Using syncmers improves long-read mapping

Abhinav Dutta[1†], David Pellow[2†], Ron Shamir[2*]

[1]Computer Science and Engineering, India Institute of Technology Patna, Patna, India
[2]Blavatnik School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel

[†]Equal contribution
[*]To whom correspondence should be addressed: rshamir@tau.ac.il

Abstract: As sequencing datasets keep growing larger, time and memory efficiency of read mapping are becoming more critical. Many clever algorithms and data structures were used to develop mapping tools for next generation sequencing, and in the last few years also for third generation long reads. A key idea in mapping algorithms is to sketch sequences with their minimizers. Recently, syncmers were introduced as an alternative sketching method that is more robust to mutations and sequencing errors.

Here we introduce parameterized syncmer schemes, and provide a theoretical analysis for multi-parameter schemes. By combining these schemes with downsampling or minimizers we can achieve any desired compression and window guarantee. We introduced syncmer schemes into the popular minimap2 and Winnowmap2 mappers. In tests on simulated and real long read data from a variety of genomes, the syncmer-based algorithms reduced unmapped reads by 20-60% at high compression while using less memory. The advantage of syncmer-based mapping was even more pronounced at lower sequence identity. At sequence identity of 65-75% and medium compression, syncmer mappers had 50-60% fewer unmapped reads, and $\sim 10\%$ fewer of the reads that did map were incorrectly mapped. We conclude that syncmer schemes improve mapping under higher error and mutation rates. This situation happens, for example, when the high error rate of long reads is compounded by a high mutation rate in a cancer tumor, or due to differences between strains of viruses or bacteria.

1 Introduction

As the volume of third-generation, long read sequencing data increases, new computational methods are needed to efficiently analyze massive datasets of long reads. One of the most basic steps in analysis of sequencing data is mapping reads to a known reference sequence or to a database of many sequences. Several long read mappers have been proposed [6, 15], with minimap2 [8] being the most popular. minimap2 is a multi-purpose sequence aligner that uses sequence minimizers as alignment anchors. Minimizers, the minimum valued k-mers in windows of w overlapping k-mers of a sequence, are used to sketch sequences. They have greatly improved computational efficiency of many different sequence analysis algorithms (e.g. [2], [7], [17]). A key criterion in evaluating minimizer schemes is their *density*, which is the fraction of k-mers selected. The inverse of the density is called the *compression rate* of the scheme.

Recent work has shown that minimizers are less effective under high error or mutation rates. Motivated by this observation, Edgar [4] recently introduced a novel family of k-mer selection schemes called *syncmers*. Syncmers are a set of k-mers defined by the position of their minimum s-long substring (s-minimizer). Syncmers constitute a predetermined subset of all possible k-mers and, unlike minimizers, they are defined by the sequence of the k-mer only and do not depend on the rest of the sequence in which they appear. Syncmers are therefore more likely to be conserved under mutations. In contrast, minimizers are selected depending on a larger window, which is more likely to contain mutations or errors. This difference is crucial in long read data, which has much higher error rate than short reads [3]. Another key difference between syncmer and minimizer schemes is that the latter guarantee selection of a k-mer in every window of w consecutive k-mers (this is called a *window guarantee*), while syncmers do not.

Edgar defined several syncmer variants, including the families of *open syncmers*, whose s-minimizer appears at one specific position, and *closed syncmers*, whose s-minimizer appears at either the first *or* the last position in the k-mer [4]. He computed the properties of a range of syncmer schemes and used them to choose a scheme with a desired compression rate. Shaw and Yu [16] recently formalized the notions of the conservation of selected positions and their clustering along a sequence, and provided a broader theoretical analysis.

In this work we generalize syncmer schemes to multiple arbitrary s-minimizer positions. We call these *parameterized syncmer schemes* (PSSs), where the parameters are the possible indices of the s-minimizer in a selected k-mer, and an n-parameter scheme uses n such indices. An example is a 3-parameter scheme that selects any 15-mer with the minimum 5-mer appearing at position 1, 5, or 9. We analyze the properties of such parameterized syncmer schemes and determine which schemes perform well for a given compression rate through theoretical analysis and empirical testing.

When using PSSs in practice, the selected k-mers must often be downsampled to achieve a desired compression rate. We demonstrate that it is possible to retain properties of syncmers such as minimum and most frequent distances between selected positions by choosing the correct parameters and downsampling rate of a PSS. We can also have a window guarantee by combining syncmers and minimizers.

Many read mappers work by indexing seeds in a reference sequence and then finding a chain of matching seeds, forming a segment with high scoring alignment with the query sequence. In the long read mapper minimap2 [8], minimizers are used as the seeds, with the advantage that any identical window of length w will have the same minimizer. However, for longer reads with a much higher error rate, conservation of the selected k-mers becomes more important than the window guarantee, especially when there are also mutations. For example, it was shown that with 90% identity between aligned sequences, only about 30% of the positions on the sequence will overlap a conserved minimizer in minimap2 [4].

We introduced syncmer schemes into two leading long read mappers: the latest release of minimap2 [9] and Winnowmap2 [5]. Winnowmap extended minimap2 and re-weighted the minimizers by frequency to obtain a better distribution of minimizers and improve mapping, especially in highly repetitive regions [6]. Winnowmap2 achieved even better performance in repetitive regions by replacing the extension phase of minimap2's seed-and-extend alignment algorithm by aligning minimal confidently alignable substrings that do not contain non-reference alleles. The latest version of minimap2 was reported to have closed the performance gap between the mappers [9].

We compared minimap2, Winnowmap2, and their modified syncmer-based versions on both simulated and real long read data. The syncmers increased the number of mapped reads across a large range of compression rates, resulting in 20-60% fewer unmapped reads at high compression. Even at lower compression, the syncmer mappers had 2-15% fewer unmapped reads. The syncmer versions took less memory but had somewhat longer mapping times. The most marked improvements were observed when percent identity of the mapped reads and reference sequences were low. With percent identity of 65 and 75% and medium compression, syncmer mappers had 50-60% fewer unmapped reads and still had 8-13% fewer incorrectly mapped reads.

While conducting this research, Shaw and Yu released a preprint that modified minimap2 to use open syncmers [16]. That work focused mostly on the theoretical properties of syncmers and provided a foundation and justification for their use. Our work greatly extends the practical use of syncmers and builds on these theoretical foundations.

The paper is structured as follows: Section 2 provides background, definitions, and terminology; Section 3 provides some theoretical analysis of PSSs; Section 4 describes the practical implementation of PSSs and their integration into minimap2 and Winnowmap2; Section 5 presents experimental results of the original and PSS-modified mappers; Section 6

discusses the results and future work.

2 Definitions and Background

2.1 Basic definitions and notations

For a string S over the alphabet Σ, a *k-mer* is a k-long contiguous substring of S. The k-mer starting at position i is denoted $S[i, i+k-1]$ (string indices start from 1 throughout). We work with the nucleotide alphabet: $\Sigma = \{A, C, G, T\}$.

k-mer order: Given a one-to-one hash function on k-mers $o : \Sigma^k \to \mathbb{R}$, we say that k-mer x_1 is less than x_2 if $o(x_1) < o(x_2)$. Examples include lexicographic encoding or random hash. We will denote this as $x_1 < x_2$ when o is clear from the context. In this work we use a random order unless otherwise noted.

Canonical k-mers: Denote the reverse complement of x by $\bar{x}$. For a given order, the *canonical form* of a k-mer x, denoted by $Canonical(x)$, is the smaller of x and $\bar{x}$. For example, under the lexicographic order, $Canonical(CGGT) = ACCG$.

2.2 Selection schemes

Selection scheme: A *selection scheme* is a function from a string to the indices of positions in it $f : \Sigma^* \to \mathbb{N}$. The scheme implicitly selects the k-mers starting at these positions. For a string $S \in \Sigma^*$, $f_k(S) = \{i_1, i_2, ..., i_n\}$ is the set of start indices of the k-mers selected by the scheme.

Minimizers: A *minimizer scheme* chooses the position of the minimum value k-mer in every window of w consecutive k-mers in S:

$$\mathcal{M}_{k,w,o}(S) = \bigcup_{j=1}^{|S|-w-k+2} \left\{ \underset{i:i\in j...j+w-1}{\operatorname{argmin}} Canonical(S[i, i+k-1]) \right\}$$

where the minimum is according to k-mer ordering o. By convention, ties are broken by choosing the leftmost position. An example of a minimizer selection scheme is shown in Figure 1A. By definition, minimizers select a k-mer in every window of w k-mers. This property is called a *window guarantee*.

Parameterized syncmers: A *parameterized syncmer scheme* (PSS) with parameters $0 < x_1 < ... < x_{n-1} < x_n \leq k - s + 1$ selects a k-mer if the minimum s-mer of that k-mer appears at one of the positions x_i in the k-mer:

$$f_{k,s,o,\{x_1,...,x_n\}}(S) \equiv Sy_{k,s,o}(x_1, ..., x_n)(S) = \{i | \mathcal{M}_{s,k-s+1,o}(Canonical(S[i, i+k-1])) \in \{x_1, ..., x_n\}\}$$

As o is fixed we will drop it from the notation where possible. An example of a PSS is shown in Figure 1B. We denote the family of all n-parameter syncmers as Sy_n. Note that,

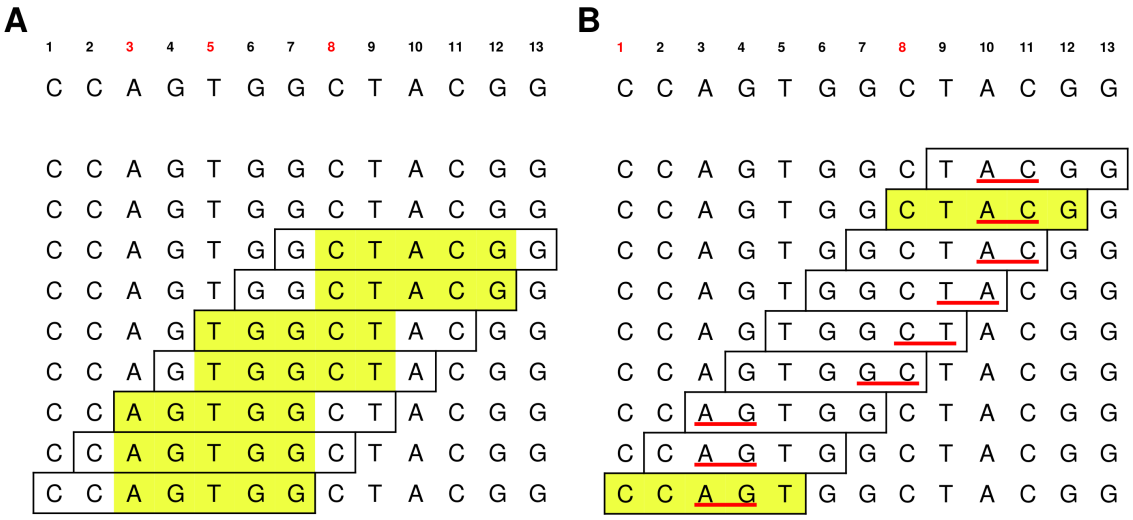

Figure 1: **Minimizer and syncmer schemes.** In both examples the lexicographic order is used and the underlying sequence is shown at the top. **(A)** Minimizers. Here $w = 3$ and $k = 5$, so the minimizer is the least 5-mer in every window of length 7. The minimizer of each window is highlighted in yellow; **(B)** Syncmers. Here we show the 1-parameter syncmer with $k = 5$, $s = 2$ and $t = 3$, $Sy_{5,2,lex}(3)$. It selects 5-mers if their 2-minimizer appears at position 3. The 2-minimizer in each 5-mer is underlined in red. 2-minimizers at position 3 are highlighted in yellow. The start positions of the k-mers in the underlying sequence that are selected by each scheme appear in red at the top. Sequence positions 6-7 constitute a gap in the syncmer selection as they are not covered by any selected k-mer.

unlike minimizers, a PSS may not have a window guarantee since it will only identify a fixed subset of all k-mers.

Under these definitions, the open and closed syncmer schemes defined in [4] are Sy_1 and $Sy_{k,s}(1, k-s+1)$, respectively. From here on, we will refer to PSSs simply as syncmers.

