# Seqwin: Ultrafast identification of signature sequences in microbial genomes

**Michael X. Wang [1,2], Bryce Kille [1], Michael G. Nute [1], Siyi Zhou [2], Lauren B. Stadler [2], and Todd J. Treangen [1,3,4,*]**

[1]Department of Computer Science, Rice University, Houston, 77005, USA
[2]Department of Civil and Environmental Engineering, Rice University, Houston, 77005, USA
[3]Department of Bioengineering, Rice University, Houston, 77030, USA
[4]Ken Kennedy Institute, Rice University, Houston, 77005, USA
[*]treangen@rice.edu

## ABSTRACT

**Motivation:** Polymerase chain reaction (PCR) enables rapid, cost-effective diagnostics but requires prior identification of genomic regions that allow sensitive and specific identification of target microbial groups, herein referred to as microbial signature sequences. We introduce Seqwin, an open-source framework designed to automate microbial genome signature discovery. Tens of thousands of microbial genomes are now available, limiting the application of existing manual and automated approaches for identifying signatures. Modern approaches that are capable of leveraging all available microbial genomes will ensure sensitive and accurate DNA signatures identification and enable robust pathogen detection for clinical, environmental, and public health applications.

**Results:** Seqwin builds weighted pan-genome minimizer graphs and uses a traversal algorithm to identify signature sequences that occur frequently in target genomes but remain rare in non-targets. Unlike earlier tools that depend on strict presence or absence of sequences, Seqwin accommodates natural sequence variation and scales to very large genome collections. When applied to genomes from *C. difficile*, *M. tuberculosis* and *S. enterica*, Seqwin recovered more high-quality signatures than alternative methods with lower computational burden. Seqwin analysis of nearly 15,000 *S. enterica* genomes yielded over 200 candidate signatures in less than 10 minutes. Seqwin provides an open-source solution for the long-standing need for scalable microbial signature discovery and diagnostic assay design.

**Availability:** Seqwin is freely available for academic use (https://github.com/treangenlab/Seqwin) and can be installed via Bioconda.

## Introduction

PCR-based infectious disease diagnostics have been the clinical standard for more than three decades, providing rapid and reliable detection of specific pathogens[1,2]. Clinical studies in the late 1980s, such as the detection of HIV-1 DNA in infants and adults, first demonstrated its utility in patient diagnosis. These early advances set the stage for the advancement of PCR-based infectious disease diagnostics. Signature sequences are genomic regions that allow accurate identification and classification of microorganisms within specific taxonomic groups[3]. An ideal signature is highly sensitive, consistently present in most target genomes, and highly specific, absent or markedly divergent in closely related non-target genomes[4–6].

Early computational methods for signature discovery mainly focused on the design of TaqMan[7] and microarray[8] assays, where sensitivity and specificity are critical[9–17]. Approaches such as YODA[10] and ProDesign[12] used exhaustive searches combined with nearest-neighbor thermodynamic filtering, while Insignia[13,14] and CaSSiS[17] leveraged maximal unique matches (MUMs) identified via a suffix tree data structure. While all of these represent significant contributions to the computational task of microbial signature discovery, previous approaches suffer from several limitations. First, tools developed for the design microarray probes typically produce short signatures ($\leq$50 bp)[10–12,15,16], which are insufficient as robust genomic markers for modern amplicon sequencing[18], digital PCR (dPCR)[19] and other PCR-based assays. Second, many of these methods were developed before the widespread adoption of next-generation sequencing (NGS), when the availability of genomic data was considerably limited. When Insignia pioneered open-source microbial signature discovery, only tens to hundreds of genomes were available for a given microbial target. Consequently, these tools were not designed to handle the massive scale of modern genomic databases that are on the scale of terabytes and petabytes. Furthermore, due to their reliance on exact matches across all target genomes, these methods are highly sensitive to sequence variation, limiting their application in the context of increasingly diverse genomic datasets. On the other hand, non-exact, match-based approaches provide improved sensitivity but poor scalability. For example, SigSeekr[20] relies on BLAST-based genome subtraction methods, systematically identifying

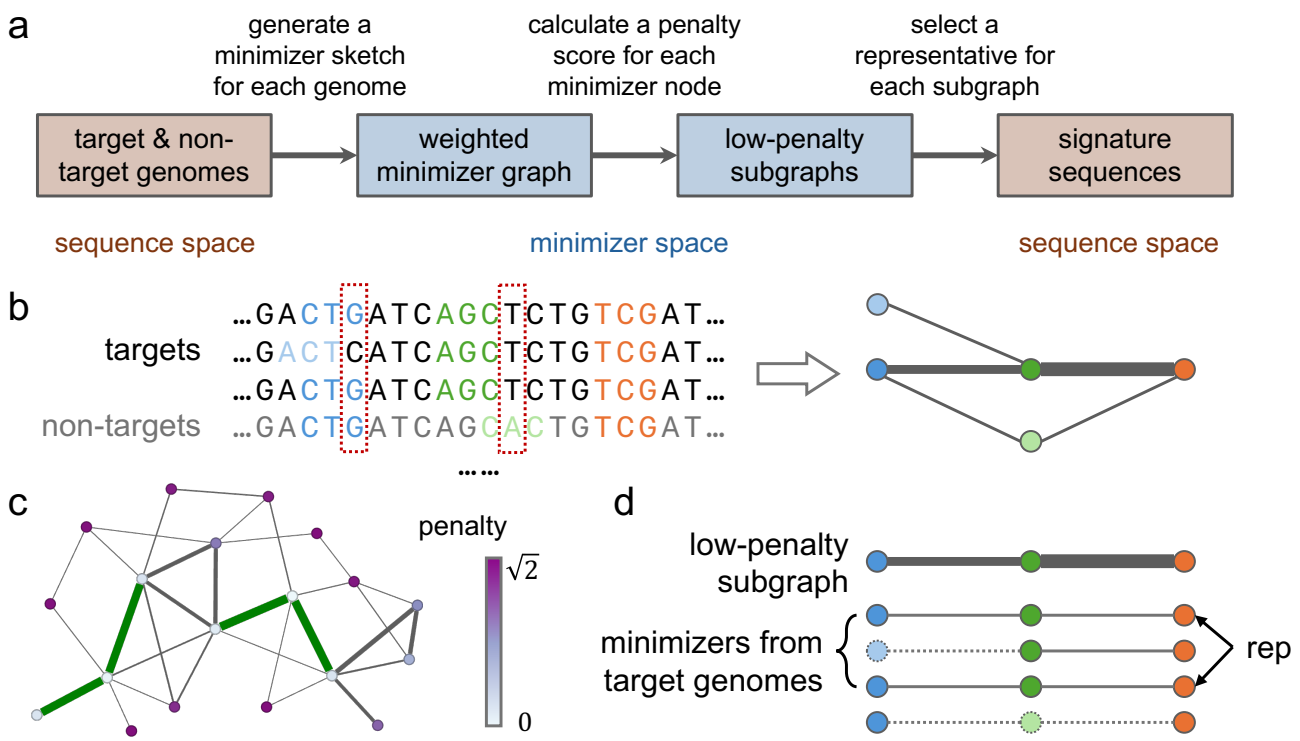


**Figure 1.** Overview of Seqwin workflow. **a** Seqwin first compresses all input genomes into a weighted pan-genome minimizer graph. It then extracts connected low-penalty subgraphs and selects a representative for each subgraph in minimizer space. It finally converts each representative back to sequence space as the signature sequence. **b** Graph construction (a toy example). Each unique minimizer is colored. Input genomes may yield different sets of minimizers due to sequence variations (red dashed box). Adjacent minimizers are connected with a weighted, undirected edge in the graph. The thickness of each edge is proportional to its weight. Target genomes (black) and non-target genomes (gray) are treated equally. **c** A penalty score is calculated for each minimizer node based on the L2 norm of its absence in target genomes and presence in non-target genomes (color map). Sets of connected nodes with low average penalty and high edge weights are extracted as low-penalty subgraphs (green). **d** A representative is selected for each low-penalty subgraph. The underlying minimizers from all target genomes are listed for a low-penalty subgraph (blue, green and orange; light blue and light green are minimizers not included in this subgraph). The most predominant minimizer ordering is selected as the representative (rep), and the corresponding genomic sequence is selected as the signature (e.g., the first or third sequence in panel **b**).

unique genomic regions, but suffers from prohibitive runtime on large-scale datasets.

