APPLES: Fast Distance-based Phylogenetic Placement

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*
[*] Corresponding author: smirarab@ucsd.edu

Abstract

Phylogenetic placement consists of adding a query species onto an existing phylogeny and has increasing relevance as sequence datasets continue to grow in size and diversity. Placement is useful for updating existing phylogenies and for identifying samples taxonomically using (meta-)barcoding or metagenomics. Maximum likelihood (ML) methods of phylogenetic placement exist, but these methods are not scalable to trees with many thousands of leaves. They also rely on assembled and aligned sequences for the reference tree and the query and thus cannot analyze unassembled reads used recently in applications such as genome skimming. Here, we introduce APPLES, a distance-based method of phylogenetic placement that improves on ML by more than an order of magnitude in speed and memory and comes very close to ML in accuracy. APPLES has better accuracy than ML for placing on trees with thousands of species and can place on trees with a hundred thousands species where ML cannot run. Finally, APPLES can accurately identify samples without assembled sequences for the reference or the query using k-mer-based distances, a scenario that ML cannot handle.

Keywords: Phylogenetic placement, Distance-based methods, Genome skimming

1 Introduction

Phylogenetic reconstruction is a hard problem (often NP-Hard) [1]. Nevertheless, researchers have developed methods that can infer trees from datasets with hundreds of thousands of sequences using both maximum likelihood (ML) [2] and the distance-based [3] approaches. These large-scale reconstructions still require significant resources. As new sequences continually become available, even large trees can quickly become outdated and need to be updated. However, a *de novo* reconstruction for every new sequence is not practical. The alternative to *de novo* reconstruction is phylogenetic placement: starting with a given *backbone* tree, update it by adding new sequence(s) onto the tree. Traditionally, placement has been studied as part of greedy tree inference algorithms that start with an empty tree and add sequences sequentially [4, 5]. Each placement requires polynomial (often linear) time with respect to the size of the backbone, and thus, these greedy algorithms are scalable (often quadratic).

More recently, placement has enjoyed a renewed interest because of its new applications. An increasingly important application of phylogenetics is sample identification: given one or more *query* sequences of unknown origins, detect the identity of the organisms that could have generated that sequence. Sample identification is essential to the study of mixed environmental samples (e.g., 16S profiling [6–8] of microbiome or metagenomics [9, 10]). It is also the essence of barcoding [11] and meta-barcoding [12, 13] widely used in biodiversity research. Driven mostly by applications to microbiome profiling, two groups have developed methods for placement using ML: pplacer [14] and EPA [15]. Researchers have also developed methods for aligning query sequence (e.g., PaPaRa [16]), for alignment and placement using divide-and-conquer (e.g., SEPP [17]), and for downstream applications to metagenomics [18–20].

The available methods for phylogenetic placement have focused on the ML inference of the best placement. The ML approach suffers from two shortcomings. It is computationally demanding, especially in memory usage, and cannot place on backbone trees with many thousands of leaves. This limitation has motivated alternative methods using local sensitive hashing [21]

and divide-and-conquer [17]. A more fundamental limitation of ML methods and these faster alternatives is that they require assembled and aligned sequences for the backbone set (and often for the query sequences). However, phylogenetic placement has the potential to be used for assembly-free and alignment-free sample identification using genome skimming [22].

A genome skim is a low coverage (e.g., 1X) shut-gun sample of a genome and is not sufficient for assembling the genome. Genome skimming promises to replace traditional marker-based barcoding of biological samples [23]. As recently shown [24], using k-mers, it is possible to accurately estimate the distance between two unassembled genome skims with low coverage. Genome skims promise to enable high-resolution sample identification (e.g., at species or subspecies levels) while keeping the cost reasonable (e.g., \$50 per reference or query species). However, ML and other methods that require alignments cannot analyze genome skims, where both the reference and the query species are unassembled.

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 and other alignment-based 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 produced by tools such as Skmer [24] or other alternatives [25–32].

In this paper, 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 and allows the choice among a set of optimization criteria and phylogenetic models. We also introduce new ways to combine existing criteria, with negligible increase in the running time. Both the running time and the memory usage of our algorithm scale linearly with the size of the backbone tree. We show that when the alignment-based placement is possible, APPLES is at least an order of magnitude faster than a fast ML methods (pplacer), uses a fraction of the memory, and comes very close to ML in its accuracy. On large backbones, APPLES has much higher accuracy than pplacer and, unlike ML, scales easily to datasets with hundreds of thousands of species. On a Drosophila dataset, we show that APPLES+Skmer can accurately perform assembly-free and alignment-free sample identification using low coverage genome skims. The code and examples are available online at https://github.com/balabanmetin/apples.

2 The APPLES algorithm

Background. Given an observed $n \times n$ matrix of pairwise sequence distances Δ, where entries δ_{ij} indicate the dissimilarity between species i and j, distance-based methods infer a tree with branch lengths. A tree T also defines a distance matrix where each entry $d_{ij}(T)$ corresponds to the path length between leaves i and j. If the input matrix matches any tree, it matches a *unique* tree and is called an additive matrix [33]. When sequences evolve on a phylogenetic tree, their Hamming distance is not additive even asymptotically. However, we can define phylogenetic distances that converge to additivity as sequence lengths increase [34]. For example, under the JC69 model [35], for hamming distance h, $\frac{3}{4}\ln(1-\frac{4}{3}h)$ is asymptotically additive.

While finding the tree matching an additive distance matrix is easy [33, 36], on limited data, distances are generally not additive. Instead, we need to solve an optimization problem to find the tree that best matches the input matrix. A natural optimization is least square errors:

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

where w_{ij} are weights used to reduce the impact of large distances (expected to have high variance) on the error. Standard ways to define weights include: $w_{ij} = 1$, the ordinary least squares (OLS) method of Cavalli-Sforza and Edwards [37], $w_{ij} = 1/\delta_{ij}^2$ due to Fitch and Margoliash [38] (FM), and $w_{ij} = 1/\delta_{ij}$ due to Beyer *et al.* [39] (BE). Finding $\arg\min_T Q^*(T)$ is NP-Complete

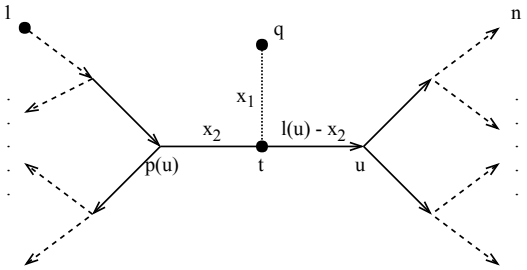


Figure 1: Any placement of q can be characterized as a tree $P(u, x_1, x_2)$, shown here. Backbone is an arborescence on leaves $\mathcal{L} = \{1 \dots n\}$, rooted at leaf 1. Query taxon q is added on the edge between u and $p(u)$, creating a node t. All placements on this edge are characterized by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

[40]. However, heuristic solutions like neighbor joining [41], alternative problems like (balanced) minimum evolution [5, 37, 42], and several tools (e.g., FastME [3] and Ninja [43]) exist.

2.1 Problem definition

Notations. Let an unrooted tree be represented as a weighted connected acyclic undirected graph $T = (V, E)$ with leaves denoted by $\mathcal{L} = \{1 \cdots n\}$. We let T^* be the arborescence of T rooted at leaf 1, i.e. all edges on the path between 1 and u is directed towards u for any $u \in V \setminus \{1\}$. Each node u in T^*, except the root, has a parent, $p(u)$, a set of children $c(u)$, and a set of siblings $sib(u) = \{v \in V \setminus \{u\} | p(v) = p(u)\}$. We let $g(u)$ denote the set of leaves that include u in their path to the root (i.e., its child leaves). We let $l(u)$ denote the length of the edge $(p(u), u)$. We let $P(u, x_1, x_2)$ be the tree obtained by adding the query taxon q on an edge $(p(u), u)$, creating three edges (t, q), $(p(u), t)$, and (t, u), with weights x_1, x_2, and $l(u) - x_2$, respectively (Fig. 1). When clear by the context, we simply write P and note that P *induces* T both in topology and branch length. We generalize the definition of $d_{uv}(T^*)$ to indicate the length of the undirected path between any two nodes and when clear from the context, we simply write d_{uv}. Finally, we always use $l(u)$, $g(u)$, $p(u)$, $c(u)$, and $sib(u)$ with respect to T^*.

To place a *query* sequence on an existing backbone phylogeny using distances, we define:

Least Squares Phylogenetic Placement (LSPP) problem:
Input : A backbone tree T on $\mathcal{L}$, a query species q, and a vector Δ_{q*} with elements δ_{qi} giving distances between q and every species $i \in \mathcal{L}$;
Output: The placement tree P that adds q on T and minimizes

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

It can be easily checked that minimizing Eq. 2 is equivalent to minimizing Eq. 1 if the backbone tree is fixed. Following the literature, we instantiate this general problem by setting $w_{qi} = 1$ (OLS [37]), $w_{qi} = 1/\delta_{qi}^2$ (FM [38]), or $w_{qi} = 1/\delta_{qi}$ (BE [39]).

2.2 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 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, we can use constant time for testing each placement after a $\Theta(n)$ precomputation.

