMMTri_final.put(name, ntempu);}
                                        else{
                                                QHMMTri_final.get(name).put(name2, bestScore);}}}}
                int numDo=2;
                if(k>1){
                        numDo=k;}
                iterlist=new int[numDo];
                for(int m=0;m<numDo;m++){
                        iterlist[m]=m+1;}
                for(String name: birdMap.keySet()){
                        HashMap<Integer,HashMap<String,Double>> tempu=new
HashMap<Integer,HashMap<String,Double>>();
                        for(int i: iterlist){
                                HashMap<String,Double>tempu2=new HashMap<String,Double>();
                                        for(String name2: birdMap.keySet()){
                                                if(name.equals(name2)==false){
                                                        tempu2.put(name2, 0.0);}
                                        tempu.put(i, tempu2);}}
                        QHMMDi.put(name, tempu);}
                for(String name: QHMMDi.keySet()){
                        Bird tempuBird=birdMap.get(name);
                        String tseq=tempuBird.sequence;
                        int whileLen=(2*numDo)-1;
                        for(int i: iterlist){
                                while(tseq.length()>whileLen){
                                        String codon1=tseq.substring(0, numDo);
                                        String codon2=tseq.substring(numDo, numDo+numDo);
                                        for(String name2: birdMap.keySet()){
                                                if(!name2.equals(name)){

        if(birdMap.get(name2).dinucFrequencies.get(i).containsKey(codon1)){

        if(birdMap.get(name2).dinucFrequencies.get(i).get(codon1).containsKey(codon2)){

        QHMMDi.get(name).get(i).put(name2,
(QHMMDi.get(name).get(i).get(name2))+(Math.abs(log2(birdMap.get(name2).dinucFrequencies.get(i).get(codon1)
.get(codon2)))));}}}}
                                        tseq=tseq.substring(3);}
                                tseq=tempuBird.sequence;}
                        for(String name2: QHMMDi.keySet()){
                                if(name.equals(name2)==false){
                                        double score1=QHMMDi.get(name).get(1).get(name2);
                                        double score2=QHMMDi.get(name).get(2).get(name2);
                                        double bestScore=Math.min(score1, score2);
                                        if(QHMMDi_final.containsKey(name)==false){
                                                HashMap<String,Double> ntempu=new
HashMap<String,Double>();
                                                ntempu.put(name2, bestScore);
                                                QHMMDi_final.put(name, ntempu);}
                                        else{
                                                QHMMDi_final.get(name).put(name2, bestScore);}}}}}

        /**
         * Calculates the score for a difference in dinucleotide transition probabilities
         * between two taxa for a given gene.
         * @param bird1 taxa 1
         * @param bird2 taxa 2
```

```
	 * @return score, stored as a double
	 */
	public static double scoreBirdDinuc(Bird bird1, Bird bird2) {
		double score1 = 0.0;
		double score2 = 0.0;
		double distScore = 0.0;
		for (int i : bird1.dinucFrequencies.keySet()) {
			for (String c1 : allDinucs) {
				for (String c2 : allDinucs) {
					if (bird1.dinucFrequencies.containsKey(c1) == false) {
						score1 = 0.0;
					} else {
						if (bird1.dinucFrequencies.get(c1).containsKey(c2) ==
false) {
							score1 = 0.0;
						} else {
							score1 = bird1.dinucFrequencies.get(i).get(c1)
									.get(c2);}}
					if (bird2.dinucFrequencies.containsKey(c1) == false) {
						score2 = 0.0;
					} else {
						if (bird2.dinucFrequencies.get(i).get(c1)
								.containsKey(c2) == false) {
							score2 = 0.0;
						} else {
							score2 = bird2.dinucFrequencies.get(i).get(c1)
									.get(c2);}}
					distScore += Math.abs(score1 - score2);}}}
		return distScore;}
	/**
	 * A shortcut function that gets the log
	 * base two on the double passed in
	 * @param num number to get the log base 2 of
	 * @return the log base two of the number
	 */

	public static double log2(double num){
		return Math.log(num)/Math.log(2);}

	/**
	 * Returns the difference in GC scores between two taxa
	 * @param bird1 taxa 1
	 * @param bird2 taxa 2
	 * @return score, stored as a double
	 */
	public static double scoreGC(Bird bird1, Bird bird2){
		return Math.abs(bird1.GC_score-bird2.GC_score);}

	/**
	 * A shortcut function for System.out.println
	 * @param o, an object to print
	 */
	public static void print(Object o){
		System.out.println(o);}
	/**
	 * Make a file
	 * @param filenameANDpath, the absolute path of the desired new file
	 */
	public static void mkFile(File filenameANDpath){
		File newfile=filenameANDpath;
		try{
			newfile.createNewFile();}
		catch(IOException ioe){
			System.err.println("Error in making files\n "+ioe);}}

	/**
	 * Makes subsequence files for use in the bootstrapping method.
```

```
         * @param sameStart, a boolean indicating if all sequences should be sub-setted starting at the
same location
         * @param numBoots, the number of times to bootstrap
         * @param propBases, the proportion of bases wanted in the bootstrap
         * @param birdMap, the data structure containing all the sequences
         * @throws FileNotFoundException
         * @throws UnsupportedEncodingException
         */
        public static void genBootstrapSeqs(boolean sameStart, int numBoots, double propBases,
HashMap<String, Bird> birdMap) throws FileNotFoundException, UnsupportedEncodingException{
                new File("tmp").mkdir();
                new File("bootTrees").mkdir();
                int numBases=0;
                int startIndex=0;
                int sizeOfSeq=0;
                for(int i=0; i<numBoots; i++){
                        if(sameStart==false){
                                try {
                                        PrintWriter writer=new
PrintWriter("tmp/bootstrapIter"+i+".fasta","UTF-8");
                                        for(String berd: birdMap.keySet()){
                                                String id=birdMap.get(berd).id;
                                                String seq=birdMap.get(berd).sequence;
                                                writer.println(id);
                                                sizeOfSeq=seq.length();
                                                numBases=(int) Math.round(propBases*sizeOfSeq);
                                                startIndex=genRandInt(0,(sizeOfSeq-numBases)-2);
                                                writer.println(seq.substring(startIndex,
startIndex+numBases-1));
                                                print(seq.substring(startIndex, startIndex+numBases-1));}
                                        writer.close();}
                        catch (FileNotFoundException | UnsupportedEncodingException e) {
                                e.printStackTrace();}}
                        else{
                                String brd=(String) birdMap.keySet().toArray()[0];
                                String id=birdMap.get(brd).id;
                                String seq=birdMap.get(brd).sequence;
                                sizeOfSeq=seq.length();
                                numBases=(int) Math.round(propBases*sizeOfSeq);
                                startIndex=genRandInt(0,(sizeOfSeq-numBases)-2);
                                PrintWriter writer=new PrintWriter("tmp/bootstrapIter"+i+".fasta","UTF-
8");
                                for(String berd: birdMap.keySet()){
                                        id=birdMap.get(berd).id;
                                        seq=birdMap.get(berd).sequence;
                                        writer.println(id);
                                        if(startIndex+numBases>seq.length()){
                                                writer.println(seq.substring(startIndex, seq.length()-
1));}
                                        else{
                                                writer.println(seq.substring(startIndex,
startIndex+numBases-1));}}
                                writer.close();}}}
        /**
         * Shortcut function for generating a random integer.
         * @param Min the mimimum bound
         * @param Max the maximum bound
         * @return a randomly generated integer
         */
        public static int genRandInt(int Min, int Max){
                return (Min + (int)(Math.random() * ((Max - Min) + 1)));}

