https://doi.org/10.1093/nargab/lqab071

CONSULT: accurate contamination removal using locality-sensitive hashing

Eleonora Rachtman[1], Vineet Bafna[2] and Siavash Mirarab[3,*]

[1]Bioinformatics and Systems Biology Graduate Program, UC San Diego, CA 92093, USA, [2]Department of Computer Science and Engineering, UC San Diego, CA 92093, USA and [3]Department of Electrical and Computer Engineering, UC San Diego, CA 92093, USA

Received March 10, 2021; Revised June 30, 2021; Editorial Decision July 19, 2021; Accepted July 19, 2021

ABSTRACT

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 skim-based mitochondrial assemblies.

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 (1). Low-cost sequencing projects remain an attractive alternative in biodiversity and ecological research (2,3). Such studies can include a large number of samples sequenced at 1-2× read coverage, often called genome skims (4–6). Traditionally, genome-skimming data were used for assembling the over-represented organelle genome using one of several approaches that have been developed (7–13). 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 biodiversity-related questions, including sample identification and population genetics (14). 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 (15–19) and using those distances for phylogenetic placement (20,21).

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 (22,23). 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 (24–26). 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, we have shown (27) 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 (28–30) missed contaminants frequently and were replaced by methods that use sequence similarity to search against libraries of

*To whom correspondence should be addressed. Tel: +1 858 822 6245; Email: smirarabbaygi@eng.ucsd.edu

potential contaminants (31). The current practice is to repurpose classification methods used in the taxonomic characterization of microbial metagenomes to identify extraneous reads in genomic datasets (31). Metagenome classifiers use a variety of approaches, including read alignment as nucleotide or protein, *k*-mer mapping, and alignment of marker genes (32–34). 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 (35–39) 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 (36) 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 genome 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 (40–43). Much of the microbial diversity on earth is not reflected with close representatives in the reference datasets (44,45). 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 (43). We recently showed (27) that even at the domain-level, the sensitivity of the leading method Kraken-II (36) 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 (46–49), including recent whole-genome databases with up to 25000 genomes (50–52). 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 (53). 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 neighboring regions in the assembly graphs (9,10). An alternative is to rely on differential coverage of organelle and nuclear genomes to distinguish the two (11,12). Finally, when a close reference genome is available, using that genome and read mapping can be used (e.g. (8)). However, given that >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 assignments. 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.

MATERIALS AND METHODS

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 (54). 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 $a \times p$ for some approximation factor $a > 1$. LSH schemes are known for many distances (55–61). 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-mers is not immediately clear. Thus, here, we focus on HD.

To design a 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:

$$\rho(d) = 1 - \left(1 - \left(1 - \frac{d}{k}\right)^h\right)^l \qquad (1)$$

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 Supplementary Figure S1, $\rho(d)$ 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 (Figure 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).

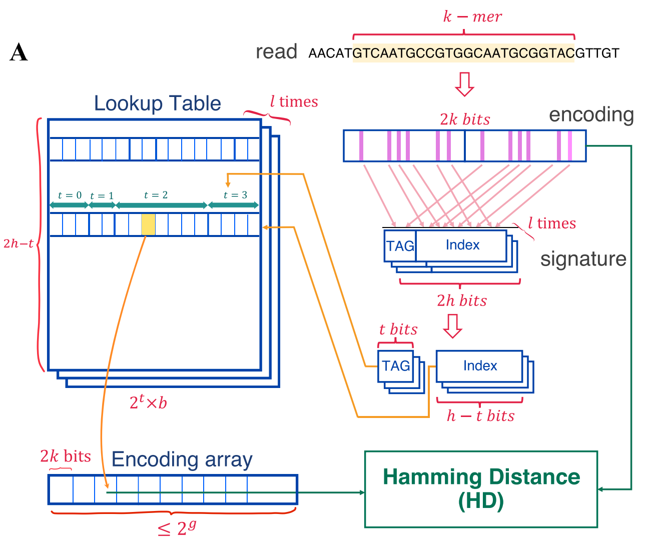
Encoding* k*-mers. Let us assume we have up to 2^g k-mers in the reference set. Every reference k-mer is encoded in a $2k$-bit number and is kept in an *encoding array* of maximum size 2^g (Figure 1A). We use a specific Left/Right encoding that allows very fast calculation of HD using a native `popcount`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