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

We now sketch the proof and describe the method. For a general weighting scheme $w_{qi} = \delta_{qi}^{-k}$

for a *constant* $k \in \mathbb{N}$ (e.g., $k = 0, 1$, or 2 for OLS, BE, or FM), Eq. 2 can be re-written as:

$$Q(P) = \sum_{i \in g(u)} \delta_{qi}^{-k} \big(\delta_{qi} - d_{ui}(P) - x_1 + x_2 - l(u)\big)^2 + \sum_{i \notin g(u)} \delta_{qi}^{-k} \big(\delta_{qi} - d_{p(u)i}(P) - x_1 - x_2\big)^2 \tag{3}$$

We need to compute Eq. 3 for all possible placement edges. Computing the equation in its current form requires linear time per node, which will give us a quadratic time algorithm. However, we can compute Eq. 3 in constant time for any placement P after a linear time preprocessing using dynamic programming. To describe this algorithm, we need some new notations. For any node u and exponents $a \in \mathbb{Z}$ and $b \in \mathbb{N}^+$, let $S(a, b, u) = \sum_{i \in g(u)} \delta_{qi}^a d_{ui}^b$ and for $b = 0$, let $S(a, 0, u) = S'(a, u) = \sum_{i \in g(u)} \delta_{qi}^a$. Note: $S'(0, u) = |g(u)|$. 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 $R(a, 0, u) = R'(a, u) = \sum_{i \notin g(u)} \delta_{qi}^a$.

As shown in Eq. S1 of Appendix A, we can rearrange Eq. 3 such that computing $Q(P)$ requires only computing $S(a, b, u)$ and $R(a, b, u)$ values for $-k \le a \le 2 - k$ and $0 \le b \le 2$. Moreover, as Appendix A proves, $S(a, b, u)$ and $R(a, b, u)$ can be computed using recursion:

$$S(a,b,u) = \begin{cases} \delta_{qu}^a & u \in \mathcal{L} \setminus \{1\} \ \ \& \ \ b = 0 \\ 0 & u \in \mathcal{L} \setminus \{1\} \ \ \& \ \ b \neq 0 \\ \sum\limits_{j=0}^{b} \sum\limits_{v \in c(u)} l(v)^j \binom{b}{j} S(a, b-j, v) & u \notin \mathcal{L} \setminus \{1\} \end{cases} \tag{4}$$

$$R(a,b,u) = \begin{cases} \delta_{q1}^a & u = 1' = c(1) \quad \& \ \ b = 0 \\ 0 & u = 1' = c(1) \quad \& \ \ b \neq 0 \\ \sum\limits_{j=0}^{b} l(p(u))^j \binom{b}{j} R(a, b-j, p(u)) + \sum\limits_{j=0}^{b} \sum\limits_{v \in sib(u)} l(v)^j \binom{b}{j} S(a, b-j, v) & u \notin \{1, 1'\} \end{cases} \tag{5}$$

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 and therefore are constant.

To summarize, after two traversals of the tree, we have all the $S(a, b, u)$ and $R(a, b, u)$ values and we can compute $Q(P(u, x_1, x_2))$ in constant time. To find the optimal placement edge, we first optimize $Q(P(u, x_1, x_2))$ for all $u \in V \setminus \{1\}$ and then find $\arg\min_u Q(P(u, x_1, x_2))$ by plugging optimal x_1 and x_2 for each edge in $Q(P(u, x_1, x_2))$. Thus, to have a linear time algorithm, we need to find x_1 and x_2 that minimize $Q(P)$ for a fixed u in constant time. We take the derivative of Eq. 3 with respect to x_1 and x_2, and set the derivative to zero. This gives us a system of equations that can be trivially solved in constant time to find unique x_1 and x_2, except if $\delta_{qi} \le 0$ for some i (proved in Appendix A).

Lemma 2. *For a fixed node $u \in V \setminus \{1\}$, $\arg\min_{x_1, x_2} Q(P(u, x_1, x_2))$ satisfies*

$$\begin{bmatrix} R'(-k,u) + S'(-k,u) & R'(-k,u) - S'(-k,u) \\ R'(-k,u) - S'(-k,u) & R'(-k,u) + S'(-k,u) \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{6}$$

$$c_1 = R'(1-k,u) + S'(1-k,u) - l(u)S'(-k,u) - R(-k,1,u) - S(-k,1,u)$$
$$c_2 = R'(1-k,u) - S'(1-k,u) + l(u)S'(-k,u) - R(-k,1,u) + S(-k,1,u)$$

When there is an i with $\delta_{qi} = 0$, we make q sister to i, breaking ties arbitrarily.

2.3 Non-negative branch lengths

The solution to Eq 6 does not necessarily conform to constraints $0 \le x_1$ and $0 \le x_2 \le l(u)$. However, the following lemma allows us to easily impose the constraints by choosing optimal boundary points when unresticted x_1 and x_2 fall outside boundaries.

Lemma 3. *With respect to variables x_1 and x_2, $Q(P(u, x_1, x_2))$ is a convex function.*

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 $\delta_{qi} \geq 0$ 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. □

2.4 Alternative optimization problems: minimum evolution and hybrid

An alternative to directly using MLSE (Eq. 1) is the minimum evolution (ME) principle [37, 44]. 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.

Phylogenetic insertion using the ME criterion has been previously studied for the purpose of creating an algorithm for greedy minimum evolution (GME) [5]. Desper and Gascuel 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 has a subtle but consequential difference from our ME placement. Their algorithm does not compute 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), precomputed and saved in the aforementioned data structure. In the context of the greedy algorithm, 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 the scratch would still require $\Theta(n^2)$. Thus, computing the placement for a new query would not be possible in linear time, except 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. This allows us to develop a linear time algorithm, starting from a tree of size n.

Hybrid optimization: combining MLSE and ME. In one of our tests, reported below, we observed that 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 list of edges, we place the query on the edge satisfying the ME criteria (i.e., the one with the minimum x_1).

3 Experimental setup

We evaluate APPLES in two settings: placement on gene trees using aligned sequences and sample identification using assembly-free genome skims. For aligned data, we compare APPLES to ML placement; for the assembly-free setting, existing placement methods cannot be used.

3.1 Datasets

We use two simulated datasets that provide full-length aligned sequences and a third dataset that consists of sets of reads from real genomes, used for testing the assembly-free scenario.

GTR. We use an existing simulated dataset with 101 species [45]. Sequences were simulated under the General Time Reversible (GTR) model and the Γ model of site rate heterogeneity using INDELible [46] on gene trees that were simulated using SimPhy under the coalescent model evolving on species trees generated under the Yule model. Thus, this dataset is simulated using the same model used by ML placement methods that use GTR+Γ (no model misspecification). We took all 20 replicates of this dataset with mutation rates between 5×10^{-8} and 2×10^{-7}

per generation; we then randomly selected 5 estimated gene trees among those with ≤20% RF distance between estimated and true gene tree. This gave us 100 backbone trees of size 100.

RNASim. RNASim [47] is 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 gives an inhomogeneous stochastic process without a global substitution matrix. Due to its complexity, RNASim is challenging but also realistic because it allows us to test both ML and APPLES under a substantially misspecified model. An RNASim dataset of 10^6 sequences is available from prior publications [48, 49]. We use this dataset to create three sets of datasets by subsampling the full RNASim dataset in order to test both accuracy and scalability.

i) Heterogeneous: We first randomly subsampled the full dataset to create 10 datasets of size 10^4. 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: We randomly subsampled the tree of size 10^6 to create 5 replicates of datasets of size 5×10^2, 10^3, 5×10^3, 10^4, 5×10^4, 10^5, and 2×10^5 (1 replicate). For replicates that contain more than or equal to 5×10^3 species, we filtered the sites that contain gaps in at least 95% of the sequences in the alignment.

Drosophila genome skims. For the reference set, we used a dataset of 21 *Drosophila* genomes (Table S1) downloaded from InsectBase [50], and genome-skims with 0.1Gb, 0.5Gb, and 1Gb sequence were generated from the assemblies using the short-read simulator tool ART [51], with read length $l = 100$ and default error profile. For the query sequences, we used a different dataset [52] from an independent study [53] with high coverage short reads (SRAs). This dataset includes 12 species in common with our reference set. Since the data come from two projects, the query genome skim can have a non-zero distance to the same species in the reference set. We used Megablast [54] to search reads against Drosophila mitochondrial genomes and a dataset of microbial genomes (Table S2) known to be found in *Drosophila* [55–57] and filtered out the hits. We also removed human contamination using Bowtie2 [58]. We randomly subsampled reads, before filtering, to create genome skims with 0.1Gb, 0.5Gb, or 1Gb of data (10 replicates). Since Drosophila species have different genome sizes, with 0.1Gb data, our subsampled genome skims range in coverage from 0.4X to 0.8X, and coverage proportionally scales for 0.5Gb and 1Gb.

3.2 Methods

We compare APPLES to the ML approach implemented in pplacer. We chose pplacer [14] instead of EPA [15] because the two tools have similar accuracy but pplacer is substantially faster [14]. We run pplacer in its default mode using GTR+Γ model (the only option). 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. We were not able to test other methods. LSHPlace is a fast algorithm [21], but we could not find any implementation of it. The distance-based insertion algorithm of FastME [5] is available only as part of the greedy algorithm but is not available as a stand-alone feature to place on a given tree. SEPP [17] performs alignment+placement, but our goal is only to test the placement step and not the alignment.

