TreeFlow: probabilistic programming and automatic differentiation for phylogenetics

Christiaan Swanepoel,[*,1,2] Mathieu Fourment,[3] Xiang Ji,[4]
Hassan Nasif,[5] Frederick A Matsen IV,[5,6,7] Alexei Drummond[1,8]

[1]Centre for Computational Evolution, The University of Auckland, Auckland, New Zealand;

[2]School of Computer Science, The University of Auckland, Auckland, New Zealand, 1010;

[3]Australian Institute for Microbiology and Infection, University of Technology Sydney, Ultimo NSW, Australia;

[4]Department of Mathematics, Tulane University, New Orleans, USA;

[5]Public Health Sciences Division, Fred Hutchinson Cancer Research Center, Seattle, Washington, USA;

[6]Department of Statistics, University of Washington, Seattle, USA;

[7]Department of Genome Sciences, University of Washington, Seattle, USA;

[8]School of Biological Sciences, The University of Auckland, Auckland, New Zealand, 1010;

Abstract

Probabilistic programming frameworks are powerful tools for statistical modelling and inference. They are not immediately generalisable to phylogenetic problems due to the particular computational properties of the phylogenetic tree object. TreeFlow is a software library for probabilistic programming and automatic differentiation with phylogenetic trees. We show that it provides reasonable performance for gradient-based inference algorithms compared to specialist computational libraries for phylogenetics. We also demonstrate how TreeFlow can be used to quickly implement and assess new models.

1 Introduction

Traditionally, phylogenetic analyses have been performed by specialist software [1–3]. A number of software packages exist that implement a broad but predefined collection of models and associated specialized inference methods. Typically, inference is handled by carefully crafted but computationally costly stochastic optimisation or Markov Chain Monte Carlo (MCMC) methods [4, 5]. In contrast, in other realms of statistical analysis, *probabilistic programming* software libraries have entered into widespread use. These allow the specification of almost any model as a probabilistic program, and inference is provided automatically with a generic inference method. Exploiting the power of probabilistic programming in phylogenetic analyses could significantly accelerate research.

Probabilistic programming tools, such as BUGS [6], Stan [7], PyMC3 [8], Pyro [9], and TensorFlow Probability [10] allow users to specify probabilistic models by describing the generative process with code as a *probabilistic program*. Advancements in automatic inference methods have allow these tools to

perform efficient inference on almost any model. Some notable examples, including Automatic Differentiation Variational Inference [11] (ADVI) and Hamiltonian Monte Carlo [12] (HMC), use local gradient information from the model's probability density function to efficiently navigate the parameter space.

One key technology that enables these gradient-based automatic inference algorithms is *automatic differentiation*. Automatic differentiation refers to methods which efficiently calculate machine-precision gradients of functions, specified by computer programs, without extra analytical derivation or excessive computational overhead compared to the function evaluation. Automatic differentiation frameworks that have statistical inference packages built on top of them include Theano [13], Pytorch [14], JAX [15] and TensorFlow [16]. Some of these extend to non-trivial computational constructs such as control flow and recursion [17].

The structure of the phylogenetic tree object is a major barrier to implementing probabilistic programming for phylogenetics. It is not clear how the association between its discrete and continuous quantities (the topology and branch lengths respectively) should be represented and handled in inference. Also, the combinatorial explosion of the size of discrete part of the phylogenetic state space presents a major challenge to any inference algorithm. Generic random search methods for discrete variables, as in the naive implementation of MCMC sampling, do not scale appropriately to allow inference on modern datasets with thousands of taxa.

One core computation that presents a challenge to automatic differentiation is the *phylogenetic likelihood* [18]. This computes the probability function of a collection of sequences given a phylogenetic tree, integrating out the character states at ancestral nodes . This is most efficiently performed through dynamic programming, which requires sequential control flow or recursion, and thus can

be non-trivial to implement in a functional automatic differentiation framework such as TensorFlow. Additionally, since the computational cost scales linearly with the number of sequences, naively computing the likelihood's gradient with respect to each branch of the tree yields a quadratic computational cost [19].

Efforts have already been made to apply probabilistic programming to phylogenetic methods [20–22]. It has been shown that *universal probabilistic programming languages*, a particularly expressive extension of the class of traditional probabilistic programming languages, can be used to express generative processes for trees, and use them to generate Sequential Monte Carlo inference schemes for complex speciation models [21]. Another tool, LinguaPhylo, provides a domain specific modelling language for phylogenetics and has the capability to generate a specification for an MCMC sampler for inference [22]. These approaches both potentially lack the inherent scalability that the automatic inference methods that accompany general-purpose probabilistic modelling tools provide.

Applying scalable inference methods to phylogenetics is a major challenge. *Probabilistic path Hamiltonian Monte Carlo* has been developed to use gradient information to sample phylogenetic posteriors across multiple tree topologies [23], though moves between tree topologies are similar to the random-walk proposal distributions available to standard MCMC routines. Hamiltonian Monte Carlo proposals for continuous parameters within tree topologies has been paired with standard MCMC moves between topologies for estimating local mutation rates and divergence times using scalable algorithms for gradient calculation [24, 25]. Another approach has used Stan to apply ADVI to phylogenetic inference within a fixed rooted tree topology [20]. Finally, variational inference has been applied to phylogenetic tree inference using a conditional clade-based distribution on unrooted tree topologies [26]. This approach has

been extended to a more expressive family of approximating distributions [27], and applied to rooted phylogenies [28].

