UNIVERSITY OF CALIFORNIA SAN DIEGO

Alignment-free genomic distance estimation: from conventional methods to machine learning

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Bioinformatics and Systems Biology

by

Eleonora Rachtman

Committee in charge:

Professor Siavash Mirarab, Chair
Professor Vineet Bafna, Co-Chair
Professor Melissa Gymrek
Professor Pavel Pevzner
Professor David Smith

2023

The Dissertation of Eleonora Rachtman is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

DEDICATION

Dedicated to my family, the memory of my grandparents and our close family friend who was a doctor and inspired me to become one too.

EPIGRAPH

The secret of getting ahead is getting started.
The secret of getting started is breaking your complex overwhelming tasks into small manageable tasks, and starting on the first one.

Mark Twain

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank numerous of people who helped me through my PhD. I want to thank my advisor Dr. Siavash Mirarab for giving me a chance to join his lab five years ago. I have truly enjoyed my time there. I am tremendously grateful for all the opportunities that Dr. Mirarab has provided me with as well as all the multiple collaborations I had a chance to do while working as a member of his team. Dr. Mirarab's knowledge combined with his leadership style were a guiding light for me and I am glad to have met such an amazing mentor on my career path.

Next I want to thank Dr. Vineet Bafna for his guidance and insightful comments on a number of manuscripts that we have written together. His expertise was proven to be invaluable for the successful completion of these projects.

I also would like to thank other members of my committee, Dr. Melissa Gymrek, Dr. Pavel Pevzner and Dr. David Smith for their tremendous help and support at different stages of my PhD journey.

I would like to acknowledge my collaborators, colleagues and friends at bioinformatics program and beyond. I have met a number of fantastic researchers who I have had the privilege to learn from and made a number of genuine life long friends.

Finally, I would like to thank my family, especially my husband and my parents for their constant encouragement and understanding. My work would have taken twice as long if not for their support. Very special thank you goes to my mom who taught me that "the road will be mastered by the walking" and showed me with her own example that everything I want lies on the other side of consistency.

Chapter 1, in full, is a reprint of the material as it appears in The impact of contaminants on the accuracy of genome skimming and the effectiveness of exclusion read filters. Rachtman, E.; Balaban, M.; Bafna, V., & Mirarab, S., Molecular Ecology Resources, 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 2, in full, is a reprint of the material as it appears in CONSULT: accurate

contamination removal using locality-sensitive hashing. Rachtman, E.; Bafna, V., & Mirarab, S., NAR Genomics and Bioinformatics, 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in Quantifying the uncertainty of assembly-free genome-wide distance estimates and phylogenetic relationships using subsampling. Rachtman, E.; Sarmashghi, S.; Bafna, V., & Mirarab, S., Cell Systems, 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, is currently being prepared for submission for publication of the material. Rachtman, E.; Jiang Y.; Mirarab, S. Alignment-free phylogenetic placement using machine learning. The dissertation author was the primary investigator and author of this material.

VITA

2011 Master of Science, San Diego State University

2023 Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Rachtman, E., Jiang, Y., & Mirarab, S. (2023). Machine learning enables alignment-free distance calculation and phylogenetic placement using *k*-mer frequencies (in preparation).

Rachtman, E., Sarmashghi, S., Bafna, V., & Mirarab, S. (2022). Quantifying the uncertainty of assembly-free genome-wide distance estimates and phylogenetic relationships using subsampling. Cell Systems, 13(10), 817-829.e3.

Rachtman, E., Bafna, V., & Mirarab, S. (2021). CONSULT: accurate contamination removal using locality-sensitive hashing. NAR Genomics and Bioinformatics, 3(3).

Rachtman, E., Balaban, M., Bafna, V., & Mirarab, S. (2020). The impact of contaminants on the accuracy of genome skimming and the effectiveness of exclusion read filters. Molecular Ecology Resources, 20(3), 649–661.

Şapcı, A. O. B., Rachtman, E., & Mirarab, S. (2023). CONSULT-II: Taxonomic Identification Using Locality Sensitive Hashing BT - Comparative Genomics (K. Jahn & T. Vinař, eds.). Cham: Springer Nature Switzerland.

Sarmashghi, S., Balaban, M., Rachtman, E., Touri, B., Mirarab, S., & Bafna, V. (2021). Estimating repeat spectra and genome length from low-coverage genome skims with RESPECT. PLOS Computational Biology, 17(11), e1009449.

ABSTRACT OF THE DISSERTATION

Alignment-free genomic distance estimation: from conventional methods to machine learning

by

Eleonora Rachtman

Doctor of Philosophy in Bioinformatics and Systems Biology

University of California San Diego, 2023

Professor Siavash Mirarab, Chair
Professor Vineet Bafna, Co-Chair

Inferring evolutionary relationships based on comparative analysis of genomic data remains a fundamental question in biology. Conventionally, these analyses involve cumbersome and computationally expensive steps such as assembly, gene annotation, and multiple sequence alignment. Alternatively, phylogenomic analyses can be conducted using alignment-free approaches, often using *k*-mers to compute the evolutionary distance. However, despite being fast and accurate, *k*-mer-based methods have their own challenges. Crucially, this approach can be used with low-coverage sequencing of samples (i.e., genome skims), which can reduce costs.

A major challenge in analyzing genome skims is the presence of extraneous sequences

in genomic data. We show that contaminants reoccurring in multiple samples can impact *k*-mer-based distance estimation and thus phylogenetic inference. To combat this problem, we introduce CONSULT, an algorithm for efficiently removing extraneous reads from sequencing samples. We demonstrate that CONSULT has higher accuracy for contamination detection than leading methods such as Kraken-II and improves distance calculation for genome skims. Additionally, we show that CONSULT can be used to distinguish organelle reads from nuclear reads, improving the quality of skims-based mitochondrial assemblies.

Another challenge in using *k*-mer-based phylogenetic methods is the absence of a solid statistical procedure to estimate uncertainty, limiting the use of these methods in practice. To address this problem, we developed an algorithm for quantifying the uncertainty of alignment-free phylogenies using subsampling and relying on sound statistical principles. We demonstrate that our method is reasonably fast and can correctly identify uncertain branches on phylogenies constructed using real and simulated datasets.

As a final challenge, we tackle the problem of updating phylogenies with new genomes while avoiding alignment or even assembly. As sequencing data becomes readily available, *de novo* tree reconstruction becomes infeasible. However, placement into an existing tree provides an efficient alternative. Attempts in alignment-free phylogenetic placement have both scalability and accuracy limitations. We approach this problem by representing each genome as a vector of *k*-mer frequencies and leveraging machine learning to estimate distances between such vectors. We demonstrate that our method, kf2d, outperforms existing *k*-mer-based approaches in distance calculation and allows placing new samples on phylogenies constructed from heterogeneous data types.

Introduction

Molecular sequence comparison is one of the most basic and fundamental problems in computational biology, and has been widely used to study the evolution of whole genome sequences and gene regulatory regions, gene function prediction, sequence assembly, and finding the relationships among microbial communities [193]. The most widely used methods for molecular sequence comparison are alignment-based including the Smith-Waterman algorithm [219], BLAST [4], BLAT[103], etc. Although alignment-based approaches are most accurate and powerful for sequence comparison when they are feasible, their applications are limited in some situations.

First, alignment-producing programs assume that homologous sequences comprise a series of linearly arranged and more or less conserved sequence stretches. However, this assumption is often violated for a wide range of data types [259]. Thus due to duplications, translocations, large insertions/deletions, and horizontal gene transfers each genome becomes a mosaic of unique lineage-specific segments (i.e., regions shared with a subset of other genomes) [193]. This situation makes it difficult to use alignment-based methods to investigate the relationship among these whole genome sequences. Second, alignment-based approaches are generally memory-consuming and time-consuming and thus are of limited use with multigenome-scale sequence data. Even for more mathematically optimal dynamic programming solutions time complexity is in the order of the product of the lengths of the input sequences [62]. Therefore, despite the wealth of tools and more than 15 years of research [7, 48, 49, 104, 179, 210] the problem of long sequence alignment is not fully resolved [61]. Finally, a somewhat arbitrary selection of various alignment parameters (e.g., substitution matrices, gap penalties, and threshold

values for statistical parameters) which varies between applications can greatly affect the quality of downstream results [249]. For all these scenarios, alignment-free methods for genome and metagenome comparison provide promising alternative approaches.

Alignment-free sequence comparison is defined as any method of quantifying sequence similarity or dissimilarity that does not use or produce alignment (assignment of residue–residue correspondence) at any step of algorithm application [259]. Alignment-free approaches are computationally less expensive (as they generally have linear complexity depending only on the length of the query sequence [31]) and therefore suitable for efficient whole genome comparisons [25, 101, 118, 217]. Alignment-free methods might also be resistant to shuffling and recombination events and are applicable when low sequence conservation cannot be handled reliably by alignment [241].

Among the many types of alignment-free sequence comparison approaches, word-count-based approaches are most popular due to their easy adaption to NGS data [221]. These methods first count the number of occurrences of word patterns (e.g., k-mers; subsequences of length k) along a sequence or in a NGS sample using different algorithms such as Jellyfish [139], DSK [195], and KMC 2 [54]. Secondly, similarity/dissimilarity measures such as Hamming distance or Jaccard index are defined between any pair of sequences based on the k-mer frequencies. Constructed distance matrices after being phylogenetically corrected by various evolutionary models (typically Jukes and Cantor 1969) are utilized in a variety of downstream analyses. All things considered, given their ease of use and abundance of choices (more than 100 techniques to consider [241]) alignment-free sequence approaches have been successfully applied to a variety of different problems including the study of evolutionary relationships of whole genome sequences and gene regulatory regions, comparison of metagenomes and metatranscriptomes, binning of contigs, detection of horizontal gene transfer, and virus-host infectious associations based on next generation sequencing (NGS) data [193].

Our interest in alignment-free sequence analysis is also due to their extended ability to efficiently handle low coverage (1-2$\times$) sequencing data, that are collectively referred to as

genome skims [45, 225, 244]. The term 'genome skimming' was first coined by Straub et al. [225] as a way of 'navigating the tip of the genomic iceberg'; that is, shallow sequencing of DNA that results in comparatively deep sequencing of the high-copy fraction of the genome (plastome, mitogenome, and repetitive elements) [59]. This is understood since traditionally genome-skimming data were used primarily for assembling the over-represented organelle genome using one of several approaches that have been developed [1, 3, 8, 41, 57, 77, 97]. However, when genome skimming approaches discard the majority of nuclear reads (as much as 99% of the sequence data) it is not only wasteful, but can also limit the power of discrimination at, or below, the species level [30]. We believe that the application of alignment-free methods to the analysis of the full shotgun skimming samples (we refer to them as bags of unassembled reads) will allow us to extract biological knowledge from these data to the full extent. This vision has already been pursued by several methods that enable computing distances among skims [64, 112, 204, 231, 258] and using those distances for phylogenetic placement [14, 18].

In our current work, we aim to build upon existing methodologies to develop scalable approaches for the analysis of big genomic low-coverage datasets in order to answer questions in areas of biodiversity, including sample identification, population genetics, and derivation of more accurate phylogenomic relationships.

Chapter 1

On the impact of contaminants on the accuracy of genome skimming and the effectiveness of exclusion read filters

The ability to detect the identity of a sample obtained from its environment is a cornerstone of molecular ecological research. Thanks to the falling price of shotgun sequencing, genome skimming, the acquisition of short reads spread across the genome at low coverage, is emerging as an alternative to traditional barcoding. By obtaining far more data across the whole genome, skimming has the promise to increase the precision of sample identification beyond traditional barcoding while keeping the costs manageable. While methods for assembly-free sample identification based on genome skims are now available, little is known about how these methods react to the presence of DNA from organisms other than the target species. In this paper, we show that the accuracy of distances computed between a pair of genome skims based on k-mer similarity can degrade dramatically if the skims include contaminant reads; i.e., any reads originating from other organisms. We establish a theoretical model of the impact of contamination. We then suggest and evaluate a solution to the contamination problem: Query reads in a genome skim against an extensive database of possible contaminants (e.g., all microbial organisms) and filter out any read that matches. We evaluate the effectiveness of this strategy when implemented using Kraken-II, in detailed analyses. Our results show substantial improvements in accuracy as a result of filtering but also point to limitations, including a need for relatively close matches in

the contaminant database.

1.1 Introduction

Anthropogenic pressure and other natural causes have resulted in severe disruption of the global ecosystems in recent years, including loss of biodiversity and invasion of non-native flora and fauna. Conservationists, struggling with an unprecedented rate of extinction, are using innovative approaches to measure the changing biodiversity of the planet. Genome sequencing provides an attractive alternative to physical sampling and cataloging, as falling costs have made it possible to shotgun sequence a reference specimen sample for at most $10 per Gb (with another $60 for sample prep). However, the analysis typically requires assembling and finishing a reference genome, which can still be prohibitively costly. It could be many decades before the biodiversity of our planet is represented in the form of finished genomes (and cataloged genomic variants) and before biodiversity measurements for each population can be acquired on an ongoing basis.

The standard molecular technique for measuring biodiversity at the organismal level is barcoding [80, 205, 229], which involves DNA sequencing of taxonomically informative and group-specific marker genes (e.g., mtDNA COI [80, 213], 12S/16S [240], plastid genes [87], and ITS [208]). Existing reference databases and computational methods enable measurements of biodiversity using barcodes [192, 224, 229]. However, since barcodes are short regions, their phylogenetic signal is limited [81]. For example, 896 of the 4,174 species of wasps could not be distinguished from other species using COI barcodes [186].

As an alternative, a *genome skim* is a low-coverage acquisition of short reads from a sample, typically around 1-5 Gbp [45, 59], providing 0.1-10× coverage, and usually insufficient for assembling nuclear contigs. Falling sequencing costs have made genome-skimming cost-effective while providing richer data than barcoding, but the data is harder to analyze.

Skimming applications often rely on assembling organelle genomes [e.g., 136, 244] from their over-represented reads. This approach throws away the vast majority of the reads, potentially limiting the resolution. Moreover, organelle genomes may not represent the rest of the genome and are not always easy to assemble. Ideally, we should use both reads from both nuclear and organelle genomes. However, methods that seek to mine all information from genome skims must be *assembly-free* and *map-free* and face additional challenges.

Recently, Sarmashghi et al. [204] developed a method, Skmer, that accurately computes the genomic distance between genome skims by simply analyzing k-mers (short substrings of length k) in both genome skims. Skmer is based on three principles. First, as observed by [168], the *Jaccard index*, J (the size of the intersection of two sets divided by the size of their union) between *k-mer* sets of the two genomes can be computed efficiently. Second, J can be used to estimate the genomic distance (D) between two species by carefully accounting for dependence on coverage, sequencing error, and genome length. Third, both coverage and error rate can be computed from genome skim data by modeling histograms of k-mer frequencies. By combining these three principles, Skmer provides excellent accuracy in estimating distances between genome skims. These distances can then be used for taxonomic identification and phylogenetic placement [18] of query genome skims with respect to a set of reference genome skims. Previous results have shown high accuracy and increased resolution compared to barcodes when using genome skims for taxonomic identification [18, 204].

The Skmer methodology, however, completely ignores the very real possibility that *a genome skim includes extraneous reads originating from other species*, often bacteria, viruses, or fungi, that cohabit inside the biological organism. With a slight abuse of terminology, we refer to all reads originating from species other than the target species being identified as *contamination*. Contamination of genome skims is unavoidable in many cases as microorganisms that co-exist with a species are often hard or impossible to separate from the original sample. To make matters worse, lab protocols used for genome skimming also can add human and other forms of contamination. The standard organelle-based analyses of skims manages to deal with sequencing

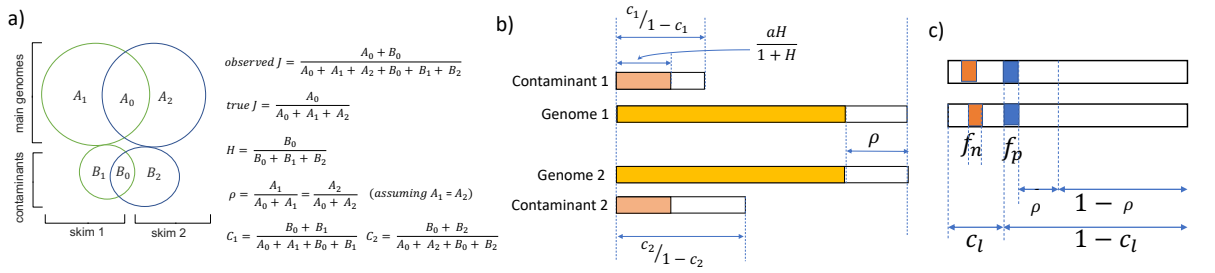


Figure 1.1. Model. (a) Definition of terms. Contaminating k-mers change the estimated Jaccard in a complex manner. (b) Assuming equal lengths for the two genomes, all quantities are measured as a fraction of the number of k-mers in each genome: $1-\rho$ of the k-mers are shared between the base genomes; additionally, out of a total of $a = \frac{c_1}{1-c_1} + \frac{c_2}{1-c_2}$ contaminants, $\frac{aH}{1+H}$ are shared, making the total number of shared k-mers equal to $(1-\rho) + \frac{aH}{1+H}$ and the total number of distinct k-mers in the union equal $(1+\rho) + \frac{a}{1+H}$. See A.1 for details. (c) Impact of false positives and negatives in contaminant removal in the disjoint contaminant scenario. We keep $(1-c_l)(1-f_p) + c_l f_n$ of the k-mers in each set, with the intersection proportion being $(1-c_l)(1-f_p)(1-\rho)$.

errors and contamination by focusing on and assembling a small portion of the reads. These contaminates have the potential to mislead the Jaccard-based calculation of distance using methods such as Skmer. Thus, to take advantage of *all* reads across the genome, contaminants will have to be dealt with.

In this paper, we study the impact of contamination on Skmer estimates of the genomic distance. We then study whether the negative impact of contamination can be reduced using "exclusion filters": search every read of a skim against a library of all known contaminants (e.g., bacterial, fungal, and viral genomes), filter out reads that map to the library, and use the remaining reads to compute the distance. The efficacy of this exclusion filtering approach is unclear and can depend on several factors, which we thoroughly explore here. We study these effects both based on a theoretical model and in careful simulation and empirical analyses using a leading read matching tool called Kraken-II [250].

1.2 Material and methods

1.2.1 Theoretical Exposition

Consider two genomes of equal length and separated by genomic distance D, defined as the portion of positions that do not match in a perfect alignment of the two genomes. Let ρ denote the proportion of k-mers in one species that are absent in the other. The Jaccard-index of k-mers is given by (Fig. 1.1a):

$$J = \frac{\text{Intersection of } k\text{-mer sets}}{\text{Union of } k\text{-mer sets}} = \frac{1-\rho}{2-(1-\rho)} = \frac{1-\rho}{1+\rho}\,.$$

We model the evolutionary process producing the two genomes as follows. Starting from one genome, mutate each position independently with probability D to get the second genome. With this model, ρ becomes a random variable. Ignoring the dependency between adjacent k-mers and assuming k-mer independence, we get $\mathbb{E}(\rho) = 1-(1-D)^k$. As [64] show, we can estimate:

$$\hat{D} = 1 - \left(\frac{2J}{1+J}\right)^{\frac{1}{k}} \tag{1.1}$$

Skmer further models coverage and sequencing error and uses

$$\hat{D} = 1 - \left(\frac{2(\zeta_1 L_1 + \zeta_2 L_2)J}{\eta_1 \eta_2 (L_1+L_2)(1+J)}\right)^{1/k} \tag{1.2}$$

where η_i, ζ_i, and L_i are parameters related to coverage, error, and genome length, all automatically estimated by Skmer from k-mer profiles. As the simpler equation is easier to manipulate, we use (1.1) in our theoretical exposition. However, our empirical results will use the Skmer software, which uses (1.2). Throughout the paper, we use the relative error to quantify any error in estimating D:

$$\text{relative error of } \hat{D} = \frac{\hat{D}-D}{D} \tag{1.3}$$

where D is the true genomic distance and $\hat{D}$ is the estimated genomic distance.

Impact of contamination

Contamination can clearly alter Jaccard and hence the estimated genomic distance (Fig. 1.1a). The impact of contamination depends on factors such as the amount and exact composition of contaminants. For exposition purposes, let us assume that an identical proportion of k-mers (denoted by c_l) of both skims are contaminated, and contaminant k-mers are entirely disjoint between the two genome skims. Then, J becomes a function of c_l:

$$J = \frac{(1-c_l)(1-\rho)}{2-(1-c_l)(1-\rho)} \approx \frac{(1-c_l)(1-D)^k}{2-(1-c_l)(1-D)^k}$$

where the approximation is achieved by replacing ρ with its expectation.

Under these assumptions, Jaccard reduces under contamination and extent of reduction depends on c_l and to a lesser degree on D (Fig. B.1a). If the impact of contamination on Jaccard is ignored, distance will be overestimated at a level that strongly depends on the true distance (Fig. B.1b). When D is sufficiently high, substantial levels of contamination result in relatively low errors. However, with smaller distances, contamination can drastically increase the relative error. At $D = 0.1\%$ (e.g., within species differentiation), 3% contamination is enough to cause 100% relative error. Thus, under the simple disjoint contamination model, contamination has a large negative impacts *only* when the distance between base genomes is small.

Disjoint contamination assumption, however, is quite strong. When both samples are contaminated with the same species (say, human), the assumption of disjoint contaminant k-mers can mislead. To generalize, consider two genomes with an equal number of k-mers L. Let c_1 denote the fraction of the k-mers from sample 1 that are contaminated. Then, the ratio of contaminated k-mers to true k-mers in genome 1 is given by $\frac{c_1}{1-c_1}$ (A.1). Define c_2 in an analogous fashion for genome 2, and let $a = \frac{c_1}{1-c_1} + \frac{c_2}{1-c_2}$ (Fig. 1.1a). Removing the disjoint contamination assumption, define the Jaccard index between the k-mers of the contaminants of

the two samples as H. Then, as shown in A.1 and Figure 1.1b,

$$J = \frac{(1-\rho)(1+H)+aH}{(1+\rho)(1+H)+a} \,. \tag{1.4}$$

Plotting this formula shows that depending on H, the estimated Jaccard may over-estimate or under-estimate the true Jaccard, and converting the Jaccard to distance without any consideration of contaminants can lead to over or under-estimate the true distance (Fig. 1.2a). Once again, error depends on the true distance D, where most dramatic error happens when distance is low and H is also low. Introduction of H shows that contamination can result in both over and under-estimation of error. In particular, for larger values of D, if contaminants are similar between the two samples, relatively low levels of contamination can lead to severe under-estimation of distance. For example, with $D = 0.18\%$, if the samples are contaminated at 5% with somewhat similar species with $H = 0.5$, the estimated distance will be under-estimated by 43%.

Impact of exclusion filtering

One approach to deal with contamination is using exclusion filters: search *all* reads in a genome skim against a (potentially incomplete) library of *known* contaminants and filter out reads that match the library. This approach will impose a trade-off between two types of possible errors. A false positive (**FP**) occurs when we incorrectly filter out a read that belongs to the target genome. A false negative (**FN**) occurs when we fail to filter out a read that belongs to contaminants, perhaps due to an insufficient similarity between the read and genomes included in the exclusion library. The exact choice of the method and parameters used for mapping reads to reference contaminant libraries, in addition to the composition of the reference library, create a trade-off between FP and FN error. The trade-off poses an important question: which type of error, FP or FN, is more damaging? Falling back on the disjoint contaminant k-mer assumption, we can approximate impact of FP and FN on J given one more assumption: A k-mer shared between the two genome skims is either kept or removed from both skims.

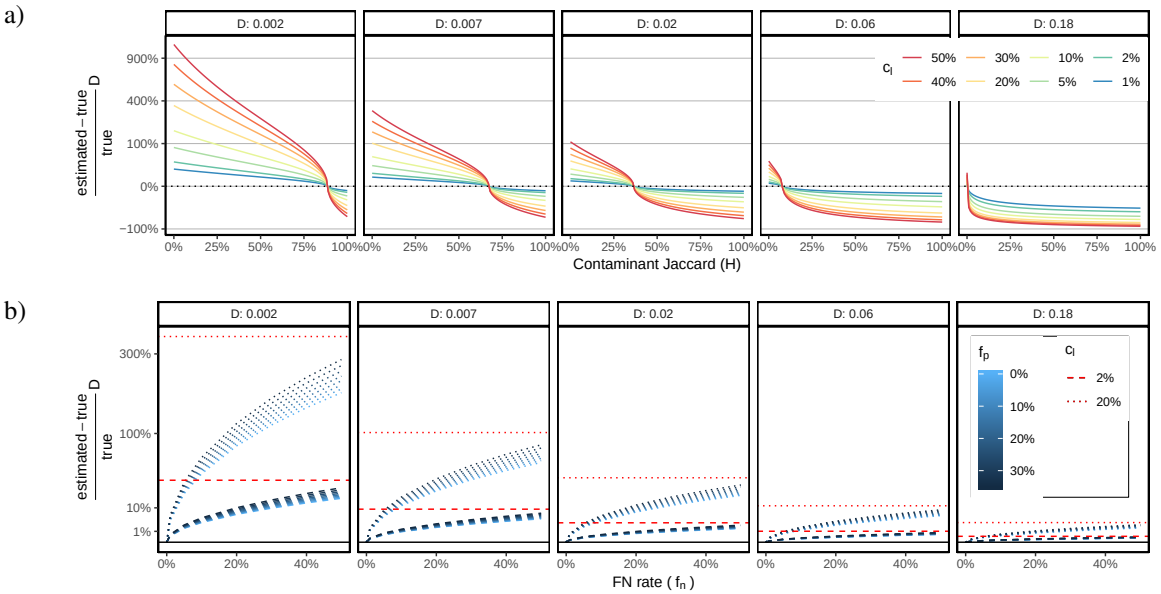


Figure 1.2. Theoretical modeling. (a) Impact of contamination on the genomic distance estimated from Jaccard according to theoretical expectation assuming contaminant k-mers of the two skims have a Jaccard of H (1.4). For several D and varying H, relative error is shown for eight contamination levels $c_l = c_1 = c_2$. (b) Error in Skmer distance (computed using (1.1), with Jaccard approximated using (1.5)) in the presence of filtering and with the disjoint contaminant k-mer assumption for various levels of FP portion (f_p), FN (f_n) rate, and c_l. Red lines show the error in the absence of filtering. y-axis is in square root scale and $k = 31$.

Let f_p be the portion of all k-mers that we remove by mistake (FP) and f_n be the portion of the contaminant k-mers that we fail to remove (FN). The proportion of k-mers shared between genome skims after filtering is $(1-c_l)(1-f_p)(1-\rho)$ (Fig. 1.1c). Additionally, $(1-c_l)(1-f_p)+c_l f_n$ of the k-mers in each set are retained after filtering for the total number of unique k-mers to be $2((1-c_l)(1-f_p)+c_l f_n)-(1-c_l)(1-\rho)(1-f_p)$. Thus,

$$J = \frac{(1-c_l)(1-\rho)(1-f_p)}{(1-c_l)(1+\rho)(1-f_p)+2f_n c} \tag{1.5}$$

By plotting this equation as we vary the four parameters (D, c_l, f_p, and f_n), we observe that filtering can successfully reduce the impact of contamination under many but not all conditions (Figs. 1.2b and B.2). Filtering can be very effective in making Jaccard index close to what we would obtain without contamination, and overall, Jaccard is more sensitive to FN

errors than it is to FP errors. Impact of filtering on genomic distance depends on the level of contamination, false negatives, and most of all, the true genomic distance. Reassuringly, in this model, the estimated accuracy of distance after filtering is reasonably high in most cases (Figs. B.2 and B.3). Nevertheless, in the most challenging cases, filtering cannot sufficiently reduce the error. With $D = 0.2\%$, unless f_n is low or c_l is moderate, error can be very high. Overall, f_p errors are less damaging than f_n. Practically, it seems that with $f_n \leq 0.2$, highly accurate estimates of distance are possible unless contamination levels are very high *and* the genomic distance is very low.

1.2.2 Empirical analyses using Kraken-II

Our empirical experiments validate the effectiveness of exclusion filtering, focusing on a leading k-mer-based read mapping tool called Kraken-II, originally designed for metagenomics and adopted here for contamination filtering. We start by describing Kraken-II and then detail the setup for the four experiments performed.

Kraken

Kraken-II works by mapping all k-mers of a read to k-mers in a reference library and calls a read a match if the number of k-mers matching is *strictly* larger than a user-provided threshold called the confidence level (α). Kraken-II uses LCA-mapping to find lowest taxonomic level at which the read can be confidently matched. It also uses wild-carding s random positions of each k-mer [36] to increase the sensitivity of matches. We will explore both k and α settings but fix other parameters. We set minimizer length $l = k$ or use the maximum allowed l value (31) for reference databases built with k ¿ 31. We set s, the number of wild-card positions, to its maximum allowable value, $l/4$. We design our reference Kraken-II libraries to include a set of potential contaminants and as query, we use the bag of all reads in a genome skim (details described below). We will use microbial genomes to simulate contamination, and thus, all reference libraries we use are microbial. In contrast, our base genomes are Eukaryotic (plants or

insects).

Experiments

We present four experiments that explore the impact of D (equivalently, ρ), c_l, f_p, and f_n. In addition, we test the running time of Kraken-II. Below, we describe the setup used in each experiment.

Exploring FN and FP of Kraken-II. We start by examining the sensitivity of Kraken-II to two parameters: k and α. We also consider completeness of the reference library, which is expected to have a direct effect on f_p and f_n rates. A lack of sufficiently close genomes to the contaminant can prevent Kraken-II from finding a match, and presence of genomes similar to non-contaminant genomes can cause FP matches. Thus, we define a third variable, M, as the genomic distance between a query and its closest match in the library. We control M by carefully selecting species included in the reference library and those used as query.

To control M, we use an available reference phylogeny of 10,575 bacterial and archaeal genomes [257]. Five genomes from this set had IDs that did not exist in NCBI anymore. We assigned remaining genomes to the reference library (10,460 genomes), the query set (100), or both (10). Based on the available phylogeny, we select 10 sets of 10 query genomes such that all genomes in a set have similar patristic (tree) distance to their closest leaf in the tree, not counting the query genomes. These sets had mean tree distance of $\{0.01, 0.02, 0.04, 0.06, 0.09, 0.10, 0.18, 0.23, 0.57, 1.20\}$ and at most 25% divergence from the mean. We also randomly chose 10 genomes to be added to both reference and query sets. Then, for each of the 110 query genomes, we used Mash to compute M: its minimum distance to any of the 10,470 reference genomes. We then binned the 110 queries into 10 bins based on M (Table C.11). Finally, we added 10 plant genomes (Table C.12) to the set of query genomes in every bin. Plant species are from a different domain of life compared to the reference set and should not match the library; thus, they allow us to measure FP and TN rates.

We built Kraken-II reference libraries for selected k values (ranging from 23 to 35) using the 10,470 bacterial and archael reference genomes. Kraken-II only allows adding additional custom genomes of interest to its existing standard reference libraries. We used Kraken-II RefSeq viral genome database as a base library. All custom reference libraries were constructed without masking low complexity sequences.

We used the ART simulation tool [88] with HiSeq 2500 single read profile, 150bp read length with 10bp standard deviation to generate ≈1.4GB of synthetic reads (1000× coverage for each genome) for all query genomes. Every query genome was then downsampled to 1G for normalization purposes.

Reads in each query bin were queried against every constructed reference library for each k using several confidence levels (0 – 0.3). We then calculated TP, FP, FN, TN for every bin. TP is the count of bacterial/archael reads matched to Bacterial or Archaeal domains; FP is the count of plant reads matched to Bacterial or Archaeal domains; TN is the number of plant reads that are left unclassified by Kraken; FN is the number of unclassified bacterial/archaeal sequences. We use standard definitions of FPR=(FP)/(FP+TN) and Recall=TP/(TP+FN) and construct ROC curves in the standard fashion for every tested condition.

Skmer distances (simulation). We next study the impact of contamination on distances computed from pairs of genome skims simulated from Drosophila assemblies. We first emulate the disjoint contaminant scenario by contaminating one of the two genome skims at a level c_l. We used *D. simulans w501* to simulate the contaminated genome skim and used *D. simulans WXD1*, *D. sechellia*, or *D. yakuba* to simulate the uncontaminated skim. Based on assemblies, the distances between *D. simulans w501* and the three other species are 0.2%, 2.1%, and 6.3%, respectively, and we treat these as true distances. To add contamination, we use the same 110 query genomes described earlier but bin M into four ranges: $[0,0]$, $(0,0.05]$, $(0.05,0.15]$, and $(0.15,0.25]$, which include 10, 43, 19, and 17 species, respectively, corresponding to a total size of 37Mb, 76Mb, 37Mb, and 35Mb. Since our base Drosophila genomes are roughly 150Mb in

size, we can add up to 25% contaminant reads for all bins, except for the $(0, 0.05]$ bin, where we can add up to 60%. We concatenated all the genomes in each bin and used ART with the same settings indicated above to generate contaminant reads, which we then mixed with reads simulated from the main genome at levels varying from 0% to 60% (for the second bin) or to 25% (for all other bins) for a total of 0.1Gb per skim (thus, no more than $1\times$ coverage). These read contamination levels translate to similar k-mer contamination levels (Table C.3). We report the relative error in estimated distances as we increase the contamination level, both with and without Kraken-II filtering. Kraken is run with the same reference library used in the previous analysis.

We then simulate a scenario where *both* genome skims are contaminated with *overlapping* sets of species. Here, we only use the $M \in (0, 0.05]$ bin and fix read contamination level to 15%. To control H, we randomly split bacterial reads into three parts: two unique parts and one part that served as an overlap. Every sample was generated by mixing unique and overlap contaminant portions with Drosophila genome skims at controlled ratios, with overlap set to 0% – 50%. Since unique parts can have evolutionary similar species, even the case of 0% overlap results in some k-mer overlap. Thus, we estimated contamination overlap (H) empirically using Jellyfish [139] and saw it varied between 11% and 41% (Table C.4). Finally, to have $H = 0\%$, we added the disjoint set experiment with $M \in (0, 0.05]$ and $c_l = 15\%$ to this set as well.

Skmer distances on real data. To move beyond simulations, we also evaluate effectiveness on real data with real contaminants. To do so, we utilized data from recent Drosophila assembly study by Miller et al. [155]. We subsampled available short read sequencing data (e.g., SRA files) to obtain 100Mb genome skims for 14 Drosophila species. We removed adapters, deduplicated and merged paired end reads using BBtools [40]. Then, we determined distances for all pairs of genomes before and after filtering them with Kraken-II. Distance error for every pair of genomes was estimated relative to the true distance defined to be the value computed by running Skmer on corresponding assemblies. In this experiment, we used a standard reference library available

from Kraken-II distribution. This database includes RefSeq assemblies of all available bacterial, archaeal, viral and human (GRCh38) genomes as well as the UniVec_Core subset of the UniVec database (a total of 168483 genomes, as of July, 2019). We used default Kraken-II settings.

Impact of filtering on phylogeny. On the real Drosophila data, we also infer phylogenetic trees from distances and measure phylogenetic error. To estimate the phylogeny, we use FastME 2.0 software [115] with JC69+Γ [98] model of evolution. Alpha parameter of JC69+Γ model is set equal to 1, which is the default value in FastME. We infer phylogenies from distance matrices obtained from assemblies and from genome skims before and after filtering. As the gold standard reference tree used for error calculations, we use the tree obtained from Open Tree of Life (OTL) [84](Fig. B.4). We estimated branch lengths of the true tree using OTL tree topology and assembly distances under JC69+Γ model. We measure phylogenetic error using three metrics. (1) Normalized Robinson and Foulds [197] (RF) distance is total number of branches not matching. (2) Normalized weighted RF (wRF) distance is similar but each present or absent branch in each tree is weighted by the absolute difference between its lengths in the two trees, and then the total sum is normalized by the sum of branch lengths of the two trees. (3) Fitch-Margoliash [69] is the weighted least squares error (FME) for species i, given as: $Q(i) = \sum_{i \neq j} (D_{ij}/d_{ij} - 1)^2$ where D_{ij} is the (corrected) distance between species i and j, and d_{ij} is sum of the branch lengths on the path connecting i and j on the phylogeny inferred using D. We also report cumulative FME of a phylogeny, which is $Q = \sum_{i=1}^{N} Q(i)$. Denoting FME error on true and estimated phylogenies with $Q(i)$ and $\hat{Q}(i)$ respectively, relative FME error is defined similarly to (1.3).

1.3 Results

1.3.1 Sensitivity of Kraken-II (FN and FP analysis)

The ability of the default version of Kraken-II ($k = 35, \alpha = 0$) to find a match in the database is a direct function of M, the distance of the query to the closest match (Figs B.5

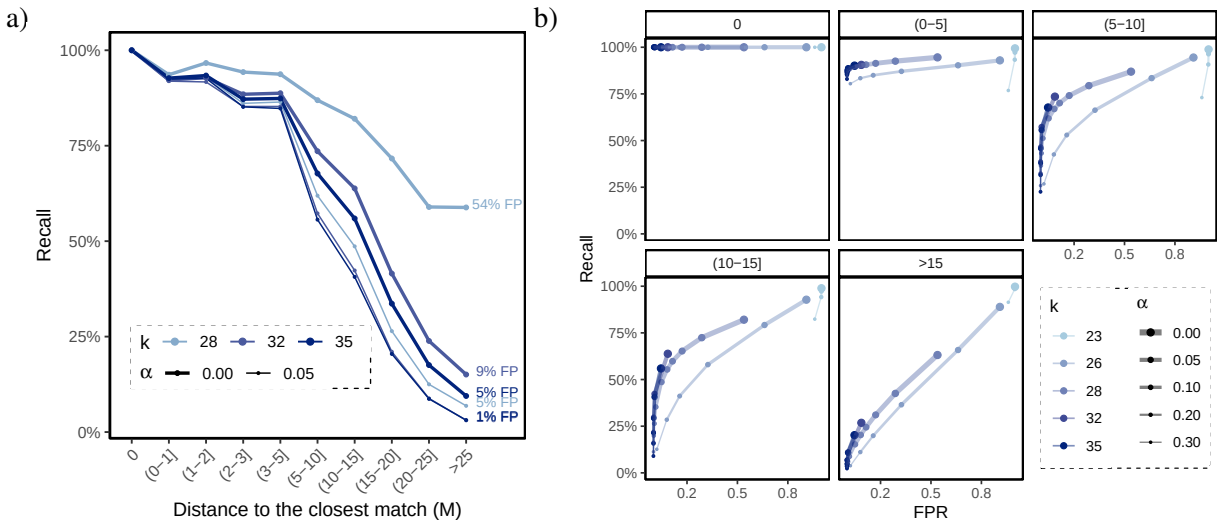


Figure 1.3. Sensitivity analysis of Kraken-II. (a) For three selected values of k and two selected values of α, the lines show recall for different bins of M (x-axis). Each line is labelled with its associated FPR. Note that FPR is a function of the plant genomes, which are identical across bins; thus, FPR is not a function of the match parameter M. (b) ROC curves for all k and α values across different bins of M, reduced to 5 ranges for ease of visualization (boxes). All k were tested with α $\{0.0, 0.05, 0.1, 0.15, 0.2, 0.3\}$. Additionally, k=28 was tested with α $\{0.01, 0.02, 0.03, 0.04\}$. See Figure B.7 for all 10 bins.

and 1.3a). When the query has a close match in the library (e.g., $D < 0.05$), Kraken-II is able to match 80% – 100% of reads, which would result in tolerable f_n rates of 20% or less. As M increases, the ability of Kraken-II to classify degrades linearly with M up until around $M \approx 0.3$ where Kraken-II fails to classify almost all reads (Fig. B.5). Interestingly, when Kraken-II finds a match, it is often able to classify the read all the way down to the species level (Fig. B.6).

Consistent with these results, when mixed plant/microbe skims are queried using the default Kraken, the recall of the filtering step is reasonably high (e.g., $> 85\%$) and the FP is low (4.5%) for $M \leq 0.05$ (Fig. 1.3a). When $0.05 < M \leq 0.1$ or $0.1 < M \leq 0.15$, there is a substantial reduction in the recall to 67% and 56%, respectively; for $0.15 < M$, recall is less than 33%, and thus filtering is not effective in those conditions.

Given the low recall in some conditions and our expectation that FP error is less damaging than FN, one may aspire to increase the sensitivity of Kraken-II by adjusting its parameters k and α. However, our careful analysis of FP versus FN shows very limited ability to control the rates

in a reasonable range (Figs. 1.3ab and B.7). Many settings of k and α result in FP error above 50% and often close to 100%. Many of the settings also have high FP without improving recall compared to default settings (Figs. 1.3b). The only settings that seem to provide a reasonable trade-off between FPR and recall are $k \in \{35, 32, 28\}$ and $\alpha \leq 0.05$. Focusing on these settings (Fig. 1.3a), we observe that setting $k = 28$ and $\alpha = 0$ provides a substantial increase in recall but increases FPR to unacceptable levels (55%). $k = 32$ improves recall compared to the default setting for $M \geq 0.05$ bins by a consistent but relatively small margins (5–8%), but also increases the FPR to 8.5%. Overall, changing parameters do not result in substantial improvements over the default settings.

1.3.2 Impact of filtering on Skmer distances (simulated contaminants)

Disjoint contaminants. Focusing on simulated contamination between pairs of Drosophila genome skims, when only one species is contaminated, increasing the contamination level results in increasing error in estimated Skmer distances, going up to 90% error for $D = 2\%$ and 1000% for $D = 0.2\%$ when $c_l = 60\%$ (Fig. 1.4a). As theory suggested, here, the strongest detrimental effect appears for $D = 0.2\%$.

Filtering using default Kraken-II dramatically reduces the error *when* the contaminant has an exact or close match in the reference library (Fig. 1.4a). For $M \leq 0.05$, remarkably high levels of contamination are tolerated after filtering. For example, for $0 < M \leq 0.05$ and $D = 2.1\%$, even with high c_l in 25% – 50%, distances have only 0.3% – 4% relative error after filtering. For $D = 6.3\%$, error after filtering is never more than 5% for $M \leq 0.05$. Even in the most challenging case of $D = 0.2\%$, $c_l = 25\%$ leads to only 6% error after filtering in contrast to 206% error before filtering. Despite the improved accuracy overall, in some cases, filtering can increase the error slightly but noticeably, perhaps due to FP filtering of correct reads. For $M = 0$ and $D = 6.3\%$, if contamination is below 5%, no filtering is better than filtering, which always results in ≈0.6% relative error regardless of the level of contamination. Interestingly, in some cases, filtering can result in *underestimation* of distances (e.g., up to 1% for $D = 2.1\%$ and

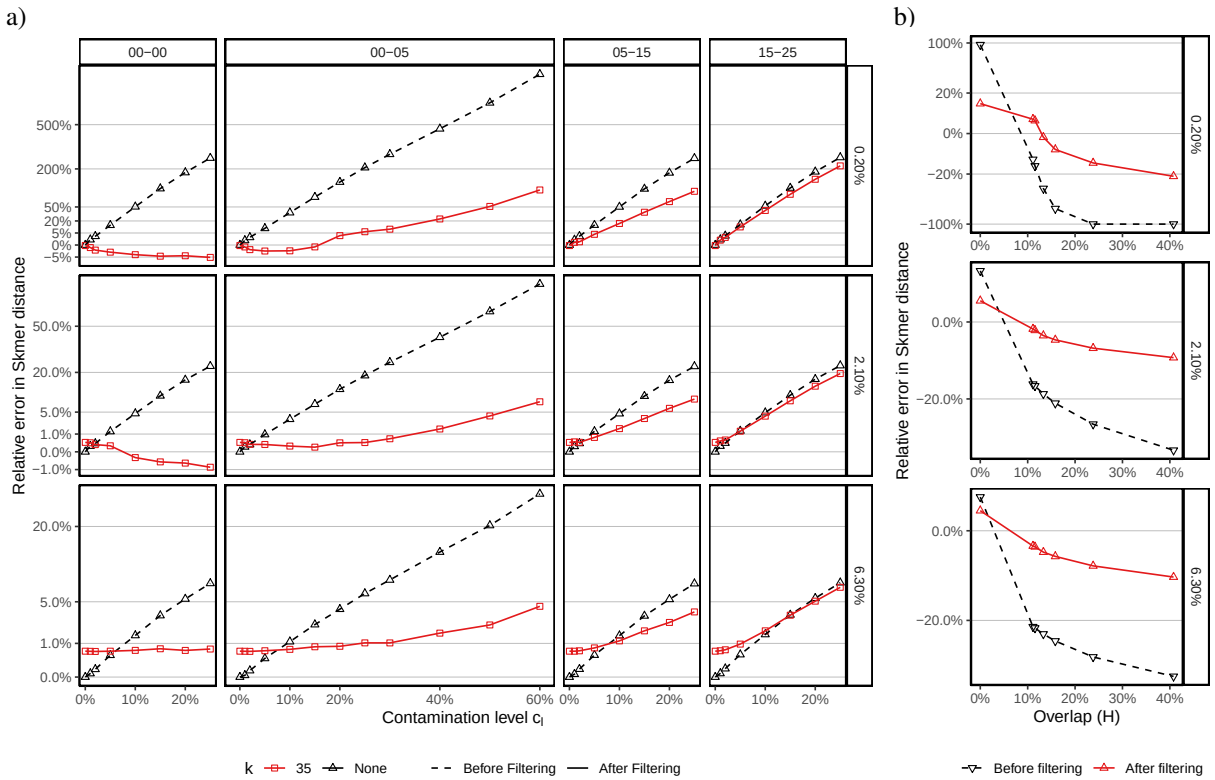


Figure 1.4. Filtering on simulated Drosophila genome skims. Relative error of Skmer distances without (dashed) and with (solid) Kraken-II filtering. Three pairs of Drosophila are chosen to be at true distance $D = 0.2\%$, $D = 2.1\%$, or $D = 6.3\%$ (rows). Contaminants are selected such that they are at distance M from their closest match in the reference contaminant library. (a) Simulating contaminants in only one of the two species (disjoint contaminants) for four ranges of M (columns) and various levels of contamination (x-axis). (b) Contaminating both genomes such that the overlap between contaminants measured by Jaccard similarity is $0\% \leq H \leq 41\%$. Here, $c_l = 15\%$ per species and $M \in (0, 0.05]$. Y-axis is on square root scale; see B.8 for normal scale and a range of k and α values.

$M = 0$).

In contrast, for contaminants without a close match with $M > 0.05$, filtering fails to fully remove contaminants. Nevertheless, for $0.5 < M \leq 0.15$, filtering has substantial benefits. For example, error is reduced from 180% and 16% with no filtering to 65% and 6%, respectively for $D = 0.2\%$ and $D = 2.1\%$. These reductions, while substantial, may not be sufficient. Even worse, for $M > 0.15$, filtering has very little or no ability to reduce the error and decreases or increases the error by very small margins.

Finally, changing k, α settings of Kraken-II does not consistently improve the accuracy

above and beyond the default setting (Fig. B.8). Using $\alpha = 0.05$ can very slightly reduce the error for the $D = 6.3\%$ case but is not dramatically different. Thus, we will exclusively use the defaults in the next experiments.

Overlapping contaminants. When both skims are contaminated with overlapping species, as theory suggested, we see *under-estimation* of distances (Fig. 1.4b). These under-estimations can be dramatic, going all the way down to -100% (i.e., the estimated distance is 0). Once again, filtering using Kraken is able to improve results dramatically, resulting in relative error that does not exceed 23% for $D = 0.2\%$ and is at most 6% in the remaining cases.

1.3.3 Impact of filtering on Skmer distances (real contaminants)

In the experiment on real unassembled Drosophila sequences, absent any filtering, Skmer often under-estimates distances (Fig. 1.5a). The under-estimation of distances is consistent with our theory assuming $H > 0$ (Fig. 1.2). Kraken-II run on these data identifies between 5.5% and 15.1% of the reads as belonging to human or microbes (Fig. 1.5b). Interestingly, for most Drosophila species, Kraken-II assigns $\sim$40–50% of the matched reads to one of three genera (Homo, Acetobacter, and Clostridium), indicating that many pairs of genome skims have similar contaminants (i.e., $H > 0$). Therefore, the under-estimations of distances matches the theory. Consistent with this explanation, we observe that the error in computed distances is associated with the percentage of the reads found by Kraken-II to be of human or microbial origin (Fig. B.9).

Filtering reads using Kraken-II dramatically reduces the errors in Skmer distances (Fig. 1.5c). Over all pairs, the mean absolute relative error reduce from 9.1% before filtering to 3.4% after filtering. In some cases, reductions are dramatic. For example, the relative error in pairwise distances between *D. virilis* and *D. bipectinata*, *D. eugracilis* and *D. mauritiana*, decreased from 46.2%, 36.9% and 35.9% before filtering to 1.3%, 0.8%, and 1.0% after filtering. In a minority of cases, error increased after filtering but the increase in error never exceeded 8% (*D. mauritiana* vs. *D. mojavensis*) while reductions in error could be as high as 45% (*D. virilis*

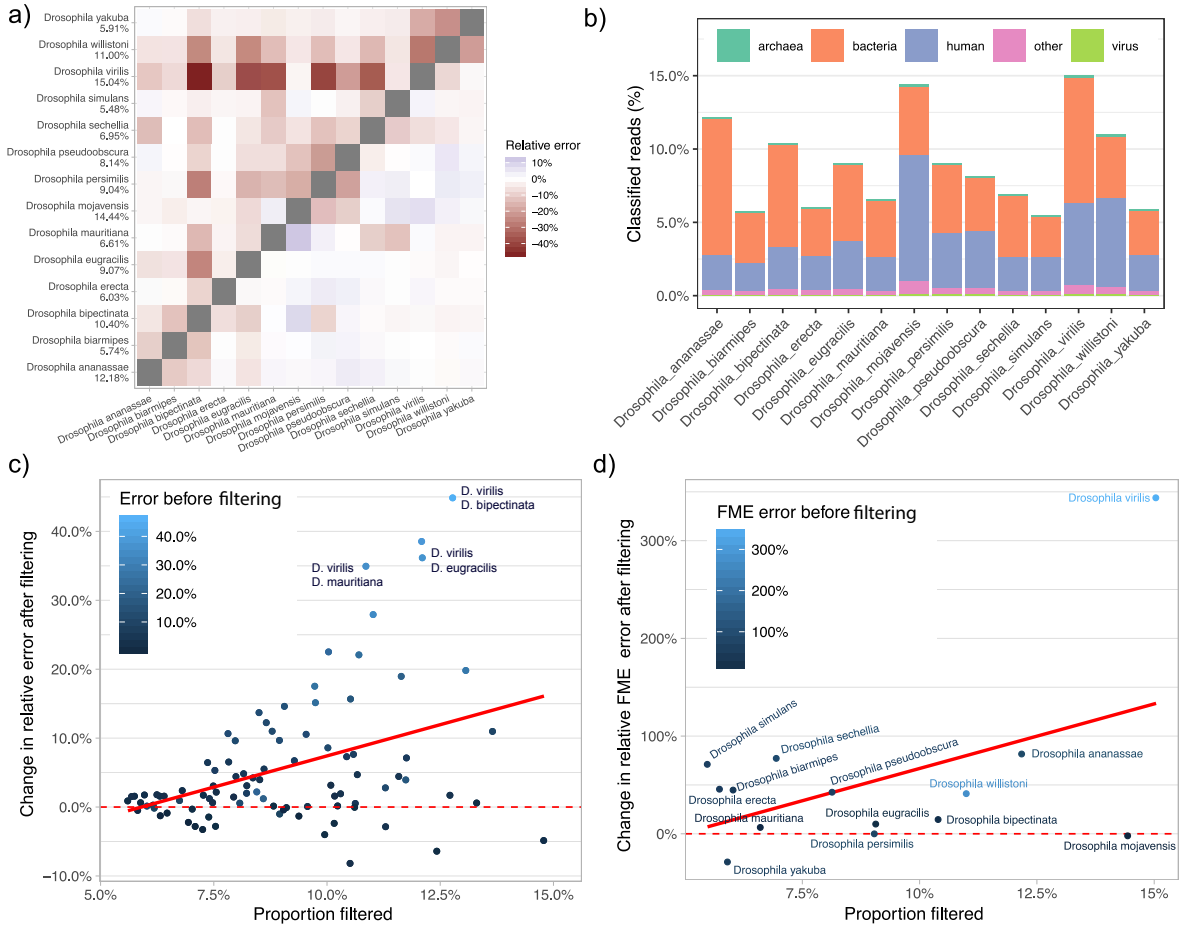


Figure 1.5. Filtering of real contaminants. (a) Relative distance error before (upper triangle) and after (lower triangle) filtering per pair of Drosophila species. Numeric lables on y-axis represent percentage of reads filtered per species. (b) Percent of reads classified by Kraken-II to different groups. "Other" corresponds "cellular organisms" (shared between domains). (c) Change in the relative distance error after filtering. Positive values indicate a reduction in error. Highlighted dots correspond to Drosophila pairs mentioned in the text. (d) Change in relative FME error per species after filtering. Solid red line: a trend line fitted to the points.

vs. *D. bipectinata*) (Fig. 1.5c). The wide range of error reductions is unsurprising given that the actual level of contamination in original sample can vary substantially. The magnitude of improvement in distance estimates is positively correlated with the percentage of reads filtered (Fig. 1.5c), the genomic distance (Fig. B.10a), and the magnitude of the error before filtering (Fig. B.10b).

When there is error after Kraken-II filtering, it tends to be due to *over*-estimation of

distance, as opposed to under-estimation observed before filtering, suggesting that Kraken-II perhaps over-filters reads. Most extreme cases of over-filtering involve distance estimates between a single species, *D. mojavensis*, and other species such as *D. bipectinata*, *D. mauritiana* and *D. virilis*. The *D. mojavensis* is the only species with high levels of Kraken-II filtering but low error rates in pairwise comparisons. Interestingly, *D. mojavensis* also includes the highest levels of contamination from unknown sources.

1.3.4 Impact on phylogenetic reconstruction

The phylogeny inferred from Skmer distances computed from the assembly and modeled using JC69+Γ is topologically identical to the gold standard OTL phylogeny, and its total FME error is only 0.03 (Fig. B.4). However, the phylogeny estimated using the same method but using genome skims has two wrong branches (RF = 4), and a FME of 1.26. Thus, absent filtering, genome skims produce trees with substantial error.

Improvements in estimated genomic distances due to filtering translate to improved phylogenetic trees. The tree topology improves only slightly and has one incorrect branch (RF = 2) after filtering. However, the improvements in estimated branch lengths, as reflected in wRF and total FME error, are dramatic. Filtering leads to nearly 70% decrease in total FME metric, from 1.26 to 0.38, and a similar level of reduction is observed for wRF (Table C.5). Examining individual branch lengths, the phylogeny using filtered data is much more similar to the true tree (Fig. B.4).

When we use FME to measure the impact of filtering on the phylogenetic error of individual species, we observe patterns consistent with reductions in distance error (Fig. 1.5d). Individually, majority of species have reduced FME after filtering, with the most extreme FME reduction happening for *D. virilis* by nearly %350. Consistent with previous results, we observe that the FME error of *D. mojavensis* does not decrease (but it also does not increase). As expected, gains in phylogenetic error measured by FME correlate with the amount of filtering performed by Kraken (Fig. 1.5d). However the correlation disappears when the species *D. virilis*

is excluded from the analysis (Fig. B.11).

1.3.5 Running time

We assessed running time performance of Kraken-II on skims of five randomly selected species from different domains of life (Table C.6) using both single-threaded and multi-threaded (24 threads) modes of operation. Run time was found to linearly increase with the skim size, regardless of the number of threads used (Fig. B.12). With 1Gb of reads, the running time of the single-threaded version was below 100 seconds on an Intel Xeon CPU in all cases we tested. Running Kraken-II with 24 threads reduced the speed by a factor of 10, offering a significant improvement. The main limitation with Kraken-II is its significant memory requirement during queries, which requires between 100Gb and 120Gb for our reference libraries.

1.4 Discussion

The use of genome skimming in the literature has mostly relied on assembled organelle genomes [e.g., 45, 59, 136, 244]. These approaches rely on assembly construction pipelines [e.g., 96] to remove contaminants (i.e., to avoid mis-assembly or to filter out mis-assembled contigs). Elsewhere, we have advocated going beyond organelle genomes and using all reads in an assembly-free fashion to increase the resolution of taxonomic identification [18, 204]. However, this goal has been hampered by the presence of contaminants. This study showed a relatively effective way of dealing with contamination, hence bringing genome skimming based on nuclar reads one step closer to a reality.