Distance calculation. We modified the FastME code (FastME* hereafter) to compute distances only between query and backbone sequences (not among backbone sequences). FastME*

also ensures that when estimating model parameters, positions that have a gap in at least one of the two sequences are always ignored. We compute phylogenetic distances under the parameter-free JC69 [35] model, the six-parameter TN93 [59] model, and the 12-parameter general Markov model [34]. We also pair JC69 and TN93 with the Gamma model of sites rate heterogeneity using the standard approach [60]. Pairing Gamma and GTR is theoretically possible for infinitely long sequences; however, the method can run into problems on real data [60]. 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 [34]; however, log-det cannot account for rate across sites heterogeneity[60]. The α parameter of the Gamma model cannot be computed from pairwise sequence comparisons [61]; however, in the placement setting, we simply use the parameter computed from the backbone tree. We used the α parameter computed by RAxML [62] run on the backbone alignment and given the backbone tree. In our analysis on assembly-free datasets, we first compute the genomic distance using Skmer [24]. We then correct these distances assuming the JC69 model. For genome skims, our data consists only of estimated hamming distances; thus, JC69, without Gamma model of variable rates, is the only model we can use.

Backbone trees. Using FastME*, we estimated the backbone tree from the JC69 distance matrix of the reference set of 21 genome skims (Fig. S1). For simulated datasets, we estimate the topology of backbone tree by running RAxML [62] on the true alignment using GTRGAMMA model and use this tree as the backbone for pplacer. To handle large trees, we used FastTree-2 [2] to estimate backbone tree for RNASim-varied size and re-estimated branch lengths on the fixed topology using RAxML. For the backbone of APPLES, we always used the same tree topology as pplacer but re-estimated branch lengths using FastTree-2 under the JC69 model.

APPLES*. The default version of APPLES, referred to as APPLES*, uses the FM weighting scheme, the MLSE selection criterion, enforcement of non-negative branch lengths, and JC69 distances. When not specified, these default parameters are used, with one exception. Based on results shown below, we always enforce non-negative x_1 and x_2 constraints for MLSE but we never enforce it for ME.

3.3 Evaluation

To evaluate the accuracy, 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. We measure the accuracy of the placement using delta error (Δe) [17]: the number of branches of the true tree missing from P minus the number of branches of the true tree missing from $T \setminus i$ (induced on the same leafset). Note that $\Delta e \geq 0$ because adding i cannot decrease the number of missing branches in $T \setminus i$. We used this leave-one-out strategy for testing accuracy on GTR, RNAsim-heterogeneous, and RNAsim-varied diameter datasets. However, due to their large size, we only removed and added back 200 randomly chosen leaves per replicate on the RNAsim-varied size dataset. For the real Drosophila dataset, for each of the 12 query species, we remove the corresponding species from the 21-species backbone tree before placing it and measuring the error (for 10 replicate skims).

4 Results

4.1 Assembly-free and Alignment-free placement of genome skims

APPLES is able to correctly place each of the 12 query Drosophila genome skims when 0.5Gbp or 1Gbp of data are available using ME or Hybrid optimization criteria (Table 1). With only 0.1Gbp of reads, APPLES-Hybrid has the best accuracy and is wrong in one case, and then, only by one branch. This recalcitrant species is *Drosophila eugracilis*, which is placed on an extremely short branch of length 10^{-4} surrounded by other short branches (Fig. S1). In contrast to APPLES, choosing the closest branch is mislead by as much as seven branches (mean error:

	0.1G			0.5G			1G		
	%	Δe	e_{max}	%	Δe	e_{max}	%	Δe	e_{max}
APPLES-MLSE	75	0.3	1	83	0.2	1	83	0.2	1
APPLES-ME	92	0.2	2	100	0	0	100	0	0
APPLES-HYBRID	92	0.1	1	100	0	0	100	0	0
CLOSEST	83	0.57	5	83	0.64	7	83	0.64	7

Table 1: Assembly-free placement of genome skims. Over the 12 placements, we show percentage of correct placements (shown as %), average delta error (Δe), and maximum delta error (e_{max}) for APPLES and assignment to the CLOSEST species. Results are shown for skims with 0.1, 0.5, and 1Gbp of reads (Drosophila genome sizes range between 133Mbp to 224Mbp). The query sequences are from a separate dataset from the backbone dataset and include 12 out of 21 species of Drosophila in the backbone tree. The subsampling of query sequences is repeated 10 times, all resulting in identical results.

	Low		Medium		High		Heterogeneous	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
APPLES*	0.15	0.39	0.18	0.50	0.18	0.44	0.17	0.44
closest	0.88	1.57	0.88	1.60	0.85	1.52	0.87	1.56
pplacer	0.13	0.35	0.11	0.34	0.13	0.36	0.13	0.35

Table 2: The delta error for APPLES* , closest match, and pplacer on the RNASim-varied diameter dataset (low, medium, or high) and the RNASim-heterogeneous dataset. Mean and standard error is shown over 1250 placements, corresponding to 5 backbone trees and 250 placements per replicate.

0.6 branches) even at higher coverage levels. Interestingly, APPLES-MLSE is correct *less* often than APPLES-ME but also has a lower maximum error.

4.2 Alignment-based placement

We first compare the accuracy and scalability of APPLES* to other methods and then show analyses comparing other settings of APPLES to APPLES*.

Comparison to Maximum Likelihood (ML)

No model misspecification. On the GTR 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 an under-parameterized model, the accuracy of the two methods is within 0.1 edges on average. In contrast, using the closest match results in poor accuracy (correct only 50% of the times and mean $\Delta e = 1.0$ edges).

Model misspecification. On the small RNASim data (subsampled clades of ≈ 250 species), where both APPLES* and pplacer face model misspecification, the accuracy of APPLES* is extremely close to ML using pplacer. On the heterogeneous subset (Fig. 2b and Table 2), pplacer and APPLES* find the best placement in 88% and 86% of cases and have a mean delta error of 0.13 and 0.17 edges, respectively. Both methods are much more accurate than using the closest match, which has a delta error of 0.9 edges.

Impact of diameter. When we control the tree diameter (low, medium, or high), APPLES* and pplacer remain very close in accuracy (Fig. 2c). The changes in error are small and not

	n = 500		n = 10^3		n = 5×10^3		n = 10^4		n = 10^5		n = 2×10^5	
	%	Δe	%	Δe	%	Δe	%	Δe	%	Δe	%	Δe
APPLES*	75	0.33	75	0.36	77	0.36	80	0.31	85	0.24	86	0.31
closest	52	1.12	56	1.14	56	0.97	59	0.89	62	0.65	66	0.56
pplacer	81	0.20	82	0.20	2	15.09	–	–	–	–	–	–

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 shown over 1000 placements (except $n = 2 \times 10^5$, which is shoen over 200 placements).

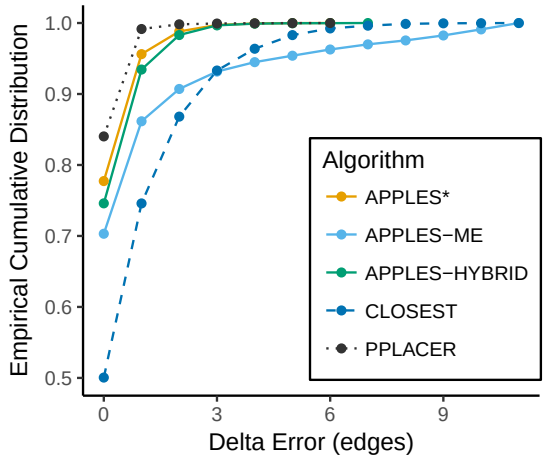
monotonic as the diameters change (Table 2). The two methods are most divergent in the medium diameter case, where pplacer has its lowest error (Δe =0.11 edges) and APPLES* has its highest error (Δe =0.18 edges). Interestingly, the error of both methods at low and high diameter levels are very close to each.

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.

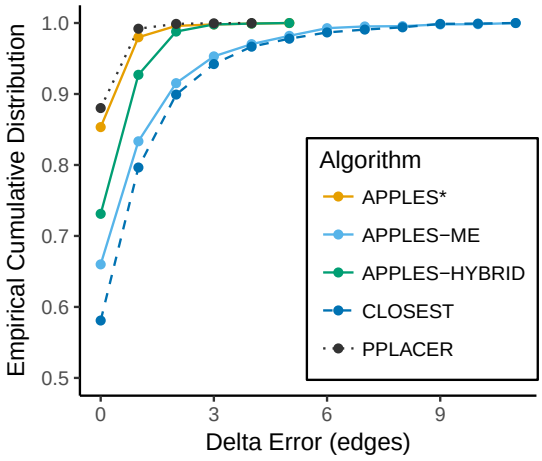
Impact of taxon sampling. The real advantage of APPLES* over pplacer becomes clear for placing on larger backbone trees (Fig. 2d and Table 3). For backbone sizes of 500 and 1000, pplacer continues to be slightly more accurate than APPLES* (mean Δe of pplacer is better than APPLES* by 0.13 and 0.16 edges, respectively). However, with backbones of 5000 leaves, pplacer fails to run on some cases and has 41 times higher error than APPLES* on the rest (perhaps due to numerical issues). For backbones trees with at least 10^4 leaves, pplacer was not able to run and closest is not very accurate (finding the best placement in only 62% of cases). However, APPLES* continues to be accurate for all backbone sizes. Importantly, 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). We observe that as the backbone size increases, APPLES* has reduced error despite placing on larger trees (Table 3). For example, using a backbone tree of 10^5 leaves, APPLES* is able to find the best placement of query sequences in 85% of cases, which is better than the accuracy of either APPLES* or pplacer on any backbone size. Thus, an increased taxon sampling helps accuracy, but pplacer has limits on the size of the tree it can handle.

