Bayesian Models for Phylogenetic Trees

Clarence Leung*[1]

[1]McGill Centre for Bioinformatics, McGill University, Montreal, Quebec, Canada

ABSTRACT

Introduction: Inferring genetic ancestry of different species is a current challenge in phylogenetics because of the immense raw biological data to be analyzed. Computational techniques are necessary in order to parse and analyze all of such data in an efficient but accurate way, with many algorithms based on statistical principles designed to provide a best estimate of a phylogenetic topology. **Methods:** In this study, we analyzed a class of algorithms known as Markov chain Monte Carlo (MCMC) algorithms, which uses Bayesian statistics on a biological model, and simulates the most likely evolutionary history through continuous random sampling. We combined this method with a Python-based implementation on both artificially generated and actual sets of genetic data from the UCSC Genome Browser. **Results and Discussion:** We observe that MCMC methods provide a strong alternative to the more computationally intense likelihood algorithms and statistically weaker parsimony algorithms. Given enough time, the MCMC algorithms will generate a phylogenetic tree that eventually converges to the most probable configuration.

*Corresponding author:
clarence.leung@mail.mcgill.ca
Received: 2 January 2012
Revised: 2 March 2012

INTRODUCTION

GENETICS AND PHYLOGENY

The rise of genetics in the past few decades has led to a significant shift in the techniques and primary focuses of biological research. Quantitative data analysis has begun to replace the purely observational techniques used in the past, and new fields of study have sprouted in response to considerable data that we have yet to analyze and apply *(11)*.

Phylogenetics, the study of the evolutionary relatedness between organisms, has been one such field bombarded by an influx of raw data. From the study of taxonomy, the identification and classification of organisms, phylogenetics research has merged quantitative data and gene inheritance theories into the development of past evolutionary inferences. The techniques used in phylogenetic analysis range from computing evolutionary distances between two individual organisms, to estimating divergence times between species, to even analyzing diversification rates of all life at the macroscopic level *(6)*.

Fig 1. A sample DNA dataset represented in digital format. Source: Wikimedia Commons

The phylogenetic tree is an intuitive way of representing these phylogenetic relationships, and has long been used by researchers to model the unique genetic characteristics of each individual species. Topologically, a phylogenetic tree is composed of two groups of sub-structures, nodes and branches. The nodes of a phylogenetic tree can represent any unit of phylogeny, ranging from a single gene to an entire taxon of organisms *(3, 8)*. Similarly, the branches of a tree can represent any method of describing the differences between any two units, at the genetic level (such as the Hamming distance between any two DNA sequences) or even time *(10)*. Though phylogenetic trees have historically been used to model taxonomy because they are similar to the actual evolutionary patterns of adaptive radiation and lineage splitting, phylogenetic trees in today's studies have been used as more dynamic models to demonstrate the specific processes, such as nucleotide mutation, that lead to those macroscopic changes *(2)*.

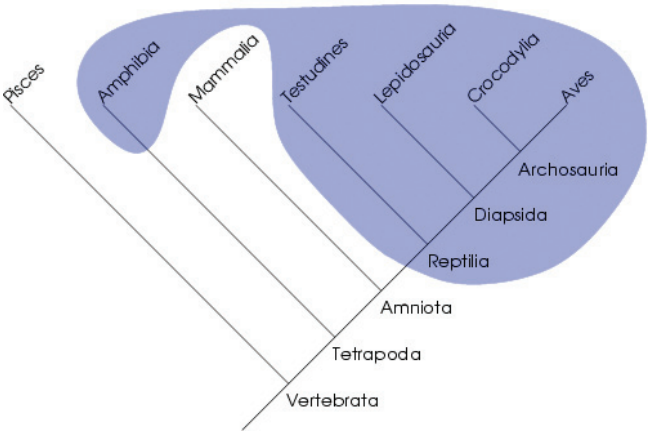

Fig 2. A rooted phylogenetic tree for sample rRNA genes, showing the branches Bacteria, Archaea, and Eucaryote. Source: Wikimedia Commons

In recent years, phylogenetic inferences have begun to rely increasingly on large amounts of molecular data sets, such as the sequences of DNA or amino acids because of the growing ease and reduced costs of obtaining sequence data due to more efficient sequencing techniques *(5)*. As more and more quantitative data is gathered in various fields, phylogeneticists must analyze and process that data to modify their predictions of evolutionary relationships.