Downsampled syncmers: Given a uniformly random hash function $h : \Sigma^k \to [0, H]$, for a given string S, *downsampling* selects syncmers only from the set of $|\Sigma|^k/\delta$ k-mers with the lowest hash values:

$$\mathcal{DS}_{k,s,o,\{x_1,...,x_n\},h,\delta}(S) = \{i | i \in Sy_{k,s,o}(x_1, ..., x_n)(S) \wedge h(S[i, i+k-1]) < H/\delta\}$$

We call δ the *downsampling rate*.

Windowed syncmers: A syncmer scheme may leave large gaps between selected positions on some input sequences. Windowed syncmers fill in these gaps using a minimizer scheme, thus providing a window guarantee. For clarity in the definition below let Sy represent $Sy_{k,s,o}(x_1, ..., x_n)(S)$.

$$f_{k,s,w,o,\{x_1,...,x_n\}}(S) = \left\{ i | i \in Sy \bigcup i \in \mathcal{M}_{k,w,o}(S[j, j+w-1]) \, \forall j \; s.t. \; Sy(S[j, j+w-1]) = \emptyset \right\}$$

Letting $\mathcal{S}$ represent all k-mers that can be syncmers in $Sy_{k,s,o}(x_1, ..., x_n)$, an equivalent definition would be: $\mathcal{M}_{k,w,o'}(S)$ where o' is defined such that $x \in \mathcal{S}, x' \in \Sigma^k \backslash \mathcal{S} \implies x < x'$.

2.3 Properties and evaluation criteria of schemes

We define some properties of selection schemes and several metrics that will allow us to compare different schemes.

Density and compression: The *density* [13] of a scheme is the expected fraction of positions selected by the scheme in an infinitely long random sequence: $d(f) = \mathbb{E}[|f(S)|/|S|]$ as $|S| \to \infty$. *Compression* is defined as $c(f) = 1/d(f)$, i.e. the factor by which the sequence S is "compressed" by representing it using only the set of selected k-mers.

Conservation: Conservation [16] is the expected fraction of positions covered by a selected k-mer in sequence S that will also be covered by the same selected k-mer in the mutated sequence S' where S' is generated by iid base mutations with rate θ. Define the set of positions covered by the same selected k-mer in both sequences

$$\begin{aligned} B(f, \theta, k) = \{ i \,|\, \exists j \in \{i-k+1, i-k+2, ..., i\} \\ s.t. \; j \in f(S) \cap f(S') \wedge S[j, j+k-1] = S'[j, j+k-1] \} \end{aligned}$$

then the *conservation* of the scheme is defined as $Cons(f, \theta, k) = \mathbb{E}\left[\frac{|B(f,\theta,k)|}{|S|}\right]$.

Spread and distance distribution: One key feature of a scheme is the distribution of distances between selected positions. This would tell us the frequency with which selected

positions appear close together or far apart. Shaw and Yu studied the probability distribution of selecting *at least one* position in a window of length α. We refer to the vector $P(f, \alpha)$ of these probabilities as the *spread*.

We define the distribution of the distances between *consecutive* selected positions: $Pr(f) = [Pr(f, 1), Pr(f, 2), ...]$, where $Pr(f, n)$ is the probability that position $i + n$ is the next selected position given that position i is selected. We refer to this as the *distance distribution* of the scheme.

pN **metric:** The pN metric ($N \in [0, 100]$) is the Nth percentile of the distance distribution, i.e., it is the length l for which $N\%$ of the distances between consecutive selected positions are of length $\leq l$.

ℓ **and** ℓ_2 **metrics:** Let the lengths of the uncovered gaps between k-mers selected by a scheme on a sequence S be $l_1, l_2, ...$ We define $\ell = \frac{1}{|S|} \sum_i l_i$ and $\ell_2 = \sqrt{\frac{1}{|S|} \sum_i l_i^2}$.

Many of these properties can be determined theoretically in expectation for given sequence and mutation models and the selection scheme. They can also be determined empirically for a specific sequence. Some metrics may also be applied either to all the positions selected by a scheme in a reference, or only to the selected positions that are *conserved* after mutation or sequencing error. We refer to the latter using the subscript mut, for example, $\ell_{2,mut}$.

2.4 Analysis of syncmer schemes – prior work

Edgar recently introduced syncmers as an alternative to minimizers and other selection schemes with the goal of improving conservation rather than density, arguing that the latter is dictated by the application and system constraints [4]. He introduced open and closed syncmers. Rotated variants of syncmers, in which the minimizer is allowed to circle around from the end of the k-mer to the beginning were also introduced, but we found they were not useful in practice and do not address them here. Analyses of syncmer densities, window guarantees, and distributions were provided in [4] for open, closed, and downsampled syncmers.

Shaw and Yu greatly extended the framework for theoretical analysis of syncmers [16]. They defined the spread and conservation of a scheme. The two are connected through the number of unmutated k-mers overlapping a given position, $\alpha(\theta, k)$, for a given mutation rate, θ. Letting $P(f) = [P(f, 1), P(f, 2), ...P(f, k)]$ be the spread, and $P(\alpha(\theta, k)) = [P(\alpha(\theta, k) = 1), P(\alpha(\theta, k) = 2), ..., P(\alpha(\theta, k) = k)]$, then $Cons(f, \theta, k) = P(f) \cdot P(\alpha(\theta, k))$. Note that there is a closed form expression for calculating $P(\alpha(\theta, k) = \alpha))$, and that $P(f, 1) = d(f)$. Their theoretical framework allowed Shaw and Yu to obtain expressions for the spread (and therefore conservation) of open and closed syncmers and other selection schemes.

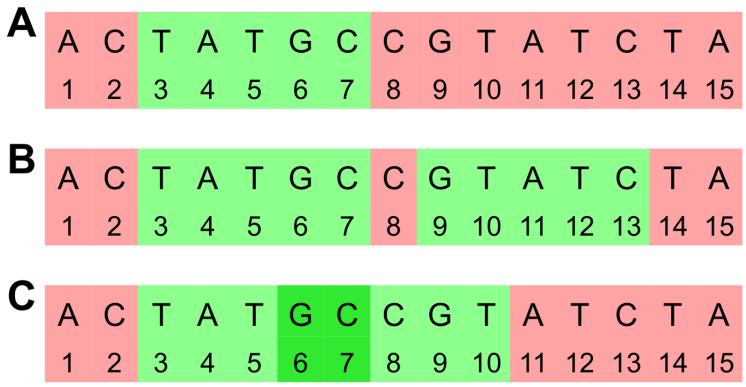

Figure 2: **Illustration of s-minimizers generating syncmers.** A window of $\alpha = 5$ consecutive 11-mers. **A**: When $s = 5$ and $t = 3$, then the s-minimizer of the entire window generates a syncmer when its starting index is in the green region. If the s-minimizer is in one of the red regions then a syncmer may be generated by the s-minimizer of the remaining part of the window. For a two parameter scheme the s-minimizer creates two syncmer generating regions that may be disjoint (**B**) if $s > t_2 - t_1$ or overlapping (**C**) if $s < t_2 - t_1$. In this example, $t_1 = 3$ and $t_2 = 9$ in **B** and $t_2 = 6$ in **C**.

3 Analysis of parameterized syncmer schemes

3.1 Recursive expressions for conservation of PSSs

We extend the analysis of [16] to obtain recursive definitions of the spread for general 2-parameter schemes where the s-minimizer indices can take on any values. We show how to incorporate downsampling and extend the definitions further to the conservation of schemes with more parameters. We used these expressions to compute the expected conservation for all PSSs of k-mer lengths 11, 13, 15, 17, and 19 and mutation rates 0, 0.05, 0.15 and 0.25. These results are provided in Supplementary File 1, Table 1.

Consider a window of α consecutive k-mers. We assume randomly distributed sequence throughout. Let s_β be the s-minimizer in the window, at position β. Then if t is a parameter of the syncmer scheme, s_β generates a syncmer if it is not in the first $t-1$ or last $k-t$ positions in the window. If β is not in a position where it generates a syncmer, we recursively check to the left or right of β to see if a syncmer is generated by the s-minimizer of that region. See Figure 2 for an example.

For a 1-parameter scheme f with k-mer length k, s-minimizer length s, and parameter t let $P(\alpha)$ be the probability of selecting at least one syncmer in a window of α adjacent k-mers. Then, assuming a uniformly random hash over the s-mers, and conditioning on

the position of the s-minimizer of the α-window, β:

$$P(\alpha) = p_\beta \sum_\beta P(\alpha|\beta) = \frac{1}{k+\alpha-s}\left[\sum_{\beta=1}^{t-1} P(\alpha-\beta) + \sum_{\beta=t}^{t+\alpha-1} 1 + \sum_{t+\alpha}^{k+\alpha-s} P(\beta-k+s-1)\right].$$

The probability of any of the $k+\alpha-s$ starting positions being the s-minimizer is denoted as p_β and assumed to be uniform. If β is in the first $t-1$ or last $k-t$ starting positions (red regions in Figure 2A), then a syncmer may be generated by the remaining $\alpha-\beta$ positions to the right or $\beta-k+s-1$ positions to the left, respectively. Note we define $\sum_{x=i}^{j} = 0$ when $i > j$ and $P(x) = 0$ when $x \leq 0$.

When downsampling syncmers, there is a probability of $1/\delta$ that an s-minimizer in the syncmer generating region (i.e. the green region in Figure 2A) really generates a syncmer. If it does not, then the left and right regions are considered recursively, yielding the following expression, where we simplify notation by letting $P(\alpha-\beta) = P_R$ and $P(\beta-k+s-1) = P_L$:

$$P(\alpha) = \frac{1}{k+\alpha-s}\left[\sum_{\beta=1}^{t-1} P_R + \sum_{\beta=t}^{t+\alpha-1}\left(\frac{1}{\delta} + (1-\frac{1}{\delta})\left(P_R + P_L - P_R \cdot P_L\right)\right) + \sum_{t+\alpha}^{k+\alpha-s} P_L\right].$$

In the case of 2-parameter schemes, two syncmers are generated by s_β in regions that will overlap if the parameters t_1 and t_2 are within s of each other, and will be disjoint otherwise (see Figure 2B,C). Combining these two cases into a single recursive expression yields:

$$P(\alpha) = \frac{1}{k+\alpha-s}\cdot\left[\sum_{\beta=1}^{t_1-1} P_R + \sum_{\beta=t_1}^{\min\limits_{(t_2-1,t_1+\alpha-1)}} 1 + \sum_{\substack{\beta=min\\(t_2,t_1+\alpha)}}^{t_1+\alpha-1} 1 + \sum_{\substack{\beta=min\\(t_2,t_1+\alpha)}}^{t_2-1} \left(P_R+P_L\right) + \sum_{\substack{\beta=max\\(t_1+\alpha,t_2)}}^{t_2+\alpha-1} 1 + \sum_{t+\alpha}^{k+\alpha-s} P_L\right].$$

When downsampling is used then the 1 in the second and fifth sums is replaced by $\frac{1}{\delta} + (1-\frac{1}{\delta})\left(P_R + P_L - P_R \cdot P_L\right)$ as in the one parameter case. The third sum expresses the overlapped region where either parameter creates a syncmer, when it exists. When *both* generated syncmers are downsampled then the left and right sides are recursively checked, thus the 1 is replaced by $(1-(1-\frac{1}{\delta})^2 + (1-\frac{1}{\delta})^2\left(P_R + P_L - P_R \cdot P_L\right)$.

This expression can be greatly simplified by introducing the notation $count(\beta)$ that represents the number of syncmers generated by the s-minimizer s_β. For example, $count(\beta) = 0$ in the red region of Figure 2 and $count(\beta) = 2$ in the overlapped region when $\beta = 6$ or 7 in Figure 2C. The general expression for $P(\alpha)$ for *any* PSS is:

$$P(\alpha) = \frac{1}{k+\alpha-s}\cdot\left[\sum_{\beta=1}^{k+\alpha-s}\left(1-(1-\frac{1}{\delta})^{count(\beta)}\right) + (1-\frac{1}{\delta})^{count(\beta)}\left(P_R + P_L - P_R \cdot P_L\right)\right].$$

Note that this definition relies on the definition $P(x) = 0, x \leq 0$ to include the correct terms for the correct values of β.

