Bioinformatics, 2023, **39(9)**, btad512
https://doi.org/10.1093/bioinformatics/btad512
Advance Access Publication Date: 21 August 2023
Original Paper

Sequence analysis

Minmers are a generalization of minimizers that enable unbiased local Jaccard estimation

Bryce Kille [1,*], Erik Garrison [2], Todd J. Treangen [1], Adam M. Phillippy[3,*]

[1]Department of Computer Science, Rice University, Houston, TX, United States
[2]Department of Genetics, Genomics and Informatics, University of Tennessee Health Science Center, Memphis, TN, United States
[3]Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, MD, United States

*Corresponding authors. Department of Computer Science, Rice University, 6100 Main Street, Duncan Hall 3122, Houston, TX 77005-1892, United States. E-mail: blk6@rice.edu (B.K.); Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Building 49, Room 4A22 49, Convent Drive, Bethesda, MD 20892, United States. E-mail: adam.phillippy@nih.gov (A.M.P.)

Associate Editor: Peter Robinson

Abstract

Motivation: The Jaccard similarity on *k*-mer sets has shown to be a convenient proxy for sequence identity. By avoiding expensive base-level alignments and comparing reduced sequence representations, tools such as MashMap can scale to massive numbers of pairwise comparisons while still providing useful similarity estimates. However, due to their reliance on minimizer winnowing, previous versions of MashMap were shown to be biased and inconsistent estimators of Jaccard similarity. This directly impacts downstream tools that rely on the accuracy of these estimates.

Results: To address this, we propose the *minmer* winnowing scheme, which generalizes the minimizer scheme by use of a rolling minhash with multiple sampled *k*-mers per window. We show both theoretically and empirically that minmers yield an unbiased estimator of local Jaccard similarity, and we implement this scheme in an updated version of MashMap. The minmer-based implementation is over 10 times faster than the minimizer-based version under the default ANI threshold, making it well-suited for large-scale comparative genomics applications.

Availability and implementation: MashMap3 is available at https://github.com/marbl/MashMap.

1 Introduction

The recent deluge of genomic data accelerated by population-scale long-read sequencing efforts has driven an urgent need for scalable long-read mapping and comparative genomics algorithms. The completion of the first Telomere-to-Telemore (T2T) human genome (Nurk *et al.* 2022) and the launch of the Human Pangenome Project (Wang *et al.* 2022a) have paved the way to mapping genomic diversity at unprecedented scale and resolution. A key goal when comparing a newly sequenced human genome to a reference genome or pangenome is to accurately identify homologous sequences, i.e. DNA sequences that share a common evolutionary source.

Algorithms for pairwise sequence alignment, which aim to accurately identify homologous regions between two sequences, have continued to advance in recent years (Marco-Sola *et al.* 2021). While a powerful and ubiquitous computational tool in computational biology, exact alignment algorithms are typically reserved for situations where the boundaries of homology are known *a priori*, due to their quadratic runtime costs and inability to model non-linear sequence relationships, such as inversions, translocations, and copy number variants. Because of this, long-read mapping or whole-genome alignment methods must first identify homologous regions across billions of nucleotides, after which the exact methods can be deployed to compute a base-level "gapped" read alignment for each region. To efficiently identify candidate mappings, the prevailing strategy is to first sample *k*-mers and then identify consecutive *k*-mers that appear in the same order for both sequences: known as "seeding" and "chaining," respectively.

For many use cases, an exact gapped alignment is not needed and only an estimate of sequence identity is required. As a result, methods have been developed which can predict sequence identity without the cost of computing a gapped alignment. Jaccard similarity, a metric used for comparing the similarity of two sets, has found widespread use for this task, especially when combined with locality sensitive hashing of *k*-mer sets (Brown and Irber 2016, Ondov *et al.* 2016, Jain *et al.* 2017, 2018a, Baker and Langmead 2019, Ondov *et al.* 2019, Shaw and Yu 2023). By comparing only *k*-mers, the Jaccard can be used to estimate the average nucleotide identity (ANI) of two sequences without the need for an exact alignment (Ondov *et al.* 2016, 2019, Blanca *et al.* 2022).

To accelerate mapping and alignment, *k*-mers from the input sequences are often down-sampled using a "winnowing scheme" in a way that reduces the input size while still enabling meaningful comparisons. For example, both MashMap (Jain *et al.* 2017, 2018a) and Minimap (Li 2018) use a minimizer scheme (Roberts *et al.* 2004), which selects only the "smallest" *k*-mer from all *w*-length substrings of the

Received: 18 April 2023; **Revised:** 19 July 2023; **Editorial Decision:** 16 August 2023; **Accepted:** 18 August 2023

genome. Of relevance to this study, MashMap2 then uses these minimizers to approximate the Jaccard similarity between the mapped sequences, and these estimates have been successfully used by downstream methods, such as FastANI (Jain *et al.* 2018b) and MetaMaps (Dilthey *et al.* 2019).

However, a recent investigation noted limitations of the "winnowed minhash" scheme introduced by MashMap (Belbasi *et al.* 2022). Although the original MashMap paper notes a small, but negligible bias in its estimates (Jain *et al.* 2017), Belbasi *et al.* (2022) proved that no matter the length of the sequences, the bias of the minimizer-based winnowed minhash estimator is never zero.

To address this limitation, we propose a novel winnowing scheme, the "minmer" scheme, which is a generalization of minimizers that allows for the selection of multiple k-mers per window. We define this scheme, characterize its properties, and provide an implementation in MashMap3. Importantly, we show that minmers, unlike minimizers, enable an unbiased prediction of the local Jaccard similarity.

2 Preliminaries

Let Σ be an alphabet and $\mathcal{S}_k(S)$ be a function, which returns the set of all k-mers in S. Similarly, given a sequence S, we define $W_i^{(w)}(S)$ as the sequence of w k-mers in S starting at the ith k-mer. When w and S are clear from context, we use W_i. We use the terms sequence and string interchangeably.

2.1 Jaccard similarity and the minhash approximation

Given two sets A and B, their Jaccard similarity is defined as $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$. The Jaccard similarity between two sequences R and Q can be computed as $J(\mathcal{S}_k(R), \mathcal{S}_k(Q))$ for some k-mer size k.