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 S1 in Supplementary Material. Notations: $\mathcal{S}$: 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($\mathcal{S}$)
    $E \leftarrow$ 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
    $\mathcal{K} \leftarrow \emptyset$
    for each $a$ in {all $m$-mers of $\mathcal{S}$} do                ▷ Minimization
        Append min{all $k$-mers of $a$} to $\mathcal{K}$
    $j \leftarrow -1$                                                  ▷ Pointer to $E$
    for each $k$-mer $a \in \mathcal{K}$ 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 $\texttt{popcount}(z_{up} \vee z_{low})$
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
```

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



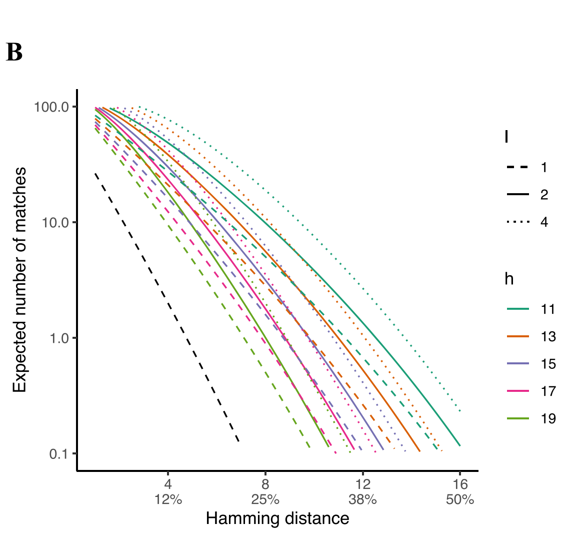


Figure 1. (**A**) A set-associative lookup table is indexed by h randomly selected LSH signatures extracted from each k-mer and points to the encoding array. (**B**) We show $(L - k + 1)\rho(d)$: the expected number of k-mers that match a k-mer in a reference genome at distance d from the read for various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

library, we save l one-to-many mappings from each k-mer signature to at most b encodings, implemented as a lookup table (Figure 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 2, Supplementary Material), which we use as index to a row of the lookup table; thus, the maximum 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 (Supplementary Figure S2) and so some rows can fill up sooner than others. To avoid losing k-mers to imbalances (Supplementary Table S1), 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 (Supplementary Figures S2 and S3). 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 (50,52), leading to 8 to 20 billion unique k-mers *after* minimization (Table 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 128 GB 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 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 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 (Supplementary Table S2). Thus, we set $l = 2$ by default.

We set $k = 32$ to reduce the chances of non-homologous k-mer matches. As Figure 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

Table 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 minimization. TOL (50) and GTDB (52) 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	159 509	20.562188135	8.173628125	6.210280798
Tree of Life (TOL) (50)	10 470	26.634996609	14.026601864	7.999975120
Genome Taxonomy Database (GTDB) (52)	31 910	22.840757178	19.376574640	7.999986111
Mitochondrial RefSeq	11 138	NA	0.201551248	0.194863421

usage, running time, and accuracy well (Supplementary Table S2). 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 bigger than 128 GB. Instead, we set $b = 7$, making our total memory usage close to 122 GB for 8 billion *k*-mers (Supplementary Table S2). 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 (62) and then minimize (63) them down to 32-mers; this step reduced the *k*-mer count to include in a reference library (Table 1). We skip the minimization step for small reference datasets with $<10^{30}$ 32-mers. Since the Jellyfish output is pseudo-randomly ordered (64), further randomization is not needed.

Experimental validation

We test CONSULT in two applications: (i) as an exclusion-filtering method that seeks to find and remove contaminants among nuclear reads and (ii) as an inclusion-filtering method that 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) (50), Genome Taxonomy Database (GTDB) R05-RS95 (52) and bacterial and archaeal species present in standard Kraken-II (35,36) (Table 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 1672 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 not be assigned a taxonomic ID using its automatic detection methods (36). We have taken care to add these genomes to the library by assigning them 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 our previous study (27), 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 accuracy of exact matches. Subsequently, all queries were divided into bins based on their distances to the closest match in a reference database (Supplementary Table S3) and 10 plant genomes (Supplementary Table S4) 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 (65). Reads for the TOL query set were simulated at 10 MB using ART (66) (see Appendix C).