Figure 3: ℓ **vs.** ℓ_2 **metric.** The selected positions of three different selection schemes S_1, S_2 and S_3 on the same sequence. Selected k-mers are highlighted and underlined. All schemes have the same number of selected k-mers, but the metrics are different. $\mathbf{S_1}$: $\ell = 0.529$, $\ell_2 = 2.974$. $\mathbf{S_2}$: $\ell = 0.529$, $\ell_2 = 1.81$. $\mathbf{S_3}$: $\ell = 0.647$, $\ell_2 = 2.808$. While S_1 and S_2 have the same ℓ value, the k-mers selected by S_2 are more evenly spread and thus it has much lower ℓ_2. Some of the k-mers selected by S_3 overlap, resulting in a higher ℓ value than the other schemes. However, because the gaps between covered bases are more evenly spread, the ℓ_2 value is lower than that of S_1. Intuitively, it will be easier to map reads using seeds selected by S_3 than S_1 despite the lower ℓ value, suggesting that ℓ_2 is a more appropriate metric.

The value of $P(\alpha)$ can thus be computed efficiently for any PSS and used to calculate the conservation using the formula from [16].

3.2 Choosing an appropriate metric to compare schemes

While Edgar shows convincingly that conservation is a more appropriate metric for selection schemes than density, we argue that $\ell_{2,mut}$ contains additional important information for the purpose of mapping. Specifically, observe that, for given mutation rate θ, k, and selection scheme f, we have $\mathbb{E}\left[\ell_{mut,\theta,f,k}\right] = 1 - Cons(\theta, f, k)$. While ℓ (and conservation) counts the number of bases that are not covered by conserved selected k-mers, it treats all gap lengths equally. In contrast, $\ell_{2,mut}$ penalizes a few large gaps more than many smaller gaps with the same total length. See the example in Figure 3. When the selected k-mers are used as seeds for mapping, it is important to avoid large gaps, in order to enable read mapping across gaps. Thus ℓ_2 provides additional salient indication to ℓ on how the selection scheme will affect mapping performance.

3.3 Calculating the distance distribution

For a given scheme, the distribution of distances between adjacent syncmer positions is specified by $Pr(d = x)$, the probability that the distance d is x. To calculate this probability, we define the new quantities $F(\alpha)$ and $L(\alpha)$ denoting the probability that *only* the first or *only* the last k-mer in a window of α k-mers is a syncmer, respectively. We refer to these k-mers as K_1 and K_α respectively. Note that due to symmetry $F(\alpha) = L(\alpha)$. Note

also that $1 - P(\alpha)$ gives the probability that *no* k-mer in an α-window is a syncmer.

We compute $F(\alpha)$ by conditioning on β as before. For simplicity we divide the sum over β into cases based on the syncmers that are generated by s_β rather than breaking up the sum across different values of β. With some abuse of notation, we let K_i represent the event the that s_β generates K_i as a syncmer.

$$F(\alpha) = \frac{1}{k+\alpha-s} \sum_{\beta=1}^{k+\alpha-s} \begin{cases} \frac{1}{\delta}.(1-\frac{1}{\delta})^{count(\beta)-1} \cdot (1 - P(\alpha-\beta)) & K_1 \\ (1-\frac{1}{\delta})^{count(\beta)} \cdot F(\beta-k+s-1) \cdot (1-P(\alpha-\beta)) & \text{otherwise} \end{cases}$$

In the first case we have the probability that K_1 is not downsampled, any other syncmer generated by s_β is downsampled, and there are no other syncmers generated to the right of β. In the second case we have the probability that any syncmers generated by s_β are downsampled, no syncmers are generated to the right of β, and the recursive computation of the probability that the s-minimizer of the segment to the left of β generates a syncmer at K_1.

Similarly, define $D(\alpha)$ to be the probability that in a window of α k-mers *only* the first *and* last k-mers are syncmers. Then

$$D(\alpha) = \frac{1}{k+\alpha-s} \sum_{\beta=1}^{k+\alpha-s} \begin{cases} (\frac{1}{\delta})^2 \cdot (1-\frac{1}{\delta})^{count(\beta)-2} & K_1, K_\alpha \\ \frac{1}{\delta} \cdot (1-\frac{1}{\delta})^{count(\beta)-1} \cdot F(\alpha-\beta) & K_1, \neg K_\alpha \\ \frac{1}{\delta} \cdot (1-\frac{1}{\delta})^{count(\beta)-1} \cdot F(\beta-k+s-1) & K_\alpha, \neg K_1 \\ (1-\frac{1}{\delta})^{count(\beta)} \cdot F(\beta-k+s-1) \cdot F(\alpha-\beta) & \text{otherwise} \end{cases}$$

3.4 Calculating $\ell_{2,mut}$

To compute the desired metric $\ell_{2,mut}$ we must calculate the metrics from the previous section but only with *conserved* syncmers. We add the subscript '*mut*' to a value to indicate that only conserved syncmers are considered. The impact of mutations is similar to that of downsampling shown in the previous section, except that when a syncmer is lost due to mutation, the surrounding k-mers are also lost. In this case we consider no downsampling to make the expressions simpler.

Let Ω_β be the set of syncmers generated by s_β, and ω_{β_i} be the members of this set. Note that $|\Omega_\beta| = count(\beta)$. Then,

$$P_{mut}(\alpha) = \frac{1}{k+\alpha-s} \sum_{\beta=1}^{k+\alpha-s} \Big(Pr(\exists \text{ conserved } \omega_{\beta_i} \in \Omega_\beta) + Pr(\nexists \text{ conserved } \omega_{\beta_i} \in \Omega_\beta, \exists \text{ syncmer to the left or right}) \Big)$$

For convenience we call the first probability $P_{conserved}$ and the second $P_{recurse}$.

$P_{conserved}$ is computed using the inclusion-exclusion principle:

$$P_{conserved} = \sum_i Pr(\omega_i \text{ conserved}) - \sum_{i<j} Pr(\omega_i, \omega_j \text{ conserved}) + \sum_{i<j<k} Pr(\omega_i, \omega_j, \omega_k \text{ conserved}) - ...$$

Every term in this series is calculated as $(1-\theta)^{countBases}$ where θ is the mutation rate and $countBases$ counts the number of bases covered by the conserved k-mers (i.e. if two conserved syncmers overlap, the shared bases are counted only once).

$P_{recurse}$ is more complicated. We again sum over all values of β. When Ω_β is empty (e.g. β is in the red region), then the recursion is similar to the case without mutation. Otherwise, all of the syncmers are lost due to mutation, and we additionally sum over the possible positions of the first and last points of mutation in Ω_β, named f and l, respectively.

$$\sum_{\beta=1}^{k+\alpha-s} \frac{1}{k+\alpha-s} \begin{cases} P_{mut}(\alpha-\beta) + P_{mut}(\beta-k+s-1) - P_{mut}(\alpha-\beta)\cdot P_{mut}(\beta-k+s-1) & \Omega_\beta = \emptyset \\ \sum_{f\le l} Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta, f, l) \times & \\ \quad (P_{mut}(left) + P_{mut}(right) - P_{mut}(left)\cdot P_{mut}(right)) & \text{otherwise} \end{cases}$$

Here $left = max(0, min(f-k, \beta-k+s-1))$ and $right = max(0, min(\alpha-l, \alpha-\beta))$. We expand the joint probability as

$$Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta, f, l) = \theta^y \cdot (1-\theta)^x \cdot Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta | f, l)$$

where y is 1 if $l = f$ and 2 otherwise, and x is the number of unmutated bases that is fixed by the given values of f and l. The conditional probability is written as

$$Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta | f, l) = 1 - Pr(\exists \text{ conserved } \omega_\beta \in \Omega_\beta | f, l)$$

and is computed using the inclusion-exclusion principle as above.

$F(\alpha)$ and $D(\alpha)$ are extended to the case of mutation similarly:

$$F_{mut}(\alpha) = \frac{1}{k+\alpha-s} \times$$
$$\sum_{\beta=1}^{k+\alpha-s} \begin{cases} \sum_{f\le l} Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta \setminus K_1, K_1 \text{ conserved}, f, l)\cdot(1-P_{mut}(right|b)) & K_1, \Omega_\beta \setminus K_1 \neq \emptyset \\ Pr(K_1 \text{ conserved})\cdot(1-P_{mut}(\alpha-\beta|b)) & K_1, \Omega_\beta \setminus K_1 = \emptyset \\ \sum_{f\le l} Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta, f, l)\cdot(1-P_{mut}(right|b))\cdot F_{mut}(left|b') & \neg K_1, \Omega_\beta \neq \emptyset \\ F_{mut}(\beta-k+s-1)\cdot(1-P_{mut}(\alpha-\beta)) & \text{otherwise} \end{cases}$$

$$D_{mut}(\alpha) = \frac{1}{k+\alpha-s}\times$$
$$\sum_{\beta=1}^{k+\alpha-s}\begin{cases}\sum_{f\le l} Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta \setminus K_1, K_1 \text{ conserved}, f, l)\cdot F_{mut}(right|b) & K_1, \neg K_\alpha, \Omega_\beta \setminus K_1 \neq \emptyset \\ (1-\theta)^k \cdot F_{mut}(\alpha-\beta|b) & K_1, \neg K_\alpha, \Omega_\beta \setminus K_1 = \emptyset \\ \sum_{f\le l} Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta \setminus K_\alpha, K_\alpha \text{ conserved}, f, l)\cdot F_{mut}(left|b) & K_\alpha, \neg K_1, \Omega_\beta \setminus K_\alpha \neq \emptyset \\ (1-\theta)^k \cdot F_{mut}(\beta-k+s-1|b) & K_\alpha, \neg K_1, \Omega_\beta \setminus K_\alpha = \emptyset \\ \sum_{f\le l} Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta \setminus \{K_1\cup K_\alpha\}, \{K_1\cup K_\alpha\} \text{ conserved}, f, l) & K_1, K_\alpha, \Omega_\beta \setminus \{K_1\cup K_\alpha\} \neq \emptyset \\ Pr(\{K_1\cup K_\alpha\} \text{ conserved}) & K_1, K_\alpha, \Omega_\beta \setminus \{K_1\cup K_\alpha\} = \emptyset \\ \sum_{f\le l} Pr(\nexists \text{ conserved } \omega_\beta \in \Omega_\beta, f, l)\cdot F_{mut}(right)\cdot F_{mut}(left|b) & \neg K_1, \neg K_\alpha, \Omega_\beta \neq \emptyset \\ F_{mut}(\alpha-\beta)\cdot F_{mut}(\beta-k+s-1) & \neg K_1, \neg K_\alpha, \Omega_\beta = \emptyset\end{cases}$$

Here we again divide into cases depending on whether there are syncmers that can be lost. We have also used recursive expressions that are similar to the above except we are given that b bases to the left or right of the defined region are conserved. These are calculated using similar techniques as above.

Finally, we can use these expressions to compute:

$$\ell_{mut} = \sum_{x=k+1}^{\infty}(x-k)\cdot D_{mut}(x+1)$$

$$\ell^2_{mut} = \sqrt{\sum_{x=k+1}^{\infty}(x-k)^2\cdot D_{mut}(x+1)}$$

Note that, unlike $P(\alpha)$, which can be computed efficiently, the computation of these metrics includes an infinite sum. The sum can be truncated at an appropriate distance x, however there are still many more terms than in the computation of the conservation. In practice, simulating a very long sequence, selecting syncmers, and simulating mutations to determine these metrics empirically is much less time consuming and yields results that are very close to the true values. We used this simulation method to compute $\ell_{2,mut}$ for $k = 11, 13, 15, 17$ and 19, mutation rates $0.05, 0.15$ and 0.25. and all 2- and 3-parameter schemes, presented in Supplementary File 1, Table 2 (note that for 1-parameter schemes the best ℓ_2 and ℓ are the same, and thus already known from [16]).

3.5 Achieving the target compression

A simple extension of the expression for compression of open and closed syncmers yields that the compression of an n-parameter PSS is $\approx \frac{k-s+1}{n}$, where we assume that s is long enough relative to k that the s-minimizer is likely to be unique. As we show in Supplementary File 1, Table 4, it is preferable to achieve a specific compression with minimal

downsampling. For example, the ℓ_2 of the best 2-parameter scheme with a downsampling rate of 2 is an order of magnitude worse than that of the best 1-parameter scheme without downsampling. Thus, to choose the best PSS for a given target compression, we can choose the one with compression closest to, but below, the desired compression and then downsample to reach the desired compression.

