Variational Inference with Node Embeddings (VINE) for Scalable Bayesian Phylogenetics

Adam Siepel[1,*], Rebecca Hassett[1], and Stephen J. Staklinski[1]

[1]Simons Center for Quantitative Biology, Cold Spring Harbor Laboratory, Cold Spring Harbor, NY
[*]Corresponding author: asiepel@cshl.edu

Abstract

Bayesian phylogenetic inference is now widely used but remains heavily reliant on Markov chain Monte Carlo (MCMC) sampling, which is computationally intensive and requires careful convergence monitoring. Variational inference (VI) is an appealing alternative that approximates posterior distributions without sampling, but existing variational approaches for phylogenetics have seen limited adoption owing to constraints in accuracy, speed, and scalability. Here we introduce Variational Inference with Node Embeddings (VINE), a variational phylogenetic inference method with striking improvements over prior work. VINE supports both standard DNA substitution models and CRISPR barcode-mutation models for cell-lineage phylogenies. Its key innovations are: embedding taxa in a high-dimensional Euclidean space; backpropagating gradients through fast distance-based phylogeny inference algorithms; introducing a sampling-free approximate estimator for the VI evidence lower bound; and enhancing posterior flexibility using normalizing flows. Across simulated and empirical datasets, VINE yields accurate posterior approximations for datasets with as many as 1000 taxa in a fraction of the time required for state-of-the-art MCMC-based methods.

Introduction

One of the great advances in computational molecular biology since its origins in the 1960s is the reconstruction of evolutionary trees, or phylogenies, first from amino-acid sequences and later from nucleic-acid sequences [1–5]. Phylogenetic reconstruction allows a glimpse into the deep past by extracting subtle signals of ancient relationships from genetic data obtained from living organisms today, with applications ranging from recent viral evolution to the earliest branching events on the tree of life. Beyond tree reconstruction, phylogenetic methods have also been adapted to address a broad range of related applications, including large-scale comparative genomics [6], the reconstruction of population-level genetic histories [7], and charting and combating the spread of infectious diseases such as the SARS-CoV-2 pandemic [8].

In the 1980s, the early distance- and maximum-parsimony-based phylogenetic reconstruction methods of the 1960s and 70s began to give way to approaches based on Fisher's method of maximum likelihood (ML). In a long series of battles with the methodologically conservative pheneticists (see, e.g., [9]),

Felsenstein and his colleagues gradually demonstrated that ML could be effectively adapted for the complex problem of tree reconstruction by an innovative combination of probabilistic models of nucleotide or amino-acid substitution, dynamic programming for likelihood calculations on trees, numerical optimization of continuous parameters, and heuristic searches over the space of tree topologies [10–13]. ML methods for phylogenetic reconstruction offered the benefits of statistical consistency and asymptotic efficiency with robustness to known pitfalls such as long-branch attraction. They also extended naturally to likelihood ratio tests for various hypotheses of interest and to a wide variety of modeling extensions, including rate variation among sites and relaxed molecular clocks. By the mid 1990s, ML methods had become dominant in applied phylogenetics. Over the ensuing decades, they have been refined and accelerated in widely used tools such as PAML [14], PhyML [15], HyPhy [16], RAxML [17], and IQ-TREE [18, 19], with remarkable flexibility, scalability, and speed.

Starting in the late 1990s, however, phylogenetic methods developers became interested in the new idea of treating phylogeny reconstruction as a problem in Bayesian inference, whereby a full posterior distribution was inferred by Markov chain Monte Carlo (MCMC) methods for the tree topology, the branch lengths, and the parameters governing the substitution process [20–22]. In this way, the inherent uncertainty about the evolutionary parameters of interest given the data could be accounted for in the inference procedure. Bayesian inference of phylogenies was initially resisted by some leaders in the field—particularly the practice of treating the tree topology as a random variable rather than a component of the model itself [23]. Over time, however, the approach has become broadly accepted and mature implementations such as MrBayes [24, 25], BEAST/BEAST 2 [26–28] and RevBayes [29] have become extraordinarily widely used. Any remaining misgivings about the practice have been overshadowed by its many practical advantages. A particular benefit has been the ability to summarize confidence in reconstructed clades using posterior probabilities rather than the bootstrap values used in ML, which can be difficult to interpret. More broadly, Bayesian thinking is now pervasive in phylogenetics: even many non-statisticians are accustomed to addressing the problem of uncertainty in reconstructed trees through posterior distributions.

Despite many recent advances in Bayesian phylogenetics, however (e.g., [30]), Bayesian methods still lag considerably behind the leading ML methods in speed and scalability. They are primarily limited by a reliance on MCMC sampling, which explores the full space of tree topologies through a series of discrete rearrangement operations, and inevitably stalls as the number of taxa n becomes large. (The number of possible rooted binary tree topologies is a double factorial function of n, $(2n-3)!!$, meaning that it grows faster than any simple exponential function.) In addition, MCMC-based methods often require tuning of proposal distributions and careful monitoring of convergence, sometimes making them challenging to use for non-experts.

Recognizing these limitations, investigators have recently explored a variety of alternative methods for characterizing the posterior distribution over phylogenetic trees and associated parameters. Most of these efforts have made use of some form of variational inference (VI) (sometimes called *variational Bayes* in a Bayesian setting), which in other fields has perhaps been the most successful alternative to MCMC for Bayesian inference in complex models [31, 32]. The core idea of VI is, instead of *sampling* from a high-dimensional posterior distribution, to propose a flexible alternative distribution (often from a parameteric

family, but potentially from a neural network or other source), and then to compel that distribution to approximate the true posterior by minimizing a measure of divergence between probability distributions (reviewed in [33]). In many applications, the resulting approximation can be shown to be quite good and the optimization can be accomplished considerably more quickly, and with less concern about convergence, than MCMC sampling. In phylogenetics, the first method of this kind—Variational Bayesian Phylogenetic Inference (VBPI)—was proposed six years ago by Zhang and Matsen [34] (see also [35, 36]) and subsequently extended to make use of normalizing flows and graph neural networks [37–40], among other features. VBPI has now been followed by a number of other variational phylogenetic inference methods including VaiPhy [41], VBPI-Mixtures [42], GeoPhy [43], and Dodonaphy [44] (see also [45–47]). In general, however, research in this area is still at the exploratory stage, and the available methods have yet to achieve the scalability and accuracy needed for applied phylogenetics. To our knowledge, VI has yet to be used in any published phylogenetic analysis beyond benchmarking for methods development.

An additional reason to seek more scalable Bayesian methods for phylogenetic inference comes from the recent trend toward reconstruction of cell-lineage trees from CRISPR-based lineage-tracing data [48–52]. These data sets are rapidly growing in size, now often including thousands of cells, and fast maximum-parsimony, maximum-likelihood, and machine-learning methods are emerging to process them [53–58]. At the same time, current lineage-tracing data sets often contain limited phylogenetic information, stemming from small numbers of editable sites, which leads to considerable uncertainty in tree reconstruction [59, 60]. For this reason, there has been recent interest in developing Bayesian methods for tree reconstruction from cell-lineage data [60–63]. Thus far, however, these methods have relied on MCMC using platforms such as BEAST 2 and therefore have similar limits in scalability as MCMC-based methods for species tree reconstruction.

In this paper, we revisit the problem of VI for phylogenetics, with an eye toward applications to both DNA alignments and CRISPR-based lineage-tracing data. We begin with a recently proposed method [43, 44] that makes use of embeddings of taxa in a continuous space, conversion to phylogenies by standard distance-based reconstruction methods, and optimization of model parameters by stochastic gradient ascent (SGA). We introduce a number of simplifications, extensions, and innovations to this general framework that dramatically improve its accuracy, speed, and scalability. Our methods are implemented in a freely available computer program, called VINE (Variational Inference with Node Embeddings), that supports a variety of nucleotide substitution models, a recently developed model for CRISPR-barcode editing, extensions to normalizing flows and hyperbolic geometry, and several standard prior distributions for Bayesian phylogenetics, among other features. We show that VINE is competitive with the best available phylogenetic methods in accuracy but has substantially increased speed. For the first time, we demonstrate that variational phylogenetic inference can be considerably faster than mature MCMC-based implementations such as BEAST 2 and MrBayes.

Results

Variational inference using continuous embeddings

The central problem in variational inference (VI) of phylogenies is to approximate the Bayesian posterior distribution of phylogenetic trees given a set of observed genotypes $\mathbf{X}$ using a flexible variational distribution $q(\tau, \mathbf{b}; \boldsymbol{\theta})$, where τ denotes the *topology* of a phylogenetic tree, $\mathbf{b}$ is a corresponding vector of *branch lengths*, and $\boldsymbol{\theta}$ denotes the free parameters of the variational distribution. Typically, q is fitted to the data by adjusting $\boldsymbol{\theta}$ to minimize the Kullback-Leibler (KL) divergence from the true posterior distribution, $p(\tau, \mathbf{b} \mid \mathbf{X})$, using some form of gradient-based optimization.

The starting point for our work is a clever method introduced by Mimori and Hamada [43] (see also [44]). From our perspective, Mimori and Hamada's method has three essential components: first, the tips of the tree (corresponding to the observed data) are embedded in a continuous space of d dimensions, with an induced matrix of pairwise distances $\mathbf{D}$; second, classical distance-based methods for tree reconstruction, such as the neighbor-joining method [64], are leveraged to deterministically convert the distance matrix $\mathbf{D}$ to a tree with branch lengths $(\tau, \mathbf{b})$, permitting calculation of the likelihood of the data by standard methods; and third, the free parameters of a simple variational distribution for embedded points (a multivariate normal distribution) are optimized by stochastic gradient ascent (SGA). Because the end result of the mapping from continuous space is a fully defined phylogenetic tree, the method is compatible with any of the many continuous-time Markov models of nucleotide substitution that have been introduced over the past fifty years (e.g., [65–67]) and with efficient likelihood calculations by Felsenstein's pruning algorithm [11]. As the authors noted, the resulting model can be thought of as a type of variational autoencoder [68, 69], where the decoder takes the form of the deterministic mapping from embedded taxa to trees.

In this article, we adapt and extend these core ideas in several ways. First, we introduce several simplifications to streamline and accelerate previous methods. In particular, we encode tree topologies and branch lengths together, in one unified continuous embedding, rather than separately [43]; we make use of a standard (but accelerated) neighbor-joining algorithm rather than a strictly continuous relaxation [44]; and we default to a Euclidean rather than a hyperbolic geometry for embeddings [43, 44] (although we allow a hyperbolic geometry as an option). Second, we operate in substantially higher-dimensional spaces ($d = 5$ or higher) than previous methods (typically $d = 2$ or $d = 3$) and tune our SGA algorithm to efficiently optimize the resulting large numbers of parameters, finding that high dimensionality is more important for performance than the geometry of the embedding. Third, we introduce a new algorithm for efficient backpropagation of gradients through the neighbor-joining algorithm, taking advantage of its piecewise differentiable structure. Fourth, we show that a Taylor approximation of the objective function for VI results in nearly identical results in a fraction of the time. We also introduce normalizing flows to accommodate nonlinearities in the approximate posterior distribution as well as richer parameterizations of the variational covariance matrix. In this way, we are able to obtain large improvements in efficiency, and, for the first time, outperform MCMC in speed.

With these innovations, our method for variational phylogenetic inference (VINE) can be summarized

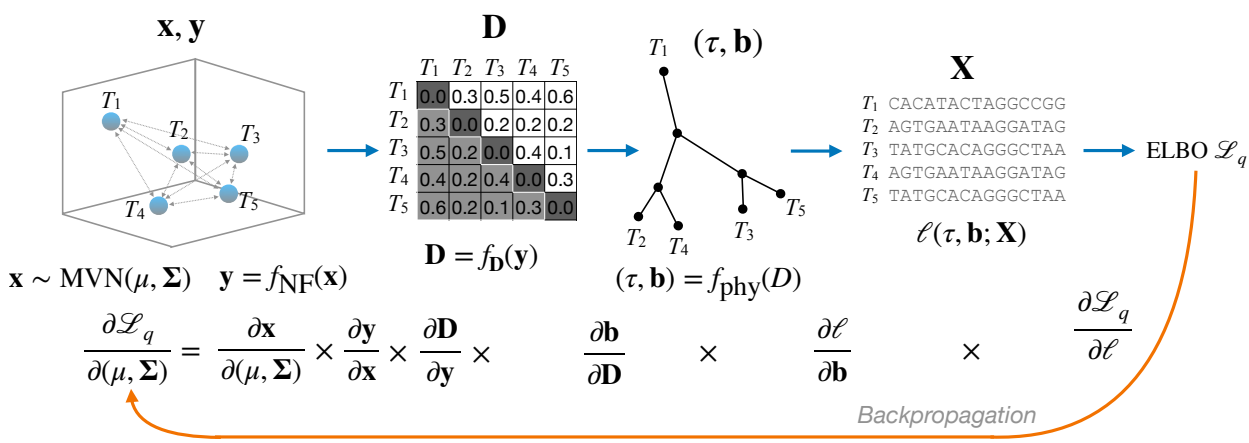

Figure 1: Conceptual diagram showing flow of information from continuous embedding of taxa to Evidence Lower Bound (ELBO; blue arrows), followed by flow in the reverse direction to the gradient via backpropagation (orange arrow). Each of n taxa is represented as a point in a d-dimensional space. These embeddings, $\mathbf{x}$, are sampled from a multivariate normal distribution, $\mathrm{MVN}(\boldsymbol{\mu}, \mathbf{\Sigma})$. Nonlinear normalizing flows are optionally applied to convert $\mathbf{x}$ to $\mathbf{y}$, then a distance matrix $\mathbf{D}$ is computed from the transformed points $\mathbf{y}$. A distance-based phylogenetic reconstruction algorithm (neighbor-joining or UPGMA) converts $\mathbf{D}$ to a tree τ with branch lengths $\mathbf{b}$, allowing calculation of a phylogenetic log likelihood $\ell(\tau, \mathbf{b}; \mathbf{X})$ from an alignment $\mathbf{X}$ in the usual manner. The ELBO is computed as an expectation of this quantity plus additional terms (see **Methods**). The gradient of this objective function with respect to the original variational parameters ($\boldsymbol{\mu}$, $\mathbf{\Sigma}$) is then computed by a series of backpropagation steps through each of the component transformations, allowing for efficient optimization by stochastic gradient ascent (SGA).

as a sequence of operations that begins with a vector of points $\mathbf{x}$ sampled in a high-dimensional embedding space, optionally transforms them by nonlinear normalizing flows, then converts them first to a distance matrix $\mathbf{D}$, and then to a phylogenetic tree $(\tau, \mathbf{b})$, whose log likelihood $\ell(\tau, \mathbf{b}; \mathbf{X})$ with respect to a sequence alignment (or CRISPR mutation matrix) $\mathbf{X}$ can be calculated by standard methods (**Fig. 1**; full details in **Methods**). The Evidence Lower Bound (ELBO)—the objective function for VI—is estimated either by averaging over several samples of $\mathbf{x}$ or using a Taylor approximation. Backpropagation is performed analytically for each step in this chain of operations, allowing for direct, efficient calculation of the gradient of the ELBO with respect to the free parameters of the sampling distribution. This step ignores discontinuities in the tree-reconstruction step, which is piecewise but not globally differentiable, but we show that it is sufficient for the purposes of stochastic optimization. After the model is fitted to the data, trees from the approximate posterior distribution can be obtained by simply sampling new points $\mathbf{x}$ and passing them through the transformations to $\mathbf{D}$ and $(\tau, \mathbf{b})$.

VINE is comparable to MCMC in model fit and offers substantially improved speed

We first evaluated VINE's performance against simulated data, where the true evolutionary history is known and model misspecification can be controlled. We simulated phylogenies under a birth-death model, followed by subsampling and rescaling, to achieve branching patterns and overall tree scales (in expected substitutions per site) roughly similar to those of real phylogenies for mammals (see **Methods** for details). We then then simulated DNA alignments by allowing nucleotide identities to evolve along the branches of these trees under the simplified but fairly realistic HKY substitution model [66], with typical values for the nucleotide composition and transition-transversion rate ratio. We simulated trees for numbers of taxa n ranging from 10 to 1000, and we generated both short alignments of $L = 300$ bp and longer ones of $L = 10,000$ bp, with the shorter alignments designed to permit faster phylogenetic reconstruction but with more statistical uncertainty, and the longer ones requiring more computational resources but allowing for more certainty. For each combination of n and L, we generated ten replicate samples to permit statistical analysis of our results.

In our benchmarking experiments, we compared VINE to three recently developed VI methods—VaiPhy [41], GeoPhy [43] and Dodonaphy [44]—as well as to two of the leading MCMC-based Bayesian phylogenetic inference packages, MrBayes [25] and BEAST 2 [27, 28]. Notably, GeoPhy and Dodonaphy are the two previous methods to use a similar embedding scheme to the one we adopted, although unlike VINE, both rely heavily on hyperbolic geometries (see **Discussion**). We also tried to include VBPI-GNN [38] in our experiments but found that it was not sufficiently fast for practical consideration (it required more than 11 hours to run one replicate from our smallest data set). Because the previously published VI methods all assume the simple Jukes-Cantor (JC) substitution model [65], we started by assuming that model for inference with all methods (but see below). As a baseline, we also reconstructed trees using a straightforward neighbor-joining implementation with no further optimization (as implemented in VINE with the `--nj-only` option).

Across all simulated data sets, we found that VINE was able to obtain reconstructed phylogenetic models that fit the data well. As would be expected for a likelihood-based method, its maximized log likelihoods were reliably higher than the log likelihoods of the true (generating) models at all data-set sizes (**Fig. 2A**). Inspection of individual trees showed that the reconstructions were generally close to the ground truth, with occasional errors that tended to correspond to difficult-to-resolve features of trees (**Fig. 3**). VINE's performance by these measures was very close to the performance of the leading MCMC-based methods, MrBayes and BEAST 2, which also generally fit the data well (**Fig. 2A**) and also produced trees closely resembling the ground truth (**Fig. 3**). The previous VI methods also fit the data reasonably well, but had slightly lower log likelihoods, with GeoPhy and VaiPhy obtaining values close to those of the true models, and Dodonaphy typically failing to match other methods (as its authors have acknowledged [44]) (**Fig. 2A**). All methods but Dodonaphy were generally able to outperform the simple neighbor-joining baseline.

Because the main goal of VI is to improve the speed of inference, we also kept track of the CPU time required for each method in each experiment. Comparisons of running times are complicated, however, by questions of how long to sample with MCMC-based methods as well as differences in multithreading

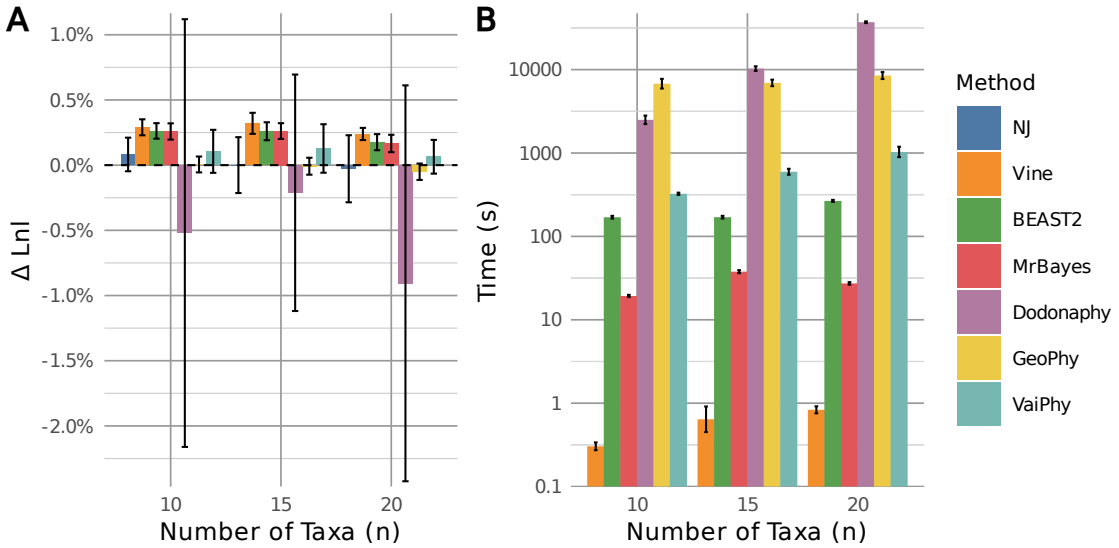

Figure 2: Performance of VI and MCMC-based methods on small simulated data sets. **(A)** Maximized log likelihood during model fitting, relative to the log likelihood of the true (generating) model (zero line). **(B)** Compute time required when running without parallelization or GPU acceleration on an HPE ProLiant DL380 Gen10 server (see **Methods**), in seconds per replicate (note log scale). Results shown are for 300 bp alignments and estimation under the Jukes-Cantor substitution model, with ten replicates per bar. Error bars represent one standard deviation. VINE (*orange*) was applied with $d = 5$, a Euclidean geometry, the Taylor approximation, and the CONST variance parameterization without normalizing flows (see **Methods**). Results are also shown for neighbor-joining (*blue*), BEAST 2 (*green*), MrBayes (*red*), Dodonaphy (*violet*), GeoPhy (*yellow*). and VaiPhy (*turquoise*).

schemes and GPU utilization. To keep our comparisons as straightforward as possible, we ran all methods using a single CPU core and we used measures based on split chains and effective sample sizes to calibrate the number of MCMC iterations in a data-set-dependent manner (see **Methods**).