Running time and memory.

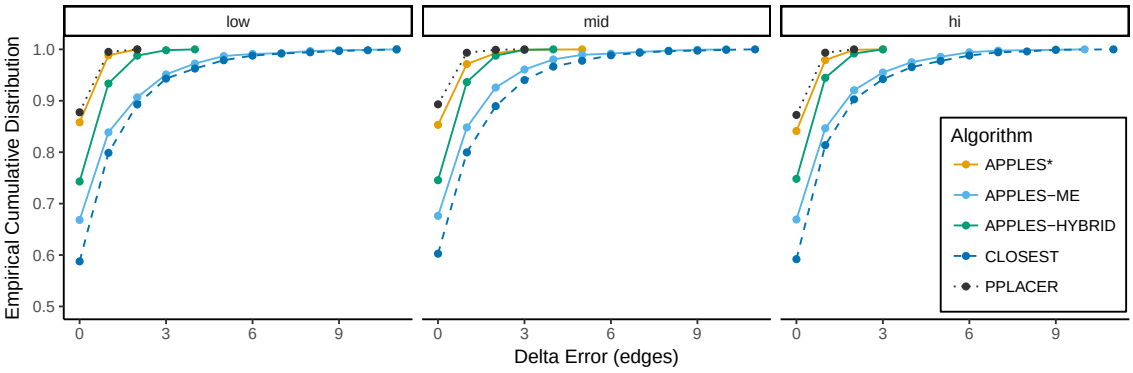
As noted before, on backbones with 10^4 or more leaves, pplacer could not run (producing infinity likelihood on backbones of size 10^4). As the backbone size increases, the running time of pplacer and APPLES increase close to linearly with the size of the backbone tree (Fig. 3a). However, APPLES is on average 18 times faster than pplacer on a backbone tree with 5000 leaves. Similarly, the memory of both methods increases linearly with the backbone size, but APPLES requires dramatically less memory (Fig. 3b). For example, for placing on a backbone with 5000 leaves, pplacer requires 25GB of memory whereas APPLES requires only 0.25GB. APPLES easily scales to a backbone of 10^5 sequences, running in only 1.8 minutes and using 13GB 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. The time and memory used by the placement algorithm itself are 0.7 minutes and 5GB for the tree with 10^5 leaves.



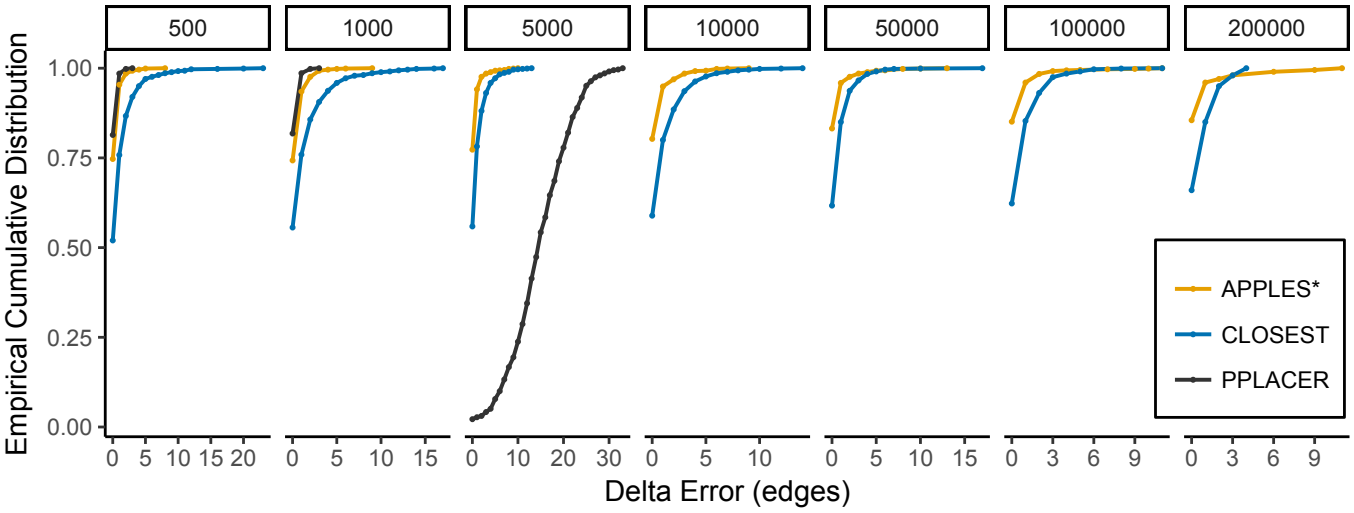
(a)



(b)

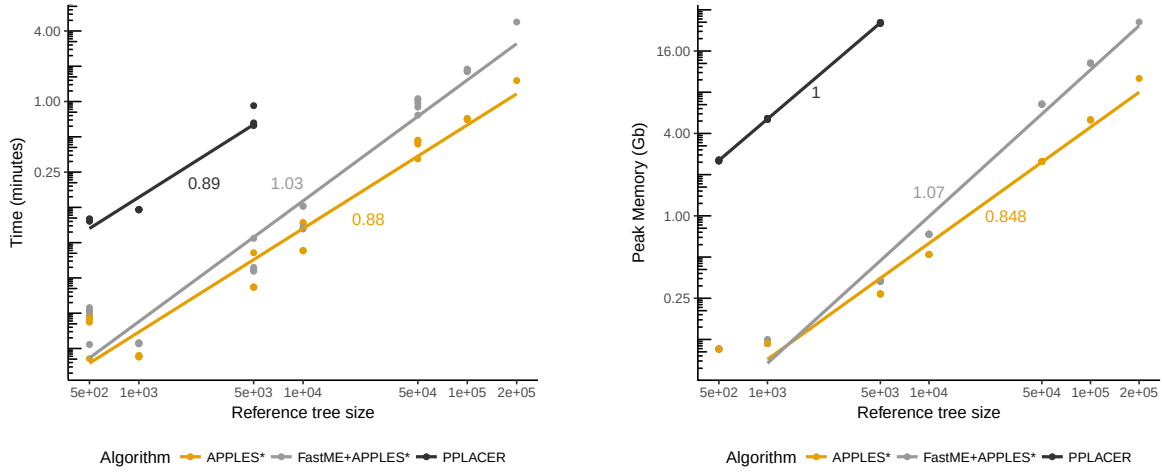


(c)



(d)

Figure 2: **APPLES versus pplacer.** The 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 (dashed), closest match (dotted), and APPLES with FM weighting and JC69 distances and MLSE (APPLES*), ME, or Hybrid optimization. (a) GTR dataset. (b) RNASim-Heterogeneous. (c) RNASim-varied backbone diameter: low, medium, or high (d) RNASim- varied backbone tree size, ranging from 500 to 2×10^5. Distributions are over 10^4 (a), 2450 (b), 3675 (c), and 1000 (d) points. pplacer on backbones of size 5000 managed to run for only 551 out of 1000 placements (producing infinity likelihood in other cases), and we show results over cases where it did manage to run.



(a) (b)

Figure 3: Comparison of running time (a) and peak memory usage (b) between PPLACER and APPLES*for a single placement. Both methods are tested on the varied size RNAsim datasets (x-axis: size of the backbone tree). Pplacer fails to run for backbones of 10^4 leaves or more. For APPLES, measurements are shown with and without the distance calculation step performed using FastME*. Lines are fitted in the log-log scale; thus, the slope of the line (indicated on the figure) gives an empirical estimate of the polynomial degree of the asymptotic growth curve. All curves grow close to linearly (slopes $\approx$1). All calculations are on 8-core, 2.6GHz Intel Xeon CPUs (Sandy Bridge) with 64GB of memory.

Comparing versions of APPLES

We now compare different settings of APPLES. First, comparing the five models of sequence evolution, we see similar patterns of accuracy across all models despite their varying complexity (Fig. S3). Since the JC69 model is parameter-free and results in similar accuracy to others, we use 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 increases the accuracy for MLSE but reduces the accuracy for ME (Fig. S4). Thus, from now on, we always impose non-negative constraints on MLSE but not on ME. Consistent with this, our default Hybrid method includes the constraint for the first MLSE step but not for the following ME step (Fig. S4). 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. S2). The OLS scheme, which does not penalize long distances, performs substantially worse than FM and BE. Fixing the weighting scheme to FM and comparing the three optimization strategies (MLSE, ME, and Hybrid), the MLSE approach has the best accuracy (Fig. 2b), finding the correct placement 84% of the time (mean error: 0.18), and 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 (unlike the assembly-free dataset).