(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 (43). Marine microbial species are known to be poorly represented in public repositories (67). 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 1× coverage using ART with the same settings as TOL (see Appendix C). We also included the same 10 plant species as TOL to compute the FP rates.

(iii) *Real skims.* We tested the 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 (Supplementary Table S5) from NCBI (PRJNA427774) and subsampled them down to 200 MB using seqtk (68). We removed adapters, deduplicated these samples and merged paired-end reads using BBTools (69) (see Appendix C). 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 the original study (70) were used to compute true distances.

Tools compared. We compared performance of CONSULT with Kraken-II (35,36), CLARK (37), CLARK-S (38) and Bowtie2 (71,72). These are among leading identification tools based on recent benchmarking studies (73–76). 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 previously (27) 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 C 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. The 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 C). 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 its highest sensitivity setting (see Appendix C).

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 10 Mb queries from the TOL query set (Supplementary Table S7). 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 (Supplementary Table S8). 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 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 based our experiment on data from the DNAMark project that skimmed 210 vertebrate species (NCBI project PRJNA607895) and attempted to assemble their mitochondrial genomes (77). We selected 42 (Supplementary Table S6) 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 (Supplementary Table S6).

We compare three assembly pipelines. (i) We include assemblies generated using Novoplasty (10) made available by Margaryan *et al.* (77). (ii) For each of the 42 samples, we preprocessed the raw SRA files by removing adapters using AdapterRemoval (78) and merging paired-end reads with BBTools (69) (see Appendix C). We assembled these *unfiltered* reads using plasmidSPAdes (12), which relies on read coverage to distinguish nuclear and organelle genomes. (iii) 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 (79) to obtain the assembly.

To assess the completeness of the assemblies, we first annotated all three assemblies (original, unfiltered and filtered) using MITOS (80) 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 $\sim$16 kbp in size and the number of genes should be close to 37 (81).

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 as-

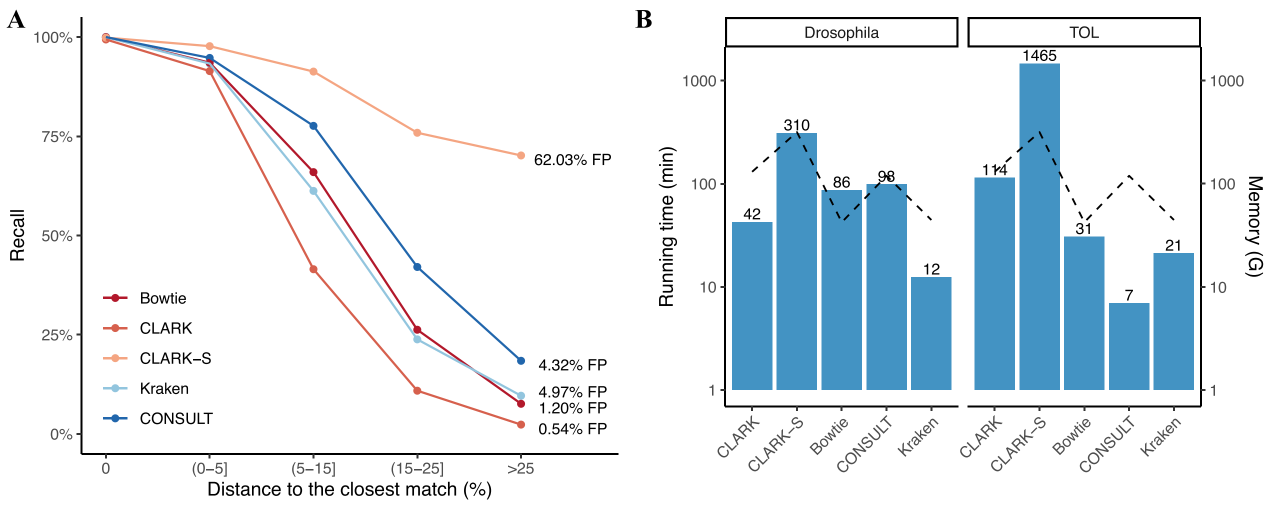


Figure 2. Results on the data simulated based on the TOL dataset. (**A**) Lines show recall for different query bins for five tools run in default settings (see Materials and Methods section). Each line is labeled with its associated FPR, computed using plant genomes added to the query set, which are identical across bins; thus, FPR is identical in all bins. (**B**) Processing speed (bars) and memory consumption (dotted line) 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 10 MB microbial queries. Computation on a machine with Intel Xeon 2.2 GHz CPU using 24 threads and 350G of RAM.

semblies, we mostly take the the longest contig (as long as it is ≥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 (82). If no PCG or rRNA genes were assigned to any contigs in assembly, the generated reference is considered as having failed annotation.

RESULTS

Exclusion filtering of contamination from nuclear reads

Controlled distances. In the controlled distance experiment, CONSULT has the best recall among the methods that are able to control the FP rate (Figure 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 (Figure 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.