However, computing the exact Jaccard for $\mathcal{S}_k(R)$ and $\mathcal{S}_k(Q)$ is not an efficient method for determining similarity for long reads and whole genomes. Instead, the minhash algorithm provides an estimator for the Jaccard similarity while only needing to compare a fraction of the two sets. Assuming U is the universe of all possible elements and $\pi : U \rightarrow [|U|]$ is a function which imposes a randomized total order on the universe of elements, we have that

$$J(A,B) = \Pr(\min_{x \in A}(\pi(x)) = \min_{x \in B}(\pi(x)).$$

This equivalency, proven by Broder (1997), is key to the minhash algorithm and yields an unbiased and consistent Jaccard estimator $\hat{J}$ with the help of a sketching function π_s. Let π_s return the lowest s items from the input set according to the random total order π. Then, we define the minhash as

$$\hat{J}(A,B) = \frac{|\pi_s(A \cup B) \cap \pi_s(A) \cap \pi_s(B)|}{|\pi_s(A \cup B)|}.$$

Importantly, this Jaccard estimator has a standard deviation that scales with $\mathcal{O}(1/\sqrt{s})$ and is therefore independent of the size of the original input sets. While there are a number of variants of minhash, which provide the same guarantee (Cohen 2016), we will be using the "bottom-s sketch" (as opposed to the s-mins and s-partition sketch) since it ensures a consistent sketch size regardless of the parameters and requires only a single hash computation per element of $\mathcal{S}_k$. Additionally, the simplicity of the bottom-s sketch leads to a streamlined application of the sliding window model, which we describe next.

2.2 Winnowing

While sequences can be reduced into their corresponding sketch via the method described above, this is a "global" sketch and it is difficult to determine where two sequences share similarity. In order to perform local sketching, Schleimer *et al.* (2003) and Roberts *et al.* (2004) independently introduced the concept of "winnowing" and "minimizers." In short, given some total ordering on the k-mers, a window of length w is slid over the sequence and the element with the lowest rank in each window (the "minimizer") is selected, using the left-most position to break ties (Roberts *et al.* 2004). By definition, winnowing ensures that at least one element is sampled per window and therefore there is never a gap of more than w elements between sampled positions. Here, we extend the winnowing concept to allow the selection of more than one element per window (the "minmers"), and we refer to the set of all minmers and/or their positions as the "winnowed" sequence.

2.2.1 Winnowing scheme characteristics

Definition 2.1. *A winnowing scheme has a* (w, s)*-window guarantee if for every window of w k-mers, there are at least* $\min(\#_{distinct}, s)$ *k-mers sampled from the window, where* $\#_{distinct}$ *is the number of distinct k-mers in the window.*

This definition is more general than the commonly used w-window guarantee, which is equivalent to the $(w, 1)$-window guarantee. While not all winnowing schemes must have such a guarantee, this ensures that no area of the sequence is under-sampled.

Recently, Shaw and Yu (2022) provided an analytical framework for winnowing schemes and showed that mapping sensitivity is related to the distribution of distances (or "spread") between sampled positions, and precision is related to the proportion of unique values relative to the total number of sampled positions. As the overarching goal of winnowing is to reduce the size of the input while preserving as much information as possible, winnowing schemes typically aim to optimize the precision/sensitivity metrics given a particular density.

Definition 2.2. *The density d of a winnowing scheme is defined as the expected frequency of sampled positions from a long random string, and the density factor* d_f *is defined as the expected number of sampled positions in a window of* $w + 1$ *k-mers.*

There has been significant work on improving the performance of minimizers by identifying orderings that reduce the density factor (Marçais *et al.* 2017). Minimizer schemes, which use a uniformly random ordering, have a density factor of $d_f = 2$ and recent schemes like Miniception (Zheng *et al.* 2020) and PASHA (Ekim *et al.* 2020) are able to obtain density factors as low as 1.7 for certain values of w and k.

For the remainder of this work, we will assume that $w \ll 4^k$, i.e. the windows are not so large that we expect duplicate k-mers in a random string. This ensures that each

k-mer in a window has probability s/w of being in the sketch for that window.

2.2.2 Winnowing scheme hierarchies

Recent winnowing methods have focused on schemes that select at most a single position per window, which simplifies analyses but restricts the universe of possible schemes. Minimizers belong to the class of "forward" winnowing schemes, where the sequence of positions sampled from adjacent sliding windows is non-decreasing (Marçais *et al.* 2018). More general is the concept of a w-local scheme (Shaw and Yu 2022), defined on windows of w consecutive k-mers but without the forward requirement. Non-forward schemes are more powerful and are not limited by the same density factor bounds as forward schemes. While the need of non-forward schemes to "jump back" in order to obtain lower sampling densities is acknowledged by Marçais *et al.* (2018), there are currently no well-studied, non-forward, w-local schemes.

2.3 MashMap

MashMap is a minimizer-based tool for long-read and whole-genome sequence homology mapping that is designed to identify all pairwise regions above some sequence similarity cutoff (Jain *et al.* 2017, 2018a). Specifically, for a reference sequence R and a query sequence Q comprised of w k-mers, MashMap aims to find all positions i in the reference such that $J(A, B_i) \geq c$, where $A = \mathcal{S}_k(Q)$ and $B_i = W_i^{(w)}(R)$, and c is the sequence similarity cutoff. For ease of notation, we will use B to refer to the sequence of k-mers from the reference sequence R.

Importantly, MashMap only requires users to specify a minimum segment length and minimum sequence identity threshold, and the algorithm will automatically determine the parameters needed to return all mappings that meet these criteria with parameterized confidence under a binomial mutation model.

To simplify the computation of the minhash, prior versions of MashMap first winnowed the query and reference sequences using the minimizer scheme after which the Jaccard was estimated from the bottom-s sketches of the minimizers. It is this use of minimizers, though, which was recently shown to introduce bias into the Jaccard estimation (Belbasi *et al.* 2022).

Here, we replace the minimizer-based approach of prior versions of MashMap with minmers. While the problem formulation remains the same, our method for computing the reference index and filtering candidate mappings is novel. We will first introduce the concept of minmers, which enable winnowing the input sequences while still maintaining the k-mers necessary to compute an unbiased Jaccard estimation between any two windows of length at least w. We will then discuss the construction of the reference index and show how query sequences can be efficiently mapped to the reference such that their expected ANI is above the desired threshold.

3 The minmer winnowing scheme

Minmers are a generalization of minimizers that allow for the selection of more than one minimum value per window. The relationship between minmers and minimizers was noted by Berlin *et al.* (2015) but as a global sketch and without the use of a sliding window. Here, we formalize a definition of the minmer winnowing scheme.

Definition 3.1. *Given a tuple (w, s, k, π), where w, k, and s are integers and π is an ordering on the set of all k-mers, a k-mer in a sequence is a minmer if it is one of the smallest s k-mers in any of the subsuming windows of w k-mers.*

Similar to other w-local winnowing schemes, ties between k-mers are broken by giving priority to the left-most k-mer. From the definition, it follows that by letting $s = 1$, we obtain the definition of the minimizer scheme. Compared to minimizers with the same w value, minmers guarantee that at least s k-mers will be sampled from each window. However, as a non-forward scheme, a minmer may be one of the smallest s k-mers in two non-adjacent windows, yet not one of the smallest s k-mers in an intervening window (Fig. 1). To account for this and simplify development of this scheme, we define a "minmer interval" to be the interval for which the k-mer at position i is a minmer for all windows starting within that interval. Thus, a single k-mer may have multiple minmer intervals starting at different positions.