COMPUTATIONAL PHYLOGENETICS

Computational methods are necessary to investigate these sequence datasets, as their sheer size make manual data analysis nearly impossible. Algorithmic methods of solving phylogenetic problems have been developed to solve these problems in a feasible computational time and compute significant results including phylogenetic tree inferences. Though there are many types of algorithms used in phylogenetics, they all rely on the use of computational statistics in order to generate predictions of the evolutionary past, given the molecular sequences of modern day organisms.

Algorithms in phylogenetic tree inference will take as input molecular sequences of modern-day organisms, and set them as the external nodes of the tree, or the nodes that do not have child nodes below it. Through different calculation methods, these algorithms generate statistical phylogenetic predictions of the desired genetic information of the parent nodes and the branches between each node, working up through the tree until the root is reached. The final output of the algorithm is a completed phylogenetic tree, with the most likely inferences for the node and branch parameters solved. These algorithms can be divided into one of three main types, classified by their method of calculation *(2, 6, 9)*.

MAXIMUM PARSIMONY:
Using this class of algorithms, the most likely phylogenetic tree is the one that has the least evolutionary change that results in the observed data. Each sub-tree of the phylogenetic tree is scored by a scoring mechanism, which is determined by the number of steps required to change from a parent node to a child node, for each node in the sub-tree. These algorithms are simple, but often inconsistent, as adding more data to a calculated tree will sometimes lead to a completely different tree, rather than converging on the same result.

MAXIMUM LIKELIHOOD:
Under this class of algorithms, the sequence data of a specific sub-tree is scored, and the scoring function works up the tree. However, instead of scoring with a function that calculates the least amount of change, the transition probability of that character changing at any point in the sub-tree is calculated by a specific probability model based on the character of each sequence, such as a single nucleotide or amino acid. Scoring is based on the combined likelihoods determined by the function, and produces a much more likely tree than parsimony. However, these algorithms tend to be extremely computationally intensive, especially as the number of parameters accepted by the probability model increases.

BAYESIAN INFERENCE:
Under this class of algorithms, the same character transition probability model is used, as in maximum likelihood. However, rather than determining the tree heuristically with a scoring function, as in maximum likelihood, Bayesian algorithms run continuous simulations of generating evolutionary trees through Markov chain Monte Carlo (MCMC) algorithms. These trees are randomly generated with probability based on a given density, and are based upon Bayes' theorem, which determines the conditional probability of a specific tree given existing data. These algorithms are faster and more efficient than other algorithms, and are able to accept many different parameters while generating converging trees.

MARKOV CHAIN MONTE CARLO ALGORITHMS

We want to calculate the probability of a specific tree given the current data. By Bayes' theorem, we note this as:

$$P(T|D) = P(T) \cdot \frac{P(D|T)}{P(D)}$$

Where P(T|D) represents the probability of the tree, given the data, P(T) represents the overall probability of the tree, and P(D|T)/P(D) the effect of T on the probability of D. In Bayesian inference, these are the posterior distribution, the prior distribution, and the likelihood, respectively. These probabilities are simple to calculate with discrete probabilities, but difficult to calculate with biological probability models, which require multi-dimensional integrals that can combine both continuous and discrete calculations.

Instead of directly sampling from the complex distribution, A possible solution is to construct a Markov chain, a chain system that transitions from one state to another while satisfying the Markov property, where the probability of the next state of the system is dependent only on the current state of the system. Formally, we define a Markov chain as:

$$P(X_{n+1} = x|X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = P(X_{n+1} = x|X_n = x_n).$$

Where all of the possible probabilities of the Markov chain $\{x_1, x_2, \ldots, x_n\}$ are known as its space.

To sample from a complex distribution, a Markov chain of probability distributions can be created such that its equilibrium state, where the Markov chain converges to some steady state, will be equivalent to that complex distribution. Thus, the correctness of the MCMC sampler will improve after each iteration of the Markov chain, and the probability distribution will be closer to the actual distribution we are trying to sample. Generating the Markov chain is not difficult, as we can start from any given distribution, and adjust it depending on the given density function at each iteration.

However, determining when an MCMC sampler has converged is more difficult because the randomness of the algorithm could potentially generate several states without having actually made any changes, but not be equivalent to the target distribution.