4 Methods

We modified the code of minimap2 (v2.22-r1105-dirty) and Winnowmap2 (v2.03) to select our syncmer variants as seeds instead of minimizers. The code is available from `https://github.com/Shamir-Lab/syncmer_mapping`.

4.1 Syncmer schemes implementation

The implementation of the syncmer schemes defined in Section 2 is straightforward. Sequences are scanned from left to right, the canonical k-mer at each position is identified using a random hash h_1, and the index of the minimum s-mer under another random hash h_2 is determined.

For downsampled schemes, syncmers are selected if their hash value normalized between 0 and 1 is below the downsampling rate. Note that a different hash function than h_1 must be used to ensure random downsampling. Windowed schemes are integrated into the minimizer selection scheme of the mappers except that syncmers are selected in each window first. If no syncmer is present, then the minimizer is selected.

Pseudocode describing these implementations is presented in Algorithms 1 and 2.

Algorithm 1 Syncmer selection (regular, downsampled)

Input: Sequence S, syncmer parameters x_1,x_2, ...x_n, k-mer length k, s-mer length s, downsampling rate δ (default: 1)

Output: P, a list of selected positions.

$P \leftarrow \{\}$
for $j \in 1$ to $|S| - k + 1$ **do**
    $m = \underset{t \in [0, k-s]}{\operatorname{argmin}} h_1(Canonical(S[j + t, j + t + s]))$
    if $m \in \{x_1, x_2..., x_n\}$ **and** $h_2(S[j, j + k - 1]) < 1/\delta$ **then**
        $P \leftarrow P \cup \{j\}$
return P

Additional implementation and optimization details are presented in Supplementary section S1

Algorithm 2 Windowed syncmer selection

Input: Sequence S, syncmer parameters x_1,x_2,...x_n, k-mer length k, s-mer length s, window w, downsampling rate δ (default: 1)

Output: P, a list of selected positions.

$P \leftarrow \{\}$
for $j \in 1$ to $|S| - w + 1$ **do**
  $hasSync \leftarrow$ FALSE
  for $l \in 0$ to $w - k$ **do**
    $m = \underset{t \in [0, k-s]}{\operatorname{argmin}} h_1(Canonical(S[j + l + t, j + l + t + s]))$
    if $m \in \{x_1, x_2..., x_n\}$ **and** $h_2(S[j + l, j + l + k - 1]) < 1/\delta$ **then**
      $P \leftarrow P \cup \{j + l\}$
      $hasSync \leftarrow$ TRUE
  if $hasSync$ **is** FALSE **then**
    $m = \underset{l \in [0, w-k]}{\operatorname{argmin}} h_3(Canonical(S[j + l, j + l + k - 1]))$
    $P \leftarrow P \cup \{j + m\}$
return P

5 Results

We evaluated different PSSs on real genomes and compared them to theoretical results from Section 3. We also evaluated PSS-based mapping compared to the original minimizer-based versions of minimap2 and Winnowmap2 on simulated and real read data.

The sequences used for these experiments were: human GRCh38.p13 [14], human chromosome X from CHM13 (v1.0), *E. coli* K12 [1], and a microbial sample BAC containing assemblies of 15 microbes for which PacBio long read data is available [12] (three of the microbes were used in [16], see Supplementary Section S3 for more details on the samples selected). Information about the sequences is presented in Table 1 .

We simulated PacBio and ONT reads from Chromosome X and from BAC with a depth of 10. Details of simulation parameters are found in the Supplement Section S2. For real datasets we selected a random set of 10K ONT reads of the NA12878 cell line with read length capped at 10kb (SRA accession ERR3279003), and 1K PacBio reads for each of the BAC microbes [12]. Details are available in Table 2.

5.1 Properties of parameterized syncmer schemes

Our theoretical analysis of PSS properties (Section 3) relies on a number of assumptions. Specifically, it assumes uniform iid sequences and mutations, assumes only substitution mutations, and treats the sequence as a single forward strand. We therefore examined the

Dataset name	Species	Source	# scaffolds	Total length (nt)
GRCh	Human	GRCh38 [14]	639	3.111G
CHM13X	Human	CHM13 X chromosome [10]	1	154.3M
BAC	Microbial (see S3)	PacBio [12]	24	59.1M
ECK12	*E. coli K-12*	GCF_000005845.2 [1]	1	4.6M

Table 1: **Genome information.** Basic information about the reference genomes used in our experiments. # scaffolds is the number of individual sequences present in the reference genome fasta file and can include unplaced scaffolds, alternates, etc. Total length counts the length of all of the scaffolds together, excluding ambiguous bases.

Dataset name	Read type	Source	# reads	Mean length (std)
pbsim_x	PacBio simulated	CHM13X	173891	8871.1 (5570.1)
pbsim_bac	PacBio simulated	BAC	66428	8894.2 (5617.4)
ns_chm13	ONT simulated	CHM13	1000	8722.8 (7030.7)
pb_bac	PacBio	BAC	15000	9488.3 (5207.2)
ont_na12878	ONT	ERR3279003	10000	7131.6 (2348.5)

Table 2: **Reads information.** The long read datasets used in our experiments. Some source names are from Table 1.

Dataset	Scheme	Compression	ℓ	ℓ_2	$p90$	$p100$	# conserved
ECK12	$\mathcal{M}_{15,10}$	76.929	0.859	13.758	211	1092	60,337
	$Sy_{15,5}(3,9)$	63.085	0.845	12.99	183	1078	73,577
	$\mathcal{M}_{15,19}$	154.13	0.9139	17.845	378	1981	30,115
	$Sy_{15,6}(6)$	116.038	0.896	16.175	303	1542	40,001
CHM13X	$\mathcal{M}_{15,10}$	54.339	0.8113	11.695	153	1202	2,838,860
	$Sy_{15,5}(3,9)$	45.647	0.796	11.132	133	1293	3,379,427
	$\mathcal{M}_{15,19}$	107.907	0.88	15.119	270	1946	1,429,555
	$Sy_{15,6}(6)$	83.199	0.859	13.83	219	1927	1,854,097

Table 3: **Properties of conserved k-mers in minimizer and syncmer schemes under mutation.** Substitutions were introduced in the references at a rate of 15%. The values shown are for the conserved selected k-mers.

properties of PSSs on real genomes where these assumptions do not necessarily apply, and compared them to minimizer schemes.

We used $k = 15$ and selected the best syncmer schemes (based on $\ell_{2,mut}$) with theoretical compression 5.5 and 10. The default minimizer scheme of minimap2 uses $k = 15, w = 10$ yielding the theoretical compression of 5.5. A theoretical compression of 10 is achieved by minimap2 with $w = 19$. The properties of these schemes on the ECK12 and CHM13X sequences without mutation are shown in Supplementary section S4, Table S1. For *unmutated* reference genomes, minimizers outperformed syncmers, with much lower ℓ_2 and $p100$ values for schemes with the same compression.

To test the schemes on *mutated* sequences, we selected k-mers using the different schemes, simulated iid substitutions to the CHM13X sequence at a rate of 15%, and computed the properties of the conserved k-mers selected by the schemes. The performance is summarized in Table 3. Under mutation the advantage of syncmers is clear: syncmers have 19-33% more conserved positions and better performance in all metrics.

The windowed and downsampled variants are shown in Supplementary Tables S3 and S2. As expected, shorter window lengths require more selected positions to be filled by minimizers and have markedly lower ℓ_2 than the unwindowed versions.With mutations, windowed syncmers with short window lengths do even better than the unwindowed PSSs, even with relatively few conserved minimizer positions (see Table S3).

Figure 4 shows the distribution of distances between selected positions for $Sy_{15,5}(3,9)$ on CHM13X. Figure 4A shows the distance distribution of syncmers selected only using forward strand k-mers. It matches the theoretical distribution from Section 3 closely, with a minimum distance of 3 and a sharp peak at 6. In mapping, the read orientation is

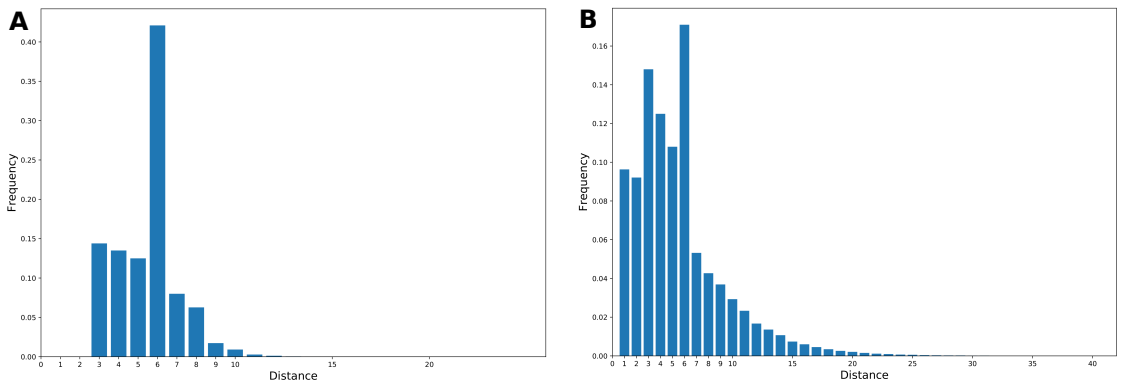

Figure 4: **Distribution of distances in syncmer scheme.** The distribution of distances between consecutive selected positions of syncmer scheme $Sy_{15,5}(3,9)$ on the CHM13X reference is shown. **A.** The distribution of syncmers selected only in the forward orientation. **B.** Canonical syncmers. For visualization purposes the distribution is shown only for distances with frequency $> 10^{-6}$. The true maximum distance is 161 for canonical k-mers (see Supplementary Table S1) and 76 for the forward k-mers, but the frequency of the longer distances is extremely low.

unknown and canonical syncmers are used. Figure 4B shows the results using canonical syncmers. The distance distribution still retains the peak at 6 and a local maximum at 3, but now adjacent positions are selected, and it has a much longer tail of distances, as reflected also in the $p100$ values shown in Supplementary Table S1.

5.2 The fraction of unmapped reads

We mapped reads using minimap2 and Winnowmap2 with $\mathcal{M}_{15,10}$ (low compression), $\mathcal{M}_{15,50}$ (medium), and $\mathcal{M}_{15,100}$ (high) on 4 datasets. For each dataset, the syncmer-minimap and syncmer-winnowmap parameters were selected to have the best performance based on theoretical $\ell_{2,mut}$ for the same compression achieved by minimap2. In all cases this resulted in $Sy_{15,5}(3,9)$ matching the low compression, and $Sy_{15,4}(6)$ matching the medium and high compression. The other scheme parameters were manually selected to closely match the real compression. The exact compression, window length, and downsampling rates are given in the table in Supplementary File 1, Table 5.

Figure 5 shows the percentage of unmapped reads achieved by each of the mappers for simulated PacBio and ONT reads mapped to the human reference genome. See Supplementary Figure S7 for additional results, including windowed mappers. Syncmer variants performed essentially the same or better than the original mappers in all cases, with a marked advantage at high compression. All mappers did much better on the PacBio reads than on ONT reads, which have a higher proportion of deletions and substitutions. The

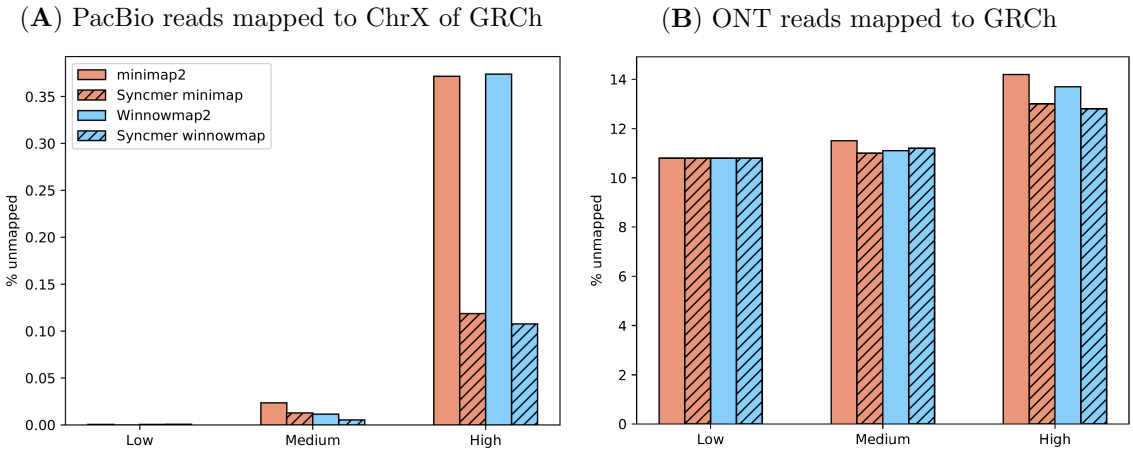

Figure 5: **Percentage of unmapped reads – simulated datasets.** The percentage of unmapped reads is plotted for two simulated read datasets mapped to their reference sequences. Results are shown for low, medium and high compression. **(A)** PacBio reads simulated from the CHM13X sequence mapped against ChrX sequences from GRCh38. **(B)** 1000 ONT reads simulated from CHM13 mapped against GRCh38.

jump in the fraction of unmapped reads between medium and high compression may indicate that in order to overcome the large fraction of non-conserved seeds, existing mappers need to use a lower compression with many redundant seeds.

