APPLES: Scalable Distance-based Phylogenetic Placement with or without Alignments

Metin Balaban[1], Shahab Sarmashghi[2], and Siavash Mirarab,[2,*]

[1] *Bioinformatics and Systems Biology Graduate Program, UC San Diego, CA 92093, USA*
[2] *Department of Electrical and Computer Engineering, UC San Diego, CA 92093, USA*

**E-mail: smirarab@ucsd.edu.*

Abstract

Placing a new species on an existing phylogeny has increasing relevance to several applications. Placement can be used to update phylogenies in a scalable fashion and can help identify unknown query samples using (meta-)barcoding, skimming, or metagenomic data. Maximum likelihood (ML) methods of phylogenetic placement exist, but these methods are not scalable to reference trees with many thousands of leaves, limiting their ability to enjoy benefits of dense taxon sampling in modern reference libraries. They also rely on *assembled* sequences for the reference set and aligned sequences for the query. Thus, ML methods cannot analyze datasets where the reference consists of unassembled reads, a scenario relevant to emerging applications of genome-skimming for sample identification. We introduce APPLES, a distance-based method for phylogenetic placement. Compared to ML, APPLES is an order of magnitude faster and more memory efficient, and unlike ML, it is able to place on large backbone trees (tested for up to 200,000 leaves). We show that using dense references improves accuracy substantially so that APPLES on dense trees is more accurate than ML on sparser trees, where it can run. Finally, APPLES can accurately identify samples without assembled reference or aligned queries using kmer-based distances, a scenario that ML cannot handle. APPLES is available publically at `github.com/balabanmetin/apples`.

Key words: Phylogenetic placement, Distance-based methods, Genome-skimming.

Phylogenetic placement is the problem of finding the optimal position for a new *query* species on an existing *backbone* (or, reference) tree. Placement, as opposed to a *de novo* reconstruction of the full phylogeny, has two advantages. In some applications (discussed below), placement is all that is needed, and in terms of accuracy, it is as good as, and even better than (Janssen *et al.*, 2018), *de novo* reconstruction. Moreover, placement can be more scalable than *de novo* reconstruction when dealing with very large trees.

Earlier research on placement was motivated by scalability. For example, placement is used in greedy algorithms that start with an empty tree and add sequences sequentially (e.g., Felsenstein, 1981; Desper and Gascuel, 2002). Each placement requires polynomial (often linear) time with respect to the size of the backbone, and thus, these greedy algorithms are scalable (often requiring quadratic time). Despite computational challenges (Warnow, 2017), there has been much progress in the *de novo* reconstruction of ultra-large trees (e.g., thousands to millions of sequences) using both maximum likelihood (ML) (e.g., Price *et al.*, 2010; Nguyen *et al.*, 2015) and the distance-based (e.g., Lefort *et al.*, 2015) approaches. However, these large-scale reconstructions require significant resources. As new sequences continually become available, placement can be used to update existing trees without repeating previous computations on full dataset.

More recently, placement has found a new application in sample identification: given one or more *query* sequences of unknown origins, detect the identity of the (set of) organism(s) that could have generated that sequence. These identifications can be made easily using sequence matching tools such as BLAST (Altschul *et al.*, 1990) when the query either exactly matches or is very close to a sequence in the reference library. However, when the sequence is novel (i.e., has lowered similarity to known sequences in the reference), this *closest* match approach is not sufficiently accurate (Koski and Golding, 2001), leading some researchers to adopt a phylogenetic approach (Sunagawa *et al.*, 2013;

Nguyen *et al.*, 2014). Sample identification is essential to the study of mixed environmental samples, especially of the microbiome, both using 16S profiling (e.g., Gill *et al.*, 2006; Krause *et al.*, 2008) and metagenomics (e.g., von Mering *et al.*, 2007). It is also relevant to barcoding (Hebert *et al.*, 2003) and meta-barcoding (Clarke *et al.*, 2014; Bush *et al.*, 2017) and quantification of biodiversity (e.g., Findley *et al.*, 2013). Driven by applications to microbiome profiling, placement tools like pplacer (Matsen *et al.*, 2010) and EPA(-ng) (Berger *et al.*, 2011; Barbera *et al.*, 2019) have been developed. Researchers have also developed methods for aligning query sequence (e.g., Berger and Stamatakis, 2011; Mirarab *et al.*, 2012) and for downstream steps (e.g., Stark *et al.*, 2010; Matsen and Evans, 2013). These publications have made a strong case that for sample identification, placement is sufficient (i.e., *de novo* is not needed). Moreover, some studies (e.g., Janssen *et al.*, 2018) have shown that when dealing with fragmentary reads typically found in microbiome samples, placement can be *more* accurate than *de novo* construction and can lead to improved associations of microbiome with clinical information.

Existing phylogenetic placement methods have focused on the ML inference of the best placement – a successful approach, which nevertheless, suffers from two shortcomings. On the one hand, ML can only be applied when the reference species are *assembled* into full-length sequences (e.g., an entire gene) and are *aligned*; however, in new applications that we will describe, assembling (and hence aligning) the reference set is not possible. On the other hand, ML, while somewhat scalable, is still computationally demanding, especially in memory usage, and cannot place on backbone trees with many thousands of leaves. As the density of reference substantially impacts the accuracy and resolution of placement, this inability to use ultra-large trees as backbone also limits accuracy. This limitation has motivated alternative methods using local sensitive hashing (Brown and Truszkowski, 2013) and divide-and-conquer (Mirarab *et al.*, 2012).

Assembly-free and alignment-free sample identification using genome-skimming (Dodsworth, 2015) can also benefit from phylogenetic placement. A genome-skim is a

shutgun sample of the genome sequenced at low coverage (e.g., 1X) – so low that assembling the nuclear genome is not possible (though, mitochondrial or plastid genomes can often be assembled). Genome-skimming promises to replace traditional marker-based barcoding of biological samples (Coissac *et al.*, 2016) but limiting analyses to organelle genome can limit resolution. Moreover, mapping reads to reference genomes is also possible only for species that have been assembled, which is a small fraction of the biodiversity on Earth. Sarmashghi *et al.* (2019) have recently shown that using shared k-mers, the distance between two unassembled genome-skims with low coverage can be accurately estimated. This approach, unlike assembling organelle genomes, uses data from the entire nuclear genome and hence promises to provide a higher resolution (e.g., at species or sub-species levels) while keeping the low sequencing cost. However, ML and other methods that require assembled sequences cannot analyze genome-skims, where both the reference and the query species are unassembled genome-wide bags of reads.

Distance-based approaches to phylogenetics are well-studied, but no existing tool can perform distance-based placement of a query sequence on a given backbone. The distance-based approach promises to solve both shortcomings of ML methods. Distance-based methods are computationally efficient and do not require assemblies. They only need distances (however computed). Thus, they can take as input assembly-free estimates of genomic distance estimated from low coverage genome-skims using Skmer (Sarmashghi *et al.*, 2019) or other alternatives (Haubold, 2014; Leimeister *et al.*, 2017; Yi and Jin, 2013; Benoit *et al.*, 2016; Fan *et al.*, 2015; Ondov *et al.*, 2016; Jain *et al.*, 2018). While alignment-based phylogenetics has been traditionally more accurate than alignment-free methods when both methods are possible, in these new scenarios, only alignment-free methods are applicable.

Here, we introduce a new method for distance-based phylogenetic placement called APPLES (Accurate Phylogenetic Placement using LEast Squares). APPLES uses dynamic programming to find the optimal distance-based placement of a sequence with running

time and memory usage that scale linearly with the size of the backbone tree. We test APPLES in simulations and on real data, both for alignment-free and aligned scenarios.

Materials and Methods

Problem Statement.

Notations. Let an unrooted tree $T = (V, E)$ be a weighted connected acyclic undirected graph with leaves denoted by $\mathcal{L} = \{1 \cdots n\}$. We let T^* be the rooting of T on a leaf 1 obtained by directing all edges away from 1. For node $u \in V$, let $p(u)$ denote its parent, $c(u)$ denote its set of children, $sib(u)$ denote its siblings, and $g(u)$ denote the set of leaves at or below u (i.e., those that have u on their path to the root), all with respect to T^*. Also let $l(u)$ denote the length of the edge $(p(u), u)$.

Distances. The tree T defines an $n \times n$ matrix where each entry $d_{ij}(T)$ corresponds to the path length between leaves i and j. We further generalize this definition so that $d_{uv}(T^*)$ indicates the length of the undirected path between any two nodes of T^* (when clear, we simply write d_{uv}). Given some input data, we can compute a matrix of all pairwise sequence distances Δ, where the entry δ_{ij} indicates the dissimilarity between species i and j. When the sequence distance δ_{ij} is computed using (the correct) phylogenetic model, it will be a noisy but statistically consistent estimate of the tree distance $d_{ij}(T)$ (Felsenstein, 2003). Given these "phylogenetically corrected" distances (e.g. $\frac{3}{4}\ln(1 - \frac{4}{3}h)$ is the corrected hamming distance h under the Jukes and Cantor (1969) model), we can define optimization problems to recover the tree that best fits the distances. A natural choice is minimizing the (weighted) least square difference between tree and sequence distances:

$$Q^*(T) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(\delta_{ij} - d_{ij}(T))^2 \,. \tag{1}$$

Here, weights (e.g., w_{ij}) are used to reduce the impact of large distances (expected to have high variance). A general weighting schema can be defined as $w_{ij} = \delta_{ij}^{-k}$ for a *constant* value $k \in \mathbb{N}$. Standard choices of k include $k = 0$ for the ordinary least squares (OLS)

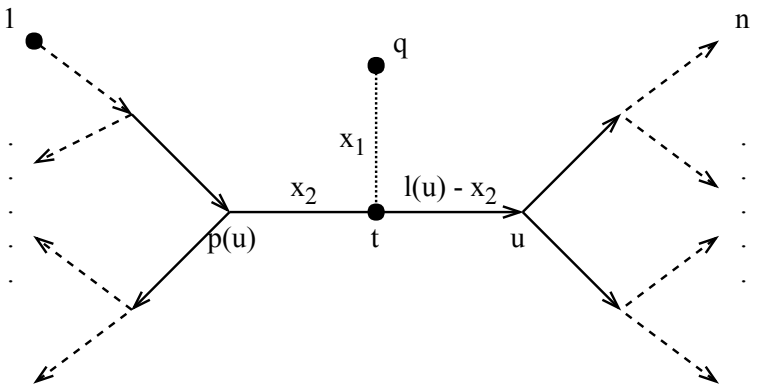

Fig. 1. Any placement of q can be characterized as a tree $P(u, x_1, x_2)$, shown here. The backbone tree T^* is an arborescence on leaves $\mathcal{L} = \{1 \dots n\}$, rooted at leaf 1. Query taxon q is added on the edge between u and $p(u)$, creating a node t. All placements on this edge are characterized by x_1, the length of the pendant branch, and x_2, the distance between t and $p(u)$.

method of Cavalli-Sforza and Edwards 1967, $k = 1$ due to Beyer *et al.* 1974 (BE), and $k = 2$ due to Fitch and Margoliash 1967 (FM).

Finding $\arg\min_T Q^*(T)$ is NP-Complete (Day and Sankoff, 1987). However, decades of research has produced heuristics like neighbor-joining (Saitou and Nei, 1987), alternative formulations like (balanced) minimum evolution (Cavalli-Sforza and Edwards, 1967; Desper and Gascuel, 2002), and several effective tools for solving the problem heuristically (e.g., FastME by Lefort *et al.* 2015, DAMBE by Xia 2018, and Ninja by Wheeler 2009).

Phylogenetic placement. We let $P(u, x_1, x_2)$ be the tree obtained by adding a query taxon q on an edge $(p(u), u)$, creating three edges (t, q), $(p(u), t)$, and (t, u), with lengths x_1, x_2, and $l(u) - x_2$, respectively (Fig. 1). When clear, we simply write P and note that P *induces* T both in topology and branch length. We now define the problem.

Least Squares Phylogenetic Placement (LSPP).

Input: A backbone tree T on $\mathcal{L}$, a query species q, and a vector Δ_{q*} with elements δ_{qi} giving sequence distances between q and every species $i \in \mathcal{L}$;

Output: The placement tree P that adds q on T and minimizes

$$Q(P) = \sum_{i=1}^{n} w_{qi}(\delta_{qi} - d_{qi}(P))^2 \tag{2}$$

Linear Time Solution for LSPP

The number of possible placements of q is $2n-3$. Therefore, LSPP can be solved by simply iterating over all the topologies, optimizing the score for that branch, and returning the placement with the minimum least square error. A naive algorithm can accomplish this in $\Theta(n^2)$ running time by optimizing Eq. 2 for each of the $2n-3$ branches. However, using dynamic programming, the optimal solution can be found in linear time.

THEOREM 1 The LSPP problem can be solved with $\Theta(n)$ running time and memory.

The proof (given in Appendix A) follows easily from three lemmas that we next state. The algorithm starts with precomputing a fixed-size set of values for each nodes. For any node u and exponents $a \in \mathbb{Z}$ and $b \in \mathbb{N}^+$, let $S(a,b,u) = \sum_{i \in g(u)} \delta_{qi}^a d_{ui}^b$ and for $b = 0$, let $S(a,0,u) = S'(a,u) = \sum_{i \in g(u)} \delta_{qi}^a$. Note that $S'(0,u) = |g(u)|$. Similarly, for $u \in V \setminus \{1\}$, let $R(a,b,u) = \sum_{i \notin g(u)} \delta_{qi}^a d_{p(u)i}^b$ for $b > 0$ and let $R(a,0,u) = R'(a,u) = \sum_{i \notin g(u)} \delta_{qi}^a$.

LEMMA 2 The set of all $S(a,b,u)$ and $R(a,b,u)$ values can be precomputed in $\Theta(n)$ time with two tree traversals using the dynamic programming given by:

$$S(a,b,u) = \begin{cases} \delta_{qu}^a & u \in \mathcal{L} \setminus \{1\} \ \ \& \ \ b = 0 \\ 0 & u \in \mathcal{L} \setminus \{1\} \ \ \& \ \ b \neq 0 \\ \sum_{j=0}^{b} \sum_{v \in c(u)} l(v)^j \binom{b}{j} S(a, b-j, v) & u \notin \mathcal{L} \setminus \{1\} \end{cases} \tag{3}$$

$$R(a,b,u) = \begin{cases} \delta_{q1}^a & u = 1' = c(1) \ \ \& \ \ b = 0 \\ 0 & u = 1' = c(1) \ \ \& \ \ b \neq 0 \\ \sum_{j=0}^{b} \left(l(p(u))^j \binom{b}{j} R(a, b-j, p(u)) + \sum_{v \in sib(u)} l(v)^j \binom{b}{j} S(a, b-j, v) \right) & u \notin \{1, 1'\} \end{cases} \tag{4}$$

LEMMA 3 Equation 2 can be rearranged (see Eq. S2 in Appendix A) such that computing $Q(P)$ for a given $P = P(u, x_1, x_2)$ requires a constant time computation using $S(a,b,u)$ and $R(a,b,u)$ values for $-k \leqslant a \leqslant 2-k$ and $0 \leqslant b \leqslant 2$.

Thus, after a linear time precomputation, we can compute the error for any given placement in constant time. It remains to show that for each node, the optimal placement on the branch above it (e.g., x_1 and x_2) can be computed in constant time.