Definition 3.2. *A tuple (i, a, b) is a minmer interval for a sequence S if the k-mer at position i is a minmer for all windows W_j where $j \in [a, b)$, but not W_{a-1} or W_b.*

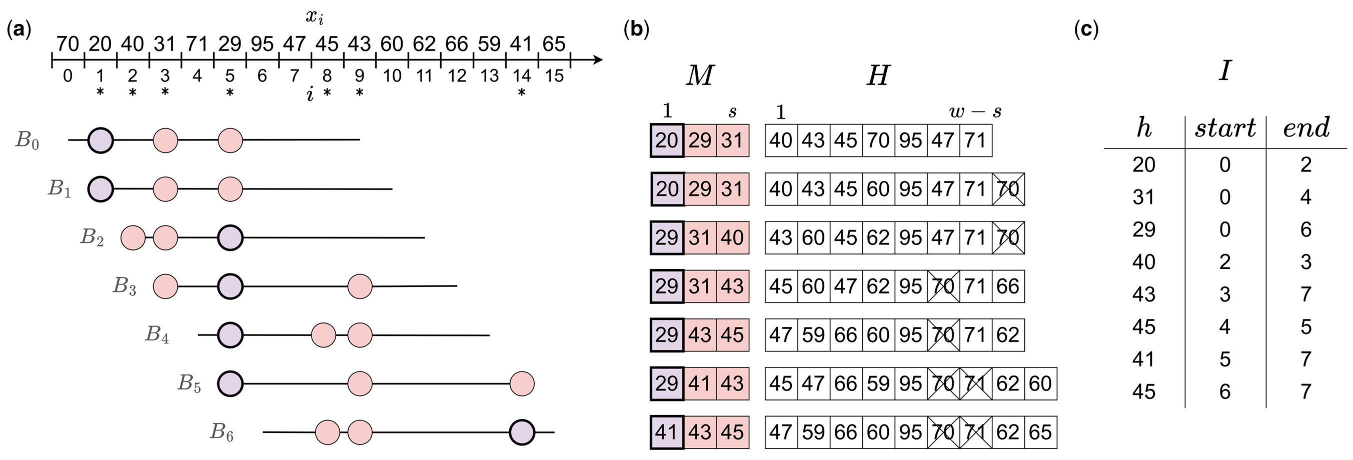
Any region of w k-mers may contain more than s minmers (e.g. B_0 and B_1 contain six minmers while $s = 3$ in Fig. 1a), and so to naively compute the minhash between a query and W_j would require identification of the s smallest minmers in W_j. Minmer intervals are convenient because for any window start position j, the s smallest k-mers in W_j are simply the ones whose minmer intervals contain j. Thus, indexing S with minmer intervals enables the efficient retrieval of the smallest s k-mers for any window without additional sorting or comparisons.

Another benefit of minmer intervals is that the smallest s k-mers for any window of length $w' > w$ are guaranteed to be a subset of the combined (w, s)-minmers contained in that window. This subset can be easily computed with minmer intervals, since the set of (w, s)-minmer intervals that overlap with the range $[i, i + w' - w]$ are also guaranteed to include the s smallest k-mers of the larger window, and the overlapping minmer intervals can be inspected to quickly identify them.

3.1 Constructing the rolling minhash index

In this section, we will describe our rolling bottom-s sketch algorithm for collecting minmers and their corresponding minmer intervals. Popic and Batzoglou (2017) proposed a related rolling minhash method for short-read mapping, but using an s-mins scheme without minmer intervals. For the remainder of the section, we will assume no duplicate k-mers in a window and an ideal uniform hash function, which maps to [0, 1]. Duplicate k-mers are handled in practice by keeping a counter of the number of active positions for a particular k-mer, similar to the original MashMap implementation (Jain *et al.* 2017). Minmer intervals longer than the window length sometimes arise due to duplicate k-mers and are split into adjacent intervals of length at most w. This bound on the minmer interval length is necessary for the mapping step.

For ease of notation, we now consider B as a sequence of k-mer hash values $x_0, x_1, \ldots, x_n$ where each $x_i \in [0, 1]$ and refer to these elements as hashes and k-mers interchangeably. We use a min-heap H and a sorted map M, both ordered on the



h	*start*	*end*
20	0	2
31	0	4
29	0	6
40	2	3
43	3	7
45	4	5
41	5	7
45	6	7

Figure 1. Constructing the rolling minhash index. (a) A sliding window B_i of length $w = 10$ is moved over the hashes of all k-mers. At each position i of the sliding window, the positions with the $s = 3$ lowest hash values are marked as minmers. The three minmers for each window are highlighted with colored circles, with the smallest hash in each window (the minimizer) having a bolded outline. Sampled minmers are also identified by an asterisk below their position. (b) The values of the hashes in the map M and heap H as the window slides over the sequence. The expired k-mers in the heap are crossed out. (c) The final sorted minmer interval index I.

hash values, to keep track of the rolling minhash index. As the window slides across B, M will contain the minmer intervals for the lowest s hashes in the window and H will contain the remaining hashes in the window. We denote the minmer interval of a hash x in M by $M[x]^{(start)}$ and $M[x]^{(end)}$. In practice, H may contain "expired" k-mers, which are no longer part of the current window, however by storing the k-mer position as well, we can immediately discard such k-mers whenever they appear at the top of the heap. To prevent expired k-mers from accumulating, all expired k-mers from the heap are pruned whenever the heap size exceeds $2w$.

After initialization of H and M with the first w k-mers of B, we begin sliding the window for each consecutive position i and collect the minmer intervals in an index I. For each window B_i, there will be a single "exiting" k-mer x_{i-1} and a single "entering" k-mer x_{i+w-1}, each of which may or may not belong to the lowest s k-mers. Therefore, we have four possibilities, examples of which can be seen in Fig. 1.