        /**
         * Just see if a value is parse-able to a double
         * Used in command line argument processing.
         * @param value the string being checked
         * @return a boolean representing saying if it is a double
         */
```

```
public static boolean isDouble(String value) {
    try {
        Double.parseDouble(value);
        return true;
    } catch (NumberFormatException e) {
        return false;}}

/**
 * Internally used command that populates
 * the codon to amino acid hashmap
 */
public static void popCodonMap(){
        codonAAMap.put("TTT","F");
        codonAAMap.put("TTC","F");
        codonAAMap.put("TTA","L");
        codonAAMap.put("TTG","L");
        codonAAMap.put("CTT","L");
        codonAAMap.put("CTC","L");
        codonAAMap.put("CTG","L");
        codonAAMap.put("CTA","L");
        codonAAMap.put("ATT","I");
        codonAAMap.put("ATC","I");
        codonAAMap.put("ATA","I");
        codonAAMap.put("ATG","M");
        codonAAMap.put("GTT","V");
        codonAAMap.put("GTC","V");
        codonAAMap.put("GTA","V");
        codonAAMap.put("GTG","V");
        codonAAMap.put("TCT","S");
        codonAAMap.put("TCC","S");
        codonAAMap.put("TCA","S");
        codonAAMap.put("TCG","S");
        codonAAMap.put("CCT","P");
        codonAAMap.put("CCC","P");
        codonAAMap.put("CCA","P");
        codonAAMap.put("CCG","P");
        codonAAMap.put("ACT","T");
        codonAAMap.put("ACC","T");
        codonAAMap.put("ACA","T");
        codonAAMap.put("ACG","T");
        codonAAMap.put("GCT","A");
        codonAAMap.put("GCC","A");
        codonAAMap.put("GCA","A");
        codonAAMap.put("GCG","A");
        codonAAMap.put("TAT","Y");
        codonAAMap.put("TAC","Y");
        codonAAMap.put("TAA","STOP");
        codonAAMap.put("TAG","STOP");
        codonAAMap.put("CAT","H");
        codonAAMap.put("CAC","H");
        codonAAMap.put("CAA","Q");
        codonAAMap.put("CAG","Q");
        codonAAMap.put("AAT","N");
        codonAAMap.put("AAC","N");
        codonAAMap.put("AAA","K");
        codonAAMap.put("AAG","K");
        codonAAMap.put("GAT","D");
        codonAAMap.put("GAC","D");
        codonAAMap.put("GAA","E");
        codonAAMap.put("GAG","E");
        codonAAMap.put("TGT","C");
        codonAAMap.put("TGC","C");
        codonAAMap.put("TGA","STOP");
        codonAAMap.put("TGG","W");
        codonAAMap.put("CGT","R");
        codonAAMap.put("CGC","R");
        codonAAMap.put("CGA","R");
        codonAAMap.put("CGG","R");
```

```
        codonAAMap.put("AGT","S");
        codonAAMap.put("AGC","S");
        codonAAMap.put("AGA","R");
        codonAAMap.put("AGG","R");
        codonAAMap.put("GGT","G");
        codonAAMap.put("GGC","G");
        codonAAMap.put("GGA","G");
        codonAAMap.put("GGG","G");}
/**
 * Internally used function that builds the Amino acid
 * to Codon map used in algorithm. It relies on popCodonMap()
 * to have been called prior to the calling of this method.
 */
public static void popAAtoCodonMap(){
        for(String i:codonAAMap.keySet()){
                String AA=codonAAMap.get(i);
                if(AAcodonMap.containsKey(AA)==false){
                        ArrayList<String>tempu=new ArrayList<String>();
                        tempu.add(i);
                        AAcodonMap.put(AA,tempu);}}}
/**
 * A shortcut method for getting translating codons into amino acids
 * @param codon the codon to be translated
 * @return the 1 letter amino acid code as String
 */
public static String translateCodon(String codon){
        return codonAAMap.get(codon);}
/**
 * Internally used command that populates a list of
 * every possible dinucleotide combination
 */
public static void popAllDinucs(){
        allDinucs.add("AA");
        allDinucs.add("AT");
        allDinucs.add("AG");
        allDinucs.add("AC");
        allDinucs.add("TT");
        allDinucs.add("TA");
        allDinucs.add("TC");
        allDinucs.add("TG");
        allDinucs.add("GG");
        allDinucs.add("GC");
        allDinucs.add("GA");
        allDinucs.add("GT");
        allDinucs.add("CC");
        allDinucs.add("CA");
        allDinucs.add("CT");
        allDinucs.add("CG");}
/**
 * Internally used command that populates a list of every possible
 * codon.
 */
public static void popAllCodons(){
        allCodons.add("TTT");
        allCodons.add("TTC");
        allCodons.add("TTG");
        allCodons.add("TTA");
        allCodons.add("TCT");
        allCodons.add("TCC");
        allCodons.add("TCA");
        allCodons.add("TCG");
        allCodons.add("TAT");
        allCodons.add("TAC");
        allCodons.add("TAA");
        allCodons.add("TAG");
        allCodons.add("TGT");
        allCodons.add("TGC");
        allCodons.add("TGA");
```