Our study showed that Kraken-II is able to find contaminants that are within 5–10% genomic distance to the closest match in a reference library in a computationally efficient manner. Our modeling showed that FP errors were perhaps less detrimental to distance calculations than FN. Analysis of different k and α parameters did not reveal parameter combinations that could improve upon the using default settings of Kraken-II ($k = 35, \alpha = 0$). Analyses of real data demonstrated that contamination removal can dramatically improve Skmer distance estimates in

the presence of contaminants. These more accurate distance metrics computed after filtering can lead to reduced phylogenetic branch length error by up to 70% and can also improve the tree topology.

1.4.1 Usefulness of Theoretical Models

Simplified assumptions allowed us to establish theoretical models of the impact of contamination on estimated distance. The theory predicted that even small levels of similarity between contaminants (H) can lead to substantial under-estimation of distance when distance is large. Consistently, on the real data, where distances are often > 0.10, we observe under-estimation by 5% or more in 47 out of 91 pairs. Our results also showed high levels of similarity between contaminants of Drosophila genomes (where three genera made up 40–50% of contaminants). Thus, there is a reassuring match between the theoretical model and the observed data.

Kraken filtering improved accuracy on simulated and real data. On real data, it occasionally over-corrected errors, leading to over-estimation of the distance. These may be due to FP filtering, reduced coverage after filtering, or other factors not fully understood here. In our runs of Kraken-II on real data, we observed 5%–15% filtering. The lower value can be explained by $\sim$5% Kraken-II FP rate when run under its default setting. The upper value is consistent with $\sim$10% contamination level, a scenario that can happen in real sequencing projects [149, 203].

Another potential use of the theory could have been developing filter-free methods of dealing with contamination. Just as impact of coverage and error on Jaccard can be modelled, we can compute the Jaccard index with no filtering but *correct* for the modelled impact of contamination on Jaccard. Given reliable estimates of H, c_1, and c_2, we can manipulate (1.4) to update (1.1) and obtain:

$$\hat{D} = 1 - \left(\frac{(2+a)J}{1+J} - \frac{aH}{1+H} \right)^{1/k}.$$

Adding coverage and error models, we can update (1.2) to:

$$\hat{D} = 1 - \Big(\frac{(2+a)(\zeta_1 L_1 + \zeta_2 L_2)J}{\eta_1 \eta_2 (L_1 + L_2)(1+J)} - \frac{aH}{1+H}\Big)^{1/k} \tag{1.6}$$

This equation allows for filter-free contamination-aware distance calculation. Unfortunately, however, this equation is extremely sensitive to correct estimation of all parameters, including H, c_1, and c_2 (Fig. B.13). Even small mistakes (1-5% relative error) in the estimated contamination level or Jaccard can lead to dramatic errors in the estimated distance computed using (1.6). Since computation of these parameters is noisy, we do not advocate this filter-free method despite its theoretical elegance.

1.4.2 Filtering methods

Filtering requires a tool to answer queries of the following type: "Does this particular read belong to one of the genomes in a given reference library?". We chose Kraken-II for answering these queries because of its high accuracy and reasonable scalability, as established in several bench-marking studies from the metagenomics field [146, 151, 211, 254]. It is also one of the most widely-used tools, with an active user support and stable and robust software development. Further, we explored three parameters of Kraken-II: k-mer length, confidence score and database content. Other parameters such as minimizer length (mostly relevant to storage and not accuracy) and minimizer space count are not explored here [36]. In our experiments we kept the number of wild-carding positions at its recommended upper limit and turned masking off but there might be a set of settings which in combination with masking can produce a more optimal sensitivity. We note that results from Wood et al. [250] have indicated that Kraken-II is not very sensitive to particular parameter settings.

Alternatives to Kraken-II exist, and future studies can compare them to Kraken-II for genome skimming. BLAST [4] and MegaBLAST [159] are the obvious alternatives but are an overkill for our problem. These tools perform alignment and can yield higher sensitivity than

Kraken-II but are orders of magnitude slower [251, 254]. However, they produce more precise results (maps to individual species) than what we need.

Beyond alignment tools, most alternatives to Kraken-II are also *k*-mer-based, but differ in the way reference library is constructed and how the query is run. *k*-mer-based methods inculde LMAT [5], and CLARK(-S) [171, 172]. Benchmarking studies [e.g., 146, 151, 211, 254] do not indicate any consistent advantage in using these methods over Kraken-II, and many of them are slower. Among sufficiently fast tools are KrakenUniq [34], Bracken [127], and Centrifuge [105]. KrakenUniq is recommended for use in cases where FP can be detrimental (e.g. in pathogen identification/diagnoses), but our theory and empirical data suggest FP is less important and FN in our application. Bracken [127], an extension of Kraken-II, is focused on improving aggregated abundance profiles, a feature that is irrelevant to our usage. Centrifuge [105] uses FM-index lookups and within-species compression for mapping a read to one or more species. Compared to Kraken-II, Centrifuge is slower and needs more time for building its reference database. We leave its comparison to Kraken for future work.

A separate set of *k*-mer-based methods have been developed for finding RNAseq experiments that include a specific *k*-mer. [220] introduced Sequence Bloom Tree (SBT) to allow very fast queries of a *k*-mer versus a reference set of experiments by creating a hierarchy of compressed bloom filters that store *k*-mers. Mantis [174] is an alternative to Bloom filters based on counting quotient filters and is reported to be more memory efficient and faster that SBT-based methods. While these tools have been developed mainly for RNASeq analyses, in the future, they can perhaps be adopted for mapping reads to genomes with minimal changes to the algorithm. In fact, Kraken might implement counting quotient filter data structure in its future releases [250].

Beyond these tools, many other metagenomic methods have been designed for finding the taxonomic composition of a mixed sample [e.g., 125, 154, 165, 212]. However, these tools do not seek to classify *every* read from anywhere in the genome; they are either marker-based or use composition data. Thus, these tools are irrelevant to our queries.

1.4.3 Remaining gaps

In our study, we focused solely on prokaryotic and human contamination. Real contamination is more complex and can include eukaryotic microorganisms, traces of endosymbionts and diet, and various forms of lab contamination. Thus, many applications will benefit from more inclusive Kraken-II contaminant libraries. At a minimum, fungi need to be considered, especially for plants. Moreover, removing reads from organelle genomes, which are expected to be over-represented, may further improve accuracy.

Luckily, Kraken-II enables a straightforward mechanism for extending reference libraries. Our future efforts will include building a larger library of potential contaminants that includes fungi and perhaps expected sources of diet. However, such libraries will have to be group specific; for example, for skimming insects, we can treat plants as contaminants whereas in skimming plants, we should treat insects as contaminants. Ideally, individual genome skimming reference libraries for a target group (e.g., all insects) should be furnished with a relevant contaminant library especially designed for that group based on the knowledge of taxonomic groups expected to be present in its diet and its endosymbiont. Clearly, this approach runs into its limitations when endosymbionts or the diet happen to be from species with similar genomes to the target species.

The fundamental limitation of our exclusion filtering approach is that we need to know what broad group of species is expected to contaminate. This limitation is a result of our implicit assumption that a read is correct unless we find evidence to the contrary. Even when such biological knowledge is available – it may not be – this approach can fail to capture lab-introduced contamination (e.g., a plant species that was contaminated with fish due to failures in sample preparation or sequencing on the same lane).

Another issue is the inclusion of the human genome in the reference libraries to find human contamination. When the target species is a mammalian species, reads that belong to the target may incorrectly map to the human (false positive). For example, in a test analysis, we

observed that Kraken-II maps roughly 20% of reads from rhinos to human. Luckily, this problem has a simple solution. We can require a higher α when mapping to human. Thus, all reads can be searched against the library with the default $\alpha = 0$, and those reads that map to human can be mapped again with a high α; reads are classified as human contamination if they continue to map at the higher α. We tested this approach on the rhino genome and observed that with $\alpha = 0.5$, only 2% of reads map to human. We have provided a script for performing this two-step filtering.

Inclusion filtering is an attractive alternative to exclusion filters. Given a reference database of purified (perhaps using exclusion filters) genome skims, we can build a Kraken reference library from species in the skimming reference library. Then, for every new query genome skim, we can use that library to find reads that seem to match the broad taxonomic group of interest and only *include* those reads in the calculation of Jaccard. Our results indicate that this method would work only if the skimming reference database is so dense that each new query skim is expected to have a close match (e.g., $<$5%) to one of the reference skims. Moreover, this approach is predicated on the reference library being free of contaminants. Despite these shortcomings, we believe this approach should be further explored in the future.

Finally, better algorithms for read matching seem necessary. Our results showed that Kraken-II provides a reasonable solution. Nevertheless, the method remains incapable of finding domain level matches when the closest match is moderately distant from the query. We believe it is possible to design more sensitive read mapping techniques that can match a species even when its closest match is $>$ 10% distance. Note that in genome skimming, we are only interested to know whether a read belongs to a large taxonomic group, as opposed to metagenomics, when abundances and exact matches are desired. Given the less demanding needs of the skimming application, we anticipate that better algorithms can be developed in future to increase recall with little or no loss of specificity and speed.

1.5 Availability of data and materials

Scripts and summary data tables are publicly available on https://github.com/noraracht/kraken_scripts.git. Raw data used in the manuscript is deposited in https://github.com/noraracht/kraken_raw_data.git. Additionally, data repositories are stored in zenodo https://doi.org/10.5281/zenodo.3588625 [188] and https://doi.org/10.5281/zenodo.3588569 [189]. The detailed description of genomic datasets used in our experiments, accession numbers of the assemblies and the exact commands used to simulate genome skims are provided in Supplemental Material.

Chapter 1, in full, is a reprint of the material as it appears in The impact of contaminants on the accuracy of genome skimming and the effectiveness of exclusion read filters. Rachtman, E.; Balaban, M.; Bafna, V., & Mirarab, S., Molecular Ecology Resources, 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 2

CONSULT: Accurate contamination removal using locality-sensitive hashing

A fundamental question appears in many bioinformatics applications: Does a sequencing read belong to a large dataset of genomes from some broad taxonomic group, even when the closest match in the set is evolutionarily divergent from the query? For example, low-coverage genome sequencing (skimming) projects either assemble the organelle genome or compute genomic distances directly from unassembled reads. Using unassembled reads needs contamination detection because samples often include reads from unintended groups of species. Similarly, assembling the organelle genome needs distinguishing organelle and nuclear reads. While k-mer-based methods have shown promise in read-matching, prior studies have shown that existing methods are insufficiently sensitive for contamination detection. Here, we introduce a new read-matching tool called CONSULT that tests whether k-mers from a query fall within a user-specified distance of the reference dataset using locality-sensitive hashing. Taking advantage of large memory machines available nowadays, CONSULT libraries accommodate tens of thousands of microbial species. Our results show that CONSULT has higher true-positive and lower false-positive rates of contamination detection than leading methods such as Kraken-II and improves distance calculation from genome skims. We also demonstrate that CONSULT can distinguish organelle reads from nuclear reads, leading to dramatic improvements in skims-based mitochondrial assemblies.

2.1 Introduction

Despite the decreased cost of whole-genome sequencing, carrying out large-scale cohort studies of non-human species using assembled genomes is still daunting [198]. Low-cost sequencing projects remain an attractive alternative in biodiversity and ecological research [59, 237]. Such studies can include a large number of samples sequenced at 1-2$\times$ read coverage, often called genome skims [45, 225, 244]. Traditionally, genome-skimming data were used for assembling the over-represented organelle genome using one of several approaches that have been developed [1, 3, 8, 41, 57, 77, 97]. More recently, noting that skimming also produces a large number of unassembled reads from the nuclear genome, researchers have been inspired to use those unassembled reads to answer questions of interest in area of biodiversity, including sample identification and population genetics [30]. This vision can be realized using assembly-free and alignment-free methods where bags of unassembled reads represent *both* the labeled species (i.e., reference) and the new sample that need to be identified (i.e., query), and these bags of reads are directly compared. This vision has been pursued by several methods that enable computing distances among skims [64, 112, 204, 231, 258] and using those distances for phylogenetic placement [14, 18].

The broad use of skimming data for biodiversity is within reach, but a significant hurdle remains: contamination. Analyzing raw unassembled reads without mapping to reference genomes is particularly vulnerable to the presence of extraneous sequencing reads that do not belong to the species of interest [53, 163]. Foreign DNA originating from parasites, symbionts, diet, bacteria, and human are often mixed in with supposedly single-species genome skims with the sequencing step further contributing to the contamination [10, 46, 201]. With a slight abuse of terminology, we broadly refer to all external DNA outside of the genomes of interest as contaminants. Such contamination has the potential to reduce accuracy of distances estimated from genome skims. Using theoretical modeling and experimental studies, [189] have shown that contamination can lead to over and underestimation of distances between genome skims

using assembly-free methods such as Skmer.

Examination of raw reads for contamination detection is not a new challenge. Early filtering techniques that relied on *k*-mer-coverage or GC content [145, 207, 234] missed contaminants frequently and were replaced by methods that use sequence similarity to search against libraries of potential contaminants [58]. The current practice is to re-purpose classification methods used in the taxonomic characterization of microbial metagenomes to identify extraneous reads in genomic datasets [58]. Metagenome classifiers use a variety of approaches, including read alignment as nucleotide or protein, *k*-mer mapping, and alignment of marker genes [28, 178, 252]. Among these, marker-based and protein-based methods cannot be used for contamination removal as they will only detect reads from markers or coding sequence (CDS) regions. Among the remaining methods, *k*-mer-based methods [e.g., 5, 171, 172, 250, 251] are widely-used and present a reasonable compromise between speed and sensitivity. In particular, thanks to its speed, accuracy, and user-friendly implementation, Kraken-II [250] is widely used.

For contamination removal, unlike taxonomic classification, we are interested in detecting the broad taxonomic group of a read. For example, given a skim from an insect, we seek to find reads that can be rejected as belonging to Arthropoda. Thus, reads clearly belonging to prokaryotes, fungi, or plants would be judged as contamination. Metagenomic classification tools *do* classify reads at high levels but have not necessarily been optimized for higher level classification. Instead, their goal has been increasing specificity (e.g., detecting species). While detecting higher levels should be easier in principle, methods remain inadequate.

A shortcoming of metagenomic tools is their reduced ability to match reads when evolutionary close species are not available in the reference set [124, 162, 173, 242]. Much of the microbial diversity on earth is not reflected with close representatives in the reference datasets [44, 60]. Thus, contamination removal tools should ideally identify the broad group of species generating a read even when the reference is *sparse*. Current methods are not sensitive enough. For instance, the phylum level classification lacks sensitivity when tested on novel data [173]. We recently showed [189] that even at the domain-level, the sensitivity of the

leading method Kraken-II [250] degrades dramatically as the distance to the closest match in the database increases above ≈8% demonstrating that even leading metagenomic methods have serious limitations for contamination removal. The limited sensitivity of methods has spurred the development of many reference sets [55, 135, 185, 214], including recent whole-genome databases with up to 25000 genomes [11, 175, 257]. However, despite their substantial size, these databases (or close to a million prokaryotic genomes available on RefSeq and GenBank databases) include only a fraction of the estimated 10^{12} extant microbial species [126]. Thus, better reference datasets are not enough; we need more sensitive sequence matching methods.

Sensitive read matching tools can also help the organelle-based use of genome skims. Assembling organelle genomes needs some way of telling apart nuclear and organelle reads. Existing methods rely on either a seed-and-extend method where a seed (e.g., COI barcodes, available for millions of species) is used to find a part of the organelle genome and seek neighbouring regions in the assembly graphs [57, 77]. An alternative is to rely on differential coverage of organelle and nuclear genomes to distinguish the two [1, 8]. Finally, when a close reference genome is available, using that genome and read mapping can be used [e.g., 41]. However, given that more than 10,000 organelle genomes from across the tree of life are already available in RefSeq, an alternative approach seems fruitful. We can build a database of all existing organelle genomes and use a sensitive read matching tool to find which reads look like they belong to the organelle genome. The assembly can then proceed simply using reads that match the database at some distance.

In this paper, we introduce a read matching method and apply it to both contamination removal and organelle read detection. Our method, called CONSULT (CONtamination Spotting Using Locality-sensitive hashing Techniques), uses k-mers in a query sequence to search a reference database and detects whether any of the k-mers match any sequence in the database allowing for inexact matches up to a user-defined threshold. The general strategy is similar to Kraken-II, except CONSULT allows mismatches using the Locality-sensitive hashing (LSH) technique; however, unlike Kraken-II, CONSULT does not currently produce taxonomic assign-

ments. We compare CONSULT to leading methods both as a contamination removal tool and as a pre-processing step to help organelle assembly and show its superior accuracy in both settings.

2.2 Materials and Methods

2.2.1 CONSULT

Background: LSH and the motivation to use it

Exact k-mer matching is not sufficient for matching reads to a database when the closest species in the reference set is evolutionary distant. Here, we always chose $k > 20$ so that k-mers are expected to be unique except when they derive from a common ancestor (e.g., repeats). Let us examine an example. Consider a case where the query genome is at distance $d = 0.15$ to its closest match M in the reference set. While due to lack of independence among adjacent k-mers, the probability of shared k-mers is hard to compute, we can still compute the expected number of k-mers shared between a read of length $L = 150$ and M, which is only $(L-k+1)(1-d)^k = 0.4$ for $k = 35$ (default in Kraken-II). Thus, most reads would not match the reference dataset. Existing methods have recognized the need for inexact k-mer matching. For example, Kraken-II masks 7 positions from each k-mer to increase the expected number of matches (1.3 in the previous scenario), allowing many but not all reads to match. Note that most methods avoid keeping *all* k-mers of reference genomes in the reference set, further reducing the expected number of matches per read.

We approach inexact matching using LSH, which is a widely-used hashing technique for clustering similar items or finding neighbors of a data points within a distance threshold [78]. LSH uses a family of functions that hash data points into buckets so that data points near each other, and *only* such data points, are located in the same buckets with high probability. An LSH requires hashing functions that guarantee the probability of two items with distance below a desired threshold p falling in the same bucket is higher than that of two items with distances greater than $a \cdot p$ for some approximation factor $a > 1$. LSH schemes are known for

many distances [6, 37, 50, 76, 108, 138, 161]. Among these, Hamming distance (HD) allows a straight-forward hashing scheme, described below, that is also fast to compute. Moreover, HD has a natural interpretation in terms of evolutionary distances and is readily defined between two k-mers. With more complex schemes such as MinHash (corresponding to Jaccard index) and edit distance, the interpretation of distances for k-mer is not immediately clear. Thus, here, we focus on HD.

To designing LSH for HD, we hash a k-mer by simply picking a random (but fixed) position in that k-mer, which would put two sequences of Hamming distance d in the same bucket with probability $1-\frac{d}{k}$. While this probability decreases with d (thus, forms a valid LSH), the probability reduces only linearly with d and is not expected to be effective. However, this simple hash function can be amplified using AND and OR constructions. Given two k-mers represented by l hash functions each constructed using h randomly-positioned bits (i.e., AND construction), the probability that *at least one* of the hash functions (i.e., OR construction) fall in the same bucket is:

$$\rho(d) = 1-(1-(1-\frac{d}{k})^h)^l \tag{2.1}$$

By varying k, l, and h, we can control $\rho(d)$. Note that $l=1$ and $h=k-s$ reproduces the masking strategy used by Kraken-II (s is the number of masked bits). Ideally, $\rho(d)$ should be close to 1 for $d \leq p$ and should quickly drop close to zero for $d \gg p$. As shown in Figure B.14, $\rho(d)$ can produce an inverted S-shaped figure, and fixing k, many settings of l and h can lead to high $\rho(d)$ for low distances (e.g., $d \leq p = 3$) and much lower $\rho(d)$ for higher distances (e.g., $d > 6$).

CONSULT Algorithm

The inputs to CONSULT is set of reference genomes, represented as a set of k-mers, one or more query reads, and two adjustable parameter: c and p. It seeks to address the following problem: Are there at least c k-mers in a given read that each have at most distance p to some k-mer in the reference library? While the naive solution to this problem requires comparing

each k-mer in each query read to each k-mer in the library, CONSULT uses LSH to circumvent that need. To build its library, CONSULT saves reference k-mers in a LSH-based lookup table (Fig. 2.1a), further described below. At the query time, the lookup table enables CONSULT to compare a given k-mer to a small (bounded) number of reference library k-mers to compute the HD between the query and the reference k-mers. A read is called a match as soon as at least c reference k-mers are found that match the query k-mer, meaning that their HD is no larger than p (Algorithm 1).

Encoding k-mers. Let's assume we have up to 2^g k-mers in the reference set. Every reference k-mer is encoded in a $2k$-bit number and is kept in an *encoding array* of maximum size 2^g (Fig. 2.1a). We use a specific Left/Right encoding that allows very fast calculation of HD using a native `popcount` instruction, an XOR, an OR, and a shift (see procedures `LeftRightEncode` and `HD` in Alg. 1).

Lookup Table. To find a constant-size subset of k-mers for computing HD, we use LSH. Hash values are generated by randomly selecting h 2-bits at randomly-chosen (but fixed) positions from the $2k$-bit encodings, repeating the process l times to produce l *signatures*. When building the reference library, we save l one-to-many mappings from each k-mer signature to at most b encodings, implemented as a lookup table (Fig. 2.1a). When a k-mer is added to the encoding array, each of the l lookup tables is updated to point to its position, skipping a table if the corresponding row is full. The lookup tables should ideally allow around 2^g elements to accommodate all encodings, which can be achieved by setting $b \approx 2^{g-2h}$. At the query time, for each k-mer of a read and its reverse complement, we generate its l signatures (using a trick enabled by x86 instruction "extended shift" (`shld`); see `Signature` in Algorithm 3, supplementary material), which we use as index to a row of the lookup table; thus, the constant number of k-mers we will test equals $b \times l$.

Note that there is no guarantee that signatures will appear uniformly in the reference set

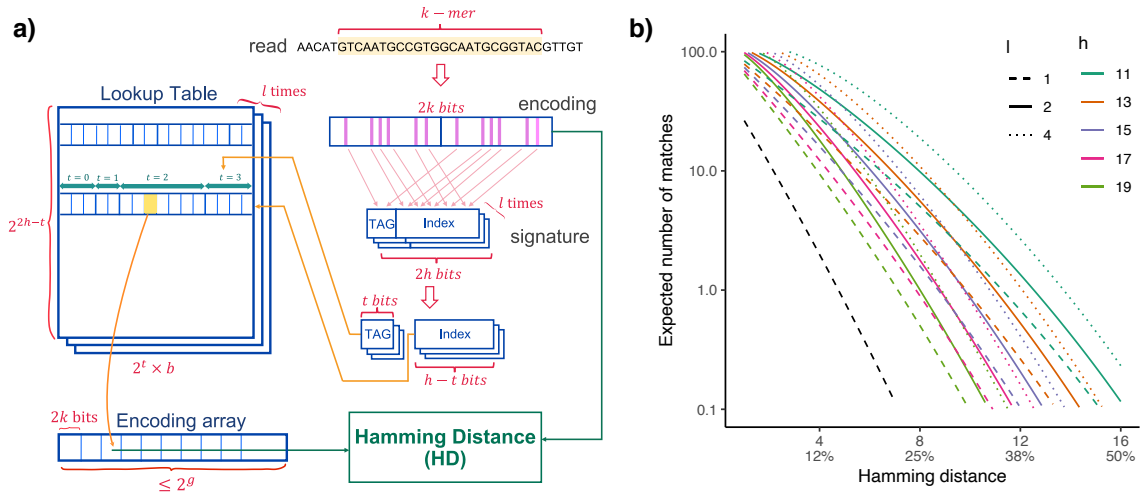


Figure 2.1. Architecture. (a) A set-associative lookup table is indexed by h randomly-selected LSH signatures extracted from each k-mer and points to the encoding array. (b) We show $(L-k+1)\rho(d)$: the expected number of k-mers that match a k-mer in a reference genome at distance d from the read for various h, l settings; $k=32$. Black line: $k=35$, $h=35-7$ and $l=1$, similar to default Kraken-II.

Algorithm 1. CONSULT algorithm. Here, we omit the set-associative design and tags and the fast `SHLD`-based computation of signatures for simplicity; for the more complete pseudocode, see Algorithm 3 in supplementary material. Notations: $\mathscr{S}$: all reference sequences. Defaults: $m=35, k=32, h=15, l=2, b=7, p=3, c=1, g=33$. $[a]$ denotes $\{0,\ldots,a-1\}$.

---

procedure BUILDLIBRARY($\mathscr{S}$)
  $E \leftarrow$ array of $\leq 2^g$ elements, each $2k$ bits
  for each $i \in [l]$ **do**
    $M_i \leftarrow h$ unique random numbers in $[k]$
    $S_i \leftarrow 2^{2h}$ array of lists of size at most b with g-bit elements
  $\mathscr{K} \leftarrow \emptyset$
  for each a in {all m-mers of $\mathscr{S}$} **do** ▷ Minimization
    Append min{all k-mers of a} to $\mathscr{K}$
  $j \leftarrow -1$ ▷ Pointer to E
  for each k-mer $a \in \mathscr{K}$ in pseudo-random order **do**
    $e \leftarrow$LEFTRIGHTENCODE(a)
    Included $\leftarrow$ False
    for each $i \in [l]$ **do**
      $s \leftarrow$ SIGNATURE(M_i, e)
      if $S_i[s]$ is not full **then**
        if not Included **then**
          $j \leftarrow j+1$
          $E[j] \leftarrow e$
          Included $\leftarrow$ True
        Append j to $S_i[s]$
  save $DB = (E, M, S)$ to disk

procedure SIGNATURE(M, e) ▷ Extract Signature
  return 2h-bit number with bits $i, i+32$ of e for $i \in M$

procedure LEFTRIGHTENCODE(a) ▷ Encoding
  $R \leftarrow 2k$-bit zeros
  for letter a_i in a **do**
    $R_i = 1$ if $a_i \in \{G, T\}$
    $R_{i+32} = 1$ if $a_i \in \{C, T\}$
  return R

procedure HD(a,b) ▷ Hamming Distance
  $z_{low}, z_{up} \leftarrow$ lower and upper k-bits of $a \oplus b$
  return `popcount`$(z_{up} \vee z_{low})$

procedure QUERYREAD(r, DB) ▷ Query a read
  $l \leftarrow 0$
  for k-mer a in r and its reverse complement **do**
    $e \leftarrow$LEFTRIGHTENCODE(a)
    for each $i \in [l]$ **do**
      $s \leftarrow$ SIGNATURE(M_i, e)
      for $h \in S_i[s]$ **do**
        if HD($E[h], e$) $\leq p$ **then**
          $l \leftarrow l+1$
          if $l \geq c$ **then**
            return r is a match
  return e is not a match

(Fig. B.15) and so some rows can fill up sooner than others. To avoid losing k-mers to imbalances (Table C.9), we use a set-associative lookup: The most significant t bits (default=2) of a signature are used as a tag and the remaining $2h-t$ bits as the index to the lookup table. Thus, the table has 2^{2h-t} rows, and each signature can have between 0 and $b \times 2^t$ entries. This design improves utilization of the table (Figs. B.15, B.16). We sort elements in each row by the tag and save tag boundaries.

HD calculation. Given that we use LSH, one may wonder why computing HD explicitly is necessary. LSH can only provide probabilistic guarantees: k-mers from very distant genomes have small but non-negligible chances of matching. Modern reference libraries for prokaryotes include $> 10,000$ representatives genomes [175, 257], leading to 8 to 20 billion unique k-mers *after* minimization (Table 2.1). Against such huge reference libraries, small probabilities of incorrect matches blow up. To guard against false positives, CONSULT makes sure a k-mer is called a match *only if* its actual hamming distance to a k-mer in the library is below p. Thus, LSH is not the final arbitrator of distance; it only helps reduce hamming distance calculations. A side-effect of computing HD is that it requires keeping reference library k-mers in memory. For example, to keep 8 billion 32-mers in memory (our target in this study), we need $8 \times 2^{33} = 64$G bytes for encodings. Luckily, modern server nodes have upwards of 128GB RAM, allowing this high memory usage.

Parameter settings. We will explore the parameters c and p in our experiments and will select default values. The choice of k, l, and h presents intricate trade-offs between memory, running time, recall, and precision. The total memory usage is roughly $2^{g-3} \times (2k + g \times l)$ bytes. Fixing $g = 33$, to fit the entire library in a 128GB memory machine, we need $2k + 33 \times l \leq 128$. Since $k > 20$ is needed for uniqueness of k-mers, we find $l < 3$. When the true distance of a read from a species in the database is d, the expected number of k-mers matching is $E_d = (L-k+1)\rho(d)$. Despite dependencies, the number of k-mer matches is distributed around the mean. To avoid

false positive matches, we want E_d to be far below 1 for high distances (e.g., $d > 40\%$) and to be high for small distances (e.g., $d < 15\%$). As Figure 2.1b suggests, both $l = 1$ or $l = 2$ allow this goal in theory. The expected number of matching k-mers is substantially lower for $l = 1$ than $l = 2$; for example, with $k = 32$, $h = 15$, and $d = 3$, we expect 48 and 27 k-mers matches, respectively. Since CONSULT stops looking for matches as soon c k-mers match, fewer expected matches translate to longer running times. Moreover, the equation shown in Figure 2.1b is an over-estimate because not all k-mers in the reference library are in the memory. In preliminary experiments, we observed that $l = 1$ could reduce sensitivity (Table C.10). Thus, we set $l = 2$ by default.

We set $k = 32$ to reduce the chances of non-homologous k-mer matches. As Figure 2.1b shows, $h = 11$ and $h = 13$ lead to many matches at a high distance, which would increase the running time. We found that $h = 15$ balances memory usage, running time, and accuracy well (Table C.10). Given this choice, since our goal is to allow $g = 33$, we should ideally set $b = 2^{33-2h} = 8$, which unfortunately leads the library to be slightly larger than 128GB. Instead, we set $b = 7$, making our total memory usage close to 122GB for 8 billion k-mers (Table C.10). Note that indices of the encoding array become 33-bits, but using a simple trick (keeping two encoding arrays along with an indicator bit), we can keep them as words.

Library construction. To build CONSULT databases, we first find all canonical 35-mers from all genomes in the reference set using Jellyfish [139] and then minimize [196] them down to 32-mers; this step reduced the k-mer count to include in a reference library (Table 2.1). We skip the minimization step for small reference datasets with $< 10^{30}$ 32-mers. Since the Jellyfish output is pseudo-randomly ordered [137], further randomization is not needed.

2.2.2 Experimental validation

We test CONSULT in two applications: 1) as an exclusion-filtering method that seeks to find and remove contaminants among nuclear reads, and 2) as an inclusion-filtering method that

Table 2.1. *k*-mer counts (in billions) for reference datasets before and after minimization. Number of *k*-mers corresponds to CONSULT databases constructed with default settings of the tool. Number of 35-mers indicated for Kraken before minimization computed as a sum of unique canonical *k*-mers extracted for Bacterial (20 billion) and Archaeal (0.5 billion) portion of the database. Subsequently Bacterial and Archaeal Kraken *k*-mer lists were concatenated and minimized. Mitochondrial *k*-mer set included all canonical 32-bp *k*-mers that were extracted without minization. TOL [257] and GTDB [175] databases are from previous publications.

Dataset	**Species count**	**35-bp *k*-mer count(bln)**	**32-bp minimizer count(bln)**	***k*-mers included in database**
Bacteria/Archaea Kraken	159509	20.562188135	8.173628125	6.210280798
Tree of Life (TOL) [257]	10470	26.634996609	14.026601864	7.999975120
Genome Taxonomy Database (GTDB) [175]	31910	22.840757178	19.376574640	7.999986111
Mitochondrial RefSeq	11138	NA	0.201551248	0.194863421

seeks to detect mitochondrial reads in a genome skim to facilitate better mitochondrial assembly.

Exclusion Filtering of contaminants

Reference libraries. For software validation and contamination removal testing we constructed reference libraries from three available microbial genomic datasets: Tree of Life (TOL) [257], Genome Taxonomy Database (GTDB) R05-RS95 [175] and bacterial and archaeal species present in standard Kraken-II [250, 251] (Table 2.1). TOL was composed of 10,575 microbial species and a reference phylogeny. Five genomes had IDs that did not exist in NCBI and were excluded from this set. The remaining genomes were assigned to the reference set (10,460 genomes), the query set (100) or both (10). GTDB included 30,238 bacterial and 1,672 archaeal genomes, that were selected to represent 194,600 samples clustered at 95% nucleotide identity. The Kraken library consisted of 158,627 Bacterial and 882 Archaeal samples available in RefSeq (as of July 2019). Kraken reference sets were used without modification. When building the reference library for the TOL and GTDB libraries, Kraken-II removes genomes that can't be assigned a taxonomic ID using its automatic detection methods [250]. We have taken care to add these genomes to the library by assigning the to the root of the taxonomic tree (but also show

results without them).

Experiments. We performed three experiments to test exclusion filtering of contaminants.

(i) *Controlled distances.* Similar to [189], we first evaluated the ability of CONSULT to find a match when the query is within a range of phylogenetic distances to the closest species present in a database. To control the proximity of the query to its closest match in the reference library, we selected 100 genomes from TOL such that their distances to their closest species in the tree uniformly covered a broad range of [0.0-0.3). These queries were removed from the reference set and remaining TOL genomes were used to construct CONSULT database. We also randomly selected 10 genomes to keep in both the query set and reference set which allowed us to evaluate TP and FN. Subsequently, all queries were divided into bins based on their distances to the closest match in a reference database (Table C.11) and 10 plant genomes (Table C.12) were added to the set of queries in every bin. Plant species are from a different domain of life compared to the TOL reference set and should not match the library; thus, they allowed us to measure FP and TN. All distance values in this experiment were computed using Mash [167]. Reads for the TOL query set were simulated at 10MB using ART [89] (see Appendix D).

(ii) *Novel genomes.* We next assessed the ability of CONSULT to match genomic reads that belonged to novel microorganisms not observed in reference sets. To generate queries we used samples from Global Ocean Reference Genomes (GORG), a collection of 12,715 marine Bacterial and Archaeal single-cell assembled organisms [173]. Marine microbial species are known to be poorly represented in public repositories [227]. Since very few reads from these samples are expected to map to the reference genomes, they represent a particularly challenging classification case for databases with standard compositions and provide a suitable test case. To generate queries, we obtained GORG assemblies from NCBI (project PRJEB33281; five

assemblies were missing) and simulated query reads for every sample at 1x coverage using ART with the same settings as TOL (see Appendix D). We also included the same 10 plant species as TOL to compute the FP rates.

(*iii*) *Real skims.* We tested ability of CONSULT to remove contaminants from real genomic sequencing reads. For studying real genome skims, we obtained high-coverage raw SRA's of 14 Drosophila species (Table C.13) from NCBI (PRJNA427774) and subsampled them down to 200 MB using seqtk [122]. We removed adapters, deduplicated these samples and merged paired-end reads using BBTools [40] (see Appendix D). To filter out human reads, we queried Drosophila samples against the Kraken database that included only the human reference genome, and subsequently extracted unclassified reads to use in contamination removal experiment. Drosophila reference assemblies available from [155] were used to compute true distances.

Tools compared. We compared performance of CONSULT with Kraken-II [250, 251], CLARK [172], CLARK-S [171] and Bowtie2 [109, 111]. These are among leading identification tools based on recent benchmarking studies [146, 151, 211, 254]. Kraken-II is a taxonomic sequence classifier that maps k-mers of the query to the lowest common ancestor (LCA) of all genomes known to contain a given k-mer. We constructed Kraken reference libraries for genomes that belonged to TOL, GTDB, and Bacterial/Archaeal portion of the standard Kraken database. Kraken reference libraries were built without masking low-complexity sequences, but using default settings otherwise. We note that [189] found defaults were the most effective setting for contamination removal.

CLARK, CLARK-S, and Bowtie2 are tested only in the experiment (i), and thus, their reference databases were built using the TOL dataset. CLARK is a method that does supervised sequence classification based on discriminative k-mers. We constructed the CLARK database

using standard parameters (e.g., k=31 default classification mode). We set taxonomy rank to phylum (default is species) to achieve better sensitivity for contamination removal (see Appendix D for details). CLARK-S is a CLARK version that exploits multiple spaced k-mers and offers higher sensitivity at the expense of more RAM and slower classification speed. CLARK-S database was constructed on top of the custom CLARK database described above and querying was performed using default full mode of classification (see Appendix D). Bowtie2 is a standard general-purpose alignment tool. We built the Bowtie index reference for TOL genomes and performed local alignment of the queries using highest sensitivity setting (see Appendix D).

Evaluation. In experiments (i) and (ii) we report the recall and false positive rate: matched (unmatched) prokaryotic reads are TPs (FNs) and matched (unmatched) plant read are FPs (TNs). On the TOL dataset, we also compared running time and memory consumption of all tools for running a randomly sampled set of 30 small 10Mb queries from the TOL query set (Table C.15). To reduce the impacts of database loading on running time, we report results when the query is a single file concatenating 15 Drosophila skims sampled at 2G bp (Table C.16). In experiment (ii), we also report the percentage of reads from each microbial genome that match.

In experiment (iii), based on the results of the first two experiments, Drosophila genome skims were filtered against GTDB database, and Skmer was used to compute distances between all pairs of samples before and after filtering. Distance values obtained from Drosophila assemblies were considered the ground truth. We computed relative distance error for every sample before and after filtering in order to identify whether contamination removal improved distance estimates.

Inclusion-filtering of mitochondrial reads to help organelle assembly

We test whether CONSULT can help improve the quality of mitochondrial assembly by finding mitochondrial reads in a genome skim without a need for the standard seed-and-extend approach, the use of a very close reference genome, or reliance on coverage differences. To do

so, we constructed a CONSULT reference database out of all 11,138 mitochondrial genomes available in NCBI (RefSeq release 204), which included $\approx$ 200 million 32-mers (Table 2.1). We then asked whether CONSULT can use this broadly sampled database to identify mitochondrial reads in SRA files, including from species not present in the reference set.

We base our experiment on data from the DNAMark project that skimmed 210 vertebrate species (NCBI project PRJNA607895) and attempted to assemble their mitochondrial genomes [141]. We selected 42 (Table C.14) out of 210 samples as follows. We included all 18 samples where the original study failed to assemble mitochondrial genomes, all 6 samples that produced poor quality short contigs (3–10.5 kbp), and 18 randomly selected *good* samples with contig length $>$12 kbp, used as a positive control. Our SRAs include 24 species not present in the CONSULT reference dataset and 14 species not represented at the genus level (Table C.14).

We compare three assembly pipelines. *a*) We include assemblies generated using Novoplasty [57] made available by Margaryan *et al.* [141]. *b*) For each of the 42 samples, we preprocessed the raw SRA files by removing adapters using AdapterRemoval [209] and merging paired-end reads with BBTools [40] (see Appendix D). We assembled these *unfiltered* reads using plasmidSPAdes [8], which relies on read coverage to distinguish nuclear and organelle genomes. *c*) We first used CONSULT to search preprocessed reads against the reference mitochondrial database and then used only the matching reads as input to SPAdes with default settings [20] to obtain the assembly.

To assess the completeness of the assemblies, we first annotated all three assemblies (original, unfiltered, and filtered) using MITOS [26] to find the known mitochondrial genes. We report the total length of the largest mitochondrial contig, gene counts for different gene groups (protein coding genes (PCG), rRNA, tRNA), and identities of annotated genes for PCG and rRNA. The length of mitochondrial genomes should be approximately $\sim$16 kbp in size and the number of genes should be close to 37 [32].

Finally, note that we select one contig as the final mitochondrial assembly for each of the three methods. For original assemblies, only a single contig is available. For new assemblies,

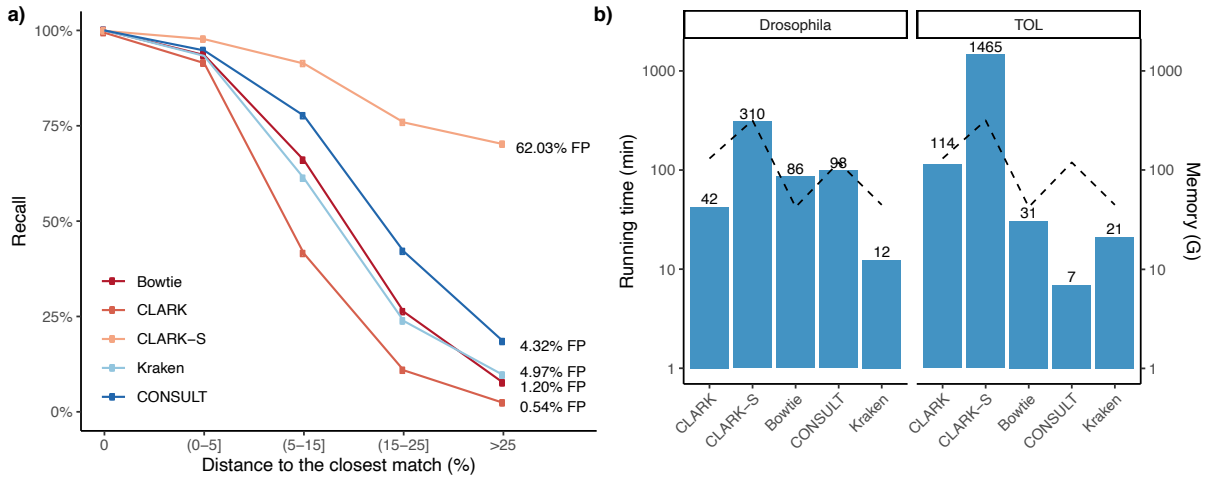


Figure 2.2. Results on the data simulated based on the TOL dataset. (a) Lines show recall for different query bins for five tools run in default settings (see Methods). Each line is labeled with its associated FPR, computed using plant genomes added to the query set, which are identical across bins; thus, FPR is identical in all bins. (b) Processing speed and memory consumption for different tools searched against the TOL library using the same settings as part (a). Drosophila benchmark set included a single 30G Drosophila query. TOL set was composed of 30 10MB microbial queries. Computation on a machine with Intel Xeon 2.2 GHz CPU using 24 threads and 350G of RAM.

we mostly take the the longest contig (as long as it is $\geq$200 bp) as the assembly. However, in assemblies produced from unfiltered reads, the largest contigs sometimes had nuclear origin. In such cases, we instead use the longest annotated contig with at least one annotated mitochondrial PCG or rRNA gene. The identity of mitochondrial contigs was additionally verified by MitoZ [148]. If no PCG or rRNA genes were assigned to any contigs in assembly, the generated reference is considered as having failed annotation.

2.3 Results

2.3.1 Exclusion filtering of contamination from nuclear reads

(i) **Controlled distances.** In the controlled distance experiment, CONSULT has the best recall among the methods that are able to control the FP rate (Fig. 2.2a). CLARK-S, which is specifically designed to match species absent from a reference database, has the highest sensitivity but FP rates close to 62%, making it ineffective for contamination removal. CLARK

has very low FP rates (0.5%) but also much lower recall than other methods. Overall, Bowtie has a similar recall to Kraken-II with a substantially lower FP rate (1.2% versus 4.9%). CONSULT is slightly better than Kraken-II in terms of FP (4.3%) but improves recall over Kraken-II and all other tools substantially. All the tools are able to match almost all prokaryotic reads to the database when the query has an exact match in the database, and all tools have at least 91% recall when the closest match in the reference library is up to 5% distant from the query. Substantial differences between methods appear when the closest match is $> 5\%$ distant to its closest match. For example, for queries at 5–15% distance to the reference set, CONSULT matches 78% of reads while Bowtie and Kraken-II match 66% and 61%, respectively.

The running time of CONSULT on the TOL DB is comparable to Bowtie but slower than CLARK and Kraken-II when tested on large query files (Fig. 2.2b). With multiple small query files, while CONSULT is the fastest, timing is hard to interpret because CONSULT analyzes all inputs in a run, amortizing the DB load time, while others need to be run per query file (a simple issue to fix). Bowtie and Kraken-II have the lowest memory footprint, followed by CONSULT, which uses 120GB.

(ii) **Novel genomes.** Next, we turn to the GORG dataset, where CONSULT matches a far larger number of microbial reads to the reference libraries compared to Kraken-II, regardless of the reference database (Fig. 2.3). Not only does CONSULT match more reads (has higher recall), it has fewer false positives, especially for GTDB (Fig. 2.3a). We thus tested both methods with several settings that had reduced false positive rates compared to their defaults, achieved for CONSULT by changing the (c, p) settings and for Kraken-II by increasing its α (i.e., percentage of k-mers in query sequence required for classification). For all levels of FP rate, CONSULT had better recall than Kraken-II for all databases tested (Figs. 2.3a). In default settings, CONSULT controls the FP rate at 7.0% on the large GTDB dataset, whereas Kraken-II has 13.5% FP. Recall that Kraken-II removed genomes without taxonomic assignment, and we added those back. If these genomes are not added to the library, the recall of Kraken degrades substantially but its

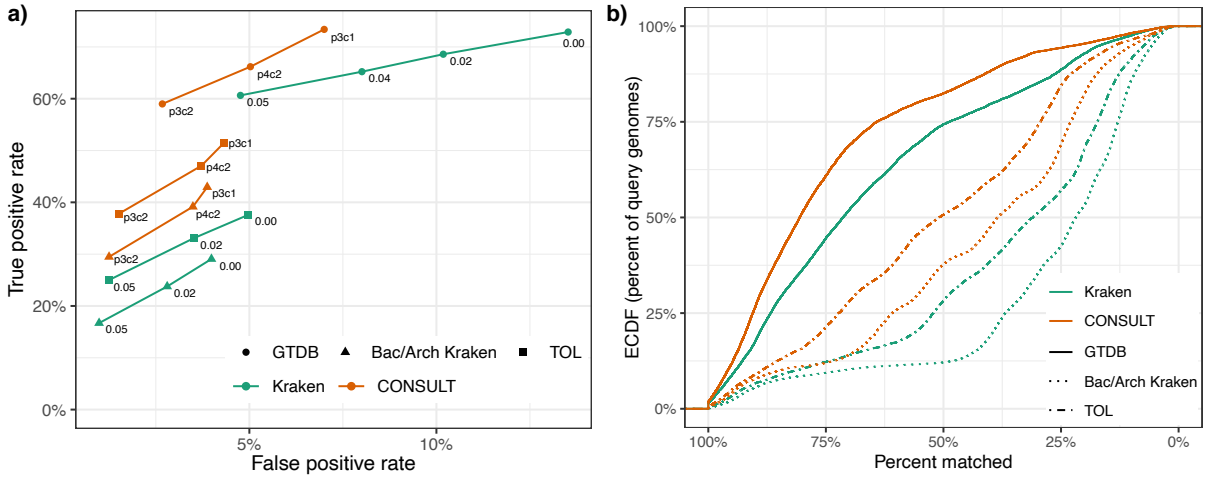


Figure 2.3. CONSULT vs Kraken-II on the GORG dataset. (a) The ROC curve showing the mean of recall vs FP rate (i.e., plant queries matched to a DB) with various settings for each method and searching against three libraries (GTDB, TOL, and the default Bac/Arch Kraken DB). Kraken-II was run with confidence level $\alpha \in \{0.00, 0.02, 0.05\}$. Additionally, $\alpha = 0.04$ was included on the GTDB database to better control FP rate. CONSULT libraries were searched using $p \in \{3, 4, 5, 6\}$ and $c \in \{1, 2, 3, 4\}$; see Figure B.19 for all combinations. (b) The empirical cumulative distribution of the percentage of reads in each microbial GORG genome matched to each of the three reference databases; a point $(x\%, y\%)$ means for $y\%$ of GORG genomes, $\geq x\%$ of the reads matched the DB. Here, we show default settings for CONSULT and Kraken-II on TOL and Bac/Arch databases, but $\alpha = 0.04$ for Kraken-II GTDB to control its FP rate.

precision improves (Fig. B.17). To enable a better comparison between Kraken-II and CONSULT, we chose the $\alpha = 0.04$ setting of Kraken-II that had 8% FP rate, which is only slightly higher than CONSULT. With these settings, CONSULT matched more reads than Kraken-II for 95% of the microbial species when searched against GTDB (Fig. B.18). CONSULT and Kraken-II match at least $^3/_4$ of reads for 61% and 44% of genomes, respectively. Comparing the three databases, GTDB results in the most matches for both methods, followed by TOL and Kraken.

Adjusting the (c, p) setting of CONSULT trades off recall and FP rate (Fig. B.19). For example, allowing up to 4 mismatches between k-mers in query and reference library produces more liberal ($c = 1$) or more conservative ($c = 2$) settings compared to the default where 3 mismatches are allowed. These combinations of parameters might be recommended for situations where a stricter FP control is required ($c = 2$) or when FP is less damaging ($c = 1$).

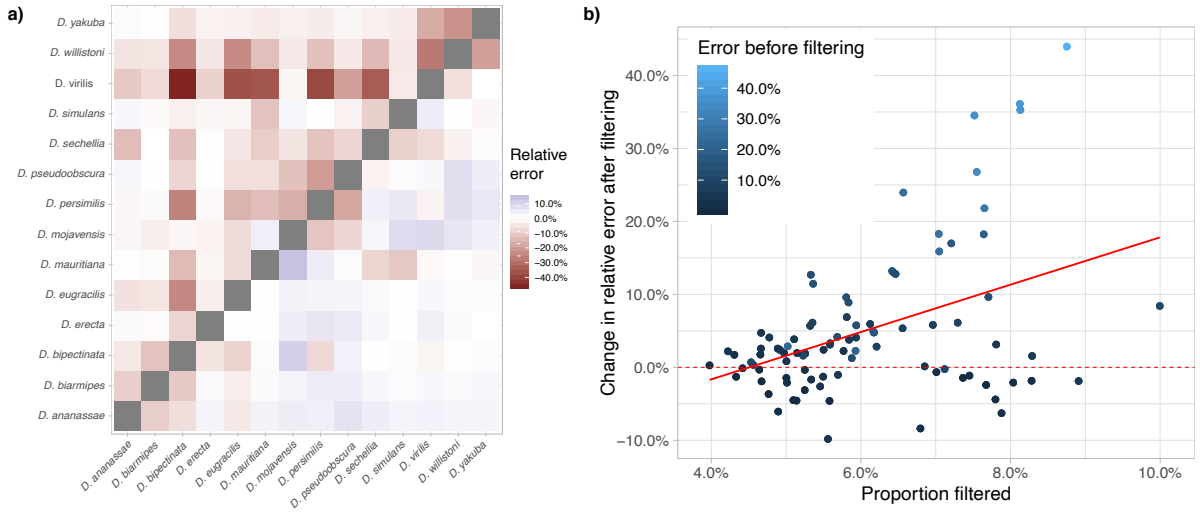


Figure 2.4. CONSULT on Drosophila skimming data. (a) Relative distance error before (upper triangle) and after (lower triangle) filtering per pair of Drosophila species. Numeric labels on y-axis represent percentage of bacterial/archaeal reads filtered per sample. (b) Change in the distance error after filtering compared to the error before filtering versus amount filtered (mean of both species); positive values indicate reduction in error. Each dot represents a pair of species.

All $p \geq 5$ and $c \leq 2$ lead to very high FP; e.g., $p = 6, c = 1$ leads to 100% recall but also 90% FP rate (Fig. B.19).

Kraken takes around 3 minutes to load 166GB GTDB database (on a machine with 350GB of RAM), which is substantially larger than the load time of 45G TOL database (half a minute). CONSULT loads databases once for all query genomes and the loading time doesn't exceed 3 minutes. Recall that CONSULT keeps the library size fixed at 128GB while Kraken-II keeps all k-mers; thus, when the number of species added to the database grows, CONSULT becomes more memory-efficient. Once the library is loaded, Kraken-II takes 0.18 seconds on average per query genome and CONSULT takes 0.09 seconds.

(*iii*) **Real skims.** Testing CONSULT on real genome skims from Drosophila demonstrates that Skmer distance calculation can improve dramatically as a result of filtering (Fig. 2.4a). Errors are reduced by as much as 44% between pairs of species (Fig. 2.4b). While distances tend to be underestimated before filtering, they tend to be slightly overestimated after filtering (Fig. 2.4a).

CONSULT removes between 3.9% and 10.2% of reads from these Drosophila genomes, and there is a positive correlation between the amount of data removed and the improvement in the estimated distances (Fig. 2.4b). In the most extreme case, distances are improved by more than 40% when less than 9% of reads are removed.

2.3.2 Inclusion-filtering of organelle reads

Using CONSULT to find organelle reads before assembly dramatically improves the quality of the assembly, both compared to the unfiltered approach that relies on coverage and seed-and-extend method used in the original study (Fig. 2.5).

When using raw reads we obtained complete or partial mitochondrial assemblies for 25 out of 39 assembled samples. Three samples didn't generate any contigs. Remaining 14 samples produced contigs of variable size but failed in annotation since estimated length of PCG and rRNA genes appeared shorter than expected. In contrast, when preceded by CONSULT filtering, reads were successfully assembled for all 42 samples, including the 18 samples that were left unassembled in the original study and 6 that had poor assemblies.

Assemblies produced by filtered reads were in all but one case either longer or comparable in size in comparison to assemblies generated by unfiltered reads (Fig. 2.5a). Similarly, they had a higher number of mitochondrial genes annotated in all but one case (Fig. 2.5b). The exception is the sample SRR12432391 that leads to a 35,920 bp contig when assembled from raw reads. This length is almost twice the average length of the mitochondrial genomes, which indicates a possible mis-assembly or chimeric contig. After filtering, 29 of the 42 samples had at least 27 out of 37 genes and 12 out of 15 non-tRNA genes annotated.

The completeness of an assembly after CONSULT filtering was not impacted by the presence of the corresponding species or genus in the RefSeq reference set (Fig. 2.5a, b). Cases where assemblies after filtering remain incomplete include both novel and observed samples. For example, of the 13 assemblies with less than 12 non-tRNA genes, four were represented exactly in the database, five had genomes from the same genus and four were not present at

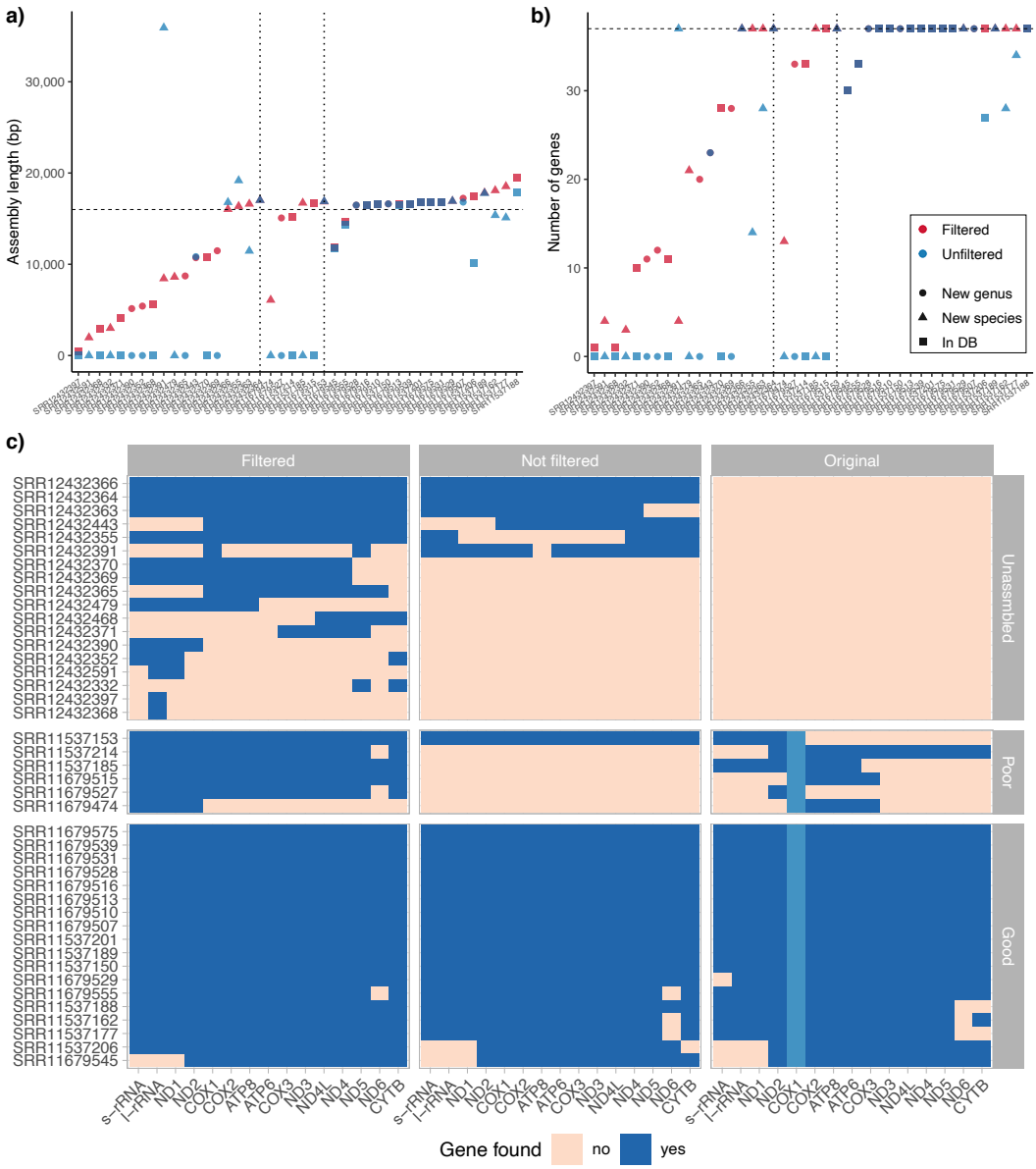


Figure 2.5. Mitochondrial assembly results. (a) Comparison of contig length between filtered and unfiltered samples. Dot shapes indicate whether any assembly from the same species or the same genus was a part of the RefSeq reference set that was used to construct CONSULT database. From left to right: sample that belonged to previously unassembled, poorly assembled, or good quality assemblies (boundaries indicated by vertical dots). The horizontal line is at expected lengths: 16 Kbp. (b) Total mitochondrial gene count for filtered and not filtered samples (counting unique PCG, rRNA and tRNA genes). Vertical line at expected number of genes: 37. (c) Mitochondrial genes identified in assemblies for filtered, unfiltered and original references. Light blue color highlights COI (COX1) gene that was used as a seed sequence for assemblies generated by the DNAMark project. In the unfiltered case, three samples (SRR12432370, SRR12432371 and SRR12432397) did not generate any contigs while 14 others generated contigs but did not have any genes annotated.

either level. Among the nine samples with no representation from the same genus, in five cases, CONSULT filtering improved gene recovery between 11 to 33 genes, made no difference in one case with an incomplete assembly, and recovered full assemblies in three remaining cases. Thus, effective filtering did not require representation from the same species or genus in the CONSULT reference library.