Acknowledging that our results depend somewhat on these implementation choices (see **Discussion**), we nevertheless found quite stark differences in the running times required for these simulated data sets. The previously developed VI methods, in particular, were highly compute-intensive, requiring from many minutes to several hours to reconstruct trees for even our smallest alignments. The fastest of these methods, VaiPhy, took more than five minutes per replicate for our smallest (10-taxon, 300-site) simulated alignments, and seventeen minutes per replicate for similar alignments with 20-taxa. By contrast, VINE was several orders of magnitude faster than the other VI methods. In all cases with $n \leq 20$ taxa and 300 sites, VINE obtained good-quality phylogenies in less than one second, showing speedups of more than 1000-fold relative to VaiPhy and ~10,000-fold relative to Dodonaphy. The MCMC-based methods were much faster than the experimental VI methods but not as fast as VINE. MrBayes required between about 20 seconds per replicate for the 10-taxon trees and 27 seconds per replicate for the 20-taxon trees. In these experiments, BEAST 2 was slower than MrBayes by about an order of magnitude, largely owing to slower convergence

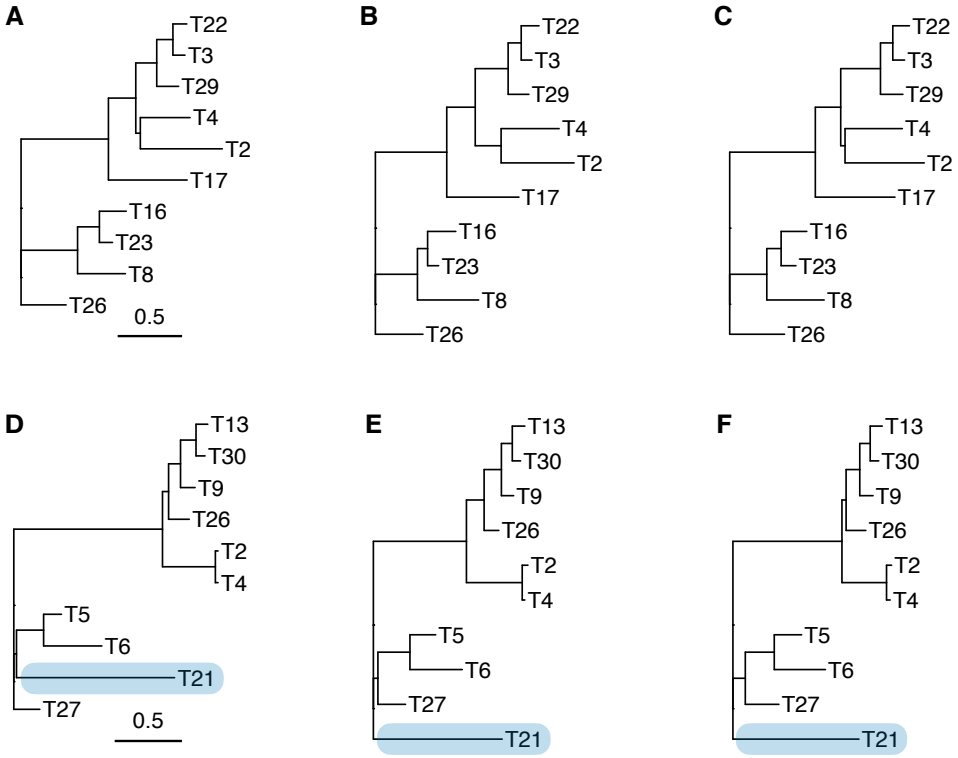

Figure 3: Reconstruction of **(A)** a simulated 10-taxon tree by **(B)** VINE and **(C)** BEAST 2 based on a 300 bp simulated alignment. A second example is shown in **(D)**, **(E)**, and **(F)**. Horizontal branch lengths are drawn to scale (see scale bars in substitutions per site). For both VINE and BEAST 2, samples from the posterior distributions are summarized by maximum-clade-credibility (MCC) trees, such that the product of posterior probabilities across all clades is maximized (constructed using TreeAnnotator [28] with a posterior-probability threshold of zero). In the first example, both reconstructed tree topologies are correct. In the second example, both reconstructions misplace taxon T21 (highlighted in *blue*), a challenging case owing to a combination of a long external branch with short internal branches. Here the HKY model was used both for data simulation and reconstruction. VINE also used the CONST variance parameterization, a Euclidean embedding, the Taylor approximation, and no normalizing flows. Taxon labels are nonconsecutive owing to subsampling in tree generation.

by our criteria (see **Methods**). Even relative to MrBayes, however, VINE was able to obtain comparable reconstructions with speeds 30–60 times greater. Because running times began to reach many hours per replicate for some of the VI methods, we did not extend these experiments beyond 20 taxa.

To consider data sets of more realistic size for applied phylogenetics, we ran VINE and the two MCMC-based methods on the larger simulated alignments, consisting of up to 1000 taxa and either 300 or 10,000 sites. In this case, we assumed the more realistic HKY model for inference, and we evaluated the quality of all reconstructed phylogenies by both likelihood-based measures of model fit and measures of discordance from the true (simulated) trees. We found again, that the maximized log likelihoods were similar across methods and generally better than those of the true model, although VINE did fall slightly short of the MCMC-based methods for the largest alignments ($n \geq 500$ taxa; (**Fig. 4A**, **Supplementary Fig. S1A**). To control for overfitting in model estimation, we also evaluated the average log likelihoods of sampled trees on held-out sequence alignments generated by the same models, and found—again as expected—that both VINE and the MCMC-based methods performed only slightly worse than the true models, with deviations of $\sim$1% (**Fig. 4B**, **Supplementary Fig. S1B**). By the Robinson-Foulds measure of topological discordance [70], VINE and the MCMC-based methods also performed fairly similarly, although VINE did show some increased discordance from the true trees, particularly at intermediate values of n (**Supplementary Fig. S2**). This difference appears to be a consequence of its reduced posterior variance, as discussed further below; VINE tends to converge on a small set of tree topologies and therefore is less effective at "hedging its bets" relative to the truth in measures of topological discordance.

The trend with running times remained similar to what we observed with smaller alignments, with VINE showing speedups of about 40–50 times relative to MrBayes, which, in turn, improved on BEAST 2 by about another 5–10 times (**Fig. 4C**, **Supplementary Fig. S3**). For the largest trees $n \geq 500$, VINE displayed somewhat less relative improvement but it was still more than an order of magnitude faster than MrBayes. VINE was again able to obtain trees for 10-taxa in less than a second, on average, increasing to about 4 seconds for 50 taxa and 13 seconds for 100 taxa. After 100 taxa, the running times began to ramp up more quickly, but VINE was still able to obtain good phylogenies for 500 taxa in about 10 minutes and for 1000 taxa in about 50 minutes. By comparison, MrBayes required running times of 20 seconds for 10 taxa up to 9.6 hours for 1000 taxa (**Fig. 4C**). With longer alignments, interestingly, the difference between MrBayes and BEAST 2 was much less pronounced, but VINE still clearly improved on both of them (**Supplementary Fig. S3**). Overall, these experiments demonstrate that VINE is the first variational phylogenetic inference method to offer both comparable model-fitting performance to MCMC-based methods and significant improvements in speed (see **Discussion**).

Performance improves with the dimensionality of the embedding space

We noticed that model fit seemed to be quite sensitive to the dimensionality d of the embedding space, more than to the use of a hyperbolic geometry rather than a Euclidean one. We therefore carried out a series of experiments where we evaluated the impact of the dimensionality d on both goodness of fit and running time, again using simulated data. We focused on data sets with $n = 25$ and $n = 50$ taxa and considered

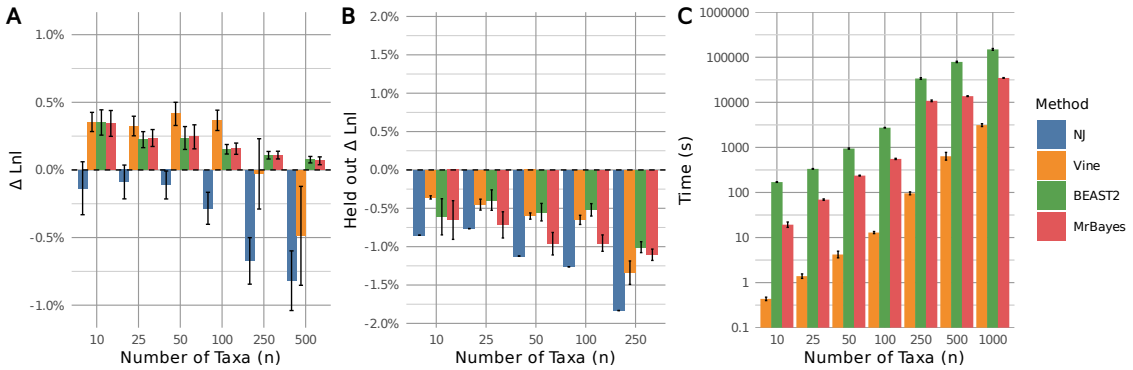

Figure 4: Performance of VINE vs. MCMC-based methods on larger simulated data sets. **(A)** Maximized log likelihood during model fitting, relative to the log likelihood of the true (generating) model (zero line). **(B)** Average log likelihood across posterior samples for held-out data, also relative to the true model. **(C)** Compute time required when running without parallelization or GPU acceleration on an HPE ProLiant DL380 Gen10 server (see **Methods**), in seconds per replicate (note log scale). Results shown are for 300 bp alignments and estimation under the HKY substitution model, with ten replicates per bar (see also **Supplementary Figs. S1 & S3**). Error bars represent one standard deviation. VINE (*orange*) was applied with $d = 6$, a Euclidean geometry, the Taylor approximation, and the CONST variance parameterization without normalizing flows (see **Methods**). Results are also shown for neighbor-joining (*blue*), BEAST 2 (*green*), and MrBayes (*red*). Results for $n = 1000$ are omitted in (A) and results for $n = 500$ and $n = 1000$ are omitted in (B) owing to failures of some runs; work is in progress to complete them.

values of d between 2 and 8 under both the Euclidean and hyperbolic embedding schemes.

We found, under the Euclidean embedding, that the maximized log likelihood was poor with $d = 2$ and $d = 3$ (e.g., ∼200 units lower than the value reported by BEAST for $n = 25$ and ∼800 units lower for $n = 50$ taxa; see **Fig. 5**). As d increased, however, the log likelihood rapidly improved from $d = 2$ to $d = 4$, tapering off at about $d = 4$. We expected that this improvement in model fit would come with a time penalty, since the number of free parameters in the model is roughly equal to nd (see **Methods**). Interestingly, however, we found that the running time slightly *decreased* with d rather than increasing, owing to more efficient convergence of the optimization algorithm. For example, with $n = 25$ taxa, the average time to convergence decreased from about 2 seconds for $d = 2$ to about 1.4 seconds at $d = 4$ and then reached a minimum of 1.2 seconds by $d = 6$. Similarly, with $n = 50$ taxa, the time to convergence was cut nearly in half from $d = 2$ to $d = 6$.

Under the hyperbolic embedding scheme, the log likelihood was similarly sensitive to the embedding dimension, but the running time behaved in a less predictable manner (**Supplementary Fig. S4**). In addition, the running time was generally greater, and with higher variance across replicates, owing to delays in convergence of the optimizer.

Overall we found that, once SGA is suitably tuned (see **Methods**), it is capable of optimizing hundreds

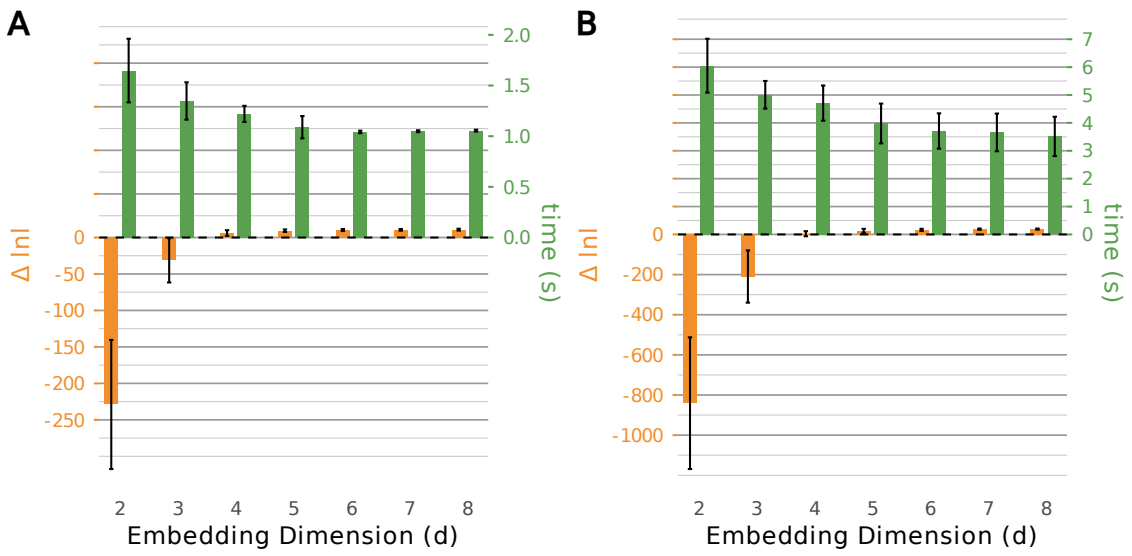

Figure 5: Performance improvement of VINE with increasing dimensionality d of the embedding space. Shown are running times per replicate (*green*) and deviations of the maximized log likelihood from that of the true model (*orange*) for **(A)** $n = 25$ taxa and **(B)** $n = 50$ taxa. Results are for alignments of 300 bp and estimation under the HKY model and CONST variance parameterization, with a Euclidean geometry and the Taylor approximation and without normalizing flows (see also **Supplementary Fig. S4**). The number of free parameters under this parameterization is $nd + 2$, so it ranges from 52–202 in (A) and 102–402 in (B).

of parameters in a highly efficient manner, and there seems to be little cost, and substantial benefit, to operating at values of $d = 6$ or higher. At the same time, we found no reason to prefer the more complex hyperbolic geometry over a simple Euclidean scheme for embedding. At least in our hands, any advantages in flexibility from the hyperbolic geometry are offset by difficulties in fitting the model (see **Discussion**).

VINE shows top performance in cell-lineage phylogeny inference

As noted, a major area of interest in phylogeny inference today is the reconstruction of cell lineage trees from CRISPR-based barcoding data This problem is closely related to standard phylogeny inference, but certain unusual features of the CRISPR-based editing process—which tends to produce insertions and deletions (indels) rather than point mutations, and which can be "silenced" in a site-specific manner if target sites are disrupted—have led to the development of customized mutation models [53, 54, 58, 61, 71]. It has been shown that incorporation of these models into likelihood-based phylogenetic inference methods can lead to substantial improvements in reconstructed trees [58].

We therefore extended VINE to support the mutation model recently used in the LAML (Lineage Analysis via Maximum Likelihood) program [58] (see also [61])—a continuous-time Markov model that captures

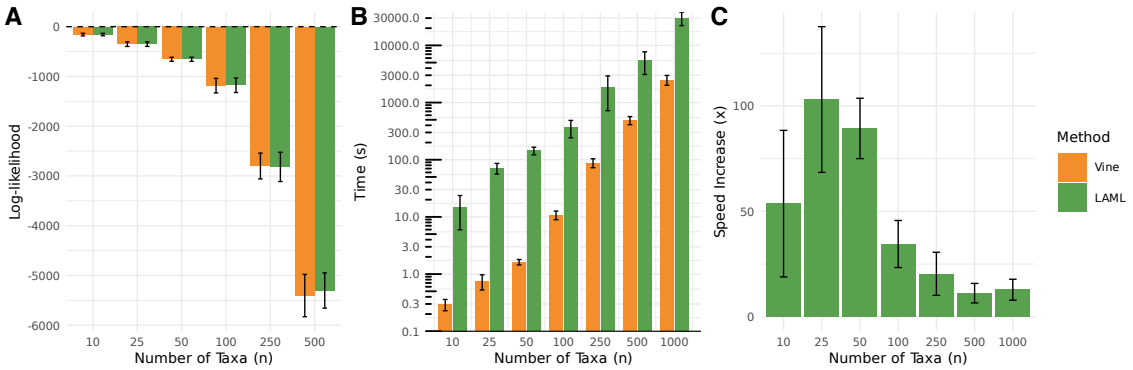

Figure 6: Comparison of VINE and LAML [58] on simulated CRISPR-barcoding data, for various numbers of taxa n. **(A)** Maximized log likelihood during model fitting. **(B)** Compute time per replicate. **(C)** Speed increase of VINE relative to LAML. Results are for 10 simulated data sets for each value of n with 30 barcode sites and editing parameters based on real data [73] (see **Methods**). Error bars represent one standard deviation.

the irreversibility of CRISPR-mediated indels, site-specific mutation rates, barcode silencing, and missing data (available via the `-i CRISPR` option in VINE; see **Methods**). We ensured that VINE produces time-resolved ultrametric trees (with all tips equidistant from the root) similar to those inferred by LAML, by using the UPGMA (Unweighted Pair Group Method with Arithmetic Mean) algorithm [72] in place of neighbor-joining in CRISPR mode (see **Methods**). We then benchmarked VINE against LAML on simulated CRISPR barcode mutation matrices. Other tree-reconstruction programs are available for this problem, including Cassiopeia [53], TiDeTree [61], and Startle [55], but LAML appears to be the best-performing available method by a considerable margin at present [58], so we focused on it for our comparisons.

As with our DNA alignments, we began by simulating trees and CRISPR barcode mutation matrices, taking advantage of the barcode-editing simulation tools in the Cassiopeia package and approximately matching mutation patterns observed in real data (see **Methods**). We assumed a barcode cassette with ten arrays of three target sites for a total of 30 editing sites per cell, and we generated 10 simulation replicates for trees with 10, 25, 50, 100, and 250 taxa. We then applied VINE and LAML to each simulted alignment and recorded measures of model fit and tree discordance. Before comparing the reported likelihoods, we verified that VINE returned exactly the same values (up to numerical precision) for the same data and parameter values.

We found that VINE was able to obtain trees of similar log likelihood (lnl) to those reported by LAML across all simulated data sets (**Fig. 6A** and **Supplementary Fig. S5**). Indeed, the maximized log likelihoods were nearly identical through data sets with $n = 50$ taxa (average differences of $\sim$1 unit of lnl). At $n = 100$, VINE was able to obtain slightly better models (by $\sim$6 lnl on average), and at $n = 250$ and $n = 500$ LAML was slightly better (by $\sim$17 and $\sim$101 lnl, respectively), but these average differences were small relative to the standard deviation across replicates and were not statistically significant. The topological distances of the inferred trees from the true ones were also similar under both methods (**Supplementary Fig. S6**).