2 Description

TreeFlow is a library for probabilistic programming in Python. It is built on TensorFlow, a computational framework for machine learning. TreeFlow leverages Tensorflow's capabilities for accelerated numerical computation and automatic differentiation. It uses the existing probabilistic modelling infrastructure provided by TensorFlow Probability, which implements standard statistical distributions and inferential machinery [10]. TreeFlow provides a phylogenetic tree representation for TensorFlow and associated input and output methods, a range of widely used phylogenetic distributions and functions, and tools for applying modern statistical inference methods to phylogenetic models.

TensorFlow's core computational object is the Tensor, a multi-dimensional array with a uniform data type. Phylogenetic trees are not immediately at home in a Tensor-centric universe, as they are a complex data structure, often defined recursively, with both continuous and discrete components. TreeFlow represents trees as a structure of Tensors; floating point Tensors representing the times and branch lengths, and integer Tensors representing the topology. Common tree operations such as traversals and navigating to ancestor nodes can be performed through these integer Tensors. TensorFlow has extensive support for 'nested' structures of Tensors, including using them as arguments to compiled functions and defining distributions over complex objects. This means computations and models involving phylogenetic trees can be expressed naturally.

A range of phylogenetic distributions and models are implemented in TreeFlow. These are primarily generative models for phylogenetic trees and models of molecular sequence evolution. Generative models of phylogenetic trees, such as

Kingman's coalescent [29] and Birth-Death-Sampling speciation processes [30], can be used to infer parameters related to population dynamics from genetic sequence data. TreeFlow implements models of nucleotide sequence evolution such as the Jukes-Cantor [31], HKY85 [32], and General Time Reversible (GTR) [33] models. It includes a standard approach for dealing with heterogeneity in mutation rate across sites based on a marginalizing over a discretized site rate distribution [34]. The probabilistic programming framework, however, allows for the use of any appropriate distribution as the base site rate distribution rather than just the standard single-parameter Gamma distribution. For example, it is straightforward to replace the base Gamma distribution with a Weibull distribution, which has a quantile function that is much easier to compute [20]. Thanks to TensorFlow's vectorized arithmetic operations, it is also natural to model variations in mutation rate over lineages by specifying parameters for multiple rates (possibly with a hierarchical prior) and multiplying by the branch lengths of the phylogenetic (time) tree. Models which can be naturally expressed this way include the log-Normal random local clock [35] and auto-correlated relaxed clock models [36].

A computation that requires special treatment is the *phylogenetic likelihood*, the probability of a sequence alignment given a phylogeny and model of sequence evolution. This typically involves integrating out character states at unsampled ancestral nodes using a *dynamic programming* computation known as *Felsenstein's pruning algorithm* [18]. The postorder tree traversal and dynamic data structure are not obviously compatible with Tensorflow's immutable data structures and focus on vectorized computations. Additionally, naive implementations result in gradient computations with problematic scaling. The computational cost of computing the derivatives of this likelihood with respect to all the branches of the phylogenetic tree could grow quadratically with re-

spect to the number of taxa, and would prohibit gradient-based inference on large datasets [19]. These issues are overcome in TreeFlow by implementing the dynamic programming data structure with TensorFlow's TensorArray construct [17]. The TensorArray is a data structure representing a collection of tensors which allows efficient implementation of the sequential computation. The *write-once* property enforced on its constituent tensors ensures that gradient computations have appropriate scaling, as evidenced by benchmarks (see Figure 1).

Another useful tool for phylogenetic inference implemented in TreeFlow is the *node height ratio transform* [37]. This has been used to infer times on phylogenetic trees using maximum likelihood in the PAML software package [38]. The ratio transform parametrizes the internal node heights of a tree as the ratio between a node's height and its parent's height. The heights can be computed from the ratios in a pre-order tree traversal. This transformation has a triangular Jacobian matrix, which means computing the determinant required for change of variable of a probability density can be computed in linear time with respect to the number of internal node heights [20]. In combination with a log transformation of the root height and a logit transformation of the ratios, a multivariate distribution that takes real values can be transformed into a distribution on internal node heights of rooted phylogenies. This has been applied to Bayesian phylogenetic inference in the context of automatic differentiation variational inference [20] and Hamiltonian Monte Carlo [24]. The ratio transform is implemented using a TensorArray-based computation as a TensorFlow Probability Bijector which provides a convenient interface for transforming real-valued distributions into phylogenetic tree distributions.

TensorFlow Probability distributions can be composed into a *probabilistic graphical model* using TensorFlow Probability's joint distribution functionality

[39]. The code to specify a joint distribution provides a concise representation of the model used in a data analysis. The ability to implement phylogenetic models in this framework means that automatic inference algorithms implemented in TensorFlow can be leveraged. The discrete topology element of phylogenetic trees is an obstacle in the usage of these algorithms, which are typically restricted to continuous latent variables. Often, the phylogenetic tree topology is not the latent variable of interest, and is not a significant source of uncertainty [40]. This can be the case when divergence times or other substitution or speciation model parameters are the focus. In this scenario, useful results can be obtained by performing inference with a fixed tree topology, such as one obtained from fast maximum likelihood methods. This is approach taken by the NextStrain platform [41], which uses the scalability afforded by a fixed tree topology to allow large-scale rapid phylodynamic analysis of pathogen molecular sequence data.