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 (Figure 3). Not only does CONSULT match more reads (has higher recall), it has fewer false positives, especially for GTDB (Figure 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 (Figure 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 precision improves (Supplementary Figure S4). 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 (Supplementary Figure S5). CONSULT and Kraken-II match at least ¾ of reads for 61% and 44% of genomes, respectively. Compar-

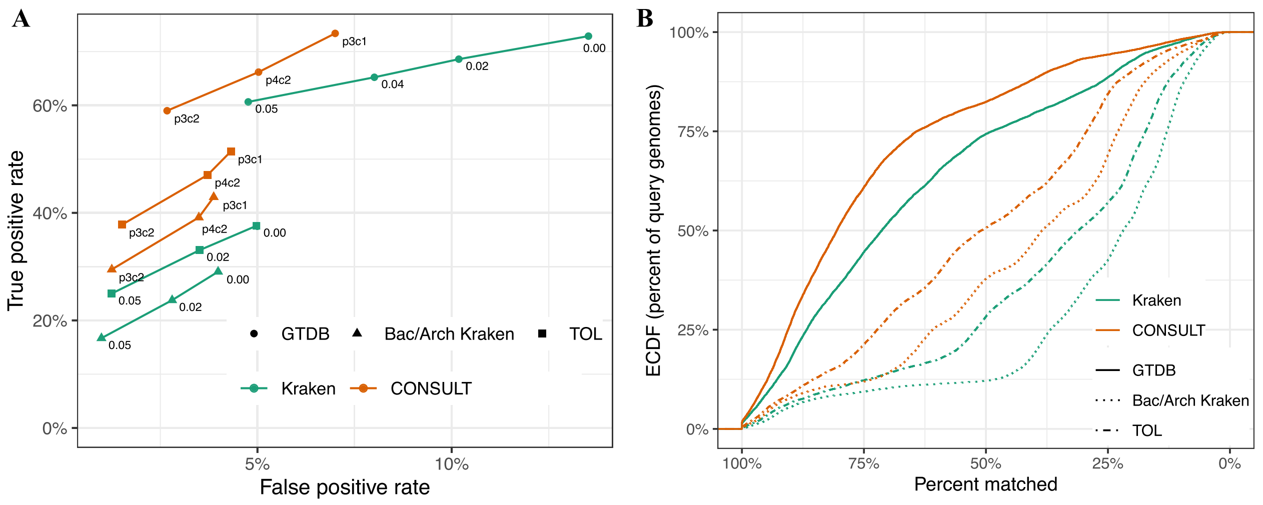


Figure 3. CONSULT versus Kraken-II on the GORG dataset. (**A**) The ROC curve showing the mean of recall versus FP rate (i.e. plant queries matched to a DB) with various settings for each method and searching against three libraries (GTDB, TOL, and the default Bac/Arch Kraken DB). Kraken-II was run with 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 Supplementary Figure S6 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.

ing 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 (Supplementary Figure S6). 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 three 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 ($c = 1$). 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 (Supplementary Figure S6).

Kraken takes around 3 min to load 166GB GTDB database (on a machine with 350 GB 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 does not exceed 3 min. Recall that CONSULT keeps the library size fixed at 128 GB 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 s.

Real skims. Testing CONSULT on real genome skims from Drosophila demonstrates that Skmer distance calculation can improve dramatically as a result of filtering (Figure 4A). Errors are reduced by as much as 44% between pairs of species (Figure 4B). While distances tend to be underestimated before filtering, they tend to be slightly overestimated after filtering (Figure 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 (Figure 4B). In the most extreme case, distances are improved by >40% when <9% of reads are removed.

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 (Figure 5).

When using raw reads we obtained complete or partial mitochondrial assemblies for 25 out of 39 assembled samples. Three samples did not 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 (Figure 5A). Similarly, they had a higher number of mitochondrial genes annotated in all but one case (Figure 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 (Figure 5

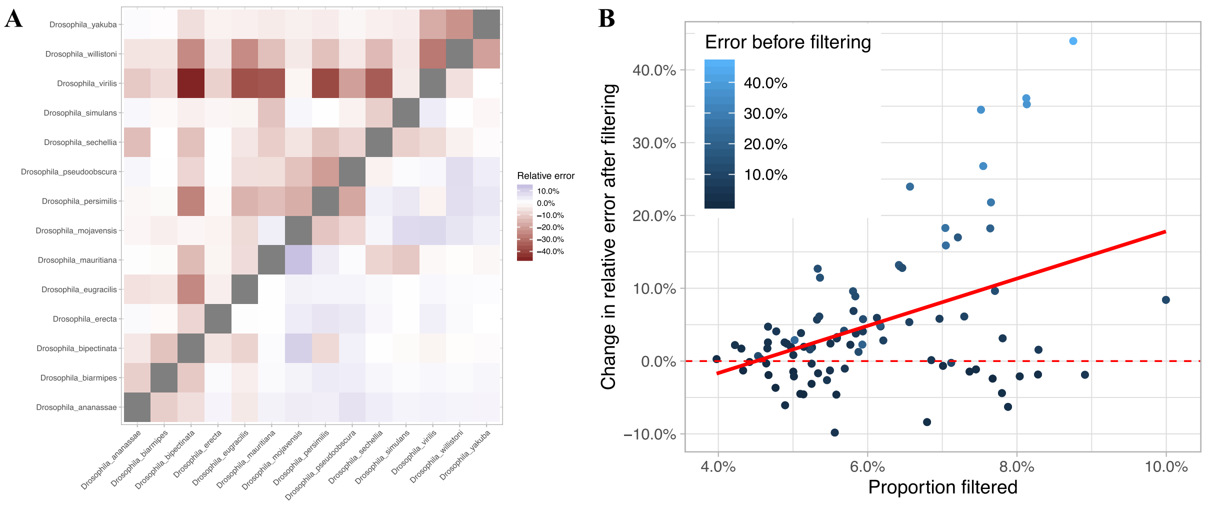


Figure 4. CONSULT on Drosophila skimming data. (**A**) Relative distance error before (upper triangle) and after (lower triangle) filtering per pair of Drosophila species. (**B**) Change in the distance error after filtering compared to the error before filtering versus amount filtered (mean of both species); positive values indicate reduction in error. Each dot represents a pair of species.

A,B). Cases where assemblies after filtering remain incomplete include both novel and observed samples. For example, of the 13 assemblies with <12 non-tRNA genes, four were represented exactly in the database, five had genomes from the same genus and four were not present at either level. Among the nine samples with no representation from the same genus, in five cases, CONSULT filtering improved gene recovery between 11 and 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 (Figure 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 (Supplementary Figure S9). 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 (Supplementary Figure S9). 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 ≈7 × running time improvement with filtered versus unfiltered reads (estimate includes CONSULT time), going from ≈ 65 min to ≈ 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).

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 Supplementary Table S10). 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-II, 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 (83–89). 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

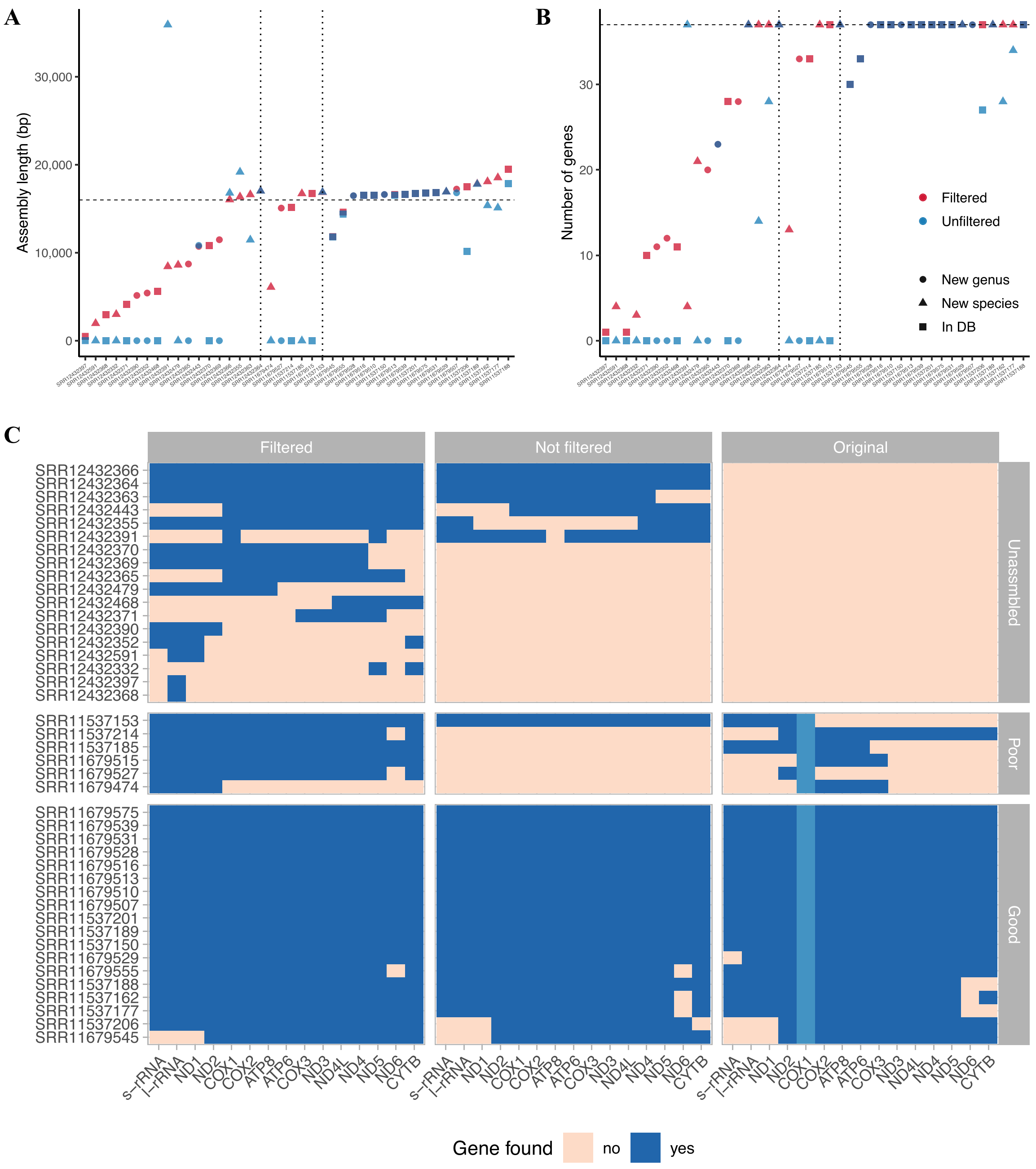


Figure 5. Mitochondrial assembly results. (**A**) Comparison of contig length between filtered and unfiltered samples. Dot shapes indicate whether any assembly from the same species or the same genus was a part of the RefSeq reference set that was used to construct CONSULT database. From left to right: 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. The COI (COX1) gene that was used as a seed sequence for assemblies generated by the DNAMark project is shown in a different shade. 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.