As with the DNA models, VINE was able to achieve these high levels of accuracy at considerably greater speed than LAML (**Fig. 6B**). Running times ranged from less than a second for $n = 10$ to about 40 minutes for $n = 1000$. VINE was roughly 50–100 times faster than LAML for $n \leq 50$ taxa (**Fig. 6C**). As the number of taxa increased, this speed advantage declined somewhat but VINE was still 11–12 times faster than LAML for $n \geq 500$. Notably, the other likelihood-based methods available for this problem, such as TiDeTree [61], are considerably slower than LAML, so to our knowledge, VINE is now the fastest such method available. Overall, VINE is able to fit CRISPR barcoding data at least as well as competing methods, at considerably greater speed, and it does so by approximating a full posterior distribution rather than a single point estimate—a feature that can be particularly important in cell-lineage reconstruction, where there is often a great deal of uncertainty in tree inference (see **Discussion**).

Capturing the full posterior variance remains challenging

We observed in our experiments with simulated data that, while VINE achieved high likelihoods, the variance of its approximate posterior distributions tended to collapse during optimization. For example, under our initial version of the model (CONST), which assumed a simple diagonal covariance structure with a single free parameter (with $\mathbf{\Sigma} = e^{\eta}\mathbf{I}$), the η parameter was driven toward strongly negative values and approached the floor we had set for it (such that $e^{\eta} = 1 \times 10^{-3}$; see **Supplementary Fig. S7**).

We therefore experimented with various alternative parameterizations of the covariance matrix, with the goal of capturing more of the structure of the true posterior. We introduced a fully parameterized diagonal covariance matrix (DIAG), a covariance matrix proportional to a double-centered version of the initial distance matrix (DIST), and a general low-rank parameterization of the covariance matrix (LOWR) (see **Methods** for details). We found even under these richer parameterizations, however, that the covariance still tended to collapse. This behavior is known to occur in VI when the approximate posterior distribution has insufficient flexibility to capture the structure of the true posterior, and the KL divergence is maximized by shrinking the approximate distribution to a restricted region at a single mode [33, 74].

To counter this problem, we introduced two additional extensions to our model. First, we regularized the variance in a manner dependent on the choice of parameterization. For example, in the CONST and DIST parameterizations, we applied an ℓ_2 penalty to the free parameter η, pushing it toward zero (see **Supplementary Fig. S7**); and in the DIAG parameterization, we applied a similar ℓ_2 penalty to all free log-variance terms (see **Methods**). Second, as outlined above, we introduced optional normalizing flows to accommodate nonlinearities between the MVN-distributed points $\mathbf{x}$ and the embedding $\mathbf{y} = f_{\text{NF}}(\mathbf{x})$ from which the distance matrix $\mathbf{D}$ is computed. We allowed for two types of normalizing flows: a *radial flow*, which allows for radial contraction or expansion of points around a designated center (itself a free parameter estimated from the data); and a *planar flow*, which moves points based on their location relative to a hyperplane that is estimated from the data (see **Methods** for complete details). Together with our four parameterizations, these two strategies gave us a variety of means for tuning the representation of the approximate posterior to better reflect the true distribution.

We evaluated the effectiveness of these strategies by carrying out experiments with simulated DNA alignments of various sizes, across a grid of combinations of our four parameterizations, strengths of regularization penalties, the two normalizing flows, and with the Euclidean vs. the hyperbolic geometries. In each case, we allowed VINE to estimate an approximate posterior distribution from the same ten simulated alignments, and we considered not only the quality of the model fit, but also the variance of the posterior. In assessing these posterior distributions, we focused on 95% credible intervals (CIs) for all $\binom{n}{2}$ pairwise distances between taxa along the reconstructed trees, and kept track of the fraction of estimated 95% CIs that contained the true value used in simulation. For comparison, we computed the same quantity based on BEAST 2.

We found, indeed, that the posterior variance was poorly characterized with our simplest models (CONST parameterization, no regularization, no normalizing flows, and Euclidean geometry). In this case, while the likeihoods were excellent (as shown in **Fig. 2**), the 95% CIs were extremely narrow and fewer than 25% of true values were contained within them (**Fig. 7**). These intervals were even narrower for smallest trees ($n = 10$), where only about 6% of true values were contained within them. By contrast, BEAST 2 performed exceptionally well by this measure, with 92–97% inclusion of true values across all values of n. The extensions of our baseline model, however, did improve VINE's ability to capture the posterior variance, to varying degrees. The best model used the DIST parameterization, a moderate variance regularization, both normalizing flows, and the Euclidean geometry. In this case, the 95% CI inclusion reached about 70% for $n = 10$ taxa and declined to about 45% for $n = 100$ (**Fig. 7**). Thus, it appears to be possible to adapt our VI methods to capture more of the true posterior variance but even our best models still fall short of MCMC-based methods in this respect (see **Discussion**).

Applications to real data

To demonstrate the applicability of VINE to real data, we applied it to two data sets, one consisting of multi-species DNA alignments for use in traditional phylogenetic reconstruction, and one consisting of CRISPR-based mutation matrices for use in cell-lineage tree reconstruction. For the DNA alignments, we used a collection of eight alignments originally assembled by Lakner et al. [75], which contain various types of nucleic-acid sequence data (DNA, rRNA, rDNA, and mtDNA) for between 27 and 59 taxa (median 42) and range in length from 378 to 2520 (median 1736) sites (**Supplementary Table S1**). These alignments are fairly typical of modest-sized data sets frequently analyzed in applied phylogenetics and have been widely used in recent benchmarking studies (e.g., [43, 44, 76]). We followed other studies in focusing on the first eight of the original twelve alignments. As in earlier section, we compared VINE with BEAST 2 and MrBayes using the HKY substitution model. We omitted the other VI methods from this comparison owing to their long running times.

For the CRISPR-based mutation matrices, we used a data set based on a lung-cancer xenograft mouse model, which contains nearly 100 clonal populations (CPs) ranging in size from >11,000 to ∼30 cells [73]. Larger data sets now exist, but this one contains particularly rich mutational data for a broad range of CPs, and it has been widely analyzed. For our purposes, we excluded the largest 25 CPs (CPs <30 in their

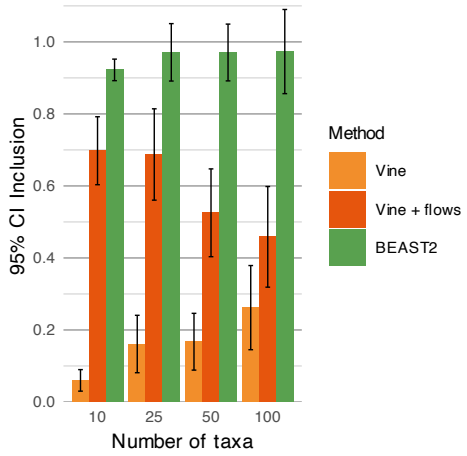

Figure 7: Accuracy of posterior distributions, as measured by the fraction of all pairwise distances between taxa that fall within the estimated 95% credible interval (95% CI Inclusion). Results are shown for simulated trees of $n \in \{10, 25, 50, 100\}$ taxa for the baseline version of VINE (VINE), the best version of VINE (VINE + flows), and BEAST 2. Bars represent averages over ten replicates. The baseline version of VINE uses the CONST variance parameterization, no variance regularization, and no normalizing flows. The best version uses the DIST parameterization, a variance regularization multiplier of three (`-var-reg 3`), and both the radial and planar flows. A Euclidean geometry and Monte Carlo estimation of the ELBO were used in both cases.

nomenclature), most of which take many hours to analyze, and focused on the remaining 58, which ranged in size from 26 to 163 cells (median 50 cells). As above, we compared VINE with LAML on these data, assuming a uniform prior distribution for mutation rates and allowing for a free silencing-rate parameter (see **Methods**).

As expected from our simulation experiments, all methods were similar in model fit, as evaluated by the maximized log likelihoods (lnl) for the training data. On the DNA alignments, VINE achieved lnl values within 0.5% of BEAST 2 and MrBayes on average, and all methods reliably outperformed the baseline neighbor-joining algorithm (**Supplementary Fig. S8A**). We observed lnl values roughly similar to those reported in recent benchmarking studies but somewhat higher in all cases, owing to our use of the HKY model. The reconstructed trees were generally quite similar in appearance under all reconstruction methods.On the CRISPR data, VINE obtained slightly higher lnl values than LAML on average (by 24.0 units of lnl, or 1.3%) with some variability across clonal populations (**Supplementary Fig. S8B**). As with the simulated data, we found that VINE and LAML behaved similarly for the smaller trees but VINE often outperformed LAML on the larger ones. In most cases, the reconstructed trees were similar but in a few cases they were quite different.

As with the simulated data, VINE was able to fit these phylogenetic models with substantially improved running times. On the eight DNA alignments, it was about nine times faster than MrBayes and 17 times faster than BEAST 2, on average (**Fig. S8A**). Running times for VINE ranged from 6–36 seconds, in com-

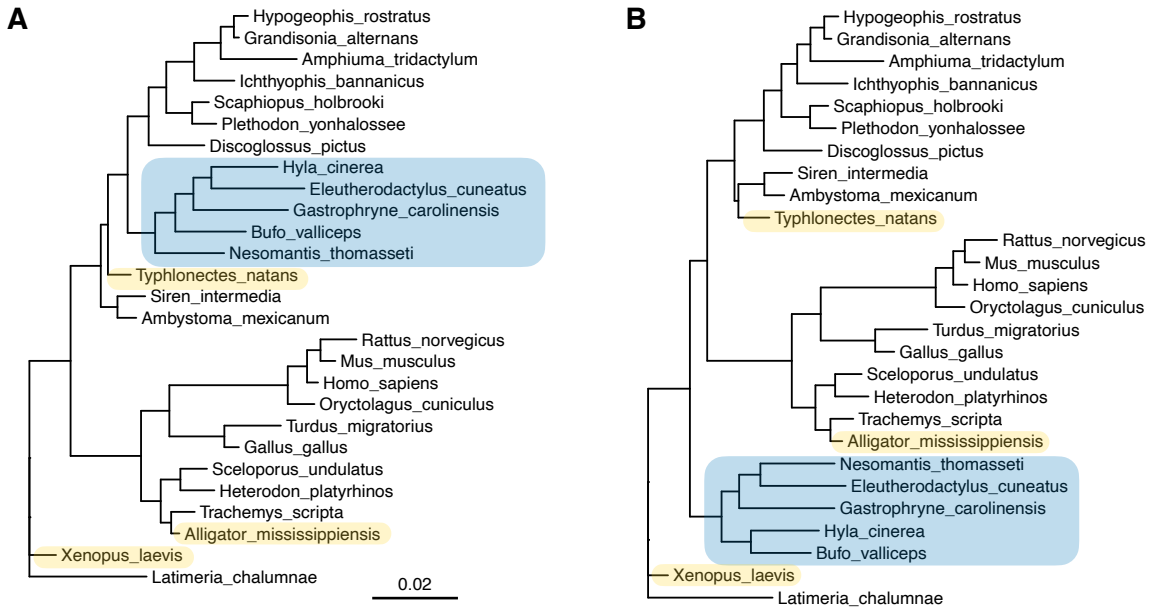

Figure 8: Examples of reconstructed phylogenies for real alignments of nucleic acid sequences by **(A)** VINE and **(B)** BEAST 2 under the HKY model. Results are for data set 1 (ds1) from ref. [75] (details in **Supplementary Table S1**). As in **Fig. 3**, samples from the posterior distributions are summarized using maximum-clade-credibility (MCC) trees with horizontal branch lengths drawn to scale (scale bar). Notice that the two trees are generally in agreement, with the exception of the placement and precise arrangement of the neobatrachian frogs (highlighted in *blue*), which are incorrect in both cases (they should group with *Discoglossus pictus* and *Xenopus laevis*). In addition, three other taxa (*yellow*) are placed incorrectly in both trees relative to the accepted phylogeny, not surprisingly for a tree based only on highly constrained 18s rRNA sequences. In this case, VINE was run with the CONST variance parameterization, a Euclidean embedding, the Taylor approximation and no normalizing flows.

parison to 1–4 minutes for MrBayes and 2–5 minutes for BEAST 2. With these longer alignments, VINE obtains an additional boost in speed by subsampling sites in the alignment early in the optimization process (see **Methods**). On the CRISPR data sets VINE was faster than LAML by orders of magnitude, across CPs, and by 235 times on average (**Fig. S8B**). The average running time for these CPs was about 6 seconds for VINE in comparison to 25.3 minutes for LAML. The largest clones took about two hours with LAML but less than one minute with VINE. These experiments did not take advantage of LAML's multithreading capabilities which would undoubtedly improve running times considerably, although similar improvements are anticipated with VINE (see **Discussion**).

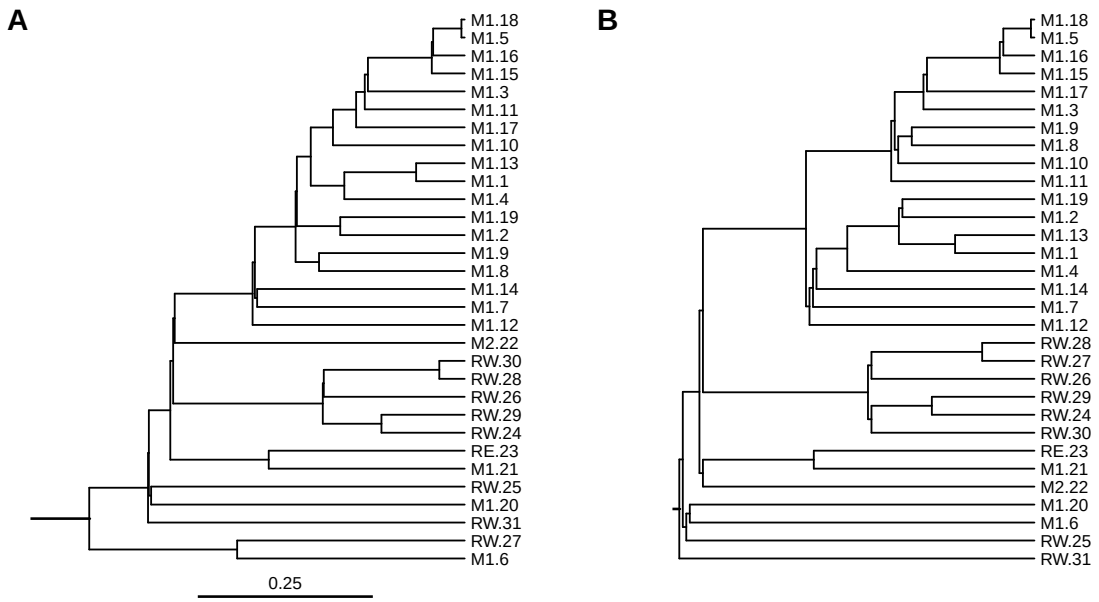

Figure 9: Examples of reconstructed phylogenies for CRISPR-based mutation matrices by **(A)** VINE and **(B)** LAML. Shown are trees for clonal population (CP) 92 from ref. [73], representing the MCC summary for VINE and the ML tree reported by LAML. Taxa have been numbered arbitrarily but prefixes from the original data set have been retained, because they indicate tissues of origin in the lung-cancer xenograft mouse model (*RW* & *RE*: right lung; *M1* & *M2*: mediastinal lymph tissue). Notice that the inferred trees are ultrametric (with tips aligned) and are assumed to have leading branches, on which mutations shared by all cells can accumulate. Notice also that the reconstructed trees have several differences, owing in part to sparse mutation data. Despite their differences, both programs reported nearly identical maximum log likelihoods (–583.95 and –583.62, respectively).

Discussion

Bayesian methods are now widely used in phylogenetic inference, but as data sets steadily grow in size and complexity, the computational cost of MCMC sampling becomes increasingly burdensome. In this article, we have introduced a new computational method, called VINE, that demonstrates for the first time that variational phylogenetic inference can be competitive with state-of-the-art MCMC methods in terms of model fit while offering substantially shorter running times. In addition to several common DNA substitution models, VINE also supports inference of cell-lineage phylogenies based on CRISPR barcode-editing mutation matrices. On this task, VINE fits the data as well as the best available method, LAML, but is significantly faster across a broad range of simulated and real data sets.

Development of VI for Bayesian phylogenetic inference has been an active area of recent methodological development in phylogenetics, with several innovative new programs appearing over the last few years [34, 37–44]. In our design of VINE, we drew heavily from this emerging literature, focusing in particular on the use of a continous embedding of taxa, distance-based phylogenetic reconstruction, and stochastic

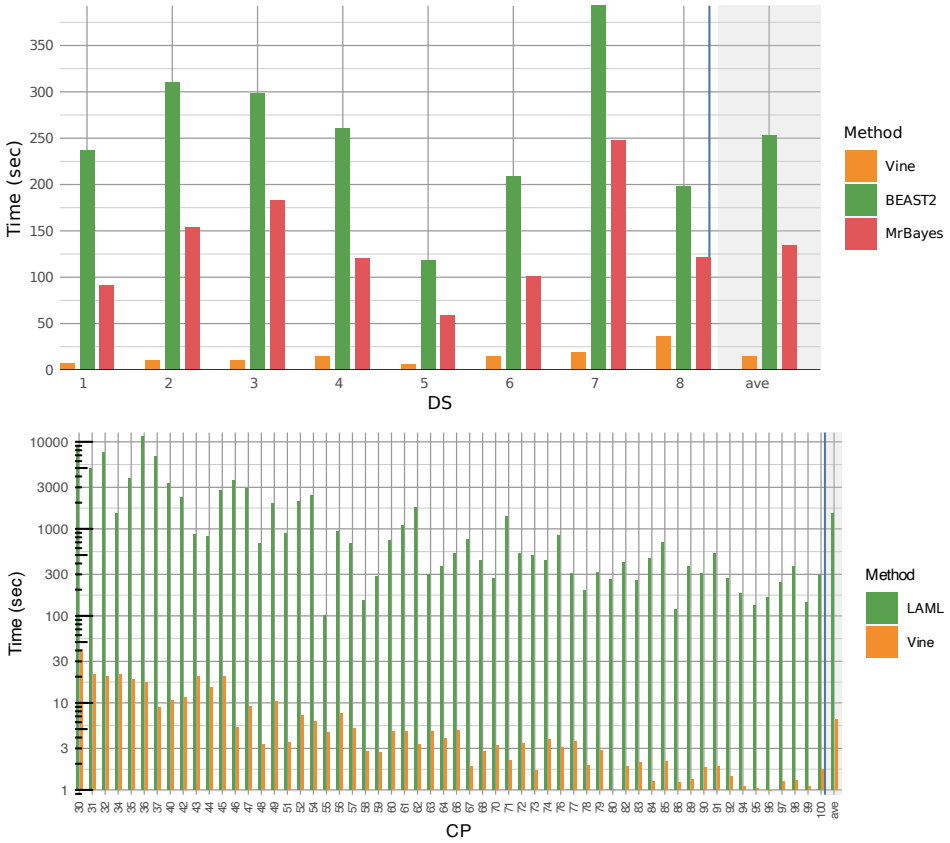

Figure 10: Compute time required for VINE and other methods to estimate phylogenetic trees from real data. **(A)** Comparison of VINE and MCMC-based methods BEAST 2 and MrBayes for eight multispecies alignments assembled in ref. [75] (details in **Supplementary Table S1**) **(B)** Comparison of VINE and LAML for 57 CRISPR-based mutation matrices from ref. [73]. Results are shown in seconds per input data set, with the average values on the right. Notice the log scale in (B).

gradient ascent for optimization of the evidence lower bound (ELBO) [43, 44]. This approach has several important advantages over earlier attempts at phylogenetic VI, by avoiding the need to enumerate discrete topological features [34] and aligning naturally with powerful tools for stochastic optimization. It effectively leverages the long history of distance-based methods for phylogenetic reconstruction (e.g., [64, 72]), but marries them with classical likelihood-based strategies within the new framework of variational inference. In our experiments, however, we found that previously published phylogenetic VI methods are not yet viable alternatives to MCMC for applied phylogenetics, and typically require orders of magnitude longer run times on data sets of even modest size. By introducing several new algorithmic innovations and optimizing our code, we were able to improve speeds by factors of hundreds to thousands, making it practical to use VI in applied phylogenetics.