5 Discussions and future work

We introduced APPLES: a new method for adding query species onto large backbone trees using both unassembled genome skims and aligned data. The accuracy of APPLES was very close to ML using pplacer in most settings where ML could run but surpassed it when backbones were large. The accuracy advantages of ML were especially small for the more realistic simulation, RNASim, where both methods face model misspecification. As expected by the strong evidence in the literature [63, 64], improved taxon sampling increased the accuracy of placement. This observation is a main motivation for using scalable methods such as APPLES* instead of ML

methods, which have to restrict their backbone to at most several thousand species. It is possible to follow up the APPLES* placements with a round of ML placement on smaller trees, but the small differences in accuracy of pplacer and APPLES* did not give us compelling reasons to try such hybrid approaches.

Our comparisons between versions of APPLES answered many questions but left others to future work. *i*) We observed that the Hybrid approach of APPLES had the best accuracy in our tests on assembled data whereas simple MLSE was the most accurate for aligned sequences. The reasons for this disparity are not clear and may be related to other factors such as the size of the dataset, the error in the backbone, or level of divergence from a strict molecular clock. *ii*) We observed no advantage in using models more complex than JC69; however, this may be due to our re-estimation of model parameters (e.g., base compositions) for each pair of sequences. More complex models may perform better if we simply estimate model parameters from the backbone and reuse them for queries. *iii*) In our assembly-free test, we used Skmer [24] to get distances because alternative alignment-free methods of estimating distance generally either require assemblies [e.g., 25–27] or higher coverage than Skmer [e.g., 28, 29, 31]; 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). *iv*) Finally, in the aligned use-case, we used the true alignments in all the simulation tests and left an exploration of the impact of alignment error on different methods to future work. On a related note, APPLES can be combined with SEPP to perform alignment and placement in a unified pipeline.

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. S5 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 very large trees in a scalable fashion, without a need for quadratic time/memory computation of distance matrices [2].

APPLES was more than an order of magnitude faster and less memory-hungry than pplacer, but it has room for improvement. The python APPLES code is not optimized and can dramatically improve. For example, APPLES can save precomputed values of Equations 4 and 5 for each backbone tree in a file, eliminating the need to recompute them. Also, online processing of the backbone alignment can dramatically reduce the memory usage for the distance calculation. Future implementations of APPLES will improve speed and memory by applying such optimizations.

Acknowledgments. This work was supported by the National Science Foundation (NSF) grant IIS-1565862 and NIH subaward 5P30AI027767-28 to M.B. and S.M., and NSF grant NSF-1815485 to 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

[1] T. Warnow, *Computational phylogenetics: An introduction to designing methods for phylogeny estimation* (Cambridge University Press, 2017), ISBN 1316886921.

[2] M. N. Price, P. S. Dehal, and A. P. Arkin, FastTree-2 Approximately Maximum-Likelihood Trees for Large Alignments, *PLoS ONE* **5**, e9490 (2010), ISSN 1932-6203.

[3] V. Lefort, R. Desper, and O. Gascuel, FastME 2.0: A comprehensive, accurate, and fast distance-based phylogeny inference program, *Molecular Biology and Evolution* **32**, 2798 (2015), ISSN 15371719.

[4] J. Felsenstein, Evolutionary trees from DNA sequences: A maximum likelihood approach, *Journal of Molecular Evolution* **17**, 368 (1981), ISSN 0022-2844.

[5] R. Desper and O. Gascuel, Fast and Accurate Phylogeny Reconstruction Algorithms Based

on the Minimum-Evolution Principle, *Journal of Computational Biology* **9**, 687 (2002), ISSN 1066-5277.

[6] S. R. Gill, M. Pop, R. T. DeBoy, P. B. Eckburg, P. J. Turnbaugh, B. S. Samuel, J. I. Gordon, D. A. Relman, C. M. Fraser-Liggett, and K. E. Nelson, Metagenomic Analysis of the Human Distal Gut Microbiome, *Science* **312**, 1355 (2006), ISSN 1095-9203.

[7] L. Krause, N. N. Diaz, A. Goesmann, S. Kelley, T. W. Nattkemper, F. Rohwer, R. A. Edwards, and J. Stoye, Phylogenetic classification of short environmental DNA fragments., *Nucleic acids research* **36**, 2230 (2008), ISSN 1362-4962.

[8] S. Janssen, D. McDonald, A. Gonzalez, J. A. Navas-Molina, L. Jiang, Z. Z. Xu, K. Winker, D. M. Kado, E. Orwoll, M. Manary, et al., Phylogenetic Placement of Exact Amplicon Sequences Improves Associations with Clinical Information, *mSystems* **3**, 00021 (2018), ISSN 2379-5077.

[9] C. von Mering, P. Hugenholtz, J. Raes, S. G. Tringe, T. Doerks, L. J. Jensen, N. Ward, and P. Bork, Quantitative Phylogenetic Assessment of Microbial Communities in Diverse Environments, *Science* **315**, 1126 (2007), ISSN 0036-8075.

[10] J. C. Venter, K. Remington, J. F. Heidelberg, A. L. Halpern, D. Rusch, J. A. Eisen, D. Wu, I. Paulsen, K. E. Nelson, W. Nelson, et al., Environmental genome shotgun sequencing of the Sargasso Sea., *Science (New York, N.Y.)* **304**, 66 (2004), ISSN 1095-9203.

[11] P. D. N. Hebert, A. Cywinska, S. L. Ball, and J. R. deWaard, Biological identifications through DNA barcodes, *Proceedings of the Royal Society B: Biological Sciences* **270**, 313 (2003), ISSN 0962-8452, arXiv:0005074v1.

[12] L. J. Clarke, J. Soubrier, L. S. Weyrich, and A. Cooper, Environmental metabarcodes for insects: In silico PCR reveals potential for taxonomic bias, *Molecular Ecology Resources* **14**, 1160 (2014), ISSN 17550998.

[13] A. Bush, R. Sollmann, A. Wilting, K. Bohmann, B. Cole, H. Balzter, C. Martius, A. Zlinszky, S. Calvignac-Spencer, C. A. Cobbold, et al., Connecting Earth observation to high-throughput biodiversity data, *Nature Ecology & Evolution* **1**, 41559 (2017), ISSN 2397-334X.

[14] F. A. Matsen, R. B. Kodner, and E. V. Armbrust, pplacer: linear time maximum-likelihood and Bayesian phylogenetic placement of sequences onto a fixed reference tree., *BMC bioinformatics* **11**, 538 (2010), ISSN 1471-2105.

[15] S. A. Berger, D. Krompass, A. Stamatakis, D.K., A. Stamatakis, D. Krompass, and A. Stamatakis, Performance, Accuracy, and Web Server for Evolutionary Placement of Short Sequence Reads under Maximum Likelihood, *Systematic Biology* **60**, 291 (2011), ISSN 1076-836X.

[16] S. a. Berger and A. Stamatakis, Aligning short Reads to Reference Alignments and Trees., *Bioinformatics* **27**, 2068 (2011), ISSN 1367-4811.

[17] S. Mirarab, N. Nguyen, and T. Warnow, SEPP: SATé-Enabled Phylogenetic Placement., *Pacific Symposium On Biocomputing* 247–58 (2012), ISSN 17935091.

[18] N. N.-p. Nguyen, S. Mirarab, B. B. Liu, M. Pop, and T. Warnow, TIPP: Taxonomic Identification and Phylogenetic Profiling, *Bioinformatics* **30**, 3548 (2014), ISSN 1367-4803.

[19] M. Stark, S. A. Berger, A. Stamatakis, and C. von Mering, MLTreeMap–accurate Maximum Likelihood placement of environmental DNA sequences into taxonomic and functional reference phylogenies., *BMC genomics* **11**, 461 (2010), ISSN 1471-2164.

[20] F. A. Matsen and S. N. Evans, Edge Principal Components and Squash Clustering: Using the Special Structure of Phylogenetic Placement Data for Sample Comparison, *PLoS ONE* **8** (2013), ISSN 19326203.

[21] D. G. Brown and J. Truszkowski, LSHPlace: fast phylogenetic placement using locality-sensitive hashing., *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* 310–9 (2013), ISSN 2335-6936.

[22] S. Dodsworth, Genome skimming for next-generation biodiversity analysis, *Trends in Plant Science* **20**, 525 (2015), ISSN 13601385.

[23] E. Coissac, P. M. Hollingsworth, S. Lavergne, and P. Taberlet, From barcodes to genomes: extending the concept of DNA barcoding, *Molecular Ecology* **25**, 1423 (2016), ISSN 09621083.

[24] S. Sarmashghi, K. Bohmann, M. T. P. Gilbert, V. Bafna, and S. Mirarab, Assembly-free and alignment-free sample identification using genome skims, *bioRxiv (abstract appeared at RECOMB 2018)* 230409 (2018), ISSN 16113349.

[25] B. Haubold, Alignment-free phylogenetics and population genetics, *Briefings in Bioinformatics* **15**, 407 (2014), ISSN 1467-5463.

[26] C.-A. Leimeister and B. Morgenstern, kmacs: the k -mismatch average common substring approach to alignment-free sequence comparison, *Bioinformatics* **30**, 2000 (2014), ISSN 1460-2059.

[27] C.-A. Leimeister, S. Sohrabi-Jahromi, and B. Morgenstern, Fast and accurate phylogeny reconstruction using filtered spaced-word matches, *Bioinformatics* btw776 (2017), ISSN 1367-4803.

[28] H. Yi and L. Jin, Co-phylog: an assembly-free phylogenomic approach for closely related organisms, *Nucleic Acids Research* **41**, e75 (2013), ISSN 1362-4962.