1) $x_{i-1} > \max(M)$ and $x_{i+w-1} > \max(M)$
Neither the exiting nor the entering k-mer is in the sketch. Insert x_{i+w-1} into H.
2) $x_{i-1} > \max(M)$ and $x_{i+w-1} \leq \max(M)$.
The exiting k-mer was not in the sketch, but the entering k-mer will be. Since the incoming k-mer x_{i+w-1} enters the sketch, the largest element in the sketch must be removed. Therefore, $M[\max(M)]^{(end)}$ is set to i and the minmer interval is appended to the index I. $\max(M)$ is then removed from M and the new k-mer x_{i+w-1} is inserted to M, marking $M[x_{i+w-1}]^{(start)} = i$.
3) $x_{i-1} \leq \max(M)$ and $x_{i+w-1} > \max(M)$
The exiting k-mer was in the sketch, but the entering k-mer will not be. Since the exiting k-mer x_{i-1} was a member of the sketch, set $M[x_{i-1}]^{(end)} = i$, remove $M[x_{i-1}]$ from M and append it to I, and insert x_{i+w-1} into H. At this point, $|M| = s-1$, as we removed an element from the sketch but did not replace it. To fill the empty sketch position, k-mers are popped from H until a k-mer x, which has not expired is obtained. This k-mer is added to M, setting $M[x]^{(start)} = i$.
4) $x_{i-1} \leq \max(M)$ and $x_{i+w-1} \leq \max(M)$.
Both the exiting and entering k-mers are in the sketch. As before, set $M[x_{i-1}]^{(end)} = i$ and remove $M[x_{i-1}]$ from M and append it to I. The entering k-mer belongs in the sketch, so set $M[x_{i+w-1}]^{(start)} = i$.

Our implementation of M uses a balanced binary tree and H is pruned in $\mathcal{O}(w)$ time by constructing a new heap from the w relevant k-mers. As the pruning cannot occur more than once every w k-mers, the amortized time complexity of the pruning step is $\mathcal{O}(1)$ and therefore each sliding window update is $\mathcal{O}(\log(w))$. In order to efficiently use the index for mapping, we sort I based on the start positions of the minmer intervals and in addition, we compute a reverse lookup table T, which maps hash values to their corresponding ordered lists of minmer intervals.

The expected size of the index is $nd^*_{(w,s)}$, where $d^*_{(w,s)}$ is the minmer interval density (defined in Section 4.1.2) and is $\mathcal{O}(\frac{s}{w})$. Therefore, the initial winnowing complexity is $\mathcal{O}(n\log(w))$ and the time complexity for sorting the intervals is $\mathcal{O}\left(\frac{ns}{w}\log\left(\frac{ns}{w}\right)\right)$ in expectation. As the index consists solely of minmer intervals, the space complexity is $\mathcal{O}\left(\frac{ns}{w}\right)$ in expectation.

3.2 Querying the rolling minhash index

MashMap computes mappings in a two-stage process. In the first stage, all regions within the reference that may contain a mapping satisfying the desired ANI constraints are obtained. In the second stage, the minhash algorithm is used to estimate the Jaccard for each candidate mapping position i produced by the first stage. As the second stage is the most computationally intensive step, we introduce both a new candidate region filter and a more efficient minhash computation to improve overall runtime. We assume here that query sequences are w k-mers long. In practice, sequences longer than w are split into windows of w k-mers, mapped independently, and then chained and filtered as described in Jain *et al.* (2018a).

3.2.1 Stage 1: candidate region filter

First, the query sequence A is sketched using a min-heap to obtain the s lowest hash values. All m minmer intervals in the reference with matching hashes are obtained from T and a sorted list L is created in $\mathcal{O}(m\log(s))$ time, where L consists of all minmer start and end positions and is sorted on genomic position in ascending order. In this way, we can iterate through the list and keep a running count of the overlapping minmer intervals by incrementing the count for each start-point and decrementing the count for each end-point.

Unlike the previous versions of MashMap that look for all mappings above a certain ANI threshold, MashMap3 provides the option to instead filter out all mappings, which are not likely to be within Δ_{ANI} of the best predicted mapping ANI. This significantly reduces the number and size of the candidate regions passed on to the more expensive second stage.

This filter, described in more detail in Supplementary Section S1.1, leverages the fact that the numerator of the minhash formula for A and B_i is hypergeometrically distributed when conditioned on $|\pi_s(A)\cap\pi_s(B_i)|$. As a result, we can obtain the probability distribution of the minhash for a mapping using the cardinality of the intersection of the minmers alone. MashMap3 then uses these distributions filter to out any candidate mappings where the probability of the candidate mapping being within Δ_{ANI} of the best candidate mapping is below some threshold.

3.2.2 Stage 2: efficiently computing the rolling minhash

Given a candidate region $[a, z)$, the goal of Stage 2 is to calculate the minhash for all A, B_i pairs for $i\in[a,z)$. In order to track the minhash of A and B_i for each i, MashMap2 previously used a sorted map to track all active seeds in each window. We improve upon this by observing that the minhash can be efficiently tracked using only $\pi_s(A)$, $\pi_s(A)\cap\pi_s(B_i)$, and the number of minmers from $\pi_s(B_i)$ in-between each consecutive pair of minmers from $\pi_s(A)$. This allows MashMap3 to use a static array of s elements to compute the rolling minhash estimate for each window. While each iteration requires a binary search on the array and therefore has the same complexity as modifying an ordered map, $\mathcal{O}(\log s)$, the performance of the array implementation in practice is much faster. The details of this data structure can be found in Supplementary Section S1.2.

3.2.3 Early termination of Stage 2

Instead of computing the Stage 2 step for each candidate region obtained in the first stage, we aim to terminate the second stage once we have confidently identified all mappings whose predicted ANI is within Δ_{ANI} of the best predicted ANI. We do this by sorting the candidate regions in decreasing order of their maximum interval overlap size obtained in Stage 1. The Stage 2 minhash calculation is then performed on each candidate region in order, keeping track of the best predicted ANI value seen. Let κ be numerator of the minhash that corresponds to an ANI value Δ_{ANI} less than the best predicted ANI value seen so far and Y_i be a random variable for the numerator of the minhash for A and B_i. Then, given a candidate region with a maximum overlap size of $c_i<\kappa$, we know that $\Pr(Y_i\geq\kappa)=0$ and therefore no more candidate regions can contain mappings whose predicted ANI is within Δ_{ANI} of the predicted ANI of the best mapping.

4 Results

4.1 Characteristics of the minmer scheme

Here, we provide formulas for the density of minmers and minmer intervals and an approximation for the distance between adjacent minmers. Proofs of the formulas are presented in the Supplementary Material. We then compare these formulas to results on both simulated and empirical sequences. For the simulated dataset, we generated a sequence of 1 million uniform random hash values. For the empirical dataset, we used MurmurHash to hash the sequence of k-mers in the recently-completed human Y-chromosome (Rhie *et al.* 2022) with $k=18$.

4.1.1 Minmer density

To obtain the formula for the minmer density, we consider how the rank of a random k-mer changes with each consecutive window that contains it. As a result, we have a distribution of the rank of a random k-mer throughout consecutive sliding windows. This distribution enables us to not only obtain the density (Fig. 2), but also determine other characteristics such as the likelihood of being a minmer given some initial rank r_1 or given a hash value z.