One key difference of VINE from similar VI programs such as GeoPhy [43] and Dodonaphy [44] is that, by default, it makes use of a higher dimensional embedding space, typically with $d \geq 5$ rather than $d = 2$ or $d = 3$. To our surprise, we found that stochastic optimization was more, rather than less, effective at

these higher dimensions, despite the larger number of free parameters. In this way, our approach is less like conventional phylogenetics and more like modern strategies for training deep neural networks, where investigators typically rely on the remarkable effectivess of SGA to optimize models that are intentionally over-parameterized. Notably, however, the use of a hyperbolic embedding geometry offered no advantage in our hands, even after considerable effort in tuning the dimensionality, curvature, and scale of the space—despite considerable enthusiasm in the recent literature about the potential value of non-Euclidean embeddings in phylogenetics [43, 44, 47, 76].

Another key innovation in VINE is its strategy for differentiation through the neighbor-joining or UPGMA algorithms. Because gradients are approximate anyway in the setting of SGA, we chose not to work with formally differentiable relaxations of these algorithms (see [44]) and instead derived our own recursive procedure for propagating derivatives through the standard algorithms conditional on a choice of nearest neighbors (see **Methods**). Notably, once the neighbors are fixed, both algorithms can be shown to perform linear transformations on a vector of pairwise distances. This strategy allowed us to keep the procedures for backpropagation lightweight and simple, and avoid the heavy computational machinery of automatic differentiation. Our approach effectively factors out the tree topology from gradient calculations, but as we show, the fundamental stochasticity of the optimization algorithm is nonetheless sufficient to ensure that the space of topologies is explored. In this way, also, our optimization strategy is more akin to deep learning than to traditional phylogenetics.

In our benchmarking experiments, we compared the running times of VINE with those of two of the most widely used packages for MCMC-based Bayesian phylogenetics, BEAST 2 and MrBayes. There are many challenges in ensuring that such a comparison is fair, however, including the perennial issue of how to assess convergence of the Markov chain (e.g., [77]), as well as more technical concerns such as whether or not to run multiple coupled chains, how to tune proposal distributions, and how to manage parallelization, either through multithreading or the use of Graphics Processing Units. (Both BEAST 2 and MrBayes have the option of using the BEAGLE library for GPU acceleration [78], but it was not included in our experiments.) We sought to place the MCMC and VI methods on an even field by using rigorous but not onerous convergence criteria and standard models with default proposals, and running all programs on a single CPU core. Undoubtedly, other investigators might make different choices. Still, even if alternative benchmarking strategies were to diminish the speed advantages of VINE, the VI paradigm has the advantage of requiring less tuning of sampling strategies and monitoring of convergence. In addition, once the variational model has been fitted to the data, any number of independent samples can be drawn from the approximate posterior at low computational cost.

At the same time, we observed a persistent tendency in VINE to underestimate the variance of the posterior distribution—a known problem in VI when the approximating distribution does not have sufficient flexibility to accommodate the structure of the true posterior [33, 74]. We attempted to address this problem with several modeling extensions, including richer parameterizations of the covariance matrix, regularization of covariance parameters, and the use of radial and planar normalizing flows to accommodate nonlinearities in the relationship between the multivariate normal sampling distribution and the space of phylogenetic trees. These extensions improved the posterior variance somewhat, but it remained underdispersed. By

contrast, current MCMC-based methods appear to explore the posterior distribution effectively, at least at the scales we considered. At present, they should be preferred to VI in applications that require a complete representation of posterior uncertainty. It might be possible to improve the quality of the approximate posterior further by using other techniques such as mixture models (e.g., [42]), but there will likely still be limitations in the approximation of phylogenetic posterior distributions, which are notoriously complex.

In our comparisons of DNA alignments, we chose to focus on Bayesian methods to the exclusion of methods based on maximum likelihood (ML). It bears mentioning, however, that programs such as RAxML [17] and IQ-TREE [18, 19] have recently made it possible to perform ML phylogenetic inference at scale with astonishing speeds. These programs remain substantially faster than both VINE and MCMC-based methods and for the forseeable future will remain better choices for trees with many thousands of taxa. Still, these ML methods generally report a single tree with branch lengths—a point estimate of the phylogeny—rather than attempting to characterize the full posterior distribution, and in many applications it will be worthwhile to expend additional CPU cycles to characterize the uncertainty of the phylogeny.

The full scaling potential of phylogenetic VI based on continuous embeddings of taxa remains unclear. Several steps in VINE are straightforward to parallelize, including the likelihood calculations and recursive updates for backpropagation, and GPU acceleration (perhaps via BEAGLE) could also be added. It might also be possible to accelerate the program further through special-purpose optimizations, such as more aggressive subsampling of alignment columns early in SGA. At the same time, the use of an explicit distance matrix for taxa and distance-based phylogeny reconstruction on each iteration of the algorithm imposes a fundamental lower bound of $O(n^2)$ on its computational complexity. We speculate that, with extensive optimizations to the current design, VINE could perhaps be scaled up one to two orders of magnitude more, allowing practical application to alignments with 10,000 to 100,000 taxa. To scale the method further would likely require abandoning an explicit representation of each taxon and pairwise distance, for example, by hierarchically applying the algorithm to groups of taxa and then merging subtrees, in a divide-and-conquer approximation. For extremely large data sets, it might be that the traditional Bayesian paradigm (which, after all, still requires explicit likelihood calculations and sampling of trees) will best be abandoned (see, e.g., [79]), or at least applied only to subsamples of taxa. In any case, the work presented here points a path forward to increasing the range of data sets to which approximate Bayesian methods can be applied.

Methods

The core variational inference problem

The core problem is to approximate the posterior distribution of phylogenetic trees given a set of observed genotypes using an approximate variational distribution $q(\tau, \mathbf{b})$, where τ is a rooted binary tree with n tips and $\mathbf{b} \in \mathbb{R}^{2n-2}$ is a corresponding vector of nonnegative branch lengths. We denote the data by an $n \times L$ matrix $\mathbf{X}$, which may be either a standard multiple alignment of DNA sequences or a mutation matrix representing CRISPR-edited barcodes (with L observed sites in either case).

In the standard manner for variational inference (reviewed in [33]), we fit the model by minimizing the Kullback-Leibler (KL) divergence between $q(\tau, \mathbf{b})$ and the true posterior distribution, $p(\tau, \mathbf{b} \mid \mathbf{X})$. Here, we follow the typical formulation for a variational autoencoder [69], for reasons that will become clear below. The KL divergence can be expressed as,

$$\begin{aligned}
\mathrm{KL}(q(\tau, \mathbf{b}) \,\|\, p(\tau, \mathbf{b} \mid \mathbf{X})) &= \mathbb{E}_q \left[\log \frac{q(\tau, \mathbf{b})}{p(\tau, \mathbf{b} \mid \mathbf{X})} \right] \\
&= \mathbb{E}_q \left[\log \frac{q(\tau, \mathbf{b})}{p(\mathbf{X} \mid \tau, \mathbf{b})\, p(\tau, \mathbf{b})} + \log p(\mathbf{X}) \right] \qquad \text{(Bayes' rule)} \\
&= \mathbb{E}_q \left[\log q(\tau, \mathbf{b}) - \log p(\mathbf{X} \mid \tau, \mathbf{b}) - \log p(\tau, \mathbf{b}) \right] + \log p(\mathbf{X}) \\
&= -\mathbb{E}_q \left[\log p(\mathbf{X} \mid \tau, \mathbf{b}) \right] + \mathrm{KL}(q(\tau, \mathbf{b}) \,\|\, p(\tau, \mathbf{b})) + \log p(\mathbf{X}) \\
&= \log p(\mathbf{X}) - \mathcal{L}_q, \qquad (1)
\end{aligned}$$

where the expectations are evaluated with respect to $q(\tau, \mathbf{b})$ and $\mathcal{L}_q$ is defined as:

$$\begin{aligned}
\mathcal{L}_q &= \mathbb{E}_q \left[\log p(\mathbf{X} \mid \tau, \mathbf{b}) \right] - \mathrm{KL}(q(\tau, \mathbf{b}) \,\|\, p(\tau, \mathbf{b})) \\
&= \mathbb{E}_q \left[\ell(\tau, \mathbf{b}; \mathbf{X}) \right] - \mathrm{KL}(q(\tau, \mathbf{b}) \,\|\, p(\tau, \mathbf{b})), \qquad (2)
\end{aligned}$$

where in the second step we introduce the notation $\ell(\tau, \mathbf{b}; \mathbf{X}) = \log p(\mathbf{X} \mid \tau, \mathbf{b})$ for the standard phylogenetic log likelihood, that is, the log probability of the genotype data given the tree.

Because the KL divergence must be nonnegative, $\mathcal{L}_q$ is a strict lower bound on the marginal log likelihood, $\log p(\mathbf{X})$, but it can be evaluated using the more tractable variational density $q(\tau, \mathbf{b})$ and conditional log likelihood, $\ell(\tau, \mathbf{b}; \mathbf{X})$. This key quantity is known in the variational inference literature as the Evidence Lower Bound (ELBO). By choosing the free parameters of q to maximize the ELBO $\mathcal{L}_q$, the KL divergence is forced to shrink toward zero, making $q(\tau, \mathbf{b})$ approximate the true posterior distribution as closely as possible given its functional form. Notice, in addition, that the ELBO $\mathcal{L}_q$ (and its gradient with respect to the parameters of q) can readily be approximated by sampling from q, making it amenable to optimization by stochastic gradient ascent (see below).

Continuous embeddings and differentiable transformations

Our general strategy for addressing the variational inference problem—as outlined in **Fig. 1**—is to sample embeddings $\mathbf{x} \in \mathbb{R}^{nd}$ of n taxa in d-dimensional space from a multivariate normal (MVN) distribution $q = \mathrm{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with $\boldsymbol{\mu} \in \mathbb{R}^{nd}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{nd \times nd}$; to optionally convert $\mathbf{x}$ to $\mathbf{y}$ using normalizing flows; to compute an induced distance matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ from $\mathbf{y}$; to obtain a rooted phylogenetic tree τ with branch lengths $\mathbf{b} \in \mathbb{R}^{2n-2}$ from $\mathbf{D}$ using a deterministic, distance-based tree reconstruction algorithm; to compute the phylogenetic log likelihood $\ell(\tau, \mathbf{b}; \mathbf{X})$ based on that tree and the specified mutation model; and finally, to estimate the ELBO using an expectation of the phylogenetic log likelihood. We then compute the gradient of the ELBO with respect to the free parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ by backpropagation through all steps of this transformation, and optimize the parameters by SGA. Notably, all nd components of $\boldsymbol{\mu}$ are maintained as free parameters in the optimization problem, but various reduced parameterizations can be assumed for $\boldsymbol{\Sigma}$ (see below). Here we describe each of these steps in more detail.

First, to allow for pathwise differentiation with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we use the standard technique of reparameterizing the MVN variational distribution (the "reparameterization trick"). We introduce a standard multivariate normal random variate $\mathbf{z} \in \mathbb{R}^{nd}$, with $\mathbf{z} \sim \mathrm{MVN}(\mathbf{0}, \mathbf{I})$, and redefine $\mathbf{x}$ as,

$$\mathbf{x} \;=\; f_{\mathrm{MVN}}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \;=\; \boldsymbol{\mu} + \mathbf{L}\mathbf{z},$$

where $\mathbf{L}$ is a (lower-triangular) Cholesky factor such that $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top$. In this way, the expressiveness of the MVN is maintained but the randomness in samples from q now derives from a parameter-free density, and all subsequent transformations are deterministic and differentiable. As shown in the next section, backpropagation can be carried out through these transformations, from the ELBO to the tree $(\tau, \mathbf{b})$ all the way back to the variational parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

Following the linear map, $\mathbf{x} = f_{\mathrm{MVN}}(\mathbf{z})$, a series of differentiable normalizing flows can optionally be applied to accommodate nonlinear distortions to the MVN without altering the dimensionality of $\mathbf{x}$ (detailed below). We can summarize the cumulative effect of these normalizing flows as,

$$\mathbf{y} \;=\; f_{\mathrm{NF}}(\mathbf{x}; \theta_{\mathrm{NF}}) \in \mathbb{R}^{nd},$$

where θ_{NF} denotes a set of relevant free parameters. (If the normalizing flows are omitted, f_{NF} can simply be assumed to be the identity function, so that $\mathbf{y} = \mathbf{x}$.)

The third transformation produces a symmetric matrix of pairwise distances $\mathbf{D} \in \mathbb{R}^{n \times n}$ from $\mathbf{y}$:

$$\mathbf{D} \;=\; f_{\mathrm{D}}(\mathbf{y}),$$

where f_{D} is such that $D_{ij} = \mathscr{d}(\mathbf{y}_i, \mathbf{y}_j)$ where $\mathscr{d}$ is a suitable distance function between points in the embedding space. As described below, this distance function is based either on a Euclidean or a hyperbolic geometry.

Finally, the fourth transformation produces a tree with branch lengths, $(\tau, \mathbf{b})$, from $\mathbf{D}$ using a deterministic distance-based phylogeny reconstruction algorithm:

$$(\tau, \mathbf{b}) \;=\; f_{\mathrm{phy}}(\mathbf{D}),$$

where τ is a rooted binary tree on n leaves and $\mathbf{b} \in \mathbb{R}^{2n-2}$ contains the corresponding branch lengths. In the case of DNA substitution models, we use the neighbor-joining algorithm [64], and in the case of CRISPR barcode editing models, where an ultrametric tree is customary, we use the UPGMA (Unweighted Pair Group Method with Arithmetic Mean) algorithm [72]. In order to allow informative priors to be applied, we impose a rooting on neighbor-joining trees using the midpoint method (root at midpoint of longest span between taxa). Notably, in the CRISPR case only, we allow an additional free parameter for a leading branch to the root of the tree (making $2n - 1$ rather than $2n - 2$ branch-length parameters), as is typical in this literature.

Taking advantage of the reparameterization trick, let us define g to be the full transformation from $\mathbf{z}$ to $(\tau, \mathbf{b})$. We can specify g as a composition of the four functions above:

$$(\tau, \mathbf{b}) = g(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \theta_{\mathrm{NF}}) = \left(f_{\mathrm{phy}} \circ f_{\mathrm{D}} \circ f_{\mathrm{NF}} \circ f_{\mathrm{MVN}}\right)(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \theta_{\mathrm{NF}}).$$

Notice that the function g depends only on $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, as well as any free parameters θ_{NF} of the normalizing flows, because the component functions f_{phy} and f_{D} are parameter-free. For now, we will ignore θ_{NF} and focus on $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ for simplicity.

We can therefore re-express the ELBO $\mathcal{L}_q$ explicitly in terms of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, as follows:

$$\begin{aligned}\mathcal{L}_q &= \mathbb{E}_{q(\tau,\mathbf{b})}\left[\ell(\tau,\mathbf{b};\mathbf{X})\right] - \mathrm{KL}\left(q(\tau,\mathbf{b}) \,\|\, p(\tau,\mathbf{b})\right) \\ &= \mathbb{E}_{q(\mathbf{z})}[\ell(g(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\Sigma});\mathbf{X})] - \mathrm{KL}\left(q(\tau,\mathbf{b}) \,\|\, p(\tau,\mathbf{b})\right), \end{aligned} \tag{3}$$

where we take care to specify the distribution used for the expectation. Here, $q(\mathbf{z})$ denotes the standard MVN for $\mathbf{z}$ and $q(\tau,\mathbf{b})$ denotes the induced distribution over phylogenetic trees.

To proceed further, we must specify the prior distribution, $p(\tau,\mathbf{b})$. We consider two cases. In case (1), the default in VINE, we allow the prior to be defined implicitly by assuming that the MVN random variate $\mathbf{x} \sim \mathrm{MVN}(\mathbf{0},\mathbf{I})$, as is often done with variational autoencoders. In case (2), discussed below, we will allow for a general prior distribution over phylogenetic trees.

In case (1), the variational distribution is defined entirely with respect to $\mathbf{x}$; the tree $(\tau,\mathbf{b})$ is simply a deterministic function of this random variable that enables us to evaluate the phylogenetic likelihood. Therefore, we can write,

$$\begin{aligned}\mathcal{L}_q &= \mathbb{E}_{q(\mathbf{z})}[\ell(g(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\Sigma});\mathbf{X})] - \mathrm{KL}\left(q(\mathbf{x}) \,\|\, p(\mathbf{x})\right) \\ &= \mathbb{E}_{q(\mathbf{z})}[\ell(g(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\Sigma});\mathbf{X})] - \frac{1}{2}\left[\mathrm{tr}(\boldsymbol{\Sigma}) + \boldsymbol{\mu}^\top\boldsymbol{\mu} - nd - \log\det\boldsymbol{\Sigma}\right], \end{aligned} \tag{4}$$

where the last term represents the KLD for $q(\mathbf{x}) = \mathrm{MVN}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ with respect to $p(\mathbf{x}) = \mathrm{MVN}(\mathbf{0},\mathbf{I})$.

The first term of equation 4 is simply the expectation of the standard phylogenetic log likelihood under the variational distribution. Because of the complex nonlinear nature of the transformation g, there is no straightforward way to calculate this quantity exactly, but it can easily be estimated by Monte Carlo sampling (an alternative estimation method based on a Taylor approximation is described below). As a result, the approximate ELBO becomes,

$$\mathcal{L}_q \approx \frac{1}{M}\sum_{i=1}^{M}\left[\ell(g(\mathbf{z}_i;\boldsymbol{\mu},\boldsymbol{\Sigma});\mathbf{X})\right] - \frac{1}{2}\left[\mathrm{tr}(\boldsymbol{\Sigma}) + \boldsymbol{\mu}^\top\boldsymbol{\mu} - nd - \log\det\boldsymbol{\Sigma}\right], \tag{5}$$

where $\mathbf{z}_i$ is the ith sample drawn from q and M is the number of samples.

In case (2), allowing for a general phylogenetic prior, $p(\tau,\mathbf{b})$, the KLD is no longer available in closed form, so we re-express the ELBO in a form more typical for standard VI. Making the transformation g explicit, we have,

$$\begin{aligned}\mathcal{L}_q &= \mathbb{E}_{q(\mathbf{z})}[\ell(g(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\Sigma});\mathbf{X}) + \log p(g(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\Sigma}))] + \mathrm{H}\left(q(\mathbf{x})\right) \\ &= \mathbb{E}_{q(\mathbf{z})}[\ell(g(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\Sigma});\mathbf{X}) + \log p(g(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\Sigma}))] + \frac{1}{2}\left[nd(1+\log 2\pi) + \log\det\boldsymbol{\Sigma}\right] \end{aligned} \tag{6}$$

where $\mathrm{H}(q(\mathbf{x}))$ denotes the entropy of $q(\mathbf{x})$, which is also available in closed form. The expectation in equation 6 now represents the expected complete-data log likelihood, including both the prior $p(\tau,\mathbf{b})$ and

the conditional likelihood, $p(\mathbf{X}; \tau, \mathbf{b})$. These terms can be evaluated together by Monte Carlo sampling, in an analogous manner to case 1,

$$\mathcal{L}_q \approx \frac{1}{M} \sum_{i=1}^{M} \left[\ell(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X}) + \log p(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}))\right] + \frac{1}{2}\left[nd(1 + \log 2\pi) + \log\det \boldsymbol{\Sigma}\right]. \tag{7}$$

As described later, these Monte Carlo steps fit naturally into SGA, with M corresponding to the mini-batch size. In addition, for each sample $\mathbf{z}_i$ and induced tree $(\tau_i, \mathbf{b}_i) = g(\mathbf{z}_i)$, the log likelihood can be evaluated efficiently in the standard manner, using Felsenstein's pruning algorithm and either an appropriate DNA substitution model or a mutation model for CRISPR barcodes.