METROPOLIS-HASTINGS SAMPLER

A specific version of the MCMC sampler for probability distributions is the Metropolis-Hastings sampler, which allows sampling from any probability distribution if a function proportional to the corresponding probability density is known. For some complex distribution $\Pi(x)$ and its corresponding density $\pi(x)$, and some transition proposal distribution $Q(x, y)$ and its density $q(x, y)$, if we are currently at state x in the chain, we select a draw for a candidate state y from $Q(x, \cdot)$ with probability $\alpha(x)$ given by:

$$\alpha(x, y) = min\left\{\frac{\pi(y)q(y, x)}{\pi(x)q(x, y)},\ 1\right\}$$

Note that $\pi(y)/\pi(x)$ is the likelihood ratio between the current state and the next state, and $q(y,x)/q(x,y)$ is the ratio of the proposal density in two directions to adjust for density skewness. If the density is symmetric, then the ratio of the proposal density in the two directions is 1, which reduces our algorithm to only be dependent on the likelihood ratio, and converts it into a

Metropolis sampler, a subclass of the generalized Metropolis-Hastings algorithm.

If the combined probability π(y)q(y,x)/ π(x)q(x,y) is greater than 1, then the draw is automatically accepted, and the Markov chain will now be in state y. Otherwise, we transition to state y with that probability, so there is a possibility that we reject the draw and remain in our current state with probability 1 – π(y)q(y,x)/ π(x)q(x,y).

BAYESIAN INFERENCE IN PHYLOGENETICS

In MCMC algorithms acting on phylogenetic trees, a Metropolis-Hastings sampler can be created where π(x) will be the conditional density of the candidate phylogenetic tree, given the observable character sequences *(3)*. The proposal density q(x) will modify the topology of the branches of the tree that has been affected, and the character sequences of any nodes of the tree that have been affected. The density q(x) will be dependent on a character transition model that takes into account outside parameters that can include time, character transition ratios, and rates of transition *(4)*. We describe an example of such a model in the following section for a series of nucleotide sequences where nucleotide mutation is determined by the Hasegawa, Kishino, and Yano (1985) nucleotide transition model, and describe an implementation *(8)*.

DESCRIPTION OF THE MODEL

NUCLEOTIDE TRANSITION MATRIX

The Hasegawa, Kishino, and Yano (HKY85) nucleotide transition model allows for the consideration of five parameters: the four base frequencies of the possible nucleotides adenine, thymine, cytosine, and guanine, denoted as π_A, π_T, π_C, π_G and a transition/transversion rate parameter κ that allows for the differentiation between the transitions of purines (A, G) and transitions of pyramidines (C, T).

This can be expressed in the rate matrix Q:

$$Q = \begin{pmatrix} * & \kappa\pi_C & \pi_A & \pi_G \\ \kappa\pi_T & * & \pi_A & \pi_G \\ \pi_T & \pi_C & * & \kappa\pi_G \\ \pi_T & \pi_C & \kappa\pi_A & * \end{pmatrix}$$

The transition probability for some time t and nucleotide transition I to j, $Q_{ij}(t)$, can then be described as:

$$Q_{ij}(t) = \begin{cases} \pi_j + \pi_j(\frac{1}{\lambda_j} - 1)e^{-\alpha t} + (\frac{\lambda_j - \pi_j}{\lambda_j})e^{-\alpha\gamma_j t} & \text{if } i = j \\ \pi_j + \pi_j(\frac{1}{\lambda_j} - 1)e^{-\alpha t} + (\frac{\pi_j}{\lambda_j})e^{-\alpha\gamma_j t} & \text{if } i \neq j, \text{transitional event} \\ \pi_j(1 - e^{-\alpha t}) & \text{if } i \neq j, \text{transversional event} \end{cases}$$

Where α is some normalizing constant that allows all probabilities to sum to 1, $\lambda_j = \pi_A + \pi_G$ if j is a purine (A or G) or $\lambda_j = \pi_C + \pi_T$ if j is a pyramidine (C or T), and $\gamma_j = 1 + (\kappa - 1)\lambda_j$.

METROPOLIS-HASTING ON TREES

We draw a potential state for the next tree in the Markov chain with an algorithmic method in four steps:

1. REARRANGE THE LOCAL TOPOLOGY

A random internal node that has both a parent and children is chosen, which is designated our target node T. We now rearrange the local topology of the node, which consists of the neighboring nodes C1, C2, and S, representing the first child, the second child, and the sibling of the target node respectively. A new topology is chosen by randomly rearranging the positions of those nodes, and their subtrees as well.