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 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 neighbor. The past work has used LSH for guiding assembly (87), near-duplicate sequence removal (89), phylogenetic placement (86), homology detection between two genomes (83), and sequence clustering for OTU binning (84,85) and metagenomic binning (88). 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-II 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 ≈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 (90–92). It can lead to inaccurate characterization of gene content and metabolic functions (93,94), improper inference of phylogenetic relationships (95,96), and biases in genotype calling and population genomics (97,98). Contamination is also known to infiltrate reference genomes stored in public databases (99) and is particularly problematic when endogenous DNA is scarce (100–103). 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 (104). 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 (Supplementary Table S9). 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 (105). These results suggest that filtering reads with CONSULT before assembly is as effective for chloroplast as it was for mitochondria (Supplementary Table S9). Using GetOrganelle (7) 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 (105), 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 (Supplementary Figure S7). Overall, contig length of assemblies produced from CONSULT-filtered reads was either comparable or longer (Supplementary Table S9) 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 (106) identified very few chloroplast genes in the long unfiltered assemblies, indicating that they were most likely spurious (Supplementary Figure S8).

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 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.

DATA AVAILABILITY

CONSULT is implemented in `C++11` with some x86 assembly code; it is (trivially) parallelized using OpenMP to read the library and perform the search. The software is available publicly at https://github.com/noracht/CONSULT. Scripts and summary data tables are publicly available on https://github.com/noracht/lsh_scripts. Raw data used in the manuscript is deposited in https://github.com/noracht/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.

SUPPLEMENTARY DATA

Supplementary Data are available at NARGAB Online.

ACKNOWLEDGEMENTS

Author contributions: All authors conceived the idea. S.M. and V.B. developed a theoretical model. E.R. implemented the pipeline and performed experiments. All authors contributed to the analyses of data and the writing. All authors read and approved the final manuscript.

FUNDING

National Science Foundation (NSF) [IIS-1815485 to E.R., V.B., S.M.]; NSF [ACI-1548562].
Conflict of interest statement. None declared.

REFERENCES

1. Rustagi,N., Zhou,A., Watkins,W.S., Gedvilaite,E., Wang,S., Ramesh,N., Muzny,D., Gibbs,R.A., Jorde,L.B., Yu,F. *et al.* (2017) Extremely low-coverage whole genome sequencing in South Asians captures population genomics information. *BMC Genomics*, **18**, 396.
2. 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.
3. Dodsworth,S. (2015) Genome skimming for next-generation biodiversity analysis. *Trends Plant. Sci.*, **20**, 525–527.
4. Coissac,E., Hollingsworth,P.M., Lavergne,S. and Taberlet,P. (2016) From barcodes to genomes: extending the concept of DNA barcoding. *Mol. Ecol.*, **25**, 1423–1428.
5. 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. *Am. J. Bot.*, **99**, 349–364.
6. 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. *Appl. Plant. Sci.*, **2**, 1400042.
7. 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 Biol.*, **21**, 241.
8. 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**, 3115–3117.
9. 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 Res.*, **41**, e129.
10. Dierckxsens,N., Mardulyn,P. and Smits,G. (2017) NOVOPlasty: de novo assembly of organelle genomes from whole genome data. *Nucleic Acids Res.*, **45**, e18.