Chain rule for gradients

The gradient of the Monte-Carlo approximated ELBO (equation 5) with respect to the free parameters of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can easily be re-expressed in terms of gradients of the phylogenetic log likelihoods for the sampled points. In case (1) above (equation 5),

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\mu}} \mathcal{L}_q &\approx \frac{1}{M} \sum_{i=1}^{M} \left[\frac{\partial}{\partial \boldsymbol{\mu}} \ell(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X})\right] - \frac{1}{2}\left[\frac{\partial}{\partial \boldsymbol{\mu}} \boldsymbol{\mu}^\top \boldsymbol{\mu})\right] \\ &= \frac{1}{M} \sum_{i=1}^{M} \left[\frac{\partial}{\partial \boldsymbol{\mu}} \ell(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X})\right] - \boldsymbol{\mu}\end{aligned} \tag{8}$$

and

$$\frac{\partial}{\partial \boldsymbol{\Sigma}} \mathcal{L}_q \approx \frac{1}{M} \sum_{i=1}^{M} \left[\frac{\partial}{\partial \boldsymbol{\Sigma}} \ell(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X})\right] - \frac{1}{2} \frac{\partial}{\partial \boldsymbol{\Sigma}} \left[\mathrm{tr}(\boldsymbol{\Sigma}) - \log\det \boldsymbol{\Sigma}\right]. \tag{9}$$

Similarly, in case (2) (equation 7),

$$\frac{\partial}{\partial \boldsymbol{\mu}} \mathcal{L}_q \approx \frac{1}{M} \sum_{i=1}^{M} \left[\frac{\partial}{\partial \boldsymbol{\mu}} \ell(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X}) + \frac{\partial}{\partial \boldsymbol{\mu}} \log p(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}))\right] \tag{10}$$

because the entropy of $q(\mathbf{x})$ does not depend on $\boldsymbol{\mu}$, and,

$$\frac{\partial}{\partial \boldsymbol{\Sigma}} \mathcal{L}_q \approx \frac{1}{M} \sum_{i=1}^{M} \left[\frac{\partial}{\partial \boldsymbol{\Sigma}} \ell(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X}) + \frac{\partial}{\partial \boldsymbol{\Sigma}} \log p(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}))\right] + \frac{1}{2} \frac{\partial}{\partial \boldsymbol{\Sigma}} \log\det \boldsymbol{\Sigma}. \tag{11}$$

The last terms in equations 9 and 11 can be computed in closed-form in a manner that depends on the choice of parameterization for $\boldsymbol{\Sigma}$, as shown later. The key challenge is therefore to compute gradients for the phylogenetic log likelihood, $\ell(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X})$, and optionally, the corresponding log prior term. We will focus on the log likelihood here; the prior follows by analogy.

We start by observing that, in principle, we can propagate derivatives forward along the sequence of component functions f_{MVN}, f_{NF}, $f_{\mathbf{D}}$, and f_{phy} by computing the corresponding Jacobians, which are all

matrices of various sizes. In particular, for a vector of free parameters $\boldsymbol{\phi} \in \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ of dimension k, we can write,

$$
\begin{aligned}
J_{\text{MVN}} \in \mathbb{R}^{nd \times k} &= \frac{\partial \mathbf{x}}{\partial \boldsymbol{\phi}} = \frac{\partial}{\partial \boldsymbol{\phi}} f_{\text{MVN}}(\mathbf{z}_i) \\
J_{\text{NF}} \in \mathbb{R}^{nd \times nd} &= \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} f_{\text{NF}}(\mathbf{x}) \\
J_{\mathbf{D}} \in \mathbb{R}^{\binom{n}{2} \times nd} &= \frac{\partial \mathbf{D}}{\partial \mathbf{y}} = \frac{\partial}{\partial \mathbf{y}} f_{\mathbf{D}}(\mathbf{y}) \\
J_{\text{phy}} \in \mathbb{R}^{2n-2 \times \binom{n}{2}} &= \frac{\partial \mathbf{b}}{\partial \mathbf{D}} = \frac{\partial}{\partial \mathbf{D}} f_{\text{phy}}(\mathbf{D}),
\end{aligned}
$$

where in the last step we consider the branch lengths $\mathbf{b}$ only and ignore the tree topology τ, as discussed further below. To be concrete, these Jacobian matrices respectively contain the partial derivatives of all point coordinates with respect to all free parameters (J_{MVN}), all normalized-flow-transformed coordinates with respect to all untransformed point coordinates (J_{NF}), all pairwise distances with respect to all normalized-flow-tranformed coordinates ($J_{\mathbf{D}}$), and all branch lengths with respect to all pairwise distances (J_{phy}). To complete the process, we must also consider the gradient of the phylogenetic log likelihood with respect to the $2n-2$ branch lengths, a familiar quantity in phylogenetic analysis:

$$\nabla_{\mathbf{b}} \ell(\tau, \mathbf{b}; \mathbf{X}).$$

Assuming for the moment that these objects can all be obtained, the gradient of interest can be computed by a straightforward application of the chain rule:

$$\nabla_{\boldsymbol{\phi}} \ell(g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X}) = J_{\text{MVN}}^{\top} \; J_{\text{NF}}^{\top} \; J_{\mathbf{D}}^{\top} \; J_{\text{phy}}^{\top} \; \nabla_{\mathbf{b}} \ell(\tau, \mathbf{b}; \mathbf{X}).$$

The product of four Jacobian matrices has the ultimate effect of converting the $(2n-2)$-dimensional branch-length gradient to a k-dimensional parameter gradient, as required. In this way, the gradient of the phylogenetic log likelihood can be propagated backwards from the branch lengths, through the distance matrix, flows, and embedded points, to the MVN parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

As it happens, the first three of these Jacobians reflect elementary differentiable functions and are straightforward to compute analytically. Closed-form expressions are provided below in the sections below. The main difficulty lies with the fourth Jacobian, J_{phy}, which reflects the entire tranformation of a distance matrix $\mathbf{D}$ to a phylogenetic tree with branch lengths. This Jacobian is the subject of the next section.

Backpropagation through the neighbor-joining algorithm

In this section, we outline a strategy for efficient backpropagation of gradients through the neighbor-joining (NJ) algorithm, allowing explicit calculation of the critical fourth Jacobian, J_{phy}. We will show in the next section that a similar (but simpler) strategy can be used for the UPGMA algorithm as well.

We start with the observation that NJ is piecewise smooth and differentiable almost everywhere, meaning that small changes in the distance matrix $\mathbf{D}$, when propagated through the algorithm, mostly produce smooth

changes in the branch lengths $\mathbf{b}$ of the resulting tree. Exceptions occur only when changes in $\mathbf{D}$ alter the sequence of neighbors identified by the algorithm and therefore change the reconstructed tree topology. In other words, NJ is differentiable except for intermittent discontinuities that result from the step in the algorithm where the two neighbors of minimum distance are identified.

We can develop this idea further by decomposing the algorithm into two parts: (1) a sequence of selections of neighbors to be joined in the agglomerative clustering procedure; and (2) a mapping from the distance matrix $\mathbf{D}$ to the branch lengths $\mathbf{b}$ conditional on these neighbor selections. Our intuition is that the most of the information for gradient-based optimization is in this second, smooth component of the problem. We will therefore condition on the sequence of neighbors (1) for a particular value of the embedded points $\mathbf{x}$ and efficiently characterize the continuous mapping (2) given that sequence. We will then rely more heuristically on the general stochasticity of SGA to explore different tree topologies as points migrate in the embedding space.

Recall that NJ operates by iteratively updating two distance matrices, typically denoted $\mathbf{Q}$ and $\mathbf{D}$ (e.g., [80]). These matrices have quite different purposes: $\mathbf{Q}$ is used solely to determine the selection of the neighbors to be joined on each step of the algorithm, and $\mathbf{D}$ is used solely to capture the updated distances between nodes. $\mathbf{Q}$ and $\mathbf{D}$, therefore, map exactly to our components (1) and (2) of the algorithm. Since we condition on the neighbor-selection sequence, $\mathbf{Q}$ plays no further role in the derivation of the Jacobian and we can focus solely on $\mathbf{D}$.

As it turns out, the manner in which NJ updates $\mathbf{D}$ conditional on a sequence of neighbors is remarkably simple. Let us recast the problem in the following way. Let $\mathbf{D}^{(t)}$ denote an $m_t \times m_t$ symmetric distance matrix for iteration t of the algorithm, representing pairwise distances between the members of a set S_t of m_t (internal or external) active nodes. During the course of the algorithm, $\mathbf{D}^{(t)}$ will evolve from the original $n \times n$ distance matrix for the n taxa to a matrix of three remaining nodes at the termination step of the algorithm (at which point the unrooted tree is fully resolved). Thus the number of active nodes, m_t, on each step t of the algorithm is given by $m_t = n - t$ for $t = 0, \ldots, n-3$.

Let us first consider the recursive step by which NJ derives $\mathbf{D}^{(t+1)}$ from $\mathbf{D}^{(t)}$ conditional on a specified choice of neighbors (f, g). The NJ algorithm obtains $\mathbf{D}^{(t+1)}$ from $\mathbf{D}^{(t)}$ by defining a new node u representing the cluster of f and g, and replacing the rows and columns corresponding to f and g with a single row and column for u. The new row and column are then populated by the formula,

$$d_{uk}^{(t+1)} = d_{ku}^{(t+1)} = \frac{1}{2}\big(d_{fk}^{(t)} + d_{gk}^{(t)} - d_{fg}^{(t)}\big), \quad k \in S_t \setminus \{f, g, u\}, \tag{12}$$

with $d_{uu}^{(t+1)} = 0$ as for all diagonal entries. All other entries in $\mathbf{D}^{(t+1)}$ are left unchanged.

Importantly, this operation is not only smooth and differentiable, it is *linear*. Thus, we can postulate that, if $\mathbf{d}^{(t)} \in \mathbb{R}^{\binom{m_t}{2}}$ is a vectorized representation of $\mathbf{D}^{(t)}$, then there exists a matrix $\mathbf{A}^{(t)}$ such that $\mathbf{d}^{(t+1)} = \mathbf{A}^{(t)}\mathbf{d}^{(t)}$ captures the recursive transformation accomplished by the NJ algorithm. Therefore, the effect of the entire algorithm conditional on a choice of neighbors can be described as a product of such matrices,

$$\mathbf{d}^{(T)} = \mathbf{A}^{(T-1)}\mathbf{A}^{(T-2)} \cdots \mathbf{A}^{(0)}\mathbf{d}^{(0)}, \tag{13}$$

where $T = n - 3$.

In addition to altering the $\mathbf{Q}$ and $\mathbf{D}$ matrices, the NJ algorithm also must update the branch-lengths of the emerging tree. This operation can also be captured as a recursive matrix update. When the algorithm joins neighbors f and g beneath a new node u, it sets the branch lengths between f and u and between g and u, respectively, to,

$$b_{fu}^{(t)} = \frac{1}{2} d_{fg}^{(t)} + \frac{1}{2(m_t - 2)} \sum_{k \in S_t} \left(d_{fk}^{(t)} - d_{gk}^{(t)}\right),$$
$$b_{gu}^{(t)} = d_{fg}^{(t)} - b_{fu}^{(t)},$$

where the $d_{\cdot\cdot}^{(t)}$ terms are all elements of the current vector $\mathbf{d}^{(t)}$. Again, this is clearly a linear operation that can be abstractly described as,

$$\mathbf{b}^{(t)} = \mathbf{C}^{(t)} \mathbf{d}^{(t)},$$

where $\mathbf{b}^{(t)} = (b_{iu}^{(t)}, b_{ju}^{(t)})'$ and $\mathbf{C}^{(t)}$ is a small matrix.

These $\mathbf{b}^{(t)}$ vectors are disjoint, because the algorithm incrementally defines two new branch lengths per iteration. The final vector of branch lengths, $\mathbf{b}$, can be defined by stacking them together,

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}^{(0)} \\ \mathbf{b}^{(1)} \\ \mathbf{b}^{(2)} \\ \vdots \\ \mathbf{b}^{(T)} \end{bmatrix} = \begin{bmatrix} \mathbf{C}^{(0)}\mathbf{d}^{(0)} \\ \mathbf{C}^{(1)}\mathbf{d}^{(1)} \\ \mathbf{C}^{(2)}\mathbf{d}^{(2)} \\ \vdots \\ \mathbf{C}^{(T)}\mathbf{d}^{(T)} \end{bmatrix} = \begin{bmatrix} \mathbf{C}^{(0)} \\ \mathbf{C}^{(1)}\mathbf{A}^{(0)} \\ \mathbf{C}^{(2)}\mathbf{A}^{(1)}\mathbf{A}^{(0)} \\ \vdots \\ \mathbf{C}^{(T)}\mathbf{A}^{(T-1)} \cdots \mathbf{A}^{(0)} \end{bmatrix} \mathbf{d}^{(0)}, \tag{14}$$

where the last step uses equation 13. The Jacobian of interest, J_{phy}, follows directly from equation 14:

$$J_{\text{phy}} = \frac{\partial \mathbf{b}}{\partial \mathbf{D}} = \frac{\partial \mathbf{b}}{\partial \mathbf{d}^{(0)}} = \begin{bmatrix} \mathbf{C}^{(0)} \\ \mathbf{C}^{(1)}\mathbf{A}^{(0)} \\ \mathbf{C}^{(2)}\mathbf{A}^{(1)}\mathbf{A}^{(0)} \\ \vdots \\ \mathbf{C}^{(T)}\mathbf{A}^{(T-1)} \cdots \mathbf{A}^{(0)} \end{bmatrix}.$$

This derivation makes the structure of J_{phy} conceptually clear, but it is not practical to compute the matrices in equation 14 explicitly, because the intermediate Jacobians are large and dense. Instead, we compute the gradient $\nabla_{\mathbf{d}^{(0)}} \ell(\tau, \mathbf{b}; \mathbf{X})$ without forming the Jacobian, using a reverse-mode algorithm that mirrors the NJ updates.

To enable this calculation, we record during the forward execution of NJ each pair of neighbors to be joined, the new internal node that replaces them, and the associated distance values. Later, we propagate the branchwise phylogenetic gradient $\nabla_{\mathbf{b}} \ell(\tau, \mathbf{b}; \mathbf{X})$ back to $\mathbf{d}^{(0)}$ using this recorded "tape" of the forward merges as a guide. In particular, we initialize accumulated gradients for the final two active branches, and then update the gradients on each step t using the chain-rule-based recurrence,

$$\lambda_{fg}^{(t)} \mathrel{+}= \tfrac{1}{2}(\lambda_{t_f} + \lambda_{t_g}), \qquad \lambda_{fk}^{(t)} \mathrel{+}= \frac{\lambda_{t_f} - \lambda_{t_g}}{2(m_t - 2)}, \qquad \lambda_{gk}^{(t)} \mathrel{-}= \frac{\lambda_{t_f} - \lambda_{t_g}}{2(m_t - 2)},$$

for joined neighbors f, g, and all active $k \neq f, g$, where $\lambda_{t_f} = \frac{\partial \ell(\tau, \mathbf{b}; \mathbf{X})}{\partial b_{fu}}$ and $\lambda_{t_g} = \frac{\partial \ell(\tau, \mathbf{b}; \mathbf{X})}{\partial b_{gu}}$ are the derivatives with respect to the newly created branch lengths, b_{fu} and b_{gu}. These updates are combined with the gradient updates implied by equation 12, which propagate accumulated gradients $\lambda_{uk}^{(t+1)}$ from the new node u to f, g, and k. Because all transformations are linear, this reverse sweep ultimately computes all quantities of the form,

$$\lambda_{ij}^{(0)} = \frac{\partial \ell(\tau, \mathbf{b}; \mathbf{X})}{\partial d_{ij}^{(0)}},$$

which together comprise the desired gradient $\nabla_{\mathbf{d}^{(0)}} \ell(\tau, \mathbf{b}; \mathbf{X})$. Even though some intermediate Jacobians are of size $O(n^3)$, this calculation requires only $O(n^2)$ time and memory, making it asymptotically equivalent to the NJ algorithm itself.

Calculation of the Jacobian for UPGMA

In the case of the UPGMA algorithm for reconstructing an ultrametric tree, which we use with the CRISPR mutation model, the relationship between the branch lengths and the distance matrix is simpler, and the gradient $\nabla_{\mathbf{d}^{(0)}} \ell(\tau, \mathbf{b}; \mathbf{X})$ can be obtained in a single post-processing traversal of the tree. In this case, the height h_u assigned to an internal node u is simply the average of all intercluster distances between the leaves descending from its left child and those descending from its right child. Therefore, for any pair of leaves i, j, the partial derivative of h_u with respect to the distance d_{ij} is given by,

$$\frac{\partial h_u}{\partial d_{ij}} = \begin{cases} \dfrac{1}{2\,|L_u|\,|R_u|}, & \text{if } a \in L_u,\ b \in R_u, \\ 0, & \text{otherwise}, \end{cases}$$

where L_u and R_u denote the leaf sets beneath the left and right children of u, respectively. The branch lengths can be obtained as height differences between nodes and their parents: $b_u = h_{\pi(u)} - h_u$, where $\pi(u)$ is the parent of node u. The component entries of the branch-length Jacobian J_{phy} are therefore simply equal to the differences in the corresponding node-height derivatives,

$$\frac{\partial b_u}{\partial d_{ij}} = \frac{\partial h_{\pi(u)}}{\partial d_{ij}} - \frac{\partial h_u}{\partial d_{ij}}.$$

As in the NJ case, we avoid instantiating the full Jacobian by working with accumulated gradients. In particular, for each internal node u, the only quantity required by backpropagation is the directional derivative $\sum_{i<j} (\partial b_u / \partial d_{ij})\, \lambda_{ij}$, where λ_{ij} is the incoming sensitivity with respect to the pairwise distance d_{ij}. Because each node height h_u depends only on the distances between leaves in L_u and R_u, this directional derivative can be computed in a single pass over those leaf pairs. Thus the branch-length gradients are obtained by directly accumulating the relevant $1/(2|L_u||R_u|)$ contributions, without forming any intermediate matrices, and the overall cost is linear in the size of the tree.

Implementations of Neighbor-Joining and UPGMA

VINE uses custom implementations in C of the NJ and UPGMA algorithms that take a distance matrix as input and return both a tree and the Jacobian matrix J_{phy}. In the case of NJ, J_{phy} is recursively calculated in tandem with the main aggomerative steps, whereas in UPGMA it is computed after the full tree has been constructed.

Because these tree-building routines are rate-limiting for variational inference, we adapted the standard algorithms to use a min-heap for efficient identification of the pair of nodes to join on each step (e.g., the minimum entry of $\mathbf{Q}$ in NJ). This approach reduces the asymptotic running time from $O(n^3)$ for naive implementations to $O(n^2 \log n)$ (see, e.g., [81], who use further optimizations to achieve $O(n^2)$ for NJ). In addition, as noted above, the NJ implementation uses sparse-matrix and shallow-copy operations to accelerate the recursive computation of the Jacobian.

Taylor approximation of the ELBO

Estimating the ELBO by Monte Carlo sampling (equations 5 & 7) is effective for optimization but computationally expensive. With a minibatch size of M, Monte-Carlo estimation of the ELBO requires M evaluations of the entire pipeline from a draw of a MVN random variate $\mathbf{z}_i$ through the transformation $g(\mathbf{z}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ to the calculation of the phylogenetic likelihood, paired with backpropagation to obtain the gradients for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, for a total cost of $O(Mn^2 \log n)$. The time required for the entire SGA algorithm is therefore linear in the minibatch size M. In practice, M cannot be too small or the variance of the estimator begins to erode the efficiency of SGA. In our experiments, $M = 10$ is an approximate minimum effective value. Thus, Monte-Carlo estimation of the ELBO amplifies the cost of the VI algorithm by roughly ten-fold.

Intuition suggests that efficiency could be improved by exploiting properties of the MVN random variate $\mathbf{x}$, which represents the embedding of the taxa in nd-dimensional space. All downstream steps in g are deterministic, so it is the distribution of $\mathbf{x}$ that defines the expectation in the ELBO. But the distribution of $\mathbf{x}$ is dominated by its first moment, $\boldsymbol{\mu}$. The variance of $\mathbf{x}$ effectively contributes only a small degree of scatter around the mean. This observation suggests that most of the information in the ELBO could be obtained by evaluating it once with a value of $\mathbf{x}$ equal to the mean $\boldsymbol{\mu}$ rather than M times for samples scattered tightly around $\boldsymbol{\mu}$.