```
                allCodons.add("TGG");
                allCodons.add("CTT");
                allCodons.add("CTC");
                allCodons.add("CTA");
                allCodons.add("CTG");
                allCodons.add("CCT");
                allCodons.add("CCC");
                allCodons.add("CCA");
                allCodons.add("CCG");
                allCodons.add("CAT");
                allCodons.add("CAC");
                allCodons.add("CAA");
                allCodons.add("CAG");
                allCodons.add("CGT");
                allCodons.add("CGC");
                allCodons.add("CGA");
                allCodons.add("CGG");
                allCodons.add("ATT");
                allCodons.add("ATC");
                allCodons.add("ATA");
                allCodons.add("ATG");
                allCodons.add("ACT");
                allCodons.add("ACC");
                allCodons.add("ACA");
                allCodons.add("ACG");
                allCodons.add("AAT");
                allCodons.add("AAC");
                allCodons.add("AAA");
                allCodons.add("AAG");
                allCodons.add("AGT");
                allCodons.add("AGC");
                allCodons.add("AGA");
                allCodons.add("AGG");
                allCodons.add("GTT");
                allCodons.add("GTC");
                allCodons.add("GTA");
                allCodons.add("GTG");
                allCodons.add("GCT");
                allCodons.add("GCC");
                allCodons.add("GCA");
                allCodons.add("GCG");
                allCodons.add("GAT");
                allCodons.add("GAC");
                allCodons.add("GAA");
                allCodons.add("GAG");
                allCodons.add("GGT");
                allCodons.add("GGC");
                allCodons.add("GGA");
                allCodons.add("GGG");}
        /**
         * prints the help statement seen with --help or -h
         */
        public static void printHelp(){
                print("---ScrawkovPHY.java---");
                print(usage);
                print("Example: java ScrawkovPHY gene exampleFasta.fasta --normalize T");
                print("Example: java ScrawkovPHY species exampleFastaForGene1.fasta
exampleFastaForGene2.fasta --maxByPair T");
                print(" ");
                print("-------Flags available, with desciption of fucntion---------");
                print(" ");
                print("--folder");
                print("Use a folder rather than a series of files in species tree contsruction");
                print(" ");
                print("--doNJ");
                print("   Construct a neighbor joining tree in addition to the default UPGMA tree");
                print("   Default behaviour is true");
                print(" ");
```

```
                print("--njOut");
                print("   The file name for the NJ tree output. Only used if doNJ is true");
                print("   The default behaviour is to use the name njOut.nwk");
                print(" ");
                print("--maxByPair:");
                print("   maxByPair takes either true or false as a value");
                print("   If true, the algorithm will find and use the optimal reading frame for each
pairwise sequence");
                print("   The default behaviour is true ");
                print(" ");
                print("--normalize:");
                print("   normalize takes either true or false as a value");
                print("   This parameter only affects the construction of species trees");
                print("   If true, the results of each gene tree are scaled to the range of the most
disparate");
                print("   gene tree. That is, the one with the biggest difference between highest and
lowest score");
                print("   The default behaviour is true ");
                print(" ");
                print("--outputFile");
                print("   outputFile takes the name of the file you want the tree to be output in");
                print("   The default behaviour is to only print the output to the terminal, not to a
file");
                print("   It has the shortcut -o");
                print(" ");
                print("--kSize");
                print("   override the kmer size for dinucleotide aspect of QHMM.");
                print("   default value is -1, which is ignored by program.");
                print(" ");
                print("--bootSeq");
                print("   Experimental: Should subsequences of DNA be obtained for bootstrapping? Must be
run with gene");
                print("   Default behaviour is false");
                print(" ");
                print("--diagMat");
                print("   Should the PHYLIP formatted distance matrix be written along the diagonal?");
                print("   If false, the full matrix will be written.");
                print("   The default behaviour is true.");
                print(" ");
                print("--bootOut");
                print("   Experimental: Name of outPut bootstrapped tree folder. If a value is entered,
will use the species tree infrastructure to make");
                print("   bootstrap trees from the output of bootSeq. Must be run with species");
                print("   The default behaviour is to not make these trees.");
                print(" ");
                print("--help");
                print("   Print this help statement");
                print(" ");
                print("--weightGC, --weightBias, --weightTriFreq, --weightDiFreq, --weightTriQHMM, --
weightDiQHMM");
                print("   These flags accept numeric input (decimals are fine)");
                print("   These arguments are the weights applied to the 6 features used to create the
MCCI, which is a surrogate distance.");
                print("   They affect the GC content, codon bias, trinucleotide frequency, dinucleotide
frequency,");
                print("    trinucleotide transistion QHMM, and dinucleotide transtition QHMM
respectively.");
                print("   The default values are 10,0.1,1,1,1,0.00005,and 0.00005 respectively.");}}
```

```
////Bird.java

import java.util.HashMap;
/**
 * Bird is a generic representation of a taxon and its sequence.
 * @author J. Nick Fisk
 *
 */
public class Bird {
	public double GC_score; //GC content proportion
	public double CodonFreqScore;
	public String sequence; //store the sequence in RAM
	public String id; //name of sequence
	public HashMap<Integer,HashMap<String, HashMap<String, Double>>> codonFrequencies; //initMaps
	public HashMap<Integer, HashMap<String, HashMap<String, Double>>> dinucFrequencies;
	public HashMap<String,HashMap<String,Double>> biasMap;

	/**
	 * An object representing the a taxon in the algorithm.
	 * It is names bird as it was developed on birds originally.
	 * @param id the id of the taxon from the fasta file
	 * @param sequence the nucleotide sequence
	 */
	public Bird(String id, String sequence){
		this.GC_score=0.0;
		this.CodonFreqScore=0.0;
		this.sequence=sequence;
		this.id=id;
		this.dinucFrequencies=new HashMap<Integer,HashMap<String,HashMap<String, Double>>> ();
		this.codonFrequencies=new HashMap<Integer,HashMap<String,HashMap<String, Double>>> ();
		this.codonFrequencies=newCalcCodonFreq();
		//check k from main to see if we are doing generic length kmer rather than a hard 2.
		if(ScrawkovPHY.k<1){
			this.dinucFrequencies=newCalcDiFreq();}
		else{
			this.dinucFrequencies=calcKFreq(ScrawkovPHY.k);}
		this.GC_score=calcGC();
		this.biasMap=mkBiasMap();}
	/**
	 * Measure the features ultimately used in the calculation of codon bias
	 * @return HashMap of measured features
	 */
	public HashMap<String, HashMap<String,Double>> mkBiasMap(){
		String inter=this.sequence; //shallow copy of the sequence
		HashMap<String,HashMap<String,Double>> biasMap=new
HashMap<String,HashMap<String,Double>>();
		String pIter=""; //holder variable
		//go through and chop down the sequence, translating each codon at the end.
		while(inter.length()>3){
				pIter=ScrawkovPHY.translateCodon(inter.substring(0,3));
				if(biasMap.containsKey(pIter)==false){
					HashMap<String,Double> bcodonMap=new HashMap<String,Double>();
					bcodonMap.put(inter.substring(0,3), (double) 1);
					biasMap.put(pIter, bcodonMap);}
				else{
					if(biasMap.get(pIter).containsKey(inter)==false){
						biasMap.get(pIter).put(inter.substring(0,3), 1.0);}
					else{

	biasMap.get(pIter).put(inter.substring(0,3),biasMap.get(pIter).get(inter)+1);}}
				inter=inter.substring(3);}
		for(String aa: biasMap.keySet()){
			double count=0;
```