Recent years have seen a new wave of tools for genomic signature identification that emphasize scalability and flexibility. Many of these newer methods abandon all-versus-all sequence alignment in favor of ultrafast *k*-mer based strategies or clever filtering techniques. Fur[21,22] combines targeted genomic subtraction and stringent intersection methods to discover unique genomic regions; however, it struggles with highly similar background sequences and can sometimes produce few or no candidate sequences. Methods such as KmerGO[23], KEC[24] and Unikseq[25,26] employ alignment-free *k*-mer filtering combined with *k*-mer assembly or similar strategies, yet require high memory usage to store all *k*-mers from all input genomes. Furthermore, NAUniSeq[27] integrates phylogenetic guidance with *k*-mer indexing to streamline marker discovery against closely related taxa, simplifying user input but requiring accurate phylogenies and comprehensive reference databases. Table 1 summarizes these tools, including their publication year and approach. While advances in computational tools have provided more scalable solutions, each still grapples with trade-offs related to sensitivity and scalability on terabyte-sized datasets, motivating the development of solutions that can address this unmet need.

To address this gap, we developed Seqwin, a rapid and sensitive algorithm for the identification of genomic signatures based on flexible pan-genome minimizer graphs. Unlike previous minimizer graph-based methods that require each minimizer to be present in all target genomes[28,29], Seqwin allows the inclusion of all minimizers from any input genomes and penalizes those absent in targets and/or present in non-targets. It then extracts connected low-penalty subgraphs and determines a representative sequence (signature) for each subgraph with a rapid and memory-efficient work-

flow. This novel minimizer graph approach increases the retention of original sequence information while keeping a low memory profile, and enables fast and flexible search of genomic regions of interest, such as genomic signatures. On a small benchmark set of *E. coli* genomes, Seqwin achieved improved sensitivity and specificity compared to Fur[21,22] and Unikseq[25,26], while requiring less time and memory. On three benchmarking sets retrieved from NCBI Taxonomy[33], Seqwin identified a greater number of high-quality genomic signatures while using fewer computational resources (RAM and CPU). These experiments demonstrate that Seqwin not only discovers more candidate signatures than previous methods, but does so with much improved computational efficiency.

Due to its flexible design, Seqwin performs well when identifying signatures in microbial genomes of varying quality and completeness, a critically important feature given the heterogeneity and rapid expansion of contemporary genomic databases. This enables Seqwin to operate efficiently in diverse contexts (low microbial biomass clinical settings and wastewater surveillance) while minimizing bias introduced by incomplete or low-quality genome assemblies. In turn, Seqwin provides a scalable and computationally efficient framework for microbial pathogen detection, facilitating rapid identification of candidate signature regions for a variety of downstream pathogen detection use cases.

**Table 1.** Existing tools for identifying signature sequences in microbial genomes.

| Name | Year [1] | Approach |
|---|---|---|
| Insignia[13,14] | 2009 | Suffix tree |
| SigSeekr[20] | 2015 | BLAST[30] |
| KmerGO[23] | 2020 | *k*-mer |
| KEC[24] | 2021 | *k*-mer |
| Fur[21,22] | 2024 | Filtering pipeline [2] |
| NAUniSeq[27] | 2024 | *k*-mer |
| Unikseq[25,26] | 2025 | *k*-mer |
| Seqwin | 2025 | Minimizer graph |

[1] Year of the latest publication.
[2] Macle[31], Phylonium[32] and BLAST[30].

## Results

Seqwin's algorithm includes four main steps (Figure 1a):

1. Generate a minimizer sketch for each input genome and build a weighted pan-genome minimizer graph.
2. Calculate a penalty score for each graph node based on the L2 norm of its absence in target genomes and presence in non-target genomes.
3. Extract connected subgraphs with average node penalty below a threshold (calculated automatically or provided by the user).
4. Choose a representative sequence (signature) for each low-penalty subgraph and calculate its "conservation" (sensitivity) and "divergence" (specificity) scores, by aligning the sequence to all target and non-target genomes with BLAST[30].

Similar to recent *k*-mer based approaches (Table 1), Seqwin allows the inclusion of "imperfect" *k*-mers: *k*-mers that are absent in some target genomes and/or present in some non-target genomes, making it robust to variations and errors in large datasets. However, Seqwin applies a novel minimizer graph algorithm that operates on a $\sim$1% sketch of all input *k*-mers, retaining linear time and space complexity and thereby scaling to tens of thousands of microbial genomes on modest hardware.

With respect to Seqwin's methodological foundation, it first builds a weighted pan-genome minimizer graph, similar to the minimizer graph described by Coombe et al. (2020)[28] but without the restriction that each minimizer should present in all input genomes (Figure 1b). Each graph node represents a distinct minimizer observed in any genome, and each undirected edge connects two minimizers if they are found adjacent in at least one genome, weighted by the number of different genomes in which that minimizer adjacency occurs.

Second, Seqwin evaluates each minimizer node in the minimizer graph, by computing a "penalty" score for each node based on the L2 norm of its absence in target genomes and presence in non-target genomes. For example, a minimizer that is present in all target genomes and not present in any non-target genomes would have a penalty of 0, and a minimizer that is absent from all targets and present in all non-targets would have a penalty of $\sqrt{2}$. Thus, a series of consecutive low-penalty minimizers represents a genomic region that is both prevalent in target genomes and absent / dissimilar in non-target genomes. An example of a minimizer graph with node penalties is shown in Figure 1c.

Third, low-penalty subgraphs are extracted via seeded greedy breadth-first search (BFS) expansion (Figure 1c). Each subgraph consists of a set of connected nodes whose average penalty is below a penalty threshold ($\tau_v$). This tolerates a small number of (relatively) high-penalty nodes in a low-penalty subgraph, resulting in larger subgraphs (longer signatures) and increasing search flexibility. The penalty threshold $\tau_v$ is arguably the most important parameter of Seqwin, distinguishing Seqwin from other tools with strict *k*-mer presence / absence criteria. However, $\tau_v$ should be set according to the homogeneity

**Table 2.** Benchmarking of Seqwin, Fur and Unikseq on the Fur dataset (33 *E. coli* genomes)

| Target strain | # genomes[1] | Tool | # signatures | Median length (bp) | Wall-clock time (s)[2] | Peak memory usage (GB) |
|---|---|---|---|---|---|---|
| A | 6 | Seqwin | 98 | 297 | 30.8 | 0.813 |
| | | Fur | 0 | 0 | 48.9 | 0.729 |
| | | Unikseq | 303 | 231 | 456 | 32.6 |
| B1 | 14 | Seqwin | 98 | 323 | 26.5 | 0.648 |
| | | Fur | 0 | 0 | 35.6 | 0.570 |
| | | Unikseq | 391 | 192 | 463 | 32.6 |
| B2 | 5 | Seqwin | 633 | 324 | 32.2 | 0.817 |
| | | Fur | 25 | 499 | 59.5 | 0.738 |
| | | Unikseq | 442 | 179 | 457 | 32.1 |
| D | 2 | Seqwin | 583 | 332 | 33.5 | 0.886 |
| | | Fur | 31 | 884 | 62.9 | 0.773 |
| | | Unikseq | 236 | 213 | 463 | 32.4 |
| E | 4 | Seqwin | 201 | 324 | 28.5 | 0.795 |
| | | Fur | 18 | 466 | 53.0 | 0.747 |
| | | Unikseq | 198 | 145 | 455 | 32.0 |
| F | 2 | Seqwin | 485 | 344 | 32.5 | 0.884 |
| | | Fur | 17 | 692 | 62.1 | 0.770 |
| | | Unikseq | 267 | 223 | 457 | 32.3 |

[1] Genomes from all other strains were used as non-targets. Genome accessions can be found in Supplementary Table 1.
[2] A single CPU thread was used for Seqwin Fur and Unikseq.

of input genomes. For example, for species with higher intra-species genomic homogeneity (e.g., *Mycobacterium tuberculosis*), a lower penalty threshold is preferred, while for other species such as *Salmonella enterica*, a higher penalty threshold might be preferred. In order to determine $\tau_v$ for arbitrary sets of input target and non-target genomes, we derive an intuitive method to calculate $\tau_v$ based on expected $k$-mer presence / absence, and estimate the expectations with Jaccard indices[34].