11. 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**, 510.
12. 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**, 3380–3387.
13. Alqahtani,F. and MÄndoiu,I. (2020) SMART2: Multi-library Statistical Mitogenome Assembly with Repeats BT - Computational Advances in Bio and Medical Sciences. Springer International Publishing, Cham, pp. 184–198.
14. 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. *Mol. Ecol.*, **29**, 2521–2534.
15. Sarmashghi,S., Bohmann,K., Gilbert,M.T.P., Bafna,V. and Mirarab,S. (2019) Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome Biol.*, **20**, 34.
16. 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**, 638.
17. 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 Biol.*, **20**, 266.
18. 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. *et al.* (2019) Benchmarking of alignment-free sequence comparison methods. *Genome Biol.*, **20**, 144.
19. 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**, 522.
20. Balaban,M., Sarmashghi,S. and Mirarab,S. (2020) APPLES: scalable distance-based phylogenetic placement with or without alignments. *System. Biol.*, **69**, 566–578.
21. Balaban,M. and Mirarab,S. (2020) Phylogenetic double placement of mixed samples. *Bioinformatics*, **36**, i335–i343.
22. 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 Pathog.*, **12**, e1005713.
23. 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.
24. 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 Biol.*, **6**, R23.
25. Artamonova,I.I. and Mushegian,A.R. (2013) Genome sequence analysis indicates that the model eukaryote Nematostella vectensis harbors bacterial consorts. *Appl. Environ. Microb.*, **79**, 6868–6873.
26. Cornet,L., Meunier,L., Van Vlierberghe,M., Léonard,R.R., Durieu,B., Lara,Y., Misztak,A., Sirjacobs,D., Javaux,E.J., Philippe,H. *et al.* (2018) Consensus assessment of the contamination level of publicly available cyanobacterial genomes. *PLoS one*, **13**, e0200323.
27. 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. *Mol. Ecol. Resources*, **20**, 1755–0998.
28. Schmieder,R. and Edwards,R. (2011) Quality control and preprocessing of metagenomic datasets. *Bioinformatics (Oxford, England)*, **27**, 863–864.
29. 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**, 163.
30. McHardy,A.C., Martín,H.G., Tsirigos,A., Hugenholtz,P. and Rigoutsos,I. (2007) Accurate phylogenetic classification of variable-length DNA fragments. *Nat. Methods*, **4**, 63–72.
31. 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.
32. 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**, 362.
33. Bharti,R. and Grimm,D.G. (2021) Current challenges and best-practice protocols for microbiome analysis. *Brief. Bioinform.*, **22**, 178–193.

34. Wooley,J.C. and Ye,Y. (2010) Metagenomics: facts and artifacts, and computational challenges. *J. Comp. Sci. Technol.*, **25**, 71–81.
35. Wood,D.E. and Salzberg,S.L. (2014) Kraken: Ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.*, **15**, R46.
36. Wood,D.E., Lu,J. and Langmead,B. (2019) Improved metagenomic analysis with Kraken 2. *Genome Biol.*, **20**, 257.
37. 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**, 236.
38. Ounit,R. and Lonardi,S. (2016) Higher classification sensitivity of short metagenomic reads with CLARK-S. *Bioinformatics (Oxford, England)*, **32**, 3823–3825.
39. 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**, 2253–2260.
40. Liang,Q., Bible,P.W., Liu,Y., Zou,B. and Wei,L. (2020) DeepMicrobes: taxonomic classification for metagenomics with deep learning. *NAR Genom. Bioinform.*, **2**, lqaa009.
41. 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 Biol.*, **20**, 217.
42. 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 Biol.*, **19**, 165.
43. 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. *et al.* (2019) Charting the complexity of the marine microbiome through single-cell genomics. *Cell*, **179**, 1623–1635.
44. 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. *Algorithm Mol. Biol.*, **3**, 7.
45. Choi,J., Yang,F., Stepanauskas,R., Cardenas,E., Garoutte,A., Williams,R., Flater,J., Tiedje,J.M., Hofmockel,K.S., Gelder,B. *et al.* (2017) Strategies to improve reference databases for soil microbiomes. *ISME J.*, **11**, 829–834.
46. 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**, 5069–5072.