LEMMA 4 For a fixed node $u \in V \setminus \{1\}$, if $(\hat{x}_1, \hat{x}_2) = \arg\min_{x_1,x_2} Q(P(u, x_1, x_2))$, then

$$\begin{bmatrix} R'(-k,u)+S'(-k,u) & R'(-k,u)-S'(-k,u) \\ R'(-k,u)-S'(-k,u) & R'(-k,u)+S'(-k,u) \end{bmatrix} \cdot \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \begin{bmatrix} R'(1-k,u)+S'(1-k,u)-l(u)S'(-k,u)-R(-k,1,u)-S(-k,1,u) \\ R'(1-k,u)-S'(1-k,u)+l(u)S'(-k,u)-R(-k,1,u)+S(-k,1,u) \end{bmatrix} \quad (5)$$

and hence $\hat{x}_1, \hat{x}_2$ can be computed in constant time.

Non-negative branch lengths. The solution to Equation 5 does not necessarily conform to constraints $0 \leqslant x_1$ and $0 \leqslant x_2 \leqslant l(u)$. However, the following lemma (proof in Appendix A) allows us to easily impose the constraints by choosing optimal boundary points when unrestricted solutions fall outside boundaries.

LEMMA 5 With respect to variables x_1 and x_2, $Q(P(u, x_1, x_2))$ is a convex function.

Minimum evolution An alternative to directly using MLSE (Eq. 1) is the minimum evolution (ME) principle (Cavalli-Sforza and Edwards, 1967; Rzhetsky and Nei, 1992). Our algorithm can also optimize the ME criterion: after computing x_1 and x_2 by optimizing MLSE for each node u, we choose the placement with the minimum total branch length. This is equivalent to using $\arg\min_u x_1$, since the value of x_2 does not contribute to total branch length. Other solutions for ME placement exist (Desper and Gascuel, 2002), a topic we return to in the Discussion section.

Hybrid. We have observed cases where ME is correct more often than MLSE, but when it is wrong, unlike MLSE, it has a relatively high error. This observation led us to design a hybrid approach. After computing x_1 and x_2 for all branches, we first select the top $log_2(n)$ edges with minimum $Q(P(u, x_1, x_2))$ values (this requires $\Theta(n \log \log n)$ time). Among this set of edges, we place the query on the edge satisfying the ME criteria.

Datasets

We benchmark accuracy and scalability of APPLES in two settings: sample identification using assembly-free genome-skims on real biological data and placement using aligned sequences on simulated data.

Real genome-skim datasets for the assembly-free scenario

Columbicola genome-skims. We use a set of 61 genome-skims by Boyd *et al.* (2017), including 45 known lice species (some represented multiple times) and 7 undescribed species. We generate lower coverage skims of 0.1Gb or 0.5Gb by randomly subsampling the reads from the sequence read archives (SRA) provided by the original publication (NCBI BioProject PRJNA296666). We use BBTools (Bushnell, 2014) to filter subsampled reads for adapters and contaminants and remove duplicated reads. Since this dataset is not assembled, the coverage of the genome-skims is unknown; Skmer estimates the coverage to be between 0.2X and 1X for 0.1Gb samples (and 5 times that coverage with 0.5Gb).

Anopheles and Drosophila datasets. We also use two insect datasets used by Sarmashghi *et al.* (2019): a dataset of 22 *Anopheles* and a dataset of 21 *Drosophila* genomes (Table S1), both obtained from InsectBase (Yin *et al.*, 2016). For both datasets, genome-skims with 0.1Gb and 0.5Gb sequence were generated from the assemblies using the short-read simulator tool ART, with the read length $l = 100$ and default error profile. Since species have different genome sizes, with 0.1Gb data, our subsampled genome-skims range in coverage from 0.35X to 1X for Anopheles and from 0.4X to 0.8X for Drosophila.

More recently, Miller *et al.* (2018) sequenced several Drosophila genomes, including 12 species shared with the InsectBase dataset. Sarmashghi *et al.* (2019) subsampled the SRAs from this second project to 0.1Gb or 0.5Gb and, after filtering contaminants, obtained artificial genome-skims. We can use these genome-skims as query and the genome-skims from the InsectBase dataset as the backbone. Since the reference and query

come from two projects, the query genome-skim can have a non-zero distance to the same species in the reference set, providing a realistic test of sample identification applications.

Simulated datasets for the aligned sequence scenario

GTR. We use a 101-taxon dataset available from Mirarab and Warnow 2015. Sequences were simulated under the General Time Reversible (GTR) plus the Γ model of site rate heterogeneity using INDELible (Fletcher and Yang, 2009) on gene trees that were simulated using SimPhy (Mallo *et al.*, 2016) under the coalescent model evolving on species trees generated under the Yule model. Note that the same model is used for inference under ML placement methods (i.e., no model misspecification). We took all 20 replicates of this dataset with mutation rates between 5×10^{-8} and 2×10^{-7}, and for each replicate, randomly selected five estimated gene trees among those with $\leqslant$20% RF distance between estimated and true gene tree. Thus, we have a total of 100 backbone trees. This dataset is the simplest test case where model violation or miss-alignment is not a concern.

RNASim. Guo *et al.* 2009 designed a complex model of RNA evolution that does not make usual i.i.d assumptions of sequence evolution. Instead, it uses models of energy of the secondary structure to simulate RNA evolution by a mutation-selection population genetics model. This model is based on an inhomogeneous stochastic process without a global substitution matrix. The model complexity of RNASim allows us to test both ML and APPLES under a substantially misspecified model. An RNASim dataset of one-million sequences (with E.coli SSU rRNA used as the root), is available from Mirarab *et al.* 2015. We created several subsets of the full RNASim dataset.

i) Heterogeneous: We first randomly subsampled the full dataset to create 10 datasets of size $10,000$. Then, we chose the largest clade of size at most 250 from each replicate; this gives us 10 backbone trees of mean size 249.

ii) Varied diameter: To evaluate the impact of the evolutionary diameter (i.e., the highest distance between any two leaves in the backbone), we also created datasets with

low, medium, and high diameters. We sampled the largest five clades of size at most 250 from each of the 10 replicates used for the heterogeneous dataset. Among these 50 clades, we picked the bottom, middle, and top five clades in diameter, which had diameter in $[0.3, 0.4]$ (mean: 0.36), $[0.5, 0.52]$ (mean: 0.51), and $[0.65, 1.07]$ (mean: 0.82), respectively.

iii) Varied size (RNASim-VS): We randomly subsampled the tree of size one-million to create 5 replicates of datasets of size (n): 500, 1000, 5000, 10000, 50000, and 100000, and 1 replicate (due to size) of size 200000. For replicates that contain at least 5000 species, we removed sites that contain gaps in 95% or more of the sequences in the alignment.

iv) Query scalability (RNASim-QS): We first randomly subsampled the full one-million sequence RNASim dataset to create a dataset of size 500. Then for k =1 to 49,152 queries (choosing all $k = 3 \times 2^i, 0 \leqslant i \leqslant 14$) we created 5 replicates of k query sequences, again randomly subsampling from the full alignment with one-million sequences.

v) Alignment Error (RNASim-AE): Mirarab *et al.* (2015) used PASTA to estimate alignments on subsets of the RNASim dataset with up to 200,000 sequences. We use their reported alignment with 200,000 or 10,000 sequences (taking only replicate 1 in this case).

Methods

Alternative methods. For aligned data, we compare APPLES to two ML methods: pplacer (Matsen *et al.*, 2010) and EPA-ng (Barbera *et al.*, 2019). Matsen *et al.* (2010) found pplacer to be substantially faster than EPA (Berger and Stamatakis, 2011) while their accuracy was similar. EPA-ng improves the scalability of EPA; thus, we compare to EPA-ng in analyses that concerned scalability (e.g., RNASim-VS). We run pplacer and EPA-ng in their default mode using GTR+Γ model and use their best hit (ML placement). We also compare with a simple method referred to as CLOSEST that places the query as the sister to the species with the minimum distance to it. CLOSEST is meant to emulate the use of BLAST (if it could be used). For the assembly-free setting, existing phylogenetic placement methods cannot be used, and we compared only against CLOSEST.

APPLES Software. We implemented APPLES using python, internally using Treeswift (Moshiri, 2018) for phylogenetic operations. APPLES computes distances internally (for certain models described below) using vectorized numpy (Oliphant, 2006) operations but can also use input distance matrices (e.g. generated using FastME). It generates the output in the standard jplace format (Matsen *et al.*, 2012).

Distance calculation and models. In computing pairwise distances, positions that have a gap in at least one of the two sequences are always ignored. In our analyses, we exclusively use models for nucleotide sequences. We compute phylogenetic distances under the parameter-free JC69 model, the six-parameter Tamura and Nei 1993 (TN93) model, and the 12-parameter general Markov model (Lockhart *et al.*, 1994). We compute distances independently for all pairs, and not simultaneously as suggested by Tamura *et al.* (2004). We also use the Gamma model of sites rate heterogeneity for JC69 and TN93 using the standard approach (Waddell and Steel, 1997). Pairing Gamma with GTR is theoretically possible in the absence of noise; however, the method can run into problems on real data (Waddell and Steel, 1997). Thus, we do not include a GTR model directly. Instead, we use the log-det approach that can handle the most general (12-parameter) Markov model (Lockhart *et al.*, 1994); however, log-det cannot account for rate across sites heterogeneity (Waddell and Steel, 1997). The α parameter of the Gamma model cannot be computed from pairwise sequence comparisons (Steel, 2009); instead, we use the α computed from the backbone tree. We used the α parameter computed by RAxML (Stamatakis, 2014) run on the backbone alignment and given the backbone tree. For analyses with models other than JC69, we use FastME to compute distances (see Supplementary material for details) while for JC69, distances are computed by APPLES.

In analyses on assembly-free datasets, we first compute genomic distances using Skmer (Sarmashghi *et al.*, 2019). We then correct these distances using the JC69 model, without the Gamma model of rate variation.

APPLES parameters. We have chosen default parameter settings for APPLES and refer to this version as APPLES*. By default, we use FM weighting, the MLSE selection criterion, enforcement of non-negative branch lengths, and JC69 distances. When not specified otherwise, these default parameters are used.

Backbone alignments and trees. For genome-skimming experiments, we estimated the backbone tree using FastME from the JC69 distance matrix computed from genome-skims using Skmer. By design, no alignments are needed.

For simulated datasets, we present results both based on true backbone alignments (for all datasets) and estimated alignments (for large datasets). The true alignments are known from the simulations. To test the accuracy in the presence of alignment error, we use the available PASTA backbone alignment for RNASim-AE dataset. The alignments have considerable error as measured by FastSP (Mirarab and Warnow, 2011): 11.5% and 12.7% SPFN, 10.9% and 11.7% SPFP, and only 165 and 848 fully correctly aligned sites (2.2% and 6.4%), respectively for 10,000 and 200,000 sequences. As before, here, we remove sites with more than 95% gaps in the estimated alignments.

For true alignments, we ran RAxML (Stamatakis, 2014) using GTRGAMMA model for all datasets to compute the topology of the backbone tree except for RNASim-AE and RNASim-VS dataset, where due to the size, we used FastTree-2 (Price *et al.*, 2010). When estimated alignment is used (RNASim-AE dataset), we used the co-estimated tree output by PASTA, which is itself computed using FastTree-2. We always re-estimated branch lengths and model parameters on that fixed topology using RAxML (switching to GTRCAT for $n \geqslant 10,000$) before running ML methods. For APPLES, we re-estimated branch lengths using FastTree-2 under the JC69 model to match the model used for estimating distances.

Alignment of queries. For analyses with true alignment as the backbone, we also use the true alignment of the queries to the backbone. In analyses with estimated

backbone alignments, we align the query sequences to the estimated backbone alignment using SEPP (Mirarab *et al.*, 2012), which is a divide-and-conquer method that internally uses HMMER (Eddy, 1998, 2009), with alignment subset size set to 10% of the full set (default setting). We use the resulting extended alignment, after masking unaligned sites, to run both APPLES and EPA-ng, in both cases, placing on the full backbone tree. We also report results using default SEPP (which runs pplacer internally); however, here, due to limitations of pplacer, we use the default setting of SEPP, which set the placement subset size to 10% of the full set.

Evaluation Procedure

To evaluate the accuracy, we use a leave-one-out strategy. We remove each leaf i from the backbone tree T and place it back on this $T \setminus i$ tree to obtain the placement tree P. On the RNAsim-VS dataset, due to its large size, we only removed and added back 200 randomly chosen leaves per replicate. On the RNAsim-AE dataset, we remove 200 queries from the backbone at the same time (leave-many-out). Finally, for RNAsim-QS, we place k queries *in one run*, allowing the methods to benefit from optimizations designed for multiple queries, but note that queries are not selected from the backbone tree but are instead selected from the full dataset. In all cases, placement of queries is with respect to the backbone and not other queries.

Delta error. We measure the accuracy of a placement tree P of a single query q on a backbone tree T on leafset $\mathcal{L}$ with respect to the true tree T^* on $\mathcal{L} \cup \{q\}$ using delta error:

$$\Delta e(P) = |B(T^*) \setminus B(P)| - |B(T^* \restriction_{\mathcal{L}}) \setminus B(T)| \,. \tag{6}$$

where $B(.)$ is the set of bipartitions of a tree and $T^* \restriction_{\mathcal{L}}$ is the true tree restricted to $\mathcal{L}$. Note that $\Delta e(P) \geqslant 0$ because adding q cannot decrease the number of missing branches in T. We report delta error averaged over all queries (denoted as Δe). In leave-one-out experiments, placing q to the same location as the backbone before leaving it out can still

	(a) Columbicola			(b) Anopheles			(c) Drosophila		
	%	Δe	e_{max}	%	Δe	e_{max}	%	Δe	e_{max}
APPLES*	97	0.03	1	95	0.05	1	71	0.29	1
APPLES-ME	84	0.28	5	95	0.05	1	67	0.42	2
APPLES-HYBRID	87	0.16	2	95	0.05	1	67	0.33	1
CLOSEST	54	1.15	7	91	0.09	1	57	0.62	3
DE-NOVO	98	0.02	1	95	0.05	1	71	0.29	1

Table 1. **Assembly-free placement of genome-skims.** We show the percentage of placements into optimal position (those that do not increase Δe), average delta error (Δe), and maximum delta error (e_{max}) for APPLES, assignment to the CLOSEST species, and the placement to the position in the backbone (DE-NOVO) over the 61 (a), 22 (b), and 21 (c) placements. Results are shown for genome skims with 0.1Gbp of reads. Delta error is the increase in the missing branches between the reference tree and the backbone tree after placing each query.

have a non-zero delta error because the backbone tree is not the true tree. We refer to the placement of a leaf into its position in the backbone tree as the *de novo* placement. In leave-many-out experiments, we measure delta error of each query separately (not the delta error of the combination of all queries). On biological data, where the true tree is unknown, we use a reference tree (Fig. S1). For Drosophila and Anopheles, we use the tree available from the Open Tree Of Life (Hinchliff *et al.*, 2015) as the reference. For Columbicola, we use the ML concatenation tree published by Boyd *et al.* (2017) as the reference.