Theorem 4.1. *Let $d_{(w,s)}$ be the expected density of (w,s)-minmers in a random sequence. Then,*

$$d_{(w,s)}=\frac{1}{w}\sum_{r_1,r_w\in\{1\ldots w\}}\Pr(C=1|r_1,r_w)\Pr(R_w=r_w|r_1),$$

where $R_w|r_1\sim\mathrm{BetaBinomial}(r_1,w-r_1+1)$ and

$$\Pr(C=1|r_1,r_w)=\begin{cases}\displaystyle\sum_{u=0}^{\delta}\Pr(U=u)\frac{\binom{2u+r_w-r_1}{u+r_w-s}}{\binom{2u+r_w-r_1}{u}} & r_1,r_w>s\\ 1 & \text{otw}\end{cases},$$

where $U\sim\mathrm{Hypergeometric}(w-1,r_1-1,w-r_w)$ and $\delta=\min(r_1-1,w-r_w)$.

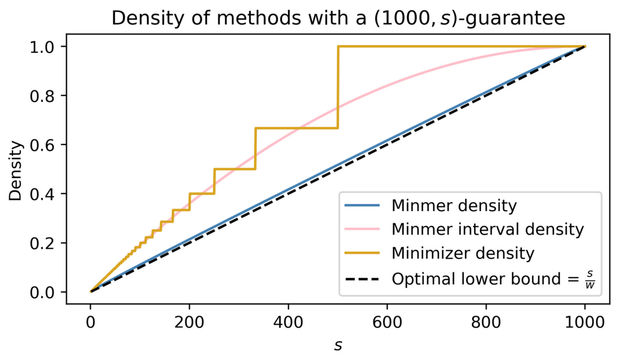


Figure 2. The density and interval density of a $(1000, s)$-minmer scheme compared to a w'-minimizer scheme, which also yields a $(1000, s)$-window guarantee. To ensure that the minimizer scheme satisfies the $(1000, s)$-window guarantee, the minimizer scheme is set with $w'=\lfloor 1000/s\rfloor$

4.1.2 Minmer interval density

Theorem 4.2. *Let $d^*_{(w,s)}$ be the density of (w,s)-minmer intervals in a random sequence, i.e. the probability that for a randomly selected position i, $\pi_s(W_i) \neq \pi_s(W_{i-1})$. Then,*

$$d^*_{(w,s)} = 1 - \frac{(w-s+1)(w-s)}{w(w+1)}.$$

We can use $d^*_{(w,s)}$ to provide an expectation on the number of elements in our minmer interval index I. As expected, letting $s=1$ yields the same density as minimizers, $2/(w+1)$, and a similar formula appears when determining the probability of observing s consecutive unsampled k-mers under the minimizer scheme (Spouge 2022). As the number of minmers is a strict lower bound on the number of minmer intervals, this result also gives an upper bound on the density of (w,s)-minmers.

4.1.3 Minmer window guarantee

As the main difference between minimizers and minmers is the window guarantee, it is important to observe the difference in the density of the minmer scheme compared to a minimizer scheme, which also satisfies the (w,s)-window guarantee. In Fig. 2, we consider the case where we have a $(1000,s)$-minmer scheme and a w'-minimizer scheme, where w' is set to obtain the same $(1000,s)$-window guarantee of the minmer scheme by letting $w' = \lfloor 1000/s \rfloor$. We observe that for sketch sizes other than 1 and 1000, for which the density of the schemes is equal, the density of the minmer scheme is strictly less than the density of the corresponding minimizer scheme. For some values of s, the density of the $\lfloor 1000/s \rfloor$-minimizer scheme is over 70% larger than the $(1000,s)$-minmer scheme.

4.1.4 Minmer spread

Let G_i be the distance between the ith selected minmer and the $(i+1)$th selected minmer. For a (w,s)-minmer scheme with a density factor d_f, we have that

$$\Pr(G_i = d) \approx \frac{\binom{w-d}{d_f-2}}{\binom{w}{d_f-1}}.$$

To see how well this approximation holds, we plot the results on both empirical and simulated data in Supplementary Fig. S2.

4.2 ANI prediction ideal sequences

We replicated the experiments for Table 1 of Belbasi *et al.* (2022) using the minmer-based MashMap3 (commit `4f4df5d`), with the exception that we report the mean predicted sequence divergence error as opposed to the median. Results for the relative median error are similar and can be found in Supplementary Fig. S3. For each divergence rate $r \in \{0.01, 0.05, 0.10\}$, 100 random windows of 10 000 bp were selected from the *Escherichia coli* genome and 10 000r positions were selected at random and mutated, ensuring that no duplicate k-mers were generated. The reads were mapped back to the reference *E.coli* genome and the predicted divergence was compared to the ground truth (Fig. 3).

The parameters of the minmer-based MashMap3 were set to obtain a similar number of sampled k-mers as the minimizer-based MashMap2 under MashMap2's default density of 0.009. Both MashMap2 and MashMap3 were run with $k=19$. As expected, the results show that the ANI values predicted by the minmer scheme are significantly closer to the ground truth than those predicted by the minimizer scheme. Notably, in the case where the true divergence was 1%, the relative error is reduced from 29.5% to 2.6% (Fig. 3).

4.3 ANI prediction on simulated reads

In addition to the ANI prediction measurements from Belbasi *et al.* (2022), we also simulated reads from the human T2T-CHM13 reference genome (Nurk *et al.* 2022) at varying error rates to determine the accuracy of the ANI predictions. We compared the minmer-based MashMap3 against the minimizer-based MashMap2 with similar densities for each run as well as against Minimap2 (Li 2018). While there have been other recent advancements in approximate read mapping, these tools either do not report the estimated ANI [e.g. Ekim *et al.* (2022)] or are based on Minimap2 [e.g. Jain *et al.* (2022) and Firtina *et al.* (2023)]. Minimap2 was run in its default mode with `-x map-ont` set, which, like MashMap, computes approximate mappings and estimates the alignment identity. MashMap2 was modified to use the binomial model for estimating the ANI from the Jaccard estimator, which has been shown to be more accurate (Belbasi *et al.* 2022).

We used Pbsim (Ono *et al.* 2013) to simulate three datasets: "CLR-95," "CLR-98," and "CLR-99," where the number following the dash represents the average ANI across reads. The standard deviation of the error rates was set to 0, and the ratio of matches, insertions, and deletions was set to 20:40:40, respectively, to ensure that mapped regions would, on average, be the same length as the reads. For each dataset, 5000 bp reads were generated with the CLR profile at a depth of two, resulting in 1.25 million reads for each dataset.