Comparison of gene annotations between newly generated and original assemblies (Fig. 2.5c) demonstrated that filtering enabled successful assembly for the most challenging low coverage samples. Thus, for samples that failed in the original study, we generated complete or nearly complete assemblies with up to 10 PCG identified and contig length $\geq$ 8719 bp for eight samples. Six samples produced partial assemblies and only four samples had contig length $\leq$ 3003 bp; even for them, we had some mitochondrial genes identified. For poorly preserved samples, we generated near-complete references for five out of six samples. In one case (SRR11679474), the main contig had only three PCGs and rRNAs genes, but even this sample contained all remaining genes scattered across five smaller contigs that the assembler did not merge with the longest contig (Fig. B.22). More generally, any gene found in the original assembly not found in the main contig of the filtered assembly was found in one of the smaller contigs (Fig. B.22). Unsurprisingly, the original seed-and-extend approach is biased toward the region including COX1, which is the seed, whereas filtered and unfiltered assemblies show no such bias. For the set of good quality samples, filtering improved gene recovery in three cases compared to unfiltered ones and five samples compared to the original assemblies; only one gene was recovered by one of original assemblies but not the filtered ones.

Additionally, since filtering reduced the number of sequencing reads that are being assembled, we observed $\approx 7\times$ running time improvement with filtered versus unfiltered reads (estimate includes CONSULT time), going from $\approx$ 65 min to $\approx$ 9.7 min per sample. This speed-up was calculated for 24 SRR that belonged to poor and good assembly groups (120G of RAM, 24 threads).

2.4 Discussion

We introduced CONSULT, a general purpose *k*-mer based read matching tool that might help in a variety of applications where there is a need to separate sequencing reads of interest from extraneous reads outside of the group. By careful engineering of the software, we have made it possible to run CONSULT on large reference datasets (e.g., tens of thousands of prokaryotic species) and large numbers of queries (e.g., hundreds of millions; see Table C.18). Our results on contamination removal showed that when the closest species in the reference set was substantially distant (≈15–20%) from the query, CONSULT improved upon existing methods such as Kraken, CLARK(-S), and Bowtie2 both in terms of sensitivity and specificity.

The use of LSH paired with hamming distance or other distances is not novel to CONSULT. Several methods in various domains of sequence analysis have used LSH [24, 38, 39, 130, 150, 190, 191]. These earlier uses all boil down to finding the nearest neighbors of a sequence *without* actually computing distances but accepting that some matches will not be within the desired radius. CONSULT uses LSH in a different way. Unlike earlier methods, CONSULT actually computes distances from each *k*-mer to a fixed-size number of potential matches. In doing so, it takes advantage of the large memory available on modern machines, which were traditionally not available; nowadays, we can easily afford to keep billions of 32-mers in memory. Thus, we use LSH only to find a small set of *k*-mers for which we compute distances exactly. As such, CONSULT does not have false positives (in the sense that it guarantees every match is below the desired threshold). It only can have false negatives. However, missing some *k*-mer matches is tolerable because if a read truly belongs, its other *k*-mers can match. Consistent with this observation, our data show that even when we include half or one-third of *k*-mers from a reference dataset in the memory (e.g., for the GTDB database; Table 2.1), the accuracy remains high.

Beyond algorithmic design, our application is quite different from these earlier adoptions of LSH. We use LSH to test whether reads belongs to a large taxonomic group allowing

substantial number of mismatches to the nearest neighbour. The past work has used LSH for guiding assembly [24], near-duplicate sequence removal [150], phylogenetic placement [38], homology detection between two genomes [39], and sequence clustering for OTU binning [190, 191] and metagenomic binning [130]. Many of these existing applications deal with far fewer and less diverse sets of sequences. In contrast, our methodology works on databases that span across entire domains of life and contain many billions of k-mers. Among methods designed for our application, Kraken adopts an approach that can be considered LSH due to its masking step.

CONSULT is more effective than existing methods such as Kraken and Bowtie when the queries are phylogenetically distant from their closest match in the reference dataset. While one may hope that denser reference sets will diminish the need for such distant matching, our results on the GORG dataset demonstrate that state-of-the-art microbial datasets are far from capturing the diversity of life with the $\approx 8\%$ distant where existing methods are accurate. Note that every genome in GORG is a single-cell assembled bacterial genome sampled randomly from the ocean; thus, these data are not exotic species put together just to challenge methods. Our results indicate that detecting even the domain of a read will require allowing many mismatches for the foreseeable future.

While we tested contamination in the context of genome skimming, we note that contamination in sequencing reads is a pervasive problem that can impact other analyses as well [71, 128, 223]. It can lead to inaccurate characterization of gene content and metabolic functions [35, 107], improper inference of phylogenetic relationships [113, 215], and biases in genotype calling and population genomics [19, 246]. Contamination is also known to infiltrate reference genomes stored in public databases [149] and is particularly problematic when endogenous DNA is scarce [75, 131, 194, 200]. Thus, CONSULT may find applications outside the settings tested here.

Our results showed that inclusion filtering of mitochondrial reads using CONSULT enabled generating complete and accurate assemblies for very poorly preserved samples where read coverage is not sufficient to use other methods. Our workflow is an example of what has

been called a "hybrid assembly method" for taking advantage of references [239]. By searching reads against all available organelle genomes and allowing mismatches, it eliminates the bias associated with template based assembly using a single reference; at the same time, it permits flexibility of *de novo* assembly. Using CONSULT for this application is reference agnostic and thus can be utilized on mislabelled samples or samples of unknown identity. Importantly, our data clearly show that there is no need to have the same species or even any representative from the same genus in the reference set for the filtering to work successfully. These strong results lead one to ask whether the assembly of the more complex plastid genomes is similarly improved by pre-filtering.

While we leave a full exploration of plastome applications to the future work, our preliminary results are encouraging (Table C.17). We built a reference database from 6537 plastid genomes available from NCBI (RefSeq release 206) and reanalyzed 60 samples obtained from a recent chloroplast assembly benchmark study [72]. These results suggest that filtering reads with CONSULT before assembly is as effective for chloroplast as it was for mitochondria (Table C.17). Using GetOrganelle [97] as the assembler, we produced complete or nearly complete chloroplast assemblies for eight samples that failed to be assembled fully without filtering (similar to the original study [72], an assembly with a contig of length at least 130 kbp was considered successful). Annotation of these assemblies showed that these complete assemblies capture many more of the expected plastid genes than the assemblies from unfiltered reads (Fig. B.20). Overall, contig length of assemblies produced from CONSULT-filtered reads was either comparable or longer (Table C.17) in comparison to unfiltered ones (increase in total genome length: 29% for successfully assembled samples). In a handful of cases, assemblies from unfiltered reads were substantially longer than those from filtered sequences. However, gene annotation using GeSeq [235] identified very few chloroplast genes in the long unfiltered assemblies, indicating that they were most likely spurious (Fig. B.21).

In all applications we explored, sequences from very diverse groups were included in the reference library. As a result, these reference sets included hundreds of millions to tens of

billions of 32-mers, of which, up to 8 billion were kept in the final library (Table 2.1). While the results clearly show that not every k-mer in the reference set has to be in the library to achieve high accuracy, there must be limits to the amount of subsampling tolerated. For example, if we consider a large and diverse set of vertebrate genomes, the number of k-mers may grow by orders of magnitude. Accommodating much larger databases will either require machines with larger memory, or more smart techniques for deciding which k-mers make it into the library.

Finally, at its core, CONSULT is simply a read matching method. Thus, while we focused on contamination detection and organelle read detection, our algorithm can also be adopted for other applications such as metagenomic profiling, OTU picking, and any question where inexact read matching is needed. Moreover, here, we performed contamination filtering using an exclusion-filter; however, a tantalizing opportunity that CONSULT may enable by allowing distant matches is inclusion filtering: find reads that seem to belong to the group of interest if assembled genomes from that phylogenetic group are available. Our results on organelle genomes, which used CONSULT as an inclusion filter, support the viability of this approach. Applying inclusion filters to nuclear genomes will have to contend with contamination in the reference assemblies, perhaps using further algorithmic innovations. We leave the exploration of such applications to future work.

2.5 Availability of data and materials

CONSULT is implemented in `C++11` with some x86 assembly code; it is (trivially) parallelized using OpenMP [169] to read the library and perform the search. The software is available publicly at https://github.com/noraracht/CONSULT. Scripts and summary data tables are publicly available on https://github.com/noraracht/lsh_scripts. Raw data used in the manuscript is deposited in https://github.com/noraracht/lsh_raw_data. The detailed description of genomic datasets used in our experiments, accession numbers of the assemblies and the exact commands used to simulate genome skims and analyze data are provided in Supplemental Material.

Chapter 2, in full, is a reprint of the material as it appears in CONSULT: accurate contamination removal using locality-sensitive hashing. Rachtman, E.; Bafna, V., & Mirarab, S., NAR Genomics and Bioinformatics, 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Quantifying the uncertainty of assembly-free genome-wide distance estimates and phylogenetic relationships using subsampling

Computing distance between two genomes without alignments or even access to assemblies has many downstream analyses. However, alignment-free methods, including in the fast-growing field of genome skimming, are hampered by a significant methodological gap. While accurate methods (many k-mer-based) for assembly-free distance calculation exist, measuring the uncertainty of estimated distances has not been sufficiently studied. In this paper, we show that bootstrapping, the standard non-parametric method of measuring estimator uncertainty, is not accurate for k-mer-based methods that rely on k-mer frequency profiles. Instead, we propose using subsampling (with no replacement) in combination with a correction step to reduce the variance of the inferred distribution. We show that the distribution of distances using our procedure matches the true uncertainty of the estimator. The resulting phylogenetic support values effectively differentiate between correct and incorrect branches and identify controversial branches that change across alignment-free and alignment-based phylogenies reported in the literature.

3.1 Introduction

Phylogenetic and population genetic analyses using assembly-free and alignment-free methods have enjoyed renewed interest in recent years. Alignment-based methods using orthologous loci have traditionally been considered more accurate than alignment-free methods for phylogenetics [29, 85], and this perception has not changed (justifiably, in our judgment). However, the main driver of recent interest in assembly-free and alignment-free methods is the increased use of data types that do not avail themselves to assembly, in particular, the study of biodiversity at high precision [30]. Biologists now routinely use low-coverage shotgun sequencing data, called genome skims [45, 244], obtained across a large number of samples to study biodiversity and ecology at high levels of taxonomic resolution accurately and relatively cheaply. A simple search of the term "genome skimming" reveals that more than 950 papers have been written on the subject since 2020. While the traditional use of skims relied on the assembly of organelle genomes (a tiny fraction of the reads), there has been an increasing recognition that better accuracy can be obtained using assembly-free and alignment-free fashion by analysis of nuclear data [204]. Assembly has to be avoided because of the low coverage (e.g., 1x), and alignment is impossible when genome skims are compared to a reference library of other genome skims (and not closely related genomes).

Alignment-free phylogenetics has a long history of method development [29, 79, 101, 117, 118, 253] and benchmarking [85, 258]. Many of these methods require the assembly of sequences instead of working with bags of reads. However, assembly-free methods have been developed for inferring phylogenies [2, 64, 255] and for estimating genomic distances between two bags of reads [e.g., 112, 167, 204, 231], which can then be used with standard distance-based phylogenetic estimation methods to infer a tree or to place a sample on an existing phylogeny [16, 18]. In particular, some of these methods, such as Skmer [204] and Afann [231], are specifically designed for datasets with shallow sequencing depth.

A major obstacle to the widespread adoption of assembly-free methods for downstream

analyses such as phylogenetics is the lack of reliable ways to measure the statistical support of inferred trees. The method of [247] automatically assigns a weight to each split it infers; however, these weights are not interpretable as the probability of correctness and can only be used for a specific distance estimation method. Lack of support makes it hard to interpret phylogenetic trees as the (inevitable) differences between inferred trees cannot be put in context without knowing if those differences have high support. Moreover, many downstream applications attempt to sum over tree uncertainty [86], or at least, contract low support branches of the tree into polytomies [216, 256]. Finally, in some applications (such as interrogating polytomies, within-species diversification, and delimitation of subspecies), whether one can resolve a relationship or not *is* the question of the interest [133, 236], and that question cannot be answered without statistical support.

While a robust set of tools for support estimation exist for alignment-based distance calculation and phylogenetics, much less is known about statistical support for assembly-free methods. Outside Bayesian methods, bootstrapping (i.e., *re*sampling – sampling with replacement) of alignment sites is the main method used for uncertainty quantification, including in phylogenetics, where it has been long used [66] and debated [e.g., 67, 83, 199, 228]. While many biologists interpret support as the probability of correctness, bootstrap is more precisely interpreted as a way to measure the variance around an estimator. Regardless of the precise meaning, in the alignment-free settings, it is unclear what needs to be resampled. We can resample reads or smaller units such as k-mers or spaced words used by most distance estimators. However, assumptions of independently and identically distributed (i.i.d) data units, which are somewhat reasonable for sites in an alignment, may not apply to such units of data. Moreover, assembly-free methods often make assumptions about a random sampling of reads across the genome, and resampling can invalidate those assumptions.

The non-parametric statistics literature has established that *sub*sampling can provide a powerful alternative to *re*sampling, with fewer strong assumptions and better generalizations [183]. Because a subsample of a shotgun sequenced set of reads follows the same distribution as

the original sample, albeit with lower coverage, it provides an attractive alternative to bootstrapping. The lower coverage samples obtained with subsampling will clearly lead to more variable distances than the original data. This bias, however, can be corrected using methods known in the non-parametric statistics literature [183].

In this paper, we introduce and carefully validate methods for estimating uncertainty around distances computed directly from genomes and genome skims using assembly/alignment-free methods. We propose using *sub*sampling combined with a correction for increased variance instead of *re*sampling. This procedure computes a distribution of distances for each pair of genomes or genome skims, as opposed to a single distance. From there, computing branch support for phylogenetic trees using distance-based methods follows the standard approach. To be more precise about the input and output, let us define them more formally.

Given are genome skims (i.e., two bags of reads), generated at low coverage (e.g., 2X), from two genomes. The two genomes have evolved from a shared common ancestor through an evolutionary process parameterized by the true genomic distance d. The simplest such model is applying substitutions according to the [100] (JC) model along two branches of total length $\frac{4}{3}\log(1-\frac{4}{3}d)$. The two genomes are then subsampled at random positions using short reads with sequencing errors. Thus, starting from a fixed common ancestor, there are two sources of randomness: the evolutionary process (e.g., substitutions) and the genome sequencing procedure (e.g., random sampling of reads and sequencing errors). The input can also be two assemblies, where only the first source of randomness exists.

We have access to an estimator $\hat{d}$ of d given two genomes or genome skims. If the estimator were to be applied to (unavailable) data generated by a procedure identical to what generated our data (e.g., in a simulation), it would be draws from some unknown distribution $\mathscr{D}$. Our goal is to estimate the uncertainty of the estimate $\hat{d}$ by generating m estimates $\hat{d}_1 \ldots \hat{d}_m$ drawn (approximately) from $\mathscr{D}$. Thus, while we have one pair of genome skims, we seek to approximate the distribution of estimates had we access to an infinite number of genome skim pairs generated identically from the same common ancestor.

For the estimator of d, in this paper, we focus on a leading assembly-free distance calculation method, Skmer, noting that our procedure is general and is applicable to any estimator. Skmer has two components: estimating sequencing parameters for each input skim and estimating shared k-mers between two given skims (see STAR Methods). The sequencing parameters are estimated by matching the k-mer frequencies against the Poisson distribution distorted by sequencing errors (see (3.3) in STAR Methods). Then the portion of shared k-mers, the Jaccard index, is computed using the min-hash technique of Mash [167] ($k = 31$ in both components). These results are then combined to a final estimate using the analytical equation (3.4) given in STAR Methods.

In simulations and on a set of real datasets, we show that our subsampling procedure paired with Skmer produces reliable support values from both genome and genome skims. We evaluate the method under conditions with model misspecification and show that while support is not always fully calibrated, it is predictive in distinguishing correct and incorrect branches.

3.2 Results

3.2.1 Subsampling Procedure

Subsampling: justification and theory

Why not bootstrapping? While the bootstrapping method of [63] provides a statistically consistent approximation of $\mathscr{D}$, some of its assumptions, such as independence of data points, are violated in our setting. Moreover, bootstrapping breaks the assumptions of the estimator. The most poignant problem is non-random coverage. Many assembly-free methods, including Skmer, assume that reads are distributed randomly throughout the genome and thus, model the number of times each position is sampled using a Poisson distribution. Once observed reads or k-mers are resampled *again*, this Poisson assumption is broken (Fig. 3.1A). [64] attempted to extend the bootstrapping procedure to account for some of the dependencies in assembly-free settings by using block bootstrapping. However, their method uses resampling and is not appropriate for

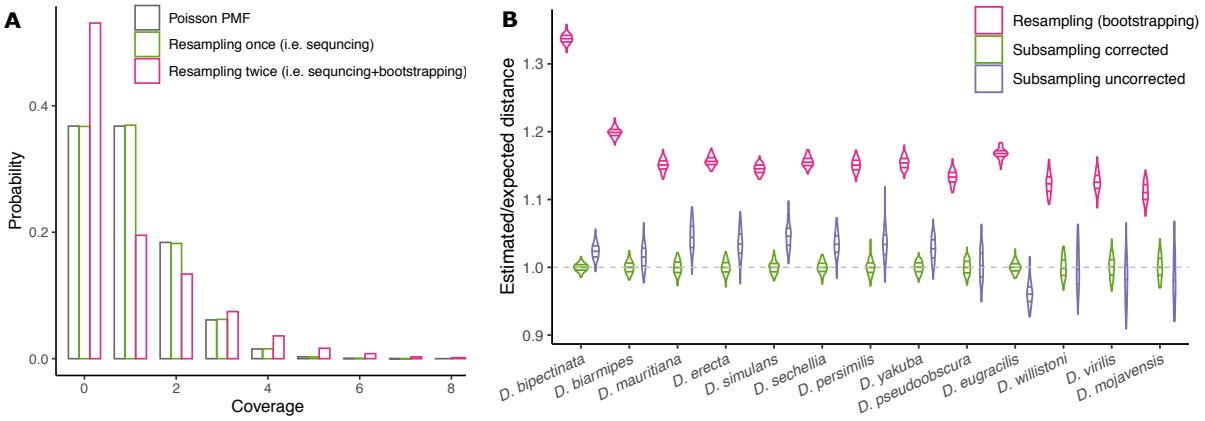


Figure 3.1. Problems with resampling. A) Theoretical model. Consider a set S of $N = 10^5$ objects (e.g., k-mers of a genome). We show the empirical distribution of the number of times an object is observed (i.e., k-mer frequencies) if S is sampled uniformly at random N times with replacement to get R; this process is similar to sequencing a genome at 1X coverage. The distribution closely matches Poisson with $\lambda = 1$. Next, we again resample R (with replacement) N times, which is equivalent to the bootstrapping reads or k-mers, getting a distribution that strongly deviates from Poisson (e.g., too many 0 counts, too few 1 counts). PMF denotes the probability mass function. B) Comparisons on a 200MB Drosophila skimming dataset between subsampling $n^{9/10}$ reads or bootstrapping n reads where $n \approx 10^6$ is the number of reads. In both cases, 100 replicates were generated, and distances were computed using Skmer. Graph shows all pairwise distance values between *Drosophila ananassae* and other Drosophila species. *Subsampling corrected* distances are using our method, as discussed in the text.

estimators that rely on random coverage of the genome. In empirical analyses, we can establish that bootstrapping leads to widely inaccurate and biased estimates (Fig. 3.1B).

Why Subsampling? An alternative to bootstrapping is subsampling without replacement: Given a set of data points, subsample them at random, apply the estimator to the subsample, and repeat to get a distribution of estimates. As detailed by [183], subsampling is a sound way of measuring estimator uncertainty and crucially depends on fewer assumptions than bootstrapping while often providing comparable power. A subsample of the reads (or k-mers) is equivalent to another genome skim with lower coverage and does not violate the assumptions of Skmer, designed explicitly for low coverages. Thus, subsampling will generate unbiased estimates (Fig. 3.1B). However, despite being unbiased, subsampling leads to incorrect variance. Thus, the distribution

of estimates from subsamples needs to be corrected so that it asymptotically matches the correct distribution, $\mathscr{D}$. To see the need for this correction, note that the variance of the estimator increases as the amount of data decreases. Thus, the subsampled estimates will have a higher variance than $\mathscr{D}$, and the overestimation of variance will depend on the size of the subsamples. If not appropriately corrected, the arbitrary size of subsamples will determine the variance, which is clearly undesirable.

We base our method on what we call subsampling theorem for short [Theorem 2.2.1 in 183]. Let $\hat{\theta}_n$ be a statistically consistent estimate of a parameter θ based on n data points (no assumption of independence is necessary). Assume that there exist some sequence of numbers τ such that $\tau_n(\hat{\theta}_n - \theta)$ weakly converges to *some* distribution as $n \to \infty$ [assumption 2.2.1 of 183]. The τ_n factor can be informally considered the rate of convergence of the estimator. Similarly, let $\hat{\theta}_b$ be the estimate from a subsample of b data points sampled from the original n. To paraphrase informally, according to the subsampling theorem, under very general regularity conditions, the empirical cumulative distribution function of many $\tau_b(\hat{\theta}_b - \theta_n)$ estimates converges to cumulative distribution function (CDF) of $\tau_n(\hat{\theta}_n - \theta)$ as $n \to \infty$ assuming that $b \to \infty$ and $b/n \to 0$. In other words, to approximate the distribution of (unobtainable) estimates centered by the true value, we can examine the observable distribution of subsampled estimates centered around the main estimate, scaled by τ_b/τ_n.

Subsampling Procedure

We use reads (for a genome skim) or k-mers (for an assembly) as the atomic data units that are subsampled, and let n denote the number of such units. To use the sub-sampling theorem, we need the appropriate choice of τ_n. By the central limit theorem (CLT), given i.i.d random samples drawn from a population with mean μ and variance σ^2, the limit $\lim_{n\to\infty} \sqrt{n}\left((\bar{X}_n - \mu)/\sigma\right)$ converges to a standard normal distribution where $\bar{X}_n$ is the sample mean. Therefore, for any estimator $\hat{\theta}$ that can be written as the mean of some random variables, $\sqrt{n}(\hat{\theta} - \theta)$ converges to a Gaussian distribution making $\tau_n = \sqrt{n}$ the correct normalizing factor.

Algorithm 2. Subsampling procedure. Input: a set of skims $\{S_1 \dots S_N\}$ with $n_1 \dots n_N$ reads, a fixed parameter $\alpha < 1$, and a tool (e.g., Skmer) $f(\{S_1 \dots S_N\})$ to compute all pairwise distances.

$\hat{d}_{ij} \leftarrow f(\{S_1 \dots S_N\})$ ▷ Run a method like Skmer to compute all pairwise distance
for $r \in 1 \dots m$ **do**
  for $i \in 1 \dots N$ **do**
    $b_i \leftarrow (n_i)^\alpha$
    $S_i^r \leftarrow$ a random subsample of size b_i of S_i.
  $d_{ij}^r \leftarrow f(\{S_1^r \dots S_N^r\})$ ▷ Compute distances from subsamples
$y_{ij}^r = \sqrt{\frac{b_i+b_j}{n_i+n_j}}(d_{ij}^r - \overline{d_{ij}}) + \hat{d}_{ij}$ ▷ First correction: using main estimate
$x_{ij}^r = \sqrt{\frac{b_i+b_j}{n_i+n_j}}(d_{ij}^r - \overline{d_{ij}}) + \overline{d_{ij}}$ ▷ Second correction: using mean estimate
for $r \in 1 \dots m$ **do**
  $t_c^r, t_m^r \leftarrow$ Distance-based trees inferred using x_{ij}^r and y_{ij}^r, respectively ▷ Using tools such as FastME
$t_c \leftarrow$ Extended majority consensus of $t_c^1 \dots t_c^m$
$t_m \leftarrow$ A distance-based tree inferred using $\hat{d}$ distances
Assign to each branch e of t_m and t_c support $\frac{1}{m}\sum_1^m [e \in t_m^i]$ and $\frac{1}{m}\sum_1^m [e \in t_c^i]$, respectively

While the coverage computation step of Skmer uses a complex estimator that cannot easily be described as a mean of random variables, the Jaccard calculation *can* be approximately described as such. For each k-mer, consider a binary random variable X_i indicating whether it is shared between the two genomes, and note $\Pr[X_i = 1] = J$. Every k-mer can be considered a random sample from this distribution. Then, the Jaccard computed from L k-mers is $J = \sum_i X_i/L$. Thus, ignoring dependence of k-mers, our Jaccard estimates do follow the CLT and admits the $\tau = \sqrt{n}$ correction. We will use $\tau = \sqrt{n}$, admittedly ignoring the first part of the Skmer procedure in deriving this correction factor (more on this later).

We propose the following procedure (Fig. 3.2A). Given is a set of N genome skims S_i each with n reads (see Algorithm 2 for a relaxation where each sample has a different number of reads). We choose a constant $\alpha < 1$ (default $\alpha = 9/10$) to set $b = n^\alpha$ noting $b \to \infty$ and $b/n \to 0$ as $n \to \infty$. We perform m (user-provided) rounds of subsampling. In each round r, we subsample b reads uniformly at random for each skim i and compute distances between these subsampled skims, giving us an estimate d_{ij}^r for each pair i, j of skims. These distances need to be next corrected and used for estimating a tree per replicate. For getting the final tree, we can use either

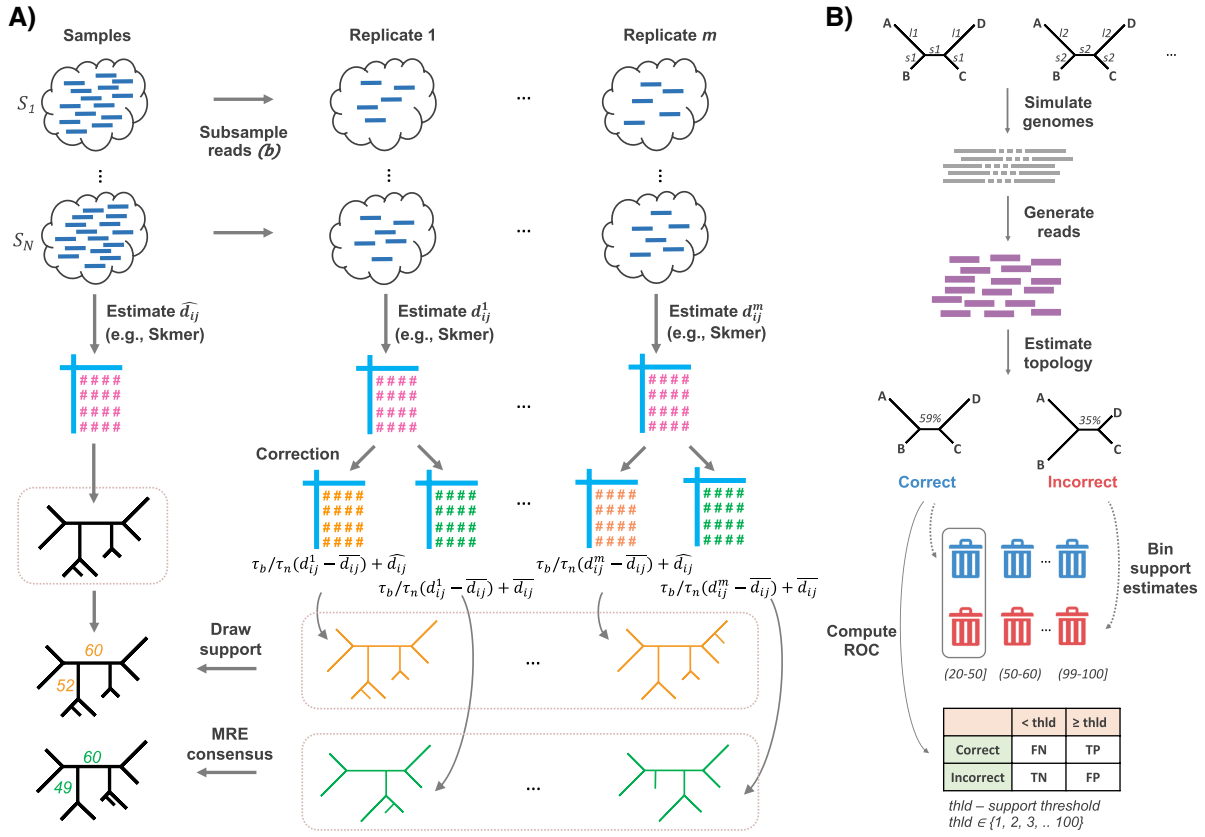


Figure 3.2. Workflow diagrams. A) Subsampling workflow. Every sample with n reads is subsampled m times to generate replicate data with $b = n^\alpha$ reads. Obtained pairwise distances d_{ij}^r are corrected with τ_b/τ_n centered by either mean ($\overline{d_{ij}}$) or main estimates ($\hat{d}_{ij}$). B) Workflow of Felsenstein zone simulations.

the *main* estimates with no sub-sampling, $\hat{d}_{ij}$, or the related quantity $\overline{d_{ij}}$, defined as the average distance (*mean* estimate) across all m sub-sample replicates. From the subsampling theorem, we can infer that $\sqrt{n^\alpha/n}(d_{ij}^r - \hat{d}_{ij})$, $\sqrt{n^\alpha/n}(d_{ij}^r - \overline{d_{ij}})$, $(\hat{d}_{ij} - d_{ij})$, and $(\overline{d_{ij}} - d_{ij})$ all asymptotically converge to the same distribution. Accordingly, we consider two expressions for correcting distance. First, when the final tree is inferred from main estimates, we center all the subsampled distances around zero, apply the correction, and then center them back around the main estimate:

$$y_{ij}^r = \sqrt{\frac{n^\alpha}{n}}(d_{ij}^r - \overline{d_{ij}}) + \hat{d}_{ij} \tag{3.1}$$

The alternative is to use the extended majority rule (i.e., greedy) consensus of the m replicate

trees as the final tree. Since this final tree does not refer to the main distances, we have no reason to use $\hat{d}_{ij}$ in the correction and instead use:

$$x_{ij}^{r} = \sqrt{\frac{n^{\alpha}}{n}}(d_{ij}^{r} - \overline{d_{ij}}) + \overline{d_{ij}} \tag{3.2}$$

Finally, when given two assemblies, we apply the subsampling procedure at the k-mer level. Because Skmer computes Jaccard using the min-hash technique (see STAR Methods), which boils down to subsampling k-mers to a sketch size K, we simply repeat the sketching process m times, with sketch sizes smaller than the original (i.e., K^{α}), to get a distribution of distances.

3.2.2 Simulation results

We start by benchmarking our method on several simulated datasets (see STAR Methods for details).

Distance variance when simulating genome pairs

We first test whether subsampling accurately captures variance in distance estimates. Recall that the definition of correct support is to match the distance distribution $\mathscr{D}$ when sequence evolution is repeated. In simulations, we *can* estimate $\mathscr{D}$ by repeating the simulation procedure. Thus, we compare the distribution from subsampling a single run with the distribution obtained across simulation replicates.

The subsampling procedure, with the square root correction, manages to obtain distance distributions close to the ideal distribution $\mathscr{D}$ obtained from simulations (Fig. 3.3A). Note that the change in the center of the distribution is expected, as the subsampled distribution is obtained for one simulation replicate and is centered around its main estimate, $\hat{d}_{ij}$. Notably, the variance of the subsampled procedure closely matches the simulated distribution $\mathscr{D}$ as long as the true distance is within the 0.02–0.15 range (Fig. 3.3B). For small distances ($d = 0.01$), subsampling noticeably overestimates the variance, and conversely, for large distances, it slightly underestimates variance.

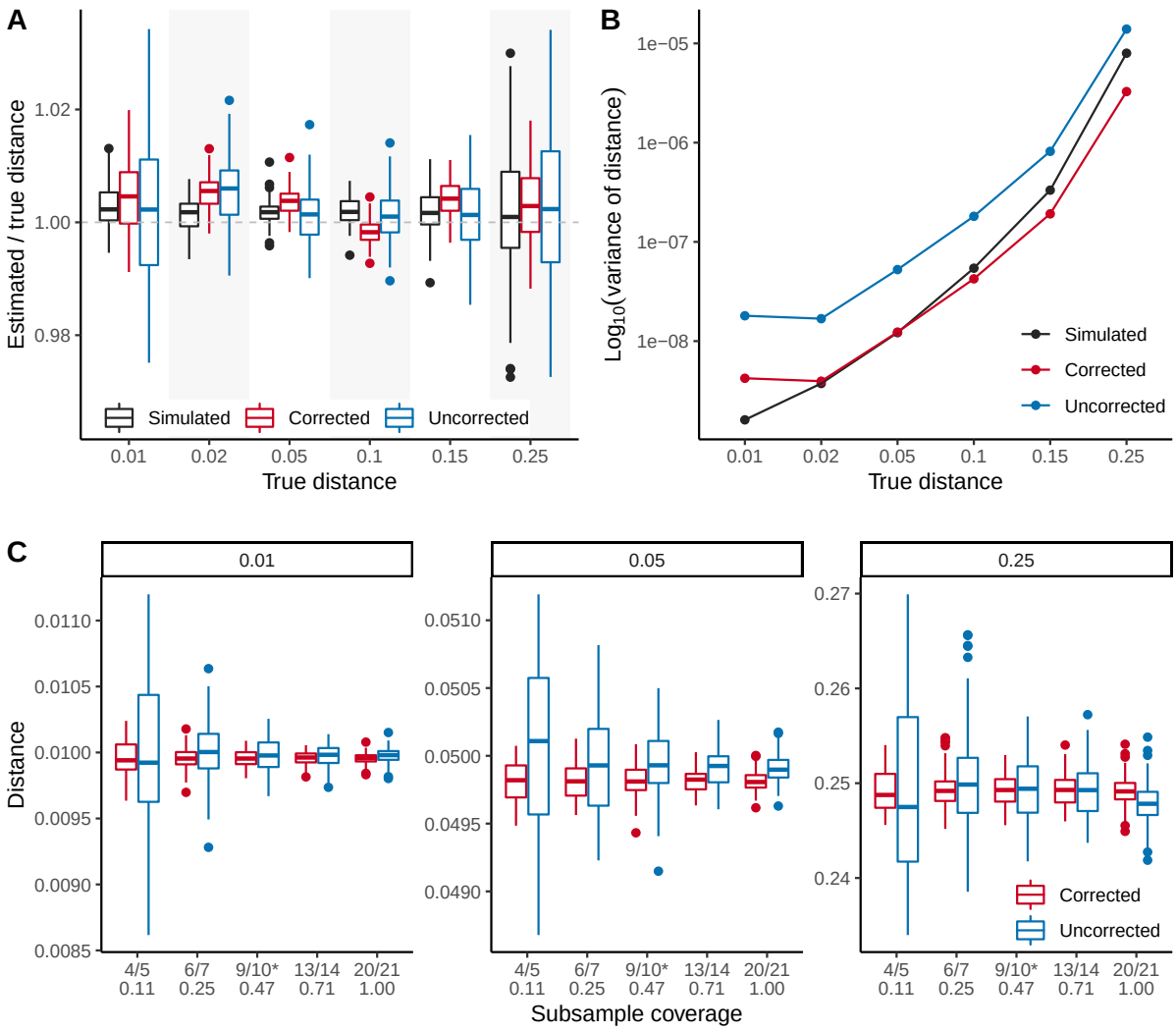


Figure 3.3. Genome pair analyses. A) Distance distributions scaled by true distance obtained across 100 simulations or 100 subsampling rounds in one simulated replicate, with and without correction using equ. (3.1). B) Variance in distance estimates in simulations (e.g., the ideal distribution $\mathscr{D}$) versus subsampled data. C) Impact of distance correction versus α. The x-axis label denotes α on top and the corresponding coverage on the bottom. The original sample had 2x coverage. $*$ shows the default α. Values on top panels correspond to the true distance between pairs of samples in the simulated tree.

Importantly, the correction is essential for getting reasonable results; when distance values were left uncorrected (i.e., eliminating $\sqrt{b/n}$), the variance produced by subsampling was dramatically larger than $\mathscr{D}$ (Fig. 3.3B).

The distribution of distances should ideally not depend on α (thus, b), and we observe

that correction dramatically reduces the dependence of distributions on α (Fig. 3.3C). For typical distances (e.g., 0.05), the variance of the distances shrinks very slightly as α increases. The (slightly underestimated) variance of large distances (0.25) does not seem to depend on α. Only in the case of a small distance ($d = 0.01$), where variance is slightly estimated with default α, do we observe a noticeable impact as the number of subsampled reads varies. Results indicate that for small distances, while the $\sqrt{b/n}$ correction is effective in dramatically decreasing the gap between ideal and observed variance, it is not exact. When we start with 16x coverage, similar patterns are observed; however, the variance does substantially reduce when subsampled coverage goes above 4x (Fig. B.23), a point we further discuss later. Overall, the corrections seem effective, if imperfect, in eliminating the effect of subsampling on the variance of the estimator.

Simulation under Felsenstein-zone quartet trees with long branch attraction

We next evaluate the calibration of branch support values drawn on final trees obtained using either consensus (3.1) or the main (3.2) estimates on a challenging dataset that simulates conditions prone to long branch attraction using the Felsenstein-zone quartet trees (see STAR Methods). We say that supports are fully calibrated if they perfectly correspond to the probability of correctness. Support values produced by both correction methods tend to be underestimated (Fig. 3.4A). For example, branches with 70-80% support are correct 86% of the times using the main estimates and 100% using consensus. While support underestimation is more severe for consensus, we observe an unusually high percentage of correct trees with support $< 50\%$ with the main estimate. Overall, these support values are conservative.

Beyond calibration, we interrogate the predictive power of support values (see STAR Methods) and observe that support values are predictive of accuracy (Fig. 3.4BC). There is a large gap between the support distribution of correct and incorrect branches, and the gap is more prominent for consensus than the main-estimate approach (Fig. 3.4B). Moreover, with the consensus method, increased support perfectly correlates with increased accuracy (Fig. 3.4A) so

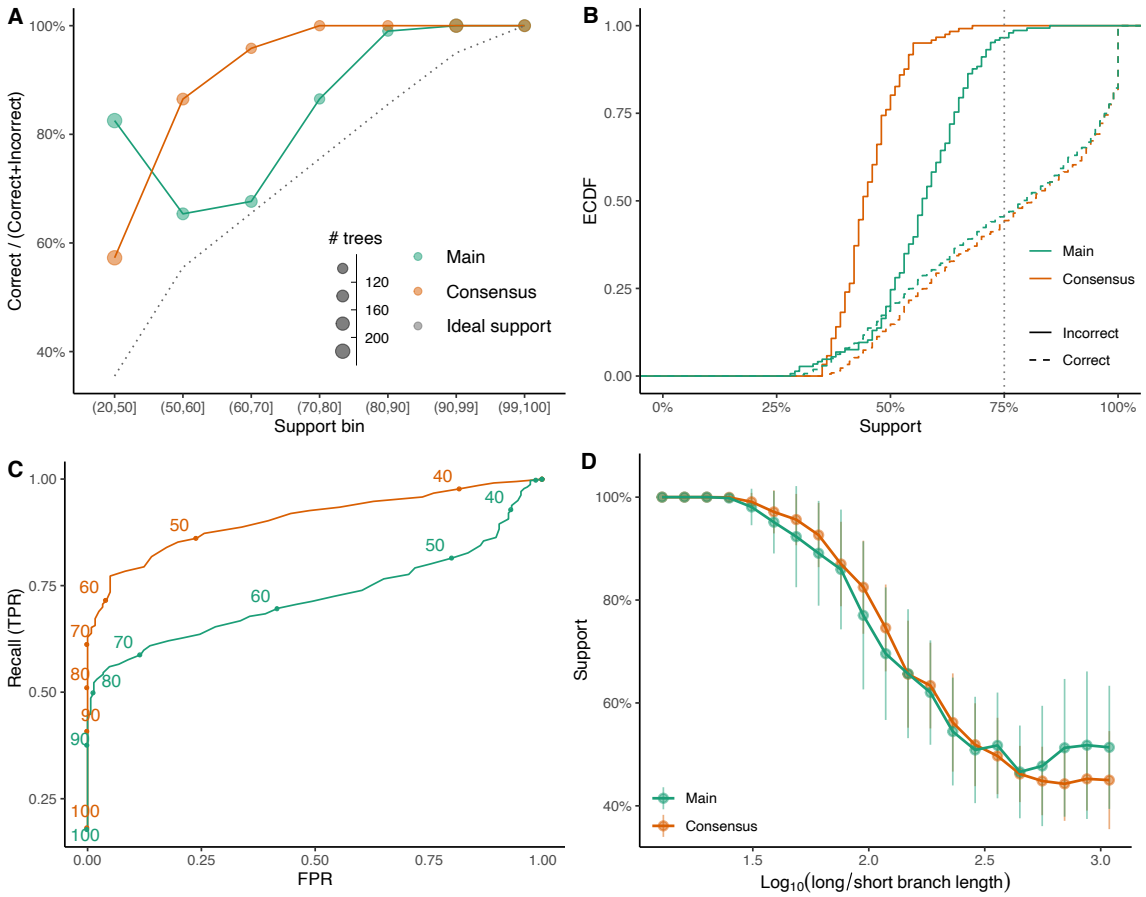


Figure 3.4. Support accuracy on simulated Felsenstein-zone phylogenies. Results are across 1000 replicates. A) Percentage of correctly inferred topologies (y-axis) for each of seven bins of branch support (x-axis). The percentage of correct branches in each bin should ideally match the median of that bin (dotted line). B) Empirical cumulative distribution function (ECDF) of support values divided between correct and incorrect branches. C) Receiver operating characteristic (ROC) curve built by considering branches with support below thresholds 0-100%, with 1% increments (see Fig. 3.2B for the definition of the confusion matrix). Selected thresholds are shown on the graph. D) Effect of branch length on estimated support.

that all incorrect topologies have support below 75%, and more than 50% of correct branches have support above 75%. Constructed ROC curves (obtained from confusion matrices built using thresholds of support; see Fig. 3.2B) show that very low false positive rates and high recall can be obtained by examining branches with $> 70\%$ support (Fig. 3.4C). Moreover, consensus is substantially more powerful in discerning correct and incorrect branches than the main estimate. Finally, note that, as expected, the value of support depends on the ratio between long and short branches (Fig. 3.4D). Support is invariably low when long branches are two to three orders of magnitude bigger than small branches.

Simulations on 8-taxon trees with model misspecification

We next assess the branch support on a simulated 8-taxon dataset meant to challenge our approach by simulating genomes that deviate from the assumptions of the models used for inference. In these simulations (see STAR Methods), model misspecification can make the Skmer estimator biased; thus, the errors in these trees are not just due to variance (of the estimator) but also bias, and there is no theory to suggest that the subsampling procedure can overcome bias. Nevertheless, we evaluate the impact of bias empirically.

With model misspecification, higher support $>50\%$ tends to be overestimated while support values $\leq 50\%$ are underestimated (Fig. 3.5A). Despite the tendency of the method to *overestimate* support for much of its range, the highest support values are reasonably reliable. In particular, among branches with 100% support, 95% are correct. Thus, unlike the previous simulations with an unbiased estimator, when the estimator is biased, a small but considerable portion of branches with 100% support are incorrect.

There appears to be a weak dependency between rate variation and the accuracy of the tree and its support value (Fig. B.24A). The portion of incorrect branches with 100% support are noticeably higher for trees with the highest rate variation parameters for the main estimates (though not for consensus). Also, all trees with three or more incorrect branches out of five were among replicates with higher rate variation parameters (Fig. B.24B). However, beyond these

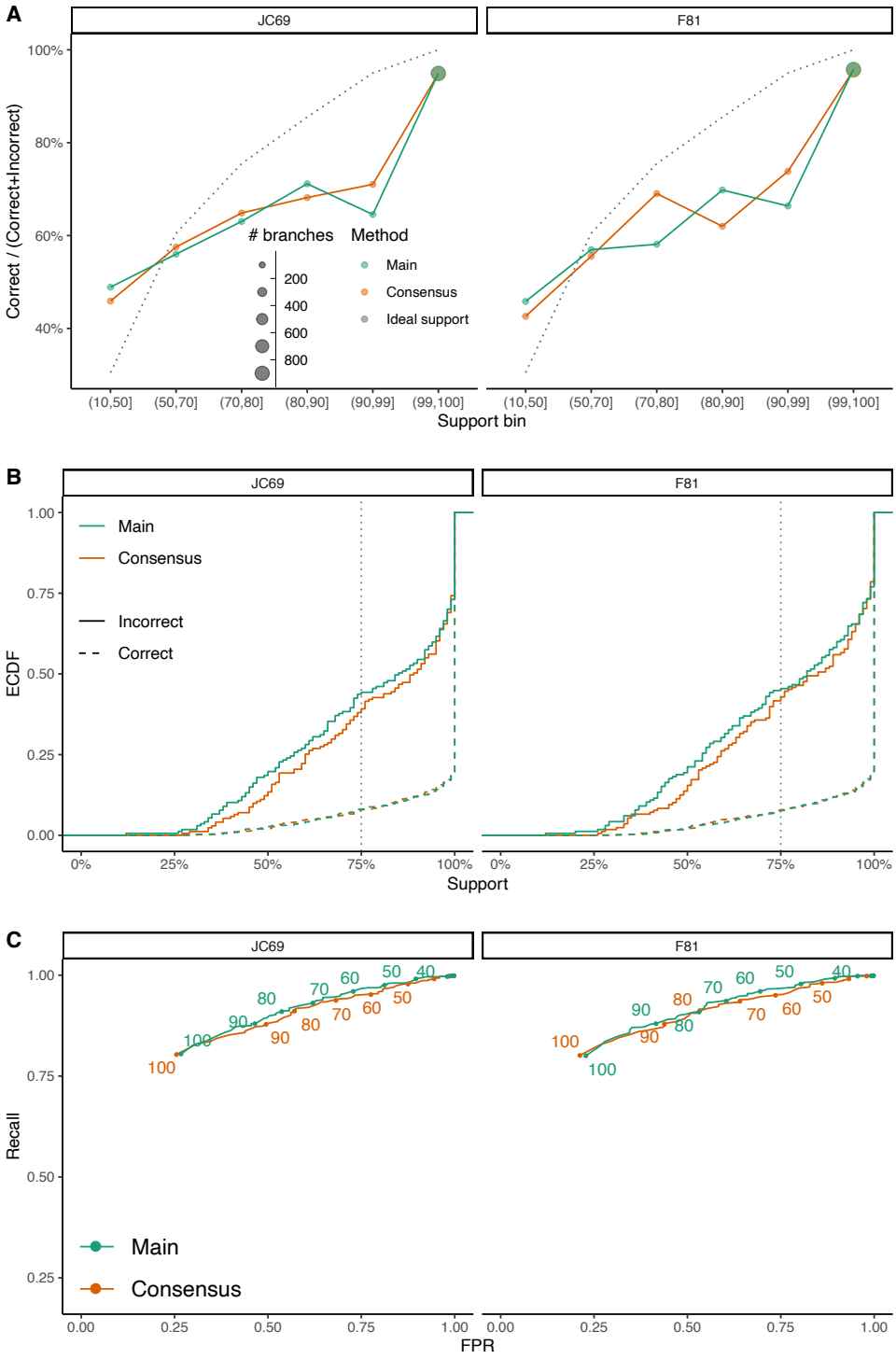


Figure 3.5. Support accuracy on the eight-taxon dataset. Results are across 120 replicates under Jukes and Cantor 1969 (JC69) and Felsenstein's 1981 (F81) models, and figures are identically set up to Figure 3.4.

extreme cases, no strong correlation between rate variation and accuracy was observed.

Despite being imperfect, the support values do have the predictive power to distinguish branch correctness (Fig. 3.5BC). We observe a large separation between the distributions of support values of correct and incorrect branches (Fig. 3.5B). As expected, incorrect branches have a wide range and seem uniformly distributed, except for an overabundance of roughly a quarter of incorrect branches that have support close to 100%. Thus, in contrast to previous simulations, the introduction of bias makes false positive rate (FPR) values below 0.25 unavailable; however, the recall does not seem to have been negatively impacted (compare Figs. 3.4C and 3.5C). All three ways of evaluation show similar trends regardless of the type of correction used, with the main-estimate correction performing slightly better.

When separating our replicates into those based on fully balanced and fully unbalanced trees (see STAR Methods), we observe only minor differences in overall support patterns (Fig. B.24C). Balanced trees tend to have somewhat higher FPR at 100% support but also higher recall, indicating that support overestimation is slightly more severe for balanced trees. The same pattern emerges when we examine branch lengths. Just as in previous simulations, the lower estimated support values tend to be mostly among short branches; however, balanced trees have higher support than caterpillar trees for branches of the same length (Fig. B.24D).

3.2.3 Results on Real datasets

Cetaceans: low resolution within species. We use two cetacean datasets (see STAR Methods) to contrast support for relationships within species (presumably uncertain) versus across species (presumably, highly confident). We see vastly different support patterns for trees inferred within or across species (Figs. 3.6AB, B.25B). The tree inferred for samples of the *Mesoplodon grayi* species has low support assigned to the majority of its branches, while the tree inferred across species has full support for every branch. This result is in line with expectations since the signal of tree-like evolution between individuals of the same species should be weak (or non-existent) and the signal across species should be strong given the use of assembled genomes.

Thus, our support values show that most of the inferred relationships between individuals of the *Mesoplodon grayi* are biologically unreliable, and these individuals should be considered part of a freely mixing population. This result is in line with the conclusions of [245] which used a host of analyses that revealed little to no population structure. The assembly-free tree inferred across cetacean species (obtained in less than an hour, including the time to download the genomes) is highly congruent with the alignment-based trees inferred using a much more complex pipeline by [144]; thus, the full support across all branches is reasonable.

Low support for conflicting results. We next use three insect datasets (lice, bee, and drosophila; see STAR Methods) to investigate whether the lack of support corresponds to the difficulty of resolving branches. On all three datasets, we consistently observe that branches with low support in Skmer-based trees are those that conflict with alignment-based trees published in the literature.

On the lice dataset, the phylogeny inferred from genome skims using the main correction has five branches with support below 100% (Fig. B.25C), and these are the only branches that differ from the ASTRAL [156] tree reported by [33] (after removing five outgroup species with no skims). Using consensus-based trees also produces support below 100% for all branches disagreeing with the ASTRAL tree but has two extra branches with low support (Fig. B.25D).

On the bee dataset, where we have both assemblies and simulated genome skims, we obtain identical tree topologies from both data types (Fig. 3.6CD). As one would expect, support values are higher for assembly-based trees (all branches 100%, except for three, which are above 92% when computed using main-estimate) than genome skims (four branches below 90% with the main estimate). Interestingly, all branches that deviated from the alignment-based tree reported by [226] have support below 100% in both of our trees. One of the two conflicting branches, the placement of *B. breviceps* is the only branch without full bootstrap support in the Sun *et al.* study, the only branch where ASTRAL and concatenation disagreed, and also, one of the only two branches without full support in the ASTRAL tree by [226]. These results are robust if we keep the number of replicates between 50 to 1000 (Fig. B.26AB).

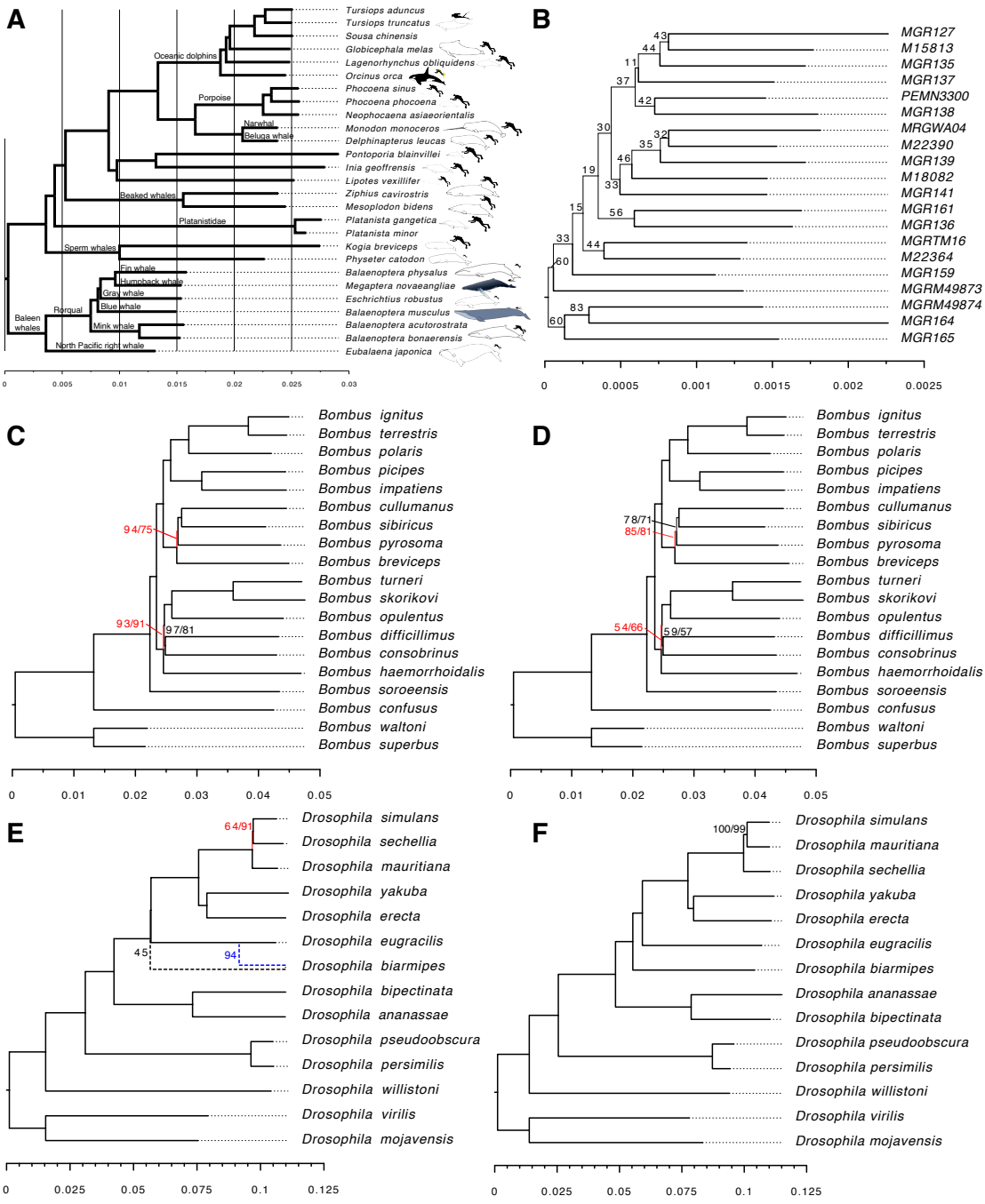


Figure 3.6. Phylogenies constructed using biological datasets. Branches with no values assigned signify 100% support. The left value corresponds to support for tree corrected with the main estimate, and the right value denotes consensus. The thickness of the branch corresponds to the magnitude of associated support. Red denotes controversial branches as compared to reference phylogenies. Blue denotes alternative topology between main and consensus trees. A) Whale phylogeny inferred from assemblies. B) Whale tree computed for multiple organisms of the same species of Gray's beaked whale *Mesoplodon grayi*. C) Bee phylogeny built from assemblies. D) Bee phylogeny obtained from simulated at 1x sequencing reads. E) Drosophila phylogeny built from assemblies. F) Drosophila phylogeny obtained from short-read SRAs.