Lastly, Seqwin determines a representative sequence for each low-penalty subgraph, based on the most predominant minimizer ordering in target genomes (Figure 1d). This involves finding the maximal consecutive occurrence of the subgraph's minimizers in the minimizer sketch of each target genome, and then choosing the most prevalent minimizer ordering across all target genomes. The genomic sequence corresponding to this minimizer ordering is selected as the signature sequence. A more detailed illustration can be found in Figure S1. Alternatively, one could first extract the genomic sequence corresponding to each maximal consecutive minimizers in each genome, and determine the representative sequence as the consensus sequence of a multiple sequence alignment (MSA). However, calculating an MSA is computationally expensive, especially when the number of target genomes is large. Therefore, Seqwin determines the "consensus" in minimizer space, instead of the conventional sequence space, by choosing the most frequent minimizer ordering of the subgraph in target genomes. This reduces the whole MSA process into counting minimizer tuples, while ensuring the most predominant sequence is selected.

Each candidate signature is evaluated for its sensitivity and specificity. Since both metrics are used to evaluate the performance of an assay (e.g., qPCR and dPCR), they can only be tested given an assay design (e.g., primers and probes) and a wet lab setting. To estimate them in silico, Seqwin runs BLAST[30] on each signature sequence against all target and non-target genomes, and summarizes BLAST results into two metrics: conservation and divergence, representing sensitivity and specificity, respectively. Conservation measures how consistently the signature sequence is conserved among target genomes, while divergence measures how dissimilar it is in non-target genomes. Seqwin outputs signature sequences with both high conservation and divergence as top candidates. Importantly, divergence is calculated based on the mismatches and gaps in the BLAST alignments, provided there are alignments in the non-target genomes. That is, those with no alignment (e.g., completely absent in non-target genomes) are not preferred, since they are more likely to be mobile genetic elements (MGEs), which have been shown to be highly problematic as signature sequences[35], given their ability to cut-and-paste or copy-and-paste into other bacterial genomes. More algorithmic details about Seqwin's workflow and signature evaluation can be found in the Methods

section.

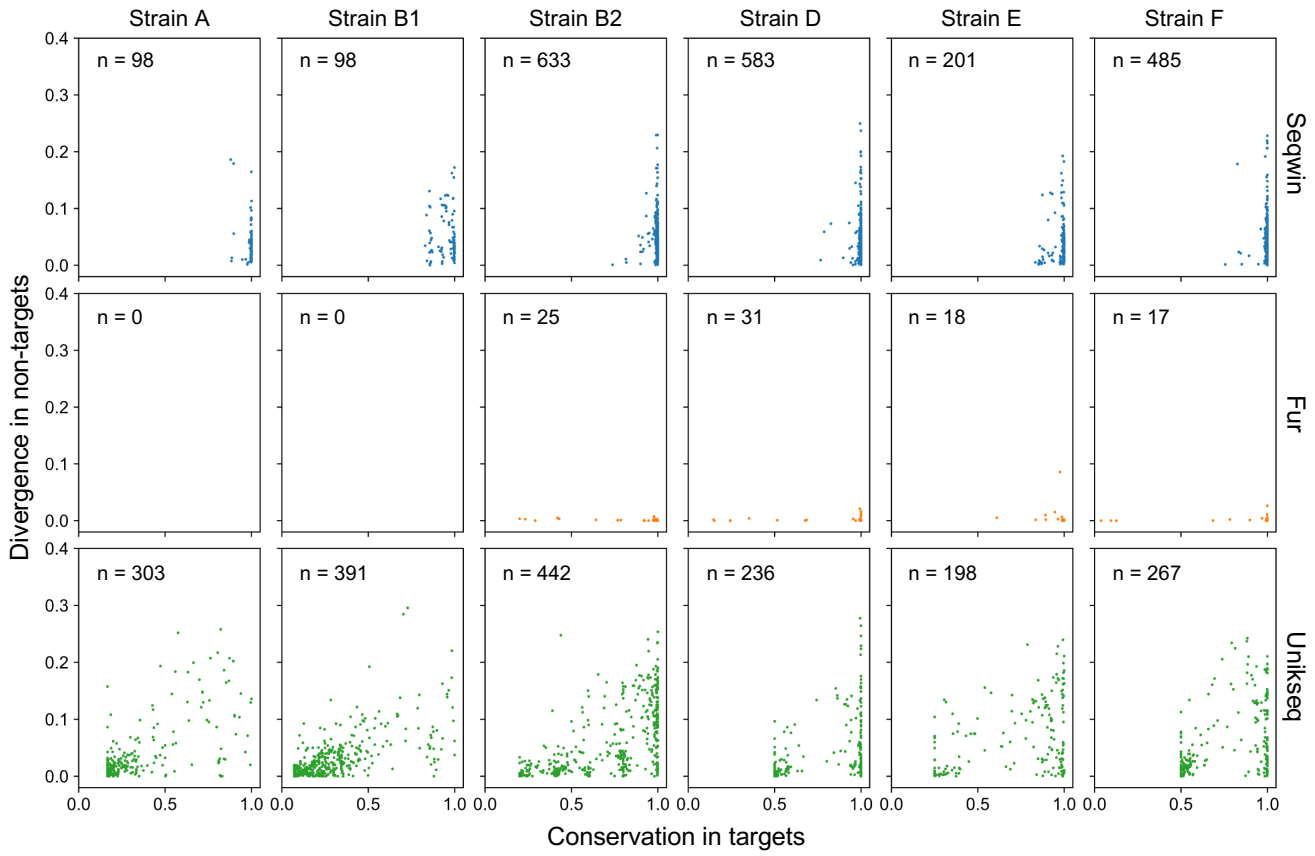


**Figure 2.** Benchmarking of Seqwin, Fur and Unikseq using the Fur dataset (33 *E. coli* genomes). Each data point represents a signature sequence, generated by one of the tools using genomes under one strain as targets, and genomes under the other five strains as non-targets (e.g., plots in column "strain A" are generated by using genomes under strain A as targets). Blue, orange and green represent Seqwin, Fur and Unikseq, respectively. The number of output signatures (data points) in each setting is shown in each single scatter plot.

## Seqwin outperforms existing tools in signature quality, running time and memory usage

To test Seqwin's ability to identify sensitive and specific signature sequences, we compared Seqwin against other computational tools tailored for this task (Table 1). For comparison we chose Fur[21,22] as a fast and memory-efficient method, and Unikseq[25,26] as the most recent one in three of the *k*-mer based methods[23–26]. We did not include NAUniSeq since it was also a *k*-mer based method but required the whole RefSeq[36] and NCBI Taxonomy[33] for phylogenetic guidance.

We first used the dataset published with Fur[21] as input genomes for the three tools (Seqwin, Fur and Unikseq). The Fur dataset consists of 33 *Escherichia coli* genomes from 6 different strains, named as A (6 genomes), B1 (14 genomes), B2 (5 genomes), D (2 genomes), E (4 genomes) and F (2 genomes). For each run of Seqwin, Fur, and Unikseq, genomes under one of the strains were used as target genomes (e.g., strain A), and genomes under all other strains (e.g., B1, B2, D, E and F) were used as non-target genomes. Thus, each tool generated 6 different sets of signatures, summarized in Table 2. Each signature were evaluated for its sensitivity (conservation) and specificity (divergence), with results shown in Figure 2. As a supplement to the divergence score, for each signature we also counted the fraction of non-target genomes with a BLAST hit, shown in Figure S2.

Next, we benchmarked the three tools against microbial genomes retrieved from NCBI Taxonomy[33]. We downloaded all available genomes (August 2025) under three pathogenic taxa and their neighboring taxa: *Clostridioides difficile* (ID 1496, 3,995 genomes), *Mycobacterium tuberculosis* (ID 1773, 8,296 genomes) and *Salmonella enterica* subspecies *enterica* (ID 59201, 14,822 genomes). Genomes indicated under all assembly quality levels were included, while those labeled as "Atypical genomes" and "Genomes from large multi-isolate projects" were excluded. Genomes under each pathogenic taxon were used as targets and genomes under the neighboring taxa were used as non-targets. For each target and non-target group, we sampled

100 or 1,000 genomes as input, with results shown in Table 3 and Figures 3 and S3. We included at least 20 non-target genomes for settings with 100 genomes, since the number of target genomes was much larger than the number of non-target genomes.