One form of statistical inference for which the gradient computation is essential is variational Bayesian inference [42]. The goal of Bayesian inference is to characterise a *posterior distribution* which represents uncertainty over model parameters. Variational Bayesian inference achieves this by optimizing an approximation to the posterior distribution which has more convenient analytical properties. Typically, the optimisation routine used in variational inference can scale to a larger number of model parameters than the random-walk sampling methods used by MCMC methods.

One concrete implementation of variational inference is *automatic differentiation variational inference* (ADVI) [11]. ADVI can perform inference on a wide range of probabilistic graphical models composed of continuous variables. It automatically constructs an approximation to the posterior by transforming a convenient base distribution to respect the domains of the model's component

distributions. It then optimizes this approximation using stochastic gradient methods [43, 44]. Typically, independent Normal distributions are used as the base distribution for computational convenience. This is known as the *mean field approximation*, and for posterior distributions that have significant correlation-structure, skew, multi-modality, or heavy tails, can introduce error in parameter estimation.

TreeFlow implements ADVI using TensorFlow Probability's bijector framework to transform a base distribution and leverages the stochastic gradient optimizers already implemented in TensorFlow. Tree variables are estimated by fixing the topology. The base distribution for the divergence times on the tree is transformed into a distribution on ratios using a logit transformation, and then into a valid set of divergence times using the ratio transformation described above [38]. ADVI opens the door to using TensorFlow's neural network framework to implement deep-learning-based variants such as variational inference with *normalizing flows* [45], which transform the base distribution through invertible trainable neural network layers to better approximate complex posterior distributions.

As well as a library for probabilistic programming with TensorFlow Probability, TreeFlow provides command line interfaces for fixed-topology inference. These allow inference on standard phylogenetic models such as those performed by specialist software. For inputs, they take a nucleotide sequence alignment in the FASTA format, a tree topology in Newick format, and a model specification in a specially structued YAML file. These allow specification of the models described above for speciation and nucleotide substitution, as well as parameter priors from a range of standard probability distributions. Command line interfaces are provided for both automatic differentiation variational inference and maximum a posteriori inference.

3 Benchmarks

The phylogenetic likelihood is the computation that dominates the computational cost of model-based phylogenetic inference. We benchmark the performance of our TensorFlow-based likelihood implementation against specialized libraries. Since gradient computations are of equal importance to likelihood evaluations in modern automatic inference regimes, we also benchmark computation of derivatives of the phylogenetic likelihood, with respect to both the continuous elements of the tree and the parameters of the substitution model. A clear difference emerges in the implementation of derivatives; TreeFlow's are based on automatic differentiation while bespoke libraries need analytically-derived gradients. Therefore, we do not necessarily expect our implementation to be as fast as bespoke software, but it does not rely on analytical derivation of gradient expressions for every model and therefore automatically supports a wider range of models.

We compare the performance of TreeFlow's likelihood implementation against BEAGLE [46]. BEAGLE is library for high performance computation of phylogenetic likelihoods. Recent versions of BEAGLE implement analytical gradients with respect to tree branch lengths [19]. We use BEAGLE via the `bito` software package [47], which provides a Python interface to BEAGLE and also numerically computes derivatives with respect to substitution model parameters using a finite differences approximation.

We also compare TreeFlow with a simple likelihood implementation [48] based on another automatic differentiation framework, JAX [15]. In contrast to TensorFlow's function mode, which the benchmarked TreeFlow implementation uses, Jax uses an eager execution model and is compatible with native Python control flow.

Benchmarks are performed on simulated data. Simulation allows the gen-

eration of a large number of data replicates with appropriate properties. Since we want to investigate the scaling of likelihood and gradient calculations with respect to the number of sequences, we simulate data for a range of sequence counts. Sequence counts are selected as increasing powers of 2 to better display asymptotic properties of the implementations. We simulate data with sequence counts ranging from 32 to 2048. Trees are simulated under a coalescent model for a given number of taxa, and then nucleotide sequences of length 1000 are simulated under a fixed rate of evolution and a HKY substitution model. Tree and sequence simulations are performed using BEAST 2 [49].

We benchmark 4 distinct computations on these datasets. Firstly, for each replicate we compute the phylogenetic likelihood under a very simple model of sequence evolution. This uses a Jukes-Cantor model of nucleotide substitution and no model of site rate variation. Secondly, we calculate derivatives with respect to branch lengths under this simple model. Thirdly, we compute the likelihood under a more sophisticated substitution model, with a discretized Weibull distribution with 4 rate categories to model site rate variation and a GTR model of nucleotide substitution. We selected the Weibull distribution for site rate variation since it is implemented in `bito`. Finally, we compute derivatives with respect to both branch lengths and the parameters of the substitution model (the 6 GTR relative substitution rates, 4 base nucleotide frequencies, and the Weibull site rate shape parameter). Each computation is performed 100 times on 10 simulated datasets.