On Drosophila dataset, we also have both trees based on assembly and genome skims (Fig. 3.6EF). The tree generated by genome skims aligned perfectly with the alignment-based tree reported by [155], and all of its branches had 99% or 100% support with both ways of computing support. The tree based on assemblies and using the main correction, however, had two branches with low support (64% and 45%), and one of the two branches (uniting *D. sechellia* and *D. simulans*) was different from the [155] tree. Interestingly, the Skmer-based resolution matches what [238] had proposed earlier based on 13 marker genes but better taxon sampling. The second difference is the position of *D. biarmipes*, which matches between Miller and our tree, but also differs from the Vanderlinde tree. Interestingly, the consensus correction produces the Vanderlinde resolution using our data, albeit with only 94% support. In summary, both branches without full support in consensus-based and main-based trees show conflicting results within the literature.

Impact of coverage. Next, to test how the coverage level of a genome skim impacts its power to resolve phylogenetic relationships, we reused the bee dataset but with varying coverage levels (Fig. B.26CD). As expected, with higher coverage of the genome skim (i.e, before subsampling), reported support values increase. However, even with coverage as high as 8x, the branch that showed signs of conflict in the literature (i.e., placement of *B. breviceps*) did not reach full support. The results indicate that the lack of support for this branch was not simply due to lack of coverage.

Support versus length. Examining the branch length versus support across all datasets with genome skims, we observe an interesting pattern (Fig. B.25A). While shorter branches, especially those smaller than 0.001, are frequently associated with lower support, the association is not perfect. For example, among branches with length in the 0.0005 – 0.001 range, support ranges between 50% and 100%. Thus, while short branches are clearly more uncertain than longer ones, not all short branches are equally uncertain.

3.3 Discussion

We have shown that subsampling, unlike resampling, represents a reliable approach to derive statistical support for distances and phylogenies estimated using assembly/alignment-free methods. On real data, we observed that lower support values were associated with controversial branches. On the simulated datasets, we saw that while branch support is not always calibrated with the probability of correctness (tends to be underestimated when there is no model violation and overestimated when there is), it is powerful (i.e., distinguishes correct and incorrect branches). Note that a method of support estimation can be very powerful and yet not be calibrated (e.g., consider a case where the estimated support is always half the probability of correctness). The lack of calibration is not unique to our subsampling procedure. Bootstrapping has long been believed to underestimate support [83], leading many biologists to use 75% support as a threshold of high support. It appears that our subsampling method has similar tendencies to underestimate support, at least in the absence of model violations.

We observed that the variance of subsampled distances is slightly over and underestimated for very small and long distances, respectively. This apparent bias is likely related to the rate of convergence formula we used; i.e., $\tau_n = \sqrt{n}$. Recall that our choice of τ had a justification only for the Jaccard estimation step and not the coverage estimation step. We can empirically assert that our choice was an appropriate choice for most distance ranges but not all (Fig. 3.3B). It is possible that the rate of convergence of Skmer is higher than $\sqrt{n}$ for small distances and slightly lower for large distances. While a more accurate choice of τ_n requires further theoretical work, we note that the current choice produces reasonable results.

Just like any other method of support estimation, our method has theoretical guarantees only for statistically unbiased estimators. It has been long appreciated in studying evolution that systematic bias leads to support overestimation, as manifested by the confident incongruence between competing phylogenies [91, 181, 182, 202] or increased support of genome-wide data compared to single genes [233]. Our method is not an exception. When we simulated data

with model misspecification, we also observed some strongly supported incorrect branches. In particular, about 5% of branches with 100% seemed to be incorrect and 25% of incorrect branches had 100% support. Thus, the interpretation of biological results should keep the possibility of model violation in mind. Note that the bias is coming from the estimator (here, Skmer+JC model), not the subsampling procedure; thus, the only principled solution is to design improved estimators under more complex models. Such estimators have been suggested in the past [64, 180] and are being actively developed [12, 47]. However, it should be noted that more complex models do not always increase accuracy. As a proof of concept, we reanalyzed three datasets (8-taxon simulations, Drosophila, and Bee real datasets) under the F81 model of evolution, which accounts for variable base frequencies. This more complex model did not substantially change our results in any of the datasets (Figs. 3.5, B.27A-D).

A more subtle form of model violation is the presence of repeats in the genome, which the Skmer approach mostly ignores. To test the impact of repeated regions on reported support estimates, we removed repeats from Drosophila and Bee assemblies using RepeatMasker [218] and recomputed phylogenies and support values. The set of uncertain relationships (i.e., those with less than 100% support) do not change after repeat filtering (Fig. B.27EF). For Bee genomes (Fig. B.27E), where repeat content did not exceed 2.88% (Table C.19), computed tree topology remained unchanged and was identical for main and consensus approaches. However, the support values for one branch did change from high (94%) to low (66%). For Drosophila (Fig. B.27F), identified repeat content was higher and more varied, ranging between 2.8–10.3% (Table C.20). On these data, both types of correction result in the same tree topologies, which was not the case before repeat masking. The support for the only branch that changed after masking went from low (45%) with one resolution to moderate (70%) with another resolution. Thus, our findings suggest that the repeat content may affect phylogeny estimation and support. Luckily, these changes seem to be mostly around branches with low support, as identified by our measure. Thus, our support values give users ways to identify unreliable branches.

Like bootstrapping, subsampling requires running the procedure many times and can

increase the running time. Using a machine with AMD EPYC 7742 2.25 GHz CPU using 24 threads and 120G of RAM, our running times were manageable ($m = 100$ in our case). For example, for the lice dataset with 61 samples, each 400MB, the entire process took 7 hours. Drosophila and bee datasets with fewer species and smaller sketches each took around 49 minutes. Thus, subsampling, while not fast, requires reasonable time. We also did not detect any benefit from further increasing m and hence the running time (Fig. B.26AB). We note that our tool allows embarrassing parallelization through `-i` and `-b` options, which we did not use here.

The subsampling procedure is integrated within the Skmer software and produces both main and consensus estimates. The simulations showed the consensus to be more accurate only when the model was not violated. Since the main estimates have been found to be more accurate elsewhere in phylogenetics [157] and here on data with model misspecification, they should still be considered a better choice. On our real datasets, in all cases, except the beaked whale genome dataset with no resolution, the two methods produced similar trees. In practice, we advise the users to try both methods and consider any branch with low support in either analysis as suspicious.

Our study has several limitations that future work should address. Most of our tests were on genome skims with low coverage, which is the most challenging case. However, testing the simpler case of genomes with low coverage is also important. We note that Skmer changes how it computes coverage when the coverage seems to be above 4x. This change in the algorithm makes the use of subsampling tricky, as the subsampled data may use a somewhat different estimator than the main. This observation may explain the pattern observed for 16x coverage analyses (Fig. B.23). Note that the use of the consensus method can ameliorate this problem as it ensures all estimates are using the same method. Thus, we suggest using the consensus method and selecting α carefully for datasets with high coverage to ensure that the (estimated) coverage is either consistently above 4x or always below it.

We conclude with a note about the generalizability of the method. The resampling method tested here can be used without much change for other distance-based assembly-free methods.

Beyond assembly-free methods, any distance-based method, including alignment-based methods, can also take advantage of this approach. In fact, deep learning methods of distance estimation have already started to adopt our proposed approach [92]. For example, alignment-based methods could compute distances based on subsets of sites and correct distances using the equations we noted. We leave it to future work to compare the accuracy of bootstrapping and subsampling in such conditions where both methods are applicable.

3.4 STAR Methods

3.4.1 Key Resources Table

Table 3.1. Key Resources Table.

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Biological samples		
Whale sequencing reads	245	Table C.22
Bee sequencing reads	226	Table C.23
Drosophila sequencing reads	155	Table C.13
Lice sequencing reads	33	Table C.24
Software and algorithms		
Skmer code	This paper	https://github.com/shahab-sarmashghi/Skmer
ART	89	https://www.niehs.nih.gov/research/resources/software/biostatistics/art/
Seqtk	N/A	https://github.com/lh3/seqtk
Kraken2	250	https://github.com/DerrickWood/kraken2
Bowtie2	111	https://sourceforge.net/projects/bowtie-bio/files/bowtie2/
CONSULT	187	https://github.com/noraracht/CONSULT
BBTools	40	https://sourceforge.net/projects/bbmap/files/
INDELible	70	http://abacus.gene.ucl.ac.uk/software/indelible/
FastME	116	http://www.atgc-montpellier.fr/fastme/
RAxML	222	https://github.com/stamatak/standard-RAxML
RepeatMasker	N/A	http://www.repeatmasker.org/
Other		
Data and summary of analyses	This paper	https://github.com/noraracht/subsample_support_scripts; https://doi.org/10.5281/zenodo.6473473

3.4.2 Resource availability

Data and code availability

- This paper analyzes existing, publicly available data. All original studies are referenced in the main text. Accession numbers for the datasets are available in this paper's supplemental information.

- The software is available publicly at https://github.com/shahab-sarmashghi/Skmer. Raw data and summary of results are deposited in https://github.com/noraracht/subsample_support_scripts. The DOI is https://doi.org/10.5281/zenodo.6473473.

- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

3.4.3 Method details

Skmer

Because we focus on Skmer as the estimator, we review the essential aspects of this estimator. Skmer has two stages and a final calculation.

Stage 1: Skmer uses k-mer frequency profiles (computed using JellyFish [140]) to estimate the amount of sequencing error and the coverage (neither of which is known). Let M_i be the number of k-mers observed i times in the genome-skim and assign $h = \text{argmax}_{i \geq 2} M_i$. Also, let $\xi = \frac{M_{h+1}}{M_h}(h+1)$. Skmer estimates:

$$\begin{aligned} \lambda &= \frac{M_1}{M_h}\frac{\xi^h}{h!}e^{-\xi} + \xi(1-e^{-\xi}) \\ \varepsilon &= 1-(\xi/\lambda)^{1/k} \end{aligned} \tag{3.3}$$

as the k-mer coverage and sequencing error rate, respectively. Then, for $i \in \{1,2\}$, let $\eta_i = 1-e^{-\lambda_i(1-\varepsilon_i)^k}$ and $\zeta_i = \eta_i + \lambda_i(1-(1-\varepsilon_i)^k)$ (for high coverage, it defines ζ_i and η_i differently; see the original paper). Also, let L_i be the estimated genome length (total sequencing amount divided by coverage). By default, Skmer sets $k = 31$, which we keep. [14] have shown that a reduced k can improve accuracy for assemblies, but we used a fixed value since measuring inaccuracies, not increasing the accuracy, is our focus.

Stage 2: Skmer uses Mash to compute the Jaccard index J between two skims.

Final calculation: Skmer computes the genomic distance between two genome skims

with Jaccard similarity J and genomic parameters estimated above using

$$\hat{d} = 1 - \left(\frac{2(\zeta_1 L_1 + \zeta_2 L_2)J}{\eta_1 \eta_2 (L_1 + L_2)(1+J)} \right)^{1/k}. \tag{3.4}$$

Skmer can also handle input assemblies by dividing them into k-mers, and omitting the correction for low coverage and sequencing error by setting $\zeta_i = 2$ and $\eta_i = 1$.

Once the hamming distance is estimated, the phylogenetic analyses use the corrected distance obtained using the standard [100] correction: $t = \frac{4}{3}\log(1 - \frac{4}{3}\hat{d})$. We use t for phylogenetic inference.

Subsampling

Unless otherwise specified, we used Algorithm 2 with $\alpha = 9/10$ and $m = 100$, estimated distances using Skmer (default settings, but increasing sketch size to 10^6 for simulated data), corrected distances using the JC model, and used FastMe [116] with default settings to compute trees. We used RAxML [222] to draw support and compute extended majority rule consensus trees (Appendix D). Note that this approach is counting bipartitions in a binary fashion and can be sensitive to rouge taxa; alternatives exist and can be adopted in the future [119].

3.4.4 Datasets

We evaluated the performance on three simulated and several real biological data.

Genome pairs. We used genome pairs to test the validity of distance distributions. We simulated 100Mbp genome pairs at phylogenetic distance $t \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.25\}$, repeating the procedure 100 times (so 700 genome pairs), and simulated at 2x and 16x coverage genome skims (Appendix D). Note that these distances are in the unit of the expected number of substitutions per site. Next, for each distance t, we arbitrary selected the 7th sample and used the subsampling procedure to generate a distance distribution, which we then compared against $\mathscr{D}$ obtained across 100 simulation runs. Note that the choice of α should ideally not change the distance distribution (if the correction factor $\sqrt{b/n}$ is correct). To empirically

test this assumption, we performed extra experiments for true distances at $\{0.01, 0.05, 0.25\}$. We repeated the subsampling procedure with $\alpha \in \{4/5, 6/7, 9/10, 13/14, 20/21\}$ leading to coverage $\{0.11, 0.25, 0.47, 0.71, 1.00\}$ when starting from 2x and $\{0.58, 1.50, 3.05, 4.89, 7.26\}$ starting from 16x. Skmer failed to estimate coverage and genome length in rare cases, and these replicates were excluded from the analysis.

We used INDELible [70] to generate long sequences (representing genomes) under the [65] (F81) model with uneven base frequencies 0.26, 0.21, 0.24, 0.29 for T, C, A and G and no indels. Note that the F81 model with base frequencies so close to 1/4 is extremely similar to the JC model used for phylogenetic distance inference by Skmer. We used ART [89] with its default short-read error profile to produce low-coverage error-prone reads that simulate genome skims (Appendix D).

Felsenstein-zone quartet trees. We simulated a challenging dataset meant to create conditions prone to long branch attraction. First, we generated 1000 quartet trees with branch lengths close to the Felsenstein zone; i.e., three short branches and two separate long branches (Fig. 3.2B). For each replicate, we draw short lengths from log-uniform distribution spaced between 0.0001 and 0.001 and draw the long lengths from a uniform distribution between 0.05 and 0.12 (Fig. B.28AB). Once the trees were available, we used INDELible, the F81 model with, and ART with settings identical to the genome pairs experiments.

8-taxon simulations with model misspecification. We simulated an 8-taxon dataset with a procedure similar to the previous datasets, with two changes to the model: 1) we used the GTR model of [232], which can greatly violate the assumptions of the JC model used for estimation, 2) we added (unmodelled) rate variation across sites using the standard Gamma model of rate variation [98]. In addition, this dataset included 8-taxa, necessitating the choice of a topology: we used both fully balanced and fully unbalanced (i.e., caterpillar) topologies and simulated 120 replicates for each. Branch lengths were randomly selected from the log-uniform distribution

spaced between 0.00001 and 0.12 (Fig. B.28C). We used Dirichlet (19, 14, 14, 19) to draw the base frequencies for A, C, G, and T. Entries of the GTR matrices were drawn from Dirichlet (50,7,12,12,14,50) for C↔T, A↔T, G↔T, A↔C, G↔C, G↔A. To set rates across-sites, the 1-centered Gamma model was used with α drawn from a log-normal distribution with log mean 2.55 and log standard deviation 0.18. Note that α is the inverse of the variance of the Gamma distribution of rate multipliers. All these hyperparameters were estimated using maximum likelihood estimation from a collection of published gene trees from the ruminant genomes project [42].

Measuring accuracy on simulated datasets

To evaluate the accuracy, after computing support using Algorithm 2, we bin all 1000 replicates into seven groups based on the support value of their only internal branch and compute how often trees in each bin are correctly inferred (Fig. 3.2B). If supports values are calibrated, the percentage of correct replicates in each bin should be close to the midpoint of that bin. However, even when support values are not calibrated, they can still be predictive (i.e., informative in separating correct and incorrect branches). We evaluate the power of support in distinguishing correct/incorrect branches using ROC curves (Fig. 3.2B) and drawing the support distribution for correct/incorrect trees. We also show the empirical cumulative distribution function (ECDF) of the support of correct and incorrect branches.

Real biological datasets.

We use five biological datasets, including those with assemblies, raw reads, and simulated genome skims.

Cetaceans. We used two cetaceans datasets, one for within species and one for cross-species relationships. We used all 27 cetacean assemblies from NCBI (Table C.21) to create the across-species dataset. For within-species, we used the dataset of 20 low-coverage genome skims of Gray's beaked whale *Mesoplodon grayi* (Table C.22) published by [245]. We removed adapters,

deduplicated these samples, and merged paired-end reads using BBTools [40] (Appendix D). Since contaminates can impact distances [189], we filtered reads that do not align to a reference genome *Mesoplodon bidens* using Bowtie2 [109, 111] (sensitive setting ; see Appendix D). For genome skims, we used $\alpha = {}^{20}/_{21}$ instead of the default ${}^{9}/_{10}$ to accommodate the low coverage of these samples before subsampling (0.78–0.96x).

Insect datasets. We used 19 bee reference genomes [226] (Table C.23) and *simulated* genome skims at 1x, 2x, 4x, and 8x coverage using ART. We also used 14 Drosophila assemblies as well as their corresponding high-coverage SRAs (Table C.13), which we downsampled to 200 MB using seqtk [122] to get real genome skims. Assemblies were used without preliminary processing. For real skims, we used a pipeline similar to beaked whales to remove adapters, deduplicated reads, and merge paired-end (Appendix D). We filtered out human reads using Kraken [250] and filtered out microbial contaminants by querying them against the GTDB database using CONSULT [187]. The rest of the procedure was identical to beaked whale genome skims, but we used the default $\alpha = {}^{9}/_{10}$.

We also used a real Lice by [33] where the original paper provides an alignment-based tree to test whether our support values can identify questionable branches. We collected 61 high coverage lice SRAs (Table C.24) and downsampled them to 400 Mbp to emulate genome skims. We removed adapters, deduplicated these samples, and merged paired-end reads (Appendix D) but no filtering of contaminants was performed.

Chapter 3, in full, is a reprint of the material as it appears in Quantifying the uncertainty of assembly-free genome-wide distance estimates and phylogenetic relationships using subsampling. Rachtman, E.; Sarmashghi, S.; Bafna, V., & Mirarab, S., Cell Systems, 2022. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Alignment-free phylogenetic placement using machine learning

Continuous increase in size of the available sequencing datasets makes *de novo* phylogenetic tree reconstruction a challenging problem. As a result, updating phylogeny through phylogenetic placement becomes an important alternative. Placement can be used to add newly sequenced samples to the reference tree obviating the need of reconstructing phyogeny from scratch as well as can be helpful in uncovering identify of unknown query samples obtained through metabarcoding, skimming, or metagenomic data. However, attempts in alignment-free phylogenetic placement have both scalability and accuracy limitations. In this paper we try to overcome challenges of conventional distance based phylogenetic placement methods by leveraging machine learning. Thus we represent each input genome as a vector of k-mer frequencies and train machine learning method to estimate distances between such vectors. Once pairwise distance matrices are computed they are used for phylogenetic placement. We demonstrate that our method, kf2d, outperforms existing k-mer-based approaches in distance calculation and allows accrutate placement new samples on phylogenies constructed from heterogeneous data types.

4.1 Introduction

The cost of DNA sequencing has reduced dramatically over the past years [143]. As size of size of the available genomic datasets grows *de novo* phylogenetic tree reconstruction becomes a challenging problem. Therefore, finding a way to update phylogeny without re-computing original tree becomes an attractive alternative [17]. As previous studies showed for high-throughput applications, phylogenetic placement—adding a new sequence to the existing reference tree—is often sufficient and in some cases even more accurate than more laborious *de novo* tree reestimation [90].

The problem of phylogenetic placement is not new and there is a number of methods that have been developed to address this question [17, 21, 23, 56, 65, 93, 142, 158]. Placement algorithms are either gene trees based or distance based and both groups have their own challenges. Thus gene tree based solutions are designed to add sequences from a single gene family to the tree showing its evolutionary history (i.e., the gene tree). However phylogenetic relationships change across the genome [52, 134] due to horizontal gene transfer (HGT) [166] and such discordance should be taken into account [243]. Gene tree based methods place on gene trees. Therefore they assume that gene trees are similar to species trees and hope that their differences are unimportant. Additionally, even prior to placement the process involves identification of marker genes and their multiple sequence alignment, steps that are very computationally laborious and might be infeasible for a number of applications [110, 120, 121]. Since alignment methods are typically computationally too slow, more recent compositional alignment free k-mer based approaches with more competitive performance have gained popularity. However, compositional distance based phylogenetic placement methods are suffering scalability and sometimes accuracy issues.

Taking these points into account we suggest that this problem should be approached differently. We believe that the key would be to use k-mer based composition of the sequence as input for distance computation and at the same time leverage machine learning methods as a way to capture more complex phylogenetic relationships between the species along the tree. The

purpose of machine learning in this application is to learn a low dimensional representation (an embedding) of the entry sequence such that distances between embeddings correspond to the phylogenetic distance between the species on reference tree.

We note that in recent years, distributed representations of words in a vector space has been increasingly used in Natural Language Processing (NLP) to improve the performance of learning algorithms [153]. These representations, called embeddings, are projection of words in a low dimensional numerical vector space capturing semantic and lexical information learned from contexts of words. These vectors can be used as features for many applications like sentiment analysis [132], translation [184] or even speech recognition [22], outperforming standard word count representation. DNA sequences have been treated in a similar fashion with some prepocessing [99, 147, 164]. Evidently there are some important distinctions between metagenomic and NLP data. Thus when sequencing read is split into possible k-mers there is no notion of delimiter and some contextual information might be less important. Nevertheless vector representation of DNA k-mers might serve as input to machine learning models and produced embeddings of sequencing reads might be successfully utilized in downstream phylogenetic analysis.

So far there have been only a few attempts to learn machine learning k-mer embeddings directly from raw metagenomic data. Most of them address the task of predicting the origin of reads (taxonomic profiling) at a certain taxonomy level or to perform phenotype classification. To assign taxonomic information to each read, FastDNA [147] uses embeddings of k-mers and claims to achieve performances comparable to the state-of-the-art. In the first step of their approach they define the length k of the k-mers that describe the DNA sequences. Then they run the fastText algorithm [99] to learn low-dimensional embeddings (dimension from 10 up to 1000). All k-mers in a sequence are replaced by their corresponding vector and summed to get an embedding of the read they belong to. Then this new vector is passed to a linear classifier, which computes the softmax function and minimizes the cross-entropy loss by gradient descent. The k-mer embeddings are directly learned from the read classification considering the

result of the loss function. The authors demonstrated that significant results of prediction appear with k-mer size > 9 bp. With a similar objective, Liang et al. [124] propose DeepMicrobes, a neural network with an architecture composed of an embedding of k-mers, a bidirectional long short-term memory (LSTM), and a self-attention layer to predict the species or the genus of a read. The authors show that performance increases with k up to $k = 12$, the largest possible that allows to fit the model in the memory of the hardware needed to train the model. Georgiou et al. [74] manage to reduce the memory footprint of the model by using a locality-sensitive hashing (LSH) of k-mers, allowing them to test models up to $k = 13$.

In the work of Woloszynek et al. [248], the objective is to add, in addition to taxonomic profiling, a method to retrieve the source environment of a metagenome (phenotype prediction). A Skip-gram word2vec algorithm [153] is trained for k-mer embeddings and a SIF algorithm [9] is used to create reads and samples embeddings. They demonstrate the usefulness of such an approach for clustering and classification tasks. Moreover, they show that such embeddings allow models to be trainable using k-mers with big k (greater than 9), which is not possible when relying on simpler representation such as one-hot encoding because of their size exponentially growing.

In this study we have developed a machine learning approach kf2d that uses as input k-mer frequency information and allows distance computation between query and reference genomes for purposes of phylogenetic placement. Our approach requires minimal data preprocessing. Method is agnostic to the origin of the reference tree and can be applied to a variety of different sequencing data types. In number of validation experiments we demonstrated that our method, kf2d, outperformed existing k-mer-based approaches in distance calculation and allowed placing new samples on phylogenies constructed from heterogeneous data types.

4.2 Results

We evaluated performance of kf2d on multiple datasets.

4.2.1 Benchmark datasets

Tree of Life. Tree of Life (TOL) [13] dataset was composed of 199330 microbial metagenomic assemblies and a reference phylogeny build using 381 marker genes. In our study we utilized 11,075 species. As a backbone tree we used original TOL phylogeny of 10575 species reported by Zhu et al. [257] and arbitrarily selected 500 queries with variable distance to the closest species from the larger tree. We note that for algorithm development and validation experiments we used original smaller TOL phylogeny with a set of 500 randomly selected queries.

Fungal dataset. We used publicly available fungal dataset [123] that included 1,672 genome assemblies and reference phylogeny generated using 290 gene trees. To select more challenging queries we picked 80 genomes with the largest distance to the backbone phylogeny. We trained kf2d on 1592 species with 28 outgroup taxa excluded.

Insect phylogeny. To test performance of kf2d further we used insect phylogeny of 49358 species [43]. Original study only reported phylogeny. For purposes of our experiment we utilized 1178 insect species for which full genomes were available for download from NCBI [206]. From available assemblies we selected 60 species with largest distance to the backbone to construct a query set.

Lichen metagenomes. Finally, to test performance kf2d on sequencing reads we utilized publicly available lichen dataset [230] that included 413 lichen metagenomes. For our analysis we selected 50 lichen samples based on species names of the fungal component reported in annotations. We picked samples for which assemblies with the corresponding genus were present on a fungal phylogeny analized above. Prior to computing k-mer frequencies for placement analysis lichen samples were filtered against Genome Taxonomy Database (GTDB) R05-RS95 database [175] using CONSULT [187] in order to remove bacterial reads. GTDB included 30,238 bacterial and 1,672 archaeal genomes, that were selected to represent 194,600 samples clustered at 95% nucleotide identity.

4.2.2 Experiments

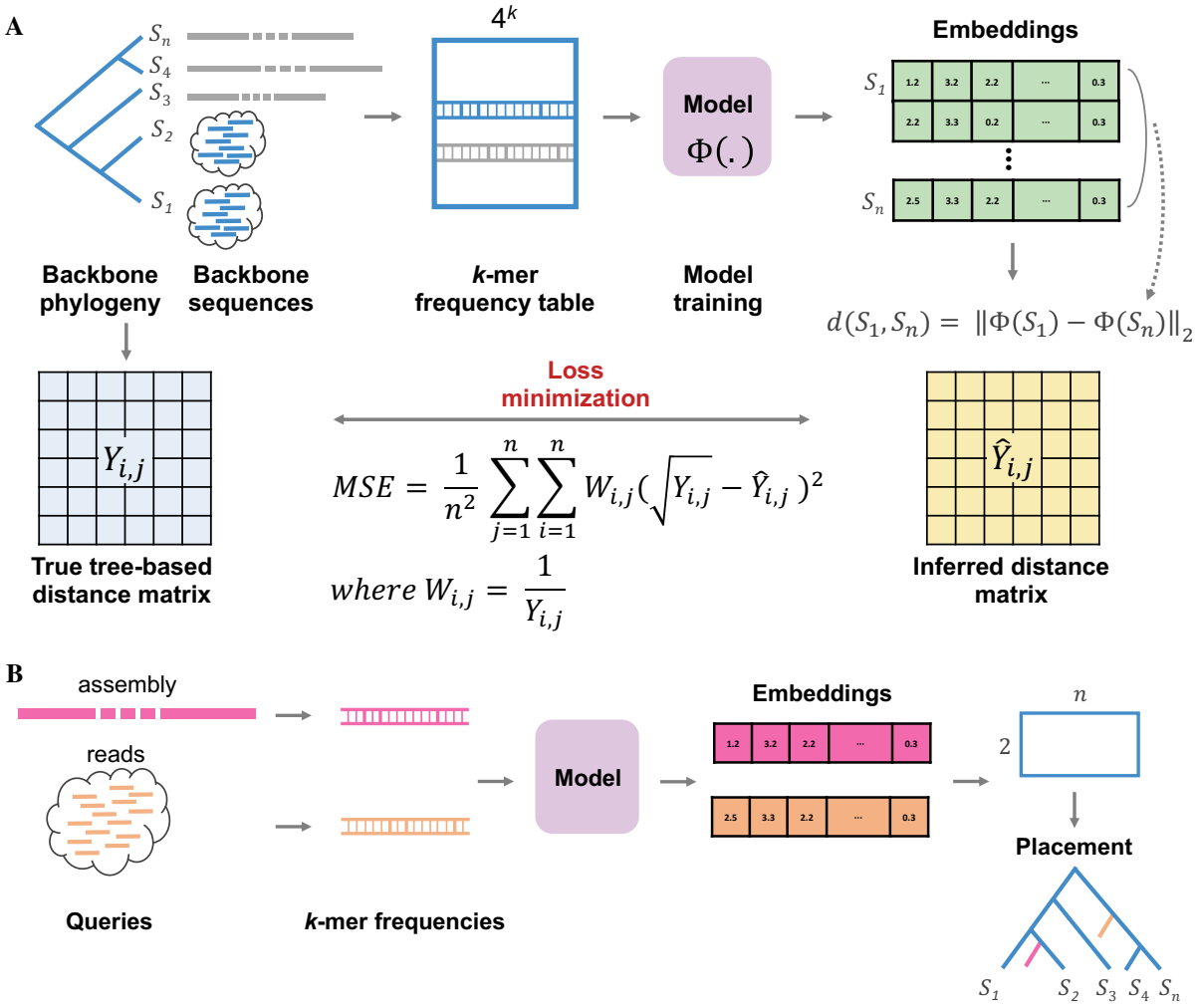


Figure 4.1. Method workflow. A) Training step. For every input backbone sequence k-mer frequency vectors are generated and summarized in feature table. The output of the model is embedding vectors that corresponds to each input sequence. Euclidean norm is computed for all pairs of embedding vectors to construct inferred distance matrix. True distance matrix is pairwise distance matrix obtained from the backbone phylogeny. Weighted mean squared error (MSE) is used to evaluate loss. B) Query step. Trained model is applied to the k-mer frequency vectors of the query sequences. Once embeddings are obtained, all pairwise distances between query and backbone embedding are computed to produce pairwise distance matrix. Pairwise distance matrix is used to perform placement.

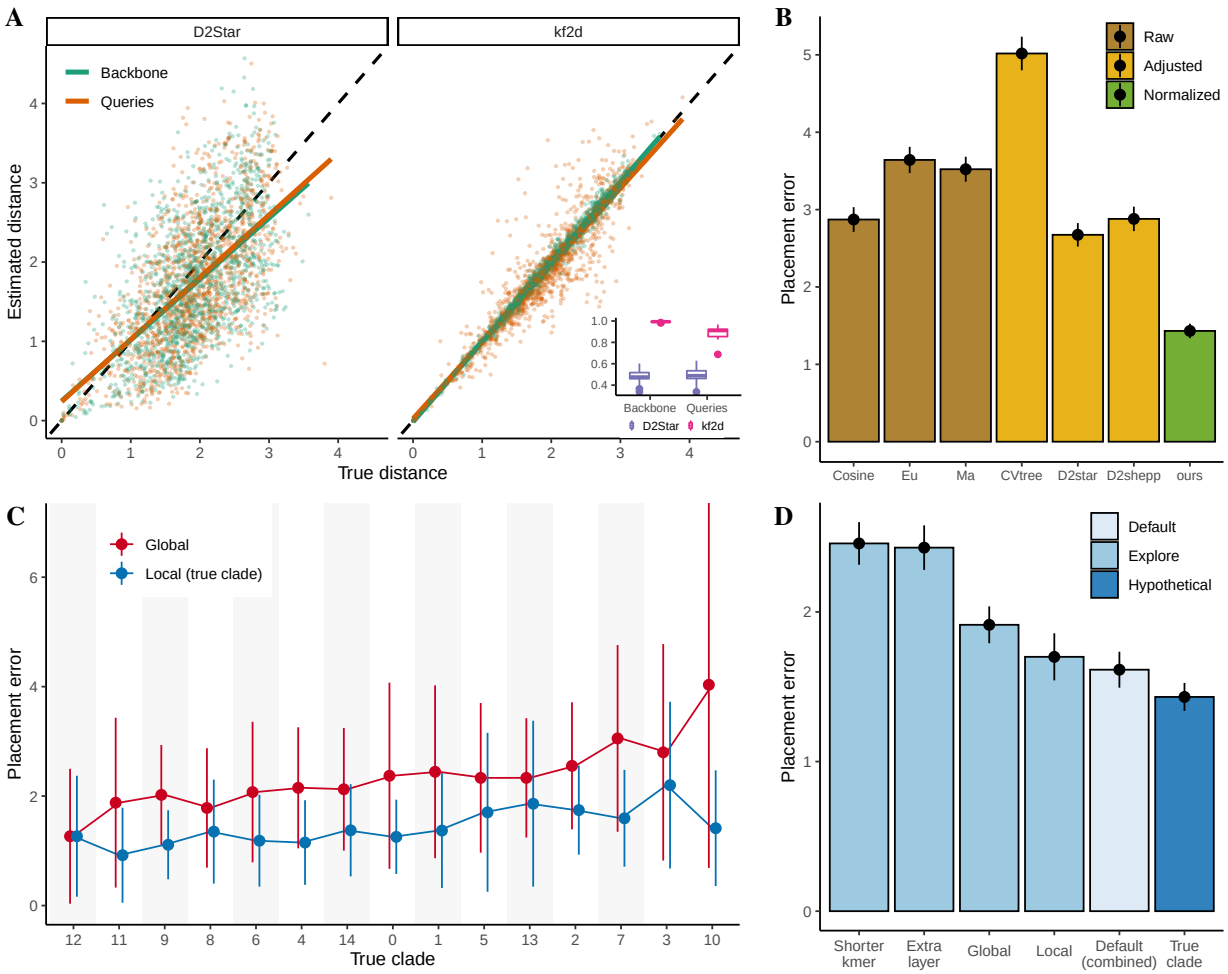


Figure 4.2. Performance of kf2d on TOL phylogeny. A) Distance comparison between kf2d and D2Star metric from CAFE on TOL phylogeny. Inset shows pearson correlation between true and estimated distance values produced by kf2d vs D2Star. B) Placement error for different CAFE metrics in comparison to kf2d. Placement errors for Co-phylog and JS metrics were 22.910 and 12.106, respectively and not shown on the diagram. C) Per clade placement error for global vs local model when samples are placed in true clades. D) Variable conditions of placement. Development TOL query set was used for all experiments on the panel.

Accuracy of distance estimation

We have compared accuracy of distance estimation between kf2d and multiple phylogenetically relevant distance metrics from CAFE suite (Fig. 4.2AB). D2Star, D2Shepp and Cosine distances most accurately represented phylogenetic relationships between bacterial species on TOL phylogeny with placement error of 2.674, 2.88 and 2.872 edges away from correct

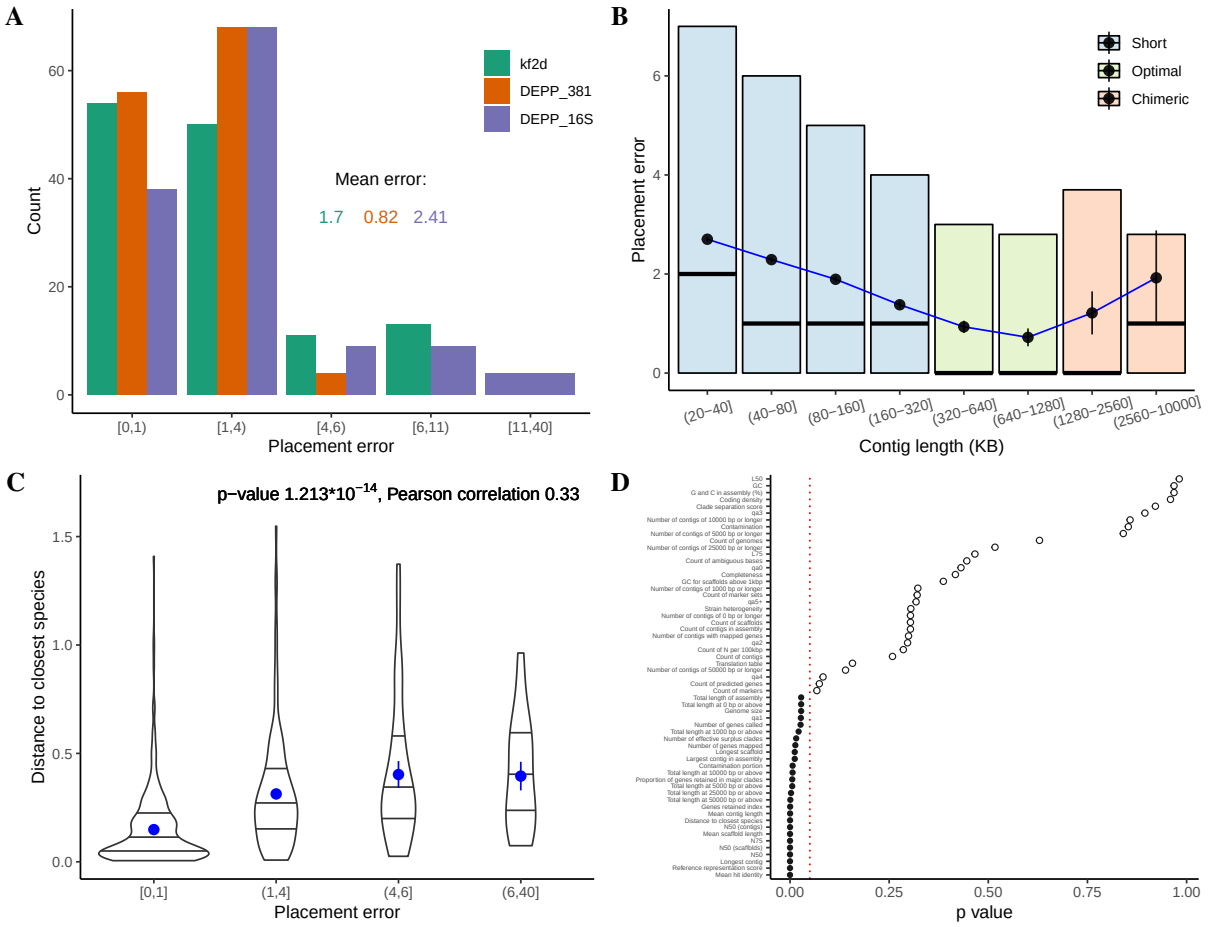


Figure 4.3. Variable conditions for placement of experimental queries on TOL phylogeny. A) kf2d is compared to DEPP with 16S rRNA gene and set of 381 markers. B) Effect of contig length. C) Distribution of placement error with variable distance from query to closest species on a reference phylogeny D) Effect multiple quality metrics obtained from GUNC [170], CheckM [176] and QUAST [152] ranked based on p value of the adjusted t-test performed for each metric individually. Detailed defintion of each quality metrics is provided in Tables C.25, C.26, C.27.

placement position (Fig. 4.2B). At the same time kf2d had placement error of 1.432 that was substantially better than values from any of the CAFE distance measures.

Direct comparison of estimated distances between kf2d and D2Star revealed that for both conditions either within reference species or between query and references kf2d produced much more accurate and most importantly unbiased phylogenetic distance estimations (Fig. 4.2A and Fig. B.29).

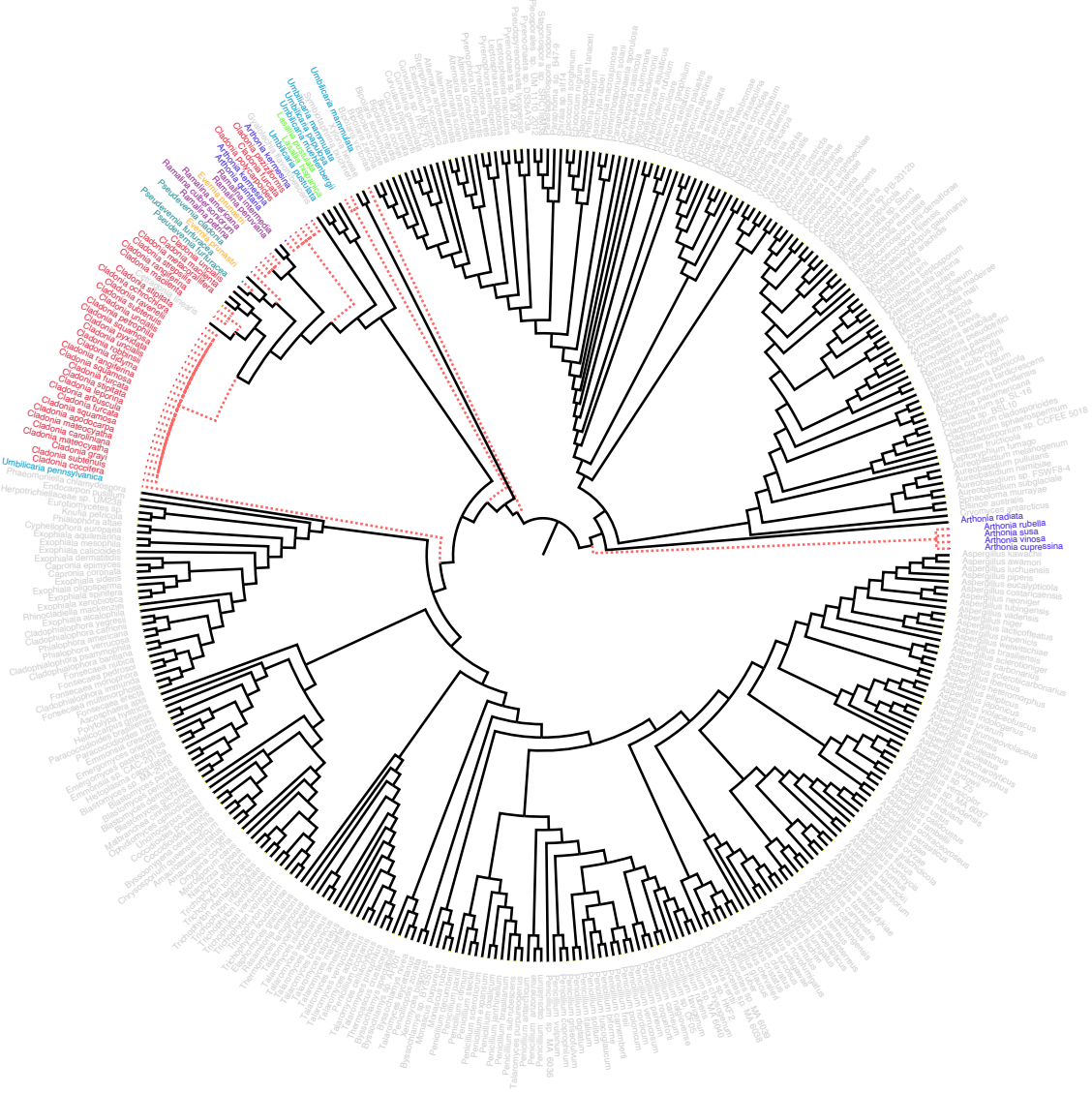


Figure 4.4. Lichen metagenomes placed on fungal phylogeny. Lichen metagenomic reads were placced on fungal phylogeny after filtering out bacterial reads. Dashed branches correspond to lichen queries. Colors signify the same genus for lichen samples as well as fungal species on a phylogeny. We extracted a subtree of 387 species rooted at the LCA of placed lichen queries.

Parameter testing

We learned that parameter selection for our model was quite robust. Thus changes such as shorter k-mer size ($k = 6$) for the input sequences or presence of extra linear layer in a model produced minimal change to the placement error. (Fig. 4.2D). Since our solution involves two steps: classification step that produces global model and per clade training that

generates individual clade models we evaluated possible effects of the model selection as well as placement using model combinations (Fig. 4.2D). We demonstrated that for larger phylogenies combinatorial solution worked the best. Thus queries for which classifier reported ambiguous clade identity should be placed using global model. At the same time query sequence where clade can be determined with high confidence should be placed using corresponding clade model (Fig. 4.2D). We also showed that different in placement error between clades is most likely guided by the presence of more challenging outlier queries in some clusters(Fig. 4.2C).

Optimal contig length

One of the main questions we aimed to answer is the recommended length of the bacterial assembly contig that can still be places correctly. Experiment revealed that optimal contig length should be around $300K$(Fig. 4.3A). Additionally we showed that shorter contigs might be placed correctly but larger distribution of placement error values was observed (Fig. 4.3A). We also observed that when contig length increases beyond $300K$ placement error want up. This is most likely due to the the fact that these extra long contigs have chimeric origin and therefore placing them might not be trivial.

DEPP results

DEPP is the method that is uses machine learning to learn genome embeddings from marker genes. In this experiment we tested effect of different number of marker genes used as a DEPP input and compared it's performance to kf2d. Our comparison showed that DEPP performance is guided primarily by number of input marker genes used in multiple sequence alignment. Therefore when number of marker genes is low kf2d produced comparable or slighly better performacence in terms of placement error (Fig. 4.3B). As number of gene trees increases placement DEPP performance becomes more accurate.

Fungal and insect placements

We assessed ability of kf2d to estimates distances for challenging queries(Table 4.1). We performed their placement and compared result to the phylogenies reported in a literature. In both cases kf2d was very accurate producing nearly perfect placements for the queries. Thus while placing queries on insects phylogeny only two species *Nesidiocoris tenuis* and *Bemisia tabaci* were placed incorrectly with mean placement error of 0.13 edges away. Fungal tree had 0.05 placement error with only three species *Middelhovenomyces tepae*, *Magnusiomyces tetraspermus*, *Acaulopage tetraceros* being misplaced.

Lichen metagenome placement

Quality of lichen metagenomic placement after filtering out bacterial reads was reasonable. Thus species from genuses *Ramalina*, *Pseudevernia*, *Lasallia*, *Umbilicaria* were placed together with the other representatives from the corresponding genuse (Fig. 4.4). Almost all *Cladonia* species clustered together and around *C. uncialis*, *C. macilenta* and *C. meracorallifera* species present on the phylogeny. Out of all lichen samples that we analyzed we observed three outliers *C. polycarpoides* and *C. ziziformis Evernia prunasti* that was placed quite far from corresponding representatives on fungal tree.

.

4.3 Methods

4.3.1 Data preparation

We used JellyFish [139] to extract counts of canonical k-mers ($k = 7$) from all input sequences that were either assemblies or SRAs (Fig. 4.1A). To account for differences in genome length each vector was normalizes such that k-mer counts would add up to 1.0. If particular k-mer was not present in a genome zero count values were added to the frequency vector. Next, we summarized computed k-mer frequency profiles into input feature table where k-mers were

organized in alphabetical order. We used Newick Utilities[102] to obtain ground truth distance matrix for a corresponding backbone phylogeny.

For larger phylogenies such as TOL single model might not be able to efficiently capture complexity of the input data. In such cases we recommend to use procedure similar to the one described in [95]. In particular we set branch length for all the branches on backbone phylogeny to 1.0 and split backbone into individual relatively similar size clades using TreeCluster [15]. In this set up for each input training sequence we have clade identity as well as distances to backbone species and training for every clade is handled independently. We note that this configurations also requires an additional step of training a classifier since clade identity needs to be determined upfront.

4.3.2 Algorithm

Input and output

Input into the software is a feature table S where number of rows corresponds to the number of backbone sequences and each entry is represented as 8192 feature vector. Output of the software is the pairwise distance matrix that represents relationships between queries and backbone sequences. Output matrix can be directly used as an input into placement software such as APPLES [17] in order to construct placement phylogeny (Fig. 4.1A).

Objective function

To compute pairwise distances between training examples we first use neural network model to generate embeddings for each entry sequence in $\mathbb{R}^m$ space, where $m = 1024$. Use of embeddings has a theoretical justification [94]. Thus according to [51, 114] for any tree T there exist a collection of points $P = \{\Phi(t_i)\}_{i=1}^{n}$ in the $\mathbb{R}^{n-1}$ Euclidean space such as the distance between the points $\Phi(t_i)$ and $\Phi(t_j)$ corresponds to $\sqrt{d_{ij}}$. Since we treat k-mer frequency vectors as a function of the tree there must exist their embedding that corresponds to the tree. In this context machine learning is used as a way to compute a function that minimizes divergence

between computed sequence embeddings and the square root of the distances between species on a given backbone phylogeny.

In other words the goal of the training process is to recover a model that maximizes a match between Eucledian norm of all pairwise distances between embedding vectors and a square root of phylogenetic distances in the reference tree in Eucledian Space. To train the model we defined mean squared error (MSE) loss as (4.1)

$$\underset{\Phi}{\operatorname{argmax}} \sum_{i,j} (\|\Phi(s_i) - \Phi(s_j)\|_2 - \sqrt{d_{ij}})^2 \tag{4.1}$$

where Φ is a model function that is applied to input k-mer frequency vectors to generate embedding $\mathbb{R}^m$ in the Euclidean space and $\sqrt{d_{ij}}$ identifies pairwise distances in the reference tree T. Estimated distance between i and j is $(\|\Phi(s_i) - \Phi(s_j)\|_2)^2$. However it is known that long distances in distance-based phylogenomics might substantially contribute to the error [68] but remain relatively unimportant with regards to placement [17, 27, 56, 73]. Therefore to facilitate accuracy they should be downweighted. As a result in practice we use weighted version of the mean squared loss (4.2)

$$\underset{\Phi}{\operatorname{argmax}} \sum_{i,j} \frac{1}{d_{ij}} (\|\Phi(s_i) - \Phi(s_j)\|_2 - \sqrt{d_{ij}})^2 \tag{4.2}$$

where $\frac{1}{d_{ij}}$ is a weight computed to each pair of distances. We also note that similar version of loss function was successfully utilized in [93].

To accomodate a case where larger phylogeny split into multiple smaller clades in addition to training an embedder we need to train a classifier so that clade identity of the query can be determined before distance to the backbone is computed. In such case to train the model we use a combination of MSE and negative log-likelihood (NLL) losses. NLL version is generalized to the multiclass classification setting

$$l(\Phi) = -\sum_{i=1}^{i=b}(x_i log(\hat{x}_i)) \tag{4.3}$$

where b is a batch size, x_i and $\hat{x}_i$ are probability vectors for corrected vs predicted clades, respectively.

Query time

We note that kf2d allows to use as a query any type of sequencing data such as assemblies, SRAs, etc. Each query sample is represented as frequency vector of canonical k-mers. To determine clade identity of the query (if backbone tree is relatively large) we first apply a classifier to the vector to learn which clade model should be used for distance computation. To construct pairwise distance matrix model for a corresponding clade is run to generate embedding of the query sequences. Next Eucledian norm of embedding vectors is computed between all pairs of reference sequences from the particular clade and a query (Fig. 4.1B). Once distance matrix is constructed placement is performed.

Placement

Model is applied to the input queries we obtain a distance matrix which represents a relationships between query sequence q and a vector of reference sequences $D_1...Dn$. We use these distances to place q onto T using distance-based placement tool APPLES developed by Balaban et al. [17]. This software uses dynamic programming algorithm to find the placement with the minimum $\sum_{i=1}^{n} D_i^{-2}(D_i - d_{qi}((T))^2$, where where $d_{qi}(T)$ represents the tree-based distance between the query and each taxon i (Fig. 4.1B).

Implementation details

We implemented and trained our model using PyTorch[177]. Model is composed of two feed forward linear layers separated by rectified linear unit (ReLU) activation function. We use Adam optimizer [106] which is a version of stochastic gradient descent that uses estimations of

the first and second moments of the gradient to adapt the learning rate for each weight of the neural network. We recommend to train for no more than 4000 epochs. Value was determined empirically based on accuracy of the placement but can be modified by the user. Batch size is equal to 16.

For classifier step we have implemented a "forked" model architecture. In this configuration output of the first fully connected layer is used as an input into both second linear layer and as well as classification layer followed by softmax function. Other details are the same as in two layer model. Such configuration allows us to simultaneously train to obtain classifier model and global embedding model as well as a set of probability values for every clade. Empirically we determined that global model produced higher placement error than more refined local models trained on every clade separately. However in cases where clade identity of the query could not be determined with high probability we found that using global model for distance computation improves accuracy.

Model selection

To choose the best architecture, we performed a search over several hyperparameters: lengths of k-mers, number of neural network layers and the number of epochs (Fig. 4.2C,D). We used placement accuracy to compare performance of the models.

4.3.3 Experimental details

We evaluated performance of kf2d in multiple experiments.

Comparison with CAFE

We compared accuracy of distance estimation between kf2d and aCcelerated Alignment-FrEe sequence analysis (CAFE) [129]. CAFE is a modern state-of-the-art tool that allows to compute 28 different alignment-free dissimilarity measures using k-mer frequency statistics extracted from genome assemblies. In our experiment we computed 8 distance measures (CVTree, D2Star (d_2^*), D2Shepp (d_2^S), Cosine (d_2), Co-phylog, Euclidean (Eu), Jensen-Shannon

divergence(JS) and Manhattan (Ma)) that prior were shown to closely correspond to phylogenetic distances. We used smaller TOL reference phylogeny and development set of query sequences. Since recomputing entire distance matrix would be too time consuming we performed this experiment at the clade level. First we split reference phylogeny into 15 clades and computed all pairwise distances within every clade separately. Next we used these distance matrices to perform query placement on entire backbone tree using APPLES and compared constructed phylogeny with the original one. Results were evaluated based on error in distance estimation (comparing to the true distances from original phylogeny) as well as placement accuracy.

Comparison of kf2d to DEPP

We compared performance of our alignment free method with conventional alignment based approach DEPP [93]. DEPP [93] is a machine learning tool that takes an a input multiple sequence alignment of marker genes for references and query genomes and generates embeddings for all entry samples. Once embedding are produced all pairwise distances between them are computed. Inferred distance matrices can be used for phylogenetic placement. For our study we performed comparison on larger TOL phylogeny using a set of 500 experimental queries that were different from development queries used for kf2d testing. DEPP pretrained model were applied to different number of randomly selected marker genes. Once distances were obtained, query placement was performed using APPLES. Accuracy of placement was evaluated by comparing topologies of newly inferred phylogeny with the structure of the original tree.

Contig length evaluation

We aimed to address a question of recommended contig length for assembly to ensure accurate placement using kf2d. We used TOL phylogeny with experimental queries. For every genome we extracted k-mer frequency vectors for individual contigs of 500 query sequences using JellyFish [139]. All settings were unchanged. In total we obtained 76079 entries. Contigs shorter than 40000 were deemed too short to provide necessary k-mer information and were

filtered out. Contig count was reduced down to 7051 entries. Next we subdivided feature table based on clade composition and apply corresponding models. Once distance matrices were computed we performaned phylogenetic placement of individual contigs on reference tree using APPLES. Since species identity of each contig is known accuracy of placement was evaluated based on location of inserted branch. Each contig was placed and evaluated individually.

Placement on fungal and insect phylogenies

Both fungal and insect phylogenies were processed in similar fashion. In this experiment we treated species in both trees as single clade. Input samples were split into backbone and query sets. We trained kf2d using backbone reference species. Once models were trained we computed all pairwise distances between obtained embeddings of the queries and a backbone species. Placement was performed with APPLES and accuracy of placement error was assessed based on comparison with phylogenies reported in original studies.