Compared to Fur, Seqwin identified more signatures with comparable running time and peak memory (Tables 2 and 3). Fur output zero signatures in several experiments, most likely due to its stringent search strategy. Compared to Unikseq, the output signatures were similar in terms of median length (Tables 2 and 3) in most settings. Although Unikseq output more signatures in most settings, many of them had lower conservation scores (Figures 2 and 3). This was especially true for *S. enterica*, where Unikseq identified only a handful of high-conservation signature sequences. Seqwin was also more efficient with respect to wall-clock time and peak memory usage.

Detailed lists of genomes downloaded and used for all experiments can be found in Supplementary Table 1-4. More details about the benchmark settings can be found in Supplementary Note 1. Signature sequences generated in all experiments are available on Figshare[37].

**Table 3.** Benchmarking of Seqwin, Fur and Unikseq on genomes from NCBI Taxonomy

| Target taxon | # genomes[1] | Tool | # signatures | Median length (bp) | Wall-clock time (s)[2] | Peak RAM usage (GB) |
|---|---|---|---|---|---|---|
| *C. difficile* (ID 1496) | 100 | Seqwin | 33 | 271 | 23.3 | 4.30 |
| | | Fur | 0 | 0 | 10.8 | 3.54 |
| | | Unikseq | 9,074 | 237 | 1,090 | 78.8 |
| | 1,000 | Seqwin | 227 | 313 | 82.2 | 5.75 |
| | | Fur | 0 | 0 | 68.1 | 4.21 |
| | | Unikseq | 10,989 | 186 | 10,600 | 565 |
| | 3,995 | Seqwin | 156 | 312 | 196[3] | 7.85 |
| *M. tuberculosis* (ID 1773) | 100 | Seqwin | 1,674 | 1184 | 24.9 | 4.66 |
| | | Fur | 655 | 372 | 12.4 | 6.14 |
| | | Unikseq | 10,520 | 199 | 1,130 | 71.8 |
| | 1,000 | Seqwin | 150 | 312 | 64.8 | 5.65 |
| | | Fur | 0 | 0 | 63.2 | 7.46 |
| | | Unikseq | 103 | 172 | 11,700 | 576 |
| | 8,296 | Seqwin | 208 | 350 | 338[3] | 11.8 |
| *S. enterica* subsp. *enterica* (ID 59201) | 100 | Seqwin | 382 | 333 | 23.1 | 4.21 |
| | | Fur | 0 | 0 | 11.4 | 6.22 |
| | | Unikseq | 593 | 249 | 1,260 | 77.2 |
| | 1,000 | Seqwin | 319 | 317 | 74.1 | 5.65 |
| | | Fur | 0 | 0 | 52.9 | 4.91 |
| | | Unikseq | 542 | 233 | 13,200 | 624 |
| | 14,822 | Seqwin | 275 | 321 | 579[3] | 21.4 |

[1] Including neighboring non-target genomes. Genome accession can be found in Supplementary Table 2-4.
[2] 20 CPU threads were used for Seqwin and Fur. Unikseq only supported a single CPU thread.
[3] Minimizer sketches were used to calculate the penalty thresholds (see Methods and Supplementary Note 1).

## Seqwin efficiently scales up to thousands of bacterial genomes

Next, we evaluated only Seqwin on all genomes downloaded from NCBI Taxonomy, as Fur did not generate any signatures for the benchmark settings with 1,000 genomes, and Unikseq was estimated to require terabytes of memory (Table 3). For nearly 15k *S. enterica* genomes, Seqwin finished in only 10 minutes using 20 CPU cores and using 22 GB peak memory (Table 3). Seqwin also maintained similar signature quantity and quality as compared to using only 1,000 genomes, as shown in

Figure 4. Note that the penalty thresholds in these experiments were calculated with minimizer sketches to save running time (see Methods for more details).

In addition, we annotated the reference genomes of the target pathogens and identified genes and mobile genetic elements (MGEs), using eggNOG-mapper[38] and the Mobilome Annotation Pipeline (MAP) under MGnify[39], respectively. "Compositional outliers" were potential MGEs identified by MAP (Figure 4), indicating genomic regions with abnormal composition (e.g., GC content) compared to their contexts. Insertion sequences were identified by MAP and were also confirmed by gene annotations, with their gene products being transposase, integrase or resolvase. In the three experiment settings, less than 10% of the signatures overlapped with predicted MGEs, and those overlapped with confirmed MGEs had relatively lower divergence (Figure 4). Annotation results of all signatures shown in Figure 4 can be found in Supplementary Table 5-7. Details of the annotation process can be found in Supplementary Note 2.

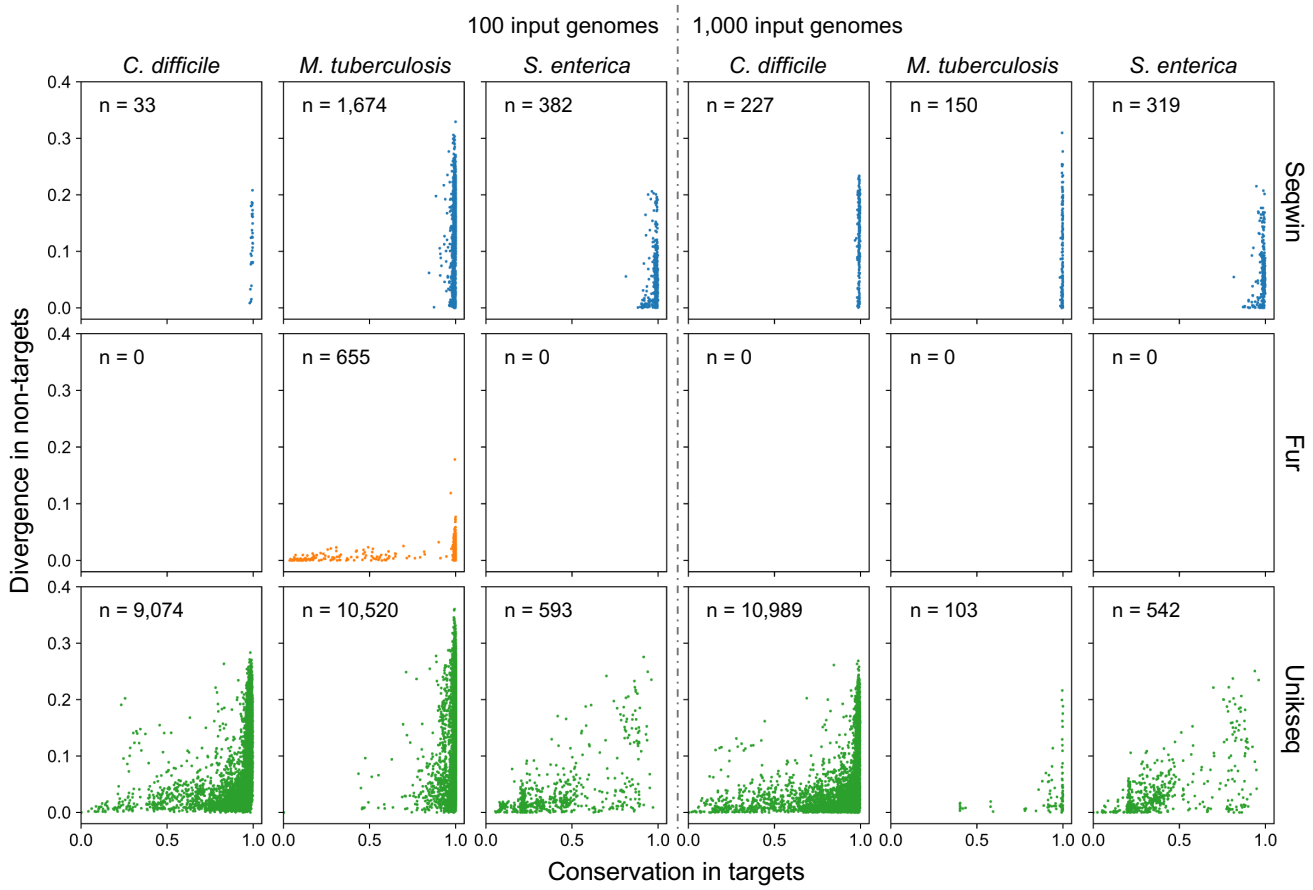


**Figure 3.** Benchmarking of Seqwin, Fur and Unikseq using genomes downloaded from NCBI Taxonomy. Each data point represents a signature sequence, generated by one of the tools using genomes under a pathogenic taxon as targets (e.g., *C. difficile*), and genomes under its neighboring taxa as non-targets. Blue, orange and green represent Seqwin, Fur and Unikseq, respectively. The number of output signatures (data points) in each setting is shown in each single scatter plot. 100 or 1,000 genomes are sampled from each target and non-target group, and provided as inputs to the three tools.