```
                        for(String entry: biasMap.get(aa).keySet()){
                                count+=biasMap.get(aa).get(entry);}
                        for(String entry:biasMap.get(aa).keySet()){
                                biasMap.get(aa).put(entry,biasMap.get(aa).get(entry)/count);}}
                return biasMap;}
        /**
         * Measures trinucleotide occurrence frequency, as well as the transition frequency
         * in sequences.
         * @param seq sequence being observed
         * @return HashMap of measured features
         */
        public HashMap<String, HashMap<String, Double>> calcCodonHelper(String seq){
                String tseq=seq;
                HashMap<String, HashMap<String, Double>> tmap=new HashMap<String, HashMap<String,
Double>>();
                int numCodons=0;
                //crawl down and chop the sequence into smaller pieces of size 3.
                while(tseq.length()>5){
                        numCodons+=1;
                        String codon1=tseq.substring(0, 3);
                        String codon2=tseq.substring(3, 6);
                        if(!tmap.containsKey(codon1)){
                                HashMap<String,Double>tempu=new HashMap<String, Double>();
                                Double count=1.0;
                                tempu.put(codon2,  count);
                                tmap.put(codon1, tempu);}
                        else{
                                if(!tmap.get(codon1).containsKey(codon2)){
                                        tmap.get(codon1).put(codon2, (double)1);}
                                else{
                                        tmap.get(codon1).put(codon2, tmap.get(codon1).get(codon2)+1);}}
                        tseq=tseq.substring(3);}
                for(String map: tmap.keySet()){
                        for(String entry: tmap.get(map).keySet()){
                                tmap.get(map).put(entry, tmap.get(map).get(entry)/numCodons);}}
                return tmap;}
        /**
         * Calls the helper function a differning number of times depending on the status
         * of the maxByPair flag.
         * @return HashMap of measured features
         */
        public HashMap<Integer,HashMap<String, HashMap<String, Double>>> newCalcCodonFreq(){
                String tseq=this.sequence;
                HashMap<Integer,HashMap<String, HashMap<String, Double>>> tmap=new
HashMap<Integer,HashMap<String, HashMap<String, Double>>>();
                int numToDo;
                //if maxByPair is true, each starting frame will be represented.
                if(ScrawkovPHY.maxByPair==true){
                        numToDo=3;}
                else{
                        numToDo=1;}
                int count=0;
                while(count<=numToDo){
                        tmap.put(count, calcCodonHelper(tseq));
                        tseq=tseq.substring(1,tseq.length());
                        count+=1;}
                return tmap;}
        /**
         * Measures dinucleotide frequencies and transition frequencies in a sequence
         * @param seq the sequence being observed
         * @return HashMap of measured features
         */
        public HashMap<String, HashMap<String, Double>> calcDiHelper(String seq){
                String tseq=seq;
                HashMap<String, HashMap<String, Double>> tmap=new HashMap<String, HashMap<String,
Double>>();
                int numCodons=0;
                while(tseq.length()>5){
```

```
                        numCodons+=1;
                        String codon1=tseq.substring(0, 2);
                        String codon2=tseq.substring(2, 4);
                        if(!tmap.containsKey(codon1)){
                                HashMap<String,Double>tempu=new HashMap<String, Double>();
                                Double count=1.0;
                                tempu.put(codon2,  count);
                                tmap.put(codon1, tempu);}
                        else{
                                if(!tmap.get(codon1).containsKey(codon2)){
                                        tmap.get(codon1).put(codon2, (double)1);}
                                else{
                                        tmap.get(codon1).put(codon2, tmap.get(codon1).get(codon2)+1);}}
                        tseq=tseq.substring(2);}
                for(String map: tmap.keySet()){
                        for(String entry: tmap.get(map).keySet()){
                                tmap.get(map).put(entry, tmap.get(map).get(entry)/numCodons);}}
                return tmap;}

        /**
         * Depending on the value of maxByPair flag, calls the helper function a variable number of times.
         * @return A HashMap of observed features
         */
        public HashMap<Integer,HashMap<String, HashMap<String, Double>>> newCalcDiFreq(){
                String tseq=this.sequence;
                HashMap<Integer,HashMap<String, HashMap<String, Double>>> tmap=new
HashMap<Integer,HashMap<String, HashMap<String, Double>>>();
                int numToDo;
                if(ScrawkovPHY.maxByPair==true){
                        numToDo=2;}
                else{
                        numToDo=1;}
                int count=0;
                while(count<=numToDo){
                        tmap.put(count, calcDiHelper(tseq));
                        tseq=tseq.substring(1,tseq.length());
                        count+=1;}
                return tmap;}
        /**
         * Performs the same function as calcCodonHelper and calcDinucHelper, but with a generic sized
kmer.
         * @param seq The sequence
         * @param k the size of the kmer to be used
         * @return
         */
        public HashMap<String, HashMap<String, Double>> calcKHelper(String seq, int k){
                String tseq=seq;
                HashMap<String, HashMap<String, Double>> tmap=new HashMap<String, HashMap<String,
Double>>();
                int numCodons=0;
                int whileLen= (k*2)+1;
                while(tseq.length()>whileLen){
                        numCodons+=1;
                        String codon1=tseq.substring(0, k);
                        String codon2=tseq.substring(k, k+k);
                        if(!tmap.containsKey(codon1)){
                                HashMap<String,Double>tempu=new HashMap<String, Double>();
                                Double count=1.0;
                                tempu.put(codon2,  count);
                                tmap.put(codon1, tempu);}
                        else{
                                if(!tmap.get(codon1).containsKey(codon2)){
                                        tmap.get(codon1).put(codon2, (double)1);}
                                else{
                                        tmap.get(codon1).put(codon2, tmap.get(codon1).get(codon2)+1);}}
                        tseq=tseq.substring(k);}
                for(String map: tmap.keySet()){
                        for(String entry: tmap.get(map).keySet()){
```

```
                                tmap.get(map).put(entry, tmap.get(map).get(entry)/numCodons);}}
                return tmap;}

        /**
         * Depending on the value of maxByPair flag, calls the helper function a variable number of times.
         * @return A HashMap of observed features
         */
        public HashMap<Integer,HashMap<String, HashMap<String, Double>>> calcKFreq(int k){
                String tseq=this.sequence;
                HashMap<Integer,HashMap<String, HashMap<String, Double>>> tmap=new
HashMap<Integer,HashMap<String, HashMap<String, Double>>>();
                int numToDo;
                if(ScrawkovPHY.maxByPair==true){
                        numToDo=k;}
                else{
                        numToDo=1;}
                int count=0;
                while(count<=numToDo){
                        tmap.put(count, calcKHelper(tseq,ScrawkovPHY.k));
                        tseq=tseq.substring(1,tseq.length());
                        count+=1;}
                return tmap;}
        /**
         * Measures the GC content of a sequence.
         * @return the difference in gc content as a double
         */
        public double calcGC(){
                String tseq=this.sequence;
                double count = tseq.length() - tseq.replace("G", "").length();
                count+=tseq.length()-tseq.replace("C", "").length();
                return count/tseq.length();}}
```

```
//Node.java
import java.util.ArrayList;