This notion can be formalized using a Taylor approximation for the ELBO. Recall that the key component of the ELBO is the expectation of the phylogenetic log likelihood, because its other component, the KLD for $q(\mathbf{x})$ relative to $p(\mathbf{x})$, is available in closed form (equation 4). We can approximate the expectation of the log likelihood using a second-order Taylor approximation around the mean, using the fact that $\mathbf{z} = \mathbf{0}$ implies $\mathbf{x} = \boldsymbol{\mu}$,

$$\begin{aligned}\mathbb{E}_{q(\mathbf{z})}[\ell(g(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X})] &\approx \ell(g(\mathbf{z} = \mathbf{0}; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X}) + \nabla_{\boldsymbol{\mu}, \boldsymbol{\Sigma}}\, \ell(g(\mathbf{z} = \mathbf{0}; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X})^\top \mathbf{0} + \frac{1}{2}\mathrm{tr}\left(\mathbf{H}\boldsymbol{\Sigma}\right) \\ &= \ell(g(\mathbf{z} = \mathbf{0}; \boldsymbol{\mu}); \mathbf{X}) + \frac{1}{2}\mathrm{tr}\left(\mathbf{H}\boldsymbol{\Sigma}\right), \end{aligned} \tag{15}$$

where $\mathbf{H}$ is the Hessian matrix for the entire transformation, $\ell(g(\mathbf{z} = \mathbf{0}; \boldsymbol{\mu}, \boldsymbol{\Sigma}); \mathbf{X})$, as evaluated at $\mathbf{z} = \mathbf{0}$ or $\mathbf{x} = \boldsymbol{\mu}$. Notice that the first-order term of the Taylor expansion, involving the gradient of the log likelihood at $\mathbf{z} = \mathbf{0}$, reduces to zero owing to choice of the mean as the expansion point. Here we focus on case (1) for the prior, $\mathbf{x} \sim \mathrm{MVN}(\mathbf{0}, \mathbf{I})$; case (2) follows by analogy (see equation 6).

Importantly, while the second-order term of equation 15 will have some indirect dependency on $\boldsymbol{\mu}$, we expect this dependency to be fairly weak, assuming the structure of the Hessian is similar throughout the neighborhood around the mean. If we ignore this dependency, then the gradient of this approximate ELBO with respect to the mean $\boldsymbol{\mu}$ depends only on the first term and the gradient with respect to the covariance $\boldsymbol{\Sigma}$ depends only on the second term. Thus, this formulation allows us to decompose the ELBO into mean- and variance-related components. The mean component has a simple form consistent with our notion that most information about this parameter comes from evaluation of the likelihood when $\mathbf{x} = \boldsymbol{\mu}$. In practiced, we find that this term is strongly dominant when the model is fitted to data.

On its face, the second term in equation 15 still poses a problem, because the $nd \times nd$ Hessian matrix is impractical to instantiate explicitly. We can considerably simplify this term, however, in two ways. First, we exploit the near-linearity of the transformation g to re-express the Hessian-covariance product $\mathbf{H}\boldsymbol{\Sigma}$ in terms of a smaller, simpler Hessian $\mathbf{H_b}$ defined over the space of branch lengths $\mathbf{b}$. Consider the portion of the full transformation $g_{x \to b}$ that maps an embedding $\mathbf{x}$ to a set of branch lengths $\mathbf{b}$, with the tree topology held fixed, and assume for simplicity that the normalizing flows are not active. Using our previous notation, $g_{x \to b}$ is a composition of the functions $f_{\mathbf{D}}$ and f_{phy} and, by the chain rule, the associated Jacobian is,

$$J_{\mathbf{bx}} \;=\; \frac{\partial \mathbf{b}}{\partial \mathbf{x}} \;=\; J_{\text{phy}}\, J_{\mathbf{D}} \in \mathbb{R}^{(2n-2)\times nd}. \tag{16}$$

As shown earlier, the map f_{phy} from $\mathbf{D}$ to $\mathbf{b}$ is linear and therefore contributes no curvature to the Hessian $\mathbf{H}$. If we additionally approximate the distance function $f_{\mathbf{D}}$ as locally linear, then we can re-express the full Hessian $\mathbf{H}$ in terms of a smaller, simpler Hessian $\mathbf{H_b} \in \mathbb{R}^{(2n-2)\times(2n-2)}$ describing the curvature of the log likelihood surface with respect to the $2n-2$ branch-length parameters. ($\mathbf{H_b}$ is the Hessian corresponding to the standard phylogenetic gradient, $\nabla_{\mathbf{b}} \ell(\tau, \mathbf{b}; \mathbf{X})$.) With this approximation, we can write $\mathbf{H} \;\approx\; J_{\mathbf{bx}}^{\top}\, \mathbf{H_b}\, J_{\mathbf{bx}}$, and then, using a cyclic permutation of the trace,

$$\mathrm{tr}(\mathbf{H}\boldsymbol{\Sigma}) \;\approx\; \mathrm{tr}\big(J_{\mathbf{bx}}^{\top}\, \mathbf{H_b}\, J_{\mathbf{bx}}\, \boldsymbol{\Sigma}\big) \;=\; \mathrm{tr}\big(\mathbf{H_b}\, J_{\mathbf{bx}}\, \boldsymbol{\Sigma}\, J_{\mathbf{bx}}^{\top}\big) \;=\; \mathrm{tr}\big(\mathbf{H_b}\, \mathbf{S}\big),$$

where $\mathbf{S} = J_{\mathbf{bx}}\, \boldsymbol{\Sigma}\, J_{\mathbf{bx}}^{\top}$ is a covariance-like matrix defined in branch-length space.

Second, we take advantage of efficient methods for computing $\mathrm{tr}\big(\mathbf{H_b}\, \mathbf{S}\big)$ by way of matrix-vector products, without realizing either matrix. We estimate the trace from k random probes $\mathbf{z}_1, \ldots, \mathbf{z}_k$ of dimension $2n-2$, using Hutchinson's method [82], with

$$\mathrm{tr}\big(\mathbf{H_b}\, \mathbf{S}\big) = \mathbb{E}\left[\mathbf{z}^{\top} \mathbf{H_b} \mathbf{S} \mathbf{z}\right] \approx \frac{1}{k} \sum_{i=1}^{k} \mathbf{z}_i^{\top} \mathbf{H_b}\, (\mathbf{S}\, \mathbf{z}_i)\,. \tag{17}$$

In turn, we compute each matrix-vector product of the form $\mathbf{u} = \mathbf{S}\, \mathbf{z}_i$ by propagating the vector $\mathbf{z}_i$ through the component Jacobian matrices $\mathbf{u} = \mathbf{S}\, \mathbf{z}_i = J_{\text{phy}}\, (J_{\mathbf{D}}\, (\boldsymbol{\Sigma}\, (J_{\mathbf{D}}^{\top}\, (J_{\text{phy}}^{\top}\, \mathbf{z}_i))))$, adapting our backpropagation machinery as needed for each transformation. Finally, we compute $\mathbf{H_b}\mathbf{u}$ as a directional derivative of the

log likelihood (e.g., [83]), taking advantage of our efficient methods for computing branch-length gradients $\nabla_{\mathbf{b}}\ell(\tau, \mathbf{b}; \mathbf{X})$ (see below). We simply approximate $\mathbf{H_b u}$ by finite differences of the branch-length gradient as,

$$\mathbf{H_b u} \approx \frac{\nabla_{\mathbf{b}}\ell(\tau, \mathbf{b} + \epsilon\,\mathbf{u}; \mathbf{X}) - \nabla_{\mathbf{b}}\ell(\tau, \mathbf{b}; \mathbf{X})}{\epsilon}, \tag{18}$$

where ϵ is a small step size. This computation requires only one additional linear-time evaluation of the gradient per probe.

For optimization by SGA, gradients of equation (15) are also required. As noted, the gradient with respect to $\boldsymbol{\mu}$ depends only on the first term of the approximation and the gradient with respect to $\boldsymbol{\Sigma}$ depends only on the second term. The gradient with respect to $\boldsymbol{\mu}$ can be obtained using our standard backpropagation technique for the phylogenetic log likelihood evaluated at $\mathbf{x} = \boldsymbol{\mu}$. For the gradient with respect to $\boldsymbol{\Sigma}$, we extended the Hutchinson framework above to compute the gradient as well as the value of the stochastic trace estimator. In particular, each probe contributes a scalar of the form $\mathbf{z}^\top \mathbf{H_b S z}$, whose gradient with respect to $\boldsymbol{\Sigma}$ is computed by backpropagating through the same sequence of Jacobian–vector products used to form $\mathbf{Sz}$. In this way, gradients with respect to $\boldsymbol{\Sigma}$ are obtained using only matrix–vector products and directional derivatives, without ever instantiating either the Hessian or its gradient explicitly.

Additional gains in efficiency can be obtained by recognizing that the quantity $\mathrm{tr}\big(\mathbf{H_b}\,\mathbf{S}\big)$ is not highly sensitive to $\boldsymbol{\mu}$ and need not be updated on every iteration of SGA. We delay computing it during a "warmup" period (50 iterations), and then update it only every 30 iterations, using an exponential moving average for both the trace quantity and its gradient. With $k = 10$ probes, this strategy requires less than one additional evaluation per iteration of our full $O(n^2 \log n)$ pipeline, decreasing the overall cost of SGA by nearly a factor of M in comparison to Monte Carlo sampling. In practice, we obtain nearly identical parameter estimates in about one fifth the time.

Parameterizations of the covariance matrix $\boldsymbol{\Sigma}$

VINE supports four parameterizations of the MVN covariance matrix $\boldsymbol{\Sigma}$, which are labeled CONST, DIAG, DIST, and LOWR in order of increasing complexity (accessible via the `--covar` option). The CONST parameterization simply assumes $\boldsymbol{\Sigma} = \lambda\mathbf{I}$ and ensures nonnegativity of λ by defining it as $\lambda = e^{\eta}$, with η as a free parameter. The DIAG parameterization allows for a general diagonal covariance matrix, with free variance along each of the nd dimensions but no covariance between them: $\boldsymbol{\Sigma} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_{nd}\}$, with each $\lambda_i = e^{\eta_i}$ for nd free parameters.

The DIST and LOWR parameterizations attempt to capture more of the covariance structure across taxa while keeping the parameterization as sparse as possible. In these cases, the same covariance structure is assumed for each of the d dimensions in the embedding space, with no covariance across these dimensions. The full covariance matrix can therefore be described as $\boldsymbol{\Sigma} = \mathbf{I}_d \otimes \boldsymbol{\Sigma}_0$, where $\mathbf{I}_d$ is a $d \times d$ identity matrix, $\otimes$ is the Kronecker product, and $\boldsymbol{\Sigma}_0$ is the $n \times n$ covariance matrix shared for each embedding dimension. In these cases, VINE gains further efficiency by performing all MVN-related calculations on d independent n-dimensional MVNs with a shared covariance structure, rather than on one nd-dimensional MVN.

The DIST parameterization further assumes $\mathbf{\Sigma}_0 = \lambda \mathbf{S}$ and $\lambda = e^{\eta}$, resulting in a single free variance parameter η for all taxa. In this case, $\mathbf{S}$ is kept fixed at a double-centered version of the initial distance matrix $\mathbf{D}_0$,

$$\mathbf{S} = -\tfrac{1}{2}\mathbf{H}\mathbf{D}_0\mathbf{H}, \qquad \mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\top}, \tag{19}$$

where $\mathbf{1}$ indicates an n-dimensional vector of all 1s and $\mathbf{1}\mathbf{1}^{\top}$ is therefore an $n \times n$ matrix of all 1s. This matrix $\mathbf{S}$ captures the expected covariance structure of a random variable evolving by Brownian motion along the branches of an unrooted tree, and therefore, the DIST parameterization is a simple way to consider the distances between taxa without restricting the tree topology.

The most flexible parameterization, LOWR, allows for a general low-rank representation of $\mathbf{\Sigma}_0$, with $\mathbf{\Sigma}_0 = \mathbf{R}\mathbf{R}^{\top}$ for a general matrix $\mathbf{R}$ of dimension $n \times w$ (with nw free parameters). By default, $w = 3$ (see `--rank`). In this case, the full covariance of the embedding is $\Sigma = (RR^{\top}) \otimes I_d$, which has rank at most wd. Consequently, VINE can reparameterize sampling from $\mathrm{MVN}(\mu, \Sigma)$ using only wd latent dimensions, instead of the full nd dimensions required by an unrestricted covariance matrix.

All of these parameterizations permit closed-form expressions for the gradient of the KLD (equation 9) or entropy (equation 11) and the Jacobian J_{MVN}. These expressions have simple forms under the CONST and DIST parameterizations. In either case, we can write $\mathbf{\Sigma} = e^{\eta}\mathbf{S}$, where $\mathbf{S} = \mathbf{I}$ for CONST. The gradient of the KLD with respect to the single free parameter η for CONST and DIST is therefore,

$$\begin{aligned} \frac{\partial}{\partial \eta}\frac{1}{2}\left[\mathrm{tr}(\mathbf{\Sigma}) - \log|\det \mathbf{\Sigma}|\right] &= \frac{1}{2}\frac{\partial}{\partial \eta}\left[\mathrm{tr}(e^{\eta}\mathbf{S}) - \log|\det(e^{\eta}\mathbf{S})|\right] \\ &= \frac{1}{2}\left[e^{\eta}\cdot \mathrm{tr}(\mathbf{S}) - nd\right], \end{aligned} \tag{20}$$

because $\log|\det(e^{\eta}\mathbf{S})| = \log(e^{\eta})^{nd} + \log\det(\mathbf{S}) = nd\eta + \log\det(\mathbf{S})$. Under the CONST parameterization, we have the additional simplification that $\mathrm{tr}(\mathbf{S}) = \mathrm{tr}(\mathbf{I}) = nd$, so the entire expression reduces to $\frac{nd}{2}\left(e^{\eta} - 1\right)$. In case (2) (equation 11), we can drop the trace term and both expressions reduce to $-\frac{1}{2}nd$.

In the DIAG case, we can write $\mathbf{\Sigma} = \mathbf{\Lambda}$, with $\mathbf{\Lambda} = \mathrm{diag}\{e^{\eta_1}, \ldots, e^{\eta_{nd}}\}$, so that $\mathrm{tr}(\mathbf{\Sigma}) = \sum_i e^{\eta_i}$ and $\log|\det \mathbf{\Sigma}| = \sum_i \eta_i$. Therefore, the partial derivative with respect to each η_i is simply,

$$\frac{\partial}{\partial \eta_i}\frac{1}{2}\left[\mathrm{tr}(\mathbf{\Sigma}) - \log|\det \mathbf{\Sigma}|\right] = \frac{1}{2}\left(e^{\eta_i} - 1\right),$$

which reduces to $-\frac{1}{2}$ in case 2.

For the LOWR parameterization, $\mathbf{\Sigma}_0 = \mathbf{R}\mathbf{R}^{\top}$ introduces non-diagonal structure and requires derivatives with respect to the entries of $\mathbf{R}$. Using standard matrix derivatives, $\partial/\partial\mathbf{R}\,\mathrm{tr}(\mathbf{R}\mathbf{R}^{\top}) = 2\mathbf{R}$ and $\partial/\partial\mathbf{R}\,\log|\det \mathbf{R}\mathbf{R}^{\top}| = 2\left(\mathbf{R}^{\top}\right)^{-1} = 2\mathbf{R}^{-\top}$. Therefore, we can express the gradient in matrix form as,

$$\frac{\partial}{\partial \mathbf{R}}\frac{1}{2}\left[\mathrm{tr}(\mathbf{\Sigma}) - \log|\det \mathbf{\Sigma}|\right] = d\left(\mathbf{R} - \mathbf{R}^{-\top}\right),$$

or simply $-d\,\mathbf{R}^{-\top}$ in case 2.

Evaluation of the Jacobian J_{MVN} leads to similarly simple expressions. Recall that the reparameterization function $\mathbf{x} = f_{\mathrm{MVN}}(\mathbf{z}) = \boldsymbol{\mu} + \mathbf{L}\mathbf{z}$, where $\mathbf{L}$ is a Cholesky factor for $\boldsymbol{\Sigma}$. Therefore, the portion of the Jacobian J_{MVN} pertaining to $\boldsymbol{\Sigma}$ is given by,

$$\frac{\partial}{\partial \boldsymbol{\Sigma}} [\boldsymbol{\mu} + \mathbf{L}\mathbf{z}] = \left(\frac{\partial}{\partial \boldsymbol{\Sigma}} \mathbf{L} \right) \mathbf{z}.$$

Notice in the CONST case, however, that we can simply write $\mathbf{L} = \sqrt{e^{\eta}}\,\mathbf{I} = e^{\frac{\eta}{2}}\mathbf{I}$, meaning that,

$$\frac{\partial}{\partial \eta} [\boldsymbol{\mu} + \mathbf{L}\mathbf{z}] = \frac{1}{2} e^{\frac{\eta}{2}} \mathbf{z}.$$

Similarly, in the DIAG case, $\mathbf{L} = \boldsymbol{\Lambda}^{\frac{1}{2}}$, where $\boldsymbol{\Lambda} = \mathrm{diag}\{e^{\eta_1}, \ldots, e^{\eta_{nd}}\}$, so for each η_i,

$$\frac{\partial}{\partial \eta_i} [\boldsymbol{\mu} + \mathbf{L}\mathbf{z}] = \tfrac{1}{2} e^{\frac{\eta_i}{2}} \mathbf{e}_i.$$

where $\mathbf{e}_i$ denotes the ith standard basis vector in $\mathbb{R}^{nd}$, i.e. the vector with a 1 in position i and 0 in all other positions.

In the DIST case, if we assume $\mathbf{L} = \sqrt{e^{\eta}}\mathbf{L}'$, where $\mathbf{L}'$ is a Cholesky factor for the double-centered distance matrix $\mathbf{S}$, then,

$$\frac{\partial}{\partial \eta} [\boldsymbol{\mu} + \mathbf{L}\mathbf{z}] = \frac{1}{2} e^{\frac{\eta}{2}} \mathbf{L}'\mathbf{z}.$$

For the LOWR parameterization, it is convenient to work at the level of $\boldsymbol{\Sigma}_0 = \mathbf{R}\mathbf{R}^{\top}$. For a single embedding dimension k, we can write

$$\mathbf{x}^{(k)} = \boldsymbol{\mu}^{(k)} + \mathbf{R}\mathbf{z}^{(k)},$$

where $\mathbf{z}^{(k)} \in \mathbb{R}^{w}$ collects the kth block of latent standard normal variables. The Jacobian of $\mathbf{x}^{(k)}$ with respect to $\mathbf{R}$ is linear in $\mathbf{z}^{(k)}$ and can be expressed elementwise as

$$\frac{\partial \mathbf{x}^{(k)}}{\partial R_{ij}} = z_j^{(k)}\, \mathbf{e}_i,$$

with $\mathbf{e}_i$ the ith standard basis vector in $\mathbb{R}^{n}$. Because the full covariance matrix is $\boldsymbol{\Sigma} = \mathbf{I}_d \otimes \boldsymbol{\Sigma}_0$, the complete Jacobian with respect to $\mathbf{R}$ is obtained by stacking these d blocks, one for each embedding dimension.

Notice that under all parameterizations, the gradient with respect to $\boldsymbol{\mu}$ is trivial.

$$\frac{\partial}{\partial \boldsymbol{\mu}} [\boldsymbol{\mu} + \mathbf{L}\mathbf{z}] = \mathbf{1}$$

where $\mathbf{1}$ denotes a vector of all 1s of length nd.

Normalizing flows

VINE supports two types of normalizing flows to accommodate nonlinearities in the relationship between $\mathbf{x} \sim \mathrm{MVN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and a transformed set of embedded points, $\mathbf{y}$: a *radial flow* and a *planar flow*. Both are disabled by default but can be activated by command-line options (`--radial-flow` and `--planar-flow`, respectively). They can be used in combination if desired.

Radial flow. The radial flow is defined by two scalar free parameters, α and β, and a free d-dimensional vector $\mathbf{c}$ (for $d+2$ degrees of freedom). Let $\mathbf{u} = \mathbf{x} - \mathbf{c}$ and $r = \|\mathbf{u}\|$. The transformation $\mathbf{y} = f_{\mathrm{NF}}(\mathbf{x})$ is,

$$\mathbf{y} = \mathbf{x} + \beta h\, \mathbf{u}, \qquad h = \frac{1}{\alpha + r}.$$

This function radially contracts or expands points toward the center $\mathbf{c}$ with strength controlled by α and β.