Figure 1 shows the results of the benchmarks with a log scale on both axes. For likelihood computation with both models and branch gradient computation with the simple model `bito`/BEAGLE are at least an order of magnitude faster than TreeFlow. This is expected as BEAGLE is a highly optimized special-purpose library written in native code. The performance gap grows much

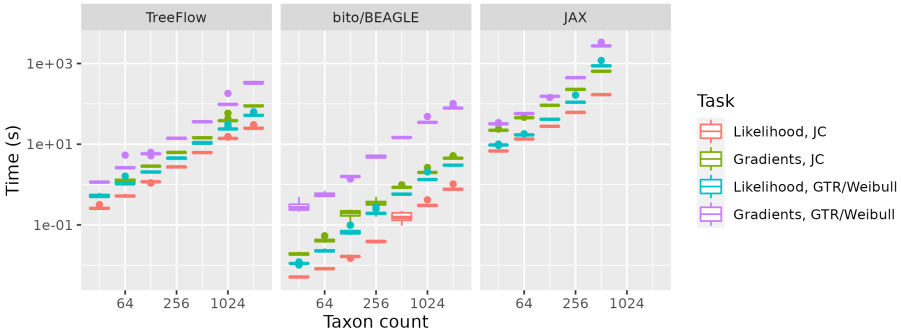

Figure 1: Times from phylogenetic likelihood benchmark (100 evaluations for 1000 sites)

Model	Computation	Method	Mean time (512 taxa)	log-log slope
JC	Likelihood	TreeFlow	6.23	1.13
		bito/BEAGLE	0.16	1.27
		JAX	168.79	1.15
	Gradients	TreeFlow	14.55	1.23
		bito/BEAGLE	0.87	1.33
		JAX	644.22	1.19
GTR/Weibull	Likelihood	TreeFlow	10.73	1.12
		bito/BEAGLE	0.58	1.40
		JAX	908.24	1.58
	Gradients	TreeFlow	**36.14**	**1.34**
		bito/BEAGLE	**14.74**	**1.41**
		JAX	**2801.35**	**1.58**

Table 1: Results of phylogenetic likelihood benchmark

smaller for computing the gradients of the more complex substitution model. `bito` performs at least 2 likelihood evaluations for each additional parameter when calculating the gradient with respect to substitution model parameters, while the overhead for substitution model parameters with automatic differentiation is minimal. We expect TreeFlow to surpass `bito`/BEAGLE for substitution models with even more parameters (e.g. codon models [50]), or those where the number of parameters grows with the number of sequences, as in the example below.

For typical real-world phylogenetic analyses, a model with many parameters such as the GTR/Weibull model in this benchmark is used. For modern Bayesian inference methods such as variational inference and Hamiltonian Monte Carlo, the gradient is the primary computation performed at each iteration. We observe that TreeFlow's times for this combination of model and computation are within an order of magnitude of `bito`/BEAGLE's, around 2-3 times slower. Therefore for applied analyses the automatic differentiation-based computations of TreeFlow presents a reasonable alternative to specialized software while similtaneously offering greater flexibility.

We also observe that the runtimes for TreeFlow are roughly an order of magnitude less than those of the JAX-based implementation. These indicate that the control flow constructs and execution model of TensorFlow are a good choice for implementing tree-based computations compared to the eager execution model of JAX.

Table 1 shows the coefficients obtained from fitting a linear model to the benchmark times where the predictor is the log-transformed number of sequences and the target is the log-transformed runtime. The slope parameter estimates from these fits is a rough empirical estimate of the polynomial degree of the computational scaling. The slope parameters for all TreeFlow compu-

tations are well below 2 and indicate a roughly linear scaling with the size of the data, and certainly one that is not worse than the analytical linear-time gradient in `bito`/BEAGLE, indicating that the TensorArray-based likelihood implementation enables linear-time branch gradients.

4 Biological examples

We used TreeFlow to perform fixed-topology phylogenetic analyses of two datasets. The first is an alignment of 62 mitochondrial DNA sequences from carnivores [51]. In both datasets maximum likelihood unrooted topologies are estimated using RAxML [1]. These topologies are rooted with Least Squares Dating [52].

In the carnivores dataset, we demonstrate the flexibility of probabilistic programming with TreeFlow by investigating variation in the ratio of transitions to transversions in the nucleotide substitution process across lineages in the tree. Early maximum likelihood analyses of mitochondrial DNA in primates detected variation in this ratio, but without a biological basis, it was attributed to a saturation effect [53]. A later simulation-based investigation showed that this was a reasonable explanation, which exposed the limitations of nucleotide substitution models for estimating the length of branches deep in the tree [54].

This problem could be approached by means of Bayesian model comparison; a lack of preference for a model allowing between-lineage variation of the ratio could indicate that the substitution model lacks the power to separate variation 'signal' from saturation 'noise'. Firstly, we construct a standard phylogenetic model with a HKY substitution model with a single transition-transversion ratio parameter (*kappa*). We perform inference using ADVI, and also using MCMC as implemented in BEAST 2 [49]. Since this model is of the form of a standard phylogenetic analysis, it could be fit using TreeFlow's command line interface. Figure 2 compares the marginal parameter estimates obtained from TreeFlow