/**
 *
 * @author Nick Fisk
 * A node containing information about the
 * organism used in searching a graph to recover a tree.
 * currently used only for a UPGMA approach, but
 * is extendible to other approaches.
 */
public class Node {
        String newick;
        ArrayList<String> names=new ArrayList<String>();
        public Node(String name){
                this.names.add(name);
                name=name.replaceAll("[()]", "");
                name=name.replaceAll(">", "");
                //name=name.replaceAll("\\","");
                //name=name.replaceAll("/","");
                //if(name.length()>30){
                        //name=name.substring(0, 29);
                //}
                name=name.replaceAll(" ", "_");
```

```
        name=name.replaceAll("-", "_");
        name.replaceAll(",", "_");
        this.newick=name;}
/**
 * Constructs a node from two other nodes and a distance
 * @param node1, a node to be joined to node2
 * @param node2, a node to be joined to node1
 * @param dist the distance betwwen the two input nodes
 *
 */
@SuppressWarnings("unchecked")
public Node(Node node1, Node node2, double dist){
        this.newick="("+node1.newick+":"+dist/2+","+node2.newick+":"+dist/2+")";
        this.names=new ArrayList<String>(node1.names);
        this.names.addAll((ArrayList<String>)node2.names.clone());}
/**
 * Find the distance between any two nodes. For nodes that
 * consist of many nodes, the overall average of all the nodes is used,
 * not just the average of the two nodes being compared
 * @param n1, the first node or cluster of nodes
 * @param n2, the second node or cluster of nodes
 * @return the distance between the two nodes
 */
public static double calcDistance(Node n1, Node n2){
        ArrayList<String>names1=new ArrayList<String>(n1.names);
        ArrayList<String>names2=new ArrayList<String>(n2.names);
        double score=0.0;
        int count=0;
        for(String i: names1){
                for(String j: names2){
                        score+=ScrawkovPHY.initScores.get(i).get(j);
                        count+=1;}}
        return score/count;}
/*
 * (non-Javadoc)
 * @see java.lang.Object#toString()
 */
public String toString(){
        return(String.valueOf(this.names));}
@Override
public boolean equals(Object o){
        if(o==this){
                return true;}
        if(!(o instanceof Node)){
                return false;}
        Node o2=(Node)o;
        for(String n: this.names){
                if(o2.names.contains(n)==false){
                        return false;}}
        for(String n: o2.names){
                if(this.names.contains(n)==false){
                        return false;}}
        return true;}
/**
 * needed for the equals override
 */
public int hashCode(){
        return this.names.hashCode();}}
```

```
//NJNode.java
import java.util.ArrayList;

/**
 * A Node class to represent the NJ graph search.
 * A different node was needed than UPGMA as the
 * boundary conditions and edge cases need be handled a little
 * differently.
 * @author J. Nick Fisk
 *
 */
public class NJNode {
	ArrayList<String> names=new ArrayList<String>();
	double dist1=Double.MIN_VALUE;
	double dist2=Double.MIN_VALUE;
	ArrayList<NJNode> njnodes=new ArrayList<NJNode>();
	String newick="";
	/**
	 * Constructor for the first nodes
	 * @param name
	 */
	public NJNode (String name){
		this.names.add(name);
		name=name.replaceAll("[()]", "");
		name=name.replaceAll(">", "");
		//name=name.replaceAll("\\","");
		//name=name.replaceAll("/","");
	//	if(name.length()>30){
		//	name=name.substring(0, 29);
		//}
		name=name.replaceAll(" ", "_");
		name=name.replaceAll("-", "_");
		name.replaceAll(",", "_");
		this.newick=name;}
	/**
	 * Constructor for grouping together two nodes
	 * with two different lengths.
	 * @param n1
	 * @param n2
	 * @param dist1
	 * @param dist2
	 */
	public NJNode(NJNode n1, NJNode n2, double dist1, double dist2){
		this.njnodes.add(n1);
		this.njnodes.add(n2);
		this.names.addAll(n1.names);
		this.names.addAll(n2.names);
		this.newick="("+n1.newick+":"+dist1+","+n2.newick+":"+dist2+")";}
	/**
	 * String representation of node
	 */
	public String toString(){
		return(String.valueOf(this.names));}
	@Override
	public boolean equals(Object o){
		if(o==this){
			return true;}
		if(!(o instanceof Node)){
			return false;}
		Node o2=(Node)o;
		for(String n: this.names){
			if(o2.names.contains(n)==false){
				return false;}}
```

```
                for(String n: o2.names){
                        if(this.names.contains(n)==false){
                                return false;}}
                return true;}
        /**
        * needed for the equals override
        */
        public int hashCode(){
                return this.names.hashCode();}}
```

Code used in EMU-Phy

```
//ScrawQ.java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import java.io.BufferedWriter;
import java.util.ArrayList;
import java.util.Scanner;

/*
 * ScrawQ, which is the working name of ScrawDB
 * is under development. It is a phyloinformatics DB
 * creation and management system.
 *
 */
public class ScrawQ {
        public static ArrayList<Character> bases=new ArrayList<Character>();
        public static void main(String[] args) {
                //a simple bool to decide whether to display the "ready for next command" prompt
                populateBases();
                boolean amTesting=false;
                if(amTesting){
                        validateDB();
                        return;}
                boolean firstRun=true;
                //print welcome message and user prompt
                System.out.println("Welcome to EMU-Phy!");
                System.out.println("Please enter a command. Type 'help' for help and 'quit' to quit");
                //connect to standard in
                Scanner scanIn=new Scanner(System.in);
                //will continue to read in input until user quits
                //or something terrible happens (execption thrown)
                while(true){
                        if(firstRun==true){
                                firstRun=false;}
                        else{
```

```
                                System.out.println("Ready for next command      (enter 'quit' to
exit)");}
                        //get the next input, save as string
                        String thisCommand=scanIn.next();
                        //A very large switch command. More efficient and readable
                        //than if-else ifs-elses
                        switch(thisCommand){
                                //if we are done and want to exit peacefully
                                case "quit":
                                        System.out.println("Thank you for using ScrawQ!");
                                        System.exit(0);
                                        break;
                                //help message to show all commands
                                case "help":
                                        printHelp();
                                        break;
                                //installs the system
                                case "install":
                                        installFileSys();
                                        break;
                                //To do: give functionality
                                //will update everything
                                case "update":
                                        updateAll();
                                        break;
                                //forcibly redo all the analysis
                                case "redo":
                                        redoAll();
                                        break;
                                case "validate":
                                        validateDB();
                                        break;
                                //add subprompt
                                case "add":
                                        boolean keepChecking=true;
                                        while(keepChecking==true){
                                                String addCommand;
                                                System.out.println("Add what?");
                                                System.out.println("Options are: group, taxa, gene,
primer, taxonomy. Enter 'cancel' to cancel or 'done' to finish");
                                                addCommand=scanIn.next();
                                                switch(addCommand){
                                                        case "group":
                                                                mkGroup();
                                                                break;
                                                        case "taxa":
                                                                addNewTaxa();
                                                                break;
                                                        case "gene":
                                                                //gene add
                                                                addGene();
                                                                break;
                                                        case "primer":
                                                                //add primer
                                                                break;
                                                        case "taxonomy":
                                                                //add taxonomy
                                                                break;
                                                        case "cancel":
                                                                keepChecking=false;
                                                                break;
                                                        case "done":
                                                                keepChecking=false;
                                                                break;
                                                        default:
                                                                System.out.println("Invalid selection.
Please try again...");
                                                                break;}}
```