Results

Assembly-free Placement of Genome-skims

On our three biological genome-skim datasets, APPLES* successfully places the queries on the optimal position in most cases (97%, 95%, and 71% for Columbicola, Anopheles, and Drosophila, respectively) and is never off from the optimal position by more than one branch. Other versions of APPLES are less accurate than APPLES*; e.g., APPLES with ME can have up to five wrong branches (Table 1). On genome-skims, where assembly and alignment are not possible, existing placement tools cannot be used, and the only alternative is the CLOSEST method (emulating BLAST if assembly was possible). CLOSEST finds the optimal placement only in 54% and 57% of times for Columbicola and Drosophila; moreover, it can be off from the best placement by up to seven branches for

the Columbicola dataset. On the Anopheles dataset, where the reference tree is unresolved (Fig. S1), all methods perform similarly.

APPLES* is less accurate on the Drosophila dataset than other datasets. However, here, simply placing each query on its position in the backbone tree would lead to identical results (Table 1). Thus, placements by APPLES* are as good as the *de novo* construction, meaning that errors of APPLES* are entirely due to the differences between our backbone tree and the reference tree. Moreover, these errors are not due to low coverage; increasing the genome-skim size 5x (to 0.5Gb) does not decrease error (Table S4).

On Drosophila dataset, we next tested a more realistic sample identification scenario using the 12 genome-skims from the separate study (and thus, non-zero distance to the corresponding species in the backbone tree). As desired, APPLES* places all of 12 queries from the second study as sister to the corresponding species in the reference dataset.

Alignment-based Placement

We first compare the accuracy and scalability of APPLES* to ML methods and then compare various settings of APPLES. For ML, we use pplacer (shown everywhere) and EPA-ng (shown only when we study scalability and work on large backbones).

Comparison to Maximum Likelihood (ML) without Alignment Error

GTR dataset. On this dataset, where it faces no model misspecification, pplacer has high accuracy. It finds the best placement in 84% of cases and is off by one edge in 15% (Fig. 2a); its mean delta error (Δe) is only 0.17 edges. APPLES* is also accurate, finding the best placement in 78% of cases and resulting in the mean Δe =0.28 edges. Thus, even though pplacer uses ML and faces no model misspecification and APPLES* uses distances based on a simpler model, the accuracy of the two methods is within 0.1 edges on average. In contrast, CLOSEST has poor accuracy and is correct only 50% of the times, with the mean Δe of 1.0 edge.

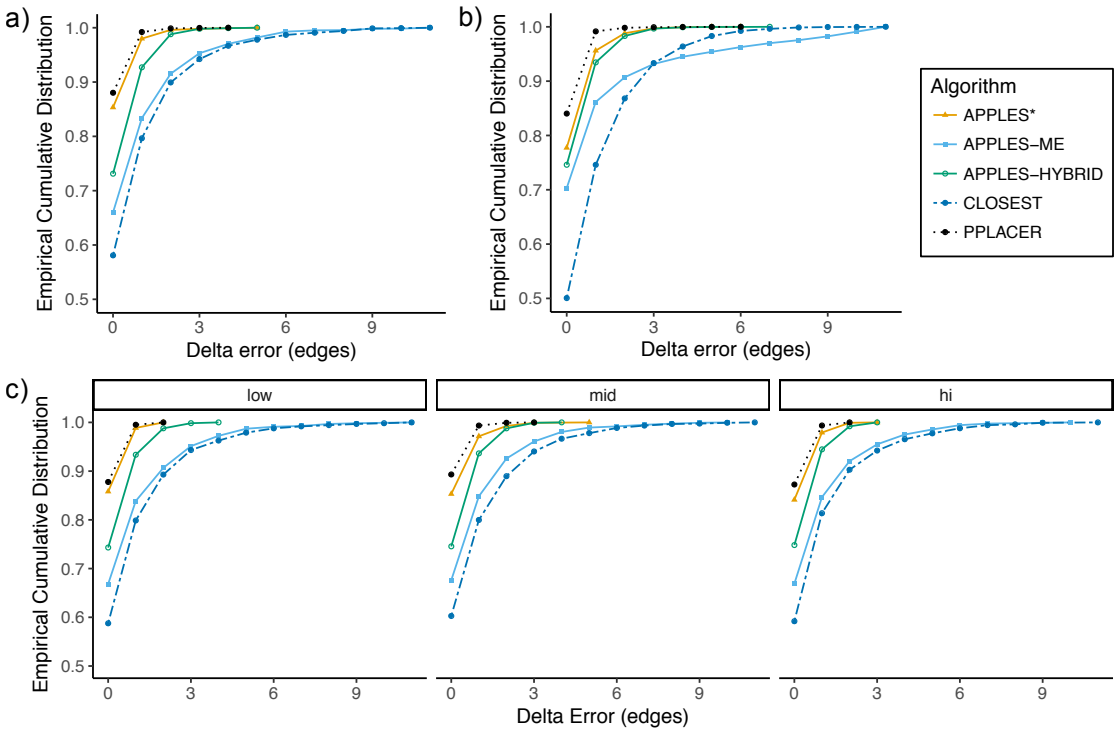

Fig. 2. **Accuracy on simulated data.** We show empirical cumulative distribution of the delta error, defined as the increase in the number of missing branches in the estimated tree compared to the true tree. We compare pplacer (dotted), CLOSEST match (dashed), and APPLES with FM weighting and JC69 distances and MLSE (APPLES*), ME, or Hybrid optimization. (a) GTR dataset. (b) RNASim-Heterogeneous. (c) RNASim-varied diameter, shown in boxes: low, medium (mid), or high. Distributions are over 10,000 (a), 2450 (b), and 3675 (c) points.

	Low			Medium			High			Heterogeneous		
	%	Δe	e_{max}	%	Δe	e_{max}	%	Δe	e_{max}	%	Δe	e_{max}
APPLES*	86	0.15	2	85	0.18	5	84	0.18	3	85	0.17	5
CLOSEST	59	0.88	13	60	0.88	13	60	0.85	14	60	0.87	14
pplacer	88	0.13	2	89	0.11	3	87	0.13	3	88	0.13	3

Table 2. The delta error for APPLES*, CLOSEST match, and pplacer on the RNASim-varied diameter dataset (low, medium, or high) and the RNA-heterogeneous dataset. Measurements are shown over 1250 placements for each diameter size category, corresponding to 5 backbone trees and 250 placements per replicate.

Model misspecification. On the small RNASim data with subsampled clades of ≈ 250 species), both APPLES* and pplacer face model misspecification. Here, the accuracy of APPLES* is very close to ML using pplacer. On the heterogeneous subset (Fig. 2b and Table 2), pplacer and APPLES* find the best placement in 88% and 85% of cases and have a mean delta error of 0.13 and 0.17 edges, respectively. Both methods are much more accurate than CLOSEST, which has a delta error of 0.87 edges on average.

	n = 500		n = 1,000		n = 5,000		n = 10000		n = 100,000		n = 200,000	
	%	Δe	%	Δe	%	Δe	%	Δe	%	Δe	%	Δe
APPLES*	75	0.32	71	0.43	77	0.37	79	0.33	84	0.25	87	0.25
CLOSEST	52	1.16	53	1.18	54	1.15	59	0.90	61	0.69	63	0.70
EPA-ng	73	0.33	73	0.31	78	0.24	79	0.22	n.p	n.p	n.p	n.p
pplacer	80	0.23	81	0.20	fail (800)	n.p	n.p	n.p	n.p	n.p	n.p	n.p

Table 3. Percentage of correct placements (shown as %) and the delta error (Δe) on the RNASim datasets with various backbone size (n). % and Δe is over 1000 placements (except $n = 200,000$, which is over 200 placements). Running pplacer and EPA-ng was not possible (n.p) for trees with at least 10,000 leaves and failed in some cases (number of fails shown) for 5,000 leaves.

Impact of diameter. When we control the tree diameter, APPLES* and pplacer remain very close in accuracy (Fig. 2c). The changes in error are small and not monotonic as the diameters change (Table 2). The accuracies of the two methods at low and high diameters are similar. The two methods are most divergent in the medium diameter case, where pplacer has its lowest error (Δe =0.11) and APPLES* has its highest error (Δe =0.18).

To summarize results on small RNASim dataset with model misspecification, although APPLES* uses a parameter-free model, its accuracy is extremely close to ML using pplacer with the GTR+Γ model.

Impact of taxon sampling. The real advantage of APPLES* over pplacer becomes clear for placing on larger backbone trees (Fig. 3 and Table 3). For backbone sizes of 500 and 1000, pplacer continues to be slightly more accurate than APPLES* (mean Δe of pplacer is better than APPLES* by 0.09 and 0.23 edges, respectively). However, with backbones of 5000 leaves, pplacer fails to run on 449/1000 cases, producing infinity likelihood (perhaps due to numerical issues) and has 41 times higher error than APPLES* on the rest (Fig. S2). Since pplacer could not scale to 5,000 leaves, we also test the recent method, EPA-ng (Barbera *et al.*, 2019). Given the 64GB of memory available on our machine, EPA-ng is able to run on datasets with up to 10,000 leaves. EPA-ng finds the correct placement less often than pplacer but is close to APPLES* (Fig. 3a). However, when it has error, it tends to have a somewhat lower distance to the correct placement, making its mean error slightly better than APPLES* (Fig. 3b and Table 3).

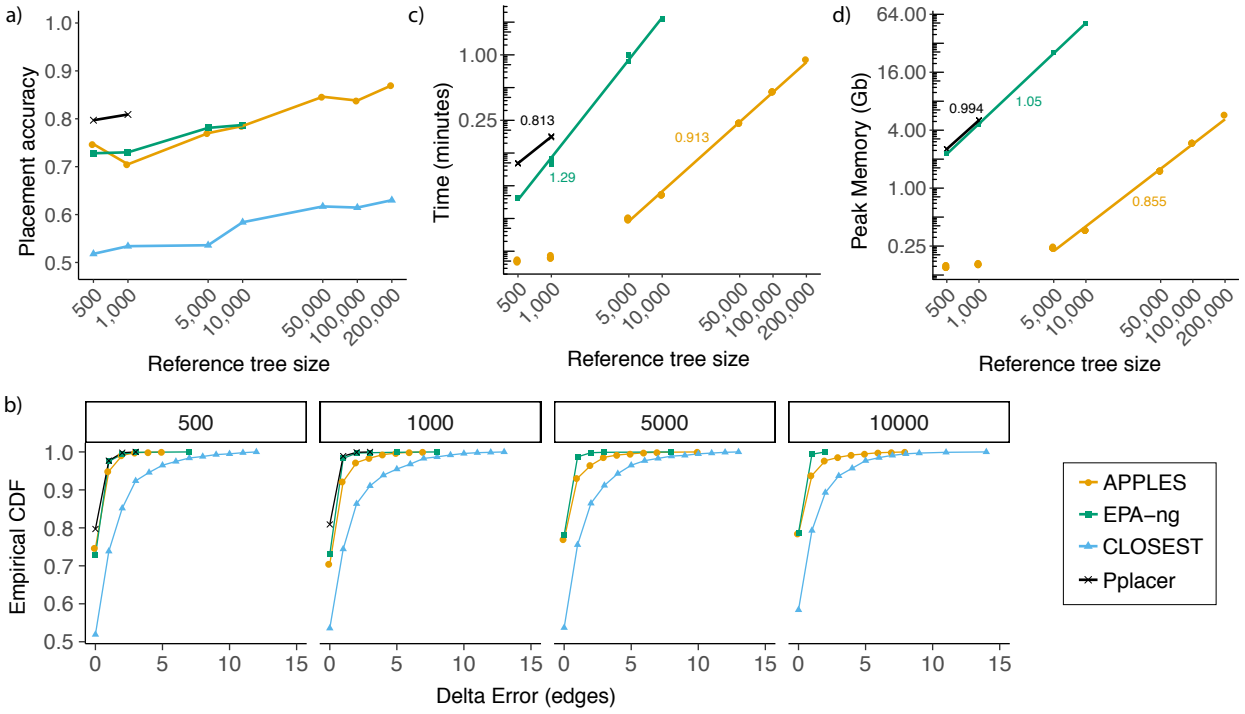

Fig. 3. **Results on RNASim-VS.** (a) Placement accuracy with taxon sampling ranging from 500 to 200,000. (b) The empirical cumulative distribution of the delta error, shown for $500 \leqslant n \leqslant 10000$ where EPA-ng can run. (c,d) Running time and peak memory usage of placement methods for a single placement. Lines are fitted in the log-log scale and their slope (indicated on the figure) empirically estimates the polynomial degree of the asymptotic growth ($t = n^a \Rightarrow \log t = a \log n$). APPLES lines are fitted to $\geqslant 5,000$ points because the first two values are small and irrelevant to asymptotic behavior. All calculations are on 8-core, 2.6GHz Intel Xeon CPUs (Sandy Bridge) with 64GB of memory, with each query placed independently and given 1 CPU core and the entire memory.

For backbones trees with more than $10,000$ leaves, pplacer and EPA-ng are not able to run given computational resources at hand, and CLOSEST is not very accurate (finding the best placement in only 59% of cases). However, APPLES* continues to be accurate for all backbone sizes. As the backbone size increases, the taxon sampling of the tree is improving (recall that these trees are all random subsets of the same tree). With denser backbone trees, APPLES* has increased accuracy despite placing on larger trees (Fig. 3a, Table 3). For example, using a backbone tree of $200,000$ leaves, APPLES* is able to find the best placement of query sequences in 87% of cases, which is better than the accuracy of either APPLES* or ML tools on any backbone size. Thus, an increased taxon sampling helps accuracy, but ML tools are limited in the size of the tree they can handle given relatively powerful machines (e.g., 64GB of memory).

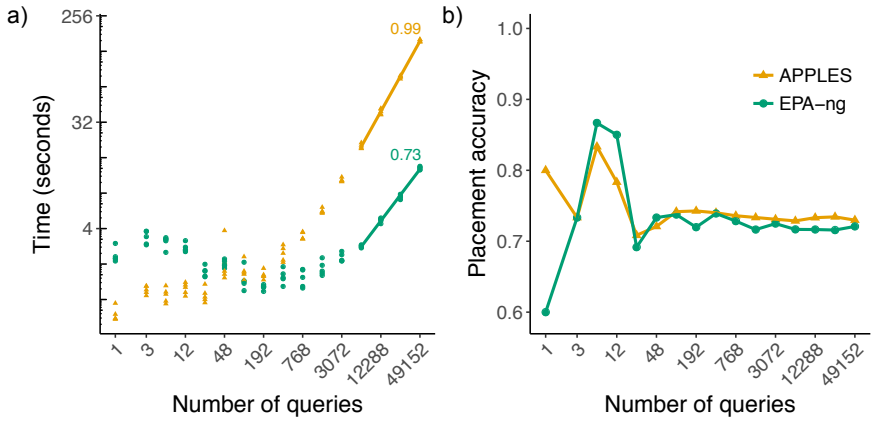

Fig. 4. **Scalability with respect to the number of queries.** We show wall-clock running time (a) and accuracy (b) with respect to increased numbers of queries (k) in one execution given 28 CPU cores on a Intel Xeon E5 CPU with 64 GB of memory. We fit a line to running times in log-log scale only for $k \geqslant 6144$ because otherwise, the preprocessing time would distort estimates (note: $t = n^a + b \not\Rightarrow \log t = a \log n + \epsilon$ except in approximation if $b \ll t$).