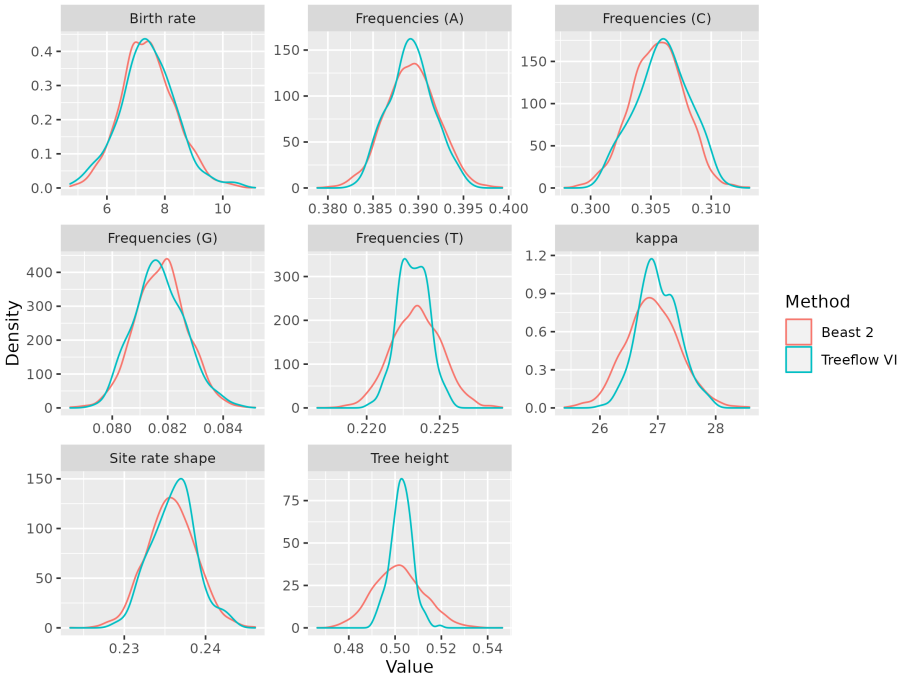

Figure 2: Carnivores base model marginal parameter estimates

and BEAST 2. The discrepancies in distribution, apparent in the estimates of the frequencies and tree height, can be attributed to the approximation error introduced by ADVI. Most importantly, the estimate of the parameter of interest, kappa, appears reasonable.

We then altered this model to estimate a separate ratio for every lineage on the tree; the implementation of this was simple and scalable as a result of TensorFlow's vectorization and broadcasting functionality. We compare the models using estimates of the marginal likelihood [55]. The *marginal likelihood*, or *evidence*, the integral of the likelihood of the data over model parameters, is typically analytically intractable and challenging to compute using Markov Chain Monte Carlo methods [56]. The closed-form approximation to the posterior distribution provided by variational inference means we can easily estimate the marginal likelihood using importance sampling, a utility provided by TreeFlow. This corrects for some of the posterior approximation error described above.

The estimated per-lineage kappa parameters are shown in Figure 4. Kappa estimates decline with the age of the lineage, which agrees with the results of the previous studies. The marginal likelihood for the transition-transversion ratio variation across lineages versus the base model was higher. This means that, under the other components of this model, the data supports variation in the ratio. This does not necessarily imply that transition-transversion ratio in the true generative process of the data varies in the same way, but could be a useful consideration in designing more sophisticated models of nucleotide substitution. The growth in uncertainty in kappa estimates deeper in the tree supports previous conclusions that nucleotide substitution models are unable to effectively estimate the number of substitutions on older branches [54]. Figure 5 shows that the kappa variation model shortens older branches, leading to a substantially reduced overall tree height estimate, with proportionally similar

```
site_category_count = 4
pattern_counts = alignment.get_weights_tensor()
subst_model = HKY()

def build_sequence_dist(tree, kappa, frequencies, site_gamma_shape):
    unrooted_tree = tree.get_unrooted_tree()
    site_rate_distribution = DiscretizedDistribution(
        category_count=site_category_count,
        distribution=Gamma(
            concentration=site_gamma_shape,
            rate=site_gamma_shape
        ),
    )
    transition_probs_tree = get_transition_probabilities_tree(
        unrooted_tree,
        subst_model,
        rate_categories=site_rate_distribution.normalised_support,
        frequencies=frequencies,
        frequencies=tf.broadcast_to(
            tf.expand_dims(frequencies, -2),
            kappa.shape + (4,)
        ),
        kappa=kappa
    )
    return SampleWeighted(
        DiscreteParameterMixture(
            site_rate_distribution,
            LeafCTMC(
                transition_probs_tree,
                expand_dims(frequencies, -2),
            ),
        ),
        sample_shape=alignment.site_count,
        weights=pattern_counts
    )

model = JointDistributionNamed(dict(
    birth_rate=LogNormal(c(1.0), c(1.5)),
    tree=lambda birth_rate: Yule(tree.taxon_count, birth_rate),
    kappa=LogNormal(c(0.0), c(2.0)),
    kappa=Sample(
        LogNormal(c(0.0), c(2.0)),
        tree.branch_lengths.shape
    ),
    site_gamma_shape=LogNormal(c(0.0), c(1.0)),
    frequencies=Dirichlet(c([2.0, 2.0, 2.0, 2.0])),
    sequences=build_sequence_dist
))
```


Figure 3: Code to specify models for carnivores analysis. Highlighted lines show changes (from previous line) to add kappa variation across lineages.