Lichen placement

A lichen is a symbiotic organism of fungus with algae or cyanobacteria. Lichen has a very different morphology, physiology, and biochemistry than any of the constituent species growing separately [230]. The algae or cyanobacteria benefit their fungal partner by producing organic carbon compounds through photosynthesis. In return, the fungal partner benefits the algae or cyanobacteria by protecting them from the environment by its filaments, which also gather moisture and nutrients from the environment, and (usually) provide an anchor to it [82]. A number of lichen related studies focus on uncovering of the details of symbiotic relationships and less on characterization of diversity and exact composition of the lichen samples [230]. This trend is mostly due to challenging metagenomic nature of the lichen shotgun DNA-sequencing samples.

For our experiment we aimed to perform fungal species identification through placement. First removed bacterial reads from input lichen samples [230]. Then we obtained k-mer frequency

Table 4.1. Placement with good quality assemblies.

Dataset	Species count	Query count	Method	Placement error	Correctly placed (%)
Fungi	1601	73	kf2d	0.0547945	0.958904
Fungi	1601	73	EPA-ng	0.0958904	0.972603
Insects	1178	60	kf2d	0.133333	0.966667
Insects	1178	60	Skmer	0.216667	0.966667

counts using JellyFish [139]. Next we computed distances from lichens to the reference genomes in fungal phylogeny using kf2d. Placement was performed with APPLES [17]. Finally, we evaluated whether fungal samples were placed in proximity to the fungal species from the same genus on larger phylogeny.

Chapter 4, in full, is currently being prepared for submission for publication of the material. Rachtman, E.; Jiang Y.; Mirarab, S. Alignment-free phylogenetic placement using machine learning. The dissertation author was the primary investigator and author of this material.

Appendix A

Derivations

A.1 Derivation of (1.4)

Definitions

- Let L be the number of unique k-mers in the base genomes of species of interest, which for each of calculation, we assume is the same between genomes.
- ρ is the fraction of L k-mers that are different between the two base genomes.
- Y_1 and Y_2 are the total number of unique k-mers from each of the two contaminants.
- c_1, c_2 are the fraction of k-mers in sample one and sample two that come from contaminants. Thus, $c_1 = \frac{Y_1}{Y_1+L}$ and $c_2 = \frac{Y_2}{Y_2+L}$.
- $a = \frac{c_1}{1-c_1} + \frac{c_2}{1-c_2} = \frac{Y_1}{L} + \frac{Y_2}{L} = \frac{Y_1+Y_2}{L}$. Thus, $Y_1 + Y_2 = aL$.
- X is the total number of k-mers shared between the two contaminant sets, and H is the Jaccard similarity between k-mers of the contaminants of the two skims. Thus,

$$H = \frac{X}{Y_1 + Y_2 - X} \implies X = \frac{H(Y_1+Y_2)}{1+H} = \frac{H(aL)}{1+H}$$

Goal. We want to derive

$$J = \frac{(1-\rho)(1+H)+aH}{(1+\rho)(1+H)+a}.$$

Derivation. We assume coverage is high enough that all L k-mers from each genome are in the skim. Then,

$$J = \frac{L(1-\rho)+X}{2L - L(1-\rho) + Y_1 + Y_2 - H}$$

where the numerator counts the number of shared k-mers from the two base genomes plus the number of shared k-mers from the contaminants, and the denominator counts the total number of

k-mers. Results are obtained as follows.

$$\begin{aligned}
J &= \frac{L(1-\rho)+X}{2L-L(1-\rho)+Y_1+Y_2-H} \\
&= H\frac{L(1-\rho)+X}{L(1+\rho)H+(Y_1+Y_2-H)(H)} \\
&= H\frac{L(1-\rho)+X}{L(1+\rho)H+X} \\
&= H\frac{L(1-\rho)+\frac{H(aL)}{1+H}}{L(1+\rho)H+\frac{H(aL)}{1+H}} \\
&= \frac{(1-\rho)+\frac{aH}{1+H}}{(1+\rho)+\frac{a}{1+H}} \\
&= \frac{(1-\rho)(1+H)+aH}{(1+\rho)(1+H)+a}
\end{aligned}$$

Appendix B

Supplementary Figures

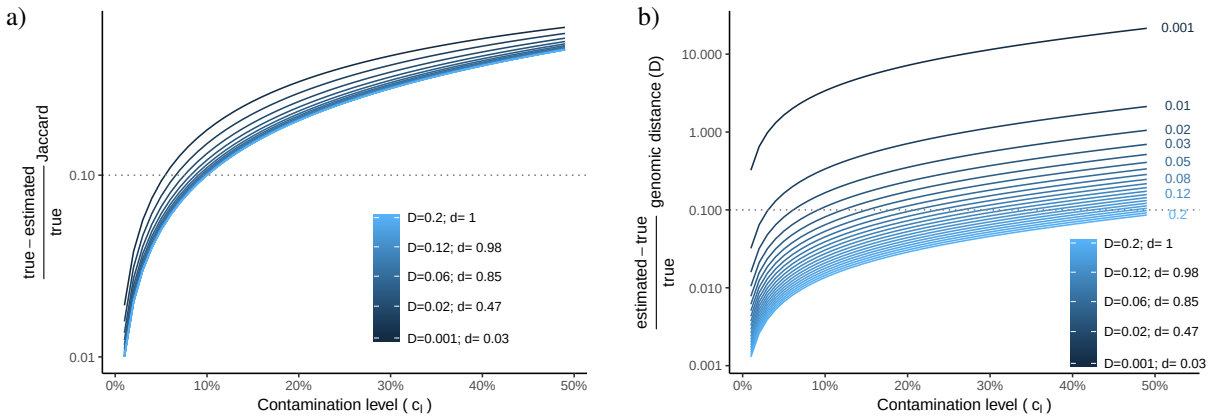


Figure B.1. Theoretical modeling. Impact of contamination on Jaccard (a) and the genomic distance estimated from Jaccard (b) according to theoretical expectation under the disjoint contaminant k-mer assumption. For various genomic distance ($D \in \{0.001\} \cup \{0.01, 0.02, \ldots, 0.2\}$) corresponding to $0.03 < \rho < 0.99$ and contamination levels $0.01 \leq c_l \leq 0.5$, the relative error of the Jaccard index (a) and the estimated Skmer distance (b) as a result of ignoring contamination are shown. $k = 31$.

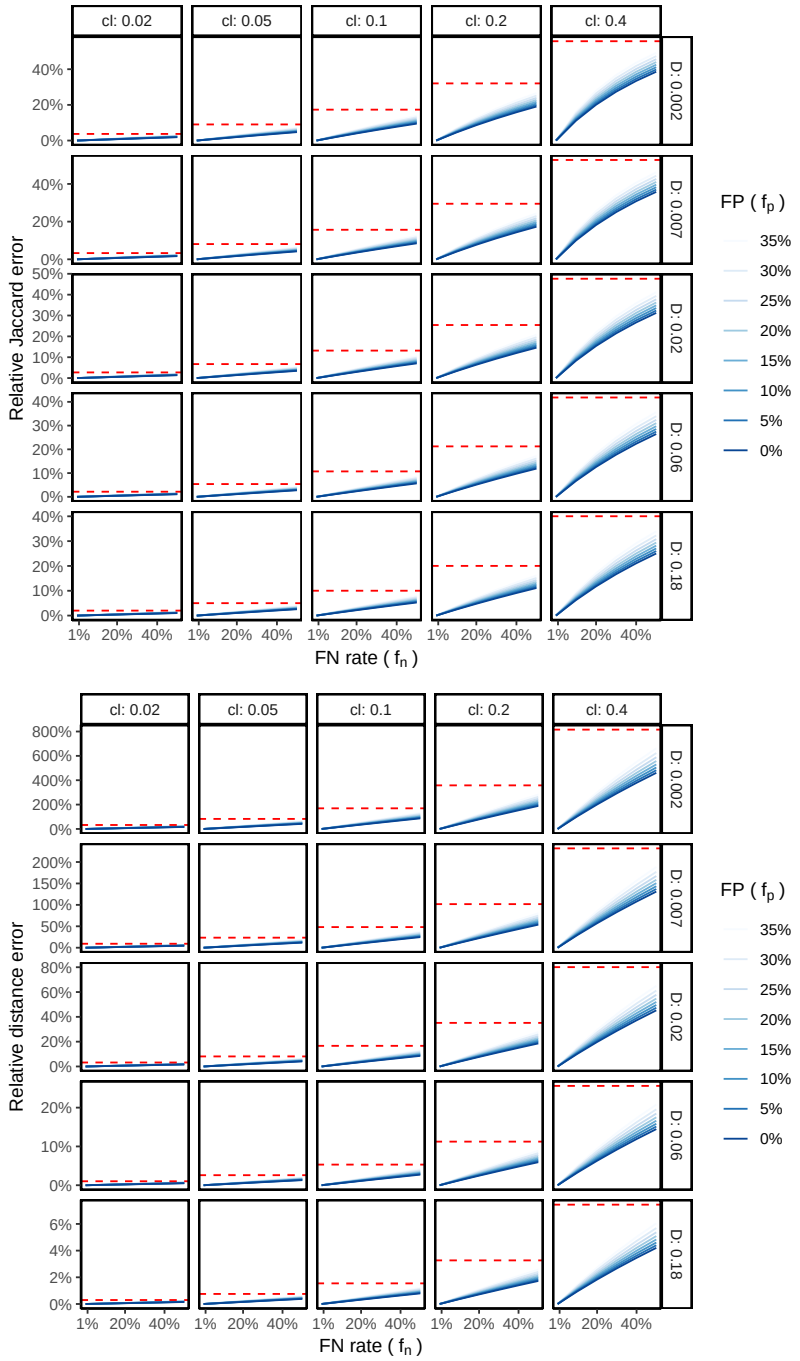


Figure B.2. Theoretical impact of filtering on Jaccard (top) and Skmer distance (bottom). Two genomes with D between 0.001 and 0.05 are both contaminated at $0.01 \leq c_l \leq 0.32$. Skmer distances are approximated using (1.1), with Jaccard approximated using (1.5). Results are for various levels of FP portion (f_p), and FN (f_n) rate. Solid lines show the relative error in Skmer distance after filtering, normalized by the true uncontaminated value, expressed as percentage. The error in the absence of filtering is shown as a horizontal dashed red line. See Figure B.3 for a tabular view.

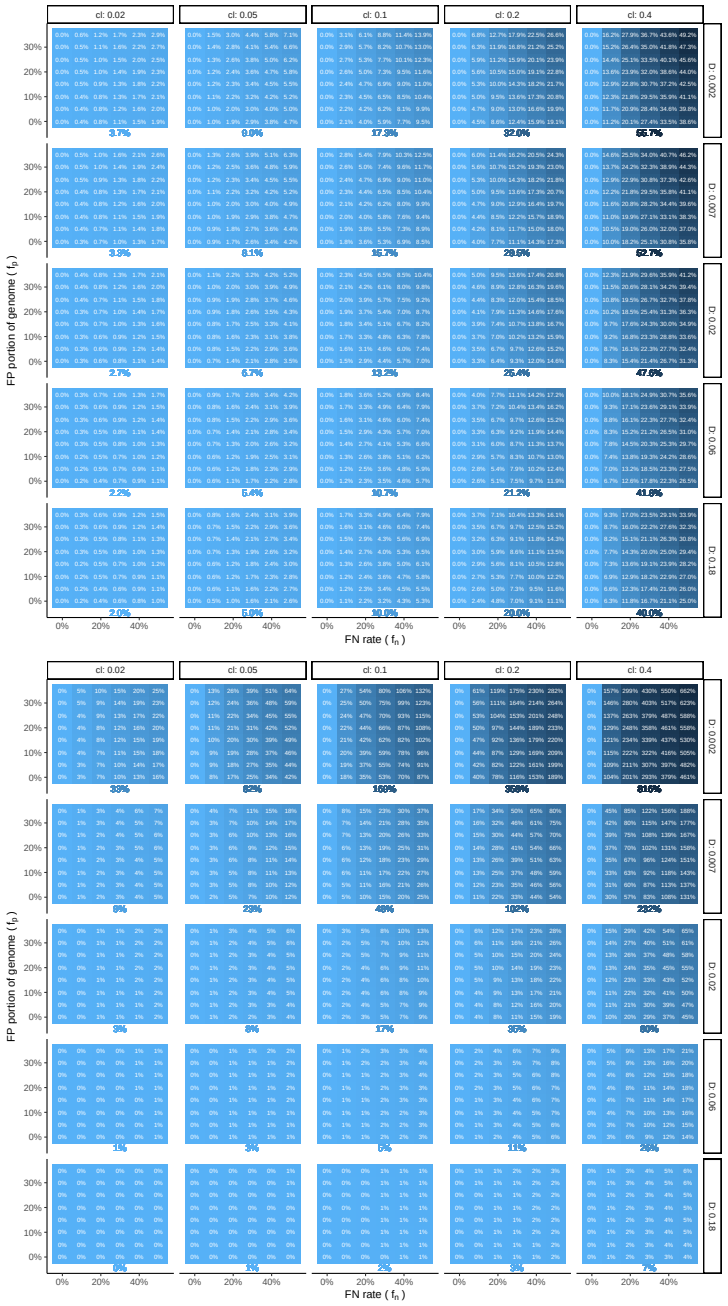


Figure B.3. Approximate impact on Jaccard (top) and distance (bottom). Two genomes with portion $0.05 \leq \rho \leq 0.75$ of their k-mers not matching (corresponding to D between 0.002 and 0.044) are both contaminated at $0.02 \leq c_l \leq 0.32$. Approximation of Jaccard using (1.5) Skmer distance D using (1.1) for various levels of FP portion (f_p), defined as the percentage of each genome skim that is filtered out by mistake, and FN Rate (f_n), defined as the proportion of the contaminating k-mers that have *not* been removed. Each box shows the error in Jaccard or distance estimation after filtering, normalized by the true value (i.e., value with no contamination), expressed as percentage. The error in the absence of filtering is shown as a single number below each box.

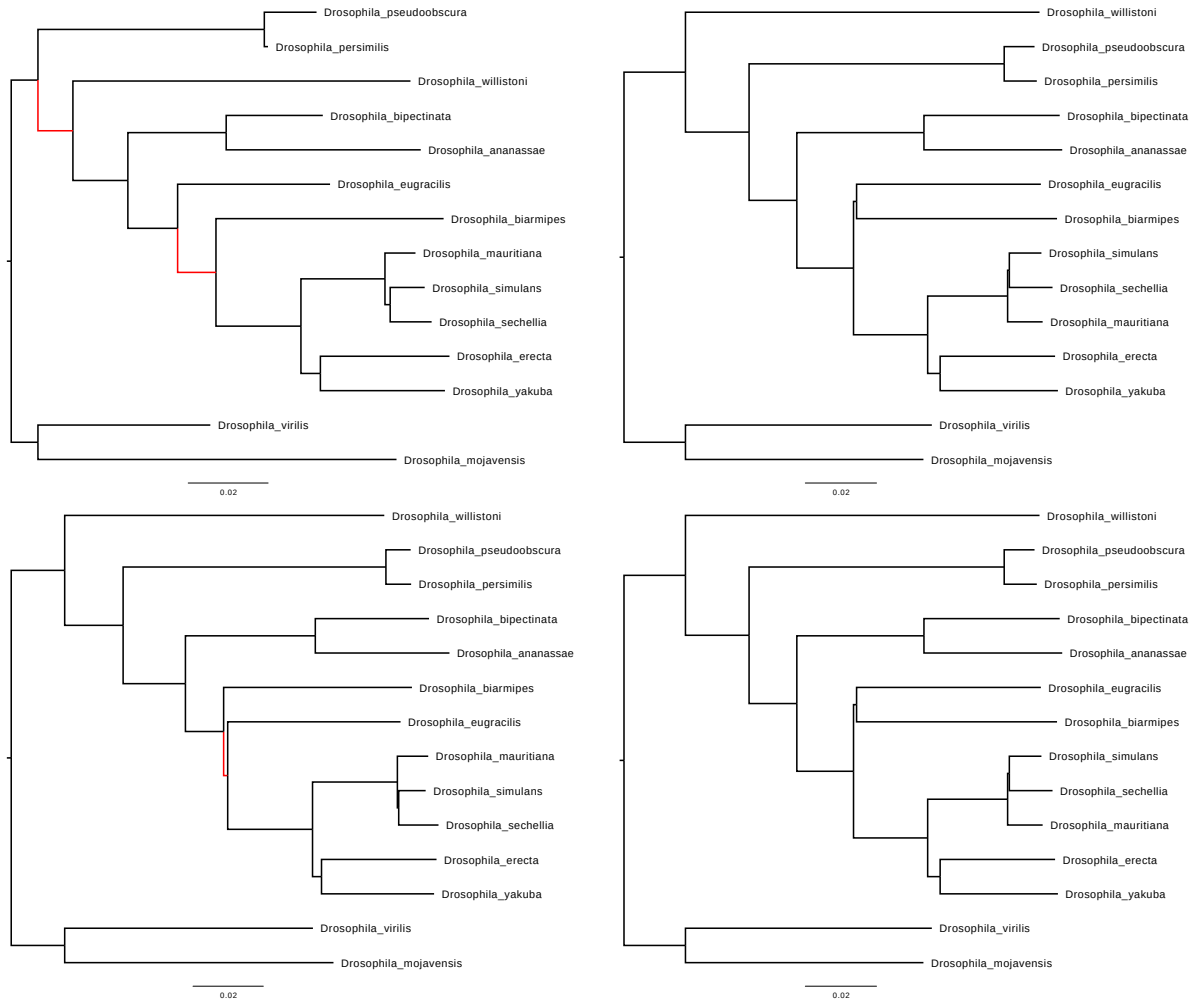


Figure B.4. Drosophila phylogenies. Drosophila phylogeny inferred from distances computed without filtering (top, left), and after filtering with Kraken-II (bottom, left) on 100Mb genome skims. Gold standard Drosophila phylogeny obtained from Open Tree of Life, whose branch lengths are computed using assembly distances, is shown twice (top and bottom, right). All trees are based on Jukes-Cantor model of evolution accounting for rate variation across sites using Γ model with $\alpha = 1$ and are inferred using FastME. Branches that do not match the gold standard phylogeny are indicated with red.

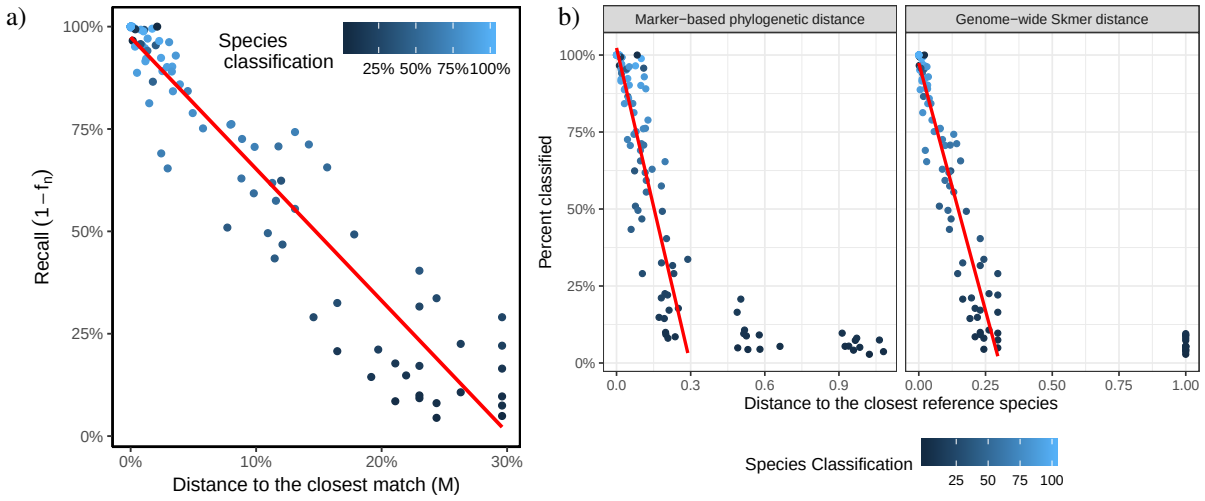


Figure B.5. Sensitivity analysis of Kraken-II. (a) For each query genome, dots show the percentage of reads classified (i.e., $1 - f_n$) by the default Kraken-II at the domain level or lower versus the distance of a query to the best match in the reference library (M), measured using Skmer. Default Skmer is not accurate for $M > 0.3$ and thus we show $M \leq 0.3$. (b) Similar to part (a), except, here, on the left, we measure M using either a phylogeny inferred from 381 marker genes and applying the inverse of the JC69 correction or by applying Skmer to the base assemblies. Using phylogenetic distances allows us to measure $M > 0.3$.

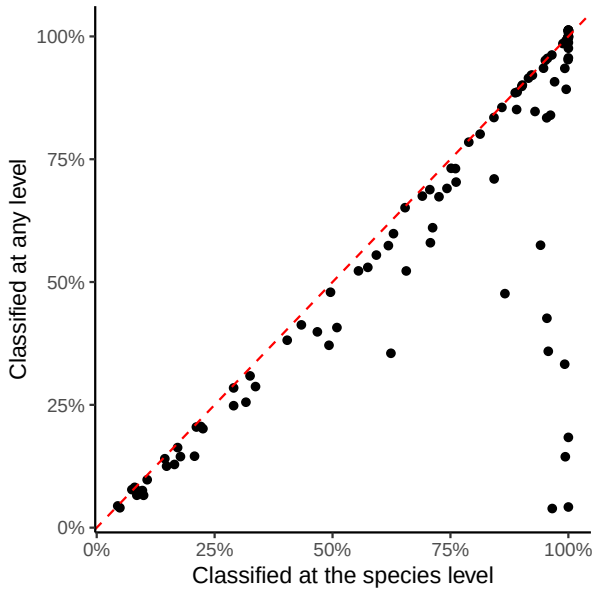


Figure B.6. Sensitivity analysis of Kraken-II. For each query genome, dots show the percentage of reads classified by Kraken-II vs percentage of reads classified at the species level.

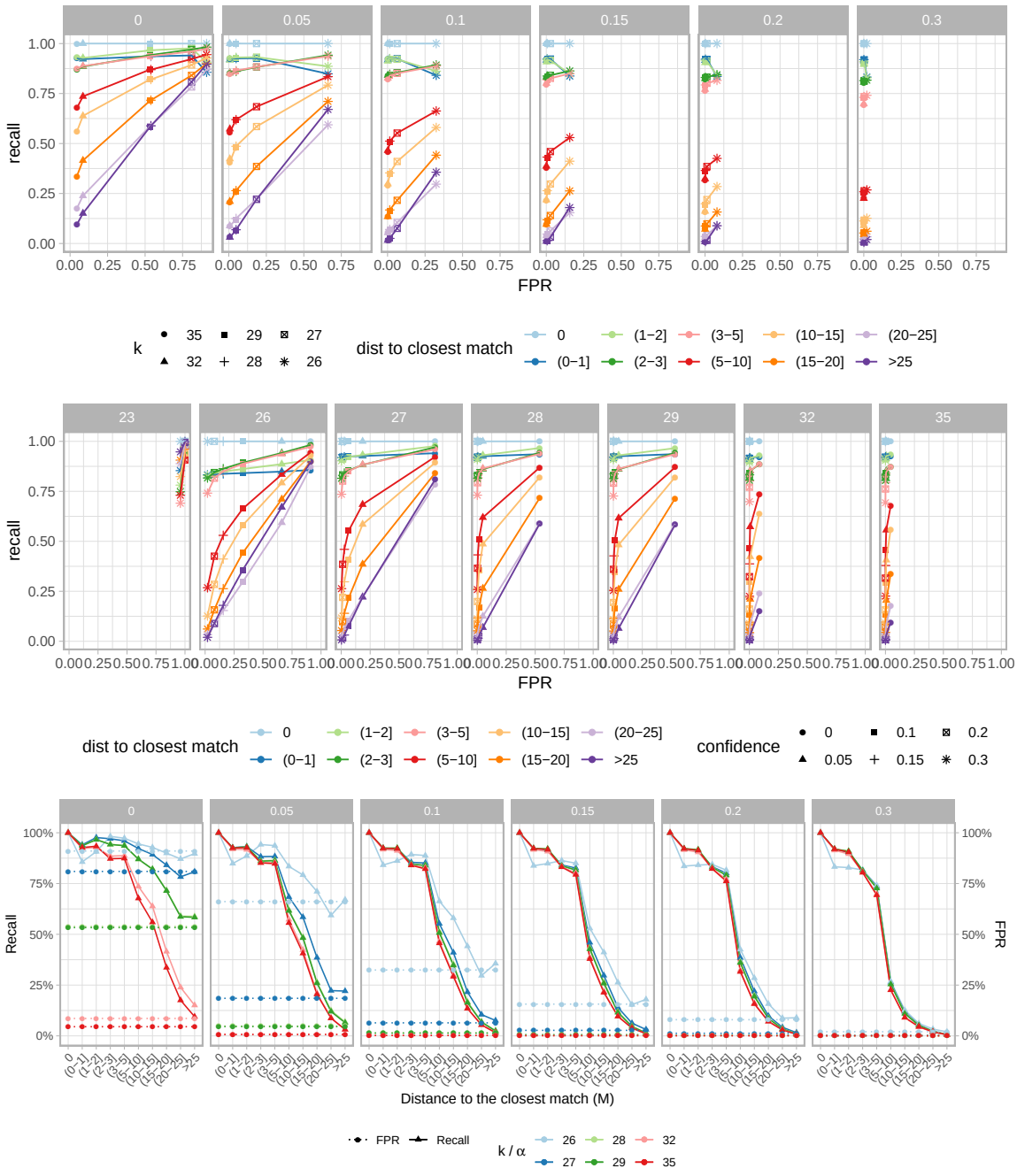


Figure B.7. Roc.

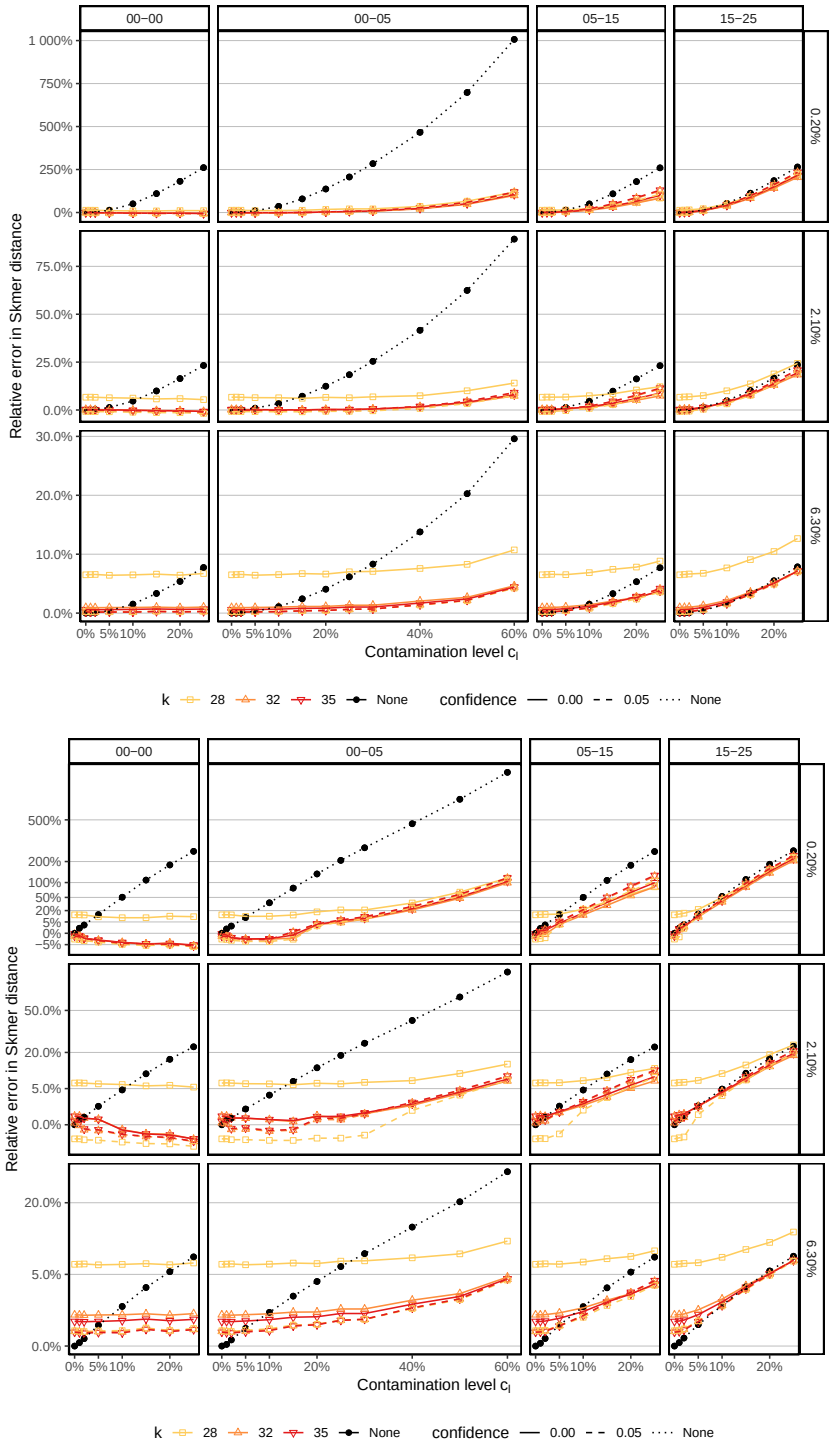


Figure B.8. Relative error of Skmer distances without (None) and with (colored) Kraken-II filtering with different confidence levels α and k. Contaminants are added based on sequences that are at distance range M from the sequences in the reference library, for four ranges of M (boxes). The pairs of Drosophilas are chosen to be at true distance $D = 0.2\%$, $D = 2.1\%$, or $D = 6.3\%$. top: normal scale; bottom: squre root scale.

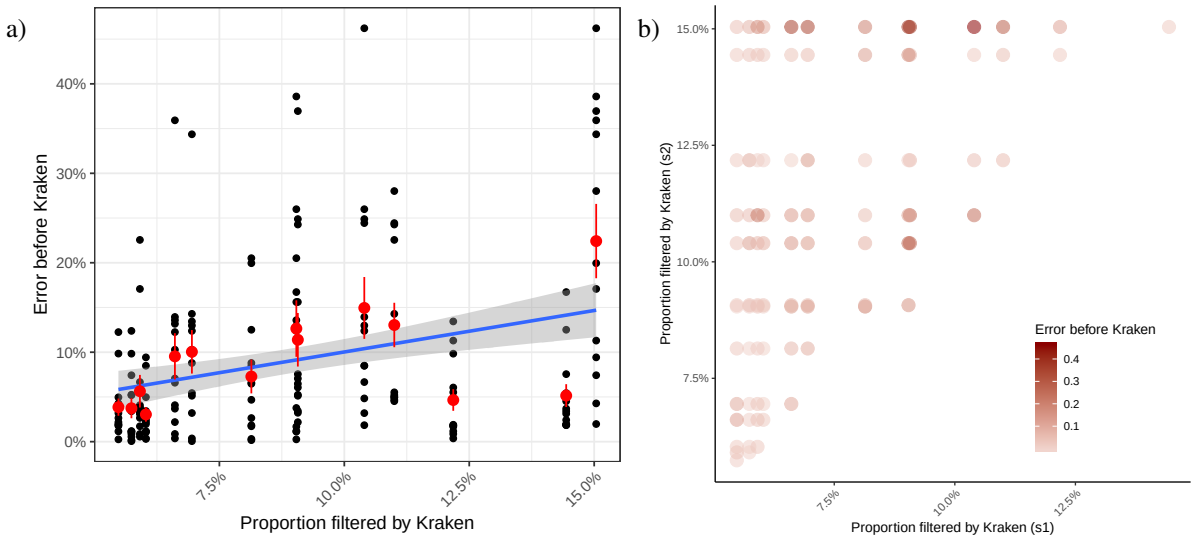


Figure B.9. Filtering with Kraken. (a) Proportion of reads filtered by Kraken-II from one of the two species being compared versus the error before Kraken-II in the genomic distance. (b) For each pair of species, colors show the relative distance error before Kraken-II versus the proportion of reads filtered from each of the two genomes. Error is associated strongly, but imperfectly, with high levels of filtering.

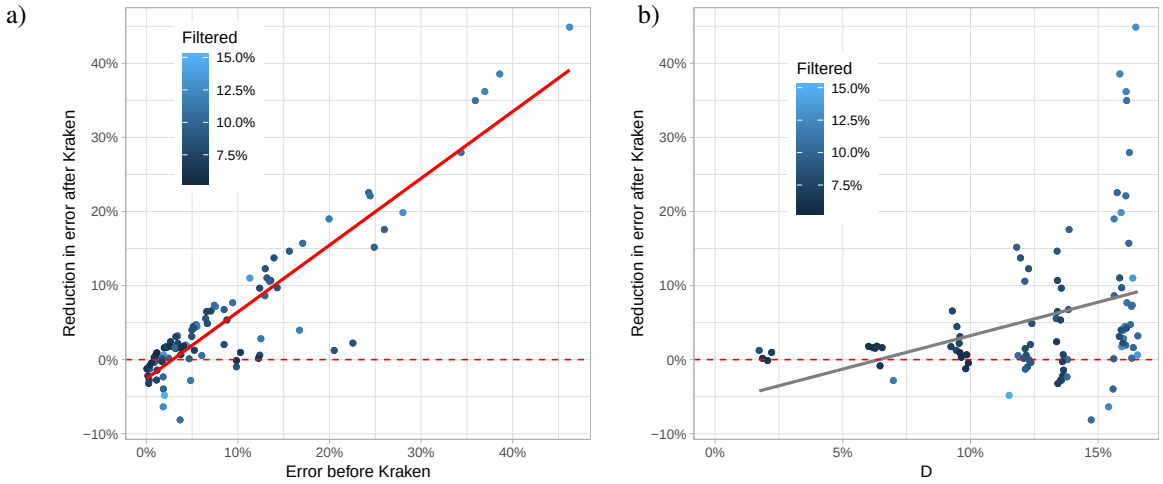


Figure B.10. Relative distance error. Change in relative distance error after filtering with Kraken-II for 100Mb Drosophila dataset versus (a) gold standard (assembly) genomic distance D, and (b) error before Kraken-II filtering.

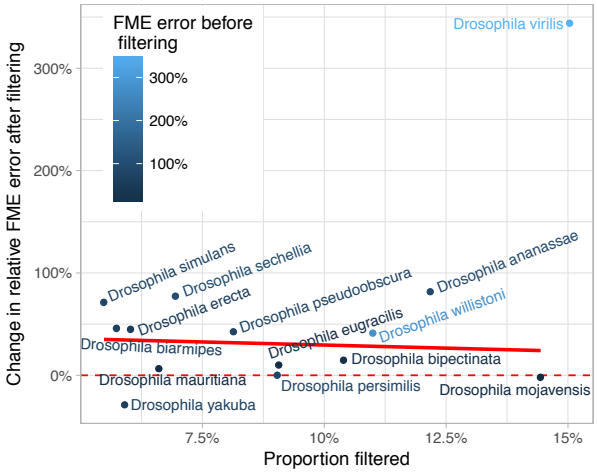


Figure B.11. Change in relative FME error per species after filtering. Solid red line: a trend line fitted to the points excluding *D. virilis*.

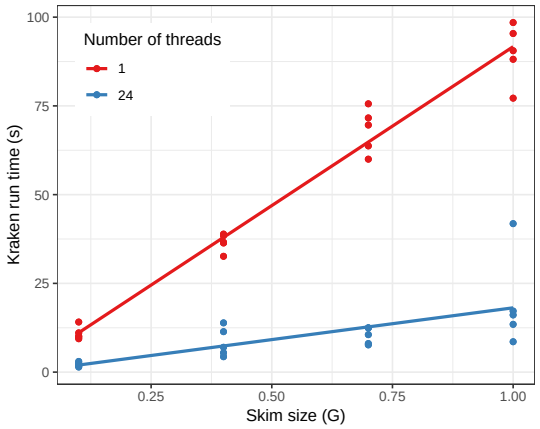


Figure B.12. Running time. Kraken-II processing speed (s per query) with respect to different genome skim sizes. Sequences were simulated using ART [88] (same settings as before) to reach at least 1.4GB of synthetic reads and subsequently downsampled to generate 1GB, 0.7GB, 0.4GB and 0.1GB genome skim benchmark set. The dataset was queried using Kraken-II default settings and standard reference library. Kraken-II was run on a machine with Intel Xeon E5-2680v3 2.5 GHz CPU and 120GB of RAM running CentOS Linux release-6-10.

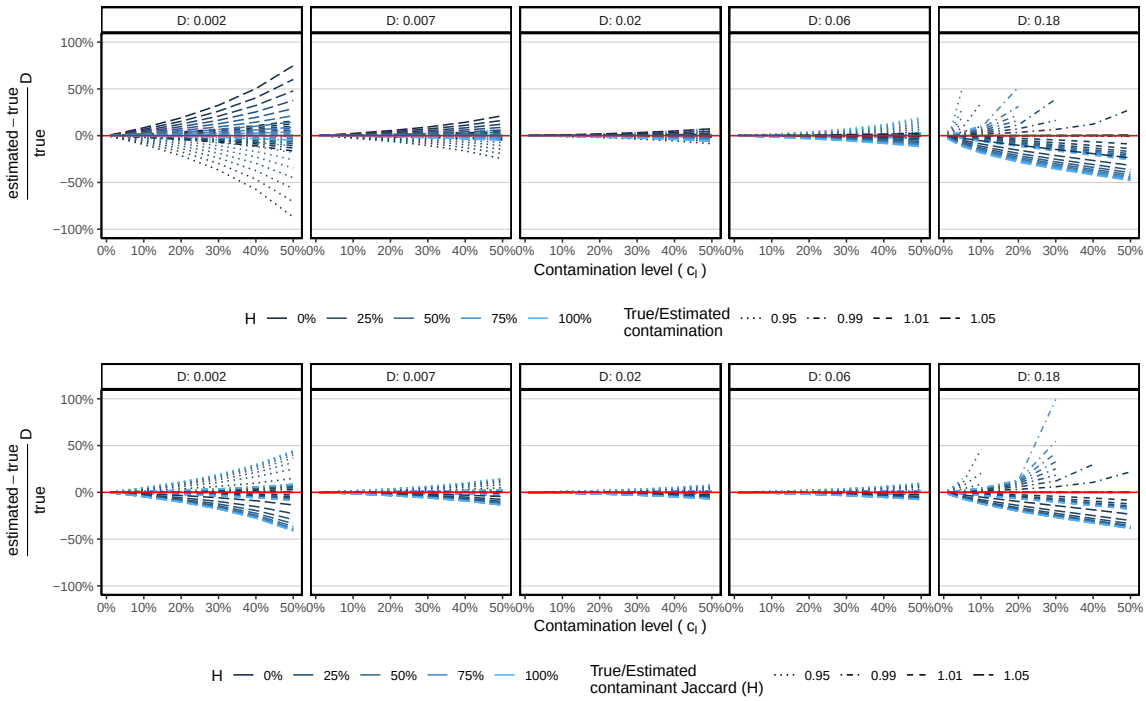


Figure B.13. The sensitivity of filter-free correction using theoretical modelling. For six values of D (boxes) and varying H, relative error is shown for various contamination levels (setting $c_l = c_1 = c_2$) when the estimated distance is corrected using (1.6) when c_l is miscalculated by small margins (1% or 5% over or under-estimated). Missing values indicate cases where (1.6) gives undefined values. $k = 31$ in all cases.

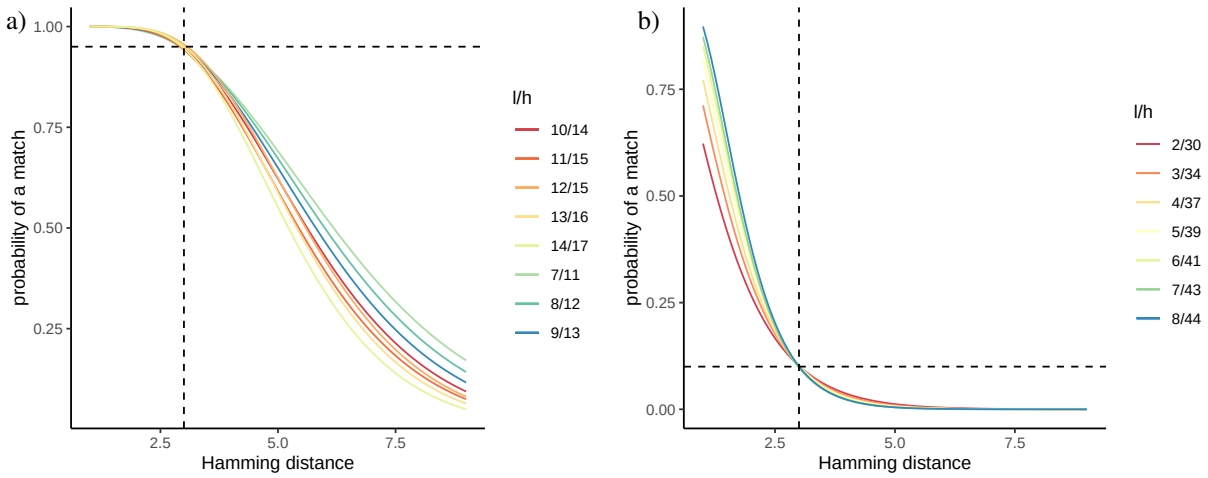


Figure B.14. Controlling true positive rate. Assuming we want a fraction ρ of k-mers in a read to match to sequences with p hamming distance to the read, we need l hashes and $h = \frac{\log(1-(1-\rho)^{\frac{1}{l}})}{\log(1-\frac{p}{k})}$ bits per hash to achieve the desired level. Figures show probability of a 32-mers matching another 32-mer ($p(d)$) if their distance d is the value shown on x-axis, for various choices of l, setting h such that $\rho(3) = 0.10$ (on the right) or $\rho(3) = 0.95$ (on the left) ($p = 3$ and $\rho(3)$ shown with dotted lines).

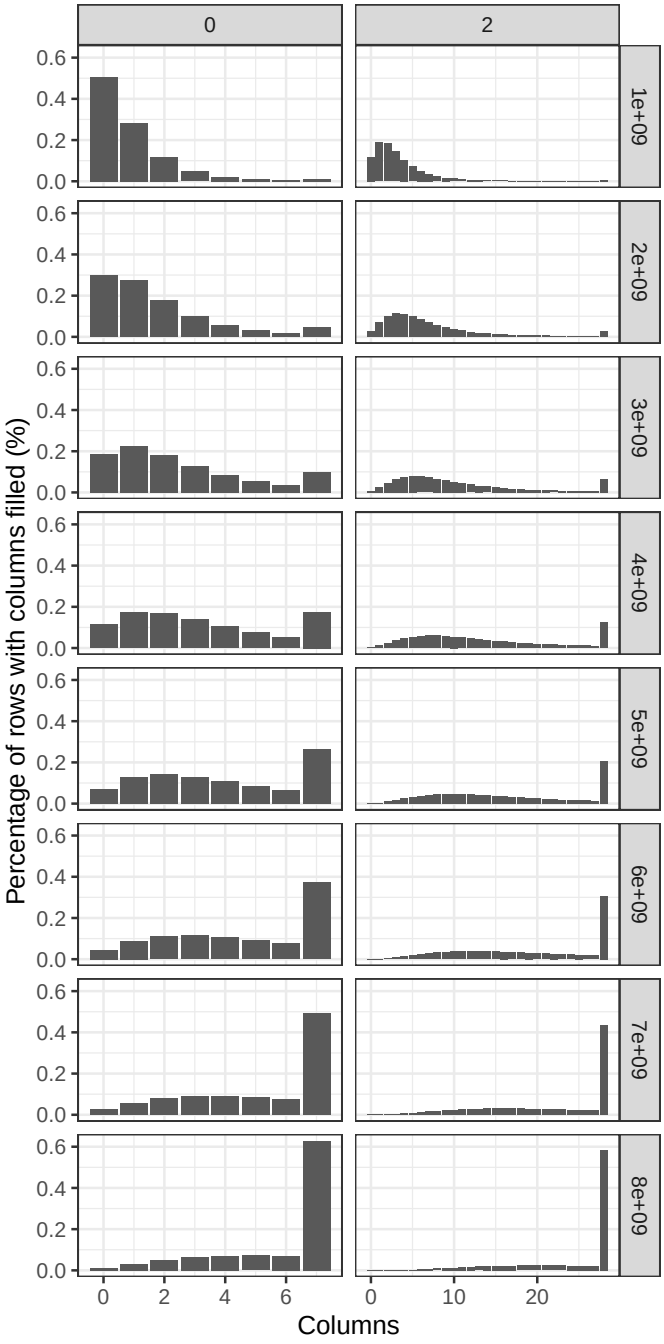


Figure B.15. Dynamics of filling out lookup tables during database construction. Each bar shows the percentage of rows where the x-axis is the number of columns that are populated with pointers to the encoding table. Row panels show counts after a number of k-mers were added to the database while column panels show tag size (0 or 2). Experiment represents a construction of TOL database. With tag 0, many rows starts to fill up around 3 billion k-mers, showing that row usage is not uniform. For most levels between 1 billion and 6 billion k-mers, fewer rows are full with tag 2 than tag 0. This pattern is the motivation for using $t = 2$ as the default.

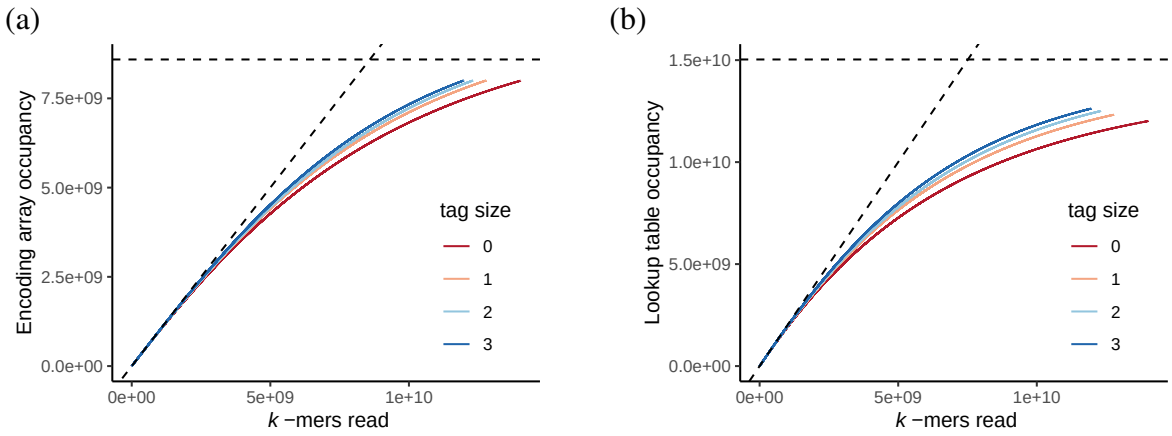


Figure B.16. Effect of tag size on efficiency of k-mer inclusion during database construction. x-axis shows the number of k-mers that were read during database construction at given point of time. y-axis represents (a) total number of encodings that were added to encoding array, and (b) total number of signatures that were added to lookup table. Experiment indicates that with $t = 0$, we have less efficient utilization of map capacity. Experiment was performed on TOL database constructed with default settings and $t \in \{0, 1, 2, 3\}$.

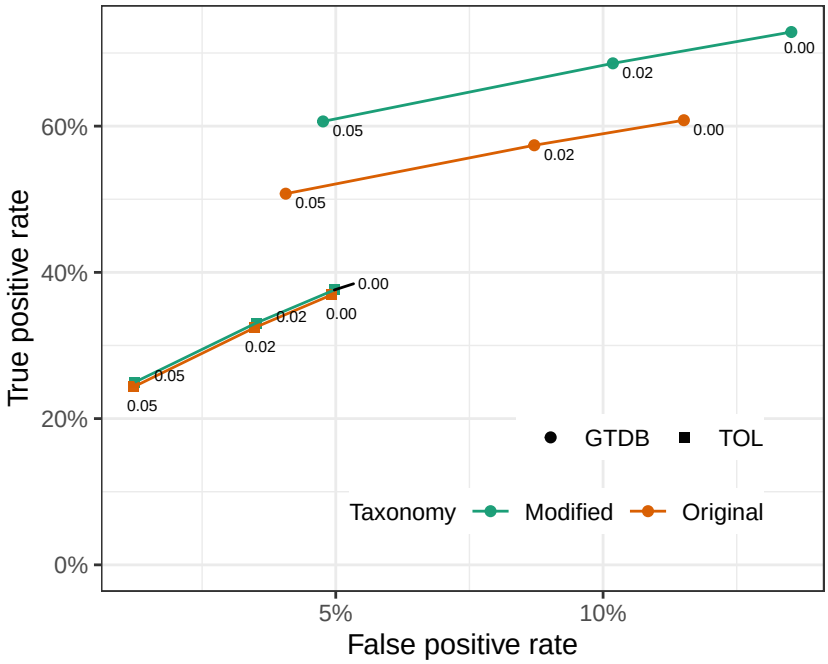


Figure B.17. Kraken-II comparing databases built with modified vs original taxonomy on GORG queries. The ROC curve showing the mean of recall vs FP rate (i.e., plant queries matched to a DB) for both custom libraries with two settings. Kraken-II was run with confidence level $\in \{0.00, 0.02, 0.05\}$. Since genomes without associated taxonomy IDs are not processed by Kraken-II [250] such genomes were added to the map at the cellular organisms level manually. This addition increased both recall and false positive rates for GTDB but did not have a substantial impact on the TOL database.

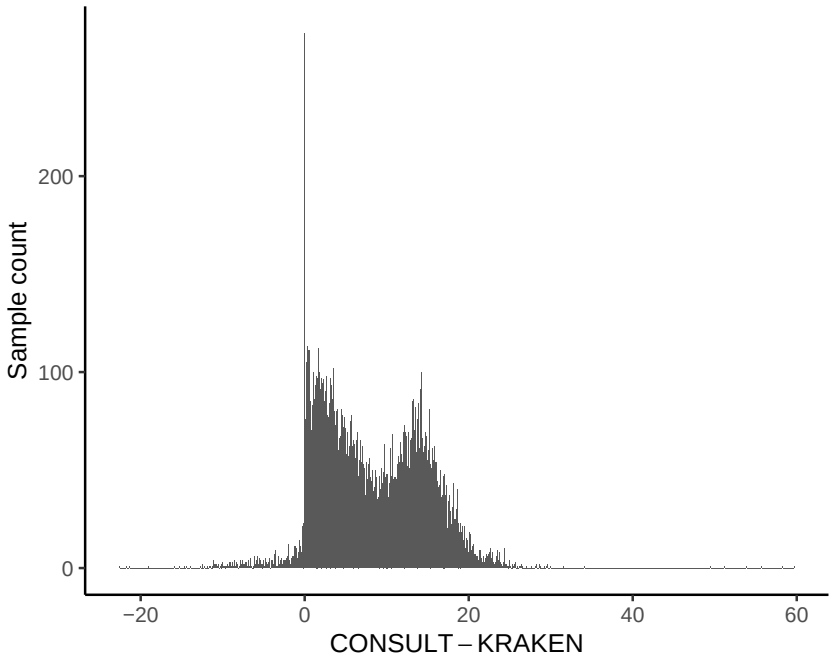


Figure B.18. Difference in percentage of classified reads between CONSULT and Kraken. The x-axis shows the difference between the percentage of reads in each genome that are classified using CONSULT and Kraken. Thus, positive difference indicates that CONSULT classifies more reads for a given sample than Kraken. The histogram is shown over all $> 12,000$ GORG samples queried against the GTDB reference library with default settings for CONSULT and $\alpha = 0.04$ for Kraken (bin size=0.1%).

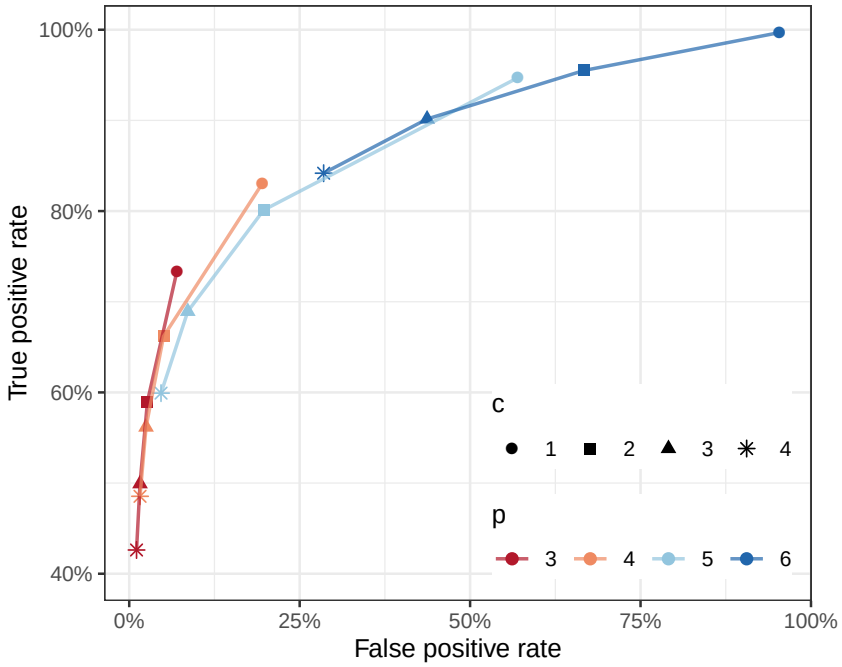
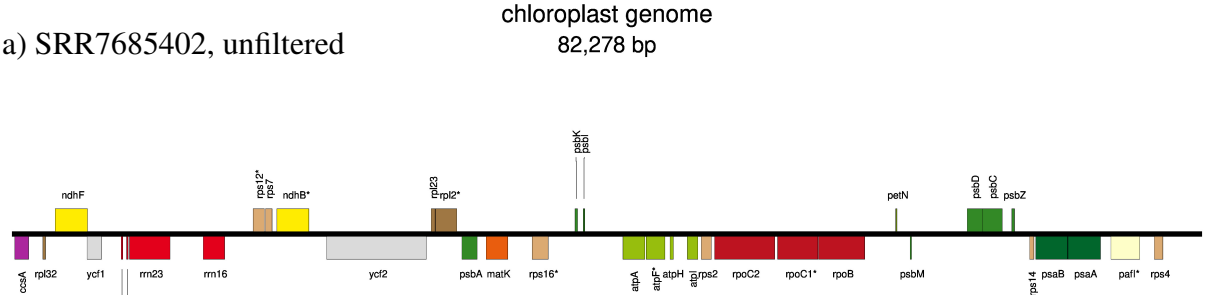
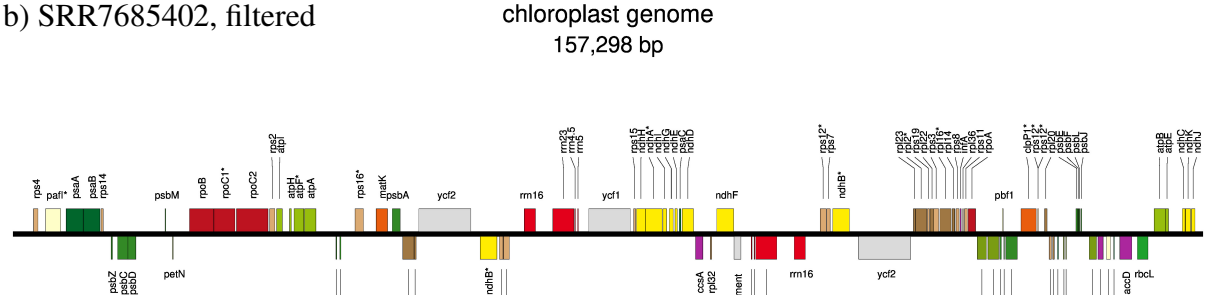


Figure B.19. Parameter exploration of CONSULT. Performance of CONSULT with GORG samples queried against GTDB at variable $p \in \{3, 4, 5, 6\}$ and $c \in \{1, 2, 3, 4\}$. CONSULT GTDB library was constructed with default settings.