We compared the performance of all mappers on real data (Table 2) across a range of compression values. The ONT reads were mapped against the reference GRCh and the PacBio bacterial reads were mapped against the BAC reference. See Figure 6. The syncmer variants consistently outperformed the original minimizer-based mappers, with syncmer-winnowmap performing the best across the larger part of the range. Full results and scheme parameters are given in Supplementary File 1, Table 6. For high compression, the minimizers had 20-40% more unmapped reads than the syncmers. At low compression rates of $5.5 - 11$, minimizers had 2-15% more unmapped reads than syncmers.

5.3 Mapping correctness

We evaluated the mapping correctness for PacBio simulated reads as done in [6] (see Supplementary section S2 for details). The percentage of incorrectly mapped reads simulated from CHM13X and the BAC genomes are shown in Figure 7. Winnowmap consistently performed better than minimap, and the syncmer variants of Winnowmap performed best overall.

Although we cannot evaluate the mapping correctness on the real datasets, the mapping quality scores can be used to compare the different mappers. On the four real datasets, reads mapped by syncmer-minimap but not by minimap2 generally had higher mapping quality than those mapped by minimap2 and not syncmer-minimap. For example, for the

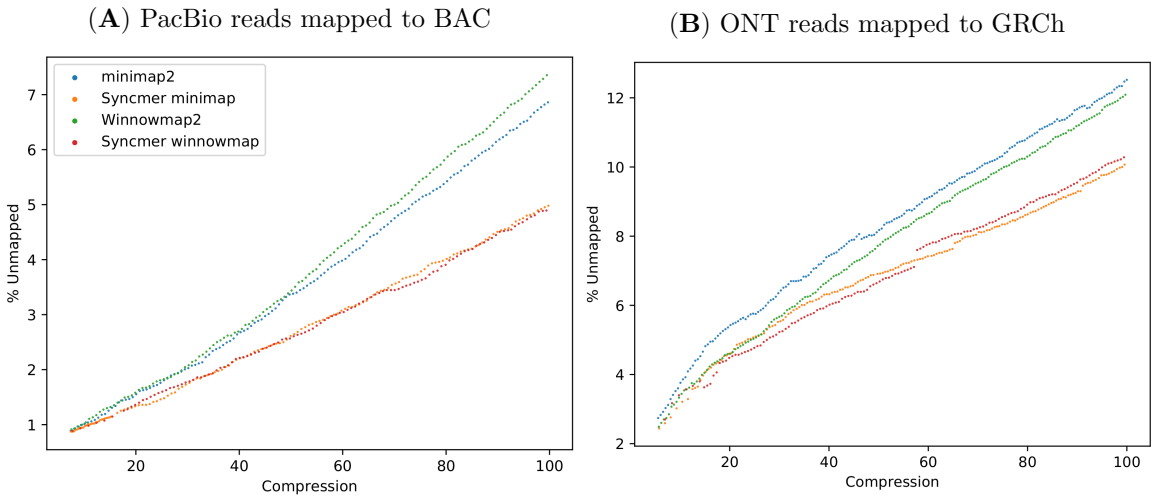

Figure 6: **Percentage of unmapped reads – real datasets.** Results are shown across a broad range of compression rates. Syncmer parameters were chosen to achieve the desired compression with lowest $\ell_{2,mut}$. **(A)** Pooled PacBio bacterial reads. **(B)** ONT human cell-line reads.

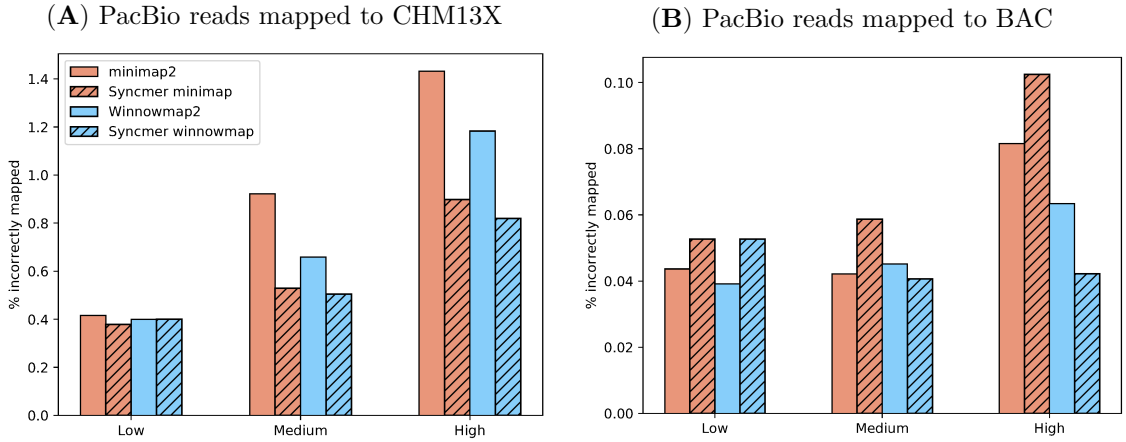

Figure 7: **Percentage of incorrectly mapped reads –simulated data.** The percentage of incorrectly mapped reads is plotted for two simulated read datasets and their reference sequences, for mappers using low, medium, and high compression. **(A)** PacBio reads simulated from the CHM13 ChrX sequence mapped against CHM13X. **(B)** PacBio reads simulated from the 15 bacterial species in BAC pooled together and mapped against the union of their references.

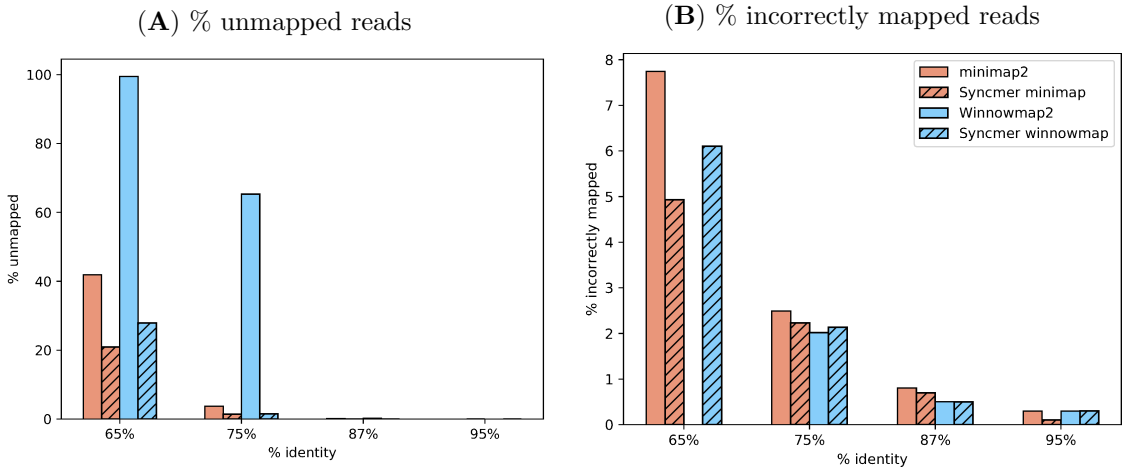

Figure 8: **Impact of percent sequence identity on mapping quality.** We varied the mutation rate of 1000 PacBio simulated reads from CHM13X. The figures present the % unmapped and incorrectly mapped for each of the tools.**(A)** % unmapped reads. **(B)** % of the mapped reads that were incorrectly mapped.

human cell line ONT reads, comparing minimap2 with $\mathcal{M}_{15,50}$ to corresponding syncmer-minimap, 39 minimap-only reads had average mapping quality 31.4 (median 27), while 94 syncmer-minimap-only reads had an average quality score of 38.7 (median 42.5). Full results for different compression rates are presented in Supplementary File 1, Table 7.

5.4 Impact of sequence identity level

We examined the impact of the level of identity between the sequenced reads and the reference to which they are aligned. Differences between the sequences can be due to high sequencing errors rate in long reads, mutations in the sequenced organism, or differences between sequenced and reference strains. We simulated 1000 PacBio reads from CHM13X at percent identity 65%, 75%, and 95% in addition to the default 87% used above. The results are shown in Figure 8 and S3. For minimap2 and Winnowmap2 we used $\mathcal{M}_{15,50}$, and in the syncmer variants we used $Sy_{15,4}(6)$ with the other parameters selected as above to match the compression of minimap2.

The syncmer variants outperformed the original tools in terms of fraction of reads mapped, with larger gains as percent identity decreases. This highlights the impact of the increased conservation of syncmers. All tools performed very well at higher percent identity, indicating that more than enough seeds were selected and conserved to adequately map all reads (and thus perhaps compression could be increased). Winnowmap2 performed noticeably worse at lower percent identity, leaving almost all reads unmapped at 65% identity. Syncmer-minimap outperformed minimap2 on the fraction of correctly mapped reads

in all cases. Winnowmap2 correctly mapped a larger fraction of the mapped reads at 75% identity, but mapped only 35% of the reads, compared to $\geq 95\%$ for the other variants. At 95% identity the syncmer variants had fewer incorrectly mapped reads.

5.5 Performance of windowed syncmer schemes

Windowed schemes combine syncmers and minimizers, allowing for a syncmer scheme to have a window guarantee with a relatively short window. In practice the windowed variants of our syncmer mappers were very similar or slightly worse than the variants without windowing for the same compression. Supplementary section S5 presents all of the results on the windowed variants of the mappers.

5.6 Runtime and memory

We compared the runtime and memory usage of the six tested mappers on the PacBio and ONT simulated reads from bacteria and human. Table 4 shows the performance for three different tasks. The third task maps bacterial reads against the human genome, to show the effect on timing of many unmapped reads. All experiments were performed on a 44-core, 2.2 GHz server with 792 GB of RAM, using 50 threads. Peak RSS (in GB) and real time (in seconds) as measured by the tools are reported. For minimap2 and Winnowmap2 $\mathcal{M}_{15,10}$ were used, for syncmer variants $Sy_{15,5}(3,9)$ was used with the same parameters matched to the minimizers as above.

In all cases, syncmer-winnowmap used the least RAM, at the cost of higher runtimes, and minimap2 achieved the fastest mapping, at the cost of higher RAM usage. Syncmer-minimap achieved a balance with the second fastest runtime and second or third lowest memory usage among the six tools.

We also looked at the runtimes and memory usage of all runs of the continuous compression experiment shown in Figure 6. Results are shown in Figures 9 and 10 (see also Supplementary Figures S5 and S6 for the windowed variants). minimap2 was consistently the fastest, followed by syncmer-minimap, which took 50-100% longer. Interestingly, the two datasets show exactly opposite trends in memory usage (Figure 10). This is because the bacterial reference genomes are relatively short, and thus the memory bottleneck is in the mapping stage, while for the human reference genome the memory bottleneck is in the indexing stage. Increasing compression lowers index size but results in longer alignments between anchors, requiring more memory in the mapping phase. Thus, when indexing is the bottleneck, increasing compression reduces memory, while when mapping is the bottleneck it increases memory. Winnowmap2 and its variants used less memory in the mapping phase while minimap2 and its variants used less memory in the indexing phase. In the case that indexing is the bottleneck, the syncmer variants had lower memory usage than the original mappers across most of the range of compression values (Figure 10B).