2. SELECT A NEW TARGET TIME

Using the new topology, a new time for the target is selected by choosing from the density:

For $max\{t_{C1'}, t_{C2'}\} \leq t_{T'} \leq t_P$:

$$g(t_{T'}) = \frac{\sum_{v \in D^m} P_{v_P v}(t_P - t_{T'}) P_{v v_{C1'}}(t_{T'} - t_{C1'}) P_{v v_{C2'}}(t_{T'} - t_{C2'})}{\int_{max\{t_{C1'}, t_{C2'}\}}^{t_P} \sum_{v \in D^m} P_{v_P v}(t_P - t_{T'}) P_{v v_{C1'}}(t_{T'} - t_{C1'}) P_{v v_{C2'}}(t_{T'} - t_{C2'})}$$

Note that the selected time of the target node must fall between the time of the parent node, given by t_p and the time of its nearest child, given by $\max(t_{C1'}, t_{C2'})$, after the nodes have had their topology rearranged.

When calculating this time, we do not actually have to sum all 4^m different nucleotide sequences when implementing the selection. Because the nucleotide sequences of the parents and the descendants in the summation are always fixed, we actually will only need to evaluate $4^3 = 64$ total terms per iteration, which significantly increases the algorithm's speed.

3. SELECT A NEW TARGET SEQUENCE

When determining the nucleotide sequence, this density can be simplified to a series of four-nucleotide draws, with frequencies dependent on a four-category multinomial distribution that is recalculated at each iteration. We can sample individually from a categorical distribution for each nucleotide in the sequence, as the probability of a mutation for each nucleotide is not dependent on any other states in the current iteration.

For $v_{T'} \in D^m$:

$$h(t_{T'}) = \frac{P_{v_P v_{T'}}(t_P - t_{T'}) P_{v_{T'} v_{C1'}}(t_{T'} - t_{C1'}) P_{v_{T'} v_{C2'}}(t_{T'} - t_{C2'})}{\sum_{v \in D^m} P_{v_P v}(t_P - t_{T'}) P_{v v_{C1'}}(t_{T'} - t_{C1'}) P_{v v_{C2'}}(t_{T'} - t_{C2'})}$$

4. ACCEPTANCE OR REJECTION OF THE CANDIDATE

After selecting this candidate state, the algorithm must decide whether or not it is accepted as the next state in the chain or rejected.

$$\alpha(x,y) = min\left(\frac{\int_{max\{t_{C1'},t_{C2'}\}}^{t_P}\sum_{v\in D^m} P_{v_P v}(t_P-t)P_{vv_{C1'}}(t-t_{C1'})P_{vv_{C2'}}(t-t_{C2'})dt}{\int_{max\{t_{C1},t_{C2}\}}^{t_P}\sum_{v\in D^m} P_{v_P v}(t_P-t)P_{vv_{C1}}(t-t_{C1})P_{vv_{C2}}(t-t_{C2})dt}\frac{P_{v_P v_{S'}}(t_P-t_{S'})(n_{T'}+1)}{P_{v_P v_S}(t_P-t_S)(n_T+1)},1\right)$$

This is the α(x, y) as noted before in the MCMC algorithm, the combination of the target distribution and the proposal density. If the probability of selection is greater than or equal to 1, then we automatically select the new state, with the new topology, time, and sequence. Otherwise, we will move to the new state with probability α, and stay at our current state (reject the candidate) with probability 1 – α.

EXPERIMENTAL DATA

We compared the MCMC algorithm with several known techniques in computational phylogenetics, as described above. An implementation of the MCMC algorithm was created in the Python programming language, with the NumPy and SciPy scientific computing libraries, and compared against the non-Bayesian algorithms found inside the PHYLIP phylogenetic inference package, written in C. The Python MCMC algorithm had its results validated with MrBayes, a generalized MCMC phylogenetic inference package written in R.

Three alternate algorithms were chosen from the PHYLIP package to demonstrate the variety of methods available to calculate the most likely phylogenetic tree:

1.DNAPARS
A maximum parsimony algorithm based on several improvements to the Fitch algorithm (1971).

2.DNAPENNY
A maximum parsimony algorithm using the ``branch and bound" method devised by Hendy and Penny (1982).

3.DNAML
A maximum likelihood algorithm based on an algorithm devised by Felsenstein (1981) and a model by Hasegawa et al. (1985).