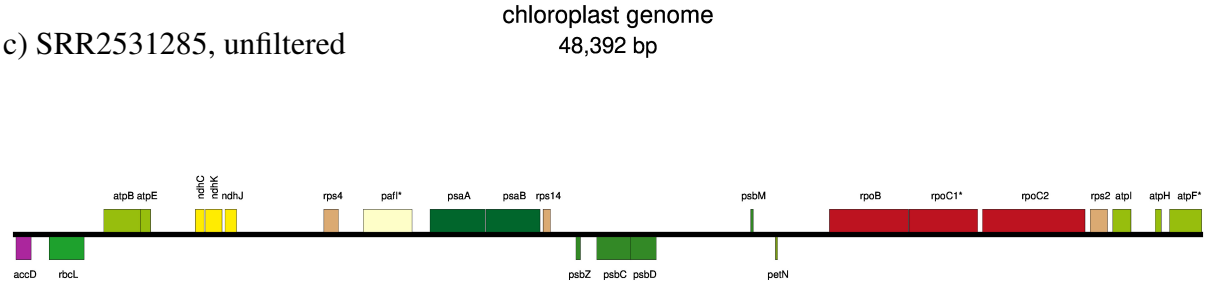
a) SRR7685402, unfiltered



b) SRR7685402, filtered



c) SRR2531285, unfiltered



d) SRR2531285, filtered

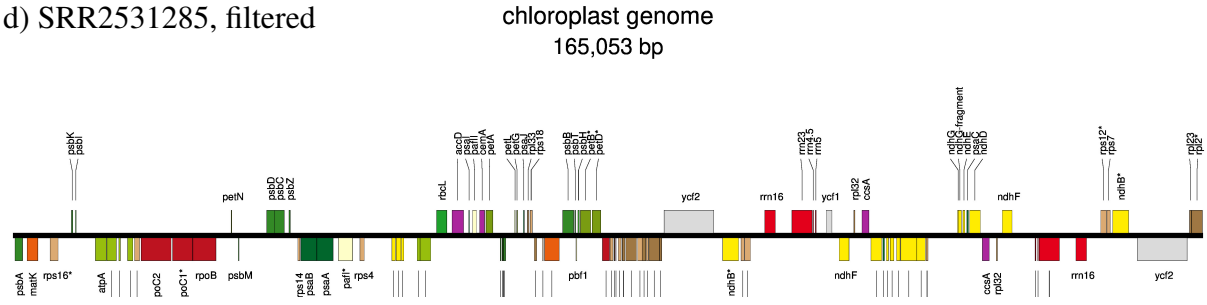


Figure B.20. Example annotations for generated chloroplast assemblies where filtering leads to complete assemblies. For two example samples, SRR7685402 (a,b) and SRR2531285 (c,d) with complete assemblies with CONSULT filtering but not without it, we show gene annotations performed using GeSeq [235]. Annotations are shown for (a,c) unfiltered and (b,d) filtered reads, demonstrating that the increased length of filtered assemblies is not spurious, as evident from the increased number of annotated genes.

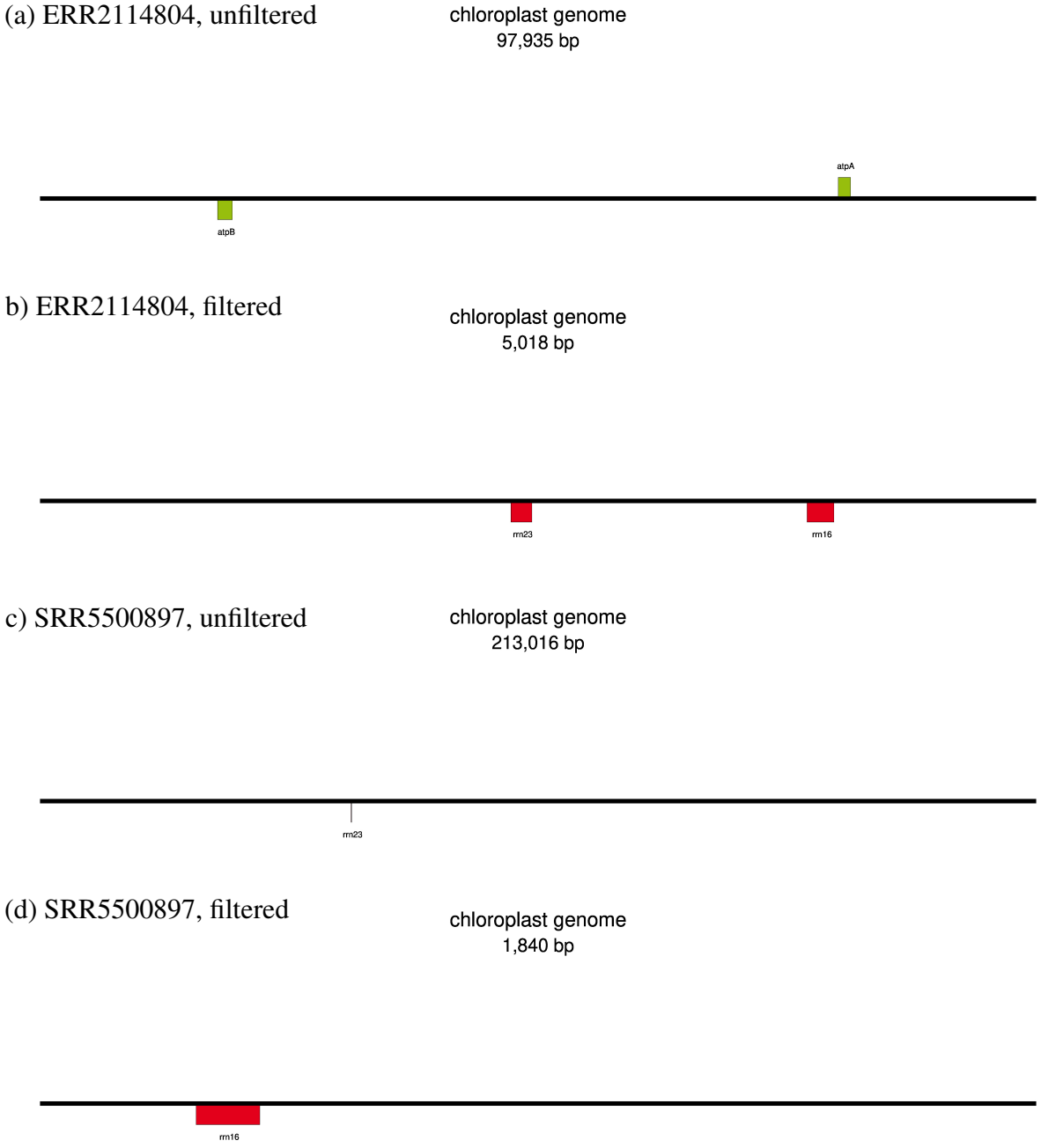


Figure B.21. Example annotations for generated chloroplast assemblies where filtering decreases assembly length. We show two example cases, ERR2114804 (a,b) and SRR5500897 (c,d), where CONSULT filtering reduces the length of the longest contig. Gene annotations are performed using GeSeq [235]. In both cases, the long (97kbp for ERR2114804 and 213kbp for SRR5500897) contigs produced by assembling the unfiltered reads are clearly spurious: In one case (a) only two genes that are supposed to be adjacent are found and are far apart; in the other case (c) no gene is found in the entire 213kbp. In both cases, filtering leads to very small contigs (b,d), which appropriately, have one or two genes annotated. Thus, filtering reduces spurious assembly.

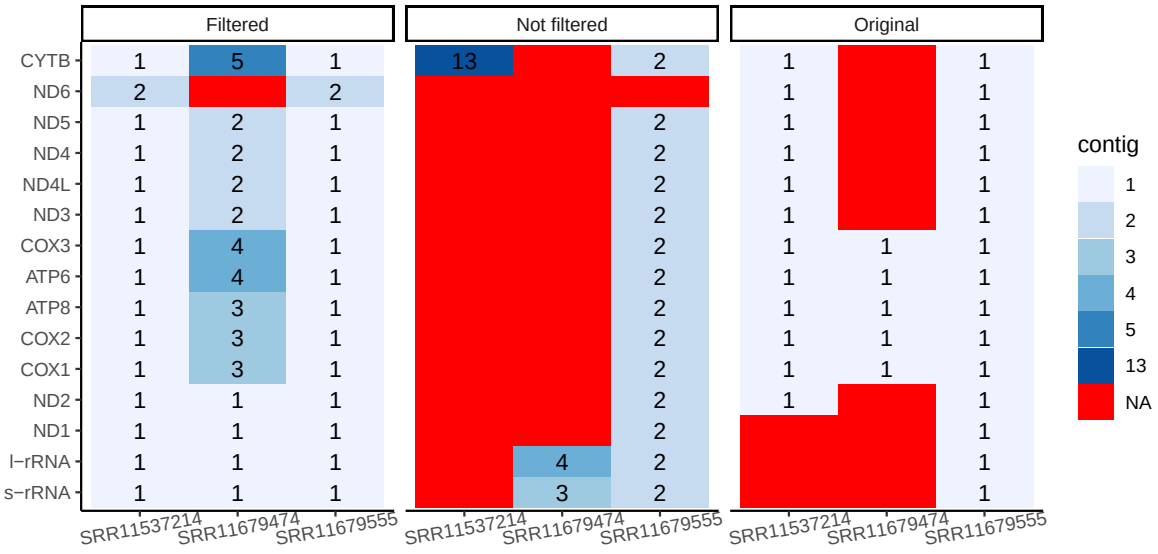


Figure B.22. Mitochondrial genes annotated in 20 largest contigs of the assemblies for filtered, unfiltered and original references. Plot demonstrates that genes missing from the largest contig of the assemblies produced by filtered reads can be found in smaller contigs of the the same assembly. Values indicate contig number in which gene was annotated.

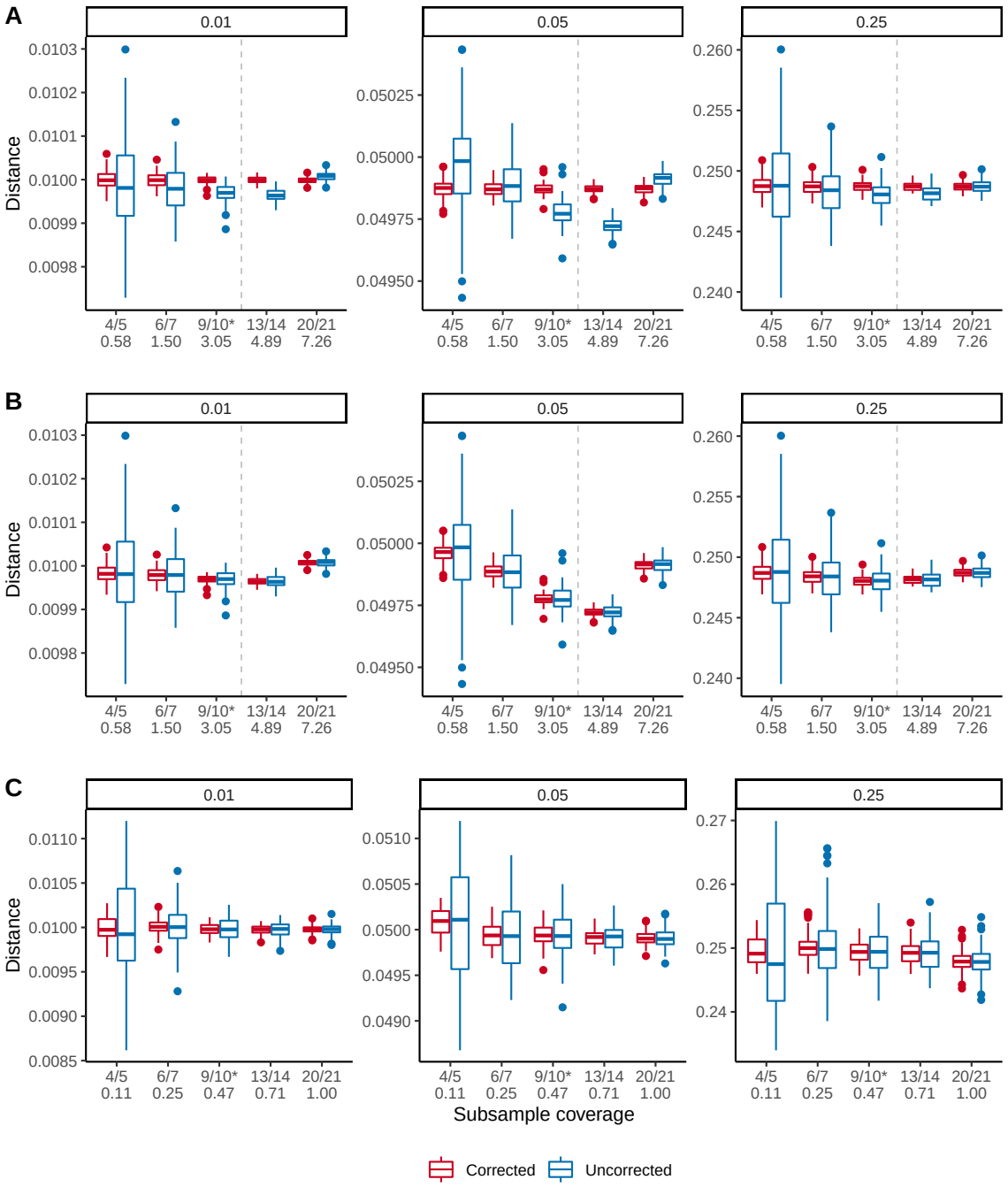


Figure B.23. Impact of distance correction versus α using samples at variable original coverage with two types of correction. A) Samples at 16x original coverage with main correction. B) Samples at 16x original coverage with consensus. C) 2x original coverage with consensus. Top portion of each subplots shows true distance. Y axis corresponds to estimated distance. Top part of x-axis label denotes α value; * signifies a suggested default. Bottom of the label is corresponding coverage. Values on top panels of the graphs correspond to the true distance between pairs of samples in simulated tree. Dashed line points to the coverage at which Skmer changes algorithm for distance computation.

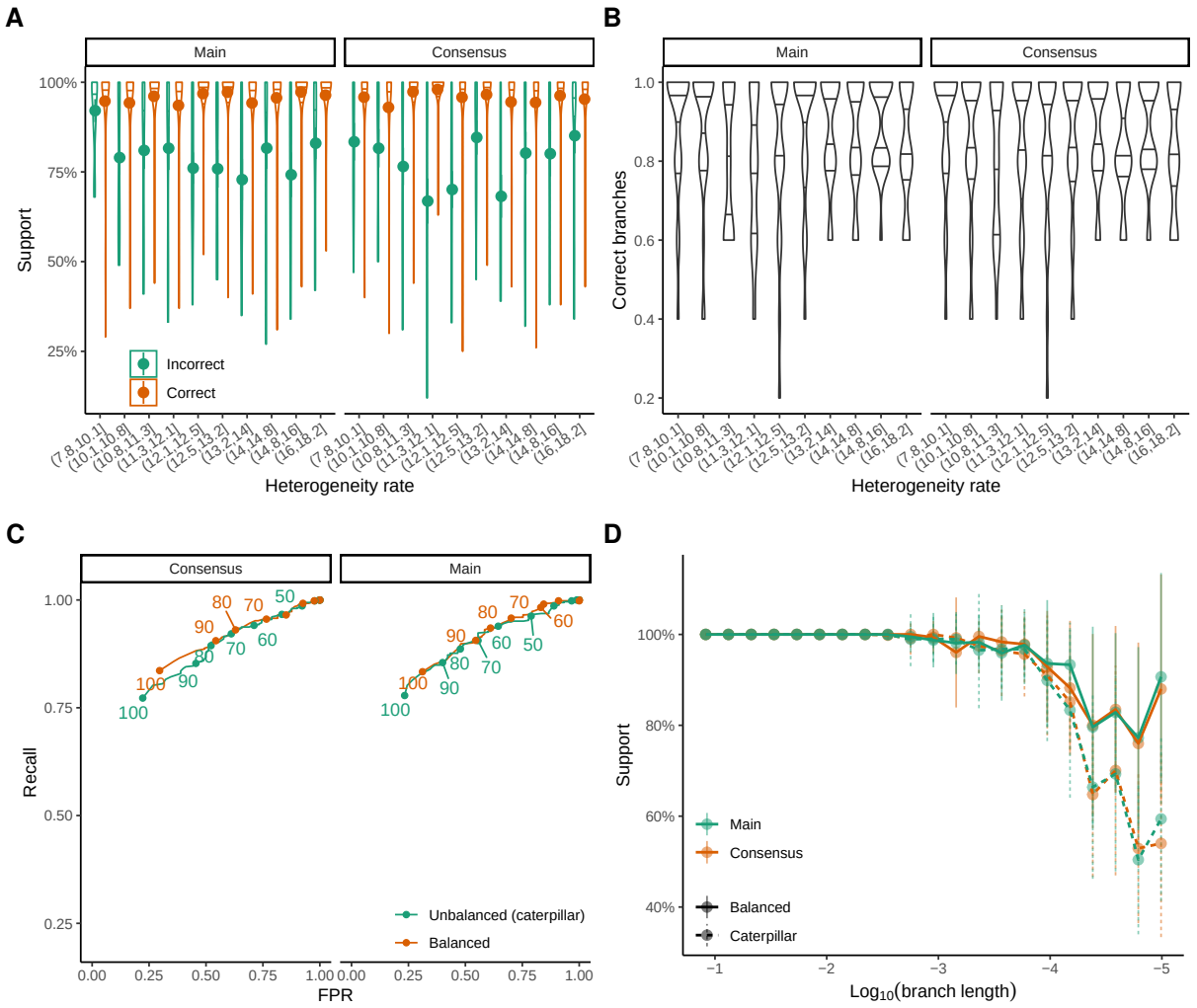


Figure B.24. Supplementary results for the 8-taxon experiment. A, B: Effect of heterogeneity rate on estimated support for eight taxa simulations. A) Plot shows heterogeneity rate vs estimated support with both types of correction for eight taxa trees. B) Diagram shows heterogeneity rate vs number of correct branches computed with both types of correction for eight taxa trees. Heterogeneity rate corresponds to α which is the shape parameter for the gamma distribution. See Section "Datasets" for exact definition. **C:** Receiver operating characteristic (ROC) for eight taxa simulations comparing balanced and unbalanced tree topologies. FPR denotes false positive rate. **D:** Effect of branch length on estimated support for the 8-taxon experiment. Diagram shows estimated branch length vs support computed for both types of correction.

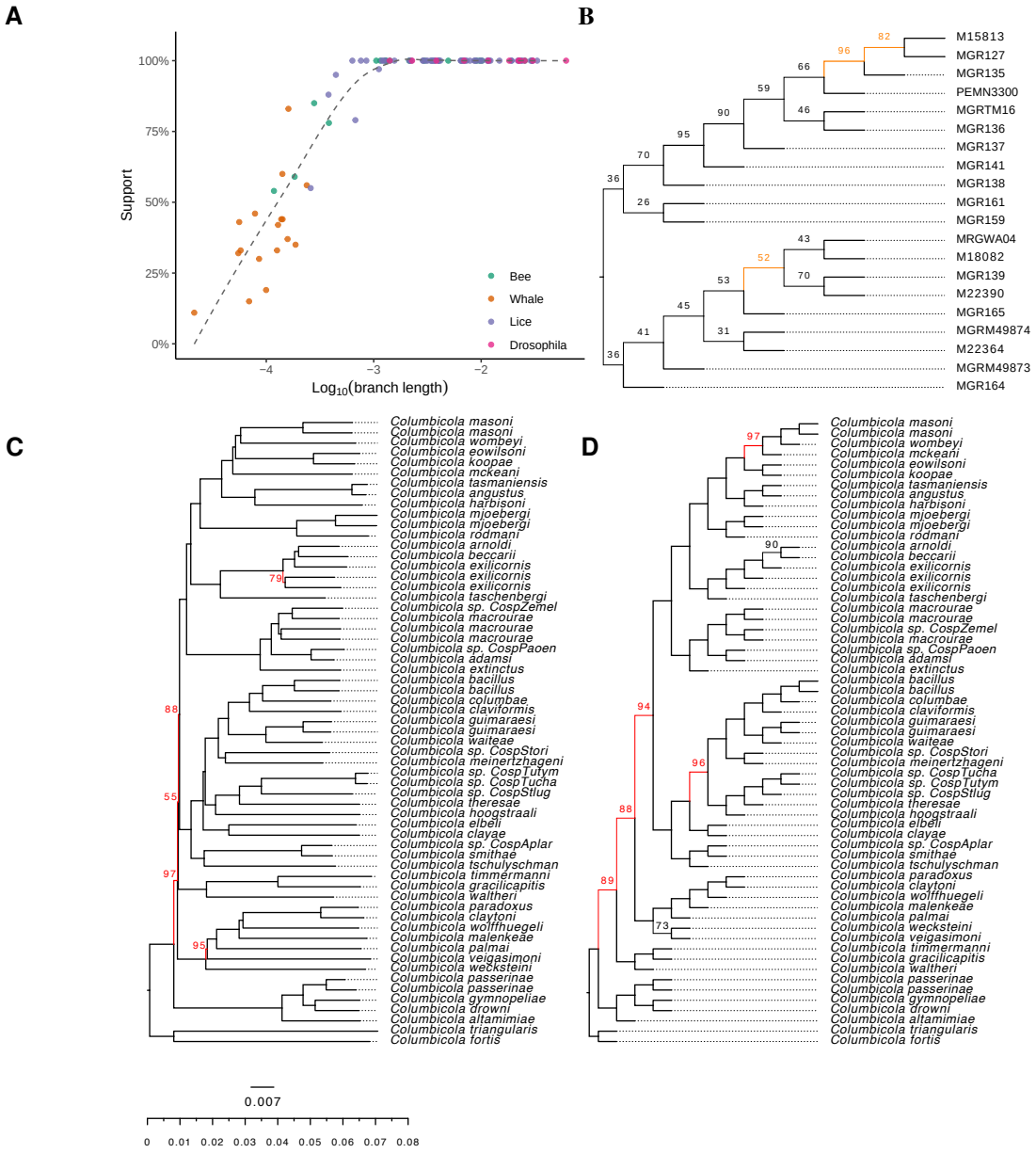


Figure B.25. Supplementary results for biological datasets. A: Comparison of branch length vs support for biological data. Only sequencing read samples are included. Support values are from phylogenies generated with main correction. **B:** Whale consensus phylogeny. Whale tree computed for multiple organisms of the same species of Gray's beaked whale *Mesoplodon grayi* using consensus. Thickness of the branch corresponds to magnitude of associated support. Tree topology is different from main tree shown in Figure 3.6B with only three shared branches highlighted in orange. None of the branches have 100% support which confirms lack of structure that we expect to see for the phylogeny built using samples of the same species. **C, D:** Constructed Lice phylogeny. Branches with no values assigned signify 100% support. Thickness of the branch corresponds to magnitude of associated support value. Red denotes conflicting branches as compared to ASTRAL reference phylogeny reported in original study. C) Main correction. D) Consensus.

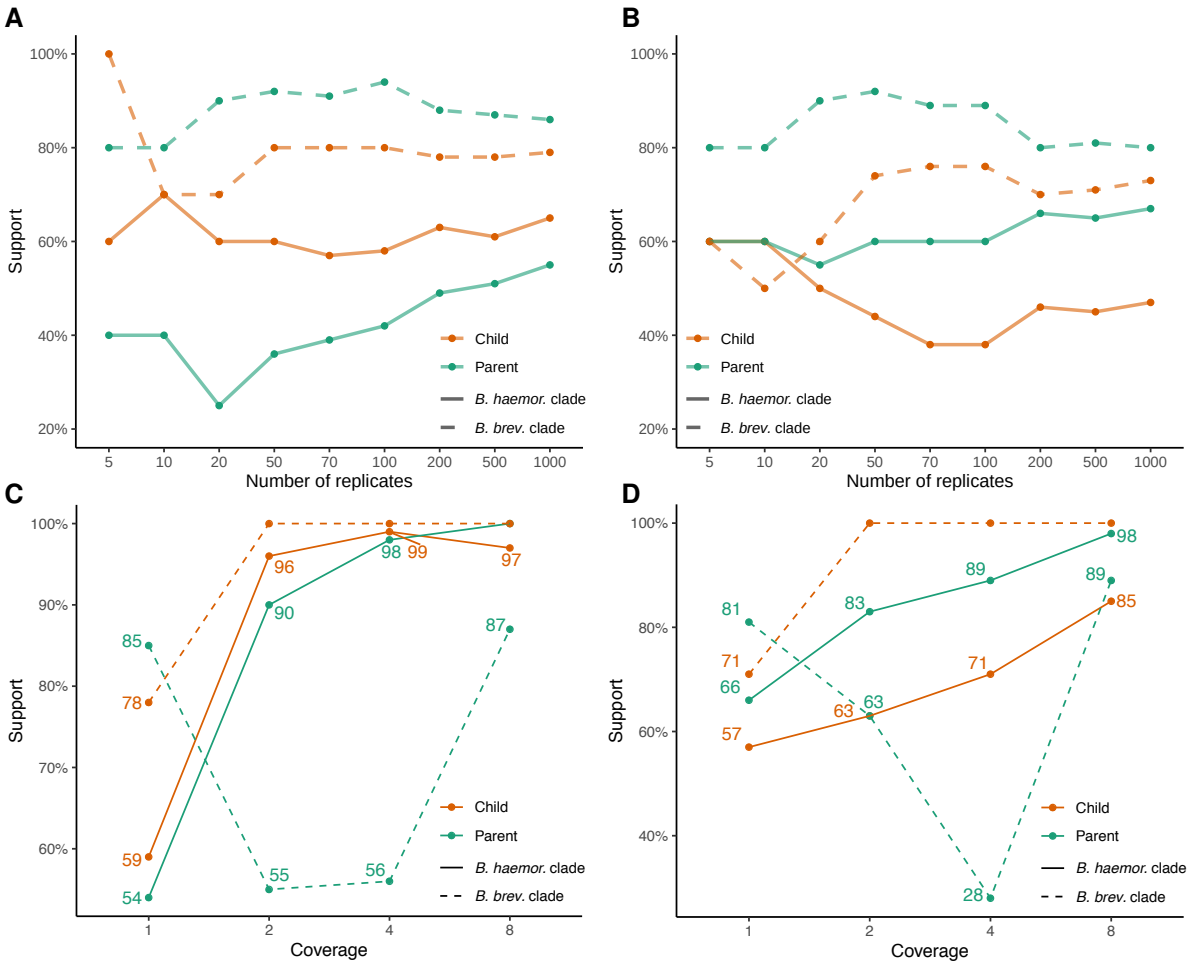


Figure B.26. Stability of analyses on biological dataset using both Main correction (A,C) and Consensus correction (B,D). **A, B:** Number of subsample replicates vs estimated support. Phylogenies are obtained from Bee sequencing reads simulated at 1x. **C, D:** Original sample coverage vs support values for Bee dataset. Values that are not displayed correspond to 100% support.

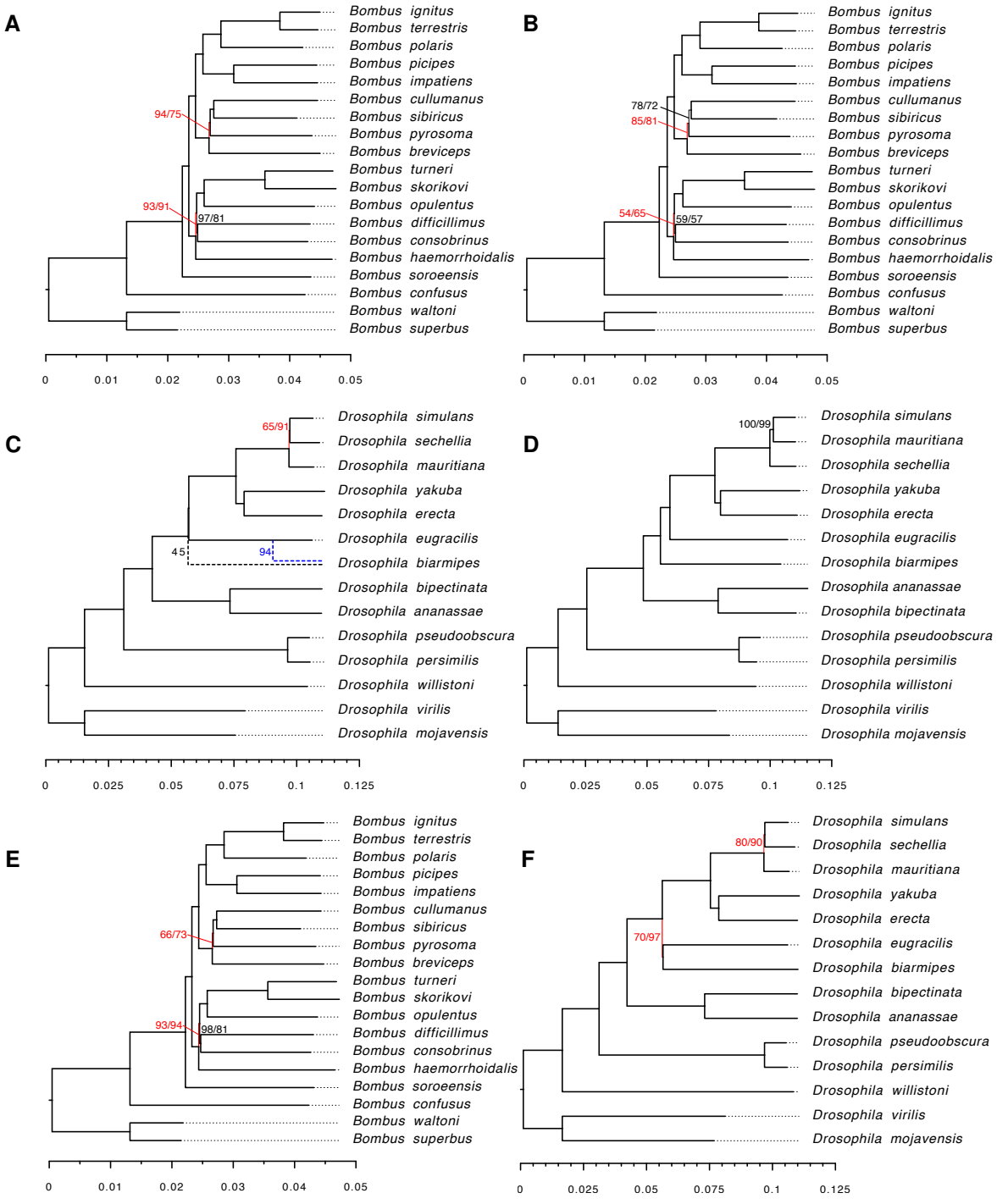


Figure B.27. Phylogenies constructed for bee (A, B, E) and Drosophila (C, D, F) biological datasets. A-D: Estimates under Felsenstein's 1981 (F81) model built from assemblies (A, C) or simulated reads at 1x (B), or real short-read SRAs (D). **E, F:** Phylogenies constructed after removing repeats from the assemblies. The branch thickness: the magnitude of associated support. Red: controversial branches as compared to reference phylogenies. Blue: alternative topology between main and consensus trees. Branches with no values assigned: 100% support. Left/Right values:support for tree corrected with main/consensus estimates.

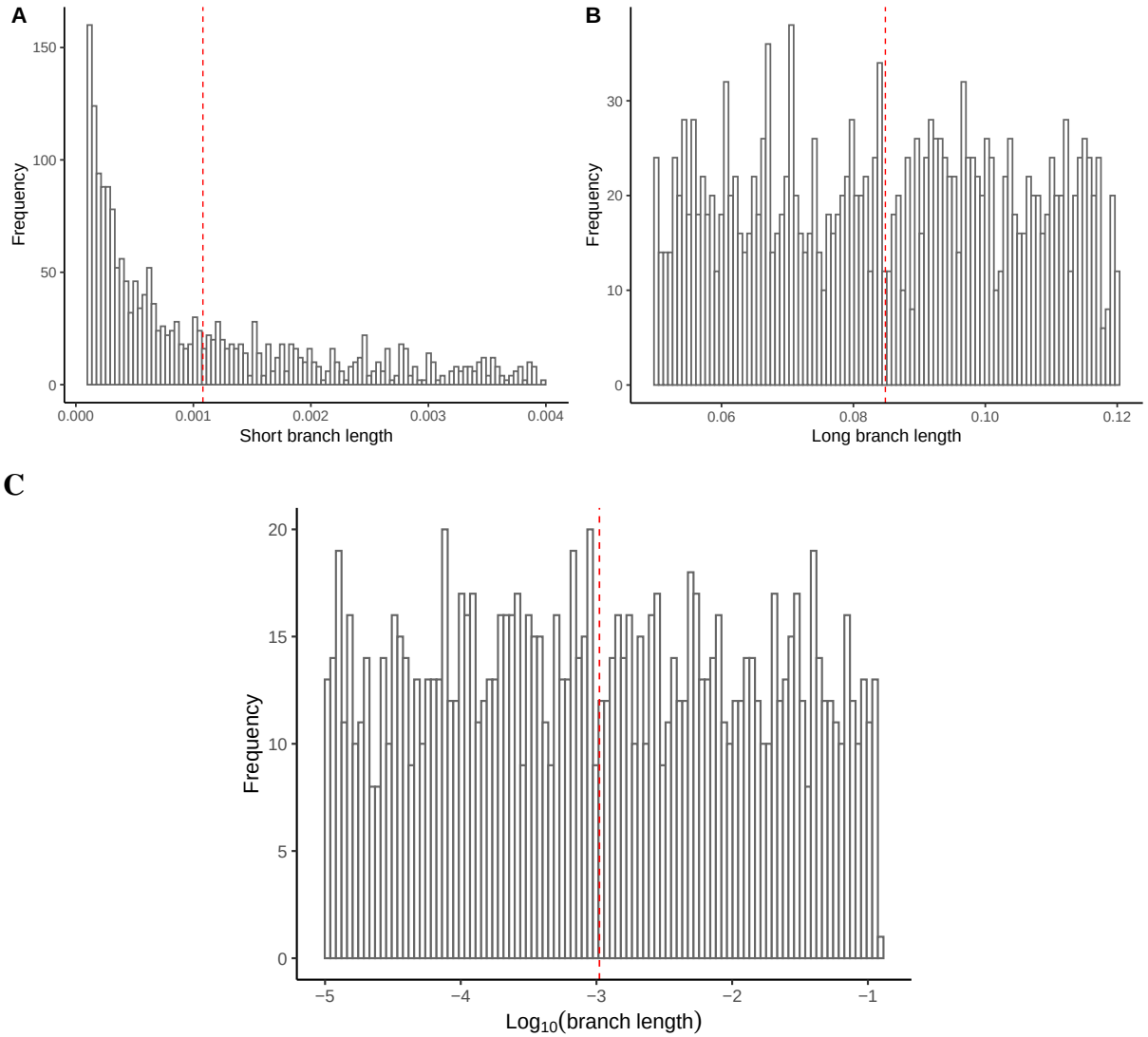


Figure B.28. Distribution of branch length values for Felsenstein tree topologies (A, B) and trees generated for eight taxa simulation experiment (C). A) Histogram of short branches. B) Histogram of long branches. Red line denotes mean branch length. C) Distribution of branch length values for eight taxa tree topologies. Red line corresponds to mean branch length.

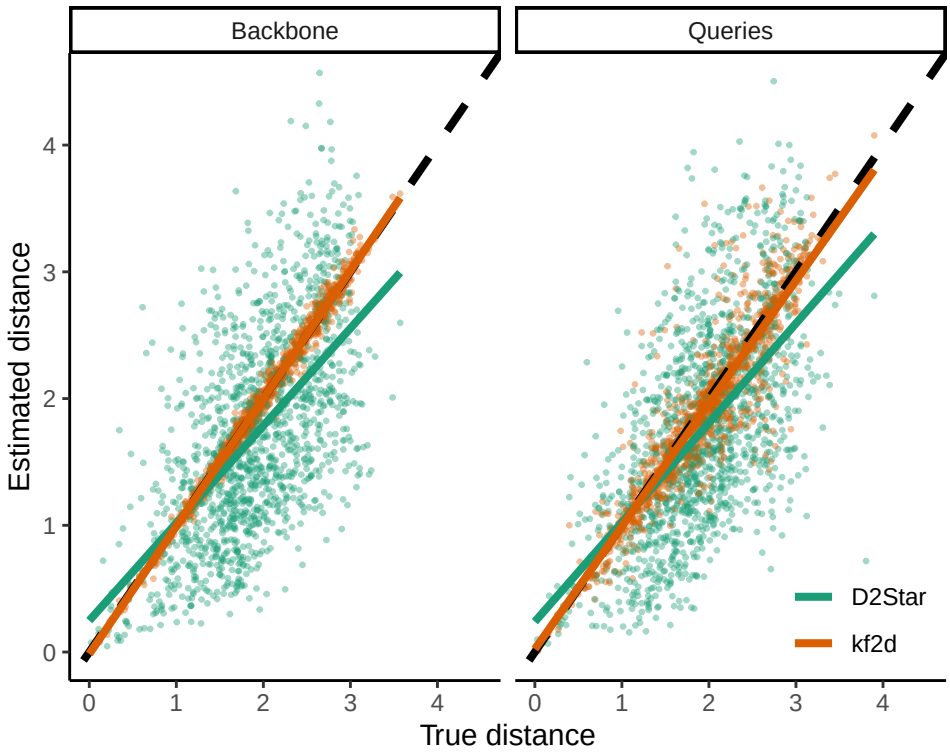


Figure B.29. Distance comparison between kf2d and D2Star distance metric from CAFE.

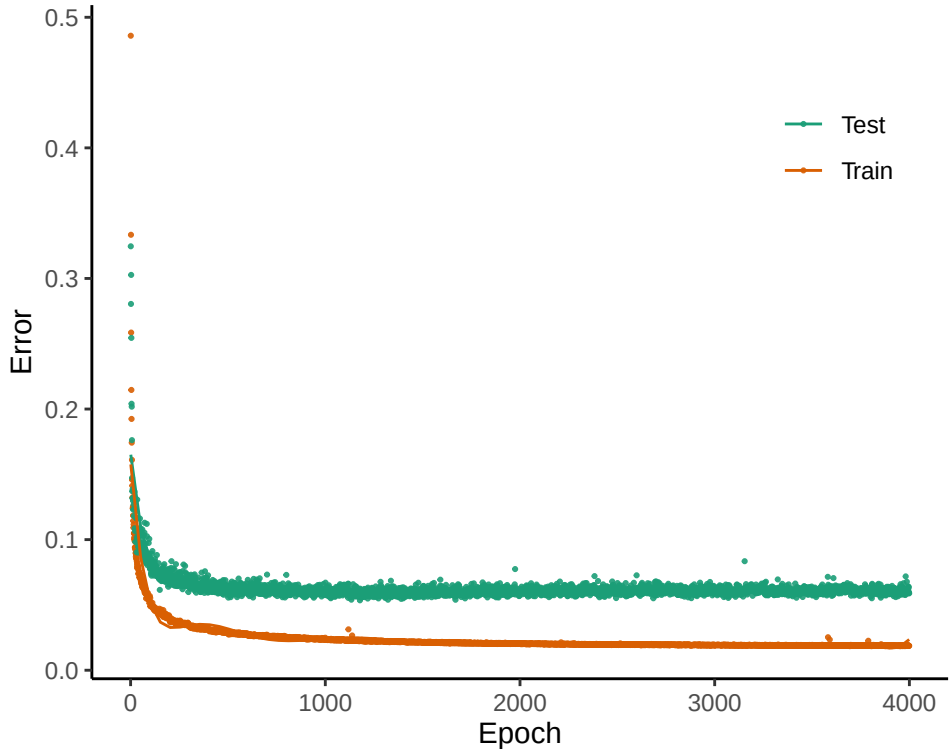


Figure B.30. Training progression.

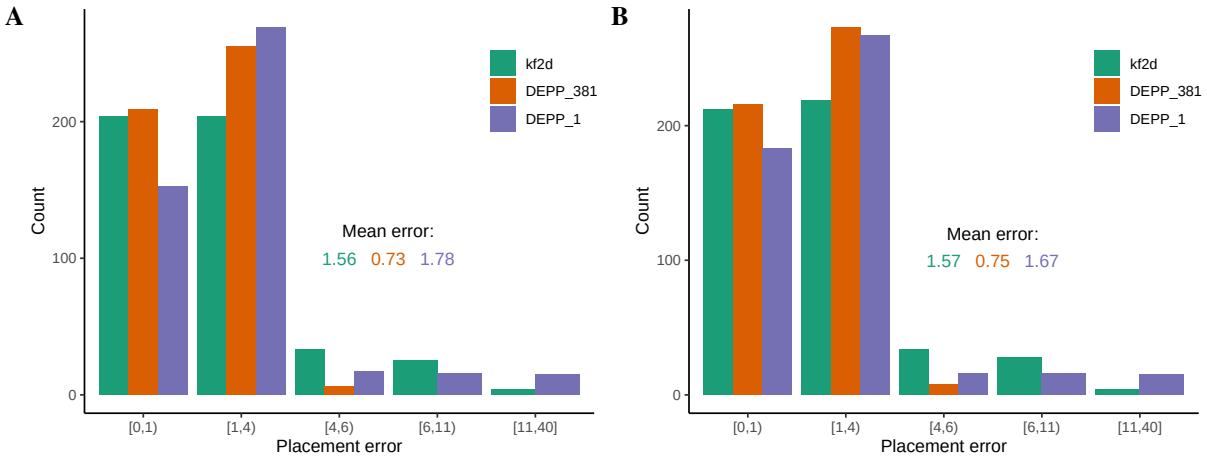


Figure B.31. Distribution of placement error for kf2d vs DEPP. kf2d was compared to DEPP with 1 gene and all 381 genes. A) Randomly selected gene was used. 30 species did not have gene present and were excluded from comparison. B) Best marker gene was used. 3 species that didn't contain the gene were excluded from comparison.

Appendix C

Supplementary Tables

Table C.1. Bin assignment based on Mash distances.

Bin	Number of genomes in bin	Minimum distance encountered within bin	Maximum distance encountered within bin	Mean distance for all species within bin
0	10	0.000	0.000	0.000
(0 - 1]	17	0.000	0.009	0.004
(1 - 2]	10	0.010	0.018	0.013
(2 - 3]	8	0.020	0.030	0.025
(3 - 5]	8	0.031	0.049	0.038
(5 - 10]	8	0.058	0.099	0.084
(10 - 15]	11	0.109	0.146	0.124
(15 - 20]	6	0.157	0.197	0.176
(20 - 25]	11	0.211	0.244	0.229
>25	21	0.263	1.000	0.729

Table C.2. Plant species used as query sequences. Plants were selected to represent a wide range of genome sizes.

Species name	Genome size (M)
Arabidopsis thaliana	119.167
Arabidopsis lyrata	202.97
Carya illinoinensis	649.75
Carya cathayensis	721.33
Nicotiana sylvestris	2221.99
Zea mays	2182.61
Oryza sativa	382.63
Coffee arabica	1094.45
Prunus persica	212.77
Bathycoccus prasinos	15.07

Table C.3. Contamination level (c_l) with the corresponding number of unique k-mers for base (*D. simulans w501*) and contaminant (bacteria) in mixture samples.

Bin	Contamination level (% reads	c_l (%): Unique contaminant k-mers in sample (%)	Unique base k-mers in sample (%)	Common k-mers between base and contaminant (%)
0-5	60	53.728625	46.271364	0.000011
0-5	50	46.000931	53.999065	0.000004
0-5	40	38.455505	61.544490	0.000005
0-5	30	30.730685	69.269315	0.000000
0-5	25	26.630857	73.369142	0.000001
0-5	20	22.286244	77.713750	0.000007
0-5	15	17.595401	82.404592	0.000007
0-5	10	12.408705	87.591295	0.000000
0-5	5	6.630852	93.369148	0.000000
0-5	2	2.769620	97.230380	0.000000
0-5	1	1.406449	98.593551	0.000000
0-5	0	0.000000	100.000000	0.000000
0-0	25	24.707830	75.292146	0.000024
0-0	20	20.899131	79.100838	0.000031
0-0	15	16.686621	83.313353	0.000026
0-0	10	11.968806	88.031170	0.000023
0-0	5	6.494518	93.505472	0.000010
0-0	2	2.742933	97.257067	0.000000
0-0	1	1.395767	98.604233	0.000000
0-0	0	0.000000	100.000000	0.000000
5-15	25	24.797418	75.202570	0.000012
5-15	20	20.977979	79.021978	0.000043
5-15	15	16.738171	83.261791	0.000038
5-15	10	12.010540	87.989451	0.000010
5-15	5	6.502650	93.497350	0.000000
5-15	2	2.745543	97.254457	0.000000
5-15	1	1.402008	98.597992	0.000000
5-15	0	0.000000	100.000000	0.000000
15-25	25	24.514304	75.485614	0.000082
15-25	20	20.760153	79.239792	0.000055
15-25	15	16.620725	83.379207	0.000068
15-25	10	11.916805	88.083179	0.000016
15-25	5	6.484925	93.515047	0.000028
15-25	2	2.738899	97.261078	0.000023
15-25	1	1.398807	98.601185	0.000009
15-25	0	0.000000	100.000000	0.000000

Table C.4. Actual amount of non-unique *k*-mers (%) in contaminant reads used to generate simulated Drosophila skims with *H* overlap.

Expected overlap (%)	0.00	1.00	5.00	10.00	25.00	50.00
Actual *k*-mer overlap (%)	11.12	11.54	13.31	15.83	23.82	40.78

Table C.5. The effect of filtering on the quality of phylogenies inferred from genomic skims.

Skim size (Mb)	**Filtering status**	**RF**	**wRF**	**total FME**
100	Pre-filtering	4	0.147	1.257
100	Post-filtering	2	0.043	0.381
200	Pre-filtering	4	0.141	1.140
200	Post-filtering	2	0.037	0.371

Table C.6. Species used as query sequences for assessing Kraken-II running time. Genomes were arbitrarily selected but represent a diverse set of both eukaryotic and prokaryotic species.

Species name	**Genome size (M)**
Arabidopsis thaliana	119.167
Anopheles gambiae	250.715
Drosophila melanogaster	137.688
Capnocytophaga sputigena	2.998
Euryarchaeota archaeon	1.504

Table C.7. List of accession numbers and URLs for Drosophila species used in contamination simulation experiment.

Species	**Assembly accession**	**URL**
Drosophila simulans w501	GCF_000754195.2	https://www.ncbi.nlm.nih.gov/assembly/GCF_000754195.2
Drosophila simulans WXD1	GCA_004382185.1	https://www.ncbi.nlm.nih.gov/assembly/GCA_004382185.1
Drosophila sechellia	GCF_000005215.3	https://www.ncbi.nlm.nih.gov/assembly/GCF_000005215.3
Drosophila yakuba	GCF_000005975.2	https://www.ncbi.nlm.nih.gov/assembly/GCF_000005975.2

Table C.8. List of accession numbers and URLs for plant species added to query set.

Species	Assembly accession	URL
Arabidopsis thaliana	GCF_000001735.4	https://www.ncbi.nlm.nih.gov/assembly/GCF_000001735.4
Arabidopsis lyrata	GCF_000004255.2	https://www.ncbi.nlm.nih.gov/assembly/GCF_000004255.2
Carya illinoinensis	Cil.genome.fa.gz	ftp://parrot.genomics.cn/gigadb/pub/10.5524/100001_101000/100571/Cil.genome.fa.gz
Carya cathayensis	Cca.genome.fa.gz	ftp://parrot.genomics.cn/gigadb/pub/10.5524/100001_101000/100571/Cca.genome.fa.gz
Nicotiana sylvestris	GCF_000393655.1	https://www.ncbi.nlm.nih.gov/assembly/GCF_000393655.1
Zea mays	GCF_000005005.2	https://www.ncbi.nlm.nih.gov/assembly/GCF_000005005.2
Oryza sativa	GCF_001433935.1	https://www.ncbi.nlm.nih.gov/assembly/GCF_001433935.1
Coffee arabica	GCF_003713225.1	https://www.ncbi.nlm.nih.gov/assembly/GCF_003713225.1
Prunus persica	GCF_000346465.2	https://www.ncbi.nlm.nih.gov/assembly/GCF_000346465.2
Bathycoccus prasinos	GCF_002220235.1	https://www.ncbi.nlm.nih.gov/assembly/GCF_002220235.1

Table C.9. k-mer counts in CONSULT database constructed with variable t. Databases were built with default settings using genomes from Bacterial and Archaeal Kraken. Data demonstrate that larger tag facilitates more uniform distribution of signatures within lookup table which leads to better utilization of signature array space and ultimately allows to populate database more efficiently.

Tag (bits)	**k-mers count in database (billion)**	**Count of fully populated rows(%), l = 0**	**Count of fully populated rows(%), l = 1**
0	5.838554966	0.354	0.364
2	6.210280798	0.330	0.339

Table C.10. Experiments on settings of CONSULT, namely number of hash bits (h), number of hash functions (l), number of k-mers per has-value (b), and tag size in bits (t). All tests were performed on TOL query set that was searched against TOL database. To avoid biases we fixed positions of the k-mer bits that were selected to generate signatures. For each setting, we show size of the library on the disk, recall for TOL queries searched against TOL database, and percentage of all k-mers among 14.026601864 billion minimized k-mers that fit the library. Parameter selection was mainly driven by our desire to maximize weighted recall and intention to keep database footprint under 128G.

h	l	b	t	Disk (GB)†	Recall	% k-mer	Notes
Changing l and b							
15	1	7	2	78	0.672	0.474	Reduced recall compared to default
15	2	7	2	114	0.696	0.570	Default
15	3	7	2	135	0.705	0.570	Memory exceeds 128GB
15	1	10	2	95	0.680	0.570	Reduced recall compared to default
15	2	10	2	120	0.700	0.570	Memory exceeds 128GB†
15	3	10	2	148	0.707	0.570	Memory exceeds 128GB
Changing b							
15	2	4	2	76	0.678	0.394	Reduced recall
15	2	7	2	114	0.696	0.570	Default
15	2	10	2	120	0.700	0.570	Increased memory with no benefit
15	2	13	2	125	0.702	0.570	Increased memory with no benefit
15	2	16	2	128	0.704	0.570	Increased memory with no benefit
Changing h							
15	2	7	2	114	0.696	0.570	Default
14	2	16	2	73	0.689	0.379	Reduced recall and k-mer representation

†: Disk space used to keep the library, as computed using the `du` command without the `--apparent-size` option. Note that the actual space used (in RAM) is slightly higher and is closer to what `du --apparent-size -h` would produce. For example, for the default settings, the memory usage in RAM is 120GB, whereas `du -h` produces 114GB on and `du --apparent-size -h` produces 120GB.

Table C.11. Bin assignment based on Mash [167] distances.

Bin	Number of genomes in bin	Minimum distance encountered within bin	Maximum distance encountered within bin	Mean distance for all species within bin
0	10	0.000	0.000	0.000
(0 - 5]	43	0.000	0.049	0.016
(5 - 15]	19	0.058	0.146	0.107
(15 - 25]	17	0.157	0.244	0.210
>25	21	0.263	1.000	0.729

Table C.12. List of plant species added to TOL query set.

Species	Genome size (M)	Assembly accession	URL
Arabidopsis thaliana	119.167	GCF_000001735.4	https://www.ncbi.nlm.nih.gov/assembly/GCF_000001735.4
Arabidopsis lyrata	202.97	GCF_000004255.2	https://www.ncbi.nlm.nih.gov/assembly/GCF_000004255.2
Carya illinoinensis	649.75	GCA_011037805.1	ftp://parrot.genomics.cn/gigadb/pub/10.5524/100001_101000/100571/Cil.genome.fa.gz
Carya cathayensis	721.33	GCA_011037825.1	ftp://parrot.genomics.cn/gigadb/pub/10.5524/100001_101000/100571/Cca.genome.fa.gz
Nicotiana sylvestris	2221.99	GCF_000393655.1	https://www.ncbi.nlm.nih.gov/assembly/GCF_000393655.1
Zea mays	2182.61	GCF_000005005.2	https://www.ncbi.nlm.nih.gov/assembly/GCF_000005005.2
Oryza sativa	382.63	GCF_001433935.1	https://www.ncbi.nlm.nih.gov/assembly/GCF_001433935.1
Coffee arabica	1094.45	GCF_003713225.1	https://www.ncbi.nlm.nih.gov/assembly/GCF_003713225.1
Prunus persica	212.77	GCF_000346465.2	https://www.ncbi.nlm.nih.gov/assembly/GCF_000346465.2
Bathycoccus prasinos	15.07	GCF_002220235.1	https://www.ncbi.nlm.nih.gov/assembly/GCF_002220235.1

Table C.13. List of SRRs and URLs for Drosophila species used in real data experiment [155] downloaded from NCBI (PRJNA427774). Note that data from *D. grimshawi* used in the original publication was not provided in this bioproject, and thus, we do not have this species included in our analyses.

Species	Run	URL
Drosophila bipectinata	SRR6425989	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425989
Drosophila erecta	SRR6425990	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425990
Drosophila ananassae	SRR6425991	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425991
Drosophila biarmipes	SRR6425992	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425992
Drosophila mauritiana	SRR6425993	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425993
Drosophila eugracilis	SRR6425995	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425995
Drosophila mojavensis	SRR6425997	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425997
Drosophila persimilis	SRR6425998	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425998
Drosophila simulans	SRR6425999	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6425999
Drosophila virilis	SRR6426000	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6426000
Drosophila pseudoobscura	SRR6426001	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6426001
Drosophila sechellia	SRR6426002	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6426002
Drosophila willistoni	SRR6426003	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6426003
Drosophila yakuba	SRR6426004	https://www.ncbi.nlm.nih.gov/sra/?term=SRR6426004

Table C.14. Samples from Denmark sequencing project [141] used in mitochondrial assembly experiment. Unassembled subset (n=18) represents species that failed mitochondrial assembly in Denmark study. Poor quality group (n=6) was assembled with short contigs of 3–10.5 kbp. Good quality assemblies (n=18) were randomly selected samples with relatively long contig length. Genus count and species count represent count of samples of the same genus or species present in RefSeq release used to construct mitochondrial CONSULT library.

Run	Species	Contig length (bp)	mtDNA (%)	GenBank ID	Available genus	Available species
Unassembled samples						
SRR12432370	*Apodemus sylvaticus*	N/A	N/A	N/A	6	1
SRR12432443	*Argyropelecus olfersii*	N/A	N/A	N/A	0	0
SRR12432479	*Branta leucopsis*	N/A	N/A	N/A	1	0
SRR12432364	*Chelon ramada*	N/A	N/A	N/A	1	0
SRR12432468	*Corvus cornix*	N/A	N/A	N/A	12	1
SRR12432397	*Cygnus olor*	N/A	N/A	N/A	4	1
SRR12432591	*Emberiza citrinella*	N/A	N/A	N/A	12	0
SRR12432369	*Eptesicus serotinus*	N/A	N/A	N/A	0	0
SRR12432332	*Gallinago gallinago*	N/A	N/A	N/A	1	0
SRR12432390	*Ichthyosaura alpestris*	N/A	N/A	N/A	0	0
SRR12432365	*Maurolicus muelleri*	N/A	N/A	N/A	0	0
SRR12432368	*Megaptera novaeangliae*	N/A	N/A	N/A	0	1
SRR12432371	*Microtus arvalis*	N/A	N/A	N/A	7	1
SRR12432366	*Myotis mystacinus*	N/A	N/A	N/A	29	0
SRR12432391	*Pipistrellus pygmaeus*	N/A	N/A	N/A	2	0
SRR12432352	*Raniceps raninus*	N/A	N/A	N/A	0	0
SRR12432355	*Sicista betulina*	N/A	N/A	N/A	1	0
SRR12432363	*Trachipterus arcticus*	N/A	N/A	N/A	1	0
Poor quality assemblies						
SRR11679515	*Tamias sibiricus*	3746	0.0049	MT410867	7	1
SRR11679527	*Lullula arborea*	3018	0.0114	MT410892	0	0
SRR11679474	*Callionymus reticulatus*	4644	0.0138	MT410925	2	0
SRR11537214	*Asio flammeus*	10466	0.0279	MN122899	1	1
SRR11537185	*Calidris alpina*	9562	0.0393	MN122893	3	0
SRR11537153	*Lycodes vahlii*	8147	0.2552	MT410895	5	0
Good quality assemblies (positive control)						
SRR11679529	*Luscinia luscinia*	14276	0.0164	MT410894	2	0
SRR11679531	*Phasianus colchicus*	16773	0.0219	MT410882	1	1
SRR11537201	*Merlangius merlangus*	16660	0.0298	MN122861	0	1
SRR11537177	*Merops apiaster*	13884	0.0406	MN122929	1	0
SRR11679575	*Oncorhynchus mykiss*	16759	0.0820	MT410879	11	1

(continued on next page)

Table C.14. Samples from Denmark sequencing project [141] used in mitochondrial assembly experiment. Unassembled subset (n=18) represents species that failed mitochondrial assembly in Denmark study. Poor quality group (n=6) was assembled with short contigs of 3–10.5 kbp. Good quality assemblies (n=18) were randomly selected samples with relatively long contig length. Genus count and species count represent count of samples of the same genus or species present in RefSeq release used to construct mitochondrial CONSULT library.