```
                                        break;
                                case "show":
                                        keepChecking=true;
                                        while(keepChecking==true){
                                                String showCommand;
                                                System.out.println("Show what?");
                                                System.out.println("Options are: groups, taxa, genes,
primers, taxonomy. Enter 'cancel' to cancel or 'done' to finish");
                                                showCommand=scanIn.next();
                                                switch(showCommand){
                                                case "groups":
                                                        try {
                                                                System.out.println("Groups are...");
                                                                System.out.println();
                                showFile("ScrawQ_Phyloinformatics/Groups/List_of_All_Groups.txt");
                                                                System.out.println();
                                                        } catch (IOException e) {
                                                                // TODO Auto-generated catch block
                                                                e.printStackTrace();}
                                                        break;
                                                case "taxa":
                                                        try {
                                                                System.out.println("Taxon are...");
                                                                System.out.println();
                                showFile("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
                                                                System.out.println();
                                                        } catch (IOException e) {
                                                                // TODO Auto-generated catch block
                                                                e.printStackTrace();}
                                                        break;
                                                case "genes":
                                                        keepChecking=true;
                                                        while(keepChecking==true){
                                                                Scanner temp=new Scanner(System.in);
                                                                String taxa=new String();
                                                                System.out.println("Name of taxa to
display genes for? Type 'display' to show all available taxa (type 'cancel' to cancel");
                                                                taxa=temp.next();
                                                                if(taxa.equals("cancel")){
                                                                        keepChecking=false;}
                                                                else if(taxa.equals("display")){
                                                                        System.out.println("Displaying
available taxa\n");
                                                                        try {

        showFile("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
                                                                                System.out.println();
                                                                        } catch (IOException e) {
                                                                                e.printStackTrace();}}
                                                                else{
                                                                        System.out.println("Displaying
Genes for Taxa: "+taxa+"...");
                                                                        try {
                                showFile("ScrawQ_Phyloinformatics/All_Taxa/"+taxa+"/Genes/GeneList.txt");
                                                                        } catch (IOException e) {
                                                                                // TODO Auto-generated
catch block
                                                                                System.out.println("That
taxa does not exist. Try adding the taxa and trying again.");}}}
                                                        break;
                                                case "primer":
                                                        break;
                                                case "taxonomy":
                                                        break;
                                                case "cancel":
                                                        keepChecking=false;
                                                        break;
                                                case "done":
```

```
                                                keepChecking=false;
                                                break;
                                        default:
                                                System.out.println("Invalid selection. Please try
again...");
                                        break;}}
                                        break;
                                case "mkGroup":
                                        mkGroup();
                                        break;
                                //help message for group only commands
                                case "groupHelp":
                                        printGroupHelpMessage();
                                        break;
                                case "newTaxa":
                                        addNewTaxa();
                                        break;
                                default:
                                        System.out.println("Invalid command: help message will be
displayed");
                                        printHelp();
                                        break;}}}

        /*
         * Initializes filesystem that will be used as DB
         * uses user input to decide to install examples or not
         */
        public static void installFileSys(){
                Scanner temp=new Scanner(System.in);
                String examplesDesired=new String();
                System.out.println("Installing EMU-Phy in working directory...!");
                System.out.println(System.getProperty("user.dir"));
                boolean keepchecking=true;
                boolean doExamples=false;
                while(keepchecking==true){
                        System.out.println("Examples desired? (Will insert example data in datasytem)");
                        System.out.println("Valid options are 'y' or 'n' or 'cancel')");
                        examplesDesired=temp.next();
                        switch(examplesDesired){
                                case "y":
                                        doExamples=true;
                                        keepchecking=false;
                                        print("Installing EMU-Phy with examples...");
                                        break;
                                case "n":
                                        doExamples=false;
                                        keepchecking=false;
                                        break;
                                case "cancel":
                                        System.out.println("Cancelling installation...");
                                        return;}}
                //make all the dirs and files basally necessary for system.
                new File("ScrawQ_Phyloinformatics").mkdir();
                new File("ScrawQ_Phyloinformatics/Groups").mkdir();
                new File("ScrawQ_Phyloinformatics/All_Taxa").mkdir();
                new File("ScrawQ_Phyloinformatics/ScrawkovPhy").mkdir();
                new File("ScrawQ_Phyloinformatics/pipelineModules").mkdir();
                File ListOfAllTaxa=new File("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
                File ListOfAllGroups=new File("ScrawQ_Phyloinformatics/Groups/List_of_All_Groups.txt");
                File ListOfAllModules=new
File("ScrawQ_Phyloinformatics/pipelineModules/List_of_All_Modules.txt");
                mkFile(ListOfAllTaxa);
                mkFile(ListOfAllGroups);
                mkFile(ListOfAllModules);
                //File dir=new File(".");
                //File files[]=dir.listFiles();
                //for(File f: files){
                        //System.out.println(f);
```

```
		//}
		//System.out.println(System.getProperty("user.home"));}
	/**
	 * addGene adds gene info to the database. Requires user to specify a taxa to add the gene for
	 */
	public static void addGene(){
		Scanner temp=new Scanner(System.in);
		String thisTaxa=new String();
		System.out.println("Name of taxa to add gene for? Type 'display' to see available taxa
(type 'cancel' to cancel");
		thisTaxa=temp.next();
		if(thisTaxa.equals("cancel")){
			return;}
		else if(thisTaxa.equals("display")){
			System.out.println("Displaying available taxa\n");
			try {
				showFile("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
				addGene();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();}}
		else{
			File dir= new File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa);
			if(!dir.exists()){
				System.out.println("That taxa does not exist. Try adding it and trying
again");
				addGene();}
			else{
				String geneName;
				System.out.println("What is the gene name?");
				geneName=temp.next();
				geneName=sanitizeInput(geneName);
				print("Using gene name: "+geneName);
				String method;
				System.out.println("DNA/RNA sequence required...");
				System.out.println("To supply a path to a file containing only this gene
in FASTA format, enter 'path'...");
				System.out.println("To supply the sequence directly, enter 'direct'...");
				System.out.println("To cancel addition of gene, enter 'cancel'...");
				method=temp.next();
				String sequence="";
				if(method.equals("cancel")){
					print("Canceling addition of gene...");
					return;}
				else if(method.equals("path")){
					class pathDoer{
						public String doPath(){
							String seq="";
							String pathToSeq;
							System.out.println("What is the path to the
sequence file (in FASTA Format)?");
							pathToSeq=temp.next();
							//DO NOT SANITIZE!!!!! Is supposed to have /s or
\s
							File seqFile=new File(pathToSeq);
							if(!seqFile.exists()){
								System.out.println("Invalid path to
file...");
								doPath();}
							else{
								//read in seq function, but for now
print something
								System.out.println("Got to a valid
file!");
								seq=readSeqFromFasta(seqFile);
								seq=validateSeq(seq);}
							return(seq);}}
					sequence=new pathDoer().doPath();}
```