Running time and memory. As the backbone size increases, the running times of pplacer and APPLES grow close to linearly with the size of the backbone tree, n, whereas running time of EPA-ng seems to grow with $n^{1.3}$ (Fig. 3c). APPLES is on average 13 times faster than pplacer and 7.5 times faster than EPA-ng on backbone trees with 1000 leaves, and is 41 times faster than EPA-ng with 10,000-taxon backbones.

The memory consumption of all methods increases close to linearly with the n, but APPLES requires dramatically less memory (Fig. 3d). For example, for placing on a backbone with 10,000 leaves, EPA-ng requires 51GB of memory, whereas APPLES requires only 0.4GB. APPLES easily scales to a backbone of $200,000$ sequences, running in only 1 minute and using 6GB of memory per query (including all precomputations in the dynamic programming). These numbers also include the time and memory needed to compute the distance between the query sequence and all the backbone sequences.

We next test the scalability of APPLES* and EPA-ng with respect to the number of queries, k, in one run given 28 CPU cores on RNASim-QS dataset. Both methods spend time on preprocessing steps that will be amortized over a large number of queries. The running time of APPLES*, as expected, increases linearly with $k > 200$ and grow more slowly for $k < 200$ (due to the preprocessing) (Fig. 4a). The patterns of running time of

EPA-ng are surprising. Increasing k from 1 to 768 *decreases* the running time instead of increasing it. While the exact reasons for the reductions are not clear to us, we note that EPA-ng developers have taken great care to implement various optimizations, especially for utilizing multiple cores. The current version of APPLES*, in contrast, simply treats all queries as independent. Only with $k > 768$ we start to see increasing running times for EPA-ng. With $k > 3072$, EPA-ng seems to grow in running with $k^{0.73}$; however, we suspect further increasing k would increase the exponent (asymptotic running time cannot theoretically be less than $O(k)$ as all queries need to be read). Aside from the fluctuation due to small sample size for small k values, the number of queries do not seem to affect the accuracy for either method (Fig. 4b).

Comparing parameters of APPLES. Comparing five models of sequence evolution available in APPLES*, we see similar patterns of accuracy across all models despite their varying complexity, ranging from 0 to 12 parameters (Fig. S3). Since the JC69 model is parameter-free and results in similar accuracy to others, we have used it as the default. Next, we ask whether imposing the constraint to disallow negative branch lengths improves the accuracy. The answer depends on the optimization strategy. Forcing non-negative lengths marginally increases the accuracy for MLSE but dramatically reduces the accuracy for ME (Fig. S4ab). Thus, we always impose non-negative constraints on MLSE but never for ME. Likewise, our Hybrid method includes the constraint for the first MLSE step but not for the following ME step (Fig. S4c).

The next parameter to choose is the weighting scheme. Among the three methods available in APPLES, the best accuracy belongs to the FM scheme closely followed by the BE (Fig. S5). The OLS scheme, which does not penalize long distances, performs substantially worse than FM and BE. Thus, the most aggressive form of weighting (FM) results in the best accuracy. Fixing the weighting scheme to FM and comparing the three optimization strategies (MLSE, ME, and Hybrid), the MLSE approach has the best accuracy (Fig. 2), finding the correct placement 84% of the time (mean error: 0.18), and

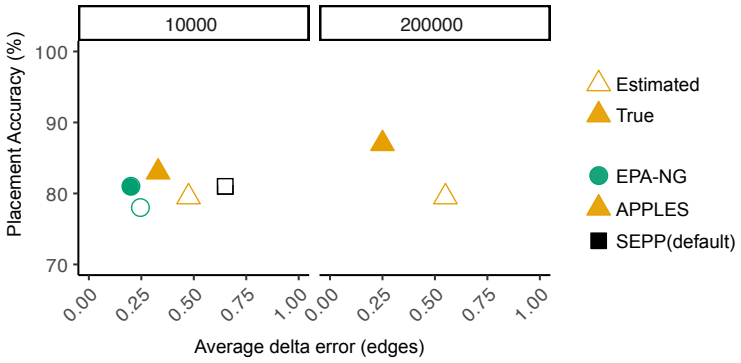

Fig. 5. **Impact of alignment error on placement accuracy.** Left and right panels show placement accuracy and mean delta error of leave-many-out experiments for backbone size of 10,000 and 200,000 (200 queries each).

ME has the lowest accuracy, finding the best placement in only 67% of cases (mean error: 0.70). The Hybrid approach is between the two (mean error: 0.34) and fails to outperform MLSE on this dataset. However, when we restrict the RNASim backbone trees to only 20 leaves, we observe that Hybrid can have the best accuracy (Fig. S6).

Comparison to ML with Alignment Error

We next test the impact of alignment errors. On RNASim-AE dataset with $n = 200,000$ sequences, we observe 80% placement accuracy using SEPP+APPLES* (Fig. 5), only 7% less than placing on the true backbone alignment with the true alignment of the query to the backbone. Similarly, on the $n = 10,000$ backbone, placement accuracy drops from 83% on the true alignment to 80% using SEPP+APPLES*, and from 81% to 78% using SEPP+EPA-ng. Despite the relatively small drops on placement accuracy, the impact of alignment error on delta error can be more pronounced (Fig. 5). The mean delta error for $n = 10,000$ and $n = 200,000$ goes up from 0.33 and 0.25 edges, respectively, to 0.48 and 0.55 edges.

Recall that SEPP+APPLES* eliminates the need for decomposing the backbone tree into smaller placement subtrees, as default SEPP must do to deal with memory requirements of pplacer (which it internally uses). Comparison of default SEPP and SEPP+APPLES* shows that incorporating APPLES* inside SEPP reduces the mean delta

error from 0.65 to 0.48, while it hurts placement accuracy by 1% (81% vs 80%). To summarize, highly accurate placement is possible even with estimated alignments with backbones of size up to 200,000 using the divide-and-conquer methods PASTA and SEPP for estimating alignments of the backbone and query, respectively.

Discussion

We introduced APPLES: a new method for adding query species onto large backbone trees using both unassembled genome-skims and aligned data. We now provide further observations on our results and on distance-based placement.

Further observations on the results

The accuracy of APPLES was very close to ML in most settings where we could run ML; the accuracy advantages of ML were particularly small for the RNASim dataset where both methods face model misspecification. As expected by the substantial evidence from the literature (Hillis *et al.*, 2003; Zwickl and Hillis, 2002), improved taxon sampling increased the accuracy of placement. Thus, overall, the best accuracy on RNASim dataset was obtained by APPLES* run on the full reference dataset, further motivating its use when large backbones are available. Despite many strides made in terms of scalability by the new method EPA-ng, ML methods still have to restrict their backbone to at most several thousand species given reasonable amounts of memory (up to 64GB in our case). We also note that it is possible to follow the APPLES* placement with a round of ML placement on smaller trees, but the small differences in accuracy of ML and APPLES* on smaller trees did not give us compelling reasons to try such hybrid approaches.

APPLES was an order of magnitude or more faster and less memory-hungry than ML tools (pplacer and EPA-ng) for single query runs. However, for placing large numbers of queries (e.g., as found in metagenomic datasets) on a relatively small sized backbone ($n = 500$), EPA-ng had an advantage thanks to heuristics it employs and carefully

engineered code. Advantages in memory consumption and scalability to large backbone trees remain for APPLES* regardless of the number of queries. The python APPLES code is not optimized nearly as much as EPA-ng and can also benefit from some of the heuristic techniques used by EPA-ng. We plan for the future versions of the code to focus on improved scalability as the number of queries increases.

By incorporating APPLES inside SEPP, we were able to create a method that can do both alignment and placement on very large backbones with reasonable computational requirements and high accuracy. We observed relatively small reductions in accuracy as a result of alignment error, a pattern that we find remarkable given the size of the tree and the amount of error in the estimated alignment. The default SEPP method deals with large backbone trees by dividing the backbone tree into "placement" subtrees and choosing which subtree to place on using bit-scores produced by HMMs trained on subsets of sequences. The original paper had shown that the best accuracy is obtained with the largest possible backbone subtrees given computational limitations. APPLES now enables us to eliminate the need for decomposition into placement subsets, and in doing so, reduces placement error.

In our analyses, we observed no advantage in using models more complex than JC69+Γ for distance calculation inside APPLES. However, these results may be due to our pairwise estimation of model parameters (e.g., base compositions). More complex models may perform better if we instead estimate model parameters on the backbone alignment/tree and reuse the parameters for queries (or simultaneously among all queries and the reference sequences). Simultaneous estimation of distances has many advantages over using independent distances for the *de novo* case (Tamura *et al.*, 2004; Xia, 2009); these results give us hope that using simultaneous distances inside APPLES can further improve its accuracy.

Branch lengths of our backbone trees were computed using the same distance model as the one used for computing the distance of the query to backbone species. Using

consistent models for the query and for the backbone branch lengths is essential for obtaining good accuracy (see Fig. S7 for evidence). Thus, in addition to having a large backbone tree at hand, we need to ensure that branch lengths are computed using the right model. Fortunately, FastTree-2 can compute both topologies and branch lengths on large trees in a scalable fashion, without a need for quadratic time/memory computation of distance matrices (Price *et al.*, 2010).

Observations on distance-based placement

Phylogenetic insertion using the ME criterion has been previously studied for the purpose of creating an algorithm for greedy minimum evolution (GME). Desper and Gascuel 2002 have designed a method that given the tree T can update it to get a tree with $n+1$ leaves in $\Theta(n)$ after precomputation of a data-structure that gives the average *sequence* distances between all adjacent clusters in T. The formulation by Desper and Gascuel 2002 has a subtle but consequential difference from our ME placement. Their algorithm does not compute branch lenghts for inserted sequence (e.g., x_1 and x_2). It is able to compute the optimal placement topology *without* knowing branch lengths of the backbone tree. Instead, it relies on pairwise distances among backbone *sequences* (Δ), which are precomputed and saved in the data-structure mentioned before. In the context of the greedy algorithm for tree inference, in each iteration, the data structure can be updated in $\Theta(n)$, which does not impact the overall running time of the algorithm. However, if we were to start with a tree of n leaves, computing this structure from scratch would still require $\Theta(n^2)$. Thus, computing the placement for a new query would need quadratic time, unless if the $\Theta(n^2)$ precomputation is allowed to be amortized over $\Omega(n)$ queries. Our formulation, in contrast, uses branch lengths of the backbone tree (which is assumed fixed) and thus never uses pairwise distances among the backbone sequences. Thus, using tree distances is what allows us to develop a linear time algorithm. Finally, we note that in our experimental analyses, we were not able to test the distance-based algorithm of Desper and

Gascuel (2002) because it is available only as part of a the greedy algorithm inside FastME but is not available as a stand-alone feature to place on a given tree.

We emphasize that our results in assembly-free tests do not advocate for the use of assembly-free methods when assemblies are available. Moreover, we have no evidence that assembly-free methods are effective in inferring deep branches of a phylogeny. Instead, our results show that assembly-free phylogenetic placement is effective in sample identification where assembly is not possible due to low coverage. In assembly-free analyses, we used Skmer to get distances because alternative alignment-free methods of estimating distance generally either require assemblies (e.g., Haubold, 2014; Leimeister and Morgenstern, 2014; Leimeister *et al.*, 2017) or higher coverage than Skmer (e.g., Benoit *et al.*, 2016; Yi and Jin, 2013; Ondov *et al.*, 2016); however, combining APPLES with other alignment-free methods can be attempted in future (finding the best way of computing distances without assemblies was not our focus). Moreover, the Skmer paper has described a trick that can be used to compute log-det distances from genome-skims. Future studies should test whether using that trick and using GTR instead of JC69 improves accuracy.

Acknowledgments

This work was supported by the National Science Foundation (NSF) grant IIS-1565862 and National Institutes of Health (NIH) subaward 5P30AI027767-28 to M.B. and S.M., and NSF grant NSF-1815485 to M.B., S.S., and S.M. Computations were performed on the San Diego Supercomputer Center (SDSC) through XSEDE allocations, which is supported by the NSF grant ACI-1053575.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. 1990. Basic local alignment search tool. *Journal of molecular biology*, 215(3): 403–10.

562 Barbera, P., Kozlov, A. M., Czech, L., Morel, B., Darriba, D., Flouri, T., and Stamatakis,
563 A. 2019. EPA-ng: Massively Parallel Evolutionary Placement of Genetic Sequences.
564 *Systematic Biology*, 68(2): 365–369.

565 Benoit, G., Peterlongo, P., Mariadassou, M., Drezen, E., Schbath, S., Lavenier, D., and
566 Lemaitre, C. 2016. Multiple comparative metagenomics using multiset k-mer counting.
567 *PeerJ Computer Science*, 2: e94.

568 Berger, S. A. and Stamatakis, A. 2011. Aligning short reads to reference alignments and
569 trees. *Bioinformatics*, 27(15): 2068–2075.

570 Berger, S. A., Krompass, D., and Stamatakis, A. 2011. Performance, accuracy, and Web
571 server for evolutionary placement of short sequence reads under maximum likelihood.
572 *Systematic biology*, 60(3): 291–302.

573 Beyer, W. A., Stein, M. L., Smith, T. F., and Ulam, S. M. 1974. A molecular sequence
574 metric and evolutionary trees. *Mathematical Biosciences*, 19(1-2): 9–25.

575 Boyd, B. M., Allen, J. M., Nguyen, N., Sweet, A. D., Warnow, T., Shapiro, M. D., Villa,
576 S. M., Bush, S. E., Clayton, D. H., and Johnson, K. P. 2017. Phylogenomics using
577 Target-restricted Assembly Resolves Intra-generic Relationships of Parasitic Lice
578 (Phthiraptera: Columbicola). *Systematic Biology*, page syx027.

579 Brown, D. and Truszkowski, J. 2013. LSHPlace: Fast phylogenetic placement using
580 locality-sensitive hashing. In *Pacific Symposium On Biocomputing*, pages 310–319.

581 Bush, A., Sollmann, R., Wilting, A., Bohmann, K., Cole, B., Balzter, H., Martius, C.,
582 Zlinszky, A., Calvignac-Spencer, S., Cobbold, C. A., Dawson, T. P., Emerson, B. C.,
583 Ferrier, S., Gilbert, M. T. P., Herold, M., Jones, L., Leendertz, F. H., Matthews, L.,
584 Millington, J. D. A., Olson, J. R., Ovaskainen, O., Raffaelli, D., Reeve, R., Rödel, M.-O.,
585 Rodgers, T. W., Snape, S., Visseren-Hamakers, I., Vogler, A. P., White, P. C. L.,

Wooster, M. J., and Yu, D. W. 2017. Connecting Earth observation to high-throughput biodiversity data. *Nature Ecology & Evolution*, 1(7): 0176.

Bushnell, B. 2014. Bbtools software package. *URL http://sourceforge. net/projects/bbmap*.

Cavalli-Sforza, L. L. and Edwards, A. W. 1967. Phylogenetic analysis. Models and estimation procedures. *American journal of human genetics*, 19(3 Pt 1): 233–57.

Clarke, L. J., Soubrier, J., Weyrich, L. S., and Cooper, A. 2014. Environmental metabarcodes for insects: In silico PCR reveals potential for taxonomic bias. *Molecular Ecology Resources*, 14(6): 1160–1170.