ARTIFICIALLY GENERATED DATA

The algorithm was initially tested on a small artificially generated dataset obtained from the PHYLIP phylogenetic inference package. This dataset is designed to test the accuracy of phylogenetic algorithms through simulations of nucleotide mutation starting from a chosen root sequence.

Sequence Label	Nucleotide Sequence
Alpha	AACGTGGCCACAT
Beta	AAGGTCGCCACAC
Gamma	CAGTTCGCCACAA
Delta	GAGATTTCCGCCT
Epsilon	CTGATGTCCGCAT
Zeta	CAGATGGCCACAT
Eta	GAGATGGCCGCAT
Theta	CTGTTCGCCACCT

Fig 3. The small artificial dataset used to test the validity of the algorithm.

The algorithms were capable of producing trees with similar topologies, although some tree rearrangement is required. We repeated this method multiple times with several different bootstrapped versions of the initial topology, using PHYLIP's BOOTSTRAP program to generate a new shuffling of DNA sequences. Using the CONSENSE program in PHYLIP, we built a consensus tree for each different algorithm type, with a chain of 500 trees for the MCMC algorithm, and 500 random trees starting from the bootstrapped topologies for the rest.

The ratios on the intersections of each branch represent the number of trees sampled that contained that specific ``majority" sub-tree. In particular, if 0.99 is the ratio on the intersection, then 0.99 of all trees sampled would contain that specific subtree on the branch.

UCSC GENOME BROWSER DATA

After validating the data with the artificial dataset, actual sequence data from the UCSC Genome Browser was used to determine the applicability of the MCMC algorithm on a larger dataset, based on experimentally-obtained sequence alignments. A 896-nucleotide strand from the MT-ND4 (mitochondrially encoded NADH dehydrogenase 4) gene on human alignment hg19 was taken and compared with similar regions found in several other primate species such as the tarsier (with alignment tarSyr1), the gorilla (with alignment gorGor3), and the orangutan (with alignment ponAbe2).

After calculating our first optimal tree, we determined the consensus tree for each algorithm using the same method as before.

The average ratios for the subtrees of each method were 0.84 for maximum parsimony, 0.88 for branch and bound, 0.98 for maximum likelihood, and 0.93 for MCMC.

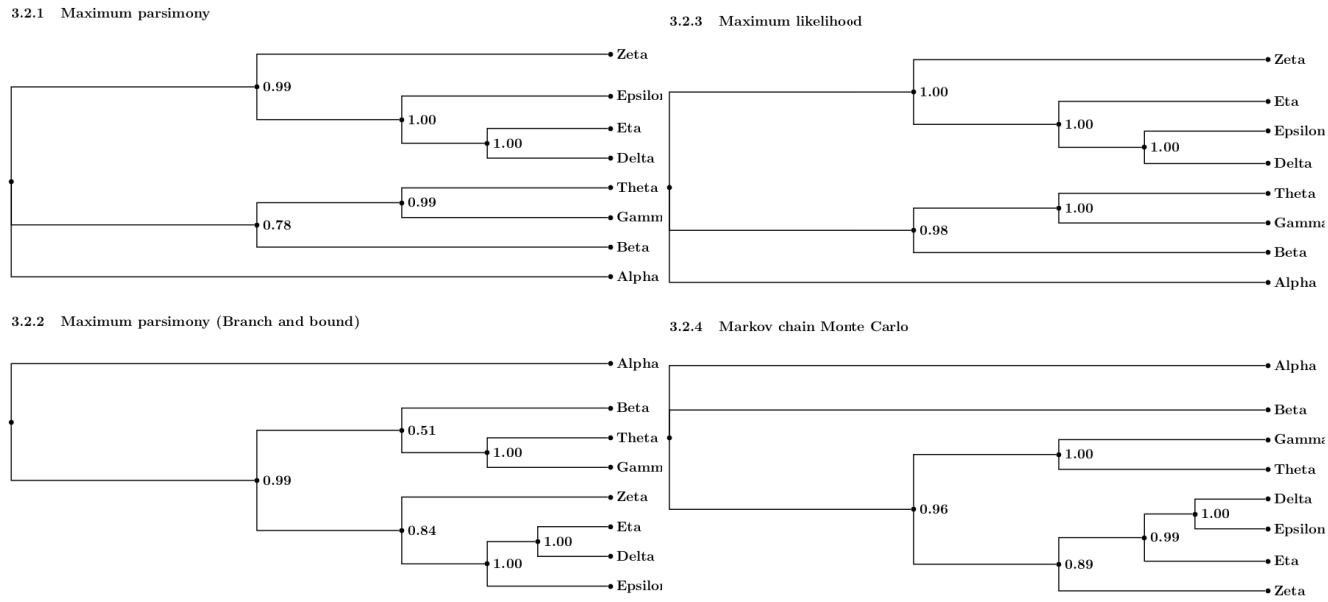

Fig 4. The resulting consensus trees for the different algorithms on the artificial dataset.