## Discussion

We demonstrate that Seqwin represents a significant advance in scalable microbial genome signature discovery through benchmarking experiments. Seqwin identified genomic signatures in tens of thousands of genomes, using sets of target and non-target genomes at input. By coupling a minimizer-graph strategy with tolerance for sequence variation and genomic diversity, Seqwin bypasses strict search criteria and high memory usage, critical limitations of previous methods. Through our experimental evaluation, we show Seqwin consistently generated higher signature sensitivity and specificity as compared to other approaches. On a benchmark of 33 diverse *E. coli* genomes, Seqwin identified more target-specific signature sequences than Fur and Unikseq while using less CPU and memory. Similarly, across large-scale datasets comprising hundreds of

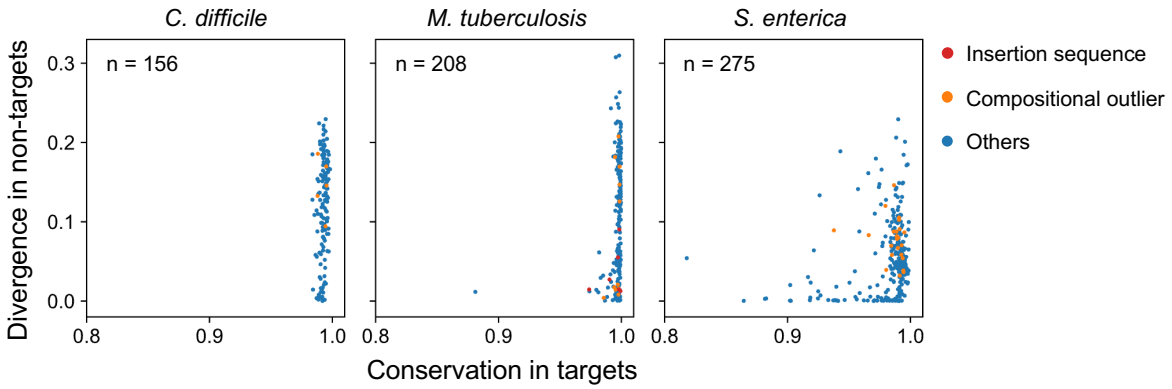


**Figure 4.** Output signatures of Seqwin using all genomes downloaded from NCBI Taxonomy. Each data point represents a signature sequence, generated by Seqwin using genomes under a pathogenic taxon as targets (e.g., *C. difficile*), and genomes under its neighboring taxa as non-targets. The number of output signatures (data points) in each setting is shown in each single scatter plot. Signatures overlapping with potential MGEs are labeled in red and orange, and others are labeled in blue. Number of orange data points (left to right): 5, 11 and 23. Number of red data points (left to right): 0, 6 and 0.

*C. difficile*, *M. tuberculosis*, and *S. enterica* genomes, Seqwin recovered a greater number of high-quality signatures with substantially lower computational cost than competing methods. Notably, Seqwin was capable of processing around 15,000 *S. enterica* genomes in under 10 minutes, and identified over 200 robust candidate signatures. These results underscore Seqwin's exceptional scalability and ability to efficiently uncover highly sensitive and specific genomic signatures in both small and massive comparative analyses. Although wet-lab validation was not performed in this study, we envision Seqwin to be coupled with established primer and probe design software, such as Olivar[40], varVAMP[41], PrimalScheme3[42], and Primer3[43], enabling sensitive and specific PCR-based assay designs. Seqwin could also be integrated into end-to-end assay-design pipelines, supporting applications in clinical and public health surveillance.

However, there remain several open problems we leave for future work. First, the seeded greedy BFS strategy used to extract low-penalty subgraphs can lead to an imbalance in signature lengths: early-extracted subgraphs tend to grow longer, whereas those found later remain shorter. One way to mitigate this would be to extend all candidate seed subgraphs in parallel, followed by the merging of adjacent low-penalty subgraphs. Second, Seqwin is implemented in Python with heavy use of vectorization (NumPy[44]) and just-in-time compilation (Numba[45]), which, while convenient, cannot fully match the speed of lower-level languages. Reimplementation in a compiled language (e.g., C++ or Rust) would improve Seqwin performance even further. For instance, a directed multigraph representation might preserve the full minimizer ordering within each signature for greater accuracy, but an initial NetworkX-based[46] prototype proved too slow, highlighting the need for a more efficient graph backend. Third, Seqwin currently requires manual specification of target and non-target genome groups. Automating this step by leveraging taxonomic databases like NCBI Taxonomy[33] could streamline signature discovery, similar to the phylogeny-guided algorithm used in NAUniSeq[27]. However, for complex applications (e.g., designing assays to detect antimicrobial-resistant pathogens across multiple lineages), using a pure taxonomy-based grouping may fail to capture all relevant variation. Thus, expert curation of target and non-target groups may remain necessary to define the search space appropriately.

In summary, inspired by pioneering approaches developed two decades ago in Insignia, Seqwin represents a highly sensitive and discriminative computational approach for microbial genome signature discovery. We anticipate Seqwin will facilitate signature search at a scale not previously possible, which in turn will facilitate automated, sensitive and specific PCR-based assay designs that are widely used for clinical and environmental applications.

# Methods

## Generation of minimizer sketch

Seqwin first computes minimizers[47,48] for each input genome with btllib[49] (version 1.7.3, with $k$-mer length of 21 and window size of 200 by default), including target genomes and non-target genomes. Each genome may contain one or more sequence records, with unknown orientation (strand). For each sequence record (sequence) in a single genome, a set of minimizers are chosen as a subset of all $k$-mers in the sequence, representing a compressed sketch of the sequence. Here, we describe the process of generating a minimizer sketch for a sequence (Algorithm S1). First, the canonical hash value of each $k$-mer in the

sequence is calculated (Equation (1))[50]. Formally,

$$\textsc{CanHash}(s) = \begin{cases} \textsc{Hash}(s) & (s \leq s_{rc}) \\ \textsc{Hash}(s_{rc}) & (s > s_{rc}) \end{cases} \tag{1}$$

where $s$ is the $k$-mer sequence, $s_{rc}$ is the reverse complement of $s$, HASH is a hash function, and $k$-mer comparison is based on lexicographical order. Hereafter, $k$-mers or minimizers are always represented by their canonical hash value, unless stated explicitly. Next, for each window of $w$ consecutive $k$-mers, the location of the $k$-mer with the smallest canonical hash value is selected, breaking ties by preferring the leftmost $k$-mer. The $k$-mer at this location is the minimizer of this window. The minimizer sketch of the sequence is the union of $k$-mers and their locations selected in each window. It is crucial that each minimizer is paired with its location in the sequence, since the same $k$-mer might be found at different locations. The minimizer sketch of a genome is the union of the sketches of each of its sequences. It should be noted that the minimizer sketch could be replaced with other $k$-mer sketching methods[51,52], as long as the "local guarantee" is held[50].

Note that the algorithms described in this manuscript (Algorithms S1 to S6) are not necessarily the most efficient implementations, but they have the same behavior as the Seqwin source code (version 0.2.0)[53].