47. Maidak,B.L. (2001) The RDP-II (Ribosomal Database Project). *Nucleic Acids Res.*, **29**, 173–174.
48. 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 Res.*, **44**, D590–D596.
49. Shi,W., Qi,H., Sun,Q., Fan,G., Liu,S., Wang,J., Zhu,B., Liu,H., Zhao,F., Wang,X. *et al.* (2019) gcMeta: a Global Catalogue of Metagenomics platform to support the archiving, standardization and analysis of microbiome data. *Nucleic Acids Res.*, **47**, D637–D648.
50. Zhu,Q., Mai,U., Pfeiffer,W., Janssen,S., Asnicar,F., Sanders,J.G., Belda-Ferre,P., Al-Ghalith,G.A., Kopylova,E., McDonald,D. *et al.* (2019) Phylogenomics of 10,575 genomes reveals evolutionary proximity between domains Bacteria and Archaea. *Nat. Commun.*, **10**, 5477.
51. Asnicar,F., Thomas,A.M., Beghini,F., Mengoni,C., Manara,S., Manghi,P., Zhu,Q., Bolzan,M., Cumbo,F., May,U. *et al.* (2020) Precise phylogenetic analysis of microbial isolates and genomes from metagenomes using PhyloPhlAn 3.0. *Nat. Commun.*, **11**, 2500.
52. 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. *Nat. Biotechnol.*, **38**, 1079–1086.
53. Locey,K.J. and Lennon,J.T. (2016) Scaling laws predict global microbial diversity. *Proc. Natl. Acad. Sci.*, **113**, 5970–5975.
54. Har-Peled,S., Indyk,P. and Motwani,R. (2012) Approximate nearest neighbors: Towards removing the curse of dimensionality. *Theor. Comput.*, **8**, 321–350.
55. Broder,A. (1997) On the resemblance and containment of documents. In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*. IEEE Comput. Soc. Salerno, Italy, pp. 21–29.
56. Narayanan,M. and Karp,R.M. (2004) Gapped Local Similarity Search with Provable Guarantees. In: *WABI 2004: Algorithms in Bioinformatics*. Vol. **3240**, Springer, Berlin, Heidelberg, pp. 74–86.
57. 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 New York*. ACM Press, NY, p. 253.
58. Gorisse,D., Cord,M. and Precioso,F. (2012) Locality-sensitive hashing for Chi2 distance. *IEEE Trans. Pattern Anal. Machine Intelligence*, **34**, 402–409.
59. 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*. Society for Industrial and Applied Mathematics, Philadelphia, PA, pp. 1018–1028.
60. Kulis,B. and Grauman,K. (2012) Kernelized locality-sensitive hashing. *IEEE Trans. Pattern Anal. Machine Intelligence*, **34**, 1092–1104.
61. Marçais,G., DeBlasio,D., Pandey,P. and Kingsford,C. (2019) Locality-sensitive hashing for the edit distance. *Bioinformatics*, **35**, i127–i135.
62. Marçais,G. and Kingsford,C. (2011) A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, **27**, 764–770.
63. 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**, 3363–3369.
64. Marçais,G. (2013) Jellyfish 2 User Guide.
65. Ondov,B.D., Treangen,T.J., Melsted,P., Mallonee,A.B., Bergman,N.H., Koren,S. and Phillippy,A.M. (2016) Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol.*, **17**, 132.
66. Huang,W., Li,L., Myers,J.R. and Marth,G.T. (2012) ART: A next-generation sequencing read simulator. *Bioinformatics*, **28**, 593–594.
67. Sunagawa,S., Coelho,L.P., Chaffron,S., Kultima,J.R., Labadie,K., Salazar,G., Djahanschiri,B., Zeller,G., Mende,D.R., Alberti,A. *et al.* (2015) Structure and function of the global ocean microbiome. *Science*, **348**, 1261359.
68. Li,H. (2018) Seqtk, toolkit for processing sequences in FASTA/Q formats. https://github.com/lh3/seqtk, (8 December 2020, date last accessed).
69. Bushnell,B., Rood,J. and Singer,E. (2017) BBMerge – Accurate paired shotgun read merging via overlap. *PLOS ONE*, **12**, e0185056.
70. 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**, 3131–3141.
71. Langmead,B. and Salzberg,S.L. (2012) Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, **9**, 357–359.
72. Langmead,B., Wilks,C., Antonescu,V. and Charles,R. (2019) Scaling read aligners to hundreds of threads on general-purpose processors. *Bioinformatics*, **35**, 421–432.
73. Ye,S.H., Siddle,K.J., Park,D.J. and Sabeti,P.C. (2019) Benchmarking metagenomics tools for taxonomic classification. *Cell*, **178**, 779–794.
74. Meyer,F., Bremges,A., Belmann,P., Janssen,S., McHardy,A.C. and Koslicki,D. (2019) Assessing taxonomic metagenome profilers with OPAL. *Genome Biol.*, **20**, 51.