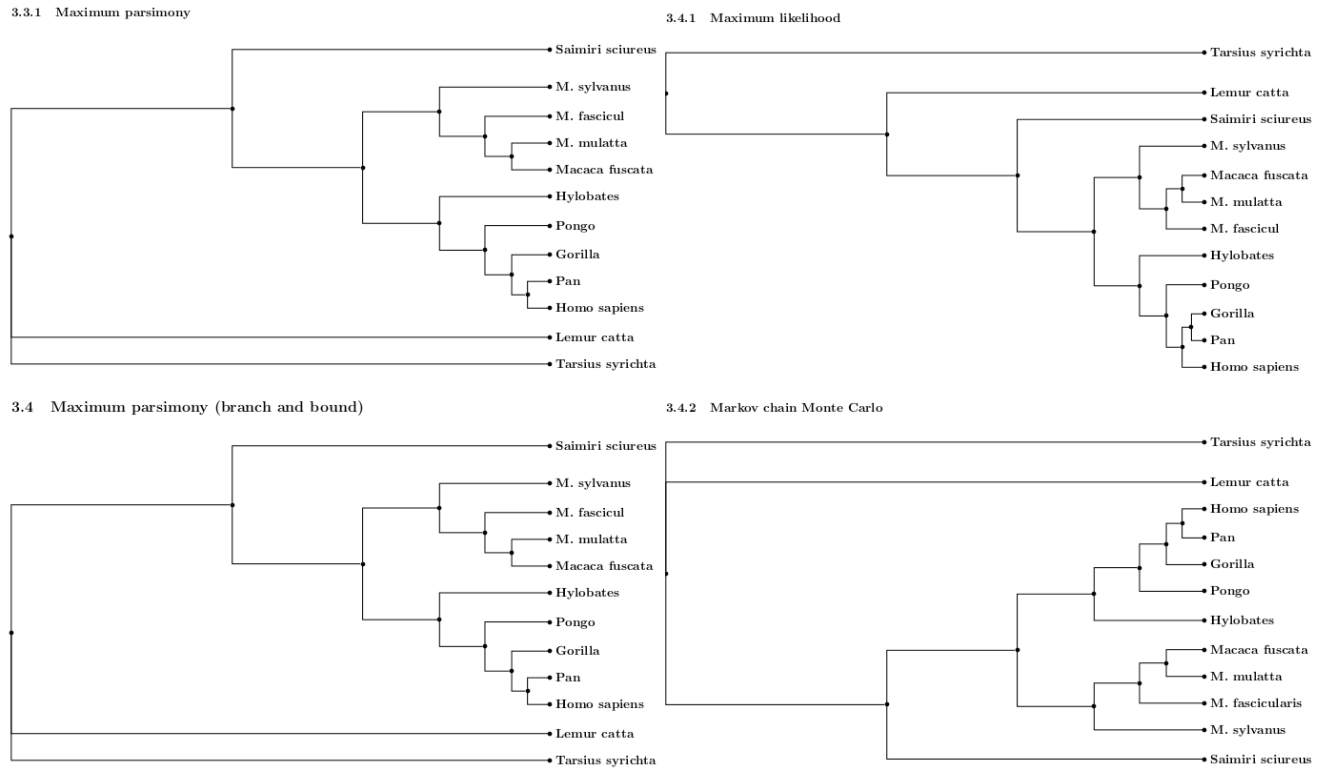

Fig 5. The resulting consensus trees for the different algorithms on the UCSC Genome Browser dataset.