Task	Method	Index time	Index mem	Map time	Map mem
bacterial reads vs BAC	minimap2	4.81	0.318	**23.4**	2.886
	Syncmer minimap	**3.73**	0.307	29.96	2.793
	Winnowmap2	4.98	0.31	40.1	2.855
	Syncmer winnowmap	6.14	**0.301**	54.53	**2.691**
ChrX reads vs CHM13X	minimap2	10.41	1.009	**56.71**	6.277
	Syncmer minimap	**9.51**	0.99	74.03	6.104
	Winnowmap2	11.51	0.995	90.243	6.226
	Syncmer winnowmap	16.44	**0.95**	138.16	**5.925**
bacterial reads vs CHM13X	minimap2			**23.1**	2.878
	Syncmer minimap	As	As	31.08	2.787
	Winnowmap2	above	above	36.54	2.845
	Syncmer winnowmap			55.34	**2.668**

Table 4: **Runtime and memory.** Time (in seconds) and RAM (in GB) needed to index the reference and map the simulated reads by each of the tools. The best performing tool in each column is indicated in bold. The second and third dataset use the same reference.

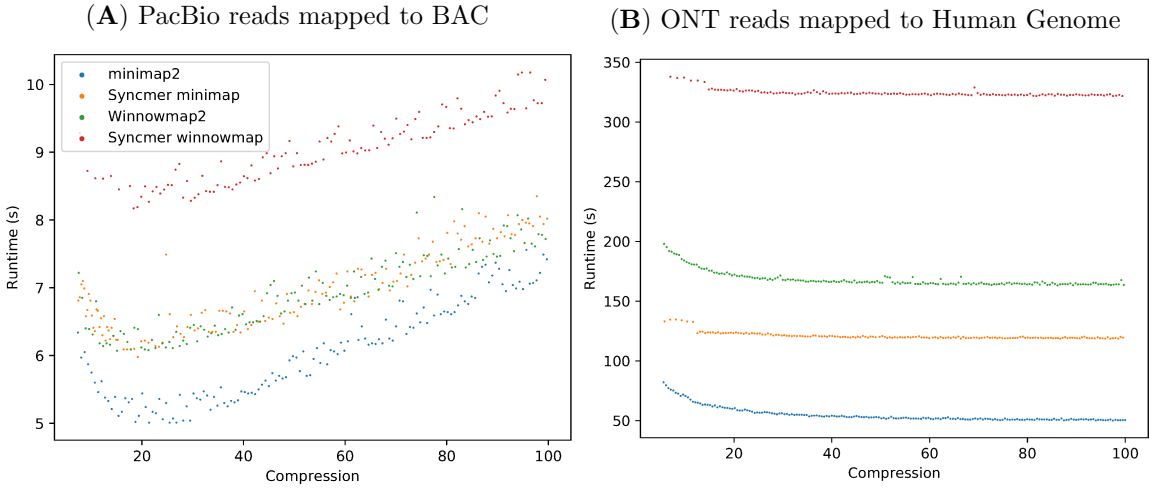

Figure 9: **Runtime vs. compression – real data.** The figures show runtime in seconds to index the reference and map reads by each method. **(A)** PacBio bacterial reads. **(B)** ONT human cell-line reads.

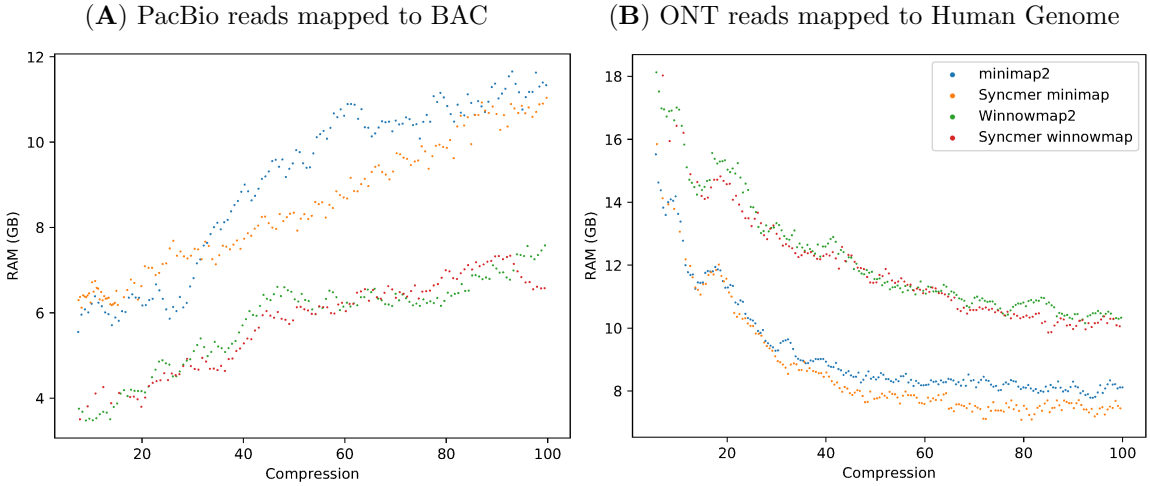

Figure 10: **Memory usage vs. compression – real data.** Peak RAM usage in GB to index the reference and map reads for the different methods. **(A)** PacBio bacterial reads. **(B)** ONT human cell-line reads.

6 Discussion

In this study we generalized the notion of syncmers to PSSs and derived their theoretical properties. We incorporated PSSs into the long read mappers minimap2 and Winnowmap2. Our syncmer mappers outperformed minimap2 and Winnowmap2 and succeeded in mapping more long reads across a range of different compression values for multiple real and simulated datasets.

As our results show, the advantage of using syncmers is most marked at high compression and high error rate, as is expected due to their higher conservation. Yet the advantage is present at the lower compression used by existing mappers. For large genomes, such as the human genome, using the higher compression enabled by syncmers also leads to lower RAM usage. Syncmer-minimap is slower than the highly optimized minimap2, taking 50-100% longer to map reads, but it is faster than Winnowmap2. Future work should focus on lowering the runtime by further optimizing the syncmer mapping implementation.

There are a number of issues and questions that this work leaves open, particularly in the theoretical analysis. First, the analysis of windowed schemes and downsampled schemes under mutation remains to be completed. Second, an expression for ℓ_2 for minimizer schemes could also be obtained. Third, can the theory be expanded to canonical k-mers? Fourth, it may be possible to obtain more robust definitions of conservation and ℓ_2 that do not depend on preserving indices between sequences, thereby allowing indels to be included in the theoretical analysis.

Another possible avenue to explore is in the definition of the selection scheme itself. Is it possible to select k-mers in a biased way to increase the compression but still retain the beneficial distance distribution of syncmer schemes? The quest for an "optimal" scheme is

not over.

Acknowledgement

Study supported in part by the Israel Science Foundation (grant No. 3165/19, within the Israel Precision Medicine Partnership program, and grant No. 1339/18) and by Len Blavatnik and the Blavatnik Family foundation. DP was supported in part by a fellowship from the Edmond J. Safra Center for Bioinformatics at Tel-Aviv University.

References

[1] F. R. Blattner, G. Plunkett, C. A. Bloch, N. T. Perna, V. Burland, M. Riley, J. Collado-Vides, J. D. Glasner, C. K. Rode, G. F. Mayhew, et al. The complete genome sequence of Escherichia coli K-12. *Science*, 277(5331):1453–1462, 1997.

[2] R. Chikhi, A. Limasset, and P. Medvedev. Compacting de Bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics*, 32(12):i201–i208, 2016.

[3] J. C. Dohm, P. Peters, N. Stralis-Pavese, and H. Himmelbauer. Benchmarking of long-read correction methods. *NAR Genomics and Bioinformatics*, 2(2):lqaa037, 2020.

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

[5] C. Jain, A. Rhie, N. Hansen, S. Koren, and A. M. Phillippy. A long read mapping method for highly repetitive reference sequences. *bioRxiv*, 2020.

[6] C. Jain, A. Rhie, H. Zhang, C. Chu, B. P. Walenz, S. Koren, and A. M. Phillippy. Weighted minimizer sampling improves long read mapping. *Bioinformatics*, 36(Supplement_1):i111–i118, 2020.

[7] M. Kokot, M. Długosz, and S. Deorowicz. KMC 3: counting and manipulating k-mer statistics. *Bioinformatics*, 33(17):2759–2761, 2017.

[8] H. Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18):3094–3100, 2018.

[9] H. Li. New strategies to improve minimap2 alignment accuracy. *arXiv preprint arXiv:2108.03515*, 2021.

[10] S. Nurk, S. Koren, A. Rhie, M. Rautiainen, A. V. Bzikadze, A. Mikheenko, M. R. Vollger, N. Altemose, L. Uralsky, A. Gershman, et al. The complete sequence of a human genome. *bioRxiv*, 2021.

[11] Y. Ono, K. Asai, and M. Hamada. PBSIM: PacBio reads simulator—toward accurate genome assembly. *Bioinformatics*, 29(1):119–121, 2013.

[12] PacificBiosciences. Microbial Multiplexing Data Set 48 plex: PacBio Sequel II System, Chemistry v2.0, SMRT Link v8.0 Analysis. `https://github.com/PacificBiosciences/DevNet/wiki/Microbial-Multiplexing-Data-Set---48-plex:-PacBio-Sequel-II-System,-Chemistry-v2.0,-SMRT-Link-v8.0-Analysis`, 2019. Accessed: 15-10-2021.

[13] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85, 2003.

[14] V. A. Schneider, T. Graves-Lindsay, K. Howe, N. Bouk, H.-C. Chen, P. A. Kitts, T. D. Murphy, K. D. Pruitt, F. Thibaud-Nissen, D. Albracht, et al. Evaluation of GRCh38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome Research*, 27(5):849–864, 2017.

[15] F. J. Sedlazeck, P. Rescheneder, M. Smolka, H. Fang, M. Nattestad, A. Von Haeseler, and M. C. Schatz. Accurate detection of complex structural variations using single-molecule sequencing. *Nature methods*, 15(6):461–468, 2018.

[16] J. Shaw and Y. W. Yu. Theory of local k-mer selection with applications to long-read alignment. *bioRxiv*, 2021.

[17] D. E. Wood, J. Lu, and B. Langmead. Improved metagenomic analysis with Kraken 2. *Genome Biology*, 20(1):1–13, 2019.

[18] C. Yang, J. Chu, R. L. Warren, and I. Birol. NanoSim: nanopore sequence read simulator based on statistical characterization. *GigaScience*, 6(4):gix010, 2017.

Supplementary information

S1 Syncmer based mapping implementations

Modifications to the mappers were minimal. Only the code that selects the k-mers to use as seeds to index the reference sequence and as anchors from the query reads was modified. Here we describe implementation details and optimizations in the code that differ from the high-level descriptions in Algorithm 1 and 2.

In minimap2 the most frequent minimizers (0.02% by default) are dropped to reduce spurious matches and lower the runtime and memory usage. We also drop the most frequent selected k-mers as the last stage of all minimap syncmer variants for consistency. In

Winnowmap, the most frequent k-mers (also 0.02% by default) are re-weighted in the minimizer order so they are less likely to be selected as minimizers. In the syncmer-winnowmap variant, we do not consider k-mer weighting, and thus we simply drop these k-mers if they are selected. However, in the *windowed* syncmer-winnowmap variant we do re-weight the frequent k-mers before selecting minimizers in empty windows.

We use several different hashes in our syncmer variants of the mappers: h_{can} to select canonical k-mers, h_s to select s-minimizers, h_{min} to select minimizers for windowed variants and h_{down} for downsampling. We require that $h_{can} \neq h_{down}$ to maintain random downsampling. In minimap syncmer variants we use hash64 from minimap2 for h_{min} and a variant of MurmurHash2 that ensures $murmur2(0) \neq 0$ to ensure randomness for the other hashes. Thus $h_{min} = -hash64/\text{UINT64\_MAX}$, $h_s(x) = murmur2(x)$, $h_{down}(x) = murmur2(x)$, and $h_{can}(x) = murmur2(x << 1 + 5)$ to ensure that it has a different value than h_{down}. For winnowmap variants we use $h_{can}(x) = lexicographic(x)$ as this is what is used by the k-mer counter Meryl, $h_{min}(x) = -(murmur2(x)/\text{UINT64\_MAX})^8$ in the case that the minimizer is one of the most frequent and $-murmur2(x)/\text{UINT64\_MAX}$ otherwise. The other hashes are as in the minimap variants.