The predicted ANIs were then compared to the gap-compressed ANIs of the ground-truth mapping, where the

Table 1. Metrics for simulated Nanopore read mapping to the human genome.[a]

	Minimap2				MashMap2				MashMap3			
Dataset	CPU time (m)	Memory (GB)	ME	MAE	CPU time (m)	Memory (GB)	ME	MAE	CPU time (m)	Memory (GB)	ME	MAE
CLR-99	154.20	**9.89**	−0.25	0.34	80.27	9.92	−0.27	0.29	**33.64**	13.07	**0.03**	**0.17**
CLR-98	147.29	**9.89**	−0.36	0.52	82.46	9.92	−0.33	0.39	**35.13**	13.09	**0.06**	**0.29**
CLR-95	96.35	**9.89**	−0.46	0.81	106.81	9.92	−0.25	**0.59**	**42.81**	13.10	**0.21**	0.62

[a] Minmer and minimizer-based MashMap implementations as well as Minimap2 were used to map simulated reads from the human reference genome using Pbsim (Ono *et al.* 2013) and the mean error and mean absolute error are reported. Bolded values signify the best performance for each dataset.

gap-compressed ANI formula is analogous to the standard ANI formula with the exception that consecutive gap columns are counted as a single gap column. The use of gap-compressed ANI is motivated by the fact that it is less sensitive to homopolymer errors and long indels. To measure bias and magnitude of error, we report the mean error (ME) and mean absolute error (MAE). The results of the simulations can be seen in Table 1, with the median errors reported in Supplementary Table S1.

For MashMap2 and MashMap3, we used a k-mer size of 19 and set the MashMap2 minimizer w to 89 and minmer s to $s = 100$ obtain a density of 0.0222 for both tools. The ANI cutoff was set to 94%, 93%, and 90% for the CLR-99, CLR-98, and CLR-95 datasets, respectively. The indexing times for Minimap2, MashMap2, and MashMap3 were 1.7, 2.8, and 9.8 min, respectively. Indexing times for MashMap2 and MashMap3 across varying densities can be found in Supplementary Fig. S4.

4.4 ANI prediction on mammalian genome alignments

To test the performance of MashMap3 at the genome-mapping scale, we computed mappings between the T2T human reference genome and reference genomes for chimpanzee (Kronenberg *et al.* 2018) and macaque (Warren *et al.* 2020). In absence of ground-truth ANI values, we used wfmash (Guarracino *et al.* 2021) to compute the gap-compressed ANI of the segment mappings output by MashMap and report the results of the mappings with $\geq 80\%$ complexity in Table 2. For a small proportion of segment mappings output by MashMap2 and MashMap3, wfmash did not produce an alignment. When the ANI threshold is 85%, these cases accounted for 0.07% of chimpanzee mappings and 0.3% macaque mappings. When the ANI threshold was 90% or 95%, <0.01% of mappings were not aligned with wfmash for both chimpanzee and macaque. We consider these mappings as false positives. For the ANI thresholds of 95%, 90%, and 85%, the winnowing scheme densities were set to 0.043, 0.053, and 0.064, respectively.

To isolate the effect of the new seeding method, we turned chaining off for both tools. As the Jaccard estimator is known to perform poorly in the presence of many degenerate k-mers, results for query regions above and below 80% complexity are reported separately, where complexity is defined as the ratio of observed distinct k-mers in a region to w. Low-complexity mappings make up for at most 1% and 3% of the mappings for chimpanzee and macaque genomes, respectively. We show the table of the metrics for the low-complexity mappings in Supplementary Table S3.

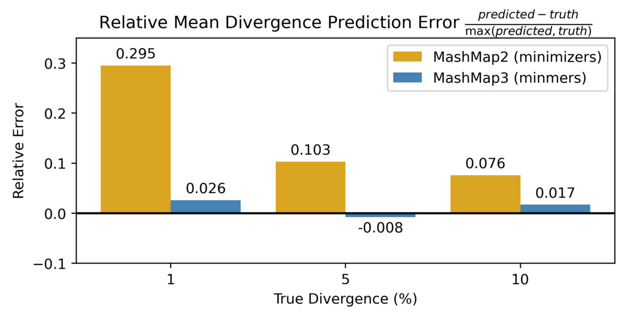


Figure 3. Eliminating the bias in MashMap. The experiments from Table 1 of Belbasi *et al.* (2022) were replicated. Divergence, defined as 1-ANI, was predicted across 100 sequences for both MashMap2 and MashMap3 using a density of 0.009 ($w = 10\,000$, $s = 78$)

5 Discussion

Minmers are a novel "non-forward" winnowing scheme with a (w, s)-window guarantee. Similar to what has been done for other proposed schemes, we have derived formulas (approximate and exact) that describe the scheme's characteristics. We have replaced minimizers with minmers in MashMap3 and demonstrated that minmers eliminate Jaccard estimator bias and enable new methods to reduce mapping runtime compared to MashMap2. In addition, we show that minmers require substantially less density than minimizers when a (w, s)-window guarantee is required.

5.1 The minmer scheme enables sparser sketches

The minimizer winnowing scheme has long been the dominant method for winnowing due to its $(w, 1)$-window guarantee, simplicity, and performance. Other 1-local methods, such as strobemers (Sahlin 2021) and syncmers (Edgar 2021) remove the window guarantee and rely on a random sequence assumption to provide probabilistic bounds on the expected distance between sampled k-mers.

Minmers represent a novel class of winnowing schemes that extend the window guarantee of minimizers. Unlike strobemers, syncmers, and other 1-local methods, the minmer scheme guarantees a lower bound on the number of k-mers sampled from a window, so long as it contains at least s distinct k-mers. This is particularly desirable for accurate Jaccard estimation and the winnowing of low-complexity

Table 2. Comparison of MashMap2 and MashMap3 for identifying mappings between pairs of mammalian genomes.[a]

		MashMap2					MashMap3				
Query species	ANI threshold (%)	Basepairs mapped (Gb)	CPU time (m)	Memory (GB)	ME	MAE	Basepairs mapped (Gb)	CPU time (m)	Memory (GB)	ME	MAE
Chimpanzee	95	2.80	39.76	**19.95**	−0.25	0.29	**2.81**	**32.76**	27.07	**0.01**	**0.22**
Chimpanzee	90	2.82	118.31	**24.55**	−0.22	0.29	2.82	**51.12**	36.20	**0.01**	**0.25**
Chimpanzee	85	2.83	787.44	44.96	−0.18	0.27	2.83	**64.48**	**39.47**	**0.02**	**0.25**
Macaque	95	0.38	30.0	**20.83**	**0.29**[b]	**0.46**	**1.08**	**28.67**	28.97	0.57[b]	0.66
Macaque	90	2.54	40.49	**23.04**	−0.30	**0.69**	**2.56**	**34.87**	35.91	**0.01**	0.74
Macaque	85	2.60	446.71	**38.13**	−0.24	**0.74**	**2.61**	**43.74**	39.49	**0.05**	0.87

[a] MashMap2 and MashMap3 were used to align the human reference genome to chimpanzee and macaque genomes. The mean error and mean absolute error metrics shown are for query segments with at least 80% k-mer complexity. Bolded values signify the best performance for each dataset. Corresponding metrics for low-complexity mappings can be found in Supplementary Table S3.