The Jacobian $J_{\mathrm{NF}}(\mathbf{x}) = \partial \mathbf{y}/\partial \mathbf{x}$ has the rank–1 update form,

$$J_{\mathrm{NF}}(\mathbf{x}) = A\, \mathbf{I}_d + S\, \mathbf{u}\mathbf{u}^\top, \qquad A = 1 + \beta h, \quad S = -\beta\, \frac{h^2}{r}.$$

The gradients with respect to the flow parameters, which are needed for SGA-based optimization, are:

$$\frac{\partial \mathcal{L}}{\partial \alpha} = \beta \left[-h^2 \mathbf{u}^\top \mathbf{g} \;-\; (d-1)\frac{h^2}{A} \;+\; \frac{-h^2 + 2rh^3}{D} \right], \tag{21}$$

$$\frac{\partial \mathcal{L}}{\partial \beta} = h\, \mathbf{u}^\top \mathbf{g} \;+\; (d-1)\frac{h}{A} \;+\; \frac{h - rh^2}{D}. \tag{22}$$

where $\mathbf{g} = \partial \mathcal{L}/\partial \mathbf{y}$ is the incoming gradient. The gradient with respect to the center vector is,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}} = -\beta h\, \mathbf{g} + \beta \frac{h^2}{r} (\mathbf{u}^\top \mathbf{g})\, \mathbf{u} - \frac{1}{r} \left(\frac{\partial \log |\det J|}{\partial r} \right) \mathbf{u},$$

where

$$\frac{\partial \log |\det J|}{\partial r} = \beta \left[-(d-1)\frac{h^2}{A} + \frac{-2h^2 + 2rh^3}{B} \right].$$

To ensure positivity and numerical stability, α and β are obtained from raw parameters a and b using the softplus function, $\alpha = \mathrm{softplus}(a)$ and $\beta = \mathrm{softplus}(b)$. By the chain rule,

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial \mathcal{L}}{\partial \alpha}\, \sigma(a), \qquad \frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial \beta}\, \sigma(b),$$

where $\sigma(\cdot)$ is the logistic function.

Planar flow. The planar flow is defined by two free d-dimensional vectors, $\mathbf{u}$ and $\mathbf{w}$, and a scalar bias b (for $2d+1$ degrees of freedom). The transformation $\mathbf{y} = f_{\mathrm{NF}}(\mathbf{x})$ is defined as,

$$\mathbf{y} = \mathbf{x} + t\, \mathbf{u}, \qquad t = \tanh(\mathbf{w}^\top \mathbf{x} + b).$$

Intuitively, the the planar flow moves each point in a fixed direction $\mathbf{u}$, by an amount that depends on its location relative to a soft boundary defined by $\mathbf{w}$ and b: points near the boundary move more, and points far from it move little.

The Jacobian $J_{\mathrm{NF}}(\mathbf{x}) = \partial \mathbf{y}/\partial \mathbf{x}$ of the planar flow is,

$$J_{\mathrm{NF}}(\mathbf{x}) = \mathbf{I}_d + \delta\, \mathbf{u}\, \mathbf{w}^\top,$$

where $\delta = 1 - t^2$.

For SGA-based optimization, if $D = 1 + \delta\, \mathbf{u}^\top \mathbf{w}$, then the gradients with respect to the planar-flow parameters are,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = t\,\mathbf{g} \;+\; \frac{\delta \mathbf{w}}{D}, \tag{23}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \delta\,(\mathbf{u}^\top \mathbf{g})\,\mathbf{x} \;+\; \frac{\delta\,\mathbf{u} \;+\; (-2t\delta)\,(\mathbf{u}^\top \mathbf{w})\,\mathbf{x}}{D}, \tag{24}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \delta\,(\mathbf{u}^\top \mathbf{g}) \;+\; \frac{(-2t\delta)\,(\mathbf{u}^\top \mathbf{w})}{D}, \tag{25}$$

where, as in the previous section, $\mathbf{g} = \partial \mathcal{L} / \partial \mathbf{y}$ is the incoming gradient.

Euclidean and hyperbolic geometries

By default, VINE embeds taxa in a d-dimensional Euclidean space, where d can be selected by the user via the `--dimensionality` option (default value of 5). Let $\mathbf{y}_i \in \mathbb{R}^d$ denote the embedded point for taxon i (after normalizing flows are applied) and let $s > 1$ be a global scale factor that allows the embedding space to be expanded relative to the distance matrix $\mathbf{D}$. The Euclidean pairwise distance is simply given by,

$$D_{ij} \;=\; \frac{\|\mathbf{y}_i - \mathbf{y}_j\|}{s}, \qquad 1 \le i < j \le n.$$

The Jacobian of the mapping $\mathbf{y} \mapsto \{D_{ij}\}$ $(\partial \mathbf{D})/\partial \mathbf{y})$ has a simple block structure. For each pair (i, j),

$$\frac{\partial D_{ij}}{\partial \mathbf{y}_i} = \frac{\mathbf{y}_i - \mathbf{y}_j}{s^2\,\|\mathbf{y}_i - \mathbf{y}_j\|}, \qquad\qquad \frac{\partial D_{ij}}{\partial \mathbf{y}_j} = -\frac{\partial D_{ij}}{\partial \mathbf{y}_i},$$

with all other partial derivatives equal to zero.

The mean $\boldsymbol{\mu}$ of the MVN variational distribution is initialized by classical multidimensional scaling (MDS) applied to the starting distance matrix $\mathbf{D}^{(0)}$. Let $\mathbf{D}_0^2$ be the (symmetric) matrix of squared distances, double-centered to obtain a Gram matrix $\mathbf{G}$. An eigendecomposition $\mathbf{G} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top$ is computed, and the initial embedding is given by the top d principal components, $\boldsymbol{\mu}_i = (\sqrt{\lambda_1} v_{i1}, \ldots, \sqrt{\lambda_d} v_{id})$, rescaled by s.

In the hyperbolic setting, VINE follows the general approach outlined by Macaulay et al. [76] and embeds points on the upper sheet of the $(d{+}1)$-dimensional hyperboloid model with negative curvature $-\kappa$ (set by `--negcurvature`, default $-\kappa = 1$). Each taxon is represented by a spatial coordinate $\mathbf{y}_i \in \mathbb{R}^d$, which is lifted to a point on the hyperboloid via,

$$x_{0,i} = \sqrt{1 + \|\mathbf{y}_i\|^2}, \qquad \tilde{\mathbf{y}}_i = (x_{0,i}, \mathbf{y}_i) \in \mathbb{R}^{d+1}.$$

The hyperbolic distance between taxa i and j is,

$$D_{ij} \;=\; \frac{\alpha}{s}\,\operatorname{arcosh}\!\big(-\langle \tilde{\mathbf{y}}_i, \tilde{\mathbf{y}}_j \rangle_L\big), \qquad 1 \le i < j \le n,$$

where $\alpha = 1/\sqrt{-\kappa}$ is the curvature radius and $\langle \tilde{\mathbf{y}}_i, \tilde{\mathbf{y}}_j \rangle_L$ is the Lorentz inner product,

$$\langle \tilde{\mathbf{y}}_i, \tilde{\mathbf{y}}_j \rangle_L = -x_{0,i} x_{0,j} + \mathbf{y}_i^\top \mathbf{y}_j.$$

Defining

$$u_{ij} = x_{0,i} x_{0,j} - \mathbf{y}_i^\top \mathbf{y}_j, \qquad c_{ij} = \frac{\alpha}{s} \frac{1}{\sqrt{u_{ij}^2 - 1}},$$

the Jacobian blocks of the mapping $\mathbf{y} \mapsto \{D_{ij}\}$ are

$$\frac{\partial D_{ij}}{\partial \mathbf{y}_i} = c_{ij} \left(\frac{x_{0,j}}{x_{0,i}} \mathbf{y}_i - \mathbf{y}_j \right), \qquad \frac{\partial D_{ij}}{\partial \mathbf{y}_j} = c_{ij} \left(\frac{x_{0,i}}{x_{0,j}} \mathbf{y}_j - \mathbf{y}_i \right),$$

again with all other partial derivatives equal to zero.

For hyperbolic embeddings, the MVN mean is initialized using a spectral variant of the *hydra* algorithm [84]. Given an initial distance matrix $\mathbf{D}^{(0)}$, we form a symmetric matrix $\mathbf{A}$ such that,

$$A_{ij} = \begin{cases} 1, & i = j, \\ \cosh\!\left(\sqrt{-\kappa}\, D_{ij}^{(0)}\, s\right), & i \neq j, \end{cases}$$

diagonalize $\mathbf{A} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top$, and set the initial spatial coordinates from the leading d eigenmodes, followed by the same global scaling by s.

The scale factor s is chosen so that the median pairwise distance between points is 25 in the Euclidean case and 4 in the hyperbolic case.

Efficient calculation of branch-length gradients

Backpropagation begins with the gradient of the phylogenetic log likelihood function with respect to the branch lengths, $\nabla_{\mathbf{b}} \ell(\tau, \mathbf{b}; \mathbf{X})$. A naive approach to obtaining this gradient would require $2n - 2$ evaluations of the likelihood function, for $O(n^2)$ total time, but the gradient can be computed in linear time by taking advantage of an "inside/outside" algorithm along the tree (see, e.g., [85]). Briefly, at each site i, Felsenstein's pruning algorithm is first performed in a post-order traversal of the tree, producing quantities of the form,

$$P(L_v \,|\, v = b),$$

at all nodes v, where L_v represents the data at the leaves beneath node v and $v = b$ indicates the condition that node v has nucleotide identity (or CRISPR mutation) b. This procedure is followed by a similar algorithm that proceeds in pre-order, from root to leaves, and computes the complementary quantities,

$$P(L_{\bar{v}}, v = b),$$

where $L_{\bar{v}}$ indicates the data at all leaves *other than* the ones beneath v. The partial derivatives of interest can then be computed in a final traversal of the tree using the property that, for a node v, its parent u, and its

sibling w,

$$\frac{\partial}{\partial b_v} \log P(\mathbf{X}_i \mid \tau, \mathbf{b}) = \frac{1}{P(\mathbf{X}_i \mid \tau, \mathbf{b})} \sum_a \sum_b \sum_c p(L_{\bar{u}}, u = a)\, p(L_v \,|\, v = b)\, p(L_w \,|\, w = c) \times P(a \to c \,|\, b_w)\, \frac{\partial}{\partial b_v} P(a \to b \,|\, b_v),$$

where $\log P(\mathbf{X}_i \mid \tau, \mathbf{b})$ is the contribution to the log likelihood at site i, $P(a \to c \,|\, b_w)$ denotes the probability of an $a \to c$ substitution along a branch of length b_w and $\frac{\partial}{\partial b_v} P(a \to b \,|\, b_v)$ denotes the partial derivative with respect to the branch length of the probability of an $a \to b$ substitution along a branch of length b_v, which can be calculated analytically in a manner dependent on the assumed substitution model. These terms are then summed across all sites i for each branch. Their computation is carried out together with the log likelihood calculation itself for minimal overhead. Note that dynamic rescaling of the inside and outside quantities is needed to avoid numerical underflow on large trees.

Stochastic gradient ascent

For stochastic gradient ascent (SGA), VINE uses a customized implementation in C of the Adam algorithm [86]. In the Monte-Carlo case, the algorithm draws a minibatch of M sets of embedded points on each iteration, $\mathbf{x}_1, \ldots, \mathbf{x}_M$ (transformed from $\mathbf{z}_1, \ldots, \mathbf{z}_M$), and uses them to obtain estimates of the ELBO and its gradient. For each minibatch, it precomputes $\mathrm{KL}(q(\mathbf{x}) \,\|\, p(\mathbf{x}))$ and its gradient, which do not depend on the sampled points. It then updates all parameters based on the gradient in the standard manner. Command-line options allow the user to control the the minibatch size (`--batchsize`; default 10) and the learning rate α (`--learnrate`; default 0.05), as well as convergence criteria for the algorithm (e.g., minimum number of minibatches, `--min-nbatches`, default 200, and number of minibatches over which to average when assessing convergence, `--nbatches-conv`, default 50). In the case of the Taylor approximation, the procedure is similar but no minibatch sampling is required; the ELBO is estimated directly from the value of the log likelihood at the current mean $\boldsymbol{\mu}$ and the Hutchinson-based estimate of the trace term (equation 15). In both cases, the Adam parameters β_1 and β_2 are both fixed at 0.9. A simple scheduler is employed to manage subsampling of sites in the alignment in early iterations of the algorithm, adaptive clipping of unusually large gradients, and gradual increases of the learning rate to its target value. The algorithm also allows for a separate set of "nuisance parameters" (including parameters for substitution rates, tree priors, and the normalizing flows), which are also optimized by SGA but are not encoded in the continuous embedding.

The SGA algorithm was carefully tuned to optimize the speed and accuracy of convergence over data sets of various sizes. During this process, we found that it was quite sensitive to the scale of the embedding space, sometimes struggling to converge when distances between points grew close to zero. Therefore, we introduced the scaling factor s that allows for a difference in scale between the embedding space and the distances used for tree reconstruction (as detailed above). In particular, VINE rescales the MVN distribution for embedded points so that median distances between points are 25 and 4, respectively, for the Euclidean and hyperbolic geometries.

Mutation models

VINE includes implementations of the Jukes-Cantor (JC) [65], HKY [66], and general time reversible (GTR) [67] DNA substitution models, as well as the CRISPR-barcode mutation model developed by Seidel et al. [61] and extended by Chu et al. [58]. For the DNA models, the substitution rate matrix $\mathbf{Q}$ is normalized in the standard manner, such that $\sum_{i \neq j} \pi_i q_{ij} = 1$, where π_i is the equilibrium frequency of nucleotide i under the model, so that branch lengths can be interpreted as having units of expected substitutions per site. For the HKY and GTR models, equilibrium frequencies for the four nucleotides are simply estimated using the relative frequencies in the alignment, whereas under the JC model they are assumed to be uniform. As a result, the JC model has no free substitution rate parameters, the HKY model has a single free parameter—the transition/transversion rate ratio κ—and the GTR model has five free parameters (after accounting for rescaling). These are considered as "nuisance" parameters that are optimized in SGA but not considered for full variational Bayes characterization. The CRISPR model has a single free parameter, corresponding to the silencing rate for barcodes, which is also treated as a nuisance parameter. VINE supports two parameterizations for the barcode-editing rate matrix: a single rate matrix shared by all sites (as in TiDeTree [61]; `--crispr-modtype GLOBAL`) or a separate rate matrix per site (as in LAML [58]; `--crispr-modtype SITEWISE`). It also allows for either a uniform distribution for all mutation-rate priors (`--crispr-mutprior UNIF`) or a prior that reflects the relative frequencies of mutations in the input matrix (`--crispr-mutprior EMPIRICAL`). The CRISPR mutation matrix is subjected to the same scaling constraint as the DNA substitution matrix, so the estimated branch lengths can be interpreted in expected mutations per site. For comparison, LAML trees must be scaled by the separately estimated mutation rate.

Priors for trees

By default, VINE assumes a prior in the embedding space of $p(\mathbf{x}) = \mathrm{MVN}(0, \mathbf{I})$, which, as discussed above, is typical for variational autoencoders. The program also allows specification, however, of more informative priors for phylogenies. If these are selected, the KLD can no longer be evaluated in closed form and the prior must be incorporated into Monte Carlo estimation (or Taylor approximation) along with the log likelihood (see equation 7). In particular, the `--treeprior` option allows for either Yule prior over tree topologies (integrating over birth rates) or a Gamma model for total branch length; and the `--relclock` option allows for a relaxed local clock with lognormal clock rates per branch. In these cases, any free parameters of the priors are also considered nuisance parameters during optimization Evaluation of priors was not a focus of this study, however, and all of our results reflect the default MVN prior.

Simulations

For our simulations of DNA alignments, we used a script based on DendroPy [87] to simulate trees under a birth–death process (birth rate 1.0, death rate 0.5), with three-fold oversampling of tips followed by pruning and rescaling, to produce tree heights of 1 substitution per site and minimum branch lengths of 0.02. Branch-

specific rates were drawn independently under an uncorrelated log-normal relaxed clock with mean of 1 and standard deviation of 0.6. Nucleotide alignments were then generated for each tree using `base_evolve` from PHAST under an HKY model with $\kappa = 4$ and equilibrium frequencies of $\pi_A = \pi_T = 0.3$ and $\pi_C = \pi_G = 0.2$. As noted in the text, we generated alignments of 300 bp and 10,000 bp, for various numbers of taxa n between 10 and 100. For held-out data, we ran `base_evolve` a second time on the same simulated trees used to generate the training data.

For CRISPR lineage-tracing simulations, we used Cassiopeia's [53] BirthDeathFitnessSimulator with birth rate 0.075 and death rate 0.005, drawing branch waiting times from the absolute value of a normal distribution (μ =1/rate, $\sigma = \mu/5$) to obtain approximately balanced ultrametric trees, which were post-scaled to a fixed height of 54 days to match the data set of ref. [73]. We then applied Cas9LineageTracingDataSimulator using a cassette of 3 CRISPR target sites repeated 10 times (30 total sites), a mutation rate of 0.01, a heritable silencing rate of 1×10^{-4}, and no stochastic silencing. Edit outcomes were restricted to a predefined library of 100 possible states with fixed probabilities, ensuring reproducible mutation signatures across simulations.

Convergence criteria for MCMC

For convergence monitoring of MCMC-based methods, we applied a uniform procedure across all analyses for both simulated and real data. First, for each combination of taxa number, sequence length, substitution model, and program (BEAST 2 or MrBayes), we ran three pilot MCMC replicates with two chains per replicate and intentionally long chain lengths, recording all samples (no thinning). For each replicate, we then retrospectively computed the rank-normalized split $\hat{R}$ [88] and the effective sample size (ESS) for each of several scalar summaries in each chain in intervals of at least 100,000 iterations. For BEAST 2, the selected statistics were the log posterior, log likelihood, tree height, and tree length, and for MrBayes they were the log likelihood and tree length. For each replicate, we then identified the earliest MCMC iteration calculation at which all parameters met ESS $\geq$ 400 and $\hat{R} \leq 1.01$. (For the large alignments—with 10,000 columns—we had to relax the $\hat{R}$ criterion in a few cases because not all pilot replicates converged.) ESS calculations were based on the implementation in Tracer [89], reimplemented in our own scripts. Finally, we averaged these convergence points across the three replicates to determine the target chain length. These target chain lengths were then applied uniformly across all ten replicates for each data set size, model, and program. Notably, the chain lengths required for our convergence and sampling criteria were generally substantially shorter for MrBayes than for BEAST 2, and this was the dominant factor in the differences in running times we observed. Note also that we experimented with a more stringent requirement of ESS $\geq$ 625 but found it to require excessively long chains in some cases, so we relaxed the threshold to 400, as recommended in ref. [77].

Models selected for analysis of DNA

We selected models in BEAST 2 (v2.7.7) and MrBayes (3.2.7a) that were as close as possible to the ones in VINE given choices available for each program, generally keeping the prior distributions diffuse so that the log likelihood dominated in inference. In particular, in BEAST 2, we used a Yule prior for the phylogeny with a single free birth-rate parameter (uniform prior) and an uncorrelated log-normal relaxed clock (exponential prior with mean 2 on ucldStdev, the standard deviation in log-space). For the κ parameter in the HKY model, we used a log-normal prior with a log mean of 1.386 (corresponding to $\kappa = 4$, the value used in our simulations) and a standard deviation of 0.1. Similarly in MrBayes, we used a uniform prior over tree topologies and the default Gamma-Dirichlet prior for branch lengths. For the HKY model, we used a Beta(4,1) prior distribution for the rate of transitions relative to the rate of all substitutions. In both programs, we did not allow for variation across sites in rates, we used random starting trees, and we used empirical frequencies for the four nucleotides (for HKY).

In VINE we used an implicit prior on trees corresponding to a standard multivariate normal distribution for $p(\mathbf{x})$ (the default, as discussed above). The HKY parameter κ was optimized as a nuisance parameter in SGA (`--hky85` option). Empirical nucleotide frequencies were used. Analyses that required sampled trees (such as the comparison of Robinson-Foulds distances) were based on 1000 samples from the approximate posterior following convergence of the ELBO (`-s 1000`). In most cases, default parameters were used for minibatch size (`-b 10`), the minimum number of minibatches before convergence (`-M 200`), and the number of minibatches over which to average when assessing convergence (`-c 50`), although for some larger trees we used larger values for the minimum number of minibatches.