```
                                else if(method.equals("direct")){
                                        class directDoer{
                                                public String doDirect(){
                                                        System.out.println("What is the sequence?");
                                                        String seq=temp.next();
                                                        seq=validateSeq(seq);
                                                        if(seq.equals("")){
                                                                print("Invalid sequence! Please use
IUPAC compliant sequences only!");
                                                                doDirect();}
                                                        return seq;}}
                                        sequence=new directDoer().doDirect();}
                                else{
                                        System.out.println("Invalid method of supplying sequence.
Addition of gene" + geneName +"aborting...\n");
                                        return;}
                                new
File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa+"/Genes/"+geneName+"/").mkdir();
                                BufferedWriter output;
                                try {
                                        output= new BufferedWriter(new
FileWriter("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa+"/Genes/"+geneName+"/sequence.txt", true));
                                        output.append(sequence);
                                        output.newLine();
                                        output.close();
                                } catch (IOException e) {
                                        e.printStackTrace();}}}
                print("Gene addition sucessful!");}

        /*
         * makes a new group based on user input
         * sanitizes the input, just in case
         */
        public static void mkGroup(){
                //connect to standard in
                Scanner temp=new Scanner(System.in);
                String thisGroup=new String();
                //get input
                System.out.println("Name of new group to add? (type 'cancel' to cancel");
                thisGroup=temp.next();
                System.out.println("adding group "+thisGroup+"...");
                //if this is not what they wanted to do, cancel
                if(thisGroup.equals("cancel")){
                        return;}
                //sanitize the input, make a new dir for the group
                //make a file to keep track of the members of the group.
                else{
                        thisGroup=sanitizeInput(thisGroup);
                        new File("ScrawQ_Phyloinformatics/Groups/"+thisGroup).mkdir();
                        File groupFile=new
File("ScrawQ_Phyloinformatics/Groups/"+thisGroup+"/members.txt");
                        mkFile(groupFile);
                        BufferedWriter output;
                        try {
                                output= new BufferedWriter(new
FileWriter("ScrawQ_Phyloinformatics/Groups/List_of_All_Groups.txt", true));
                                output.append(thisGroup);
                                output.newLine();
                                output.close();
                        } catch (IOException e) {
                                e.printStackTrace();}
                        System.out.println(thisGroup+" was added sucessfully!");}}
        /*
         * Adds a new taxa via user input
         * sanitizes the user input, just in case
         */
        public static void addNewTaxa(){
                Scanner temp=new Scanner(System.in);
```

```
                String thisTaxa=new String();
                System.out.println("Name of new taxa to add? (type 'cancel' to cancel");
                thisTaxa=temp.next();
                if(thisTaxa.equals("cancel")){
                        return;}
                else{
                        System.out.println("adding Taxa: "+thisTaxa+"...");
                        thisTaxa=sanitizeInput(thisTaxa);
                        new File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa).mkdir();
                        new File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa+"/Genes/").mkdir();
                        File taxaFile=new
File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa+"/Genes/GeneList.txt");
                        mkFile(taxaFile);
                        File aliases=new
File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa+"/nicknames.txt");
                        mkFile(aliases);
                        BufferedWriter output;
                        try {
                                output= new BufferedWriter(new
FileWriter("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt", true));
                                output.append(thisTaxa);
                                output.newLine();
                                output.close();
                        } catch (IOException e) {
                                e.printStackTrace();}
                        System.out.println(thisTaxa+" was added sucessfully!");}}
        /*
         * Adds new taxa based on a string arg. Meant for internal use.
         */
        public static void addNewTaxaInternal(String thisTaxa){
                thisTaxa=sanitizeInput(thisTaxa);
                new File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa).mkdir();
                File taxaFile=new File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa+"/GeneList.txt");
                mkFile(taxaFile);
                File aliases=new File("ScrawQ_Phyloinformatics/All_Taxa/"+thisTaxa+"/nicknames.txt");
                mkFile(aliases);}

        /*
         * Same as mkgroup, but meant for internal use based off a string arg
         */
        public static void mkGroupInternal(String thisGroup){
                thisGroup=sanitizeInput(thisGroup);
                new File("ScrawQ_Phyloinformatics/Groups/"+thisGroup).mkdir();
                File groupFile=new File("ScrawQ_Phyloinformatics/Groups/"+thisGroup+"/members.txt");
                mkFile(groupFile);
                BufferedWriter output;
                try {
                        output= new BufferedWriter(new
FileWriter("ScrawQ_Phyloinformatics/Groups/List_of_All_Groups.txt", true));
                        output.append(thisGroup);
                        output.newLine();
                        output.close();
                } catch (IOException e) {
                        e.printStackTrace();}
                System.out.println(thisGroup+" was added sucessfully!");}

        public static void validateDB(){
                validateTaxa();
                validateGroups();}
        public static void validateGroups(){
        }
        public static void validateTaxa(){
                ArrayList<String> recordedTaxa=getRecordedTaxa();
                print(recordedTaxa);
                ArrayList<String> observedTaxa=getObservedTaxa();
                print(observedTaxa);
                ArrayList<String> notInObs=new ArrayList<String>(recordedTaxa);
                notInObs.removeAll(observedTaxa);
```

```
		if(notInObs.size()>0){
			print("There is a difference between master list and observed taxa entries");
			print("Resolving difference by updating master list");
			File temp=new File("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
			///We are set to delete this without backing file up, since we have it in RAM
already to correct if something bad happens.
			boolean completed=temp.delete();
			ArrayList<String> forAllTaxa=new ArrayList<String>(observedTaxa);
			if(completed==false){
				print("There was an error deleting abherent entries");
				print("Restoring master list to last working state...");}
			else{
				File ListOfAllTaxa=new
File("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
				mkFile(ListOfAllTaxa);
				BufferedWriter output = null;
				try {
					output= new BufferedWriter(new
FileWriter("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt", true));
				} catch (IOException e) {
					e.printStackTrace();}
				for(String i : forAllTaxa){
					try{
						output.append(i);
						output.newLine();}
					catch(IOException o){
						o.printStackTrace();}}
				try {
					output.close();
				} catch (IOException e) {
					e.printStackTrace();}}}
		ArrayList<String> notInRec=new ArrayList<String>(observedTaxa);
		notInRec.removeAll(recordedTaxa);
		if(notInRec.size()>0){
			File ListOfAllTaxa=new
File("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
			BufferedWriter output = null;
			try{
				output= new BufferedWriter(new
FileWriter("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt", true));
				for(String i : notInRec){
					boolean havePrinted=false;
					File temp=new
File("ScrawQ_Phyloinformatics/All_Taxa/"+i+"/Genes");
					if(temp.exists()==false){
						print("Incomplete directory information found for taxa "+
i+"!");
						print("Updating directory to conform to minimal
requirements.");
						havePrinted=true;
						temp.mkdir();}
					temp=new
File("ScrawQ_Phyloinformatics/All_Taxa/"+i+"/Genes/GeneList.txt");
					if(temp.exists()==false){
						if(havePrinted==false){
							print("Incomplete directory information found for
taxa "+ i+"!");
							print("Updating directory to conform to minimal
requirements.");}
						mkFile(temp);}
					output.append(i);
					output.newLine();}
				output.close();}
			catch(IOException o){
				o.printStackTrace();}}}

	public static ArrayList<String> getRecordedTaxa(){
		ArrayList<String> taxaList=new ArrayList<String>();
```