Coissac, E., Hollingsworth, P. M., Lavergne, S., and Taberlet, P. 2016. From barcodes to genomes: extending the concept of DNA barcoding. *Molecular Ecology*, 25(7): 1423–1428.

Day, W. H. E. and Sankoff, D. 1987. Computational complexity of inferring phylogenies from chromosome inversion data. *Journal of Theoretical Biology*, 124(2): 213–218.

Desper, R. and Gascuel, O. 2002. Fast and accurate phylogeny reconstruction algorithms based on the minimum-evolution principle. *Journal of computational biology : a journal of computational molecular cell biology*, 9(5): 687–705.

Dodsworth, S. 2015. Genome skimming for next-generation biodiversity analysis. *Trends in Plant Science*, 20(9): 525–527.

Eddy, S. R. 1998. Profile hidden Markov models. *Bioinformatics*, pages 755–763.

Eddy, S. R. 2009. A new generation of homology search tools based on probabilistic inference. *International Conference on Genome Informatics*, 23(1): 205–11.

Fan, H., Ives, A. R., Surget-Groba, Y., and Cannon, C. H. 2015. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC Genomics*, 16(1): 522.

610 Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood
611 approach. *Journal of Molecular Evolution*, 17(6): 368–376.

612 Felsenstein, J. 2003. *Inferring phylogenies*.

613 Findley, K., Oh, J., Yang, J., Conlan, S., Deming, C., Meyer, J. A., Schoenfeld, D.,
614 Nomicos, E., Park, M., Kong, H. H., and Segre, J. A. 2013. Topographic diversity of
615 fungal and bacterial communities in human skin. *Nature*, 498(7454): 367–370.

616 Fitch, W. M. and Margoliash, E. 1967. Construction of Phylogenetic Trees. *Science*,
617 155(3760): 279–284.

618 Fletcher, W. and Yang, Z. 2009. INDELible: A flexible simulator of biological sequence
619 evolution. *Molecular Biology and Evolution*, 26(8): 1879–1888.

620 Gill, S. R., Pop, M., Deboy, R. T., Eckburg, P. B., Turnbaugh, P. J., Samuel, B. S.,
621 Gordon, J. I., Relman, D. A., Fraser-Liggett, C. M., and Nelson, K. E. 2006.
622 Metagenomic analysis of the human distal gut microbiome. *Science (New York, N.Y.)*,
623 312(5778): 1355–9.

624 Guo, S., Wang, L.-S., and Kim, J. 2009. Large-scale simulation of RNA macroevolution by
625 an energy-dependent fitness model. *arXiv*, 0912.2326.

626 Haubold, B. 2014. Alignment-free phylogenetics and population genetics. *Briefings in*
627 *Bioinformatics*, 15(3): 407–418.

628 Hebert, P. D. N., Cywinska, A., Ball, S. L., and deWaard, J. R. 2003. Biological
629 identifications through DNA barcodes. *Proceedings of the Royal Society B: Biological*
630 *Sciences*, 270(1512): 313–321.

631 Hillis, D. M., Pollock, D. D., McGuire, J. A., and Zwickl, D. J. 2003. Is sparse taxon
632 sampling a problem for phylogenetic inference? *Systematic biology*, 52(1): 124–126.

633 Hinchliff, C. E., Smith, S. a., Allman, J. F., Burleigh, J. G., Chaudhary, R., Coghill, L. M.,
634 Crandall, K. A., Deng, J., Drew, B. T., Gazis, R., Gude, K., Hibbett, D. S., Katz, L. a.,
635 Laughinghouse, H. D., McTavish, E. J., Midford, P. E., Owen, C. L., Ree, R. H., Rees,
636 J. a., Soltis, D. E., Williams, T. L., and Cranston, K. a. 2015. Synthesis of phylogeny
637 and taxonomy into a comprehensive tree of life. *Proceedings of the National Academy of*
638 *Sciences*, 112(41): 12764–12769.

639 Jain, C., Rodriguez-R, L. M., Phillippy, A. M., Konstantinidis, K. T., and Aluru, S. 2018.
640 High throughput ANI analysis of 90K prokaryotic genomes reveals clear species
641 boundaries. *Nature Communications*, 9(1): 5114.

642 Janssen, S., McDonald, D., Gonzalez, A., Navas-Molina, J. A., Jiang, L., Xu, Z. Z.,
643 Winker, K., Kado, D. M., Orwoll, E., Manary, M., Mirarab, S., and Knight, R. 2018.
644 Phylogenetic Placement of Exact Amplicon Sequences Improves Associations with
645 Clinical Information. *mSystems*, 3(3): 00021–18.

646 Jukes, T. H. and Cantor, C. R. 1969. Evolution of protein molecules. In *Mammalian*
647 *protein metabolism, Vol. III (1969), pp. 21-132*, volume III, pages 21–132.

648 Koski, L. B. and Golding, G. B. 2001. The closest BLAST hit is often not the nearest
649 neighbor. *Journal of molecular evolution*, 52(6): 540–2.

650 Krause, L., Diaz, N. N., Goesmann, A., Kelley, S., Nattkemper, T. W., Rohwer, F.,
651 Edwards, R. A., and Stoye, J. 2008. Phylogenetic classification of short environmental
652 DNA fragments. *Nucleic acids research*, 36(7): 2230–9.

653 Lefort, V., Desper, R., and Gascuel, O. 2015. FastME 2.0: A comprehensive, accurate, and
654 fast distance-based phylogeny inference program. *Molecular Biology and Evolution*,
655 32(10): 2798–2800.

656 Leimeister, C.-A. and Morgenstern, B. 2014. kmacs: the k -mismatch average common

657 substring approach to alignment-free sequence comparison. *Bioinformatics*, 30(14):
658 2000–2008.

659 Leimeister, C.-A., Sohrabi-Jahromi, S., and Morgenstern, B. 2017. Fast and accurate
660 phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics*, 33(7):
661 971–979.

662 Lockhart, P. J., Steel, M. A., Hendy, M. D., and Penny, D. 1994. Recovering evolutionary
663 trees under a more realistic model of sequence evolution. *Molecular Biology and*
664 *Evolution*, 11(4): 605–612.

665 Mallo, D., De Oliveira Martins, L., and Posada, D. 2016. SimPhy: Phylogenomic
666 Simulation of Gene, Locus, and Species Trees. *Systematic biology*, 65(2): 334–44.

667 Matsen, F. A. and Evans, S. N. 2013. Edge Principal Components and Squash Clustering:
668 Using the Special Structure of Phylogenetic Placement Data for Sample Comparison.
669 *PLoS ONE*, 8(3).

670 Matsen, F. A., Kodner, R. B., and Armbrust, E. V. 2010. pplacer: linear time
671 maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed
672 reference tree. *BMC bioinformatics*, 11(1): 538.

673 Matsen, F. A., Hoffman, N. G., Gallagher, A., and Stamatakis, A. 2012. A Format for
674 Phylogenetic Placements. *PLoS ONE*, 7(2): e31009.

675 Miller, D. E., Staber, C., Zeitlinger, J., and Hawley, R. S. 2018. Highly Contiguous
676 Genome Assemblies of 15 Drosophila Species Generated Using Nanopore Sequencing.
677 *G3: Genes, Genomes, Genetics*, 8(10): 3131–3141.

678 Mirarab, S. and Warnow, T. 2011. FastSP: linear time calculation of alignment accuracy.
679 *Bioinformatics*, 27(23): 3250–8.

680 Mirarab, S. and Warnow, T. 2015. ASTRAL-II: Coalescent-based species tree estimation
681 with many hundreds of taxa and thousands of genes. *Bioinformatics*, 31(12): i44–i52.

Mirarab, S., Nguyen, N., and Warnow, T. 2012. SEPP: SATé-Enabled Phylogenetic Placement. *Pacific Symposium On Biocomputing*, pages 247–58.

Mirarab, S., Nguyen, N., Guo, S., Wang, L.-S., Kim, J., and Warnow, T. 2015. PASTA: Ultra-Large Multiple Sequence Alignment for Nucleotide and Amino-Acid Sequences. *Journal of computational biology : a journal of computational molecular cell biology*, 22(5): 377–86.

Moshiri, N. 2018. TreeSwift: a massively scalable Python tree package. *bioRxiv*.

Nguyen, L. T., Schmidt, H. A., Von Haeseler, A., and Minh, B. Q. 2015. IQ-TREE: A fast and effective stochastic algorithm for estimating maximum-likelihood phylogenies. *Molecular Biology and Evolution*, 32(1).

Nguyen, N.-p., Mirarab, S., Liu, B., Pop, M., and Warnow, T. 2014. TIPP: taxonomic identification and phylogenetic profiling. *Bioinformatics*, 30(24): 3548–3555.

Oliphant, T. E. 2006. *A guide to NumPy*, volume 1. Trelgol Publishing USA.

Ondov, B. D., Treangen, T. J., Melsted, P., Mallonee, A. B., Bergman, N. H., Koren, S., and Phillippy, A. M. 2016. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology*, 17(1): 132.

Price, M. N., Dehal, P. S., and Arkin, A. P. 2010. FastTree-2 Approximately Maximum-Likelihood Trees for Large Alignments. *PLoS ONE*, 5(3): e9490.

Rzhetsky, A. and Nei, M. 1992. A Simple Method for Estimating and Testing Minimum-Evolution Trees. *Molecular Biology and Evolution*, 9(5): 945–945.

Saitou, N. and Nei, M. 1987. The neighbour-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4): 406–425.

Sarmashghi, S., Bohmann, K., P. Gilbert, M. T., Bafna, V., and Mirarab, S. 2019. Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome Biology*, 20(1): 34.

Stamatakis, A. 2014. RAxML version 8: A tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9): 1312–1313.

Stark, M., Berger, S. A., Stamatakis, A., and von Mering, C. 2010. MLTreeMap–accurate Maximum Likelihood placement of environmental DNA sequences into taxonomic and functional reference phylogenies. *BMC genomics*, 11(1): 461.

Steel, M. 2009. A basic limitation on inferring phylogenies by pairwise sequence comparisons. *Journal of Theoretical Biology*, 256(3): 467–472.

Sunagawa, S., Mende, D. R., Zeller, G., Izquierdo-Carrasco, F., Berger, S. a., Kultima, J. R., Coelho, L. P., Arumugam, M., Tap, J., Nielsen, H. B., Rasmussen, S., Brunak, S., Pedersen, O., Guarner, F., de Vos, W. M., Wang, J., Li, J., Dore, J., Ehrlich, S. D., Stamatakis, a., and Bork, P. 2013. Metagenomic species profiling using universal phylogenetic marker genes. *Nature methods*, 10(12): 1196–1199.

Tamura, K. and Nei, M. 1993. Estimation of the Number of Nucleotide Substitutions in the Control Region of Mitochondrial-DNA in Humans and Chimpanzees. *Molecular biology and evolution*, 10(3): 512–526.

Tamura, K., Nei, M., and Kumar, S. 2004. Prospects for inferring very large phylogenies by using the neighbor-joining method. *Proceedings of the National Academy of Sciences*, 101(30): 11030–11035.

von Mering, C., Hugenholtz, P., Raes, J., Tringe, S. G., Doerks, T., Jensen, L. J., Ward, N., and Bork, P. 2007. Quantitative Phylogenetic Assessment of Microbial Communities in Diverse Environments. *Science*, 315(5815): 1126–1130.

Waddell, P. J. and Steel, M. A. 1997. General time-reversible distances with unequal rates across sites: mixing gamma and inverse Gaussian distributions with invariant sites. *Molecular phylogenetics and evolution*, 8(3): 398–414.

731 Warnow, T. 2017. *Computational phylogenetics: An introduction to designing methods for*
732 *phylogeny estimation*. Cambridge University Press.

733 Wheeler, T. J. 2009. Large-scale neighbor-joining with NINJA. In *Algorithms in*
734 *Bioinformatics*, pages 375–389. Springer.

735 Xia, X. 2009. Information-theoretic indices and an approximate significance test for testing
736 the molecular clock hypothesis with genetic distances. *Molecular Phylogenetics and*
737 *Evolution*, 52(3): 665–676.

738 Xia, X. 2018. DAMBE7: New and Improved Tools for Data Analysis in Molecular Biology
739 and Evolution. *Molecular Biology and Evolution*, 35(6): 1550–1552.

740 Yi, H. and Jin, L. 2013. Co-phylog: an assembly-free phylogenomic approach for closely
741 related organisms. *Nucleic acids research*, 41(7): e75.

742 Yin, C., Shen, G., Guo, D., Wang, S., Ma, X., Xiao, H., Liu, J., Zhang, Z., Liu, Y., Zhang,
743 Y., Yu, K., Huang, S., and Li, F. 2016. InsectBase: a resource for insect genomes and
744 transcriptomes. *Nucleic Acids Research*, 44(D1): D801–D807.

745 Zwickl, D. J. and Hillis, D. M. 2002. Increased taxon sampling greatly reduces
746 phylogenetic error. *Systematic biology*, 51(4): 588–98.

APPENDIX

Appendix A. Proofs and derivations

Recall the following notations.

- For any node u and exponents $a \in \mathbb{Z}$ and $b \in \mathbb{N}^+$, let
  - $S(a,b,u) = \sum_{i \in g(u)} \delta_{qi}^a d_{ui}^b$
  - $R(a,b,u) = \sum_{i \notin g(u)} \delta_{qi}^a d_{p(u)i}^b$ defined for $u \in V \setminus \{1\}$
- For $b = 0$, let $S(a,0,u) = \sum_{i \in g(u)} \delta_{qi}^a$ and let $S'(a,u)$ be a shorthand for $S(a,0,u)$. Similarly, let $R(a,0,u) = R'(a,u) = \sum_{i \notin g(u)} \delta_{qi}^a$.