[b] Sampling bias leads to ANI over-estimation (see Section 5 for details).

sequence where the density of sampled k-mers from 1-local schemes can vary significantly.

Unlike the (w, s)-minmer scheme, a $\lfloor w/s \rfloor$-minimizer scheme satisfies both the (w, s)-window guarantee and the $(\lfloor w/s \rfloor, 1)$-window guarantee. However, this minimizer scheme does not yield an unbiased Jaccard estimator. Notably, the density of the $\lfloor w/s \rfloor$-minimizer scheme tracks closely with the density of (w, s)-minmer intervals (Fig. 2), which, while not necessary for the use of minmers, serve as a helpful auxiliary index for improving query performance in MashMap3.

Additionally, while the (w, s)-minmer scheme does not provide as strong constraints on distances between adjacent seeds as the $\lfloor w/s \rfloor$-minimizer scheme does, we provide an approximate distribution on the distance between adjacent minmers in Section 4.1.4 and show that the distribution holds up in both simulated and empirical data (Supplementary Fig. S2).

5.2 Minmers yield an unbiased estimator at lower computational costs

Indexing minmers rather than minimizers removes the Jaccard estimator bias present in earlier versions of MashMap. For any window, the set of sampled k-mers is guaranteed to be a superset of the bottom-s sketch of that window. Therefore, running the minhash algorithm on minmers yields the same estimator as running the minhash algorithm on the full set of k-mers.

In addition to the experiments from Belbasi *et al.* (2022), which focus on "ideal" sequences with no repetitive k-mers, we also measured the performance of the ANI prediction for different levels of divergence on the human genome across mappings of simulated reads and a sample of mammalian genomes. Our results showed that MashMap3 with minmers not only produced unbiased and more accurate predictions of the ANI than Minimap2 and MashMap2, but it did so in a fraction of the time.

We replicated the behavior of minimizers to under-predict ANI as seen in Belbasi *et al.* (2022) across all experiments. At the same time, in both the simulated reads and empirical genome alignment results, we see that MashMap3 slightly over-predicts the ANI at larger divergences. Further inspection reveals that this is due to indels in the alignment, which are not modeled by the binomial mutation model used to convert the Jaccard to ANI (Supplementary Table S2).

The optimizations to the second stage of mapping combined with the minmer interval indexing leads to significantly better mapping speeds in MashMap3. Relative to Minimap2 and MashMap2, MashMap3 spends a significant amount of time indexing the genome (Supplementary Fig. S4). This, however, serves as an investment for the mapping phase, which is significantly faster than MashMap2, particularly at lower ANI thresholds. The tradeoff of indexing time for mapping speedups is particularly useful for large references, such as pangenomes, as the quadratic time complexity of alignment dominates the linear time complexity of indexing. As an additional feature, MashMap3 provides the option to save the reference index so that users can leverage the increased mappings speeds for previously indexed genomes.

Similar to MashMap2, MashMap3 by default uses the plane-sweep post-processing algorithm described in Jain *et al.* (2018a) to filter out redundant segment mappings. We show that by using the probabilistic filtering method described in Section 3.2.1, we can discard many of these mappings at the beginning of the process as opposed to the end, yielding significant runtime improvements. As the purpose of the probabilistic filtering is to remove weaker mappings in the presence of stronger mappings at an earlier stage, the speedup becomes more prominent as the ANI threshold is decreased (Table 2).

MashMap3 is significantly more efficient at lower ANI thresholds, which is helpful for detecting more distant homologies. For example, in the human–macaque mapping, we recovered an additional 50 Mb of mapped sequence by reducing the ANI threshold from 90% to 85% while also completing over 10× quicker than MashMap2. It is also worth noting that the default ANI of MashMap2 and MashMap3 is 85%, and often the ANI of homologies between genomes is not known *a priori*.

Further motivating the improved efficiency of low ANI thresholds is the fact that thresholds above the true ANI can lead to recovering mappings, which over-predict the ANI while discarding those which accurately or under-predict the ANI. This sampling bias leads to an increase in the ANI estimation bias. We see this behavior in the human–macaque alignment with a threshold 95% ANI (Table 2). At lower ANI thresholds, we observe that the majority of mappings are in the 90%–95% ANI range.

5.3 Limitations and future directions

MashMap's Jaccard-based similarity method tends to overestimate ANI in low-complexity sequences. For downstream alignment applications, the resulting false-positive mappings can be pruned using a chaining or exact alignment algorithm to validate the mappings. Unreliable ANI estimates could also be flagged by using the bottom-s sketch to determine the complexity of a segment as described in Cohen and Kaplan (2007), but a sketching method and distance metric that better approximates ANI across all sequence and mutational contexts would be desirable.

An important characteristic of MashMap is the relatively few parameter settings necessary to tune across different use cases. Building on this, we aim to develop a methodology that can find maximal homologies without a pre-determined segment size, similar to the approach of Wang *et al.* (2022b).

6 Conclusion

In this work, we proposed and studied the characteristics of the minmer scheme and showed that they belong to the unexplored class of non-forward local schemes, which have the potential to achieve lower densities under the same locality constraints as forward schemes (Marçais *et al.* 2018). We derived formulas for the density and approximate spread of minmers, enabling them to be objectively compared to other winnowing schemes.

By construction, minmers, unlike minimizers, enable an unbiased estimation of the Jaccard. We replaced the minimizer winnowing scheme in MashMap2 with minmers and showed that minmers significantly reduce the bias in both simulated and empirical datasets.

Through leveraging the properties of the minmers, we implemented a number of algorithmic improvements in MashMap3. In our experiments, these improvements yielded significantly lower runtimes, particularly in the case when the ANI threshold of MashMap is set to the default of 85%. With the improvements in MashMap3, it is no longer necessary to estimate the ANI of homologies *a priori* to avoid significantly longer runtimes, making it an ideal candidate for a broad range of comparative genomics applications.

Acknowledgements

We would like to thank Chirag Jain for helpful discussions and his implementation of the original MashMap software, Andrea Guarracino for improvements and discussions, and Nicolae Sapoval for his feedback on proofs. We would also like to thank our reviewers for their insightful comments.

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest

None declared.

Funding

This work was supported by the National Library of Medicine Training Program in Biomedical Informatics and Data Science [grant number T15LM007093 to B.K.]; the Intramural Research Program of the National Human Genome Research Institute, National Institutes of Health to A.M.P.; in part, the National Institute of Allergy and Infectious Diseases [grant number P01-AI152999 to B.K. and T.J.T.]; in part, National Science Foundation [grant number EF-2126387 to T.J.T.]; National Institutes of Health/National Institute on Drug Abuse [U01DA047638 to E.G.]; National Institutes of Health/National Institute of General Medical Sciences [R01GM123489 to E.G.]; National Science Foundation Principles and Practice of Scalable Systems [award number 2118709 to E.G.]; the Tennessee Governor's Chairs program to E.G.