## Construction of a weighted pan-genome minimizer graph

Construction of a weighted pan-genome minimizer graph is described in Algorithm S2. For each genome, we construct an undirected graph based on its minimizers, where each node corresponds to a unique minimizer (identified by its canonical hash value), and an edge is added between two nodes if the minimizers are adjacent (regardless of their ordering) in one of the genome's sequences.

Next, the individual genome graphs are merged into a single pan-genome minimizer graph. In this merged graph, nodes represent distinct minimizers observed in any genome, and an undirected edge connects two minimizers if they are found adjacent in at least one genome. The edges are assigned a weight equal to the number of different genomes in which that minimizer adjacency occurs. In other words, if two specific minimizers appear consecutively (no matter the ordering) in the sequences of multiple genomes, the edge between their nodes is given a higher weight (reflecting the number of genomes supporting that adjacency). This results in a unified weighted graph capturing adjacency relationships of minimizers across all genomes.

## Calculation of node penalty

Since each node in the pan-genome graph represents a distinct minimizer observed in one or more genomes, suppose a minimizer node $h$ is found in $\mathrm{F_t}(h)$ target genomes and $\mathrm{F_n}(h)$ non-target genomes. Let

$$\mathrm{f_t}(h) = \frac{\mathrm{F_t}(h)}{N_{\text{target}}},\ \mathrm{f_n}(h) = \frac{\mathrm{F_n}(h)}{N_{\text{non-target}}} \tag{2}$$

where $N_{\text{target}}$ is the total number of target genomes, and $N_{\text{non-target}}$ is the total number of non-target genomes. The penalty of $h$ is defined as

$$\mathrm{p}(h) = \sqrt{(1-\mathrm{f_t}(h))^2 + \mathrm{f_n}^2(h)} \tag{3}$$

which is the L2 norm (Euclidean norm) of its absence in targets: $1-\mathrm{f_t}(h)$, and presence in non-targets: $\mathrm{f_n}(h)$. Penalty ranges from 0 (best case: the minimizer is present in all target genomes and in no non-target genomes) to $\sqrt{2}$ (worst case: the minimizer is absent from all targets and present in all non-targets). Thus, a lower penalty indicates a minimizer being more specific to the target taxon.

A node penalty threshold $\tau_v$ will be used in downstream processes, and the output signatures of Seqwin are mostly derived from low-penalty nodes ($\mathrm{p}(h) \leq \tau_v$), as described in sections below. $\tau_v$ can be determined by the user or automatically computed by Seqwin, which is described in the following section.

## Calculation of penalty threshold

Intuitively, $\tau_v$ should be determined by the expected $k$-mer absence and presence in target and non-target genomes, respectively. Specifically, consider a random $k$-mer $h$ (not necessarily a minimizer) sampled from a random target genome (select the target genome first and then select the $k$-mer). $1-\mathrm{f_t}(h)$ is the fraction of target genomes that do not include $h$ (absence), and $\mathrm{f_n}(h)$ is the fraction of non-target genomes that include $h$ (presence), as defined in Equation (2). $\tau_v$ is then calculated with the expectations of these fractions

$$\tau_v = \alpha_v \cdot \sqrt{(1-\mathbb{E}[\mathrm{f_t}(h)]) \cdot \mathbb{E}[\mathrm{f_n}(h)]} \tag{4}$$

which is the geometric mean of expected $k$-mer absence and presence, times a constant $\alpha_v$. In practice, $\alpha_v$ defaults to 0.5 and can be set by a parameter of Seqwin (`--stringency`). We use the geometric mean so that $\tau_v$ will bias toward the smaller value of the two terms, resulting in a more stringent threshold.

Consider genomes as sets of $k$-mers (duplicated $k$-mers in each genome are ignored). Then the expectations can be calculated with the Jaccard indices between genome pairs (proof can be found in Supplementary Note 3). Since the Jaccard indices can be effectively estimated with MinHash sketches[54] using Mash[34], $\tau_v$ can be calculated by running Mash on all input genomes.

Another way of estimating the expected values is using minimizer sketches, instead of MinHash sketches. This approach could be much faster in practice since Seqwin already generates minimizer sketches for all input genomes. As previously reported, minimizer sketches on average underestimate the fraction of shared $k$-mers between two sequences[55], resulting in biased estimates of the expectations. However, since this bias leads to overestimation of $k$-mer absence in targets ($1-\mathbb{E}[\mathrm{f_t}(h)]$) and underestimation of $k$-mer presence in non-targets ($\mathbb{E}[\mathrm{f_n}(h)]$), the bias of the resulted geometric mean is smaller (Equation (4)). Therefore, this is a pragmatic way of estimating $\tau_v$, especially when the number of input genomes is large and calculating pairwise Jaccard with Mash is costly. Seqwin implements this method as a faster alternative for calculating $\tau_v$, along with the unbiased Mash implementation.

## Filtering of the minimizer graph

To simplify the graph and remove defects caused by assembly errors or low-quality regions in the input genomes, low-weight edges are pruned based on a dynamic edge weight threshold. Since Seqwin focuses on minimizer nodes with penalties below $\tau_v$, from Equations (2) and (3) we have

$$\begin{aligned}\mathrm{F}(h) = \mathrm{F_t}(h) + \mathrm{F_n}(h) &\geq (1-\mathrm{p}(h)) \cdot N_{\text{target}} \\ &\geq (1-\tau_v) \cdot N_{\text{target}}, \text{ if } \mathrm{p}(h) \leq \tau_v \end{aligned} \tag{5}$$

where $\mathrm{F}(h)$ is the number of input genomes containing a certain minimizer node $h$. Since the weight of any edge incident to $h$ cannot exceed $\mathrm{F}(h)$ (an edge's weight is limited by the least frequent of its two nodes), edge weight threshold is defined as

$$\tau_e = \alpha_e \cdot (1-\tau_v) \cdot N_{\text{target}} \tag{6}$$

where $\alpha_e$ is a small constant (0.3 by default). The value of $\alpha_e$ is arbitrary and should not significantly affect the outputs, since Seqwin focuses on low-penalty nodes whose edges weights are usually much larger than $\tau_e$. We prune any edges with weight less than $\tau_e$. After pruning low-weight edges, isolated nodes with degree of zero are also removed from the graph.

## Extraction of low-penalty subgraphs

Disjoint low-penalty subgraphs are extracted from the filtered minimizer graph (Algorithm S3). Each subgraph is a set of connected minimizer nodes whose average node penalty (Equation (3)) does not exceed the threshold $\tau_v$. The procedure first identifies all candidate seed nodes with penalty $\leq \tau_v$. Starting from each seed, a greedy breadth-first search (BFS) expansion is performed by iteratively adding the adjacent node with the lowest penalty, as long as including that node keeps the subgraph's average penalty below $\tau_v$. This expansion continues until no more neighboring nodes can be added or the subgraph reaches a specified maximum size (100 by default). Any subgraph that meets a minimum size requirement (3 by default) is retained, and its nodes are marked as used to ensure that subgraphs remain disjoint (no node is part of more than one subgraph). Seeds are processed in random order, and the final list of subgraphs are be shuffled to balance subgraph sizes in downstream processes (subgraphs created first tend to be larger due to greedy optimization). Subgraphs with duplicate minimizers are discarded.

Since the expected "density" of a random minimizer sketch is $2/(w+1)$ (in most conditions)[50], where $w$ is window size, the minimum and maximum size requirements of low-penalty subgraphs can be calculated from the length requirements of signatures. For example, for a default $w$ of 200 bp and a $k$ much smaller than $w$ (e.g., 21 bp), the expected density is around 1 minimizer per 100 bp. Since Seqwin has a default minimum signature length of 200 bp, the default minimum size of low-penalty subgraphs is calculated as 3, as mentioned in the previous paragraph. Seqwin does not have a specific default maximum signature length, thus the default maximum size of low-penalty subgraphs is set to 100, which could yield signatures with lengths up to 10,000 bp under default settings.

## Choosing a representative sequence for each low-penalty subgraph

For each low-penalty subgraph (a set of connected minimizer nodes from the graph), a representative minimizer ordering is determined and the corresponding genomic sequence is output as a candidate signature. The procedure involves finding the maximal consecutive occurrence of the subgraph's minimizers in the minimizer sketch of each target genome, and then choosing the most prevalent minimizer ordering across all target genomes (Algorithm S4 and Figure 1d). A more detailed illustration of this process can be found in Figure S1.