Running other programs

We ran VaiPhy with the arguments `--max_iter 200` and `--n_particles 128` as the authors used for their benchmarking experiments [41]. For GeoPhy and VBPI-GNN, we used scripts conveniently provided by Zhou et al. [90]. For GeoPhy we used commands such as,

```
python ./scripts/run_gp.py -v -ip ds-data/ds1/DS1.nex -op results/ds1/ds1_paper
-c ./config/default.yaml -s 0 -ua -es lorentz -edt full -ed 4 -eqs 1e-1 -ast
100_1000 -mcs 3 -ci 1000 -ms 1000_000 -ul
```

which is identical to that reported in ref. [90] except that we reduced the number of steps (`-ms`) from 1,000,000 to 100,000, which appeared to be adequate for convergence in our case. For VBPI-GNN, we used commands such as,

```
python main.py -dataset DS1 -brlen_model gnn -gnn_type edge -hL 2 -hdim 100
-maxIter 400000 -empFreq -psp!
```

We followed Zhou et al. [90] in using IQ-TREE2 [19] for pre-generation of trees, using commands such as:

```
iqtree -s DS1 -bb 10000 -wbt -m JC69 -redo
```

We ran LAML (v1.0.5) with commands such as,

```
run_laml -c quinn.31.csv -t quinn.31.cass.nwk -o quinn.31 -topology_search
-noDropout
```

where `quinn.31.cass.nwk` is a starting tree produced using the Cassiopeia-Greedy algorithm:

```
python cassiopeiaGreedy.py quinn.31.csv quinn.31.cass.nwk
```

All computational experiments were run on a ∼6 yr-old HPE ProLiant DL380 Gen10 server equipped with dual Intel Xeon Gold 5120 processors (28 physical cores total, 56 hardware threads) and 192 GB of RAM, running Rocky Linux 8.10. No GPU acceleration was used.

Software implementation

VINE is written in C and available from github (https://github.com/CshlSiepelLab/vine) under a standard BSD 3-Clause License. It requires a recent installation of PHAST [91] (https://github.com/CshlSiepelLab/phast), from which it borrows DNA substitution models, likelihood calculations, and alignment-handling routines. On MacOS, VINE can be installed easily using the commands `brew tap CshlSiepelLab/tools` and `brew install vine` (PHAST will be automatically installed as a dependency.) Work is in progress to provide easy-to-install distributions for other platforms.

Funding

This work was supported by US National Institutes of Health (NIH) National Institute of General Medical Sciences Grant R35-GM127070 and National Cancer Institute (NCI) Grants R01-CA272466 and 5P30-CA045508, as well as Starr Cancer Consortium Grant I16-0060, a National Science Foundation Graduate Research Fellowship (to S.J.S.), a Starr Centennial Scholarship (to S.J.S.), and the Simons Center for Quantitative Biology at CSHL. The content is solely the responsibility of the authors and does not necessarily represent the official views of the US National Institutes of Health.

Conflict of Interest

The authors declare no competing interests.

Acknowledgments

We thank other members of the community at Cold Spring Harbor Laboratory as well as Dawid Nowak and his team at Weill Cornell Medicine for helpful feedback.

References

[1] Edwards, A. W. F. & Cavalli-Sforza, L. L. Reconstruction of evolutionary trees. In Heywood, V. & McNeill, J. (eds.) *Phenetic and Phylogenetic Classification*, 67–76 (Systematics Association, 1964).

[2] Camin, J. H. & Sokal, R. R. A method for deducing branching sequences in phylogeny. *Evolution* **19**, 311–326 (1965). URL http://dx.doi.org/10.1111/j.1558-5646.1965.tb01722.x.

[3] Eck, R. V. & Dayhoff, M. O. *Atlas of Protein Sequence and Structure* (Natl. Biomed. Res. Found., 1966).

[4] Fitch, W. M. & Margoliash, E. Construction of phylogenetic trees. *Science* **155**, 279–284 (1967). URL http://dx.doi.org/10.1126/science.155.3760.279.

[5] Cavalli-Sforza, L. L. & Edwards, A. W. F. Phylogenetic analysis: models and estimation procedures. *Evolution* **32**, 550–570 (1967).

[6] Dewar, A. E., Belcher, L. J. & West, S. A. A phylogenetic approach to comparative genomics. *Nat Rev Genet* **26**, 395–405 (2025). URL http://dx.doi.org/10.1038/s41576-024-00803-0.

[7] Nielsen, R., Vaughn, A. H. & Deng, Y. Inference and applications of ancestral recombination graphs. *Nat Rev Genet* **26**, 47–58 (2024). URL http://dx.doi.org/10.1038/s41576-024-00772-4.

[8] Attwood, S. W., Hill, S. C., Aanensen, D. M., Connor, T. R. & Pybus, O. G. Phylogenetic and phylodynamic approaches to understanding and combating the early SARS-CoV-2 pandemic. *Nat Rev Genet* **23**, 547–562 (2022). URL http://dx.doi.org/10.1038/s41576-022-00483-8.

[9] Felsenstein, J. The troubled growth of statistical phylogenetics. *Syst Biol* **50**, 465–467 (2001). URL http://dx.doi.org/10.1080/10635150119297.

[10] Felsenstein, J. Maximum likelihood and minimum-steps methods for estimating evolutionary trees from data on discrete characters. *Syst Biol* **22**, 240–249 (1973).

[11] Felsenstein, J. Evolutionary trees from DNA sequences: a maximum likelihood approach. *J Mol Evol* **17**, 368–376 (1981). URL http://dx.doi.org/10.1007/bf01734359.

[12] Felsenstein, J. PHYLIP – Phylogeny Inference Package (Version 3.2). *Cladistics* **5**, 164–166 (1989).

[13] Felsenstein, J. PHYLIP (Phylogeny Inference Package) version 3.5c (1993).

[14] Yang, Z. PAML: a program package for phylogenetic analysis by maximum likelihood. *Bioinformatics* **13**, 555–556 (1997). URL http://dx.doi.org/10.1093/bioinformatics/13.5.555.

[15] Guindon, S. & Gascuel, O. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Syst Biol* **52**, 696–704 (2003).

[16] Pond, S. L. K., Frost, S. D. W. & Muse, S. V. HyPhy: Hypothesis testing using phylogenies. *Bioinformatics* **21**, 676–679 (2004). URL http://dx.doi.org/10.1093/bioinformatics/bti079.

[17] Stamatakis, A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* **30**, 1312–1313 (2014). URL http://dx.doi.org/10.1093/bioinformatics/btu033.

[18] Nguyen, L.-T., Schmidt, H. A., von Haeseler, A. & Minh, B. Q. IQ-TREE: a fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Mol Biol Evol* **32**, 268–274 (2014). URL http://dx.doi.org/10.1093/molbev/msu300.

[19] Minh, B. Q. *et al.* IQ-TREE 2: New models and efficient methods for phylogenetic inference in the genomic era. *Mol Biol Evol* **37**, 1530–1534 (2020). URL http://dx.doi.org/10.1093/molbev/msaa015.

[20] Rannala, B. & Yang, Z. Probability distribution of molecular evolutionary trees: A new method of phylogenetic inference. *J Mol Evol* **43**, 304–311 (1996).

[21] Mau, B., Newton, M. A. & Larget, B. Bayesian phylogenetic inference via Markov chain Monte Carlo methods. *Biometrics* **55**, 1–12 (1999).

[22] Larget, B. & Simon, D. L. Markov chain Monte Carlo algorithms for the Bayesian analysis of phylogenetic trees. *Mol Biol Evol* **16**, 750–759 (1999).

[23] Felsenstein, J. *Inferring Phylogenies* (Sinauer Associates, 2004).

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

[25] Ronquist, F. *et al.* MrBayes 3.2: Efficient Bayesian phylogenetic inference and model choice across a large model space. *Syst Biol* **61**, 539–542 (2012). URL http://dx.doi.org/10.1093/sysbio/sys029.

[26] Drummond, A. J. & Rambaut, A. BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evol Biol* **7**, 214 (2007). URL http://dx.doi.org/10.1186/1471-2148-7-214.

[27] Bouckaert, R. *et al.* BEAST 2: a software platform for Bayesian evolutionary analysis. *PLoS Comput Biol* **10**, e1003537 (2014). URL http://dx.doi.org/10.1371/journal.pcbi.1003537.

[28] Bouckaert, R. *et al.* BEAST 2.5: An advanced software platform for Bayesian evolutionary analysis. *PLoS Comput Biol* **15**, e1006650 (2019).

[29] Höhna, S. *et al.* RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Syst Biol* **65**, 726–736 (2016). URL http://dx.doi.org/10.1093/sysbio/syw021.

[30] Fisher, A. A. *et al.* Scalable Bayesian phylogenetics. *Phil Trans R Soc B* **377**, 20210242 (2022). URL http://dx.doi.org/10.1098/rstb.2021.0242.

[31] Jordan, M. I., Ghahramani, Z., Jaakkola, T. S. & Saul, L. K. An introduction to variational methods for graphical models. *Machine Learning* **37**, 183–233 (1999).

[32] Wainwright, M. J., Jordan, M. I. *et al.* Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning* **1**, 1–305 (2008).

[33] Blei, D. M., Kucukelbir, A. & McAuliffe, J. D. Variational inference: a review for statisticians. *J Am Stat Assoc* **112**, 859–877 (2017). URL http://dx.doi.org/10.1080/01621459.2017.1285773.

[34] Zhang, C. & Matsen IV, F. A. Variational Bayesian phylogenetic inference. In *International Conference on Learning Representations* (2019). URL https://openreview.net/forum?id=SJVmjjR9FX.

[35] Dang, T. & Kishino, H. Stochastic variational inference for Bayesian phylogenetics: a case of CAT model. *Mol Biol Evol* **36**, 825–833 (2019).

[36] Fourment, M. & Darling, A. E. Evaluating probabilistic programming and fast variational Bayesian inference in phylogenetics. *PeerJ* **7**, e8272 (2019).

[37] Zhang, C. Improved variational Bayesian phylogenetic inference with normalizing flows. *arXiv* (2020). URL https://arxiv.org/abs/2012.00459.

[38] Zhang, C. & Matsen, F. A. A variational approach to Bayesian phylogenetic inference. *arXiv* (2022). URL https://arxiv.org/abs/2204.07747.

[39] Zhang, C. Learnable topological features for phylogenetic inference via graph neural networks. *arXiv* (2023). URL https://arxiv.org/abs/2302.08840.

[40] Xie, T., Matsen, F. A., Suchard, M. A. & Zhang, C. Variational Bayesian phylogenetic inference with semi-implicit branch length distributions. *arXiv* (2024). URL https://arxiv.org/abs/2408.05058.

[41] Koptagel, H., Kviman, O., Melin, H., Safinianaini, N. & Lagergren, J. VaiPhy: a variational inference based algorithm for phylogeny. *arXiv* (2022). URL https://arxiv.org/abs/2203.01121.

[42] Kviman, O., Molén, R. & Lagergren, J. Improved variational Bayesian phylogenetic inference using mixtures. *arXiv* (2023). URL https://arxiv.org/abs/2310.00941.

[43] Mimori, T. & Hamada, M. GeoPhy: Differentiable phylogenetic inference via geometric gradients of tree topologies. *arXiv* (2023). URL https://arxiv.org/abs/2307.03675.

[44] Macaulay, M. & Fourment, M. Differentiable phylogenetics via hyperbolic embeddings with Dodonaphy. *Bioinf Adv* **4**, vbae082 (2024).

[45] Bouckaert, R. R. Variational Bayesian phylogenies through matrix representation of tree space. *PeerJ* **12**, e17276 (2024). URL http://dx.doi.org/10.7717/peerj.17276.

[46] Hotti, A., Kviman, O., Molén, R., Elvira, V. & Lagergren, J. Efficient mixture learning in black-box variational inference. *arXiv* (2024). URL https://arxiv.org/abs/2406.07083.

[47] Chen, A. *et al.* Variational combinatorial sequential Monte Carlo for Bayesian phylogenetics in hyperbolic space. *arXiv* (2025). URL https://arxiv.org/abs/2501.17965.

[48] McKenna, A. *et al.* Whole-organism lineage tracing by combinatorial and cumulative genome editing. *Science* **353**, aaf7907 (2016).

[49] Frieda, K. L. *et al.* Synthetic recording and in situ readout of lineage information in single cells. *Nature* **541**, 107–111 (2017).

[50] Raj, B. *et al.* Simultaneous single-cell profiling of lineages and cell types in the vertebrate brain. *Nat Biotech* **36**, 442–450 (2018).

[51] Jiang, J. *et al.* scLTdb: a comprehensive single-cell lineage tracing database. *Nuc Acids Res* **53**, D1173–D1185 (2024).

[52] Askary, A. *et al.* The lives of cells, recorded. *Nat Rev Genet* 1–20 (2024).

[53] Jones, M. G. *et al.* Inference of single-cell phylogenies from lineage tracing data using Cassiopeia. *Genome Biol* **21**, 92 (2020).

[54] Feng, J. *et al.* Estimation of cell lineage trees by maximum-likelihood phylogenetics. *Ann Appl Stat* **15**, 343–362 (2021).

[55] Sashittal, P., Schmidt, H., Chan, M. & Raphael, B. J. Startle: A star homoplasy approach for CRISPR-Cas9 lineage tracing. *Cell Syst* **14**, 1113–1121 (2023).

[56] Prusokiene, A., Prusokas, A. & Retkute, R. Machine learning based lineage tree reconstruction improved with knowledge of higher level relationships between cells and genomic barcodes. *NAR Genom Bioinform* **5**, lqad077 (2023).

[57] Dai, J. & Molloy, E. K. StarCDP: Dynamic programming algorithms for fast and accurate cell lineage tree reconstruction from CRISPR-based lineage tracing data. *J Comput Biol* (2025). URL http://dx.doi.org/10.1177/15578666251386082.

[58] Chu, G., Mai, U., Schmidt, H. *et al.* Maximum likelihood inference of time-scaled cell lineage trees with mixed-type missing data using LAML. *Genome Biol* **26**, 189 (2025).

[59] Gong, W. *et al.* Benchmarked approaches for reconstruction of in vitro cell lineages and *in silico* models of *C. elegans* and *M. musculus* developmental trees. *Cell Syst* **12**, 810–826 (2021).

[60] Staklinski, S. J. *et al.* Bayesian inference of tissue-migration histories in metastatic cancer from cell-lineage tracing data. *bioRxiv* (2025). URL http://dx.doi.org/10.1101/2025.09.09.668633.

[61] Seidel, S. & Stadler, T. TiDeTree: a Bayesian phylogenetic framework to estimate single-cell trees and population dynamic parameters from genetic lineage tracing data. *Proc Biol Sci* **289**, 20221844 (2022).

[62] Seidel, S. *et al.* SciPhy: A Bayesian phylogenetic framework using sequential genetic lineage tracing data. *bioRxiv* (2024). URL http://dx.doi.org/10.1101/2024.10.01.615771.

[63] Zwaans, A., Seidel, S., Manceau, M. & Stadler, T. A Bayesian phylodynamic inference framework for single-cell CRISPR/Cas9 lineage tracing barcode data with dependent target sites. *Phil Trans R Soc B* **380**, 20230318 (2025). URL http://dx.doi.org/10.1098/rstb.2023.0318.

[64] Saitou, N. & Nei, M. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *J Mol Biol* **4**, 406–425 (1987).

[65] Jukes, T. H. & Cantor, C. R. Evolution of protein molecules. In Munro, H. (ed.) *Mammalian Protein Metabolism*, 21–132 (Academic Press, 1969).

[66] Hasegawa, M., Kishino, H. & Yano, T. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. *J Mol Evol* **22**, 160–174 (1985).

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

[68] Kingma, D. P. & Welling, M. Auto-encoding variational Bayes. *arXiv* (2013). URL https://arxiv.org/abs/1312.6114.

[69] Doersch, C. Tutorial on variational autoencoders. *arXiv* (2016). URL https://arxiv.org/abs/1606.05908.

[70] Robinson, D. & Foulds, L. Comparison of phylogenetic trees. *Mathematical Biosciences* **53**, 131–147 (1981). URL http://dx.doi.org/10.1016/0025-5564(81)90043-2.

[71] Prillo, S., Ravoor, A., Yosef, N. & Song, Y. S. ConvexML: Fast and accurate branch length estimation under irreversible mutation models, illustrated through applications to CRISPR/Cas9-based lineage tracing. *Syst Biol* syaf054 (2025). URL http://dx.doi.org/10.1093/sysbio/syaf054.

[72] Sokal, R. R. & Michener, C. D. A statistical method for evaluating systematic relationships. *Univ Kans Sci Bull* **38**, 1409–1438 (1958).

[73] Quinn, J. J. *et al.* Single-cell lineages reveal the rates, routes, and drivers of metastasis in cancer xenografts. *Science* **371** (2021).

[74] Zhang, C., Bütepage, J., Kjellström, H. & Mandt, S. Advances in variational inference. *IEEE Trans Pattern Anal Mach Intell* **41**, 2008–2026 (2018).

[75] Lakner, C., van der Mark, P., Huelsenbeck, J. P., Larget, B. & Ronquist, F. Efficiency of Markov chain Monte Carlo tree proposals in Bayesian phylogenetics. *Syst Biol* **57**, 86–103 (2008). URL http://dx.doi.org/10.1080/10635150801886156.

[76] Macaulay, M., Darling, A. & Fourment, M. Fidelity of hyperbolic space for Bayesian phylogenetic inference. *PLoS Comput Biol* **19**, e1011084 (2023). URL http://dx.doi.org/10.1371/journal.pcbi.1011084.

[77] Fabreti, L. G. & Höhna, S. Convergence assessment for Bayesian phylogenetic analysis using MCMC simulation. *Methods in Ecology and Evolution* **13**, 77–90 (2021). URL http://dx.doi.org/10.1111/2041-210X.13727.

[78] Ayres, D. L. *et al.* BEAGLE 3: Improved performance, scaling, and usability for a high-performance computing library for statistical phylogenetics. *Syst Biol* **68**, 1052–1061 (2019). URL http://dx.doi.org/10.1093/sysbio/syz020.

[79] Blassel, L., Boussau, B., Lartillot, N. & Jacob, L. Likelihood-free inference of phylogenetic tree posterior distributions. *arXiv* (2025). URL https://arxiv.org/abs/2510.12976.

[80] Li, W.-H. *Molecular Evolution* (Sinauer Associates, Inc., 1997).

[81] Elias, I. & Lagergren, J. Fast neighbor joining. *Theor Comput Sci* **410**, 1993–2000 (2009). URL http://dx.doi.org/10.1016/j.tcs.2008.12.040.

[82] Hutchinson, M. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Commun Stats—Simulation and Computation* **19**, 433–450 (1990). URL http://dx.doi.org/10.1080/03610919008812866.

[83] Pearlmutter, B. A. Fast exact multiplication by the Hessian. *Neural Computation* **6**, 147–160 (1994). URL http://dx.doi.org/10.1162/neco.1994.6.1.147.

[84] Keller-Ressel, M. & Nargang, S. Hydra: a method for strain-minimizing hyperbolic embedding of network- and distance-based data. *Journal of Complex Networks* **8**, cnaa002 (2020). URL http://dx.doi.org/10.1093/comnet/cnaa002.

[85] Siepel, A. & Haussler, D. Phylogenetic estimation of context-dependent substitution rates by maximum likelihood. *Mol Biol Evol* **21**, 468–488 (2004). URL http://dx.doi.org/10.1093/molbev/msh039.

[86] Kingma, D. & Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)* (2015).

[87] Moreno, M. A., Holder, M. T. & Sukumaran, J. DendroPy 5: a mature python library for phylogenetic computing. *Journal of Open Source Software* **9**, 6943 (2024). URL http://dx.doi.org/10.21105/joss.06943.

[88] Vehtari, A., Gelman, A., Simpson, D., Carpenter, B. & Bürkner, P.-C. Rank-normalization, folding, and localization: an improved $\hat{R}$ for assessing convergence of MCMC (with discussion). *Bayesian Analysis* **16** (2021). URL http://dx.doi.org/10.1214/20-BA1221.

[89] Rambaut, A., Drummond, A. J., Xie, D., Baele, G. & Suchard, M. A. Posterior summarization in Bayesian phylogenetics using Tracer 1.7. *Syst Biol* **67**, 901–904 (2018). URL http://dx.doi.org/10.1093/sysbio/syy032.

[90] Zhou, M. *et al.* PhyloGFN: Phylogenetic inference with generative flow networks. *arXiv* (2023). URL https://arxiv.org/abs/2310.08774.

[91] Hubisz, M. J., Pollard, K. S. & Siepel, A. PHAST and RPHAST: Phylogenetic analysis with space/time models. *Brief Bioinf* **12**, 41–51 (2011).