Data availability

No new data were generated or analysed in support of this research.

References

Baker DN, Langmead B. Dashing: fast and accurate genomic distances with hyperloglog. *Genome Biol* 2019;**20**:265.

Belbasi M, Blanca A, Harris RS *et al.* The minimizer Jaccard estimator is biased and inconsistent. *Bioinformatics* 2022;**38**:i169–76.

Berlin K, Koren S, Chin C-S *et al.* Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nat Biotechnol* 2015;**33**:623–30.

Blanca A, Harris RS, Koslicki D *et al.* The statistics of k-mers from a sequence undergoing a simple mutation process without spurious matches. *J Comput Biol* 2022;**29**:155–68.

Broder AZ. On the resemblance and containment of documents. In: *Proceedings: Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171), Salerno, Italy*. 21–9. IEEE, 1997.

Brown CT, Irber L. sourmash: a library for MinHash sketching of DNA. *JOSS* 2016;**1**:27.

Cohen E, Kaplan H. Summarizing data using bottom-k sketches. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing*. 225–34. 2007.

Cohen E. Min-hash sketches. 2016. http://www.cohenwang.org/edith/Surveys/minhash.pdf.

Dilthey AT, Jain C, Koren S *et al.* Strain-level metagenomic assignment and compositional estimation for long reads with metamaps. *Nat Commun* 2019;**10**:3066.

Edgar R. Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. *PeerJ* 2021;**9**:e10805.

Ekim B, Berger B, Orenstein Y. A randomized parallel algorithm for efficiently finding near-optimal universal hitting sets. In: *Proceedings: 2020 Research in Computational Molecular Biology, Padua, Italy*. 37–53. Springer International Publishing, 2020.

Ekim B, Sahlin K, Medvedev P *et al.* Efficient mapping of accurate long reads in minimizer space with mapquik. *Genome Res* 2023;**33**:1188–97.

Firtina C, Park J, Alser M *et al.* Blend: a fast, memory-efficient and accurate mechanism to find fuzzy seed matches in genome analysis. *NAR Genom Bioinform* 2023;**5**:lqad004.

Guarracino A, Mwaniki N, Marco-Sola S *et al.* wfmash: a pangenome-scale aligner. 2021. https://zenodo.org/record/8207753. https://github.com/waveygang/wfmash/tree/v0.10.5.

Jain C, Dilthey A, Koren S *et al.* A fast approximate algorithm for mapping long reads to large reference databases. In: *Proceedings: 2017 Research in Computational Molecular Biology, Hong Kong, China*. 66–81. Springer International Publishing, 2017.

Jain C, Koren S, Dilthey A *et al.* A fast adaptive algorithm for computing whole-genome homology maps. *Bioinformatics* 2018a;**34**:i748–56.

Jain C, Rhie A, Hansen NF *et al.* Long-read mapping to repetitive reference sequences using Winnowmap2. *Nat Methods* 2022;**19**:705–10.

Jain C, Rodriguez-R LM, Phillippy AM *et al.* High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nat Commun* 2018b;**9**:5114–8.

Kronenberg ZN, Fiddes IT, Gordon D *et al.* High-resolution comparative analysis of great ape genomes. *Science* 2018;**360**:eaar6343.

Li H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 2018;**34**:3094–100.

Marçais G, DeBlasio D, Kingsford C *et al.* Asymptotically optimal minimizers schemes. *Bioinformatics* 2018;**34**:i13–22.

Marçais G, Pellow D, Bork D *et al.* Improving the performance of minimizers and winnowing schemes. *Bioinformatics* 2017;**33**:i110–7.

Marco-Sola S, Moure JC, Moreto M *et al.* Fast gap-affine pairwise alignment using the wavefront algorithm. *Bioinformatics* 2021;**37**:456–63.

Nurk S, Koren S, Rhie A *et al.* The complete sequence of a human genome. *Science* 2022;**376**:44–53.

Ondov BD, Starrett GJ, Sappington A *et al.* Mash screen: high-throughput sequence containment estimation for genome discovery. *Genome Biol* 2019;**20**:232.

Ondov BD, Treangen TJ, Melsted P *et al.* Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biol* 2016;**17**:132.

Ono Y, Asai K, Hamada M *et al.* PBSIM: PacBio reads simulator—toward accurate genome assembly. *Bioinformatics* 2013;**29**:119–21.

Popic V, Batzoglou S. A hybrid cloud read aligner based on minhash and kmer voting that preserves privacy. *Nat Commun* 2017;**8**:15311–7.

Rhie A, Nurk S, Cechova M *et al.* The complete sequence of a human Y chromosome. Nature, 2023.

Roberts M, Hayes W, Hunt BR *et al.* Reducing storage requirements for biological sequence comparison. *Bioinformatics* 2004;**20**:3363–9.

Sahlin K. Effective sequence similarity detection with strobemers. *Genome Res* 2021;**31**:2080–94.

Schleimer S, Wilkerson D, Aiken A. Winnowing: local algorithms for document fingerprinting. In: *Proceedings: 2003 ACM SIGMOD International Conference on Management of Data. San Diego, California, USA*. 76–85. New York, NY, USA: Association for Computing Machinery, 2003.

Shaw J, Yu YW. Theory of local k-mer selection with applications to long-read alignment. *Bioinformatics* 2022;**38**:4659–69.

Shaw J, Yu YW. Fast and robust metagenomic sequence comparison through sparse chaining with skani. bioRxiv, 2023, preprint: not peer reviewed.

Spouge JL. A closed formula relevant to 'Theory of local k-mer selection with applications to long-read alignment' by Jim Shaw and Yun William Yu. *Bioinformatics* 2022;**38**:4848–9.

Wang T, Antonacci-Fulton L, Howe K *et al.*; Human Pangenome Reference Consortium. The human pangenome project: a global resource to map genomic diversity. *Nature* 2022a;**604**:437–46.

Wang Z, Zuo C, Deng D. TxtAlign: efficient near-duplicate text alignment search via bottom-k sketches for plagiarism detection. In: *Proceedings: 2022 ACM SIGMOD International Conference on Management of Data. Philadelphia, PA, USA*. 1146–59. New York, NY, USA: Association for Computing Machinery, 2022b.

Warren WC, Harris RA, Haukness M *et al.* Sequence diversity analyses of an improved rhesus macaque genome enhance its biomedical utility. *Science* 2020;**370**:eabc6617.

Zheng H, Kingsford C, Marçais G *et al.* Improved design and analysis of practical minimizers. *Bioinformatics* 2020;**36**:i119–27.