In each target genome that contains one or more minimizers from a given subgraph, we identify the longest segment of those minimizers that appear consecutively in the genome's sequence (allowing for at most one intervening minimizer not in the subgraph). Note that there could repetitive segments in a single genome. Two minimizers are defined as consecutive if 1) they appear in the same sequence of that genome, 2) the difference between their indices in the minimizer sketch is less than or equal to 2. This yields, for each genome, an ordered tuple of minimizer hashes representing that subgraph's segment in the genome.

Among all target genomes, the most prevalent minimizer ordering (treating forward and reverse ordering as equivalent) is then selected as the subgraph's representative sequence. Prevalence is measured by the number of target genomes in which a given ordering occurs, weighted by the length (number of minimizers) of the ordering to favor longer sequences. The orientation of the representative ordering is chosen to match the strand orientation that is more commonly observed for that sequence in the target genomes. One of the genomes supporting this representative minimizer order is used to determine its actual genomic sequence (the representative sequence), based on the minimizer coordinates in that genome. Representative sequences that satisfy user-defined length thresholds ($\geq$ 200 bp by default) are output as candidate signatures.

### Evaluation of candidate signatures

All output signatures are BLAST[30] (version 2.16.0) checked against all input genomes. For each signature, its best BLAST alignment (highest `bitscore`) in each input genome is kept. The BLAST command and its arguments are shown below

```
blastn -task blastn -max_hsps 1000 -max_target_seqs 50000
```

with `max_hsps` and `max_target_seqs` set to large numbers to keep all BLAST alignments.

Based on the BLAST alignments, we define two metrics to quantify sensitivity and specificity: conservation and divergence. Conservation measures how consistently the signature is conserved among target genomes, while divergence measures how dissimilar it is in non-target genomes. Formally, let the signature length be $L$. For all target genomes, we sum the number of identical bases (`nident` in BLAST outputs) in each BLAST alignment; let this sum be $I_{\text{target}}$. We define conservation as the average identity fraction in targets:

$$\text{conservation} = \frac{I_{\text{target}}}{L \cdot N_{\text{target}}} \tag{7}$$

Presumably, if a target genome has no BLAST alignment for the signature, it contributes 0 identical base (thus lowering conservation). Thus, conservation = 1 would mean the signature is identical in all target genomes, whereas a lower value indicates some targets have mismatches in the alignment (or have no alignment).

Similarly, for all non-target genomes, we sum the number of nucleotide differences (`mismatch` and `gaps` in BLAST outputs) in each BLAST alignment. Let this sum be $D_{\text{non-target}}$. We define divergence as the average fraction of differences in non-targets:

$$\text{divergence} = \frac{D_{\text{non-target}}}{L \cdot N_{\text{non-target}}} \tag{8}$$

It is important to note that the sum $D_{\text{non-target}}$ is only accumulated when there is a BLAST alignment. For a non-target genome with no BLAST alignment to the signature, it would have zero contribution to the sum. Or if the BLAST alignment is partial (e.g., only the first half of the signature is aligned), the flanking regions also have zero contribution to the sum. This way, a high divergence score indicates that the signature, while present in non-targets, has many differences, rather than being completely absent. This reduces the chance of picking up MGEs that might be completely absent in some non-target genomes. For each signature, the fraction of non-target genomes with a BLAST alignment can be found in Figures S2 to S4.

Finally, we compute a total score for each signature as the sum of its conservation and divergence. Seqwin uses this score to sort the output signatures so that those appearing first would be the top candidates.

## References


1. Eisenach, K. D., Donald Cave, M., Bates, J. H. & Crawford, J. T. Polymerase chain reaction amplification of a repetitive DNA sequence specific for Mycobacterium tuberculosis. *J. Infect. Dis.* **161**, 977–981, DOI: 10.1093/infdis/161.5.977 (1990).
2. Laure, F. *et al.* Detection of HTV1 DNA in infants and children by means of the polymerase chain reaction. *The Lancet* **332**, 538–541, DOI: 10.1016/s0140-6736(88)92659-1 (1988).
3. Slezak, T. *et al.* Comparative genomics tools applied to bioterrorism defence. *Briefings bioinformatics* **4**, 133–149, DOI: 10.1093/bib/4.2.133 (2003).
4. Albuquerque, P., Mendes, M. V., Santos, C. L., Moradas-Ferreira, P. & Tavares, F. DNA signature-based approaches for bacterial detection and identification. *Sci. Total. Environ.* **407**, 3641–3651, DOI: 10.1016/j.scitotenv.2008.10.054 (2009).

5. Segata, N. *et al.* Metagenomic microbial community profiling using unique clade-specific marker genes. *Nat. methods* **9**, 811–814, DOI: 10.1038/nmeth.2066 (2012).

6. Wu, D., Jospin, G. & Eisen, J. A. Systematic identification of gene families for use as "markers" for phylogenetic and phylogeny-driven ecological studies of bacteria and archaea and their major subgroups. *PloS one* **8**, e77033, DOI: 10.1371/journal.pone.0077033 (2013).

7. Heid, C. A., Stevens, J., Livak, K. J. & Williams, P. M. Real time quantitative PCR. *Genome research* **6**, 986–994, DOI: 10.1101/gr.6.10.986 (1996).

8. Schena, M., Shalon, D., Davis, R. W. & Brown, P. O. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science* **270**, 467–470, DOI: 10.1126/science.270.5235.467 (1995).

9. Hofstadler, S. A. *et al.* TIGER: the universal biosensor. *Int. J. Mass Spectrom.* **242**, 23–41, DOI: 10.1016/j.ijms.2004.09.014 (2005).

10. Nordberg, E. K. YODA: selecting signature oligonucleotides. *Bioinformatics* **21**, 1365–1370, DOI: 10.1093/bioinformatics/bti182 (2005).

11. Chung, W.-H. *et al.* Design of long oligonucleotide probes for functional gene detection in a microbial community. *Bioinformatics* **21**, 4092–4100, DOI: 10.1093/bioinformatics/bti673 (2005).

12. Feng, S. & Tillier, E. R. A fast and flexible approach to oligonucleotide probe design for genomes and gene families. *Bioinformatics* **23**, 1195–1202, DOI: 10.1093/bioinformatics/btm114 (2007).

13. Phillippy, A. M. *et al.* Comprehensive DNA signature discovery and validation. *PLoS computational biology* **3**, e98, DOI: 10.1371/journal.pcbi.0030098 (2007).

14. Phillippy, A. M., Ayanbule, K., Edwards, N. J. & Salzberg, S. L. Insignia: a DNA signature search web server for diagnostic assay development. *Nucleic acids research* **37**, W229–W234, DOI: 10.1093/nar/gkp286 (2009).

15. Zahariev, M., Dahl, V., Chen, W. & Lévesque, C. Efficient algorithms for the discovery of DNA oligonucleotide barcodes from sequence databases. *Mol. ecology resources* **9**, 58–64, DOI: 10.1111/j.1755-0998.2009.02651.x (2009).

16. Lee, H. P., Sheu, T.-F. & Tang, C. Y. A parallel and incremental algorithm for efficient unique signature discovery on DNA databases. *BMC bioinformatics* **11**, 1–13, DOI: 10.1186/1471-2105-11-132 (2010).

17. Bader, K. C., Grothoff, C. & Meier, H. Comprehensive and relaxed search for oligonucleotide signatures in hierarchically clustered sequence datasets. *Bioinformatics* **27**, 1546–1554, DOI: 10.1093/bioinformatics/btr161 (2011).

18. Lundberg, D. S., Yourstone, S., Mieczkowski, P., Jones, C. D. & Dangl, J. L. Practical innovations for high-throughput amplicon sequencing. *Nat. methods* **10**, 999–1002, DOI: 10.1038/nmeth.2634 (2013).

19. Hindson, B. J. *et al.* High-throughput droplet digital PCR system for absolute quantitation of DNA copy number. *Anal. chemistry* **83**, 8604–8610, DOI: 10.1021/ac202028g (2011).