Proof of Lemma 2

Proof. Recall the dynamic programming recursions of Equations 3 and 4:

$$S(a,b,u) = \sum_{j=0}^{b} \sum_{v \in c(u)} l(v)^j \binom{b}{j} S(a, b-j, v), \quad \text{for } u \notin \mathcal{L} \setminus \{1\}$$

$$R(a,b,u) = \sum_{j=0}^{b} \left(l(p(u))^j \binom{b}{j} R(a, b-j, p(u)) + \sum_{v \in sib(u)} l(v)^j \binom{b}{j} S(a, b-j, v) \right) \text{ for } u \notin \{1, 1'\}$$

Since u is not a leaf, for each leaf $i \in g(u)$, there exists a $v \in c(u)$ such that the directed path from u to i passes through v. Therefore every leaf i can be grouped under its corresponding v.

$$\begin{aligned} S(a,b,u) = \sum_{i \in g(u)} \delta_{qi}^a d_{ui}^b = \sum_{v \in c(u)} \sum_{i \in g(v)} \delta_{qi}^a \big(l(v) + d_{vi}\big)^b &= \sum_{j=0}^{b} \sum_{v \in c(u)} \sum_{i \in g(v)} \delta_{qi}^a d_{vi}^{b-j} l(v)^j \binom{b}{j} \\ &= \sum_{j=0}^{b} \sum_{v \in c(u)} l(v)^j \binom{b}{j} S(a, b-j, v) \end{aligned}$$

Similarly, given the condition $u \neq 1$, for each leaf $i \notin g(u)$, either (1) there exists $v \in sib(u)$ such that the directed path from $p(u)$ to i passes through v, or (2) undirected path between i and $p(u)$ passes through $p(p(u))$.

$$
\begin{aligned}
R(a,b,u) &= \sum_{i\notin g(u)} \delta_{qi}^{a} d_{p(u)i}^{b} = \sum_{v\in sib(u)} \sum_{i\in g(v)} \delta_{qi}^{a} \big(l(v)+d_{vi}\big)^{b} + \sum_{i\notin g(p(u))} \delta_{qi}^{a} \big(l(p(u))+d_{p(p(u))i}\big)^{b} \\
&= \sum_{j=0}^{b} \sum_{v\in sib(u)} \sum_{i\in g(v)} \delta_{qi}^{a} d_{vi}^{b-j} l(v)^{j} \binom{b}{j} + \sum_{j=0}^{b} \sum_{i\notin g(p(u))} \delta_{qi}^{a} d_{p(p(u))i}^{b-j} l(p(u))^{j} \binom{b}{j} \\
&= \sum_{j=0}^{b} \sum_{v\in sib(u)} l(v)^{j} \binom{b}{j} S(a,b-j,v) + \sum_{j=0}^{b} l(p(u))^{j} \binom{b}{j} R(a,b-j,p(u))
\end{aligned}
$$

Boundary conditions follow from definitions. For $u \notin \mathcal{L} \setminus \{1\}$, since $d_{ii} = 0$, we have $S(a,b,u) = 0$ and it's trivial to see $S'(a,u) = \delta_{qu}^{a}$. For $R(,,)$ recursions, the boundary case happens at the unique child of the root, which we denote as $1'$. Based on the definition, since the only $i \notin g(1')$ is 1, and $d_{p(1')1}^{b} = 0$, we trivially have $R(a,b,1') = 0$. For $b = 0$, $R'(a,1') = \delta_{q1}^{a}$.

A post-order traversal on T^{*} can compute $S(a,b,u)$, and a subsequent pre-order traversal can compute $R(a,b,u)$, both in constant time and using constant memory per node. Recall that a and b are both no more than k, which is a constant. Thus, time and memory complexity of this dynamic programming is $\Theta(bn)$, which translates to $\Theta(n)$ in least squares setting, where $b \leqslant 2$. □

Proof of Lemma 3.

Recall $w_{qi} = \delta_{qi}^{-k}$ and that Equation 2:

$$
Q(P) = \sum_{i=1}^{n} w_{qi} (\delta_{qi} - d_{qi}(P))^{2} = \sum_{i=1}^{n} \delta_{qi}^{-k} (\delta_{qi} - d_{qi}(P))^{2}
$$

Proof. Equation 2 can be re-written as:

$$
Q(P) = \sum_{i\in g(u)} \delta_{qi}^{-k} \big(\delta_{qi} - d_{ui}(P) - x_1 + x_2 - l(u)\big)^{2} + \sum_{i\notin g(u)} \delta_{qi}^{-k} \big(\delta_{qi} - d_{p(u)i}(P) - x_1 - x_2\big)^{2} \tag{S1}
$$

By simple rearrangement of the terms, we can rewrite Equation S1 as follows.

$$\begin{aligned} Q(P(u,x_1,x_2)) &= R'(2-k,u) + S'(2-k,u) + R(-k,2,u) + S(-k,2,u) \\ &\quad + 2\big(x_1+x_2\big)\big(R(-k,1,u) - R'(1-k,u)\big) \\ &\quad + 2\big(x_1+l(u)-x_2\big)\big(S(-k,1,u) - S'(1-k,u)\big) \\ &\quad + \big(x_1+x_2\big)^2 R(-k,1,u) + \big(x_1+l(u)-x_2\big)^2 S(-k,1,u) \\ &\quad - 2R(1-k,1,u) - 2S(1-k,1,u) \end{aligned} \tag{S2}$$

Note that computing $Q(P(u,x_1,x_2))$ requires only $S(,,u)$ and $R(,,u)$ values and $l(u)$. Thus, computing $Q(P)$ requires only computing $S(a,b,u)$ and $R(a,b,u)$ values for $-k \leqslant a \leqslant 2-k$ and $0 \leqslant b \leqslant 2$.

□

Proof of Lemma 4.

Recall definitions

$$S(a,b,u) = \sum_{i \in g(u)} \delta_{qi}^a d_{ui}^b \quad (\text{for } b>0) \qquad \text{and} \qquad S'(a,u) = S(a,0,u) = \sum_{i \in g(u)} \delta_{qi}^a$$
$$R(a,b,u) = \sum_{i \notin g(u)} \delta_{qi}^a d_{p(u)i}^b \quad (\text{for } b>0) \qquad \text{and} \qquad R'(a,u) = R(a,0,u) = \sum_{i \notin g(u)} \delta_{qi}^a$$

and recall Eq. S1:

$$Q(P) = \sum_{i \in g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{ui}(P) - x_1 + x_2 - l(u)\big)^2 + \sum_{i \notin g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{p(u)i}(P) - x_1 - x_2\big)^2 .$$

Proof. We take the derivative of Eq. S1 with respect to x_1 and set it equal to zero:

$$\frac{\partial Q(P)}{\partial x_1} = -2\sum_{i\in g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{ui}(P) - x_1 + x_2 - l(u)\big) - 2\sum_{i\notin g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{p(u)i}(P) - x_1 - x_2\big) = 0$$

$$\implies \Big(\sum_{i\in g(u)} \delta_{qi}^{-k} + \sum_{i\notin g(u)} \delta_{qi}^{-k}\Big)x_1 + \Big(-\sum_{i\in g(u)} \delta_{qi}^{-k} + \sum_{i\notin g(u)} \delta_{qi}^{-k}\Big)x_2$$
$$- \sum_{i\in g(u)} \delta_{qi}^{1-k} - \sum_{i\notin g(u)} \delta_{qi}^{1-k} + \sum_{i\in g(u)} \delta_{qi}^{-k} d_{ui}(P) + \sum_{i\notin g(u)} \delta_{qi}^{-k} d_{p(u)i}(P) + l(u)\sum_{i\in g(u)} \delta_{qi}^{-k} = 0$$

$$\implies \big(S'(-k,u) + R'(-k,u)\big)x_1 + \big(-S'(-k,u) + R'(-k,u)\big)x_2 =$$

$$S'(1-k,u) + R'(1-k,u) - S(-k,1,u) - R(-k,1,u) - l(u)S'(-k,u)$$

Similarly,

$$\frac{\partial Q(P)}{\partial x_2} = 2\sum_{i\in g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{ui}(P) - x_1 + x_2 - l(u)\big) - 2\sum_{i\notin g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{p(u)i}(P) - x_1 - x_2\big) = 0$$

$$\implies \Big(-\sum_{i\in g(u)} \delta_{qi}^{-k} + \sum_{i\notin g(u)} \delta_{qi}^{-k}\Big)x_1 + \Big(\sum_{i\in g(u)} \delta_{qi}^{-k} + \sum_{i\notin g(u)} \delta_{qi}^{-k}\Big)x_2$$
$$+ \sum_{i\in g(u)} \delta_{qi}^{1-k} - \sum_{i\notin g(u)} \delta_{qi}^{1-k} - \sum_{i\in g(u)} \delta_{qi}^{-k} d_{ui}(P) + \sum_{i\notin g(u)} \delta_{qi}^{-k} d_{p(u)i}(P) - l(u)\sum_{i\in g(u)} \delta_{qi}^{-k} = 0$$

$$\implies \big(-S'(-k,u) + R'(-k,u)\big)x_1 + \big(+S'(-k,u) + R'(-k,u)\big)x_2 =$$

$$-S'(1-k,u) + R'(1-k,u) + S(-k,1,u) - R(-k,1,u) + l(u)S'(-k,u)$$

These two linear equations have a unique solution for the pair x_1, x_2 if and only if the following matrix has the full rank:

$$H = \begin{bmatrix} R'(-k,u) + S'(-k,u) & R'(-k,u) - S'(-k,u) \\ R'(-k,u) - S'(-k,u) & R'(-k,u) + S'(-k,u) \end{bmatrix}.$$

Determinant of H is $\det(H) = 4R'(-k,u)S'(-k,u)$. Assuming that $\delta_{qi} > 0$ for all $i \in \mathcal{L}$, both $R'(-k,u) > 0$ and $S'(-k,u) > 0$ hold. Therefore, H has the full rank. However, $\delta_{qi} = 0$ for $q \neq i$ can be encountered on real data, especially for low divergence times, low evolutionary rates, or short sequences. In this case, APPLES is designed to place q on the pendant edge of i with $x_1 = 0$ and $x_2 = l(i)$. In case there are multiple leaves i that satisfy $\delta_{qi} = 0$ for $q \neq i$, we pick one of them arbitrarily. □

Proof of Theorem 1

Proof. First, using two traversals of the tree, we compute all the $S(a, b, u)$ and $R(a, b, u)$ values by Lemma 2. To find the optimal placement edge, we first optimize $Q(P(u, x_1, x_2))$ for all $u \in V \setminus \{1\}$. By Lemma 4, this task requires only constant time after the precompuations. Then, for each node, we compute $Q(P(u, x_1, x_2))$ in constant time for the optimal u, x_1, x_2 by Lemma 3. Thus, each node is processed in linear time and the whole optimization requires linear time. Note that the system of equations (shown in Lemma 4) will not have a solution iff $\delta_{qi} \leqslant 0$ for some i; if there is $\delta_{qi} = 0$, we make q sister to i, breaking ties arbitrarily. □

Proof of Lemma 5

Proof. Eigenvalues of the Hessian matrix of $Q(P(u, x_1, x_2))$ are $2R'(-k, u)$ and $2S'(-k, u)$, which are both non-negative since $\delta_{qi} \geqslant 0$ for $i \in \mathcal{L}$. Thus, the Hessian matrix is positive semidefinite and therefore $P(u, x_1, x_2)$ is a convex function of x_1 and x_2. □

Appendix B. Supplementary Figures

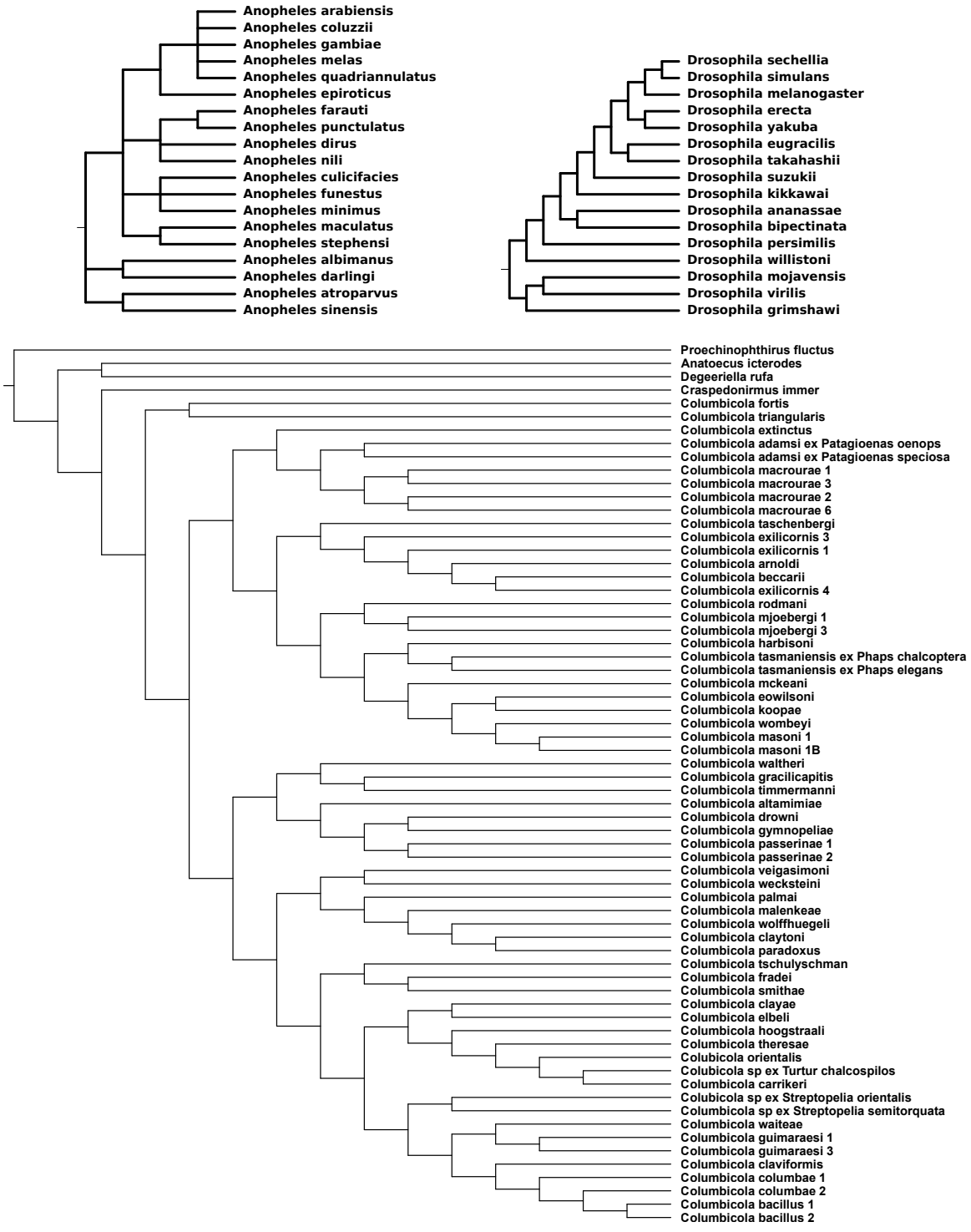

Fig. S1. **The reference biological trees obtained from Open Tree of Life (Drosophila and Anopheles) and from Boyd *et al.* (2017) (Columbicola).**

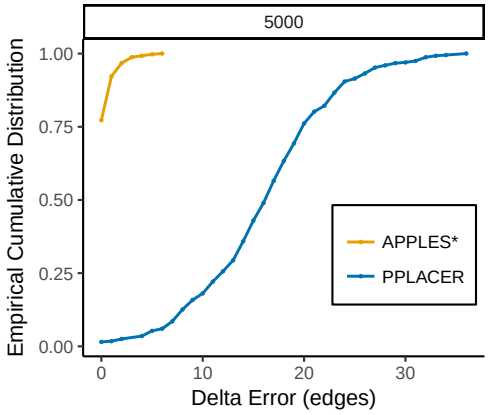

Fig. S2. **APPLES versus pplacer on 5,000 backbone trees**. The empirical cumulative distribution of the delta error is shown. We compare pplacer and APPLES*on RNASim-VS dataset with 5000 leaves. Distributions is over 551 cases where pplacer could run for the panel.