75. Sczyrba,A., Hofmann,P., Belmann,P., Koslicki,D., Janssen,S., Dröge,J., Gregor,I., Majda,S., Fiedler,J., Dahms,E. *et al.* (2017) Critical Assessment of Metagenome Interpretation – a benchmark of metagenomics software. *Nat. Methods*, **14**, 1063–1071.
76. McIntyre,A. B.R., Ounit,R., Afshinnekoo,E., Prill,R.J., Hénaff,E., Alexander,N., Minot,S.S., Danko,D., Foox,J., Ahsanuddin,S. and et,al.T (2017) Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. *Genome Biol.*, **18**, 182.
77. 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. *et al.* (2020) Mitochondrial genomes of Danish vertebrate species generated for

the national DNA reference database, DNAmark. *Environment. DNA*, **3**, 472–480.
78. Schubert,M., Lindgreen,S. and Orlando,L. (2016) AdapterRemoval v2: rapid adapter trimming, identification, and read merging. *BMC Res. Notes*, **9**, 88.
79. 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. *et al.* (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.*, **19**, 455–477.
80. 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. *Mol. Phylogenet. Evol.*, **69**, 313–319.
81. Boore,J.L. (1999) Animal mitochondrial genomes. *Nucleic Acids Res.*, **27**, 1767–1780.
82. Meng,G., Li,Y., Yang,C. and Liu,S. (2019) MitoZ: a toolkit for animal mitochondrial genome assembly, annotation and visualization. *Nucleic Acids Res.*, **47**, e63.
83. Buhler,J. (2001) Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, **17**, 419–428.
84. 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*. IEEE, Philadelphia, PA, pp. 1–6.
85. Rasheed,Z., Rangwala,H. and Barbará,D. (2013) 16S rRNA metagenome clustering and diversity estimation using locality sensitive hashing. *BMC Syst. Biol.*, **7**, S11.
86. Brown,D. and Truszkowski,J. (2013) LSHPlace: Fast phylogenetic placement using locality-sensitive hashing. In: *Pacific Symposium On Biocomputing*. pp. 310–319.
87. 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. *Nat. Biotechnol.*, **33**, 623–630.
88. Luo,Y., Yu,Y.W., Zeng,J., Berger,B. and Peng,J. (2019) Metagenomic binning through low-density hashing. *Bioinformatics*, **35**, 219–226.
89. Metsky,H.C., Siddle,K.J., Gladden-Young,A., Qu,J., Yang,D.K., Brehio,P., Goldfarb,A., Piantadosi,A., Wohl,S., Carter,A. *et al.* (2019) Capturing sequence diversity in metagenomes with comprehensive and scalable probe design. *Nat. Biotechnol.*, **37**, 160–168.
90. 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**, 721–730.
91. Steinegger,M. and Salzberg,S.L. (2020) Terminating contamination: large-scale search identifies more than 2,000,000 contaminated entries in GenBank. *Genome Biol.*, **21**, 115.
92. Lu,J. and Salzberg,S.L. (2018) Removing contaminants from databases of draft genomes. *PLoS Comput. Biol.*, **14**, e1006277.
93. 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. *Proc. Natl. Acad. Sci. USA*, **113**, 5053–5058.
94. 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 Res.*, **29**, 954–960.
95. Laurin-Lemay,S., Brinkmann,H. and Philippe,H. (2012) Origin of land plants revisited in the light of sequence contamination and missing data. *Curr. Biol.*, **22**, R593–R594.
96. 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 Biol.*, **16**, 28.
97. Ballenghien,M., Faivre,N. and Galtier,N. (2017) Patterns of cross-contamination in a multispecies population genomic project: detection, quantification, impact, and solutions. *BMC Biol.*, **15**, 25.
98. Wilson,C.G., Nowell,R.W. and Barraclough,T.G. (2018) Cross-Contamination Explains 'Inter and Intraspecific Horizontal Genetic Transfers' between Asexual Bdelloid Rotifers. *Curr. Biol. : CB*, **28**, 2436–2444.
99. Merchant,S., Wood,D.E. and Salzberg,S.L. (2014) Unexpected cross-species contamination in genome sequencing projects. *PeerJ*, **2**, e675.
100. 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 Pathog.*, **8**, 24.
101. 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 Comput. Biol.*, **9**, e1003107.
102. 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 Biol.*, **12**, 87.
103. Lusk,R.W. (2014) Diverse and widespread contamination evident in the unmapped depths of high throughput sequencing data. *PLOS ONE*, **9**, e110808.
104. 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**, e0179971.
105. 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 Biol.*, **21**, 254.
106. 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 Res.*, **45**, W6–W11.