(continued from previous page)

Run	Species	Contig length (bp)	mtDNA (%)	GenBank ID	Available genus	Available species
SRR11537188	*Buteo buteo*	13998	0.1356	MN122916	2	1
SRR11679510	*Physeter catodon*	16491	0.3210	MT410874	0	1
SRR11679507	*Taurulus bubalis*	16854	0.4018	MT410868	0	0
SRR11679528	*Arvicola amphibius*	16356	0.5788	MN122828	0	0
SRR11679539	*Sciurus vulgaris*	16511	0.8176	MN122875	11	1
SRR11537162	*Arnoglossus laterna*	15958	1.0269	MN122822	2	0
SRR11679545	*Vipera berus*	12733	1.2580	MN122848	0	1
SRR11537150	*Centrolabrus exoletus*	16494	1.6414	MT410926	0	0
SRR11679513	*Lagenorhynchus albirostris*	16393	2.0310	MT410901	2	1
SRR11679516	*Delphinus delphis*	16386	2.8277	MT410915	1	1
SRR11537206	*Vipera berus*	12733	4.0683	MN122824	0	1
SRR11537189	*Milvus milvus*	17883	4.7909	MN122837	1	0
SRR11679555	*Chroicocephalus ridibundus*	16768	5.6225	MN122820	1	1

Table C.15. Subset of TOL samples used in running time analysis. Samples were selected randomly.

Genome	Assembly accession	Bioproject	Species
G000019605	GCF_000019605.1	PRJNA224116	*Candidatus Korarchaeum cryptofilum*
G000231015	GCF_000231015.2	PRJNA224116	*Desulfurococcus fermentans*
G000816105	GCF_000816105.1	PRJNA224116	*Thermococcus guaymasensis*
G001510295	GCA_001510295.1	PRJNA279271	*Thaumarchaeota archaeon*
G001674955	GCA_001674955.1	PRJNA289040	*Candidatus Nitrosopumilus sp.*
G000007185	GCA_000007185.1	PRJNA294	*Methanopyrus kandleri*
G000204585	GCA_000204585.1	PRJNA52465	*Candidatus Nitrosoarchaeum limnia*
G000421185	GCF_000421185.1	PRJNA224116	*Ornithinimicrobium pekingense*
G001315825	GCA_001315825.1	PRJDB782	*Metallosphaera hakonensis*
G001577775	GCF_001577775.1	PRJNA224116	*Pyrococcus kukulkanii*
G001940645	GCA_001940645.1	PRJNA319486	*Candidatus Heimdallarchaeota archaeon*
G000011125	GCA_000011125.1	PRJNA211	*Aeropyrum pernix*
G000221185	GCF_000221185.1	PRJNA224116	*Thermococcus sp.*
G000770635	GCF_000770635.1	PRJNA224116	*Pontibacillus yanchengensis*
G001316265	GCF_001316265.1	PRJNA224116	*Vulcanisaeta sp.*
G001628455	GCA_001628455.1	PRJNA289734	*Marine group II euryarchaeote*
G001940655	GCA_001940655.1	PRJNA288027	*Candidatus Lokiarchaeota archaeon*
G000022365	GCF_000022365.1	PRJNA224116	*Thermococcus gammatolerans*
G000307305	GCF_000307305.1	PRJNA224116	*Enterobacteriaceae bacterium*
G000955905	GCF_000955905.1	PRJNA224116	*Candidatus Nitrosotenuis cloacae*
G001515215	GCA_001515215.1	PRJNA298487	*Hadesarchaea archaeon*
G001679155	GCF_001679155.1	PRJNA224116	*Moraxella nonliquefaciens*
G000145295	GCF_000145295.1	PRJNA224116	*Methanothermobacter marburgensis*
G000375685	GCA_000375685.1	PRJNA165545	*crenarchaeote SCGC*
G001189275	GCA_001189275.1	PRJNA258558	*Pyrobaculum sp.*
G001560165	GCF_001560165.1	PRJNA224116	*Sulfolobus acidocaldarius*
G001919175	GCA_001919175.1	PRJNA297196	*Crenarchaeota archaeon*
Cca.genome.fa.gz	GCA_011037825.1	PRJNA435846	*Carya cathayensis*
Cil.genome.fa.gz	GCA_011037805.1	PRJNA435846	*Carya illinoinensis*
TAIR10.1	GCF_000001735.4	PRJNA10719	*Arabidopsis thaliana*

Table C.16. Drosophila species used as query sequences in running time analysis.

Run	Bioproject	Species
SRR6425989	PRJNA427774	*Drosophila bipectinata*
SRR6425990	PRJNA427774	*Drosophila erecta*
SRR6425991	PRJNA427774	*Drosophila ananassae*
SRR6425992	PRJNA427774	*Drosophila biarmipes*
SRR6425993	PRJNA427774	*Drosophila mauritiana*
SRR6425995	PRJNA427774	*Drosophila eugracilis*
SRR6425997	PRJNA427774	*Drosophila mojavensis*
SRR6425998	PRJNA427774	*Drosophila persimilis*
SRR6425999	PRJNA427774	*Drosophila simulans*
SRR6426000	PRJNA427774	*Drosophila virilis*
SRR6426001	PRJNA427774	*Drosophila pseudoobscura*
SRR6426002	PRJNA427774	*Drosophila sechellia*
SRR6426003	PRJNA427774	*Drosophila willistoni*
SRR6426004	PRJNA427774	*Drosophila yakuba*
SRR12129012	PRJNA643549	*Drosophila melanogaster*

Table C.17. Samples used for chloroplast genome assembly experiment. 60 SRRs were obtained from recent chloroplast assembly benchmark study by Freudenthal *et al.* [72]. We examined 56 SRRs that failed to be assembled by any of the tools compared in the original study and 4 positive control samples (denoted as *) that were selected randomly from a group of successfully assembled novel genomes. ptDNA column represents the proportion of sequencing reads retained after filtering using CONSULT. GetOrganelle [97] was identified as the best performing plastid assembler by Freudenthal *et al.* so we choose to use it in this experiment. Before and after filtering columns show the length of the largest contig from assemblies produced by GetOrganelle. CONSULT reference database was constructed using genomes from plastid RefSeq release 206 that included 6537 plant species. Dataset was composed of 200,661,586 32bp canonical *k*-mers from which 194,117,165 were included in a reference library. Sample preprocessing was identical to mitochondrial assembly experiment.

Run	Species	ptDNA	Before filtering	After filtering
SRR2531285	*Cucurbita ficifolia*	0.1293	48392	165053
SRR8784098	*Arachis valida*	0.0614	160254	160254
SRR7221532	*Malania oleifera*	0.5869	158252	158160
SRR2531276*	*Cucurbita andreana*	0.0942	48217	157475
SRR7685402	*Cucurbita argyrosperma*	0.1851	82278	157298
SRR5513237	*Cucurbita maxima*	0.0824	66700	157286
SRR6425049*	*Stylosanthes hamata*	0.0788	156503	156503
SRR6425616	*Willughbeia edulis*	0.0953	155275	155275
SRR6425651*	*Pachypodium baronii*	0.1181	154768	154768
SRR6425646*	*Allamanda schottii*	0.1833	154537	154553
SRR2154060	*Fulcaldea stuessyi*	0.0392	152889	152889
SRR4457829	*Pityopsis graminifolia*	0.0804	136239	136185
SRR2401798	*Iodes perrieri*	0.1958	109063	109063
SRR2401805	*Mappianthus iodoides*	0.0252	61664	95815
SRR5851367	*Cuscuta australis*	0.0676	85263	85263
SRR1393892	*Intsia bijuga*	0.0384	24517	77980
SRR2401818	*Platea parviflora*	0.0767	76108	76108
SRR629604	*Leea guineensis*	0.0461	116554	64689
SRR2531274	*Cucurbita argyrosperma*	0.0863	63319	63319
SRR2531272	*Cucurbita argyrosperma*	0.1159	61852	61852
SRR8718117	*Malpighia glabra*	0.0377	29722	55578
SRR6425638	*Orthanthera albida*	0.2167	29915	48885
SRR8690427	*Lycium barbarum*	0.1528	25779	46430
SRR1401591	*Intsia palembanica*	0.0318	24499	45343
SRR6163099	*Castanospermum australe*	0.9618	42596	42596
SRR4099903	*Striga hermonthica*	0.0444	59497	42041
SRR6425652	*Malouetia tamaquarina*	0.1186	40312	40318
SRR2401834	*Cassinopsis madagascariensis*	0.0444	32550	32550
SRR2401864	*Emmotum nitens*	0.0318	30945	30945
SRR2401814	*Ottoschulzia rhodoxylon*	0.0528	56839	29703
SRR2531294	*Cucurbita pepo*	0.1301	12430	26825

(continued on next page)

Table C.17. Samples used for chloroplast genome assembly experiment. 60 SRRs were obtained from recent chloroplast assembly benchmark study by Freudenthal *et al.* [72]. We examined 56 SRRs that failed to be assembled by any of the tools compared in the original study and 4 positive control samples (denoted as *) that were selected randomly from a group of successfully assembled novel genomes. ptDNA column represents the proportion of sequencing reads retained after filtering using CONSULT. GetOrganelle [97] was identified as the best performing plastid assembler by Freudenthal *et al.* so we choose to use it in this experiment. Before and after filtering columns show the length of the largest contig from assemblies produced by GetOrganelle. CONSULT reference database was constructed using genomes from plastid RefSeq release 206 that included 6537 plant species. Dataset was composed of 200,661,586 32bp canonical *k*-mers from which 194,117,165 were included in a reference library. Sample preprocessing was identical to mitochondrial assembly experiment.

(continued from previous page)

Run	Species	ptDNA	Before filtering	After filtering
SRR1592654	*Linum lewisii*	0.0620	14637	22427
SRR2401804	*Mappia mexicana*	0.1819	22328	22328
SRR8082592	*Linum stelleroides*	0.0371	26175	18979
SRR7995544	*Rhopalocnemis phalloides*	0.0785	NA	18793
SRR5500905	*Cladophora albida*	0.0581	77409	17882
SRR5500898	*Valonia utricularis*	0.0153	77409	14022
ERR921729	*Rhynchospora pubera*	0.0997	13026	13461
SRR2401792	*Hosiea japonica*	0.0775	16051	12785
SRR3732558	*Ruellia speciosa*	0.0934	11431	11635
SRR2531293	*Cucurbita pepo*	0.1246	10215	10481
SRR6425612	*Plectaneia stenophylla*	0.0500	9937	9879
SRR2401799	*Iodes scandens*	0.0440	9513	9513
SRR2531295	*Cucurbita pepo*	0.1194	5182	8830
ERR2114804	*Azolla filiculoides*	0.0161	97935	5018
ERR2799486	*Calycadenia mollis*	0.0068	1579	3252
ERR2799490	*Calycadenia spicata*	0.0065	1706	2980
ERR2799488	*Calycadenia oppositifolia*	0.0075	2282	2920
SRR5500903	*Cladophora vadorum*	0.0122	281	2711
ERR2799491	*Calycadenia truncata*	0.0068	1620	2665
ERR2799521	*Euphorbia marginata*	0.0067	2368	2398
ERR2799538	*Limnanthes douglasii*	0.0073	2153	2290
ERR2799494	*Osmadenia tenella*	0.0078	2153	2186
SRR5500897	*Ventricaria ventricosa*	0.0020	213016	1840
ERR2799487	*Calycadenia multiglandulosa*	0.0070	1569	1569
SRR518945	*Rafflesia cantleyi*	0.0574	14801	1421
SRR629600	*Rafflesia tuan-mudae*	0.1038	14793	1392
SRR5500900	*Siphonocladus tropicus*	0.0063	2637	1236
SRR5500899	*Struvea elegans*	0.0055	517	769
ERR2799493	*Hemizonia perennis*	0.0067	NA	NA

Table C.18. Running time of CONSULT vs total number of query reads. Sample sets were selected randomly from a group of chloroplast samples. Computation on a machine with AMD EPYC 7742 2.25 GHz CPU using 24 threads and 120G of RAM.

Samples	Set1	Set2	Set3
Read count	51,393,381	190,221,848	198,400,443
Running time (min)	16.9813	49.8804	62.7144

Table C.19. Summary of repeat content identified in Bee assemblies.

Species	Contigs	Total length, M	GC, %	Bases masked, M	Repeats, %
Bombus impatiens	5460	246.856484	37.76	5.960747	2.41
Bombus terrestris	5609	248.654244	37.51	5.439655	2.19
Bombus polaris	5848	245.790074	37.64	5.479267	2.23
Bombus skorikovi	3042	242.038170	37.62	6.472741	2.67
Bombus soroeensis	5476	243.573539	37.56	5.682580	2.33
Bombus superbus	1458	230.139620	39.43	6.031274	2.62
Bombus waltoni	2240	231.149577	39.40	6.310819	2.73
Bombus opulentus	4394	242.353651	37.39	5.778336	2.38
Bombus consobrinus	7194	249.068899	37.76	7.174060	2.88
Bombus confusus	4639	239.101172	38.83	5.559645	2.33
Bombus picipes	7282	253.987937	37.92	6.101125	2.40
Bombus sibiricus	14183	262.420024	37.16	6.586067	2.51
Bombus difficillimus	4473	243.560081	37.78	5.919934	2.43
Bombus cullumanus	5480	246.986414	38.02	5.452328	2.21
Bombus turneri	3015	243.111512	37.48	6.141349	2.53
Bombus pyrosoma	4727	254.804690	37.44	6.316902	2.48
Bombus ignitus	748	242.565149	37.55	5.684718	2.34
Bombus breviceps	1317	248.124274	37.72	5.864471	2.36
Bombus haemorrhoidalis	1614	240.538686	37.68	6.295373	2.62

Table C.20. Summary of repeat content identified in Drosophila assemblies.

Species	Contigs	Total length, M	GC, %	Bases masked, M	Repeats, %
Drosophila ananassae	371	189.221946	41.75	6.847882	3.62
Drosophila biarmipes	661	182.453935	41.48	5.095512	2.79
Drosophila bipectinata	570	163.165444	41.45	5.721482	3.51
Drosophila erecta	58	130.293209	42.90	4.765988	3.66
Drosophila eugracilis	546	159.429531	40.46	6.003497	3.77
Drosophila mauritiana	266	134.165749	42.64	4.734488	3.53
Drosophila mojavensis	122	168.142858	39.68	17.336546	10.31
Drosophila persimilis	415	163.933157	44.83	8.968285	5.47
Drosophila pseudoobscura	361	159.031139	45.01	8.805636	5.54
Drosophila sechellia	109	138.120607	42.33	4.216694	3.05
Drosophila simulans	76	133.725236	42.58	4.594273	3.44
Drosophila virilis	141	169.714588	40.44	11.747065	6.92
Drosophila willistoni	489	194.955081	36.96	12.430612	6.38
Drosophila yakuba	111	143.252825	42.53	5.293281	3.70

Table C.21. Whale assemblies used to evaluate across species support.

Assembly accession	Bioproject	Species taxonomy ID	Species
GCA_000331955.2	PRJNA167475	9733	*Orcinus orca*
GCA_000442215.1	PRJNA174066	118797	*Lipotes vexillifer*
GCA_000493695.1	PRJNA72723	9767	*Balaenoptera acutorostrata*
GCA_000978805.1	PRJDB1465	33556	*Balaenoptera bonaerensis*
GCA_002189225.1	PRJNA384396	9764	*Eschrichtius robustus*
GCA_002288925.3	PRJNA360851	9749	*Delphinapterus leucas*
GCA_002837175.2	PRJNA411766	9755	*Physeter catodon*
GCA_003031525.2	PRJNA343136	189058	*Neophocaena asiaeorientalis*
GCA_003227395.1	PRJNA369596	79784	*Tursiops aduncus*
GCA_003676395.1	PRJNA475306	90247	*Lagenorhynchus obliquidens*
GCA_004027085.1	PRJNA399476	48745	*Mesoplodon bidens*
GCA_004329385.1	PRJNA509641	9773	*Megaptera novaeangliae*
GCA_004363435.1	PRJNA399467	48752	*Platanista minor*
GCA_004363455.1	PRJNA399464	302098	*Eubalaena japonica*
GCA_004363495.1	PRJNA399454	9742	*Phocoena phocoena*
GCA_004363515.1	PRJNA399465	9725	*Inia geoffrensis*
GCA_004363705.1	PRJNA399466	27615	*Kogia breviceps*
GCA_004364475.1	PRJNA399469	9760	*Ziphius cavirostris*
GCA_005190385.2	PRJNA520934	40151	*Monodon monoceros*
GCA_006547405.1	PRJNA525909	9731	*Globicephala melas*
GCA_007760645.1	PRJNA508467	103600	*Sousa chinensis*
GCA_008692025.1	PRJNA557831	42100	*Phocoena sinus*
GCA_008795845.1	PRJNA72723	9770	*Balaenoptera physalus*
GCA_009873245.3	PRJNA554522	9771	*Balaenoptera musculus*
GCA_011754075.1	PRJNA239019	48723	*Pontoporia blainvillei*
GCA_011762595.1	PRJNA608726	9739	*Tursiops truncatus*
GCA_017311385.1	PRJNA675309	118798	*Platanista gangetica*

Table C.22. Raw sequencing samples of Gray's beaked whale *Mesoplodon grayi* used to compute within species support obtained from NCBI project PRJNA702760 [245]. Two isolates with very low coverage (PEMN3299 and MGRTM15) were excluded. SRA files with the same sample names were combined.

Run	Bases, G	Biosample	Sample name
SRR13736956	2.65	SAMN17977870	M15813
SRR13736985	0.76	SAMN17977870	M15813
SRR13736955	1.68	SAMN17977871	M18082
SRR13736984	0.49	SAMN17977871	M18082
SRR13736983	0.48	SAMN17977872	M22364
SRR13736988	1.67	SAMN17977872	M22364
SRR13736977	1.52	SAMN17977873	M22390
SRR13736982	0.43	SAMN17977873	M22390
SRR13736966	2.34	SAMN17977874	MGR127
SRR13736981	0.65	SAMN17977874	MGR127
SRR13736980	0.65	SAMN17977875	MGR135
SRR13736961	2.30	SAMN17977875	MGR135
SRR13736979	0.58	SAMN17977876	MGR136
SRR13736960	2.06	SAMN17977876	MGR136
SRR13736959	2.04	SAMN17977877	MGR137
SRR13736978	0.58	SAMN17977877	MGR137
SRR13736976	0.63	SAMN17977878	MGR138
SRR13736958	2.21	SAMN17977878	MGR138
SRR13736957	1.86	SAMN17977879	MGR139
SRR13736975	0.53	SAMN17977879	MGR139
SRR13736954	1.94	SAMN17977880	MGR141
SRR13736974	0.55	SAMN17977880	MGR141
SRR13736953	1.71	SAMN17977881	MGR159
SRR13736973	0.48	SAMN17977881	MGR159
SRR13736996	2.07	SAMN17977882	MGR161
SRR13736972	0.58	SAMN17977882	MGR161
SRR13736995	1.65	SAMN17977883	MGR164
SRR13736971	0.45	SAMN17977883	MGR164
SRR13736970	0.52	SAMN17977884	MGR165
SRR13736994	1.83	SAMN17977884	MGR165
SRR13736969	0.67	SAMN17977885	MGRM49873
SRR13736993	2.39	SAMN17977885	MGRM49873
SRR13736992	1.49	SAMN17977886	MGRM49874
SRR13736968	0.40	SAMN17977886	MGRM49874
SRR13736990	2.41	SAMN17977888	MGRTM16
SRR13736965	0.69	SAMN17977888	MGRTM16
SRR13736989	2.16	SAMN17977889	MRGWA04
SRR13736964	0.63	SAMN17977889	MRGWA04
SRR13736962	0.55	SAMN17977891	PEMN3300
SRR13736986	1.88	SAMN17977891	PEMN3300

Table C.23. Bee assemblies used in real data analysis [226].

Assembly accession	Bioproject	Species taxonomy ID	Species
GCA_000188095.4	PRJNA61101	132113	*Bombus impatiens*
GCA_000214255.1	PRJNA45869	30195	*Bombus terrestris*
GCA_014737335.1	PRJNA659133	130708	*Bombus polaris*
GCA_014737355.1	PRJNA659133	395565	*Bombus skorikovi*
GCA_014737365.1	PRJNA659133	184059	*Bombus soroeensis*
GCA_014737385.1	PRJNA659133	1869276	*Bombus superbus*
GCA_014737395.1	PRJNA659133	395577	*Bombus waltoni*
GCA_014737405.1	PRJNA659133	2024865	*Bombus opulentus*
GCA_014737455.1	PRJNA659133	130686	*Bombus consobrinus*
GCA_014737475.1	PRJNA659133	217217	*Bombus confusus*
GCA_014737485.1	PRJNA659133	309970	*Bombus picipes*
GCA_014737505.1	PRJNA659133	421273	*Bombus sibiricus*
GCA_014737525.1	PRJNA659133	395520	*Bombus difficillimus*
GCA_014737535.1	PRJNA659133	2562068	*Bombus cullumanus*
GCA_014825825.1	PRJNA659133	686820	*Bombus turneri*
GCA_014825855.1	PRJNA659133	396416	*Bombus pyrosoma*
GCA_014825875.1	PRJNA659133	130704	*Bombus ignitus*
GCA_014825925.1	PRJNA659133	395515	*Bombus breviceps*
GCA_014825975.1	PRJNA659133	207636	*Bombus haemorrhoidalis*

Table C.24. Samples of Lice used for evaluation of subsampling on real data [33] from the NCBI project PRJNA296666.

Run	Bases, G	Biosample	Species
SRR3161912	7.94	SAMN04103756	*Columbicola adamsi*
SRR3161913	13.50	SAMN04103757	*Columbicola gracilicapitis*
SRR3161914	9.61	SAMN04103758	*Columbicola macrourae*
SRR3161915	11.19	SAMN04103759	*Columbicola theresae*
SRR3161916	12.77	SAMN04103760	*Columbicola claytoni*
SRR3161917	7.21	SAMN04103761	*Columbicola columbae*
SRR3161918	12.47	SAMN04103762	*Columbicola rodmani*
SRR3161919	14.09	SAMN04103763	*Columbicola veigasimoni*
SRR3161920	10.04	SAMN04103764	*Columbicola claviformis*
SRR3161921	9.05	SAMN04103765	*Columbicola altamimiae*
SRR3161922	10.08	SAMN04103766	*Columbicola drowni*
SRR3161923	7.12	SAMN04103767	*Columbicola gymnopeliae*
SRR3161924	9.43	SAMN04103768	*Columbicola extinctus*
SRR3161925	16.13	SAMN04103769	*Columbicola fortis*
SRR3161926	7.97	SAMN04103770	*Columbicola wecksteini*
SRR3161927	11.52	SAMN04103771	*Columbicola guimaraesi*
SRR3161928	9.75	SAMN04103772	*Columbicola taschenbergi*
SRR3161929	9.64	SAMN04103773	*Columbicola mckeani*
SRR3161930	9.96	SAMN04103774	*Columbicola passerinae*
SRR3161931	8.38	SAMN04103775	*Columbicola passerinae*
SRR3161932	4.01	SAMN04103776	*Columbicola palmai*
SRR3161933	11.02	SAMN04103777	*Columbicola waltheri*
SRR3161934	7.35	SAMN04103778	*Columbicola clayae*
SRR3161935	9.30	SAMN04103779	*Columbicola meinertzhageni*
SRR3161936	9.81	SAMN04103780	*Columbicola smithae*
SRR3161937	17.60	SAMN04103781	*Columbicola masoni*
SRR3161938	5.70	SAMN04103782	*Columbicola wolffhuegeli*
SRR3161939	16.00	SAMN04103783	*Columbicola exilicornis*
SRR3161940	12.32	SAMN04103784	*Columbicola waiteae*
SRR3161941	10.36	SAMN04103785	*Columbicola beccarii*
SRR3161942	16.56	SAMN04103786	*Columbicola mjoebergi*
SRR3161943	8.43	SAMN04103787	*Columbicola wombeyi*
SRR3161944	7.58	SAMN04103788	*Columbicola paradoxus*
SRR3161945	14.74	SAMN04103789	*Columbicola exilicornis*
SRR3161946	9.33	SAMN04103790	*Columbicola masoni*
SRR3161947	13.38	SAMN04103791	*Columbicola angustus*
SRR3161948	21.20	SAMN04103792	*Columbicola tasmaniensis*
SRR3161949	13.85	SAMN04103793	*Columbicola harbisoni*
SRR3161950	12.87	SAMN04103794	*Columbicola bacillus*

(continued on next page)

Table C.24. Samples of Lice used for evaluation of subsampling on real data [33] from the NCBI project PRJNA296666.

(continued from previous page)

Run	Bases, G	Biosample	Species
SRR3161951	11.67	SAMN04103795	*Columbicola sp. CospStori*
SRR3161952	14.65	SAMN04103796	*Columbicola macrourae*
SRR3161953	12.19	SAMN04103797	*Columbicola macrourae*
SRR3161954	10.57	SAMN04103798	*Columbicola sp. CospAplar*
SRR3161955	8.41	SAMN04103799	*Columbicola guimaraesi*
SRR3161956	14.87	SAMN04103800	*Columbicola malenkeae*
SRR3161957	19.06	SAMN04103801	*Columbicola mjoebergi*
SRR3161958	17.47	SAMN04103802	*Columbicola koopae*
SRR3161959	8.74	SAMN04103803	*Columbicola tschulyschman*
SRR3161960	14.22	SAMN04103804	*Columbicola eowilsoni*
SRR3161961	15.72	SAMN04103805	*Columbicola arnoldi*
SRR3161962	6.31	SAMN04103806	*Columbicola exilicornis*
SRR3161963	21.42	SAMN04103807	*Columbicola sp. CospPaoen*
SRR3161964	19.47	SAMN04103808	*Columbicola triangularis*
SRR3161965	16.23	SAMN04103809	*Columbicola timmermanni*
SRR3161966	10.01	SAMN04103810	*Columbicola elbeli*
SRR3161967	15.24	SAMN04103811	*Columbicola bacillus*
SRR3161968	11.42	SAMN04103812	*Columbicola hoogstraali*
SRR3161969	19.28	SAMN04103813	*Columbicola sp. CospTutym*
SRR3161970	11.74	SAMN04103814	*Columbicola sp. CospTucha*
SRR3161971	15.80	SAMN04103815	*Columbicola sp. CospZemel*
SRR3161972	14.43	SAMN04103816	*Columbicola sp. CospStlug*

Table C.25. GUNC quality metrics [170].

Metric	Definition
Genome	name of input genome.
Number of genes called	number of genes called by prodigal or directly provided by the user.
Number of genes mapped	number of genes mapped by diamond into GUNC refDB.
Number of contigs with mapped genes	number of contigs containing mapped genes.
Taxonomic level	taxonomic clade labels at this taxonomic level were used to calculate values in all following columns. For each genome, all scores at six levels (species level can be added using a command-line option) are calculated.
Proportion of genes retained in major clades	only major clades that have >2% of all mapped genes assigned to them are retained to calculate other scores. Value of this column is n_genes_retained/n_genes_mapped.
Genes retained index	portion of all called genes retained in major clades.
Clade separation score	a result of applying a formula explained in GUNC paper to taxonomy and contig labels of genes retained in major clades. Ranges from 0 to 1 and is set to 0 when genes_retained index is <0.4 because that is too few genes left.
Contamination portion	a portion of genes retained in major clades assigned to all clades except the one clade with the highest proportion of genes assigned to it.
Number of effective surplus clades	an Inverse Simpson Index of fractions of all clades - 1 (as 1 genome is expected). It is a score describing the extent of chimerism, i.e. the effective number of surplus clades represented at a taxlevel.
Mean hit identity	the mean identity with which genes in abundant lineages (>2%) hit genes in the reference.
Reference representation score	estimates how well a genome is represented in the GUNC DB.
pass.GUNC	If a genome passes GUNC analysis it means it is likely to not be chimeric (or that chimerism cannot be detected especially when its reference representation (RRS) is low). A genome passes if clade_separation_score $\leq$ 0.45, a cutoff benchmarked using simulated genomes.

Table C.26. CheckM quality metrics [176].

Metric	Definition
Bin Id	unique identifier of genome bin (derived from input fasta file).
Marker lineage	indicates the taxonomic rank of the lineage-specific marker set used to estimated genome completeness, contamination, and strain heterogeneity.
Count of genomes	number of reference genomes used to infer the lineage-specific marker set.
Count of markers	number of marker genes within the inferred lineage-specific marker set.
Count of marker sets	number of co-located marker sets within the inferred lineage-specific marker set.
Completeness	estimated completeness of genome as determined from the presence/absence of marker genes and the expected collocalization of these genes.
Contamination	estimated contamination of genome as determined by the presence of multi-copy marker genes and the expected collocalization of these genes.
Strain heterogeneity	estimated strain heterogeneity as determined from the number of multi-copy marker pairs which exceed a specified amino acid identity threshold (default = 90%). High strain heterogeneity suggests the majority of reported contamination is from one or more closely related organisms (i.e. potentially the same species), while low strain heterogeneity suggests the majority of contamination is from more phylogenetically diverse sources.
Genome size	number of nucleotides (including unknowns specified by N's) in the genome.
Count of ambiguous bases	number of ambiguous (N's) bases in the genome.
Count of scaffolds	number of scaffolds within the genome.
Count of contigs_x	number of contigs within the genome as determined by splitting scaffolds at any position consisting of more than 10 consecutive ambiguous bases.
N50 (scaffolds)	N50 statistics as calculated over all scaffolds.
N50 (contigs)	N50 statistics as calculated over all contigs
Mean scaffold length	mean length as calculated over all scaffolds.
Mean contig length	mean length as calculated over all contigs.
Longest scaffold	the longest scaffold within the genome.
Longest contig	the longest contig within the genome.
GC	number of G/C nucleotides relative to all A,C,G, and T nucleotides in the genome.
GC for scaffolds above 1kbp	number of G/C nucleotides relative to A,C,G, and T nucleotides in scaffolds $>$1kbp.
Coding density	the number of nucleotides within a coding sequence (CDS) relative to all nucleotides in the genome.
Translation table	indicates which genetic code was used to translate nucleotides into amino acids.
Count of predicted genes	number of predicted coding sequences (CDS) within the genome as determined using Prodigal.
qa0-qa5+	number of times each marker gene is identified.

Table C.27. QUAST quality metrics [152].

Metric	Definition
Assembly	name.
Number of contigs of x bp or longer	total number of contigs of length $\geq$ x bp.
Total length at x bp or above	total number of bases in contigs of length $\geq$ x bp.
Count of contigs in assembly	total number of contigs in the assembly.
Largest contig in assembly	length of the longest contig in the assembly.
Total length of assembly	total number of bases in the assembly.
G and C in assembly (%)	total number of G and C nucleotides in the assembly, divided by the total length of the assembly.
N50	length for which the collection of all contigs of that length or longer covers at least half an assembly.
N75	length for which the collection of all contigs of that length or longer covers at least 75% of an assembly.
L50	is the number of contigs equal to or longer than N50. In other words, L50, for example, is the minimal number of contigs that cover half the assembly.
L75	is the number of contigs equal to or longer than N75. In other words, L75, for example, is the minimal number of contigs that cover 75% of the assembly.
Count of N per 100 kbp	average number of uncalled bases (N's) per 100,000 assembly bases.

Appendix D

Supplementary method details and commands

Here we provide the exact procedures and commands that we used to run external software throughout our experiments.

Genome skim simulation

We simulated short reads with length l = 100 and coverage c, in single read mode with default error and quality profiles of Illumina HiSeq 2500 using ART version 2.5.8 with command

```
art_illumina -ss HS25 -i FASTA_FILE -l 100 -f c -na -s 10
-o FASTQ_FILE
```

Downsampling reads

To subsample reads down a specified number of reads n we used seqtk version 1.3r106 with command

```
seqtk sample -s150 INPUT_FASTQ_FILE n > OUTPUT_FASTQ_FILE
```

Kraken reference library construction

In this study we used Kraken version 2.

- To construct standard Kraken reference library we used default command

  ```
  kraken2-build --standard --no-masking --use-ftp
  --db DATABASE_NAME
  ```

- To build custom Kraken reference library we used a set of commands below:

  1. Download taxonomy

```
kraken2-build --download-taxonomy --no-masking --use-ftp
--db DATABASE_NAME
```

2. Rename file extensions to .fa

```
find .  -name "*.fna" -exec sh -c 'mv "$1" "${1%.fna}.fa"'
- {} \;
```

3. Add custom genomes to the reference library

```
find genomes/ -name '*.fa' -print0 | xargs -0 -I
-n1 kraken2-build --no-masking --add-to-library {}
--db DATABASE_NAME
```

4. To build database with specified k-mer length k, minimizer length l and number of wind-carding positions s we used

```
kraken2-build --build --no-masking --kmer-len k
--minimizer-len l --minimizer-spaces s --use-ftp
--db DATABASE_NAME
```

Kraken reference library querying

To query Kraken reference library at variable confidence level α we used

```
kraken2 --use-names --threads 24
--report REPORT_FILE_NAME --db DATABASE_NAME --confidence α
--classified-out CLASSIFIED_FASTQ_FILE
--unclassified-out UNCLASSIFIED_FASTQ_FILE QUERY_FASTQ_FILE >
KRAKEN_OUTPUT_FILE
```

Computing k-mer frequencies

To estimate k-mer frequencies we used Jellyfish version 2.3.0.

- Computing k-mer profile

```
jellyfish count -m 31 -s 100M -t 18 -C INPUT_FASTQ_FILE
-o COUNT_FASTQ_FILE
```

- Extracting k-mer statistics

```
jellyfish stats COUNT_FASTQ_FILE
```

CLARK(-S) reference library construction and querying

We used CLARK version 1.2.6.1.

During database construction, CLARK was able to set target IDs for 9976 genomes out of 10470 sample that we aimed to include. Remaining TOL genomes were added manually by setting taxonomic rank for a given sample to Proteobacteria phylum and specifying taxonomy ID as 1224.

- Custom CLARK database of discriminative *k*-mers was built in <DIR_DB/> the following way:

```
1.  We created the directory "Custom" inside <DIR_DB/>

2.  We copied the sequences of interest (in our case
reference TOL fasta files with accession numbers) in
the "Custom"

3.  We ran

./set_target.ssh <DIR_DB/> custom --phylum

We note we set taxonomy rank to phylum.
The default taxonomy rank is species.
```

- CLARK database was queried at default mode of classification (i.e., "-m 1") with script

```
./classify_metagenome.sh -O SAMPLE_FASTQ_FILE -R RESULT_CSV
-n 24
```

- CLARK-S databases of discriminative **spaced** *k*-mers was created on top of custom CLARK database using script

```
./buildSpacedDB.sh
```

- CLARK-S classification was ran with "full" mode ("-m 0" identifier) using script

```
./classify_metagenome.sh -O SAMPLE_FASTQ_FILE -R RESULT_CSV
-n 24 --spaced -m 0
```

Bowtie2 reference index construction and alignment

We used Bowtie2 version 2.4.1.

- To index reference genomes we used command

```
bowtie2-build <reference_in> <bt2_base> --threads 24
```

- To run alignment we used sensitive setting

```
bowtie2 -x <bt2_base> -U INPUT_FASTQ_FILE --local
--sensitive -p 24 -S RESULT_SAM_FILE
```

- For whale analyses, we used (GCA_004027085.1) as the reference.

Manipulation and post-processing of read alignments

We used SAMtools version 1.9.

- To obtain alignment statistics

```
samtools stats INPUT_ALN_SAM > OUTPUT_STATS_TXT

samtools flagstat INPUT_ALN_SAM > OUTPUT_STATS_TXT
```

CONSULT reference library construction and querying

- To build CONSULT reference libraries we used version 17.1 of mapping software

  1. We compiled script using
     `g++ main_map.cpp -std=c++11 -O3 -o main_map`
  2. To construct reference database with default settings we ran
     `./main_map -i INPUT_FASTA_FILE -o DB_FOLDER_NAME`

  To avoid biases during testing we fixed positions of bits that are selected during signature generation. In released version positions are selected randomly.

  For h=15 we used $l = 0$ $bits \in \{30,28,25,24,22,21,20,17,14,13,11,5,3,2,1\}$

  $l = 1$ $bits \in \{30,29,26,24,20,19,16,14,13,12,10,7,3,1,0\}$

  $l = 2$ $bits \in \{30,29,26,23,21,20,19,18,15,11,9,7,5,3,2\}$

- CONSULT database was queried using version 17.4 of search software

  1. To compile
     ```
     g++ main_search.cpp -std=c++11 -fopenmp -O3
     -o main_search
     ```
  2. To query sequence reads against reference database we ran
     ```
     ./main_search -i DB_FOLDER_NAME -c 1 -t 24
     -q QUERY_FOLDER_NAME
     ```

     where arguments are:

     ```
     -i - name of the reference database
     -c - the lowest number of k-mers that is required to
     mark sequencing read as classified.  For instance, if
     at least one k-mer match is enough to classify a read,
     "c" should be set to 1.  If at least two k-mer matches
     are required to call read a match, "c" should be set
     to 2.
     -t - number of threads
     -q - name of the folder where queries are located
     ```

  For measuring running time of CONSULT, we split query file using Linux `split -n l/48` and provided a folder containing all the files to the tool; we added split time to the total running time.

Generation of *k*-mer sets

To estimate *k*-mer frequencies we used Jellyfish version 2.3.0.

- To compute 35bp and 32bp canonical *k*-mer profiles of fasta genomic references we used

```
jellyfish count -m 35 -s 100M -t 24 -C INPUT_FASTA_FILE
-o COUNT_JF_FILE
```

```
jellyfish count -m 32 -s 100M -t 24 -C INPUT_FASTA_FILE
-o COUNT_JF_FILE
```

- To output a list of all the *k*-mers in the file associated with their counts

```
jellyfish dump COUNT_JF_FILE > OUTPUT_FASTA_FILE
```

k-mer minimization

Minimization was performed using custom c++ minimization script version 3.0 which is available for public use. This script accepts as an input Jellyfish fasta file containing 35 bp canonical *k*-mers extracted from reference genomes and outputs their 32 bp minimizers in fasta format.

- To compile the script we used

```
g++ minimization_v3.0.cpp -std=c++11 -o main_minimization
```

- We ran minimization with the command

```
./main_minimization -i INPUT_FASTA_FILE
-o MIN_OUTPUT_FASTA_FILE
```

Computing genomic distances with Mash

To estimate genomic distances we used Mash version 2.3.

- To compute genomic distance using Mash we used

```
mash dist FASTQ_FILE_ONE FASTQ_FILE_TWO
```

Computation of genomic distances

To estimate genomic distances we used Skmer 3.0.2.

- To compute genomic distance with Skmer we used

```
skmer reference FASTQ_DIRECTORY -p 24 -t
-o REF_DISTANCE_MATRIX
```

Computation of subsampling replicates and distance matrix correction

To estimate genomic distances we used Skmer 3.1.0.

- To generate subsampling replicates

```
skmer subsample -b 100 FASTQ_DIRECTORY -s 100000 -S 42
-p 24 -t -i 0
```

- To correct distance matrices

```
skmer correct -main FASTQ_DIRECTORY/REF_DISTANCE_MATRIX
-sub SUBSAMPLE_DIRECTORY
```

Preprocessing of real sequencing files

We used BBTools version 38.59 to preprocess real data sequencing reads.

- To decontaminate .fastq files we used

```
bbduk.sh in1=FASTQ_READ1 in2=FASTQ_READ2 out1=FASTQ_READ1
out2=FASTQ_READ2 ref=adapters,phix ktrim=r k=23 mink=11
hdist=1 tpe tbo
```

- To deduplicate reads we used

```
dedupe.sh in1=FASTQ_READ1 in2=FASTQ_READ2
out=DEDUP_OUTPUT_FASTQ_FILE
```

- To reformat deduplicated output files we used

```
reformat.sh in=DEDUP_OUTPUT_FASTQ_FILE out1=FASTQ_READ1
out2=FASTQ_READ2
```

- To merge paired-end reads .fastq we used

```
bbmerge.sh in1=FASTQ_READ1 in2=FASTQ_READ2
out1=OUTPUT_FASTQ_FILE
```

Preprocessing of mitochondrial sequencing files

- We used AdapterRemoval version 2.3.1 to remove adapters from raw sequencing data

```
AdapterRemoval --file1 FASTQ_READ1 --file2 FASTQ_READ2
--basename FASTQ_trmd
```

- We used BBTools version 38.59 to merge paired-end reads

```
bbmerge.sh in1=FASTQ_READ1 in2=FASTQ_READ2 out=merged.fastq
outu1=read1_unmerged.fastq outu2=read2_unmerged.fastq
```

- Subsequently we concatenated merged and unmerged outputs

```
cat merged.fastq read1_unmerged.fastq read2_unmerged.fastq
> OUTPUT_FILE
```

To collect classified reads from CONSULT we used

- We used BBTools version 38.59

```
filterbyname.sh in=INIT_FILE names=UCSEQ_FILE out=CSEQ_FILE
include=f int=f overwrite=true
```

Assembling mitochondrial reads

To assemble mitochondrail reads we used SPAdes genome assembler version 3.15.0.

- On unfiltered reads we ran

```
spades.py --plasmid --only-assembler -s INPUT_FASTQ -t 24
-m 120 -o OUTPUT_DIR
```

- To assemble filtered reads we ran

```
spades.py --only-assembler -s INPUT_FASTQ -t 24 -m 120
-o OUTPUT_DIR
```

Mitochondrial annotation

Annotation of mitochondrial assemblies was done with MITOS version 2.0.8 using RefSeq89 Metazoa reference and vertebrate mitochondrial code 2.

- `runmitos.py -i INPUT_FASTA -c 2 -o OUTPUT_DIR --linear -r REFSEQVER -R REFDIR`

- To confirm identity of largest mitochondrial contigs for unfiltered assemblies we used MitoZ version 2.4

```
python MitoZ.py annotate --genetic_code auto
--clade Chordata --outprefix test --thread_number 8
--fastafile INPUT_FASTA
```

Assembling chloroplast reads

Chloroplast assembly was performed using GetOrganelle version 1.6.4. Annotations were done using web-based annotation tool GeSeq v2.03. Sequence source option was set to plastid (land plants).

- `get_organelle_from_reads.py -u INPUT_FASTQ -t 24 -o OUTPUT_DIR -F embplant_pt`

Simulation of reference sequences

To generate sequences for phylogenetic trees we used INDELible v1.03. Software was ran using

```
indelible CONTROL_FILE
```

Control files used in this study are archived with corresponding datasets that were generated by them and are available at https://github.com/noraracht/subsample_support_scripts.

Phylogeny inference

We note that distance matrices need to be converted from standard square matrix to phylip formated matrix before phylogeny estimation can be performed. Tree topologies were inferred from distance matrices using FastMe version 2.1.5 and support was computed using RAxML version 8.2.12.

- To compute backbone we used

  ```
  fastme -i INPUT_FILE -t 24 -o OUTPUT_BACKBONE
  ```

- To get a tree with support values from mean/main corrected matrices

  ```
  raxmlHPC -f b -m GTRCAT -z bootstrapTrees -t referenceTree
  -n TEST
  ```

- To compute a majority rule consensus tree out of the subsampled mean/mean corrected replicates

  ```
  raxmlHPC -J MRE -z RaxML_bootstrap.All -p 4424 -m GTRCAT
  -n MRE_CONS
  ```

Removing repeat content

We screened DNA sequences for interspersed repeats and low complexity DNA sequences using RepeatMasker version 4.1.2-p1. Dependencies included Tandem Repeats Finder v4.09.1, sequence search engine rmblastn version 2.11.0+ and complete (with curated and uncurated families) repeat database FamDB: CONS-Dfam.h5 version 3.5 (from December 22, 2021). Software was ran using

```
RepeatMasker INPUT_FILE -pa 24
```

Generation of *k*-mer sets for short *k*-mers

To estimate *k*-mer frequencies we used Jellyfish version 2.3.0.

- To compute 7bp and 6bp canonical *k*-mer profiles of fasta genomic references we used

```
jellyfish count -m 7 -s 100M -t 24 -C INPUT_FASTA_FILE
-o COUNT_JF_FILE
jellyfish count -m 6 -s 100M -t 24 -C INPUT_FASTA_FILE
-o COUNT_JF_FILE
```

- To output a list of all the *k*-mers in the file associated with their counts in a column format

```
jellyfish dump -c COUNT_JF_FILE > OUTPUT_FASTA_FILE
```

Computation CAFE distances

To compute multiple CAFE distances we used CAFE version 1.0.0.

- To compute genomic distance with CAFE we used

```
cafe -F fna_dir -D Cosine,Co-phylog,Eu,JS,CVtree,D2star,
D2shepp,Ma -K 7 -J jellyfish_exe_path -O results
-S model_dir
```

Alignment based placement

To place genomes that went thorough multiple sequence alignment we use EPA-ng and RAxML.

- To compute model parameters we used RAxML-NG version 1.2.0.

```
raxml-ng --evaluate --msa $REF_MSA --model LG+R10
--tree $TREE --brlen scaled --threads 1 --prefix info
--redo --force
```

- To perform placement we used EPA-ng version 0.3.8.

```
epa-ng --ref-msa $REF_MSA --tree $TREE --query $QUERY_MSA
--model prefix_RAxML.bestModel --redo
```

Computing pairwise distance matrices using kf2d

To estimate genomic distances using our approach version 1.0.1.

- To train global model and classify sequences into clades

```
python main.py -f input_feature_table.csv
-t true_pairiwise.di_mtrx -q query_labels.txt
-d target_clades.txt
```

- To compute local models per clade

```
python main_v2.py -f input_feature_table.csv
-t true_pairiwise.di_mtrx -q query_labels.txt
-c classes.csv
```

- To query sequences

```
python query.py -f input_feature_table.csv
-q query_labels.txt -c classes.csv
```

where arguments are:

`-i - k-mer frequency vectors for all samples organized as input feature table`

`-t - ground truth distance matrix from reference phylogeny`

`-q - list of query IDs`

`-d - input information about true target clades`

`-c - classes information, obtained from global training`

To obtain placement results from pairwise distance comparisons

To place genomes using pairwise distance matrices we used following commands

- We performed placement we used APPLES version 2.0.9.

```
run_apples.py -d dist.mat -t backbone_tree.nwk
-o output_tree.jplace -f 0 -b 5
```

- To convert output tree from jplace to newick format we used GAPPA version 0.8.0.

```
gappa examine graft --jplace-path output_tree.jplace
--out-dir ./result --allow-file-overwriting
```

- To compute placement error we used custom script

```
bash evaluate_placement.sh true_tree.newick
result_placement_tree.newick query_list.txt
backbone_tree.nwk > output.pl_error
```

Phylogenetic tree processing

To perform tree manipulations we used Newick utilities [102] version 1.6.

- To removes branches based on labels

```
nw_prune -f input_tree.nwk query_to_remove.list
> prunned_tree.nwk
```

- To extracts branch lengths for leaf taxa and output distance matrix

```
nw_distance -mm -n -sf input_tree.newick > output.dist_mtrx
```

- To get a list of leaf labels in a tree

```
nw_labels -I input_tree.nwk > output_labels.list
```

To split phylogenetic tree into clades or subclades

We first set length for all branches on phylogeny to 1.0 using TreeSwift [160] version 1.1.26. Then we performed tree clading using TreeCluster [15] version 1.0.3.

- To generate larger clades we used

```
TreeCluster.py -i input_tree.newick -o output_clades.txt
-m sum_branch -t 1700
```

- To generate smaller subclades we used

```
TreeCluster.py -i input_tree.newick -o output_clades.txt
-m sum_branch -t 65
```

Appendix E

Algorithms

Algorithm 3. CONSULT algorithm. Notations: $\mathscr{S}$: all reference sequences. Defaults: $m = 35, k = 32, h = 15, l = 2, t = 2, b = 7, p = 3, c = 1$, $g = 33$. $[a]$ denotes $\{0, \ldots, a-1\}$. $I(s)$ returns $2h-t$ least significant bits of s and $T(s)$ returns the rest. `SHIFTS` returns the number of runs of 0s and 1s in its input as tuples. `SHLD` is the x86 instruction of the same name (Double Precision Shift Left, or extended shift). Details are omitted.

procedure BUILDLIBRARY($\mathscr{S}$)
  $E \leftarrow$ array of $\leq 2^g$ elements, each $2k$ bits
  for each $i \in [l]$ **do**
    $M_i \leftarrow h$ unique random numbers in $[k-1]$
    $S_i \leftarrow 2^{2h-t} \times (2^t b)$ array of g-bit elements
    $T_i \leftarrow 2^{2h-t} \times (2^t)$ array of 1-byte elements
    $S'_i \leftarrow 2^{2h-t} \times (2^t b)$ array of (t,g)-bit tuples
  $\mathscr{K} \leftarrow$ MINIMIZE({all m-mers of $\mathscr{S}$}, k)
  $j \leftarrow -1$
  for each k-mer $a \in \mathscr{K}$ **do**
    $e \leftarrow$LEFTRIGHTENCODE(a)
    Included $\leftarrow$ False
    for each $i \in [l]$ **do**
      $s \leftarrow$ SIGNATURE(M_i, e)
      if $S'_i[I(s)]$ is not full **then**
        if not Included **then**
          $j \leftarrow j+1$
          $E[j] \leftarrow e$
          Included $\leftarrow$ True
        Append $(T(s), j)$ to $S'_i[I(s)]$
  for each row s of each S'_i **do**
    sort values of s, note boundaries of tags
    save 2nd elements of s to $S_i[I(s)]$
    save boundaries of tags in $T_i[I(s)]$
  save $DB = (E, M, S, T)$ to disk

procedure MINIMIZE($\mathscr{K}', k$) ▷ Minimization
  $\mathscr{R} \leftarrow \emptyset$
  for each a in $\mathscr{K}'$ **do**
    Append min{all k-mers of a} to $\mathscr{R}$
  return $\mathscr{R}$ ordered pseudo-randomly

procedure LEFTRIGHTENCODE(a) ▷ Encoding
  $R \leftarrow 2k$-bit zeros
  for letter a_i in a **do**
    $R_i = 1$ if $a_i \in \{G, T\}$
    $R_{i+32} = 1$ if $a_i \in \{C, T\}$
  return R

procedure SIGNATURE(M, e) ▷ Extract signature from e
  $r \leftarrow 2h$-bit zeros
  for $(sh_{skip}, sh_{keep}) \in$ SHIFTS(M) **do**
    $e \leftarrow$ Shift e left by sh_{skip}
    $r \leftarrow$`SHLD` (sh_{keep}, e, r)
  return r

procedure QUERYREAD(r, DB)
  $l \leftarrow 0$
  for k-mer a in r and its reverse complement **do**
    $e \leftarrow$LEFTRIGHTENCODE(a)
    for each $i \in [l]$ **do**
      $s \leftarrow$ SIGNATURE(M_i, e)
      for $T_i[T(s)] \leq f < [T_i[T(s)+1]]$ **do**
        if HD($E[S_i[I(s)][f]], e) \leq p$ **then**
          $l \leftarrow l+1$
          if $l \geq c$ **then**
            return r is a match
  return e is not a match

procedure HD(a,b) ▷ Hamming Distance
  $z_{up}, z_{low} \leftarrow$ lower and upper k-bits of $a \oplus b$
  return `popcount`$(z_{up} \vee z_{low})$

Bibliography

[1] Al-Nakeeb, K., Petersen, T. N., and Sicheritz-Pontén, T. (2017). Norgal: extraction and de novo assembly of mitochondrial DNA from whole-genome sequencing data. *BMC Bioinformatics*, 18(1):510.

[2] Allman, E. S., Rhodes, J. A., and Sullivant, S. (2017). Statistically Consistent k -mer Methods for Phylogenetic Tree Reconstruction. *Journal of Computational Biology*, 24(2):153–171.

[3] Alqahtani, F. and Măndoiu, I. (2020). SMART2: Multi-library Statistical Mitogenome Assembly with Repeats BT - Computational Advances in Bio and Medical Sciences. pages 184–198, Cham. Springer International Publishing.

[4] 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):403–410.

[5] Ames, S. K., Hysom, D. A., Gardner, S. N., Lloyd, G. S., Gokhale, M. B., and Allen, J. E. (2013). Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics (Oxford, England)*, 29(18):2253–2260.

[6] Andoni, A., Indyk, P., Nguyen, H. L., and Razenshteyn, I. (2014). Beyond Locality-Sensitive Hashing. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1018–1028, Philadelphia, PA. Society for Industrial and Applied Mathematics.

[7] Angiuoli, S. V. and Salzberg, S. L. (2011). Mugsy: fast multiple alignment of closely related whole genomes. *Bioinformatics*, 27(3):334–342.

[8] Antipov, D., Hartwick, N., Shen, M., Raiko, M., Lapidus, A., and Pevzner, P. A. (2016). plasmidSPAdes: assembling plasmids from whole genome sequencing data. *Bioinformatics*, 32(22):3380–3387.

[9] Arora, S., Liang, Y., and Ma, T. (2017). A Simple but Tough-to-Beat Baseline for Sentence Embeddings. In *ICLR*.

[10] Artamonova, I. I. and Mushegian, A. R. (2013). Genome sequence analysis indicates

that the model eukaryote Nematostella vectensis harbors bacterial consorts. *Applied and environmental microbiology*, 79(22):6868–6873.

[11] Asnicar, F., Thomas, A. M., Beghini, F., Mengoni, C., Manara, S., Manghi, P., Zhu, Q., Bolzan, M., Cumbo, F., May, U., Sanders, J. G., Zolfo, M., Kopylova, E., Pasolli, E., Knight, R., Mirarab, S., Huttenhower, C., and Segata, N. (2020). Precise phylogenetic analysis of microbial isolates and genomes from metagenomes using PhyloPhlAn 3.0. *Nature Communications*, 11(1):2500.

[12] Balaban, M., Bristy, N. A., Faisal, A., Bayzid, M. S., and Mirarab, S. (2021a). Genome-wide alignment-free phylogenetic distance estimation under a no strand-bias model. *bioRxiv*, page 2021.11.10.468111.

[13] Balaban, M., Jiang, Y., Zhu, Q., McDonald, D., Knight, R., and Mirarab, S. (2023). Generation of accurate, expandable phylogenomic trees with uDance. *Nature Biotechnology*.

[14] Balaban, M. and Mirarab, S. (2020). Phylogenetic double placement of mixed samples. *Bioinformatics*, 36(Supplement_1):i335–i343.

[15] Balaban, M., Moshiri, N., Mai, U., Jia, X., and Mirarab, S. (2019a). TreeCluster: Clustering biological sequences using phylogenetic trees. *PLOS ONE*, 14(8):e0221068.

[16] Balaban, M., Roush, D., Zhu, Q., and Mirarab, S. (2021b). APPLES-2 : Faster and More Accurate Distance-based Phylogenetic Placement using Divide and Conquer. *bioRxiv preprint*, page 2021.02.14.431150.

[17] Balaban, M., Sarmashghi, S., and Mirarab, S. (2019b). APPLES: Scalable Distance-Based Phylogenetic Placement with or without Alignments. *Systematic Biology*.

[18] Balaban, M., Sarmashghi, S., and Mirarab, S. (2020). APPLES: Scalable Distance-Based Phylogenetic Placement with or without Alignments. *Systematic Biology*, 69(3):566–578.

[19] Ballenghien, M., Faivre, N., and Galtier, N. (2017). Patterns of cross-contamination in a multispecies population genomic project: detection, quantification, impact, and solutions. *BMC biology*, 15(1):25.

[20] Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., Pyshkin, A. V., Sirotkin, A. V., Vyahhi, N., Tesler, G., Alekseyev, M. A., and Pevzner, P. A. (2012). SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology : a journal of computational molecular cell biology*, 19(5):455–477.