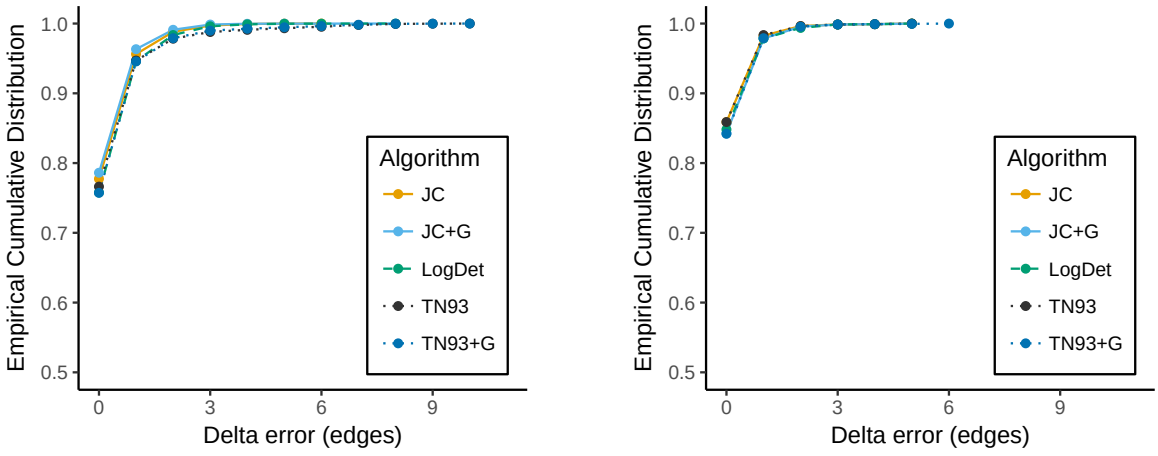

Fig. S3. **Comparing various models of DNA evolution**. For the GTR (a) and RNASim-heterogeneous (b) datasets, we show the delta error (edges) of APPLES* run with five distance matrices calculated based on different models of DNA evolution. All model parameters are estimated per pair of sequences. The five models have similar accuracy.

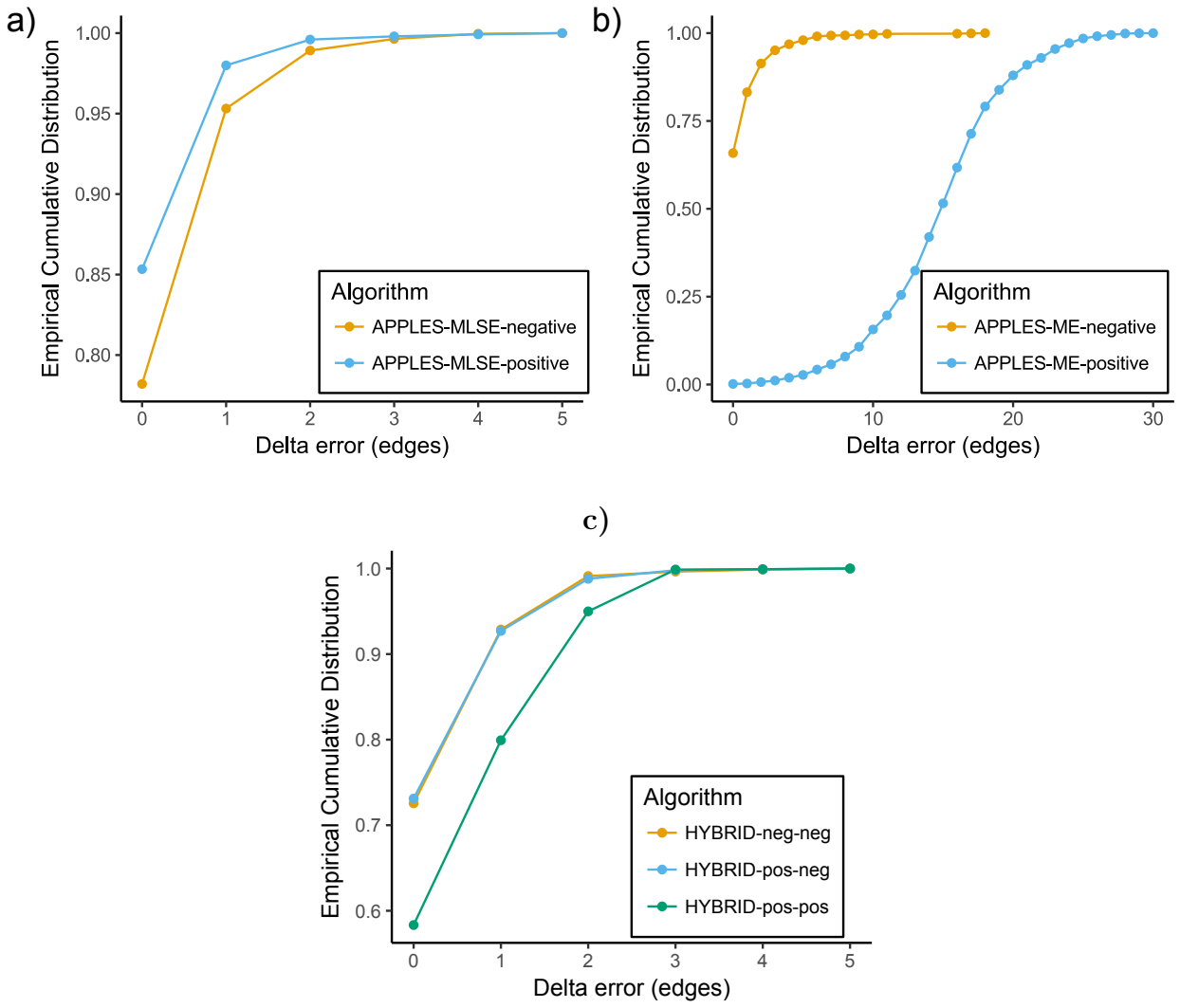

Fig. S4. **(a,b) The effect of imposing positivity constraint on error.** We show the error of (a) APPLES-MLSE and (b) APPLES-ME run both with and without enforcement of non-negative branch lengths on RNASim heterogeneous dataset. Accuracy improves substantially for MLSE whereas it reduces drastically for ME. **(c) The effect of imposing positivity constraint on accuracy on Hybrid.** The HYBRID approach does not benefit from imposing positivity constraint on its second (ME) stage.

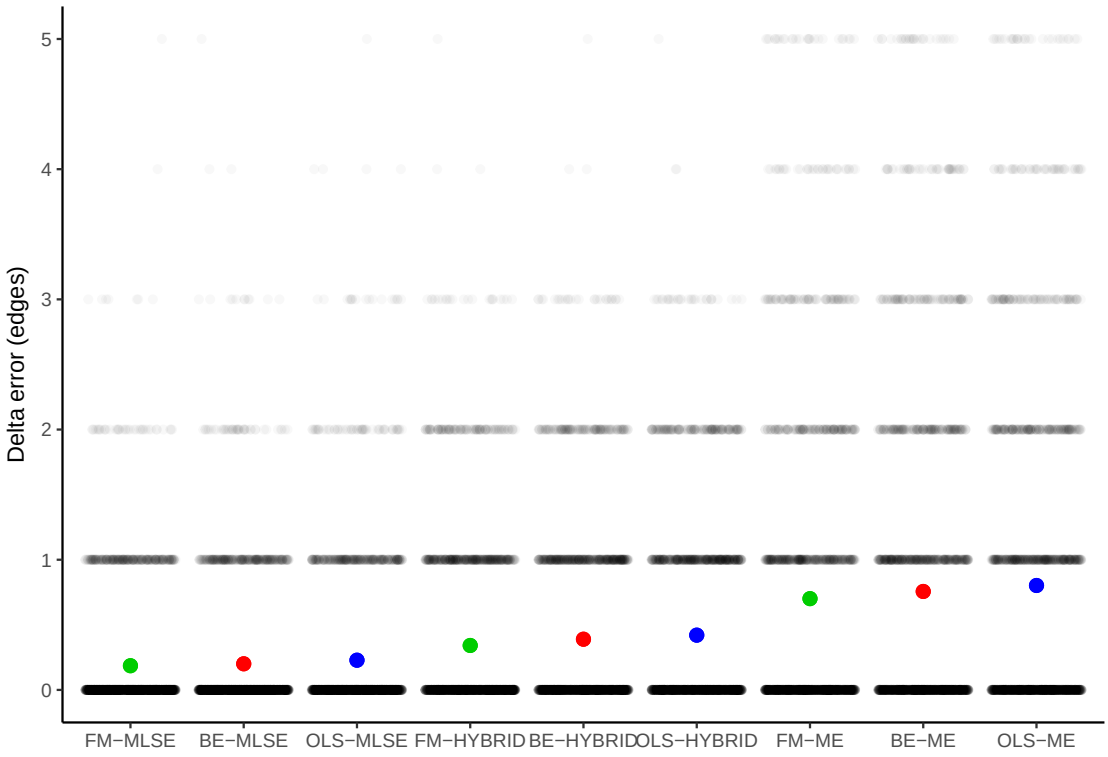

Fig. S5. **Comparing APPLES versions**. For the RNASim dataset (without controlling for the diameter), we show the delta error (edges) of APPLES run with three options for weighting: FM (green), BE (red), and OLS (blue), and three options for selection strategy (MLSE, ME, and Hybrid). For each method, the mean (colored circle) and standard errors (lines; too small to see) are shown over 2500 data points, each shown as dots. Some of the methods occasionally have error above 5 branches, but for better resolution, we cap the y-axis at 5.

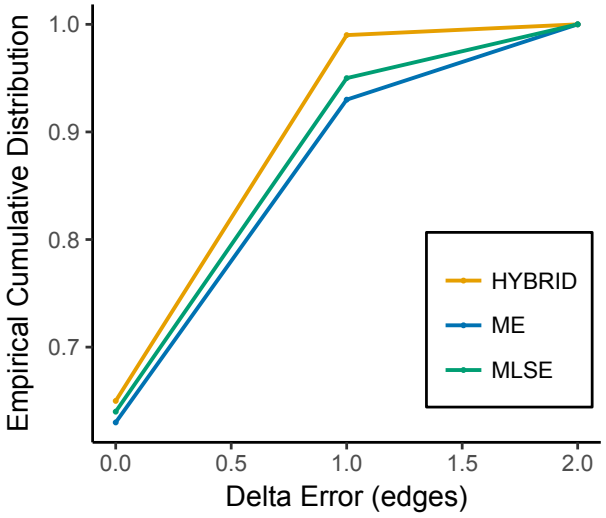

Fig. S6. **APPLES-HYBRID has higher accuracy on sparse RNAsim dataset** On the RNAsim dataset, we chose 20 sequences randomly from the larger RNAsim-heterogeneous dataset; here, APPLES-HYBRID has higher accuracy than APPLES* (MLSE).

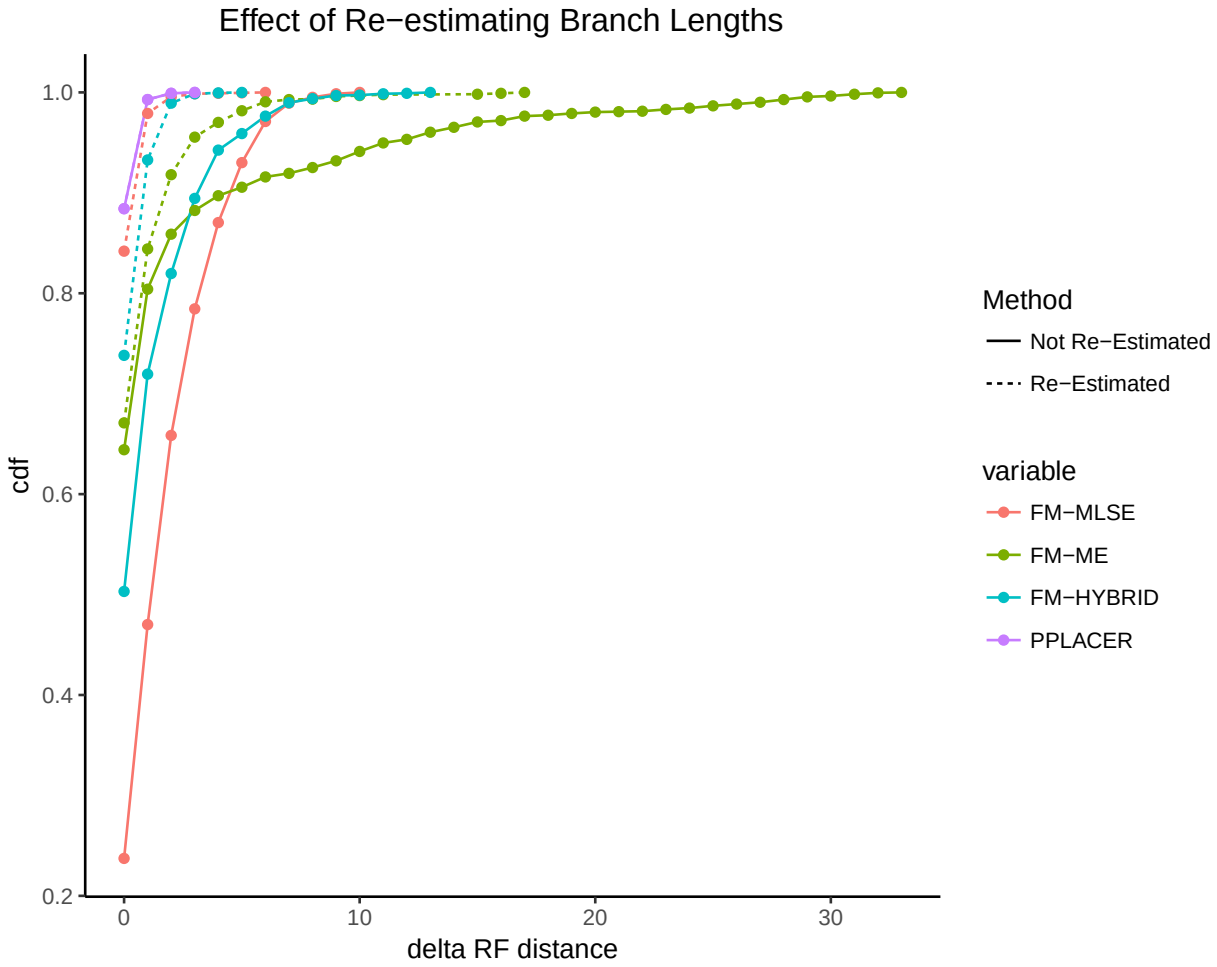

Fig. S7. **The effect of reestimating branch lengths of the backbone tree on accuracy.** We show the accuracy of pplacer and APPLES-FM with its three optimization criteria. APPLES is run both with (dotted) and without (solid) re-estimating branch lengths in the backbone tree using the same model (here, TN93+Γ) used for computing distances of query sequences to backbone sequences. FastME* is used to re-estimate branch lengths. Accuracy improves dramatically by recomputing backbone branch lengths using the same model. The case labeled "Not re-estimated" uses branch lengths produced using RAxML under the GTR+Γ model.