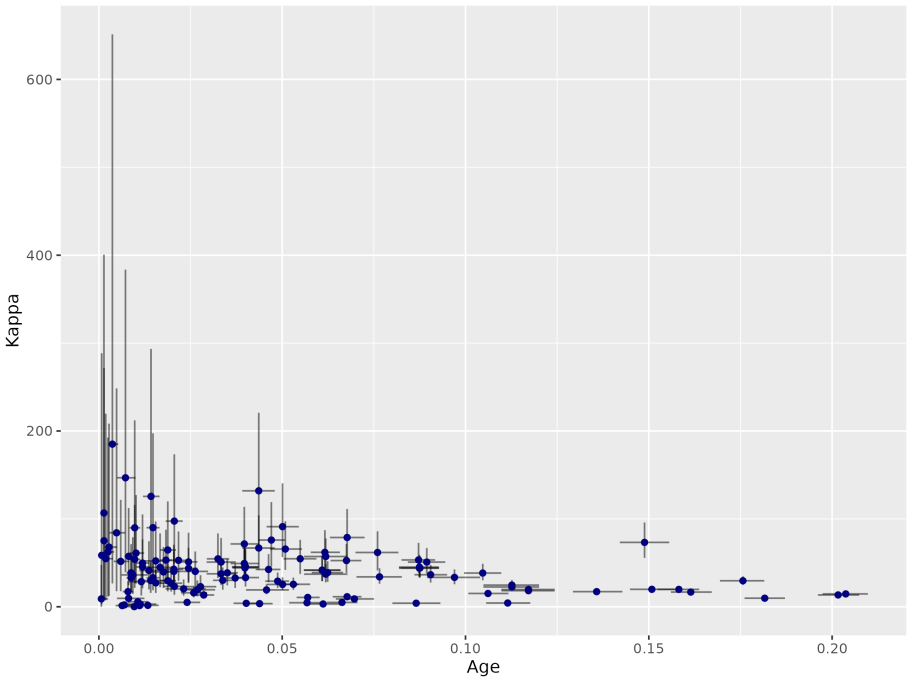

Figure 4: Lineage age (branch mean) vs estimated transition/tranversion ratio (Kappa) for carnivores dataset

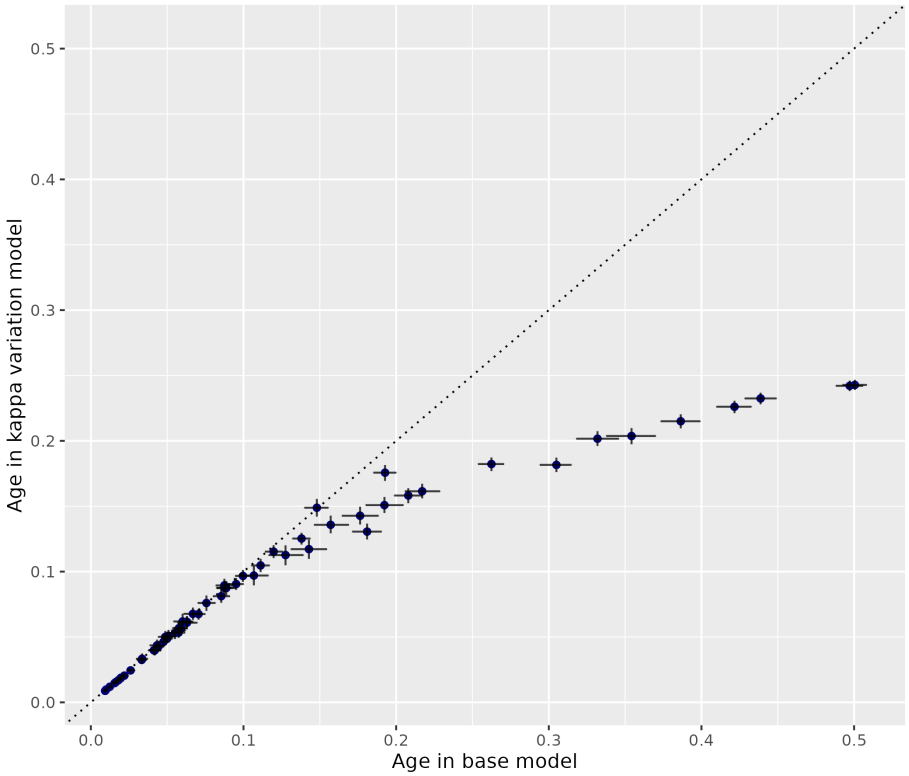

Figure 5: Node age estimates for base model vs per-lineage kappa variation model

uncertainty in node height estimates. This is not a proper dating analysis as it does not consider uncertainty in the mutation rate and is not time-calibrated, but it is clear that this model could significantly affect time estimates.

5 Discussion

The combination of flexibility and scalability of libraries like TreeFlow has the potential to be useful for rapid analysis of large modern genetic datasets, such as those assembled for tracking infectious diseases [41]. For this purpose, TreeFlow's true power would be unlocked with the implementation of improved modelling and inference tools for evolutionary rates and population dynamics.

For principled dating analyses, uncertainty in evolutionary rate parameters must be considered. Posterior distributions involving these parameters may have significant correlation structure which would be ignored by variational inference using a mean-field posterior approximation. These parameters typically receive special treatment in MCMC sampling routines [35, 57]. Implementing structured posterior approximations for these models could substantially improve the reliability of parameter estimates obtained using variational inference while scaling to extremely large datasets.

Additionally, implementing flexible models for population dynamics, such as nonparametric coalescent models [58–60], could provide valuable insights from large datasets. These models would result in complex posterior distributions, which would need to be accounted for in a variational inference scheme. If these coalescent models could be made to work effectively in TreeFlow, the probabilistic programming paradigm would allow for rapid computational experimentation with novel models of time-varying population parameters.

Finally, the most significant functionality for phylogenetic inference missing from TreeFlow is inference of the tree topology. TreeFlow's tree representation

could allow for the topology to become an estimated parameter, such as in existing on work variational Bayesian phylogenetic inference [26]. Efficiently implementing the computations required for these algorithms in the TensorFlow framework would certainly be a major computational challenge.

6 Conclusion

TreeFlow is a software library that provides tools for statistical modelling and inference on phylogenetic problems. Probabilistic programming provides flexible model definition involving phylogenetic trees and molecular sequences, and automatic differentiation enables the application of modern scalable inference algorithms. TreeFlow's automatic differentiation-based implementations of core gradient computations provide reasonable performance compared to specialist phylogenetic libraries. These building blocks have the potential to be used in valuable phylogenetic analyses of large and complex genetic datasets.