In all windowed variants, downsampling occurs before filling in empty windows with minimizers.

ONT reads were mapped using the `map-ont` option in all mappers, while PacBio reads were mapped using the `map-pb` option (`map-pb-clr` in Winnowmap and variants). The latter uses homopolymer compression (HPC) and thus has a real compression (on the non-HPC sequence) that is above the theoretical one.

S2 Simulation parameters and details

PacBio reads were simulated using PBSim [11] with error rates and read lengths roughly matched to the statistics observed in a recent benchmark of long read correction methods [3] unless otherwise indicated.

PacBio reads were simulated using PBSim with the CLR model and the following parameter settings: depth 10, mean length 9000, length std 6000, minimum length 100, maximum length 40000, mean accuracy 0.87, accuracy std 0.02, minimum accuracy 0.85, maximum accuracy 1, and difference ratio 10:48:19.

ONT reads were simulated using NanoSim [18] with default parameters and the human pre-trained model for Guppy base calls.

To evaluate mapping correctness for the PacBio simulated data we used the mapeval utility of paftools packaged with minimap2. In this tool, reads are considered correctly mapped if the overlap between the read alignment and the true read location is $\geq 10\%$ of the combined length of the true read and aligned read interval. This criterion was also used in [6].

S3 Bacterial species used

We chose a single representative assembly of each strain from [12] with the fewest, longest and most highly covered contigs and concatenated all references into a single fasta file. Reads from the same samples were downloaded. Assemblies and reads from following samples were used:

- bc1019, Bacillus cereus 971 (ATCC 14579)
- bc1059, Bacillus subtilis W23
- bc1101, Burkholderia cepacia (ATCC 25416)
- bc1102, Enterococcus faecalis OG1RF (ATCC 47077D-5)
- bc1111, Escherichia coli K12
- bc1087, Escherichia coli W (ATCC 9637)
- bc1018, Helicobacter pylori J99 (ATCC 700824)
- bc1077, Klebsiella pneumoniae (ATCC BAA-2146)
- bc1082, Listeria monocytogenes (ATCC 19117)
- bc1043, Methanocorpusculum labreanum Z (ATCC 43576)
- bc1047, Neisseria meningitidis FAM18 (ATCC 700532)
- bc1054, Rhodopseudomonas palustris
- bc1119, Staphylococcus aureus HPV (ATCC BAA-44)
- bc1079, Staphylococcus aureus subsp. aureus (ATCC 25923)
- bc1052, Treponema denticola A (ATCC 35405)

S4 Properties of syncmer schemes on real genome sequences without mutation

The theoretical properties measured on real genomes (without mutation) are shown in Table S1.

S5 Windowed syncmer scheme results

Tables S2 and S3 present the properties of windowed syncmer schemes on real genome sequences with and without mutation, respectively.

Figures S1 and S2 present the number of unmapped reads and wrongly mapped reads for simulated datasets. These correspond to Figures 5 and 7 and include the results for windowed variants. Figure S3 presents the impact of percent sequence identity on the windowed variants as well, corresponding to Figure 8.

Results on the real human and bacterial reads are presented in Figure S4, and the runtimes and RAM usage for these runs are in Figures S5 and S6. The runtime and memory usage on different tasks for the windowed variants is presented in Table S4.

Dataset	Scheme	Compression	ℓ	ℓ_2	$p90$	$p100$	# positions
ECK12	$\mathcal{M}_{15,10}$	5.503	3.23×10^{-6}	0.005	9	10	843,408
	$Sy_{15,5}(3,9)$	5.490	0.0191	0.377	10	53	845,419
	$\mathcal{M}_{15,19}$	9.989	0.0527	0.398	18	19	464,660
	$Sy_{15,6}(6)$	9.986	0.133	1.375	19	99	464,810
CHM13X	$\mathcal{M}_{15,10}$	5.489	5.83×10^{-8}	0.0005	9	10	28,099,399
	$Sy_{15,5}(3,9)$	5.523	0.0205	0.412	10	161	27,930,897
	$\mathcal{M}_{15,19}$	9.977	0.0526	0.3974	18	19	15,461,458
	$Sy_{15,6}(6)$	10.055	0.137	1.419	20	735	15,342,238

Table S1: **Properties of minimizer and syncmer schemes on real sequences – no mutation.** The best syncmer schemes with theoretical compression of 5.5 and 10 were chosen. The table shows the actual compression and other metrics on the real sequences, # positions is the number of positions that were selected by the scheme.

w	δ	c	ℓ	ℓ_2	$p90$	$p100$	# positions	# syncmer	# minimizer
10	1	4.775	2.59×10^{-8}	0.000255	9	10	32,309,357	27,969,919	4,339,438
15	1	5.322	2.59×10^{-8}	0.000255	10	15	28,987,572	27,969,919	1,017,653
20	1	5.464	0.01047	0.1862	10	20	28,230,846	27,969,919	260,927
25	1	5.501	0.0167	0.3014	10	25	28,040,255	27,969,919	70,336
50	1	5.515	0.0197	0.3899	10	50	27,970,312	27,969,919	393
100	1	5.515	0.01973	0.3915	10	99	27,969,930	27,969,919	11
10	2	5.463	6.48×10^{-8}	0.000729	9	10	28,238,334	14,002,723	14,235,611
15	2	7.435	6.48×10^{-8}	0.000729	14	15	20,748,902	14,002,723	6,746,179
20	2	8.812	0.0493	0.4201	17	20	17,505,813	14,002,723	3,503,090
25	2	9.706	0.10495	0.8324	20	25	15,892,867	14,002,723	1,890,144
50	2	10.93	0.1986	1.788	23	50	14,112,958	14,002,723	110,235
100	2	11.015	0.2067	1.9811	23	100	14,003,934	14,002,723	1211

Table S2: **Properties of windowed syncmers on CHM13X.** Properties of windowed and down-sampled variants of $Sy_{15,5}(3,9)$ are shown for a range of window lengths w on the human chromosome X sequence of CHM13. δ is the downsampling rate and c is the actual compression. The theoretical compression of the PSS (not downsampled) is 5.5. # positions is the number of positions that were selected by the scheme, # syncmers is the number of those that were selected by the PSS and # minimizers is the number of minimzers added to fill in gaps of length $\geq w$.

w	δ	c	ℓ	ℓ_2	$p90$	$p100$	# conserved	conserved syncmers	conserved minimizers
10	1	41.745	0.7826	10.713	122	1221	3,695,319	3,378,302	317,017
15	1	45.008	0.7928	11.041	131	1221	3,427,368	3,378,302	49066
20	1	45.558	0.7951	11.1125	132	1221	3,386,019	3,378,302	7717
25	1	45.646	0.7955	11.126	132	1221	3,379,466	3,378,302	1164
50	1	45.662	0.7956	11.129	133	1274	3,378,304	3,378,302	2
100	1	45.662	0.7956	11.128	133	1274	3,378,302	3,378,302	0
10	2	52.34	0.8062	11.515	147	1221	2,947,249	1,692,160	1,255,089
15	2	71.027	0.8397	12.898	190	1691	2,171,847	1,692,160	479,687
20	2	81.734	0.8572	13.774	216	1703	1,887,340	1,692,160	195,180
25	2	86.999	0.8655	14.257	230	1736	1,773,121	1,692,160	80961
50	2	91.104	0.8722	14.709	242	1736	1,693,222	1,692,160	1062
100	2	91.161	0.8723	14.718	242	1736	1,692,160	1,692,160	0

Table S3: **Properties of conserved windowed syncmers under mutation.** Properties of windowed and down-sampled variants of $Sy_{15,5}(3,9)$ are shown for a range of window lengths w on CHM13X after simulating substitutions at a rate of 15%. w is the window size and δ is the downsampling rate. Properties of the conserved selected k-mers are reported.

(**A**) PacBio reads mapped to ChrX of GRCh

(**B**) ONT reads mapped to GRCh

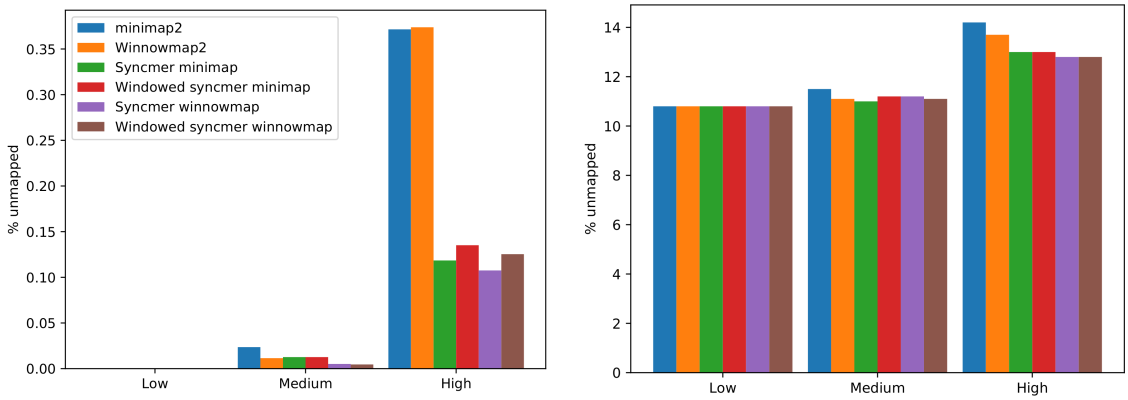

Figure S1: **Percentage of unmapped reads – simulated datasets.** The percentage of unmapped reads is plotted for two simulated read datasets mapped to their reference sequences. Results are shown for low, medium, and high compression. **(A)** PacBio reads simulated from the CHM13 ChrX sequence mapped against ChrX sequences from GRCh38. Window sizes of windowed syncmer-minimap were $w = 13, 77, 175$ for the low, medium, and high compression variants, respectively. For windowed syncmer-winnowmap the window sizes were $w = 14, 75, 170$, respectively. **(B)** 1000 ONT reads simulated from CHM13 mapped against GRCh38. Window sizes of windowed syncmer-minimap were $w = 13, 75, 175$ for the low, medium, and high compression variants, respectively and $w = 13, 75, 170$ for the corresponding windowed syncmer-winnowmap variants.

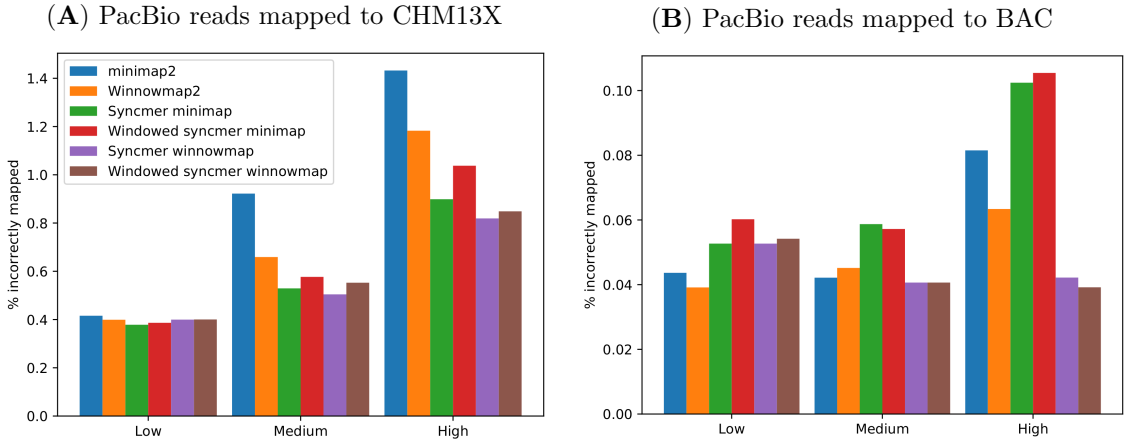

Figure S2: **Percentage of incorrectly mapped reads – simulated data.** The percentage of incorrectly mapped reads is plotted for two simulated read datasets and their reference sequences, for mappers using low, medium, and high compression. **(A)** PacBio reads simulated from the CHM13 ChrX sequence mapped against CHM13X. Window sizes of windowed syncmer-minimap were $w = 13, 80, 165$ for the low, medium, and high compression variants, respectively. For windowed syncmer-winnowmap they were $w = 14, 75, 170$, respectively. **(B)** PacBio reads simulated from the 15 bacterial species in BAC mapped against the union of their references. Window sizes of windowed syncmer-minimap were $w = 13, 75, 175$ and in windowed syncmer-winnowmap they were $w = 16, 75, 170$ for the low, medium, and high compression variants, respectively.