[29] G. Benoit, P. Peterlongo, M. Mariadassou, E. Drezen, S. Schbath, D. Lavenier, and C. Lemaitre, Multiple comparative metagenomics using multiset k -mer counting, *PeerJ Computer Science* **2**, e94 (2016), ISSN 2376-5992.

[30] H. Fan, A. R. Ives, Y. Surget-Groba, and C. H. Cannon, An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data, *BMC Genomics* **16**, 522 (2015), ISSN 1471-2164.

[31] B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy, Mash: fast genome and metagenome distance estimation using MinHash, *Genome Biology* **17**, 132 (2016), ISSN 1474-760X.

[32] C. Jain, L. M. Rodriguez-R, A. M. Phillippy, K. T. Konstantinidis, and S. Aluru, High-throughput ANI Analysis of 90K Prokaryotic Genomes Reveals Clear Species Boundaries, *bioRxiv* (2017).

[33] P. Buneman, A note on the metric properties of trees, *Journal of Combinatorial Theory, Series B* **17**, 48 (1974), ISSN 00958956.

[34] P. J. Lockhart, M. A. Steel, M. D. Hendy, and D. Penny, Recovering evolutionary trees under a more realistic model of sequence evolution., *Molecular Biology and Evolution* **11**, 605 (1994), ISSN 1537-1719.

[35] T. H. Jukes and C. R. Cantor, in *In Mammalian protein metabolism, Vol. III (1969), pp. 21-132* (1969), vol. III, 21–132.

[36] P. Erdos, M. A. Steel, L. Székely, and T. Warnow, A few logs suffice to build (almost) all trees (I), *Random Structures & Algorithms* **14**, 153 (1999), ISSN 03043975.

[37] L. L. Cavalli-Sforza and A. W. Edwards, Phylogenetic analysis. Models and estimation procedures., *American journal of human genetics* **19**, 233 (1967), ISSN 0002-9297.

[38] W. M. Fitch and E. Margoliash, Construction of phylogenetic trees, *Science* **155**, 279 (1967).

[39] W. A. Beyer, M. L. Stein, T. F. Smith, and S. M. Ulam, A molecular sequence metric and evolutionary trees, *Mathematical Biosciences* **19**, 9 (1974).

[40] W. H. Day, Computational complexity of inferring phylogenies from dissimilarity matrices, *Bulletin of Mathematical Biology* **49**, 461 (1987).

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

[42] R. Desper, Theoretical Foundation of the Balanced Minimum Evolution Method of Phylo-

genetic Inference and Its Relationship to Weighted Least-Squares Tree Fitting, *Molecular Biology and Evolution* **21**, 587 (2003), ISSN 0737-4038.

[43] T. J. Wheeler, in *Algorithms in Bioinformatics* (Springer, 2009), 375–389, ISBN 3642042406.

[44] A. Rzhetsky and M. Nei, A simple method for estimating and testing minimum-evolution trees (1992).

[45] S. Mirarab and T. Warnow, ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes, *Bioinformatics* **31**, i44 (2015), ISSN 1367-4803.

[46] W. Fletcher and Z. Yang, INDELible: A flexible simulator of biological sequence evolution, *Molecular Biology and Evolution* **26**, 1879 (2009), ISSN 07374038.

[47] S. Guo, L.-S. Wang, and J. Kim, Large-scale simulation of RNA macroevolution by an energy-dependent fitness model (2009).

[48] S. Mirarab, N. Nguyen, S. Guo, L.-S. Wang, J. Kim, and T. Warnow, PASTA: Ultra-Large Multiple Sequence Alignment for Nucleotide and Amino-Acid Sequences, *Journal of Computational Biology* **22**, 377 (2015), ISSN 1066-5277.

[49] N.-P. N.-p. D. Nguyen, S. Mirarab, K. Kumar, and T. Warnow, Ultra-large alignments using phylogeny-aware profiles, *Genome Biology* **16**, 124 (2015), ISSN 1465-6906.

[50] C. Yin, G. Shen, D. Guo, S. Wang, X. Ma, H. Xiao, J. Liu, Z. Zhang, Y. Liu, Y. Zhang, et al., InsectBase: a resource for insect genomes and transcriptomes, *Nucleic Acids Research* **44**, D801 (2016), ISSN 0305-1048.

[51] W. Huang, L. Li, J. R. Myers, and G. T. Marth, ART: a next-generation sequencing read simulator, *Bioinformatics* **28**, 593 (2012), ISSN 1460-2059.

[52] Id 427774 - bioproject - ncbi, accessed 16 October 2018, `https://www.ncbi.nlm.nih.gov/bioproject/PRJNA427774`.

[53] D. E. Miller, C. Staber, J. Zeitlinger, and R. S. Hawley, Highly Contiguous Genome Assemblies of 15 Drosophila Species Generated Using Nanopore Sequencing, *G3: Genes, Genomes, Genetics* **8**, 3131 (2018), ISSN 2160-1836.

[54] A. Morgulis, G. Coulouris, Y. Raytselis, T. L. Madden, R. Agarwala, and A. A. Schäffer, Database indexing for production MegaBLAST searches., *Bioinformatics (Oxford, England)* **24**, 1757 (2008), ISSN 1367-4811.

[55] J. A. Chandler, J. M. Lang, S. Bhatnagar, J. A. Eisen, and A. Kopp, Bacterial communities of diverse Drosophila species: ecological context of a host-microbe model system., *PLoS genetics* **7**, e1002272 (2011), ISSN 1553-7404.

[56] N. A. Broderick and B. Lemaitre, Gut-associated microbes of Drosophila melanogaster., *Gut microbes* **3**, 307 (2012), ISSN 1949-0984.

[57] K. Petkau, D. Fast, A. Duggal, and E. Foley, Comparative evaluation of the genomes of three common Drosophila-associated bacteria., *Biology open* **5**, 1305 (2016), ISSN 2046-6390.

[58] B. Langmead and S. L. Salzberg, Fast gapped-read alignment with Bowtie 2, *Nature Methods* **9**, 357 (2012), ISSN 1548-7091.

[59] K. Tamura and M. Nei, Estimation of the Number of Nucleotide Substitutions in the Control Region of Mitochondrial-DNA in Humans and Chimpanzees, *Molecular biology and evolution* **10**, 512 (1993), ISSN 0737-4038.

[60] P. J. Waddell and M. Steel, General Time-Reversible Distances with Unequal Rates across Sites: Mixing Γ and Inverse Gaussian Distributions with Invariant Sites, *Molecular Phylogenetics and Evolution* **8**, 398 (1997), ISSN 10557903.

[61] M. Steel, A basic limitation on inferring phylogenies by pairwise sequence comparisons, *Journal of Theoretical Biology* **256**, 467 (2009), ISSN 00225193.

[62] A. Stamatakis, RAxML version 8: A tool for phylogenetic analysis and post-analysis of

large phylogenies, *Bioinformatics* **30**, 1312 (2014), ISSN 14602059.

[63] D. M. Hillis, D. D. Pollock, J. A. McGuire, and D. J. Zwickl, Is sparse taxon sampling a problem for phylogenetic inference?, *Systematic biology* **52**, 124 (2003), ISSN 1063-5157.

[64] D. J. Zwickl and D. M. Hillis, Increased taxon sampling greatly reduces phylogenetic error., *Systematic biology* **51**, 588 (2002), ISSN 1063-5157.

Supplementary appendix for APPLES

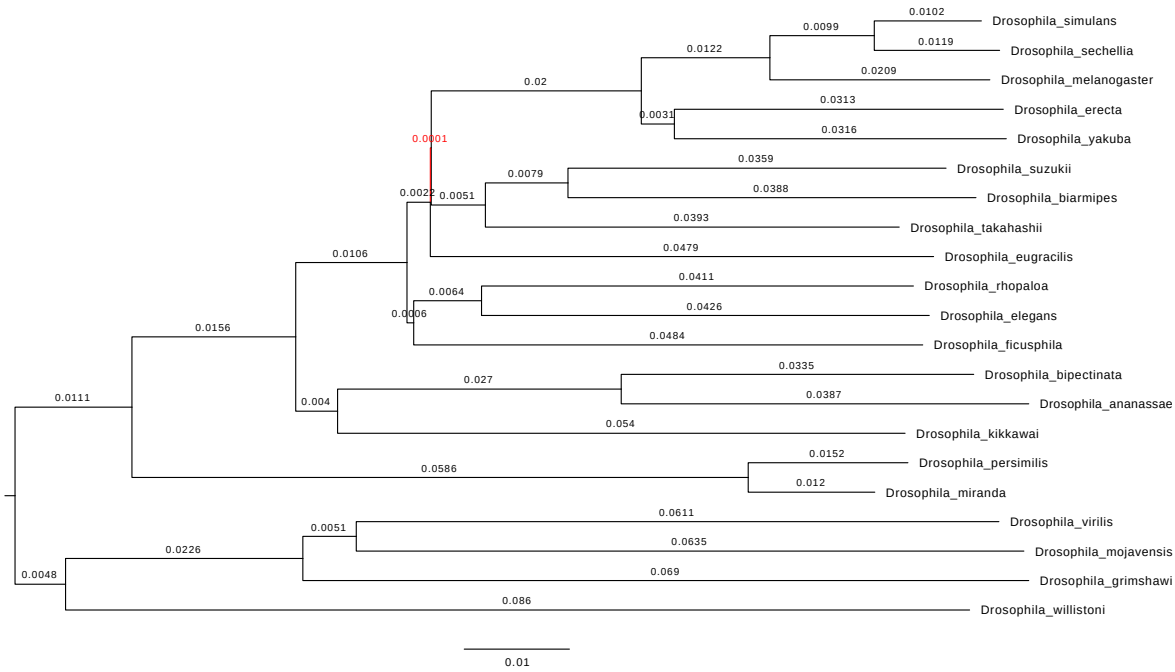


Figure S1: **The reference Drosophila tree.** The length of each branch is shown on the branch. The only species placed incorrectly using the APPLES-Hybrid approach is *Drosophila eugracilis*. Note that the correct placement of *Drosophila eugracilis* is on a super short branch of length 0.0001.