[21] Barbera, P., Kozlov, A. M., Czech, L., Morel, B., Darriba, D., Flouri, T., and Stamatakis, A. (2019). EPA-ng: Massively Parallel Evolutionary Placement of Genetic Sequences. *Systematic*

Biology, 68(2):365–369.

[22] Bengio, S. and Heigold, G. (2014). Word embeddings for speech recognition. In *Interspeech 2014*, pages 1053–1057, ISCA. ISCA.

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

[24] Berlin, K., Koren, S., Chin, C.-S., Drake, J. P., Landolin, J. M., and Phillippy, A. M. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623–630.

[25] Bernard, G., Chan, C. X., and Ragan, M. A. (2016). Alignment-free microbial phylogenomics under scenarios of sequence divergence, genome rearrangement and lateral genetic transfer. *Scientific Reports*, 6(1):28970.

[26] Bernt, M., Donath, A., Jühling, F., Externbrink, F., Florentz, C., Fritzsch, G., Pütz, J., Middendorf, M., and Stadler, P. F. (2013). MITOS: Improved de novo metazoan mitochondrial genome annotation. *Molecular Phylogenetics and Evolution*, 69(2):313–319.

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

[28] Bharti, R. and Grimm, D. G. (2021). Current challenges and best-practice protocols for microbiome analysis. *Briefings in Bioinformatics*, 22(1):178–193.

[29] Bogusz, M. and Whelan, S. (2016). Phylogenetic Tree Estimation With and Without Alignment: New Distance Methods and Benchmarking. *Systematic Biology*, 66(2):218–231.

[30] Bohmann, K., Mirarab, S., Bafna, V., and Gilbert, M. T. P. (2020). Beyond DNA barcoding: The unrealized potential of genome skim data in sample identification. *Molecular Ecology*, 29(14):2521–2534.

[31] Bonham-Carter, O., Steele, J., and Bastola, D. (2014). Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis. *Briefings in Bioinformatics*, 15(6):890–905.

[32] Boore, J. L. (1999). Animal mitochondrial genomes. *Nucleic Acids Research*, 27(8):1767–1780.

[33] Boyd, B. M., Allen, J. M., Nguyen, N.-P., Sweet, A. D., Warnow, T., Shapiro, M. D., Villa, S. M., Bush, S. E., Clayton, D. H., and Johnson, K. P. (2017). Phylogenomics using Target-Restricted Assembly Resolves Intrageneric Relationships of Parasitic Lice (Phthiraptera: Columbicola). *Systematic Biology*, 66(6):896–911.

[34] Breitwieser, F. P., Baker, D. N., and Salzberg, S. L. (2018). KrakenUniq: confident and fast metagenomics classification using unique k-mer counts. *Genome Biology*, 19(1):198.

[35] Breitwieser, F. P., Pertea, M., Zimin, A. V., and Salzberg, S. L. (2019). Human contamination in bacterial genomes has created thousands of spurious proteins. *Genome research*, 29(6):954–960.

[36] Brinda, K., Sykulski, M., and Kucherov, G. (2015). Spaced seeds improve k-mer-based metagenomic classification. *Bioinformatics (Oxford, England)*, 31(22):3584–3592.

[37] Broder, A. Z. (1997). On the resemblance and containment of documents. *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*, pages 21–29.

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

[39] Buhler, J. (2001). Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428.

[40] Bushnell, B., Rood, J., and Singer, E. (2017). BBMerge – Accurate paired shotgun read merging via overlap. *PLOS ONE*, 12(10):1–15.

[41] Calabrese, C., Simone, D., Diroma, M. A., Santorsola, M., Guttà, C., Gasparre, G., Picardi, E., Pesole, G., and Attimonelli, M. (2014). MToolBox: a highly automated pipeline for heteroplasmy annotation and prioritization analysis of human mitochondrial variants in high-throughput sequencing. *Bioinformatics (Oxford, England)*, 30(21):3115–3117.

[42] Chen, L., Qiu, Q., Jiang, Y., Wang, K., Lin, Z., Li, Z., Bibi, F., Yang, Y., Wang, J., Nie, W., Su, W., Liu, G., Li, Q., Fu, W., Pan, X., Liu, C., Yang, J., Zhang, C., Yin, Y., Wang, Y., Zhao, Y., Zhang, C., Wang, Z., Qin, Y., Liu, W., Wang, B., Ren, Y., Zhang, R., Zeng, Y., da Fonseca, R. R., Wei, B., Li, R., Wan, W., Zhao, R., Zhu, W., Wang, Y., Duan, S., Gao, Y., Zhang, Y. E., Chen, C., Hvilsom, C., Epps, C. W., Chemnick, L. G., Dong, Y., Mirarab, S., Siegismund, H. R., Ryder, O. A., Gilbert, M. T. P., Lewin, H. A., Zhang, G., Heller, R., and Wang, W. (2019). Large-scale ruminant genome sequencing provides insights into their evolution and distinct traits. *Science (New York, N.Y.)*, 364(6446).

[43] Chesters, D. (2016). Construction of a Species-Level Tree of Life for the Insects and Utility in Taxonomic Profiling. *Systematic Biology*, page syw099.

[44] Choi, J., Yang, F., Stepanauskas, R., Cardenas, E., Garoutte, A., Williams, R., Flater, J., Tiedje, J. M., Hofmockel, K. S., Gelder, B., and Howe, A. (2017). Strategies to improve reference databases for soil microbiomes. *The ISME Journal*, 11(4):829–834.

[45] 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.

[46] Cornet, L., Meunier, L., Van Vlierberghe, M., Léonard, R. R., Durieu, B., Lara, Y., Misztak, A., Sirjacobs, D., Javaux, E. J., Philippe, H., Wilmotte, A., and Baurain, D. (2018). Consensus assessment of the contamination level of publicly available cyanobacterial genomes. *PloS one*, 13(7):e0200323.

[47] Criscuolo, A. (2019). A fast alignment-free bioinformatics procedure to infer accurate distance-based phylogenetic trees from genome assemblies. *Research Ideas and Outcomes*, 5.

[48] Darling, A. C. E., Mau, B., Blattner, F. R., and Perna, N. T. (2004). Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome research*, 14(7):1394–1403.

[49] Darling, A. E., Mau, B., and Perna, N. T. (2010). progressiveMauve: Multiple Genome Alignment with Gene Gain, Loss and Rearrangement. *PLoS ONE*, 5(6):e11147.

[50] Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry - SCG '04*, page 253, New York, New York, USA. ACM Press.

[51] de Vienne, D. M., Ollier, S., and Aguileta, G. (2012). Phylo-MCOA: A Fast and Efficient Method to Detect Outlier Genes and Species in Phylogenomics Using Multiple Co-inertia Analysis. *Molecular Biology and Evolution*, 29(6):1587–1598.

[52] Degnan, J. H. and Salter, L. A. (2005). Gene tree distributions under the coalescent process. *Evolution*, 59(1):24–37.

[53] Denver, D. R., Brown, A. M. V., Howe, D. K., Peetz, A. B., and Zasada, I. A. (2016). Genome Skimming: A Rapid Approach to Gaining Diverse Biological Insights into Multicellular Pathogens. *PLOS Pathogens*, 12(8):e1005713.

[54] Deorowicz, S., Kokot, M., Grabowski, S., and Debudaj-Grabysz, A. (2015). KMC 2: fast and resource-frugal k-mer counting. *Bioinformatics*, 31(10):1569–1576.

[55] DeSantis, T. Z., Hugenholtz, P., Larsen, N., Rojas, M., Brodie, E. L., Keller, K., Huber, T., Dalevi, D., Hu, P., and Andersen, G. L. (2006). Greengenes, a Chimera-Checked 16S rRNA Gene Database and Workbench Compatible with ARB. *Appl. Environ. Microbiol.*, 72(7):5069–5072.

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

[57] Dierckxsens, N., Mardulyn, P., and Smits, G. (2017). NOVOPlasty: de novo assembly of

organelle genomes from whole genome data. *Nucleic Acids Research*, 45(4):e18–e18.

[58] Dittami, S. M. and Corre, E. (2017). Detection of bacterial contaminants and hybrid sequences in the genome of the kelp Saccharina japonica using Taxoblast. *PeerJ*, 5:e4073–e4073.

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

[60] Dress, A. W., Flamm, C., Fritzsch, G., Grünewald, S., Kruspe, M., Prohaska, S. J., and Stadler, P. F. (2008). Noisy: Identification of problematic columns in multiple sequence alignments. *Algorithms for Molecular Biology*, 3(1):7.

[61] Earl, D., Nguyen, N., Hickey, G., Harris, R. S., Fitzgerald, S., Beal, K., Seledtsov, I., Molodtsov, V., Raney, B. J., Clawson, H., Kim, J., Kemena, C., Chang, J.-M., Erb, I., Poliakov, A., Hou, M., Herrero, J., Kent, W. J., Solovyev, V., Darling, A. E., Ma, J., Notredame, C., Brudno, M., Dubchak, I., Haussler, D., and Paten, B. (2014). Alignathon: a competitive assessment of whole-genome alignment methods. *Genome Research*, 24(12):2077–2089.

[62] Eddy, S. R. (2004). What is dynamic programming? *Nature Biotechnology*, 22(7):909–910.

[63] Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26.

[64] 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.

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

[66] Felsenstein, J. (1985). Confidence Limits on Phylogenies: An Approach Using the Bootstrap. *Evolution*, 39(4):783–791.

[67] Felsenstein, J. and Kishino, H. (1993). Is there something wrong with the bootstrap on phylogenies? A reply to Hillis and Bull. *Systematic Biology*, 42(2):193–200.

[68] Fitch, W. M. (1970). Distinguishing homologous from analogous proteins. *Systematic zoology*, 19(2):99–113.

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

[70] Fletcher, W. and Yang, Z. (2009). INDELible: A flexible simulator of biological sequence

evolution. *Molecular Biology and Evolution*, 26(8):1879–1888.

[71] Francois, C. M., Durand, F., Figuet, E., and Galtier, N. (2020). Prevalence and Implications of Contamination in Public Genomic Resources: A Case Study of 43 Reference Arthropod Assemblies. *G3*, 10(2):721–730.

[72] Freudenthal, J. A., Pfaff, S., Terhoeven, N., Korte, A., Ankenbrand, M. J., and Förster, F. (2020). A systematic comparison of chloroplast genome assembly tools. *Genome Biology*, 21(1):254.

[73] Gascuel, O. (2000). On the Optimization Principle in Phylogenetic Analysis and the Minimum-Evolution Criterion. *Molecular Biology and Evolution*, 17(3):401–405.

[74] Georgiou, A., Fortuin, V., Mustafa, H., and Rätsch, G. (2019). META: Memory-efficient taxonomic classification and abundance estimation for metagenomics with deep learning.

[75] Glassing, A., Dowd, S. E., Galandiuk, S., Davis, B., and Chiodini, R. J. (2016). Inherent bacterial DNA contamination of extraction and sequencing reagents may affect interpretation of microbiota in low bacterial biomass samples. *Gut Pathogens*, 8(1):24.

[76] Gorisse, D., Cord, M., and Precioso, F. (2012). Locality-Sensitive Hashing for Chi2 Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):402–409.

[77] Hahn, C., Bachmann, L., and Chevreux, B. (2013). Reconstructing mitochondrial genomes directly from genomic next-generation sequencing reads—a baiting and iterative mapping approach. *Nucleic Acids Research*, 41(13):e129–e129.

[78] Har-Peled, S., Indyk, P., and Motwani, R. (2012). Approximate nearest neighbors: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350.

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

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

[81] Hickerson, M. J., Meyer, C. P., Moritz, C., and Hedin, M. (2006). DNA Barcoding Will Often Fail to Discover New Animal Species over Broad Parameter Space. *Systematic Biology*, 55(5):729–739.

[82] Hill, D. J. (2005). Lichens: an Illustrated Guide to the British and Irish Species. By Frank Dobson. Fifth Edition. Slough: Richmond Publishing Co Ltd. Pp. 480 with numerous keys, colour illustrations, thumbnail drawings and distribution maps. ISBN 0 85546 095 4 hardback;

IS. *The Lichenologist*, 37(6):571–572.

[83] Hillis, D. M. and Bull, J. J. (1993). An Empirical Test of Bootstrapping as a Method for Assessing Confidence in Phylogenetic Analysis. *Systematic Biology*, 42(2):182–192.

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

[85] Höhl, M. and Ragan, M. A. (2007). Is multiple-sequence alignment required for accurate inference of phylogeny? *Systematic Biology*, 56(2):206–221.

[86] Holder, M. and Lewis, P. O. (2003). Phylogeny estimation: traditional and Bayesian approaches. *Nature Reviews Genetics*, 4(4):275–284.

[87] Hollingsworth, P. M., Forrest, L. L., Spouge, J. L., Hajibabaei, M., Ratnasingham, S., van der Bank, M., Chase, M. W., Cowan, R. S., Erickson, D. L., Fazekas, A. J., Graham, S. W., James, K. E., Kim, K.-J., Kress, W. J., Schneider, H., van AlphenStahl, J., Barrett, S. C., van den Berg, C., Bogarin, D., Burgess, K. S., Cameron, K. M., Carine, M., Chacon, J., Clark, A., Clarkson, J. J., Conrad, F., Devey, D. S., Ford, C. S., Hedderson, T. A., Hollingsworth, M. L., Husband, B. C., Kelly, L. J., Kesanakurti, P. R., Kim, J. S., Kim, Y.-D., Lahaye, R., Lee, H.-L., Long, D. G., Madrinan, S., Maurin, O., Meusnier, I., Newmaster, S. G., Park, C.-W., Percy, D. M., Petersen, G., Richardson, J. E., Salazar, G. A., Savolainen, V., Seberg, O., Wilkinson, M. J., Yi, D.-K., and Little, D. P. (2009). A DNA barcode for land plants. *Proceedings of the National Academy of Sciences*, 106(31):12794–12797.

[88] Huang, W., Li, L., Myers, J. R., and Marth, G. T. (2012a). ART: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594.

[89] Huang, W., Li, L., Myers, J. R., and Marth, G. T. (2012b). ART: A next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594.

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

[91] Jeffroy, O., Brinkmann, H., Delsuc, F., and Philippe, H. (2006). Phylogenomics: the beginning of incongruence? *Trends in Genetics*, 22(4):225–231.

[92] Jiang, Y., Balaban, M., Zhu, Q., and Mirarab, S. (2021). DEPP: Deep Learning Enables

Extending Species Trees using Single Genes. *bioRxiv (abstract in RECOMB 2021)*, page 2021.01.22.427808.

[93] Jiang, Y., Balaban, M., Zhu, Q., and Mirarab, S. (2022). DEPP: Deep Learning Enables Extending Species Trees using Single Genes. *bioRxiv*, page 2021.01.22.427808.

[94] Jiang, Y., Balaban, M., Zhu, Q., and Mirarab, S. (2023a). DEPP: Deep Learning Enables Extending Species Trees using Single Genes. *Systematic Biology*, 72(1):17–34.

[95] Jiang, Y., McDonald, D., Knight, R., and Mirarab, S. (2023b). Scaling deep phylogenetic embedding to ultra-large reference trees: a tree-aware ensemble approach. *bioRxiv*.

[96] Jin, J.-J., Yu, W.-B., Yang, J.-B., Song, Y., DePamphilis, C. W., Yi, T.-S., and Li, D.-Z. (2019). GetOrganelle: a fast and versatile toolkit for accurate de novo assembly of organelle genomes. *bioRxiv*.

[97] Jin, J.-J., Yu, W.-B., Yang, J.-B., Song, Y., DePamphilis, C. W., Yi, T.-S., and Li, D.-Z. (2020). GetOrganelle: a fast and versatile toolkit for accurate de novo assembly of organelle genomes. *Genome Biology*, 21(1):241.

[98] Jin, L. and Nei, M. (1990). Limitations of the evolutionary parsimony method of phylogenetic analysis. *Molecular Biology and Evolution*, 7(1):82–102.

[99] Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. (2017). Bag of Tricks for Efficient Text Classification. In *EACL*.

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

[101] Jun, S.-R., Sims, G. E., Wu, G. A., and Kim, S.-H. (2010). Whole-proteome phylogeny of prokaryotes by feature frequency profiles: An alignment-free method with optimal feature resolution. *Proceedings of the National Academy of Sciences*, 107(1):133–138.

[102] Junier, T. and Zdobnov, E. M. (2010). The Newick utilities: high-throughput phylogenetic tree processing in the Unix shell. *Bioinformatics*, 26(13):1669–1670.

[103] Kent, W. J. (2002). BLAT–the BLAST-like alignment tool. *Genome research*, 12(4):656–664.

[104] Kent, W. J., Baertsch, R., Hinrichs, A., Miller, W., and Haussler, D. (2003). Evolution's cauldron: Duplication, deletion, and rearrangement in the mouse and human genomes. *Proceedings of the National Academy of Sciences*, 100(20):11484–11489.

[105] Kim, D., Song, L., Breitwieser, F. P., and Salzberg, S. L. (2016). Centrifuge: rapid and

sensitive classification of metagenomic sequences. *Genome research*, 26(12):1721–1729.

[106] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.

[107] Koutsovoulos, G., Kumar, S., Laetsch, D. R., Stevens, L., Daub, J., Conlon, C., Maroon, H., Thomas, F., Aboobaker, A. A., and Blaxter, M. (2016). No evidence for extensive horizontal gene transfer in the genome of the tardigrade Hypsibius dujardini. *Proceedings of the National Academy of Sciences of the United States of America*, 113(18):5053–5058.

[108] Kulis, B. and Grauman, K. (2012). Kernelized Locality-Sensitive Hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(6):1092–1104.

[109] Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359.

[110] Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):R25.

[111] Langmead, B., Wilks, C., Antonescu, V., and Charles, R. (2019). Scaling read aligners to hundreds of threads on general-purpose processors. *Bioinformatics*, 35(3):421–432.

[112] Lau, A.-K., Dörrer, S., Leimeister, C.-A., Bleidorn, C., and Morgenstern, B. (2019). Read-SpaM: assembly-free and alignment-free comparison of bacterial genomes with low sequencing coverage. *BMC Bioinformatics*, 20(S20):638.

[113] Laurin-Lemay, S., Brinkmann, H., and Philippe, H. (2012). Origin of land plants revisited in the light of sequence contamination and missing data. *Current Biology*.

[114] Layer, M. and Rhodes, J. A. (2017). Phylogenetic trees and Euclidean embeddings. *Journal of Mathematical Biology*, 74(1-2):99–111.

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

[116] Lefort, V., Desper, R., and Gascuel, O. (2015b). FastME 2.0: A Comprehensive, Accurate, and Fast Distance-Based Phylogeny Inference Program. *Molecular Biology and Evolution*, 32(10):2798–2800.

[117] Leimeister, C.-A. and Morgenstern, B. (2014). Kmacs: the k-mismatch average common substring approach to alignment-free sequence comparison. *Bioinformatics*, 30(14):2000–8.

[118] Leimeister, C.-A., Sohrabi-Jahromi, S., and Morgenstern, B. (2017). Fast and accurate

phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics*, 33(7):btw776.

[119] Lemoine, F., Domelevo Entfellner, J.-B., Wilkinson, E., Correia, D., Dávila Felipe, M., De Oliveira, T., and Gascuel, O. (2018). Renewing Felsenstein's phylogenetic bootstrap in the era of big data. *Nature*, 556(7702):452–456.

[120] Li, F.-W., Brouwer, P., Carretero-Paulet, L., Cheng, S., de Vries, J., Delaux, P.-M., Eily, A., Koppers, N., Kuo, L.-Y., Li, Z., Simenc, M., Small, I., Wafula, E., Angarita, S., Barker, M. S., Bräutigam, A., DePamphilis, C., Gould, S., Hosmani, P. S., Huang, Y.-M., Huettel, B., Kato, Y., Liu, X., Maere, S., McDowell, R., Mueller, L. A., Nierop, K. G. J., Rensing, S. A., Robison, T., Rothfels, C. J., Sigel, E. M., Song, Y., Timilsena, P. R., Van de Peer, Y., Wang, H., Wilhelmsson, P. K. I., Wolf, P. G., Xu, X., Der, J. P., Schluepmann, H., Wong, G. K., and Pryer, K. M. (2018). Fern genomes elucidate land plant evolution and cyanobacterial symbioses. *Nature Plants*, 4(7):460–472.

[121] Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM.

[122] Li, H. (2018). Seqtk, toolkit for processing sequences in FASTA/Q formats.

[123] Li, Y., Steenwyk, J. L., Chang, Y., Wang, Y., James, T. Y., Stajich, J. E., Spatafora, J. W., Groenewald, M., Dunn, C. W., Hittinger, C. T., Shen, X.-X., and Rokas, A. (2021). A genome-scale phylogeny of the kingdom Fungi. *Current Biology*, 31(8):1653–1665.e5.

[124] Liang, Q., Bible, P. W., Liu, Y., Zou, B., and Wei, L. (2020). DeepMicrobes: taxonomic classification for metagenomics with deep learning. *NAR Genomics and Bioinformatics*, 2(1).

[125] Liu, B., Gibbons, T., Ghodsi, M., and Pop, M. (2010). MetaPhyler: Taxonomic profiling for metagenomic sequences. *Proceedings - 2010 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2010*, pages 95–100.

[126] Locey, K. J. and Lennon, J. T. (2016). Scaling laws predict global microbial diversity. *Proceedings of the National Academy of Sciences*, 113(21):5970–5975.

[127] Lu, J., Breitwieser, F. P., Thielen, P., and Salzberg, S. L. (2017a). Bracken: estimating species abundance in metagenomics data. *PeerJ Computer Science*, 3:e104.

[128] Lu, J. and Salzberg, S. L. (2018). Removing contaminants from databases of draft genomes. *PLoS Computational Biology*, 14(6):1–13.

[129] Lu, Y. Y., Tang, K., Ren, J., Fuhrman, J. A., Waterman, M. S., and Sun, F. (2017b). CAFE: aCcelerated Alignment-FrEe sequence analysis. *Nucleic Acids Research*, 45(W1):W554–W559.

[130] Luo, Y., Yu, Y. W., Zeng, J., Berger, B., and Peng, J. (2019). Metagenomic binning through low-density hashing. *Bioinformatics*, 35(2):219–226.

[131] Lusk, R. W. (2014). Diverse and Widespread Contamination Evident in the Unmapped Depths of High Throughput Sequencing Data. *PLOS ONE*, 9(10):e110808.

[132] Maas, A., Daly, R., Pham, P., Huang, D., Ng, A., and Potts, C. (2011). *Learning Word Vectors for Sentiment Analysis*.

[133] Maddison, W. (1989). Reconstructing character evolution on polytomous cladograms. *Cladistics*, 5(4):365–377.

[134] Maddison, W. P. (1997). Gene Trees in Species Trees. *Systematic Biology*, 46(3):523–536.

[135] Maidak, B. L. (2001). The RDP-II (Ribosomal Database Project). *Nucleic Acids Research*, 29(1):173–174.

[136] Malé, P.-J. G., Bardon, L., Besnard, G., Coissac, E., Delsuc, F., Engel, J., Lhuillier, E., Scotti-Saintagne, C., Tinaut, A., and Chave, J. (2014). Genome skimming by shotgun sequencing helps resolve the phylogeny of a pantropical tree family. *Molecular ecology resources*, 14(5):966–75.

[137] Marçais, G. (2013). Jellyfish 2 User Guide.

[138] Marçais, G., DeBlasio, D., Pandey, P., and Kingsford, C. (2019). Locality-sensitive hashing for the edit distance. *Bioinformatics*, 35(14):i127–i135.

[139] Marçais, G. and Kingsford, C. (2011a). A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770.

[140] Marçais, G. and Kingsford, C. (2011b). A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770.

[141] Margaryan, A., Noer, C. L., Richter, S. R., Restrup, M. E., Bülow-Hansen, J. L., Leerhøi, F., Langkjær, E. M. R., Gopalakrishnan, S., Carøe, C., Gilbert, M. T. P., and Bohmann, K. (2020). Mitochondrial genomes of Danish vertebrate species generated for the national DNA reference database, DNAmark. *Environmental DNA*, n/a(n/a).

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

[143] McCombie, W. R. and McPherson, J. D. (2019). Future Promises and Concerns of Ubiquitous Next-Generation Sequencing. *Cold Spring Harbor Perspectives in Medicine*,

9(9):a025783.

[144] McGowen, M. R., Tsagkogeorga, G., Álvarez-Carretero, S., Dos Reis, M., Struebig, M., Deaville, R., Jepson, P. D., Jarman, S., Polanowski, A., Morin, P. A., and Rossiter, S. J. (2020). Phylogenomic Resolution of the Cetacean Tree of Life Using Target Sequence Capture. *Systematic biology*, 69(3):479–501.

[145] McHardy, A. C., Martín, H. G., Tsirigos, A., Hugenholtz, P., and Rigoutsos, I. (2007). Accurate phylogenetic classification of variable-length DNA fragments. *Nature Methods*, 4(1):63–72.

[146] McIntyre, A. B. R., Ounit, R., Afshinnekoo, E., Prill, R. J., Hénaff, E., Alexander, N., Minot, S. S., Danko, D., Foox, J., Ahsanuddin, S., Tighe, S., Hasan, N. A., Subramanian, P., Moffat, K., Levy, S., Lonardi, S., Greenfield, N., Colwell, R. R., Rosen, G. L., and Mason, C. E. (2017). Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. *Genome Biology*, 18(1):182.

[147] Menegaux, R. and Vert, J.-P. (2019). Continuous Embeddings of DNA Sequencing Reads and Application to Metagenomics. *Journal of Computational Biology*, 26(6):509–518.

[148] Meng, G., Li, Y., Yang, C., and Liu, S. (2019). MitoZ: a toolkit for animal mitochondrial genome assembly, annotation and visualization. *Nucleic Acids Research*, 47(11):e63–e63.

[149] Merchant, S., Wood, D. E., and Salzberg, S. L. (2014). Unexpected cross-species contamination in genome sequencing projects. *PeerJ*, 2:e675.

[150] Metsky, H. C., Siddle, K. J., Gladden-Young, A., Qu, J., Yang, D. K., Brehio, P., Goldfarb, A., Piantadosi, A., Wohl, S., Carter, A., Lin, A. E., Barnes, K. G., Tully, D. C., Corleis, B., Hennigan, S., Barbosa-Lima, G., Vieira, Y. R., Paul, L. M., Tan, A. L., Garcia, K. F., Parham, L. A., Odia, I., Eromon, P., Folarin, O. A., Goba, A., Simon-Lorière, E., Hensley, L., Balmaseda, A., Harris, E., Kwon, D. S., Allen, T. M., Runstadler, J. A., Smole, S., Bozza, F. A., Souza, T. M. L., Isern, S., Michael, S. F., Lorenzana, I., Gehrke, L., Bosch, I., Ebel, G., Grant, D. S., Happi, C. T., Park, D. J., Gnirke, A., Sabeti, P. C., and Matranga, C. B. (2019). Capturing sequence diversity in metagenomes with comprehensive and scalable probe design. *Nature Biotechnology*, 37(2):160–168.

[151] Meyer, F., Bremges, A., Belmann, P., Janssen, S., McHardy, A. C., and Koslicki, D. (2019). Assessing taxonomic metagenome profilers with OPAL. *Genome Biology*, 20(1):51.

[152] Mikheenko, A., Prjibelski, A., Saveliev, V., Antipov, D., and Gurevich, A. (2018). Versatile genome assembly evaluation with QUAST-LG. *Bioinformatics*, 34(13):i142–i150.

[153] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality.

[154] Milanese, A., Mende, D. R., Paoli, L., Salazar, G., Ruscheweyh, H.-J., Cuenca, M., Hingamp, P., Alves, R., Costea, P. I., Coelho, L. P., Schmidt, T. S. B., Almeida, A., Mitchell, A. L., Finn, R. D., Huerta-Cepas, J., Bork, P., Zeller, G., and Sunagawa, S. (2019). Microbial abundance, activity and population genomic profiling with mOTUs2. *Nature Communications*, 10(1):1014.

[155] Miller, D. E., Staber, C., Zeitlinger, J., and Hawley, R. S. (2018). Highly contiguous genome assemblies of 15 drosophila species generated using nanopore sequencing. *G3: Genes, Genomes, Genetics*, 8(10):3131–3141.

[156] Mirarab, S., Bayzid, M., Boussau, B., and Warnow, T. (2015). Response to Comment on "Statistical binning enables an accurate coalescent-based estimation of the avian tree". *Science*, 350(6257).

[157] Mirarab, S., Bayzid, M. S., and Warnow, T. (2016). Evaluating Summary Methods for Multilocus Species Tree Estimation in the Presence of Incomplete Lineage Sorting. *Systematic Biology*, 65(3):366–380.

[158] Mirarab, S., Nguyen, N.-P., and Warnow, T. (2012). SEPP: SATé-enabled phylogenetic placement. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 247–258.

[159] Morgulis, A., Coulouris, G., Raytselis, Y., Madden, T. L., Agarwala, R., and Schäffer, A. A. (2008). Database indexing for production MegaBLAST searches. *Bioinformatics*, 24(16):1757–1764.

[160] Moshiri, N. (2020). TreeSwift: A massively scalable Python tree package. *SoftwareX*, 11:100436.

[161] Narayanan, M. and Karp, R. M. (2004). Gapped Local Similarity Search with Provable Guarantees. pages 74–86.

[162] Nasko, D. J., Koren, S., Phillippy, A. M., and Treangen, T. J. (2018). RefSeq database growth influences the accuracy of k-mer-based lowest common ancestor species identification. *Genome Biology*, 19(1):165.

[163] Nevill, P. G., Zhong, X., Tonti-Filippini, J., Byrne, M., Hislop, M., Thiele, K., van Leeuwen, S., Boykin, L. M., and Small, I. (2020). Large scale genome skimming from herbarium material for accurate plant identification and phylogenomics. *Plant Methods*, 16(1):1.

[164] Ng, P. (2017). dna2vec: Consistent vector representations of variable-length k-mers. *ArXiv*, abs/1701.0.

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

[166] Ochman, H., Lawrence, J. G., and Groisman, E. A. (2000). Lateral gene transfer and the nature of bacterial innovation. *Nature*, 405(6784):299–304.

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

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

[169] OpenMP (2018). OpenMP Application Programming Interface Version 5.0.

[170] Orakov, A., Fullam, A., Coelho, L. P., Khedkar, S., Szklarczyk, D., Mende, D. R., Schmidt, T. S. B., and Bork, P. (2021). GUNC: detection of chimerism and contamination in prokaryotic genomes. *Genome Biology*, 22(1):178.

[171] Ounit, R. and Lonardi, S. (2016). Higher classification sensitivity of short metagenomic reads with CLARK-S. *Bioinformatics (Oxford, England)*, 32(24):3823–3825.

[172] Ounit, R., Wanamaker, S., Close, T. J., and Lonardi, S. (2015). CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics*, 16(1):236.

[173] Pachiadaki, M. G., Brown, J. M., Brown, J., Bezuidt, O., Berube, P. M., Biller, S. J., Poulton, N. J., Burkart, M. D., La Clair, J. J., Chisholm, S. W., and Stepanauskas, R. (2019). Charting the Complexity of the Marine Microbiome through Single-Cell Genomics. *Cell*, 179(7):1623–1635.e11.

[174] Pandey, P., Almodaresi, F., Bender, M. A., Ferdman, M., Johnson, R., and Patro, R. (2018). Mantis: A Fast, Small, and Exact Large-Scale Sequence-Search Index. *Cell Systems*, 7(2):201–207.e4.

[175] Parks, D. H., Chuvochina, M., Chaumeil, P.-A., Rinke, C., Mussig, A. J., and Hugenholtz, P. (2020). A complete domain-to-species taxonomy for Bacteria and Archaea. *Nature Biotechnology*, 38(9):1079–1086.

[176] Parks, D. H., Imelfort, M., Skennerton, C. T., Hugenholtz, P., and Tyson, G. W. (2015). CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome Research*, 25(7):1043–1055.

[177] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Wallach, H., Larochelle, H., Beygelzimer, A., d\textquotesingle Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[178] Peabody, M. A., Van Rossum, T., Lo, R., and Brinkman, F. S. L. (2015). Evaluation of shotgun metagenomics sequence classification methods using in silico and in vitro simulated communities. *BMC Bioinformatics*, 16(1):362.

[179] Pevzner, P. and Tesler, G. (2003). Genome Rearrangements in Mammalian Evolution: Lessons From Human and Mouse Genomes. *Genome Research*, 13(1):37–45.

[180] Pham, T. D. and Zuegg, J. (2004). A probabilistic measure for alignment-free sequence comparison. *Bioinformatics*, 20(18):3455–3461.

[181] Philippe, H., de Vienne, D. M., Ranwez, V., Roure, B., Baurain, D., and Delsuc, F. (2017). Pitfalls in supermatrix phylogenomics. *European Journal of Taxonomy*, (283).

[182] Phillips, M. J., Delsuc, F., and Penny, D. (2004). Genome-scale phylogeny and the detection of systematic biases. *Molecular Biology and Evolution*.

[183] Politis, D. N., Romano, J. P., and Wolf, M. (1999). *Subsampling*. Springer Science & Business Media.

[184] Qi, Y., Sachan, D. S., Felix, M., Padmanabhan, S. J., and Neubig, G. (2018). When and Why are Pre-trained Word Embeddings Useful for Neural Machine Translation?

[185] Quast, C., Pruesse, E., Yilmaz, P., Gerken, J., Schweer, T., Yarza, P., Peplies, J., and Glöckner, F. O. (2012). The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic acids research*, page gks1219.

[186] Quicke, D. L. J., Smith, M. A., Janzen, D. H., Hallwachs, W., Fernandez-Triana, J., Laurenne, N. M., Zaldívar-Riverón, A., Shaw, M. R., Broad, G. R., Klopfstein, S., Shaw, S. R., Hrcek, J., Hebert, P. D. N., Miller, S. E., Rodriguez, J. J., Whitfield, J. B., Sharkey, M. J., Sharanowski, B. J., Jussila, R., Gauld, I. D., Chesters, D., and Vogler, A. P. (2012). Utility of the DNA barcoding gene fragment for parasitic wasp phylogeny (Hymenoptera: Ichneumonoidea): data release and new measure of taxonomic congruence. *Molecular ecology resources*, 12(4):676–85.

[187] Rachtman, E., Bafna, V., and Mirarab, S. (2021). CONSULT: accurate contamination removal using locality-sensitive hashing. *NAR Genomics and Bioinformatics*, 3(3):2631–9268.

[188] Rachtman, E., Balaban, M., Bafna, V., and Mirarab, S. (2019). [dataset] kraken_raw_data_v2.

[189] Rachtman, E., Balaban, M., Bafna, V., and Mirarab, S. (2020). The impact of contaminants on the accuracy of genome skimming and the effectiveness of exclusion read filters. *Molecular Ecology Resources*, 20(3):1755–0998.

[190] Rasheed, Z., Rangwala, H., and Barbara, D. (2012). LSH-Div: Species diversity estimation using locality sensitive hashing. In *2012 IEEE International Conference on Bioinformatics and Biomedicine*, pages 1–6. IEEE.

[191] Rasheed, Z., Rangwala, H., and Barbará, D. (2013). 16S rRNA metagenome clustering and diversity estimation using locality sensitive hashing. *BMC Systems Biology*, 7(Suppl 4):S11.

[192] Ratnasingham, S. and Hebert, P. D. N. (2007). BOLD : The Barcode of Life Data System (www.barcodinglife.org). *Molecular Ecology Notes*, 7(April 2016):355–364.

[193] Ren, J., Bai, X., Lu, Y. Y., Tang, K., Wang, Y., Reinert, G., and Sun, F. (2018). Alignment-Free Sequence Analysis and Applications. *Annual Review of Biomedical Data Science*, 1(1):93–114.

[194] Riley, D. R., Sieber, K. B., Robinson, K. M., White, J. R., Ganesan, A., Nourbakhsh, S., and Dunning Hotopp, J. C. (2013). Bacteria-Human Somatic Cell Lateral Gene Transfer Is Enriched in Cancer Samples. *PLOS Computational Biology*, 9(6):e1003107.

[195] Rizk, G., Lavenier, D., and Chikhi, R. (2013). DSK: k-mer counting with very low memory usage. *Bioinformatics*, 29(5):652–653.

[196] Roberts, M., Hayes, W., Hunt, B. R., Mount, S. M., and Yorke, J. A. (2004). Reducing storage requirements for biological sequence comparison. *Bioinformatics (Oxford, England)*, 20(18):3363–3369.

[197] Robinson, D. and Foulds, L. (1981). Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1-2):131–147.

[198] Rustagi, N., Zhou, A., Watkins, W. S., Gedvilaite, E., Wang, S., Ramesh, N., Muzny, D., Gibbs, R. A., Jorde, L. B., Yu, F., and Xing, J. (2017). Extremely low-coverage whole genome sequencing in South Asians captures population genomics information. *BMC genomics*, 18(1):396.

[199] Salichos, L. and Rokas, A. (2013). Inferring ancient divergences requires genes with strong phylogenetic signals. *Nature*, 497(7449):327–31.

[200] Salter, S. J., Cox, M. J., Turek, E. M., Calus, S. T., Cookson, W. O., Moffatt, M. F., Turner, P., Parkhill, J., Loman, N. J., and Walker, A. W. (2014). Reagent and laboratory contamination can critically impact sequence-based microbiome analyses. *BMC biology*, 12:87.

[201] Salzberg, S. L., Dunning Hotopp, J. C., Delcher, A. L., Pop, M., Smith, D. R., Eisen, M. B., and Nelson, W. C. (2005). Serendipitous discovery of Wolbachia genomes in multiple Drosophila species. *Genome biology*, 6(3):R23.

[202] Sanderson, M. J., Wojciechowski, M. F., Hu, J.-M., Khan, T. S., and Brady, S. G. (2000). Error, Bias, and Long-Branch Attraction in Data for Two Chloroplast Photosystem Genes in Seed Plants. *Molecular Biology and Evolution*, 17(5):782–797.

[203] Sangiovanni, M., Granata, I., Thind, A. S., and Guarracino, M. R. (2019). From trash to treasure: detecting unexpected contamination in unmapped NGS data. *BMC Bioinformatics*, 20(4):168.

[204] 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.

[205] Savolainen, V., Cowan, R. S., Vogler, A. P., Roderick, G. K., and Lane, R. (2005). Towards writing the encyclopaedia of life: an introduction to DNA barcoding. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1462):1805–1811.

[206] Sayers, E. W., Cavanaugh, M., Clark, K., Ostell, J., Pruitt, K. D., and Karsch-Mizrachi, I. (2018). GenBank. *Nucleic Acids Research*, 47(D1):D94–D99.

[207] Schmieder, R. and Edwards, R. (2011). Quality control and preprocessing of metagenomic datasets. *Bioinformatics (Oxford, England)*, 27(6):863–864.

[208] Schoch, C. L., Seifert, K. A., Huhndorf, S., Robert, V., Spouge, J. L., Levesque, C. A., Chen, W., Bolchacova, E., Voigt, K., Crous, P. W., Miller, A. N., Wingfield, M. J., Aime, M. C., An, K.-D., Bai, F.-Y., Barreto, R. W., Begerow, D., Bergeron, M.-J., Blackwell, M., Boekhout, T., Bogale, M., Boonyuen, N., Burgaz, A. R., Buyck, B., Cai, L., Cai, Q., Cardinali, G., Chaverri, P., Coppins, B. J., Crespo, A., Cubas, P., Cummings, C., Damm, U., de Beer, Z. W., de Hoog, G. S., Del-Prado, R., Dentinger, B., Diéguez-Uribeondo, J., Divakar, P. K., Douglas, B., Dueñas, M., Duong, T. A., Eberhardt, U., Edwards, J. E., Elshahed, M. S., Fliegerova, K., Furtado, M., García, M. A., Ge, Z.-W., Griffith, G. W., Griffiths, K., Groenewald, J. Z., Groenewald, M., Grube, M., Gryzenhout, M., Guo, L.-D., Hagen, F., Hambleton, S., Hamelin, R. C., Hansen, K., Harrold, P., Heller, G., Herrera, C., Hirayama, K., Hirooka, Y., Ho, H.-M., Hoffmann, K., Hofstetter, V., Högnabba, F., Hollingsworth, P. M., Hong, S.-B., Hosaka, K., Houbraken, J., Hughes, K., Huhtinen, S., Hyde, K. D., James, T., Johnson, E. M., Johnson, J. E., Johnston, P. R., Jones, E. G., Kelly, L. J., Kirk, P. M., Knapp, D. G., Kõljalg, U., Kovács, G. M., Kurtzman, C. P., Landvik, S., Leavitt, S. D., Liggenstoffer, A. S., Liimatainen, K.,

Lombard, L., Luangsa-ard, J. J., Lumbsch, H. T., Maganti, H., Maharachchikumbura, S. S. N., Martin, M. P., May, T. W., McTaggart, A. R., Methven, A. S., Meyer, W., Moncalvo, J.-M., Mongkolsamrit, S., Nagy, L. G., Nilsson, R. H., Niskanen, T., Nyilasi, I., Okada, G., Okane, I., Olariaga, I., Otte, J., Papp, T., Park, D., Petkovits, T., Pino-Bodas, R., Quaedvlieg, W., Raja, H. A., Redecker, D., Rintoul, T. L., Ruibal, C., Sarmiento-Ramírez, J. M., Schmitt, I., Schüßler, A., Shearer, C., Sotome, K., Stefani, F. O., Stenroos, S., Stielow, B., Stockinger, H., Suetrong, S., Suh, S.-O., Sung, G.-H., Suzuki, M., Tanaka, K., Tedersoo, L., Telleria, M. T., Tretter, E., Untereiner, W. A., Urbina, H., Vágvölgyi, C., Vialle, A., Vu, T. D., Walther, G., Wang, Q.-M., Wang, Y., Weir, B. S., Weiß, M., White, M. M., Xu, J., Yahr, R., Yang, Z. L., Yurkov, A., Zamora, J.-C., Zhang, N., Zhuang, W.-Y., and Schindel, D. (2012). Nuclear ribosomal internal transcribed spacer (ITS) region as a universal DNA barcode marker for Fungi. *Proceedings of the National Academy of Sciences*, 109(16):6241–6246.

[209] Schubert, M., Lindgreen, S., and Orlando, L. (2016). AdapterRemoval v2: rapid adapter trimming, identification, and read merging. *BMC Research Notes*, 9(1):88.

[210] Schwartz, S., Zhang, Z., Frazer, K. A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W. (2000). PipMaker—A Web Server for Aligning Two Genomic DNA Sequences. *Genome Research*, 10(4):577–586.

[211] Sczyrba, A., Hofmann, P., Belmann, P., Koslicki, D., Janssen, S., Dröge, J., Gregor, I., Majda, S., Fiedler, J., Dahms, E., Bremges, A., Fritz, A., Garrido-Oter, R., Jørgensen, T. S., Shapiro, N., Blood, P. D., Gurevich, A., Bai, Y., Turaev, D., DeMaere, M. Z., Chikhi, R., Nagarajan, N., Quince, C., Meyer, F., Balvočiūtė, M., Hansen, L. H., Sørensen, S. J., Chia, B. K. H., Denis, B., Froula, J. L., Wang, Z., Egan, R., Don Kang, D., Cook, J. J., Deltel, C., Beckstette, M., Lemaitre, C., Peterlongo, P., Rizk, G., Lavenier, D., Wu, Y.-W., Singer, S. W., Jain, C., Strous, M., Klingenberg, H., Meinicke, P., Barton, M. D., Lingner, T., Lin, H.-H., Liao, Y.-C., Silva, G. G. Z., Cuevas, D. A., Edwards, R. A., Saha, S., Piro, V. C., Renard, B. Y., Pop, M., Klenk, H.-P., Göker, M., Kyrpides, N. C., Woyke, T., Vorholt, J. A., Schulze-Lefert, P., Rubin, E. M., Darling, A. E., Rattei, T., and McHardy, A. C. (2017). Critical Assessment of Metagenome Interpretation—a benchmark of metagenomics software. *Nature Methods*, 14(11):1063–1071.

[212] Segata, N., Waldron, L., Ballarini, A., Narasimhan, V., Jousson, O., and Huttenhower, C. (2012). Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature Methods*, 9(8):811–814.

[213] Seifert, K. A., Samson, R. A., deWaard, J. R., Houbraken, J., Levesque, C. A., Moncalvo, J.-M., Louis-Seize, G., and Hebert, P. D. N. (2007). Prospects for fungus identification using CO1 DNA barcodes, with Penicillium as a test case. *Proceedings of the National Academy of Sciences*, 104(10):3901–3906.

[214] Shi, W., Qi, H., Sun, Q., Fan, G., Liu, S., Wang, J., Zhu, B., Liu, H., Zhao, F., Wang, X., Hu, X., Li, W., Liu, J., Tian, Y., Wu, L., and Ma, J. (2019). gcMeta: a Global Catalogue of

Metagenomics platform to support the archiving, standardization and analysis of microbiome data. *Nucleic Acids Research*, 47(D1):D637–D648.

[215] Simion, P., Belkhir, K., François, C., Veyssier, J., Rink, J. C., Manuel, M., Philippe, H., and Telford, M. J. (2018). A software tool 'CroCo' detects pervasive cross-species contamination in next generation sequencing data. *BMC Biology*.

[216] Simmons, M. P. and Gatesy, J. (2021). Collapsing dubiously resolved gene-tree branches in phylogenomic coalescent analyses. *Molecular Phylogenetics and Evolution*, 158:107092.

[217] Sims, G. E., Jun, S.-R., Wu, G. A., and Kim, S.-H. (2009). Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 106(8):2677–2682.

[218] Smit, A., Hubley, R., and Green, P. (2013). RepeatMasker Open-4.0.

[219] Smith, T. F. and Waterman, M. S. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.

[220] Solomon, B. and Kingsford, C. (2016). Fast search of thousands of short-read sequencing experiments. *Nature Biotechnology*, 34(3):300–302.

[221] Song, K., Ren, J., Reinert, G., Deng, M., Waterman, M. S., and Sun, F. (2014). New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in Bioinformatics*, 15(3):343–353.

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

[223] Steinegger, M. and Salzberg, S. L. (2020). Terminating contamination: large-scale search identifies more than 2,000,000 contaminated entries in GenBank. *bioRxiv*, page 2020.01.26.920173.

[224] Steinke, D., Vences, M., Salzburger, W., and Meyer, A. (2005). TaxI: a software tool for DNA barcoding using distance methods. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1462):1975–1980.

[225] Straub, S. C. K., Parks, M., Weitemier, K., Fishbein, M., Cronn, R. C., and Liston, A. (2012). Navigating the tip of the genomic iceberg: Next-generation sequencing for plant systematics. *American Journal of Botany*, 99(2):349–364.

[226] Sun, C., Huang, J., Wang, Y., Zhao, X., Su, L., Thomas, G. W. C., Zhao, M., Zhang, X., Jungreis, I., Kellis, M., Vicario, S., Sharakhov, I. V., Bondarenko, S. M., Hasselmann, M., Kim, C. N., Paten, B., Penso-Dolfin, L., Wang, L., Chang, Y., Gao, Q., Ma, L., Ma, L., Zhang,

Z., Zhang, H., Zhang, H., Ruzzante, L., Robertson, H. M., Zhu, Y., Liu, Y., Yang, H., Ding, L., Wang, Q., Ma, D., Xu, W., Liang, C., Itgen, M. W., Mee, L., Cao, G., Zhang, Z., Sadd, B. M., Hahn, M. W., Schaack, S., Barribeau, S. M., Williams, P. H., Waterhouse, R. M., and Mueller, R. L. (2021). Genus-Wide Characterization of Bumblebee Genomes Provides Insights into Their Evolution and Variation in Ecological and Behavioral Traits. *Molecular Biology and Evolution*, 38(2):486–501.

[227] Sunagawa, S., Coelho, L. P., Chaffron, S., Kultima, J. R., Labadie, K., Salazar, G., Djahanschiri, B., Zeller, G., Mende, D. R., Alberti, A., Cornejo-Castillo, F. M., Costea, P. I., Cruaud, C., D'Ovidio, F., Engelen, S., Ferrera, I., Gasol, J. M., Guidi, L., Hildebrand, F., Kokoszka, F., Lepoivre, C., Lima-Mendez, G., Poulain, J., Poulos, B. T., Royo-Llonch, M., Sarmento, H., Vieira-Silva, S., Dimier, C., Picheral, M., Searson, S., Kandels-Lewis, S., Bowler, C., de Vargas, C., Gorsky, G., Grimsley, N., Hingamp, P., Iudicone, D., Jaillon, O., Not, F., Ogata, H., Pesant, S., Speich, S., Stemmann, L., Sullivan, M. B., Weissenbach, J., Wincker, P., Karsenti, E., Raes, J., Acinas, S. G., and Bork, P. (2015). Structure and function of the global ocean microbiome. *Science*, 348(6237):1261359.

[228] Susko, E. (2009). Bootstrap support is not first-order correct. *Systematic Biology*, 58(2):211–223.

[229] Taberlet, P., Coissac, E., Pompanon, F., Brochmann, C., and Willerslev, E. (2012). Towards next-generation biodiversity assessment using DNA metabarcoding. *Molecular Ecology*, 21(8):2045–2050.

[230] Tagirdzhanova, G., Saary, P., Cameron, E. S., Garber, A. I., Escandón, D. D., Goyette, S., Nogerius, V. T., Passo, A., Mayrhofer, H., akon Holien, H., Tønsberg, T., Stein, L. Y., Finn, R. D., and Spribille, T. (2023). Evidence for a core set of microbial lichen symbionts from a global survey of metagenomes. *bioRxiv*.

[231] Tang, K., Ren, J., and Sun, F. (2019). Afann: bias adjustment for alignment-free sequence comparison based on sequencing data using neural network regression. *Genome Biology*, 20(1):266.

[232] Tavaré, S. (1986). Some Probabilistic and Statistical Problems in the Analysis of DNA Sequences. *Lectures on Mathematics in the Life Sciences*, 17:57–86.

[233] Taylor, D. J. (2004). An Assessment of Accuracy, Error, and Conflict with Support Values from Genome-Scale Phylogenetic Data. *Molecular Biology and Evolution*, 21(8):1534–1537.

[234] Teeling, H., Waldmann, J., Lombardot, T., Bauer, M., and Glöckner, F. O. (2004). TETRA: a web-service and a stand-alone program for the analysis and comparison of tetranucleotide usage patterns in DNA sequences. *BMC Bioinformatics*, 5(1):163.

[235] Tillich, M., Lehwark, P., Pellizzer, T., Ulbricht-Jones, E. S., Fischer, A., Bock, R., and

Greiner, S. (2017). GeSeq – versatile and accurate annotation of organelle genomes. *Nucleic Acids Research*, 45(W1):W6–W11.

[236] Townsend, J. P., Su, Z., and Tekle, Y. I. (2012). Phylogenetic Signal and Noise: Predicting the Power of a Data Set to Resolve Phylogeny. *Systematic Biology*, 61(5):835.

[237] Trevisan, B., Alcantara, D. M. C., Machado, D. J., Marques, F. P. L., and Lahr, D. J. G. (2019). Genome skimming is a low-cost and robust strategy to assemble complete mitochondrial genomes from ethanol preserved specimens in biodiversity studies. *PeerJ*, 7:e7543–e7543.

[238] VAN DER LINDE, K. I. M., HOULE, D., SPICER, G. S., and STEPPAN, S. J. (2010). A supermatrix-based molecular phylogeny of the family Drosophilidae. *Genetics Research*, 92(1):25–38.

[239] Velozo Timbó, R., Coiti Togawa, R., M. C. Costa, M., A. Andow, D., and Paula, D. P. (2017). Mitogenome sequence accuracy using different elucidation methods. *PLOS ONE*, 12(6):e0179971.

[240] Vences, M., Thomas, M., van der Meijden, A., Chiari, Y., and Vieites, D. R. (2005). Comparative performance of the 16S rRNA gene in DNA barcoding of amphibians. *Frontiers in zoology*, 2:5.

[241] Vinga, S. (2014). Editorial: Alignment-free methods in computational biology. *Briefings in Bioinformatics*, 15(3):341–342.

[242] von Meijenfeldt, F. A. B., Arkhipova, K., Cambuy, D. D., Coutinho, F. H., and Dutilh, B. E. (2019). Robust taxonomic classification of uncharted microbial sequences and bins with CAT and BAT. *Genome Biology*, 20(1):217.

[243] Warnow, T. (2017). *Computational Phylogenetics: An Introduction to Designing Methods for Phylogeny Estimation*. Cambridge University Press.

[244] Weitemier, K., Straub, S. C. K., Cronn, R. C., Fishbein, M., Schmickl, R., McDonnell, A., and Liston, A. (2014). Hyb-Seq: Combining Target Enrichment and Genome Skimming for Plant Phylogenomics. *Applications in Plant Sciences*, 2(9):1400042.

[245] Westbury, M. V., Thompson, K. F., Louis, M., Cabrera, A. A., Skovrind, M., Castruita, J. A. S., Constantine, R., Stevens, J. R., and Lorenzen, E. D. (2021). Ocean-wide genomic variation in Gray's beaked whales, Mesoplodon grayi. *Royal Society open science*, 8(3):201788.

[246] Wilson, C. G., Nowell, R. W., and Barraclough, T. G. (2018). Cross-Contamination Explains "Inter and Intraspecific Horizontal Genetic Transfers" between Asexual Bdelloid Rotifers. *Current biology : CB*, 28(15):2436–2444.e14.

[247] Wittler, R. (2020). Alignment- and reference-free phylogenomics with colored de Bruijn graphs. *Algorithms for Molecular Biology*, 15(1):4.

[248] Woloszynek, S., Zhao, Z., Chen, J., and Rosen, G. L. (2019). 16S rRNA sequence embeddings: Meaningful numeric feature representations of nucleotide sequences that are convenient for downstream analyses. *PLOS Computational Biology*, 15(2):e1006721.

[249] Wong, K. M., Suchard, M. A., and Huelsenbeck, J. P. (2008). Alignment Uncertainty and Genomic Analysis. *Science*, 319(5862):473–476.

[250] Wood, D. E., Lu, J., and Langmead, B. (2019). Improved metagenomic analysis with Kraken 2. *Genome Biology*, 20(1):257.

[251] Wood, D. E. and Salzberg, S. L. (2014). Kraken: Ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, 15(3).

[252] Wooley, J. C. and Ye, Y. (2010). Metagenomics: Facts and Artifacts, and Computational Challenges. *Journal of Computer Science and Technology*, 25(1):71–81.

[253] Wu, G. A., Jun, S.-R., Sims, G. E., and Kim, S.-H. (2009). Whole-proteome phylogeny of large dsDNA virus families by an alignment-free method. *Proceedings of the National Academy of Sciences*, 106(31):12826–12831.

[254] Ye, S. H., Siddle, K. J., Park, D. J., and Sabeti, P. C. (2019). Benchmarking Metagenomics Tools for Taxonomic Classification. *Cell*, 178(4):779–794.

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

[256] Zhang, C., Rabiee, M., Sayyari, E., and Mirarab, S. (2018). ASTRAL-III: polynomial time species tree reconstruction from partially resolved gene trees. *BMC Bioinformatics*, 19(S6):153.

[257] Zhu, Q., Mai, U., Pfeiffer, W., Janssen, S., Asnicar, F., Sanders, J. G., Belda-Ferre, P., Al-Ghalith, G. A., Kopylova, E., McDonald, D., Kosciolek, T., Yin, J. B., Huang, S., Salam, N., Jiao, J.-Y., Wu, Z., Xu, Z. Z., Cantrell, K., Yang, Y., Sayyari, E., Rabiee, M., Morton, J. T., Podell, S., Knights, D., Li, W.-J., Huttenhower, C., Segata, N., Smarr, L., Mirarab, S., and Knight, R. (2019). Phylogenomics of 10,575 genomes reveals evolutionary proximity between domains Bacteria and Archaea. *Nature Communications*, 10(1):5477.

[258] Zielezinski, A., Girgis, H. Z., Bernard, G., Leimeister, C.-A., Tang, K., Dencker, T., Lau, A. K., Röhling, S., Choi, J. J., Waterman, M. S., Comin, M., Kim, S.-H., Vinga, S., Almeida, J. S., Chan, C. X., James, B. T., Sun, F., Morgenstern, B., and Karlowski, W. M. (2019). Benchmarking of alignment-free sequence comparison methods. *Genome Biology*, 20(1):144.

[259] Zielezinski, A., Vinga, S., Almeida, J., and Karlowski, W. M. (2017). Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biology*, 18(1):186.