7 Software Availability

TreeFlow is an open-source Python package. Instructions for installing TreeFlow, and examples of using it as a library and command line application can be found at `https://github.com/christiaanjs/treeflow`.

References

[1] Stamatakis A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. Bioinformatics. 2014;30(9):1312–1313.

[2] Drummond AJ, Suchard MA, Xie D, Rambaut A. Bayesian phylogenetics with BEAUti and the BEAST 1.7. Molecular biology and evolution. 2012;29(8):1969–1973.

[3] Huelsenbeck JP, Ronquist F. MRBAYES: Bayesian inference of phylogenetic trees. Bioinformatics. 2001;17(8):754–755.

[4] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. The journal of chemical physics. 1953;21(6):1087–1092.

[5] Hastings WK. Monte Carlo sampling methods using Markov chains and their applications. Biometrika. 1970;57(1):97–109. doi:10.1093/biomet/57.1.97.

[6] Lunn DJ, Thomas A, Best N, Spiegelhalter D. WinBUGS-a Bayesian modelling framework: concepts, structure, and extensibility. Statistics and computing. 2000;10(4):325–337.

[7] Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, et al. Stan: A probabilistic programming language. Journal of statistical software. 2017;76(1).

[8] Salvatier J, Wiecki TV, Fonnesbeck C. Probabilistic programming in Python using PyMC3. PeerJ Computer Science. 2016;2:e55.

[9] Bingham E, Chen JP, Jankowiak M, Obermeyer F, Pradhan N, Karaletsos T, et al. Pyro: Deep universal probabilistic programming. The Journal of Machine Learning Research. 2019;20(1):973–978.

[10] Dillon JV, Langmore I, Tran D, Brevdo E, Vasudevan S, Moore D, et al. Tensorflow distributions. arXiv preprint arXiv:171110604. 2017;.

[11] Kucukelbir A, Tran D, Ranganath R, Gelman A, Blei DM. Automatic differentiation variational inference. The Journal of Machine Learning Research. 2017;18(1):430–474.

[12] Duane S, Kennedy AD, Pendleton BJ, Roweth D. Hybrid monte carlo. Physics letters B. 1987;195(2):216–222.

[13] Bergstra J, Breuleux O, Bastien F, Lamblin P, Pascanu R, Desjardins G, et al. Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for scientific computing conference (SciPy). vol. 4. Austin, TX; 2010. p. 1–7.

[14] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems. 2019;32.

[15] Bradbury J, Frostig R, Hawkins P, Johnson MJ, Leary C, Maclaurin D, et al.. JAX: composable transformations of Python+NumPy programs; 2018. Available from: `http://github.com/google/jax`.

[16] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. Tensorflow: A system for large-scale machine learning. In: 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16); 2016. p. 265–283.

[17] Yu Y, Abadi M, Barham P, Brevdo E, Burrows M, Davis A, et al. Dynamic control flow in large-scale machine learning. In: Proceedings of the Thirteenth EuroSys Conference; 2018. p. 1–15.

[18] Felsenstein J. Evolutionary trees from DNA sequences: a maximum likelihood approach. Journal of molecular evolution. 1981;17(6):368–376.

[19] Ji X, Zhang Z, Holbrook A, Nishimura A, Baele G, Rambaut A, et al. Gradients do grow on trees: a linear-time O (N)-dimensional gradient for statistical phylogenetics. Molecular Biology and Evolution. 2020;37(10):3047–3060.

[20] Fourment M, Darling AE. Evaluating probabilistic programming and fast variational Bayesian inference in phylogenetics. PeerJ. 2019;7:e8272.

[21] Ronquist F, Kudlicka J, Senderov V, Borgström J, Lartillot N, Lundén D, et al. Universal probabilistic programming offers a powerful approach to statistical phylogenetics. Communications biology. 2021;4(1):1–10.

[22] Drummond AJ, Chen K, Mendes FK, Xie D. LinguaPhylo: a probabilistic model specification language for reproducible phylogenetic analyses. bioRxiv. 2022;.

[23] Dinh V, Bilge A, Zhang C, Matsen IV FA. Probabilistic path hamiltonian monte carlo. In: International Conference on Machine Learning. PMLR; 2017. p. 1009–1018.

[24] Ji X, Fisher AA, Su S, Thorne JL, Potter B, Lemey P, et al. Scalable Bayesian divergence time estimation with ratio transformations. arXiv preprint arXiv:211013298. 2021;.

[25] Fisher AA, Ji X, Nishimura A, Lemey P, Suchard MA. Shrinkage-based random local clocks with scalable inference. arXiv preprint arXiv:210507119. 2021;.

[26] Zhang C, Matsen IV FA. Variational Bayesian phylogenetic inference. In: International Conference on Learning Representations; 2018.

[27] Zhang C. Improved variational bayesian phylogenetic inference with normalizing flows. Advances in Neural Information Processing Systems. 2020;33:18760–18771.

[28] Zhang C, Matsen IV FA. A Variational Approach to Bayesian Phylogenetic Inference. arXiv preprint arXiv:220407747. 2022;.

[29] Kuhner MK, Yamato J, Felsenstein J. Estimating effective population size and mutation rate from sequence data using Metropolis-Hastings sampling. Genetics. 1995;140(4):1421–1430.