Appendix C. Supplementary Tables

Table S1. GenBank accession numbers and URLs for the dataset of 22 Anopheles genomes

Species	GenBank assembly accession	URL
Anopheles albimanus	GCA_000349125.1	http://www.insect-genome.com/data/genome_download/Anopheles_albimanus/Anopheles_albimanus_genomic.fasta.gz
Anopheles arabiensis	GCA_000349185.1	http://www.insect-genome.com/data/genome_download/Anopheles_arabiensis/Anopheles_arabiensis_genomic.fasta.gz
Anopheles atroparvus	GCA_000473505.1	http://www.insect-genome.com/data/genome_download/Anopheles_atroparvus/Anopheles_atroparvus_genomic.fasta.gz
Anopheles christyi	GCA_000349165.1	http://www.insect-genome.com/data/genome_download/Anopheles_christyi/Anopheles_christyi_genomic.fasta.gz
Anopheles coluzzii	-	http://www.insect-genome.com/data/genome_download/Anopheles_coluzzii/Anopheles_coluzzii_genomic.fasta.gz
Anopheles culicifacies	GCA_000473375.1	http://www.insect-genome.com/data/genome_download/Anopheles_culicifacies/Anopheles_culicifacies_genomic.fasta.gz
Anopheles darlingi	GCA_000211455.3	http://www.insect-genome.com/data/genome_download/Anopheles_darlingi/Anopheles_darlingi_genomic.fasta.gz
Anopheles dirus	GCA_000349145.1	http://www.insect-genome.com/data/genome_download/Anopheles_dirus/Anopheles_dirus_genomic.fasta.gz
Anopheles epiroticus	GCA_000349105.1	http://www.insect-genome.com/data/genome_download/Anopheles_epiroticus/Anopheles_epiroticus_genomic.fasta.gz
Anopheles farauti	GCA_000956265.1	http://www.insect-genome.com/data/genome_download/Anopheles_farauti/Anopheles_farauti_genomic.fasta.gz
Anopheles funestus	GCA_000349085.1	http://www.insect-genome.com/data/genome_download/Anopheles_funestus/Anopheles_funestus_genomic.fasta.gz
Anopheles gambiae	GCA_000150785.1	http://www.insect-genome.com/data/genome_download/Anopheles_gambiae/Anopheles_gambiae_genomic.fasta.gz
Anopheles koliensis	GCA_000956275.1	http://www.insect-genome.com/data/genome_download/Anopheles_koliensis/Anopheles_koliensis_genomic.fasta.gz
Anopheles maculatus	GCA_000473185.1	http://www.insect-genome.com/data/genome_download/Anopheles_maculatus/Anopheles_maculatus_genomic.fasta.gz
Anopheles melas	GCA_000473525.2	http://www.insect-genome.com/data/genome_download/Anopheles_melas/Anopheles_melas_genomic.fasta.gz
Anopheles merus	GCA_000473845.2	http://www.insect-genome.com/data/genome_download/Anopheles_merus/Anopheles_merus_genomic.fasta.gz
Anopheles minimus	GCA_000349025.1	http://www.insect-genome.com/data/genome_download/Anopheles_minimus/Anopheles_minimus_genomic.fasta.gz
Anopheles nili	GCA_000439205.1	http://www.insect-genome.com/data/genome_download/Anopheles_nili/Anopheles_nili_genomic.fasta.gz
Anopheles punctulatus	GCA_000956255.1	http://www.insect-genome.com/data/genome_download/Anopheles_punctulatus/Anopheles_punctulatus_genomic.fasta.gz
Anopheles quadriannulatus	GCA_000349065.1	http://www.insect-genome.com/data/genome_download/Anopheles_quadriannulatus/Anopheles_quadriannulatus_genomic.fasta.gz
Anopheles sinensis	GCA_000441895.2	http://www.insect-genome.com/data/genome_download/Anopheles_sinensis/Anopheles_sinensis_genomic.fasta.gz
Anopheles stephensis	GCA_000300775.2	http://www.insect-genome.com/data/genome_download/Anopheles_stephensi/Anopheles_stephensi_genomic.fasta.gz

Table S2. GenBank accession numbers and URLs for the dataset of 21 Drosophila genomes

Species	GenBank assembly accession	URL
Drosophila ananassae	GCA_000005115.1	http://www.insect-genome.com/data/genome_download/Drosophila_ananassae/Drosophila_ananassae_genomic.fasta.gz
Drosophila biarmipes	GCA_000233415.2	http://www.insect-genome.com/data/genome_download/Drosophila_biarmipes/Drosophila_biarmipes_genomic.fasta.gz
Drosophila bipectinata	GCA_000236285.2	http://www.insect-genome.com/data/genome_download/Drosophila_bipectinata/Drosophila_bipectinata_genomic.fasta.gz
Drosophila elegans	GCA_000224195.2	http://www.insect-genome.com/data/genome_download/Drosophila_elegans/Drosophila_elegans_genomic.fasta.gz
Drosophila erecta	GCA_000005135.1	http://www.insect-genome.com/data/genome_download/Drosophila_erecta/Drosophila_erecta_genomic.fasta.gz
Drosophila eugracilis	GCA_000236325.2	http://www.insect-genome.com/data/genome_download/Drosophila_eugracilis/Drosophila_eugracilis_genomic.fasta.gz
Drosophila ficusphila	GCA_000220665.2	http://www.insect-genome.com/data/genome_download/Drosophila_ficusphila/Drosophila_ficusphila_genomic.fasta.gz
Drosophila grimshawi	GCA_000005155.1	http://www.insect-genome.com/data/genome_download/Drosophila_grimshawi/Drosophila_grimshawi_genomic.fasta.gz
Drosophila kikkawai	GCA_000224215.2	http://www.insect-genome.com/data/genome_download/Drosophila_kikkawai/Drosophila_kikkawai_genomic.fasta.gz
Drosophila melanogaster	GCA_000778455.1	http://www.insect-genome.com/data/genome_download/Drosophila_melanogaster/Drosophila_melanogaster_genomic.fasta.gz
Drosophila miranda	GCA_000269505.2	http://www.insect-genome.com/data/genome_download/Drosophila_miranda/Drosophila_miranda_genomic.fasta.gz
Drosophila mojavensis	GCA_000005175.1	http://www.insect-genome.com/data/genome_download/Drosophila_mojavensis/Drosophila_mojavensis_genomic.fasta.gz
Drosophila persimilis	GCA_000005195.1	http://www.insect-genome.com/data/genome_download/Drosophila_persimilis/Drosophila_persimilis_genomic.fasta.gz
Drosophila rhopaloa	GCA_000236305.2	http://www.insect-genome.com/data/genome_download/Drosophila_rhopaloa/Drosophila_rhopaloa_genomic.fasta.gz
Drosophila sechellia	GCA_000005215.1	http://www.insect-genome.com/data/genome_download/Drosophila_sechellia/Drosophila_sechellia_genomic.fasta.gz
Drosophila simulans	GCA_000259055.1	http://www.insect-genome.com/data/genome_download/Drosophila_simulans/Drosophila_simulans_genomic.fasta.gz
Drosophila suzukii	GCA_000472105.1	http://www.insect-genome.com/data/genome_download/Drosophila_suzukii/Drosophila_suzukii_genomic.fasta.gz
Drosophila takahashii	GCA_000224235.2	http://www.insect-genome.com/data/genome_download/Drosophila_takahashii/Drosophila_takahashii_genomic.fasta.gz
Drosophila virilis	GCA_000005245.1	http://www.insect-genome.com/data/genome_download/Drosophila_virilis/Drosophila_virilis_genomic.fasta.gz
Drosophila willistoni	GCA_000005925.1	http://www.insect-genome.com/data/genome_download/Drosophila_willistoni/Drosophila_willistoni_genomic.fasta.gz
Drosophila yakuba	GCA_000005975.1	http://www.insect-genome.com/data/genome_download/Drosophila_yakuba/Drosophila_yakuba_genomic.fasta.gz

Table S3. GenBank accession numbers of microbial species used in contamination removal.

Species	GenBank assembly accession
Pasteurella langaaensis	GCA_003096995.1
Providencia stuartii	GCA_001558855.2
Serratia marcescens	GCA_000783915.2
Shigella flexneri	GCA_000006925.2
Commensalibacter intestini	GCA_002153535.1
Acetobacter malorum	GCA_002153605.1
Acetobacter pomorum	GCA_002456135.1
Lactobacillus plantarum	GCA_000203855.3
Lactobacillus brevis	GCA_003184305.1
Enterococcus faecalis	GCA_002208945.2
Vagococcus teuberi	GCA_001870205.1
Wolbachia	GCA_000022285.1

	0.1G			0.5G		
	%	Δe	e_{max}	%	Δe	e_{max}
(a) **Columbicola**						
APPLES*	97	0.03	1	92	0.08	1
APPLES-ME	84	0.28	5	87	0.21	5
APPLES-HYBRID	87	0.16	2	87	0.16	2
CLOSEST	54	1.15	7	58	0.91	8
DE-NOVO	98	0.02	1	92	0.08	1
(b) **Anopheles**						
APPLES*	95	0.05	1	95	0.05	1
APPLES-ME	95	0.05	1	95	0.05	1
APPLES-HYBRID	95	0.05	1	95	0.05	1
CLOSEST	91	0.09	1	95	0.05	1
DE-NOVO	95	0.05	1	95	0.05	1
(c) **Drosophila**						
APPLES*	71	0.29	1	71	0.33	2
APPLES-ME	67	0.42	2	67	0.48	2
APPLES-HYBRID	67	0.33	1	67	0.38	2
CLOSEST	57	0.62	3	57	0.57	2
DE-NOVO	71	0.29	1	71	0.33	2

Table S4. **Assembly-free placement of genome-skims.** We show the percentage of correct placements (those that do not increase Δe), average delta error (Δe), and maximum delta error (e_{max}) for APPLES, assignment to the CLOSEST species, and the placement to the position in the backbone (DE-NOVO) over the 61 (a), 22 (b), and 21 (c) placements. Results are shown for skims with 0.1 and 0.5Gbp of reads. Delta error is the increase in the number missing branches between the reference tree and the backbone tree after placing each query.

Appendix D. Commands

Sampling Clades

For sampling clades of size at most 250 from a tree "tree.nwk", we used the TreeCluster package, available at `https://github.com/niemasd/TreeCluster`.

```
#!/bin/bash
python TreeCluster/TreeCluster.py -i 250 -o clusters.txt -t tree.nwk
-m count_max_clade
```

Computing Distance Matrices

APPLES version 1.2.0 can compute JC69 distances between nucleotide sequences. Version 1.1.0 internally uses FastME* (based on FastME version 2.1.6.1) which is available at `https://github.com/balabanmetin/FastME-personal-copy`. We computed distance matrices based on other models (e.g. TN93+Γ) using following FastME command:

```
#!/bin/bash
fastme -c -dT --gamma=${gamma} -i aln_dna.phy -O dist.mat -T 1
```

where ${$*gamma*$}$ is Γ the rate variable.

Backbone tree estimation

When multiple sequence alignment is available, we used the following RAxML command to compute backbone tree for all datasets except RNAsim-VS dataset. We used RAxML version 7.2.6

```
#!/bin/bash
raxmlHPC-PTHREADS -m GTRGAMMA -p 88 -n REF -s aln_dna.phy -T 6
```

For RNAsim-VS dataset, we used FastTreeMP version 2.1.10 for estimating backbone topology. We run FastTreeMP with the following command:

```
#!/bin/bash
FastTreeMP -nosupport -gtr -gamma -nt -log tree.log < aln_dna.fa > tree.nwk
```

For alignment free datasets such as Drosophila dataset, we computed backbone tree using FastME* (based on FastME version 2.1.6.1) which is available at `https://github.com/balabanmetin/FastME-personal-copy`. FastME* is run with the following command:

```
#!/bin/bash
fastme -i dist.mat -o tree.nwk -T 1
```

Note that we performed Jukes-Cantor correction on the distance matrix "dist.mat" before running FastME*.

Backbone tree branch length re-estimation

When multiple sequence alignment is available, we used FastME* to recompute backbone tree branch lengths for all datasets except RNAsim-VS dataset. We run FastME* with the following command:

```
#!/bin/bash
fastme -dJ -i aln_dna.phy -u RAxML_result.REF -o tree_me.nwk
```

For RNAsim-VS dataset, we used RAxML version 7.2.6 for re-estimating ML based branch lengths and used that tree for performing placements using pplacer. RAxML is run with the following command:

```
#!/bin/bash
raxmlHPC-PTHREADS -f e -t tree.nwk -m GTRGAMMA -s aln_dna.phy -n REF -p 1984 -T 8
```

On the other hand, we used RAxML version 8.2.12 to compute backbone tree (branch lengths update only) and ML model parameters required for EPA-ng:

```
#!/bin/bash
raxmlHPC-PTHREADS -f e -t tree.nwk -m GTRGAMMA -s aln_dna.fa -n REF8 -p 1984 -T 8
```

For all large alignments with 10,000 or more sequences in RNASim-VS, RAxML version 8.2.12 fail to run due to estimated gamma rate being either too small or too large. In those cases, we ran the following command to use GTRCAT model instead:

```
#!/bin/bash
raxmlHPC-PTHREADS -f e -t tree.nwk -m GTRCAT -s aln_dna.fa -n REF8 -p 1984 -T 8
```

For the same dataset, we used FastTree again for re-estimating Minimum Evolution based branch lengths and used that tree for performing placements using APPLES. FastTree is run with the following command:

```
#!/bin/bash
FastTreeMP -nosupport -nt -nome -noml -log tree.log
-intree tree.nwk < aln_dna.fa > tree_me.nwk
```

Performing placement

We performed phylogenetic placement of a query using pplacer version 1.1.alpha19-0-g807f6f3 with the following command:

```
#!/bin/bash
pplacer -m GTR -s RAxML_info.REF -t backbone.nwk -o query.jplace aln_dna.fa -j 1
```

We performed EPA-ng placements using version 0.3.5 with the following command:

```
#!/bin/bash
epa-ng --ref-msa aln_dna.fa --tree backbone.nwk --query query.fa --outdir .
--model RAxML_info.REF8 --redo -T 1
```

Except in our RNASim-VS, RNASim-QS and estimated alignments analyses, we used APPLES version 1.1.0 (can be found at

844 `https://github.com/balabanmetin/apples/releases`) for placement. When alignment
845 is not present, we performed the placement running the following command:

```
#!/bin/bash
python ~/apples/least_squares_placement.py -t backbone.tree -d dist.mat -a FM
-s MLSE -p > apples.nwk
```

846 When alignment is present, we used the following command instead:

```
#!/bin/bash
python ~/apples/least_squares_placement.py -t backbone.nwk -a aln_dna.phy
-a FM -s MLSE -p > apples.nwk
```

847 For RNASim-VS, RNASim-QS and estimated alignments analyses, we used
848 APPLES version 1.2.0 and ran with the following command:

```
#!/bin/bash
python ~/apples/run_apples.py -t backbone.nwk -s aln_dna.fa -q query.fa
-T $numcores -o apples.jplace
```

849 where $*numcores* depends on the number of cores designated for the analysis.

850 *Working with estimated backbone and query-to backbone alignments*

851 We created a version of SEPP within which APPLES integrated. This version is
852 available at `https://github.com/balabanmetin/sepp`. We performed placement on
853 RNAsim-AE dataset with 10,000 sequences using SEPP+APPLES with the following
854 commmand:

```
#!/bin/bash
python ~/sepp/run_sepp.py -t estimated_backbone.nwk -a estimated_aln.fa
-r RAxML_info.REF -f query.fa -pl apples -x 28 -A 1000 -o apples
```

On the same dataset, we ran SEPP in default mode using the following commmand:

```
#!/bin/bash
python ~/sepp/run_sepp.py -t estimated_backbone.nwk -a estimated_aln.fa
-r RAxML_info.REF8 -f query.fa -pl pplacer -x 28 -A 1000  -P 1000 -o pplacer
```