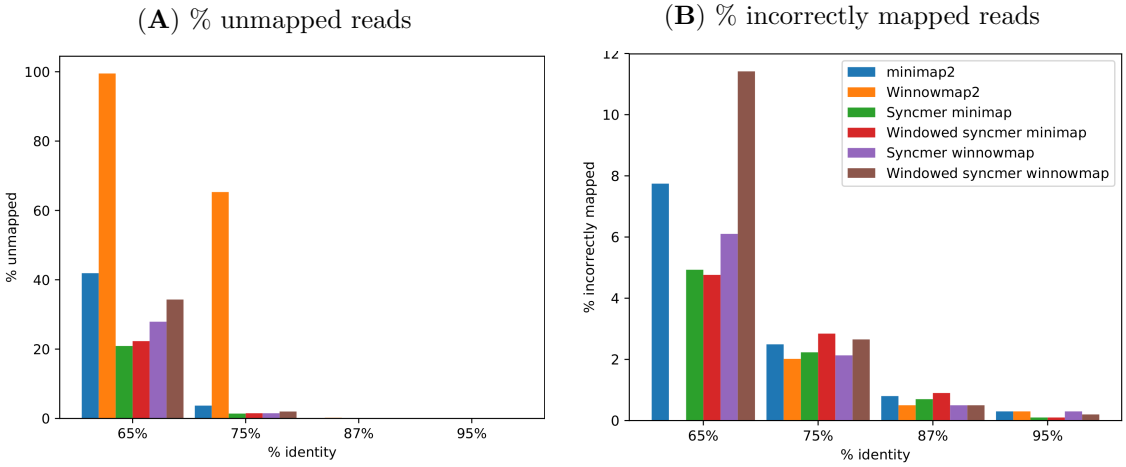

Figure S3: **Impact of percent sequence identity.** We varied the mutation rate of 1000 PacBio simulated reads from CHM13X. The figures present the % unmapped and incorrectly mapped for each of the tools. **(A)** % unmapped reads. **(B)** % of the mapped reads that were incorrectly mapped.

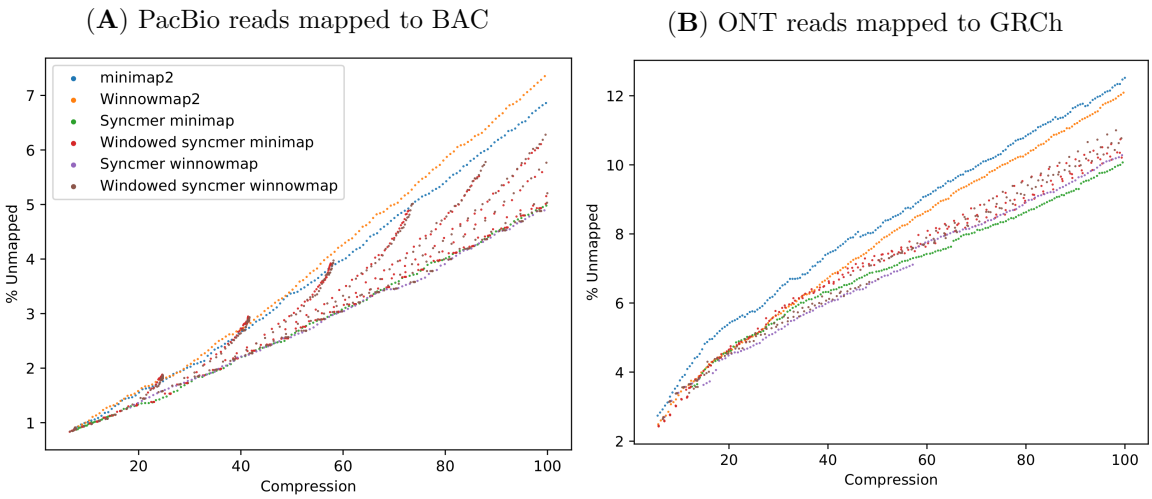

Figure S4: **Percentage of unmapped reads – real datasets.** Results are shown across a broad range of compression rates. **(A)** Pooled PacBio bacterial reads. **(B)** ONT human cell-line reads.

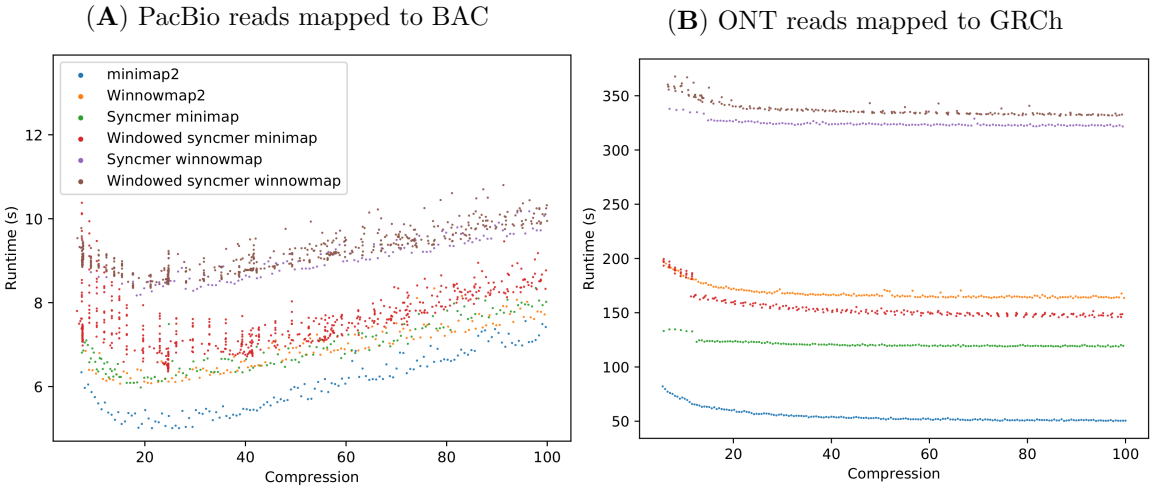

Figure S5: **Runtime vs. compression – real data.** The figures show runtime in seconds to index the reference and map reads by each method. **(A)** PacBio bacterial reads. **(B)** ONT human cell-line reads.

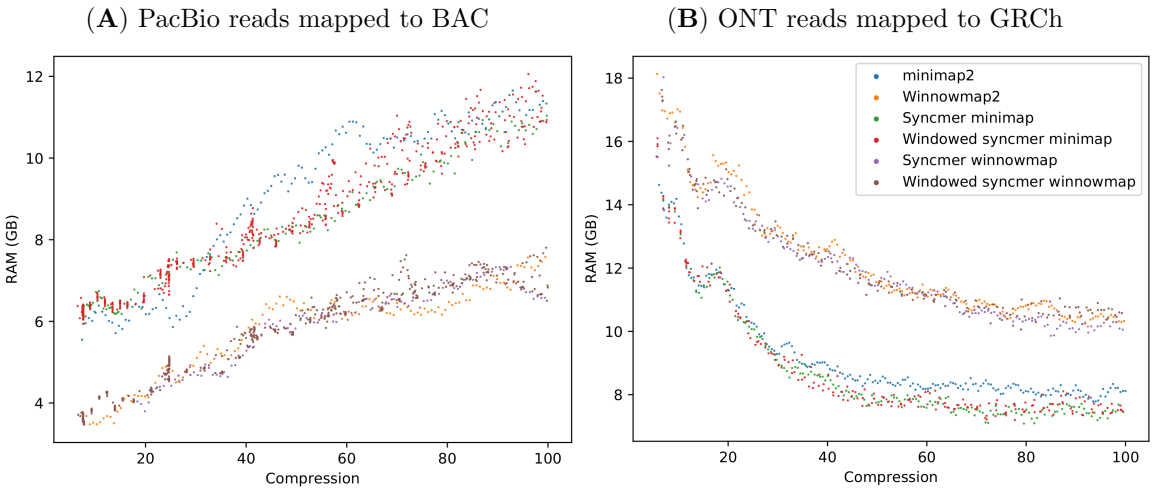

Figure S6: **Memory usage vs. compression – real data.** Peak RAM usage in GB to index the reference and map reads for the different methods. **(A)** PacBio bacterial reads. **(B)** ONT human cell-line reads.

Task	Method	Index time	Index mem	Map time	Map mem
pbsim bac vs BAC	Windowed syncmer minimap	4.41	0.316	36.75	2.824
	Windowed syncmer winnowmap	6.53	0.305	58.75	2.742
pbsim chm13x vs CHM13X	Windowed syncmer minimap	11.51	1.007	92.164	6.224
	Windowed syncmer winnowmap	17.23	0.992	148.988	6.115
pbsim bac vs CHM13X	Windowed syncmer minimap	As above	As above	36.69	2.846
	Windowed syncmer winnowmap			59.52	2.8

Table S4: **Runtime and memory.** Time (in seconds) and RAM (in GB) needed to index the reference and map reads by each of the tools. The best performing tool in each column is indicated in bold. The second and third dataset use the same reference and therefore have the same indexing results.

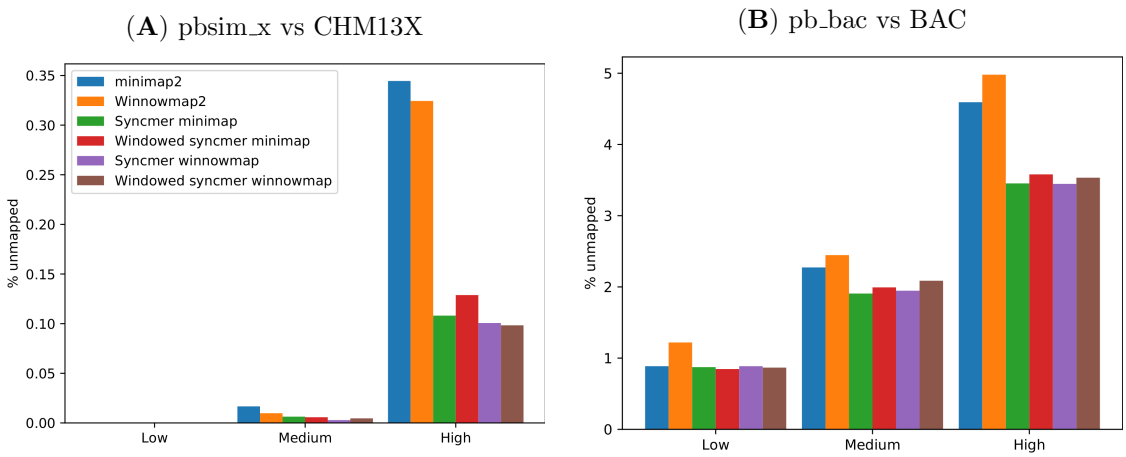

Figure S7: **Percentage of unmapped reads – additional data.** The percentage of unmapped reads is plotted for one simulated and one real read dataset mapped to their corresponding references. **(A)** PacBio reads simulated from the CHM13 ChrX sequence mapped against CHM13X. Window sizes of windowed syncmer-minimap were $w = 13, 80, 165$ for the low, medium, and high compression variants, respectively, and for windowed syncmer-winnowmap they were $w = 14, 75, 170$, respectively. **(B)** 1000 PacBio reads sampled from each of the 15 bacterial species in BAC mapped against their reference genomes. Window sizes of windowed syncmer-minimap were $w = 13, 75, 175$ and in windowed syncmer-winnowmap they were $w = 16, 75, 170$ for the low, medium, and high compression variants, respectively.

S6 Supplemental performance results

Figure S7 shows additional results for the number of unmapped reads at low, medium, and high compression rates.