[30] Stadler T. On incomplete sampling under birth–death models and connections to the sampling-based coalescent. Journal of theoretical biology. 2009;261(1):58–66.

[31] Jukes TH, Cantor CR, et al. Evolution of protein molecules. Mammalian protein metabolism. 1969;3:21–132.

[32] Hasegawa M, Kishino H, Yano Ta. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. Journal of molecular evolution. 1985;22(2):160–174.

[33] Tavaré S, et al. Some probabilistic and statistical problems in the analysis of DNA sequences. Lectures on mathematics in the life sciences. 1986;17(2):57–86.

[34] Yang Z. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. Journal of Molecular evolution. 1994;39(3):306–314.

[35] Drummond AJ, Ho SY, Phillips MJ, Rambaut A. Relaxed phylogenetics and dating with confidence. PLoS Biol. 2006;4(5):e88.

[36] Thorne JL, Kishino H, Painter IS. Estimating the rate of evolution of the rate of molecular evolution. Molecular biology and evolution. 1998;15(12):1647–1657.

[37] Kishino H, Thorne JL, Bruno WJ. Performance of a divergence time estimation method under a probabilistic model of rate evolution. Molecular biology and evolution. 2001;18(3):352–361.

[38] Yang Z. PAML 4: phylogenetic analysis by maximum likelihood. Molecular biology and evolution. 2007;24(8):1586–1591.

[39] Piponi D, Moore D, Dillon JV. Joint Distributions for TensorFlow Probability. arXiv preprint arXiv:200111819. 2020;.

[40] Yang Z, Nielsen R, Goldman N, Pedersen AMK. Codon-substitution models for heterogeneous selection pressure at amino acid sites. Genetics. 2000;155(1):431–449.

[41] Hadfield J, Megill C, Bell SM, Huddleston J, Potter B, Callender C, et al. Nextstrain: real-time tracking of pathogen evolution. Bioinformatics. 2018;34(23):4121–4123.

[42] Jordan MI, Ghahramani Z, Jaakkola TS, Saul LK. An introduction to variational methods for graphical models. Machine learning. 1999;37(2):183–233.

[43] Robbins H, Monro S. A stochastic approximation method. The annals of mathematical statistics. 1951; p. 400–407.

[44] Bottou L. Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010. Springer; 2010. p. 177–186.

[45] Rezende D, Mohamed S. Variational Inference with Normalizing Flows. In: Bach F, Blei D, editors. Proceedings of the 32nd International Conference on Machine Learning. vol. 37 of Proceedings of Machine Learning Research. Lille, France: PMLR; 2015. p. 1530–1538. Available from: http://proceedings.mlr.press/v37/rezende15.html.

[46] Ayres DL, Cummings MP, Baele G, Darling AE, Lewis PO, Swofford DL, et al. BEAGLE 3: Improved performance, scaling, and usability for a high-performance computing library for statistical phylogenetics. Systematic biology. 2019;68(6):1052–1061.

[47] Matsen E, Rich DH, Milanov O, Fourment M, Jun SH, Nassif H, et al.. bito;. Available from: https://github.com/phylovi/bito.

[48] Swanepoel C. phylojax;. Available from: https://github.com/christiaanjs/phylojax.

[49] Bouckaert R, Vaughan TG, Barido-Sottani J, Duchêne S, Fourment M, Gavryushkina A, et al. BEAST 2.5: An advanced software platform for Bayesian evolutionary analysis. PLoS computational biology. 2019;15(4):e1006650.

[50] Goldman N, Yang Z. A codon-based model of nucleotide substitution for protein-coding DNA sequences. Molecular biology and evolution. 1994;11(5):725–736.

[51] Suchard MA, Rambaut A. Many-core algorithms for statistical phylogenetics. Bioinformatics. 2009;25(11):1370–1376.

[52] To TH, Jung M, Lycett S, Gascuel O. Fast dating using least-squares criteria and algorithms. Systematic biology. 2016;65(1):82–97.

[53] Yang Z, Yoder AD. Estimation of the transition/transversion rate bias and species sampling. Journal of Molecular Evolution. 1999;48(3):274–283.

[54] Duchêne S, Ho SY, Holmes EC. Declining transition/transversion ratios through time reveal limitations to the accuracy of nucleotide substitution models. BMC evolutionary biology. 2015;15(1):1–10.

[55] MacKay DJ, et al. Information theory, inference and learning algorithms. Cambridge university press; 2003.

[56] Xie W, Lewis PO, Fan Y, Kuo L, Chen MH. Improving marginal likelihood estimation for Bayesian phylogenetic model selection. Systematic biology. 2011;60(2):150–160.

[57] Zhang R, Drummond A. Improving the performance of Bayesian phylogenetic inference under relaxed clock models. BMC Evolutionary Biology. 2020;20:1–28.

[58] Drummond AJ, Rambaut A, Shapiro B, Pybus OG. Bayesian coalescent inference of past population dynamics from molecular sequences. Molecular biology and evolution. 2005;22(5):1185–1192.

[59] Minin VN, Bloomquist EW, Suchard MA. Smooth skyride through a rough skyline: Bayesian coalescent-based inference of population dynamics. Molecular biology and evolution. 2008;25(7):1459–1471.

[60] Gill MS, Lemey P, Faria NR, Rambaut A, Shapiro B, Suchard MA. Improving Bayesian population dynamics inference: a coalescent-based model for multiple loci. Molecular biology and evolution. 2013;30(3):713–724.