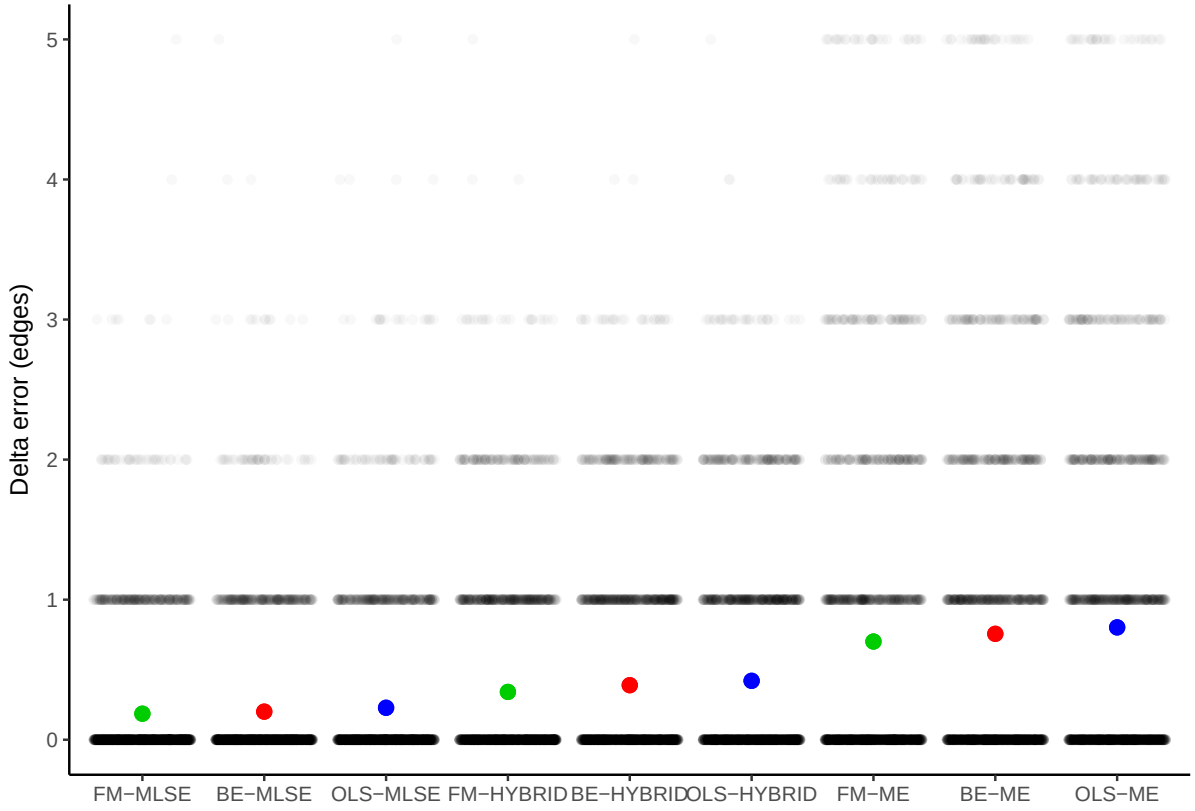


Figure S2: **Comparing APPLES versions**. For the RNASim dataset (without controlling for the diameter), we show the delta error (edges) of APPLES run with three options for weighting: FM (green), BE (red), and OLS (blue), and three options for selection strategy (MLSE, ME, and Hybrid). For each method, 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 method occasionally have error above 5 branches, but for better resolution, we cap the y-axis at 5.

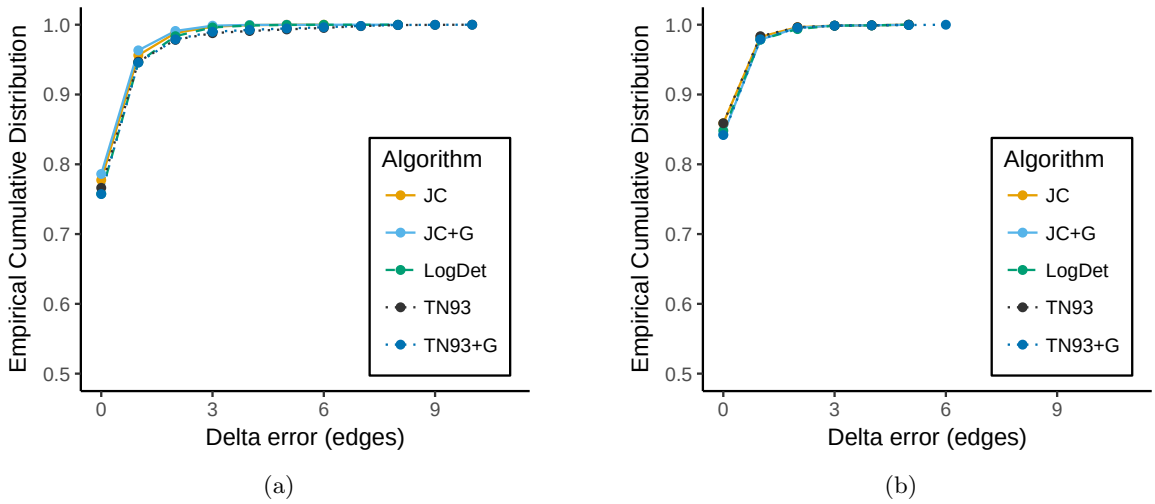


Figure S3: **Comparing various models of DNA evolution**. For the GTR (a) and RNASim-heterogeneous (b) datasets, we show the delta error (edges) of APPLES* run with five distance matrices calculated based on different models of DNA evolution. All model parameters are estimated per pair of sequences.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

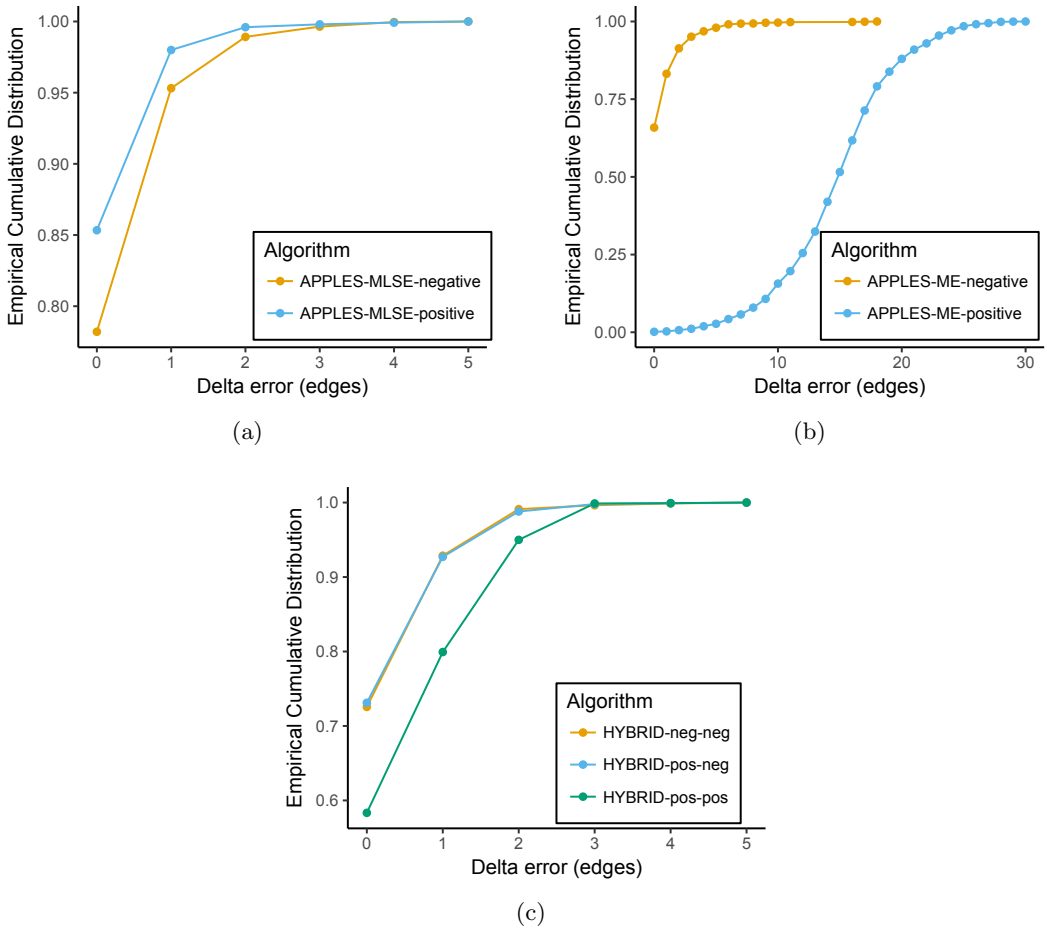


Figure S4: **The effect of imposing positivity constraint on accuracy.** We show the accuracy of (a) APPLES-MLSE and (b) APPLES-ME run both with (blue) and without (orange) enforcement of non-negative branch lengths on RNASim-heterogeneous dataset. Accuracy improves substantially for MLSE whereas it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