DISCUSSION

COMPARISON WITH OTHER ALGORITHMS

The MCMC algorithm performed well with respect to accuracy and efficiency in each of the cases we considered, both on the artificial sample dataset and actual genetic data obtained from the UCSC Genome Browser. However, the implementation of this specific algorithm was slower than established algorithms because it was developed in the Python programming language for rapid development, rather than the C programming language for computational efficiency, as were the maximum likelihood and maximum parsimony algorithms. However, in comparison with an R implementation of the MCMC algorithm in MRBAYES, the maximum likelihood algorithm took significantly more computational power and more time in order to determine the final phylogenetic tree.

From the consensus tree, the algorithms based upon Bayesian statistical models (the MCMC and maximum likelihood algorithms) generated phylogenetic trees that were more similar to each other than to ones generated by the parsimony algorithms. This is an expected outcome, as maximum parsimony is known to statistically deviate from similar but slightly different conditions. Felsenstein (1978) showed that maximum parsimony algorithms will often result in significantly different outcomes if there is a scenario under long-branch attraction scenarios, which tend to occur with rapidly evolving species *(4)*. Although Bayesian methods use probabilistic algorithms and will have more random initial phylogenetic trees, the ability for probability distributions to converge will in fact result in more stable sets of trees. However, these methods rely on having a correct stochastic model for the transition probabilities of each character (be it a nucleotide or an amino acid). Many parameters must be taken into consideration in the construction of these stochastic models, and a model with false or incomplete assumptions can create a chain of improper trees.

PYTHON FOR BIOINFORMATICS

The Python programming language, with scientific computing libraries NumPy and SciPy, were used to implement our algorithm. Unlike the more typically used low-level programming languages for bioinformatics such as C or Fortran, or those with powerful statistical libraries such as R, Python allowed for much more rapid development and prototyping of the algorithm, with many low level memory issues being abstracted by the Python interpreter. NumPy and SciPy, however, provided fast vectorization and many probability functions, many of which were wrappers for existing C or Fortran functions, which overcame some of the expected trade-off of algorithm speed for development time.

FUTURE WORK

Phylogenetics remains a significant field in both the biological sciences and computing. As biological data becomes more widely available with the development of new experimental biology techniques, modeling of DNA and proteins will become more precise as more prior parameters can be considered for the statistical model. Current research *(1, 6)* includes the application of population-level parameters beyond molecular-level parameters in the creation of the MCMC model, which has seen limited success. The difficulty now lies in the complexity of population-level parameters, some of which cannot be easily quantified. Environmental-level parameters will be the next step, but until researchers have the computational power to simulate an entire environment, this is not possible.

In addition to taking more parameters, MCMC algorithms should also be able to output more phylogenetic data than just the predicted ancestral sequences and times. Other studies have shown that extended MCMC algorithms can even assist in taxonomic classification, by helping to calculating taxonomic circumscription limits *(2)*. These MCMC algorithms not only produce sequence data, but provide possible limits for species categorization. In the future, MCMC algorithms may create a new species classification system, based on genetics, as the Linnaeus system of classifying species by only their physical characteristics is outdated.

ACKNOWLEDGEMENTS

I would like to show my gratitude to Professor Russell Steele for his guidance in this project, and especially for the creators of the open source software packages that were used in this study.

REFERENCES

1. B. Arbogast, S. Edwards, J. Wakely, J. Slowinski, *Annu Rev Ecol Syst* **33**,707-740 (2002)
2. N. Bell, J. Hyvonen, *Am J Bot* **97**, 566-578 (2010)
3. H. Fan, L. Kubatko, *Mol Phy and Evo* **59**, 354–363 (2011)
4. J. Felsenstein, *Syst Zool* **27**, 401-410 (1978)
5. P. Green, *Biometrika* **82**, 711-732 (1995)
6. J. Hey, R. Nielsen, *P Natl Acad Sci USA* **104**, 2785-2790 (2006)
7. D. Husmeier, G. McGuire, *Bioinformatics* **18**, Sup 1 (2002)
8. S. Li, D. Pearl, H. Doss, *J Am Stat Assoc* **95**, 493-508 (1996)
9. B. Mau, M. Newton, B. Larget, *Biometrics* **55**, 1-12 (1999)
10. B. Rannala, Z. Yang, *Genetics* **164**, 1645-1656 (2003)
11. F. Ronquist, A. Deans, *Annu Rev Entomol* **55**, 189-206 (2010)
12. A. Siepel, D. Haussler, *J Comp Bio* **11**, 413-428 (2004)
13. Z. Yang, B. Rannala, *Mol Bio and Evo* **14**, 717-724 (1997)