```
                File temp=new File("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
                if(!temp.exists()){
                        ///generate a blank one on the fly!
                        mkFile(temp);}
                taxaList=returnFile("ScrawQ_Phyloinformatics/All_Taxa/List_of_All_Taxa.txt");
                return taxaList;}
        public static ArrayList<String>getObservedTaxa(){
                ArrayList<String>obsTaxaList=new ArrayList<String>();
                File temp=new File("ScrawQ_Phyloinformatics/All_Taxa/");
                File[] tempList=temp.listFiles();
                for(File i: tempList){
                        String tempName=i.getName();
                        if(tempName.equals("List_of_All_Taxa.txt")==false){
                                obsTaxaList.add(tempName);}}
                return(obsTaxaList);}
        public static void printGroupHelpMessage(){
                System.out.println("Here are the commands pertaining to working with groups");
                System.out.println("groupHelp: Displays this help message");
                System.out.println("mkGroup: Makes a new group");}
        /*
         * Prints the purpose of all commands
         */
        public static void printHelp(){
                System.out.println("Below are commands and their function");
                System.out.println("help : Displays this help message");
                System.out.println("quit : Exits ScrawQ. There is no prompt for confirmation.");
                System.out.println("install: Installs ScrawQ. Only Run this once unless you want a fresh
install!");
                System.out.println("update: Updates the internal structures and trees if new data is
present.");
                System.out.println("redo: Performs all analyses regardless of if the data has been updated
or not.");
                System.out.println("mkGroup: Makes a new group");}

        /*
         * Replaces all '/','\',' ', and '.' characters with underscores
         * and alerts the user to this change
         */
        public static String sanitizeInput(String token){
                String newToken;
                newToken=token.replace('\\', '_');
                newToken=newToken.replace('/', '_');
                newToken=newToken.replace(' ', '_');
                newToken=newToken.replace('.', '_');
                if(!token.equals(newToken)){
                        System.out.println("Possible problem with user input...");
                        System.out.println("User input "+ token+"  was changed to "+newToken);}
                return newToken;}

        public static void showFile(String fileWithPath) throws IOException{
                 BufferedReader br = new BufferedReader(new FileReader(fileWithPath));
                 String line = null;
                 while ((line = br.readLine()) != null) {
                   System.out.println(line);}
                 br.close();}
        public static ArrayList<String> returnFile(String fileWithPath){
                ArrayList<String> contents=new ArrayList<String>();
                try {
                        BufferedReader br = new BufferedReader(new FileReader(fileWithPath));
                        String line = null;
                        while ((line = br.readLine()) != null) {
                                    contents.add(line);}
                        br.close();
                } catch (IOException e) {
                        e.printStackTrace();}
                return(contents);}

        /*
```

```
 * a method that just try-catches the creation of a file.
 * Just in case.
 */
public static void mkFile(File filenameANDpath){
        File newfile=filenameANDpath;
        try{
                newfile.createNewFile();}
        catch(IOException ioe){
                System.err.println("Error in making files\n "+ioe);}}
public static void populateBases(){
        bases.add('A');
        bases.add('C');
        bases.add('G');
        bases.add('T');
        bases.add('a');
        bases.add('c');
        bases.add('g');
        bases.add('t');
        bases.add('U');
        bases.add('u');
        bases.add('R');
        bases.add('r');
        bases.add('Y');
        bases.add('y');
        bases.add('S');
        bases.add('s');
        bases.add('W');
        bases.add('w');
        bases.add('K');
        bases.add('k');
        bases.add('M');
        bases.add('m');
        bases.add('B');
        bases.add('b');
        bases.add('D');
        bases.add('d');
        bases.add('H');
        bases.add('h');
        bases.add('V');
        bases.add('v');
        bases.add('N');
        bases.add('n');
        bases.add('.');
        bases.add('-');}
public static String readSeqFromFasta(File fileWithPath){
        String finalSeq="";
        boolean hitCarrot=false;
        try {
                BufferedReader read=new BufferedReader( new FileReader(fileWithPath));
                String line = null;
                while((line=read.readLine())!=null){
                        if(line.charAt(0)=='>'){
                                if(hitCarrot==false){
                                        hitCarrot=true;
                                        continue;}
                                else{
                                        return(finalSeq);}}
                        else{
                                String seq=read.readLine();
                                finalSeq+=seq;}}
        } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();}
        return finalSeq;}
public static String validateSeq(String seq){
```

```
                char[]seqChars=seq.toCharArray();
                for(char i:seqChars){
                        if(!bases.contains(i)){
                                return "";}}
                return seq.toUpperCase();}
        public static void print(Object o){
                System.out.println(o);}}
```

```
//geneTreeMethod.java
import java.util.ArrayList;

public interface geneTreeMethod {
        //format seqs should put the seqs in the database into
        // the format needed to run the genetree method and write
        // them to some location tbd. It should return the full filenames
        // of every entry. If a program requires an alignment, then
        // an alignmentMethod interface should be instanciated locally
        public ArrayList<String> formatSeqs(ArrayList<String> seqNames, ArrayList<String> seqs);
        //should be last method called which will clear the
        //files from memory
        public void removeFormattedSeqs();

        //should get the command line parameters in a way that allows
        //it to be tacked on to the end of the program call
        public String getParams(ArrayList<String> params);

        //should return the invokation of the program as a string
        //to be called from the system. Will not be OS independent
        //and will likely include calls to get params and format Seqs
        public String invokation();

        //returns a string designating the path to write files out, if
        //necessary. may be used in invokation.
        public String outDir();}

//alignmentMethod.java
import java.util.ArrayList;

public interface alignmentMethod {
        //should return a list of any size of filenames to be used in the alignment
        //if it only needs one big file, for instance, the length will be
        //size 1.
        public ArrayList<String> getFilesToAlign();

        //returns params as they would be tacked onto the invokation
        // of the program as a single string
        public String getAlignParams(ArrayList<String>params);

        //invokes the alignment command and returns the location of
        //the resulting aligned file.
        public String align(ArrayList<String>info);}
```