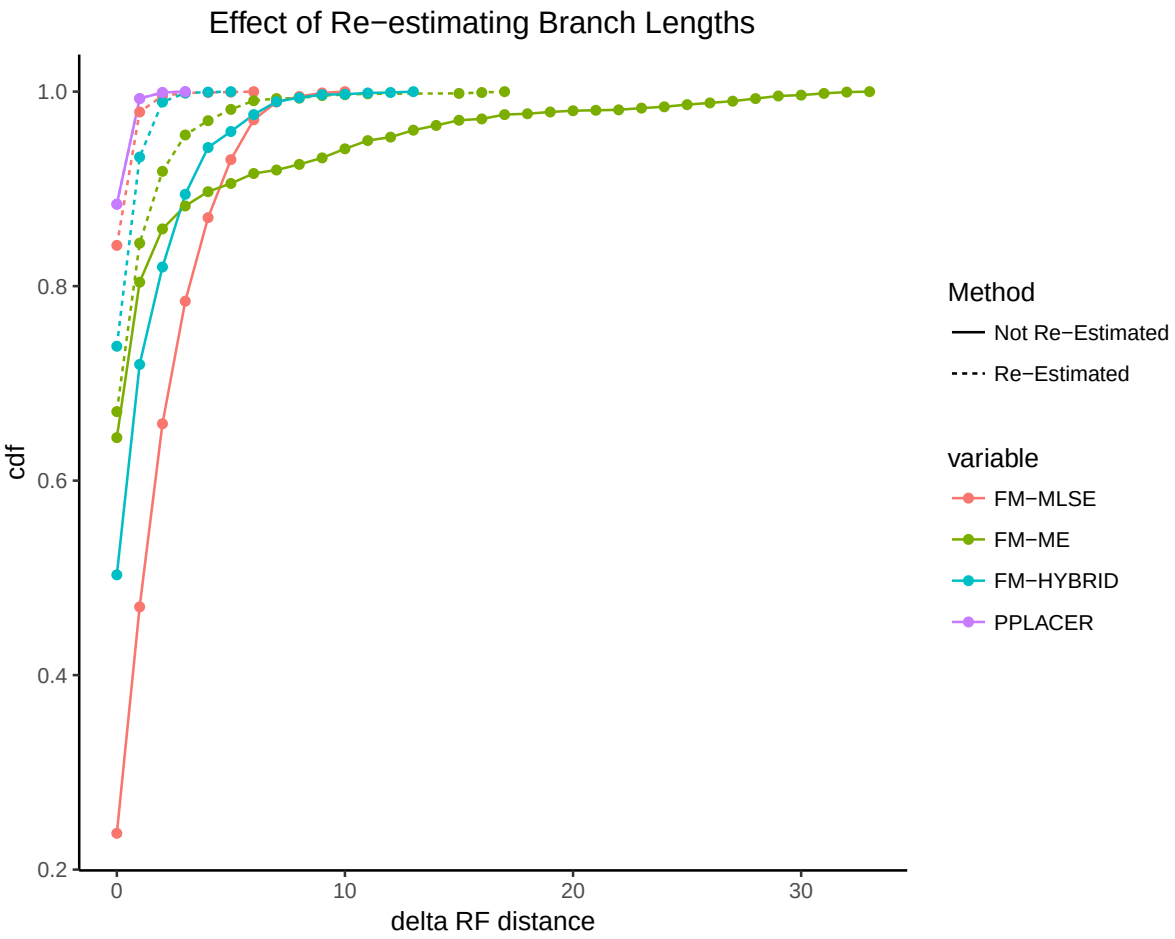


Figure S5: **The effect of reestimating branch lengths of the backbone tree on accuracy.** We show accuracy of pplacer and APPLES-FM with its three optimization criteria. APPLES is run both with (dotted) and without (solid) re-estimating branch lengths in the backbone tree using the same model (here, 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 labelled "Not re-estimated" uses branch lengths produced using RAxML under the GTR+Γ model.

Table S1: 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 S2: 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

A Proofs and derivations

Derivation of Equation 6.

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. 3:

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

We take the derivative of Eq. 3 with respect to x_1 and set it equal to zero:

$$\begin{aligned}
\frac{\partial Q(P)}{\partial x_1} &= -2\sum_{i\in g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{ui}(P) - x_1 + x_2 - l(u)\big) - 2\sum_{i\notin g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{p(u)i}(P) - x_1 - x_2\big) = 0 \\
&\implies \Big(\sum_{i\in g(u)} \delta_{qi}^{-k} + \sum_{i\notin g(u)} \delta_{qi}^{-k}\Big)x_1 + \Big(-\sum_{i\in g(u)} \delta_{qi}^{-k} + \sum_{i\notin g(u)} \delta_{qi}^{-k}\Big)x_2 \\
&\quad - \sum_{i\in g(u)} \delta_{qi}^{1-k} - \sum_{i\notin g(u)} \delta_{qi}^{1-k} + \sum_{i\in g(u)} \delta_{qi}^{-k} d_{ui}(P) + \sum_{i\notin g(u)} \delta_{qi}^{-k} d_{p(u)i}(P) + l(u)\sum_{i\in g(u)} \delta_{qi}^{-k} = 0 \\
&\implies \big(S'(-k,u) + R'(-k,u)\big)x_1 + \big(-S'(-k,u) + R'(-k,u)\big)x_2 = \\
&\quad S'(1-k,u) + R'(1-k,u) - S(-k,1,u) - R(-k,1,u) - l(u)S'(-k,u)
\end{aligned}$$

Similarly,

$$\begin{aligned}
\frac{\partial Q(P)}{\partial x_2} &= 2\sum_{i\in g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{ui}(P) - x_1 + x_2 - l(u)\big) - 2\sum_{i\notin g(u)} \delta_{qi}^{-k}\big(\delta_{qi} - d_{p(u)i}(P) - x_1 - x_2\big) = 0 \\
&\implies \Big(-\sum_{i\in g(u)} \delta_{qi}^{-k} + \sum_{i\notin g(u)} \delta_{qi}^{-k}\Big)x_1 + \Big(\sum_{i\in g(u)} \delta_{qi}^{-k} + \sum_{i\notin g(u)} \delta_{qi}^{-k}\Big)x_2 \\
&\quad + \sum_{i\in g(u)} \delta_{qi}^{1-k} - \sum_{i\notin g(u)} \delta_{qi}^{1-k} - \sum_{i\in g(u)} \delta_{qi}^{-k} d_{ui}(P) + \sum_{i\notin g(u)} \delta_{qi}^{-k} d_{p(u)i}(P) - l(u)\sum_{i\in g(u)} \delta_{qi}^{-k} = 0 \\
&\implies \big(-S'(-k,u) + R'(-k,u)\big)x_1 + \big(+S'(-k,u) + R'(-k,u)\big)x_2 = \\
&\quad -S'(1-k,u) + R'(1-k,u) + S(-k,1,u) - R(-k,1,u) + l(u)S'(-k,u)
\end{aligned}$$

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 dynamic programming (Equation 4 and Equation 5).

Recall the dynamic programming recursions of Equations 4 and 5:

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

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

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.

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

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 $p(p(u))$.

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

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 $S'(a,u) = \delta^a_{qu}$. 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 $d^b_{p(1')1} = 0$, we trivially have $R(a,b,1') = 0$. For $b = 0$, $R'(a,1') = \delta^a_{q1}$.

Time and memory complexity of this dynamic programming is $\Theta(bn)$, which translates to $\Theta(n)$ in least squares setting, where $b \leq 2$.

Rewriting of Equation 3.

By simple rearrangement of the terms, we can rewrite Equation 3 in the following form. Note that computing $Q(P(u, x_1, x_2))$ requires only $S(,,u)$ and $R(,,u)$ values and $l(u)$.

$$\begin{aligned} Q(P(u,x_1,x_2)) = {} & R'(2-k,u) + S'(2-k,u) + R(-k,2,u) + S(-k,2,u) \\ & + 2\big(x_1 + x_2\big)\big(R(-k,1,u) - R'(1-k,u)\big) \\ & + 2\big(x_1 + l(u) - x_2\big)\big(S(-k,1,u) - S'(1-k,u)\big) \\ & + \big(x_1 + x_2\big)^2 R(-k,1,u) + 2\big(x_1 + l(u) - x_2\big)^2 S(-k,1,u) \\ & - 2R(1-k,1,u) - 2S(1-k,1,u) \end{aligned} \tag{S1}$$

B Commands

B.1 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
```

B.2 Backbone tree estimation

When multiple sequence alignment is available, we used the following RAxML command to compute backbone tree for all datasets except RNAsim varied size 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 varied size 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*.

B.3 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 varied size 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 varied size 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
```

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
```

B.4 Performing placement

We performed phylogenetic placement of a query using pplacer with the following commands:

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