20. Knowles, M., Lambert, D., Huszczynski, G., Gauthier, M. & Blais, B. W. PCR for the specific detection of an Escherichia coli O157: H7 laboratory control strain. *J. Food Prot.* **78**, 1738–1744, DOI: 10.4315/0362-028x.jfp-15-147 (2015).

21. Haubold, B., Klötzl, F., Hellberg, L., Thompson, D. & Cavalar, M. Fur: Find unique genomic regions for diagnostic PCR. *Bioinformatics* **37**, 2081–2087, DOI: 10.1093/bioinformatics/btab059 (2021).

22. Vieira Mourato, B., Tsers, I., Denker, S., Klötzl, F. & Haubold, B. Marker discovery in the large. *Bioinforma. advances* **4**, vbae113, DOI: 10.1093/bioadv/vbae113 (2024).

23. Wang, Y., Chen, Q., Deng, C., Zheng, Y. & Sun, F. KmerGO: a tool to identify group-specific sequences with k-mers. *Front. microbiology* **11**, 2067, DOI: 10.3389/fmicb.2020.02067 (2020).

24. Beran, P., Stehlíková, D., Cohen, S. P. & Čurn, V. KEC: unique sequence search by k-mer exclusion. *Bioinformatics* **37**, 3349–3350, DOI: 10.1093/bioinformatics/btab196 (2021).

25. Allison, M. J. *et al.* Enabling robust environmental DNA assay design with "unikseq" for the identification of taxon-specific regions within whole mitochondrial genomes. *Environ. DNA* **5**, 1032–1047, DOI: 10.1002/edn3.438 (2023).

26. Lopez, M. L. D. *et al.* Conserved Sequence Identification Within Large Genomic Datasets Using 'Unikseq2': Application in Environmental DNA Assay Development. *Mol. Ecol. Resour.* **25**, e70014, DOI: 10.1111/1755-0998.70014 (2025).

27. Sharma, G. K. *et al.* Advancing microbial diagnostics: a universal phylogeny guided computational algorithm to find unique sequences for precise microorganism detection. *Briefings Bioinforma.* **25**, bbae545, DOI: 10.1093/bib/bbae545 (2024).

**28.** Coombe, L., Nikolić, V., Chu, J., Birol, I. & Warren, R. L. ntJoin: Fast and lightweight assembly-guided scaffolding using minimizer graphs. *Bioinformatics* **36**, 3885–3887, DOI: 10.1093/bioinformatics/btaa253 (2020).

**29.** Coombe, L., Kazemi, P., Wong, J., Birol, I. & Warren, R. L. Multi-genome synteny detection using minimizer graph mappings. *bioRxiv* 2024–02, DOI: 10.1101/2024.02.07.579356 (2024).

**30.** Camacho, C. *et al.* BLAST+: architecture and applications. *BMC bioinformatics* **10**, 1–9, DOI: 10.1186/1471-2105-10-421 (2009).

**31.** Pirogov, A., Pfaffelhuber, P., Börsch-Haubold, A. & Haubold, B. High-complexity regions in mammalian genomes are enriched for developmental genes. *Bioinformatics* **35**, 1813–1819, DOI: 10.1093/bioinformatics/btab639 (2019).

**32.** Klötzl, F. & Haubold, B. Phylonium: fast estimation of evolutionary distances from large samples of similar genomes. *Bioinformatics* **36**, 2040–2046, DOI: 10.1093/bioinformatics/btz903 (2020).

**33.** Schoch, C. L. *et al.* NCBI Taxonomy: a comprehensive update on curation, resources and tools. *Database* **2020**, baaa062, DOI: 10.1093/database/baaa062 (2020).

**34.** Ondov, B. D. *et al.* Mash: fast genome and metagenome distance estimation using MinHash. *Genome biology* **17**, 1–14, DOI: 10.1186/s13059-016-0997-x (2016).

**35.** Afshinnekoo, E. *et al.* Geospatial resolution of human and bacterial diversity with city-scale metagenomics. *Cell systems* **1**, 72–87, DOI: 10.1016/j.cels.2015.01.001 (2015).

**36.** Goldfarb, T. *et al.* NCBI RefSeq: reference sequence standards through 25 years of curation and annotation. *Nucleic Acids Res.* **53**, D243–D257, DOI: 10.1093/nar/gkae1038 (2025).

**37.** Wang, M. X., Kille, B., Nute, M. G. & Treangen, T. J. Seqwin: Ultrafast identification of signature sequences in microbial genomes via minimizer graphs. *Figshare* DOI: 10.6084/m9.figshare.30311311.v1 (2025).

**38.** Cantalapiedra, C. P., Hernández-Plaza, A., Letunic, I., Bork, P. & Huerta-Cepas, J. eggNOG-mapper v2: functional annotation, orthology assignments, and domain prediction at the metagenomic scale. *Mol. biology evolution* **38**, 5825–5829, DOI: 10.1093/molbev/msab293 (2021).

**39.** Richardson, L. *et al.* MGnify: the microbiome sequence data analysis resource in 2023. *Nucleic acids research* **51**, D753–D759, DOI: 10.1093/nar/gkac1080 (2023).

**40.** Wang, M. X. *et al.* Olivar: towards automated variant aware primer design for multiplex tiled amplicon sequencing of pathogens. *Nat. Commun.* **15**, 6306, DOI: 10.1038/s41467-024-49957-9 (2024).

**41.** Fuchs, J. *et al.* varVAMP: degenerate primer design for tiled full genome sequencing and qPCR. *Nat. Commun.* **16**, 5067, DOI: 10.1101/2024.05.08.593102 (2025).

**42.** Kent, C. *et al.* PrimalScheme: open-source community resources for low-cost viral genome sequencing. *bioRxiv* 2024–12, DOI: 10.1101/2024.12.20.629611 (2024).

**43.** Untergasser, A. *et al.* Primer3—new capabilities and interfaces. *Nucleic acids research* **40**, e115–e115, DOI: 10.1093/nar/gks596 (2012).

**44.** Harris, C. R. *et al.* Array programming with NumPy. *nature* **585**, 357–362, DOI: 10.1038/s41586-020-2649-2 (2020).

**45.** Lam, S. K., Pitrou, A. & Seibert, S. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 1–6, DOI: 10.1145/2833157.2833162 (2015).

**46.** Hagberg, A., Swart, P. J. & Schult, D. A. Exploring network structure, dynamics, and function using NetworkX. Tech. Rep., Los Alamos National Laboratory (LANL), Los Alamos, NM (United States) (2008). DOI: 10.25080/tcwv9851.

**47.** Roberts, M., Hayes, W., Hunt, B. R., Mount, S. M. & Yorke, J. A. Reducing storage requirements for biological sequence comparison. *Bioinformatics* **20**, 3363–3369, DOI: 10.1093/bioinformatics/bth408 (2004).

**48.** Schleimer, S., Wilkerson, D. S. & Aiken, A. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 76–85, DOI: 10.1145/872757.872770 (2003).

**49.** Nikolić, V. *et al.* btllib: A C++ library with Python interface for efficient genomic sequence processing. *J. Open Source Softw.* **7**, 4720, DOI: 10.21105/joss.04720 (2022).

**50.** Zheng, H., Marçais, G. & Kingsford, C. Creating and using minimizer sketches in computational genomics. *J. Comput. Biol.* **30**, 1251–1276, DOI: 10.1089/cmb.2023.0094 (2023).

**51.** Edgar, R. Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. *PeerJ* **9**, e10805, DOI: 10.7717/peerj.10805 (2021).

**52.** Kille, B., Garrison, E., Treangen, T. J. & Phillippy, A. M. Minmers are a generalization of minimizers that enable unbiased local Jaccard estimation. *Bioinformatics* **39**, btad512, DOI: 10.1093/bioinformatics/btad512 (2023).

**53.** Wang, M. X., Kille, B., Nute, M. G. & Treangen, T. J. Seqwin: Ultrafast identification of signature sequences in microbial genomes via minimizer graphs. *Zenodo* DOI: 10.5281/zenodo.17459714 (2025).

**54.** Broder, A. Z. On the resemblance and containment of documents. In *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*, 21–29, DOI: 10.1109/SEQUEN.1997.666900 (IEEE, 1997).

**55.** Belbasi, M., Blanca, A., Harris, R. S., Koslicki, D. & Medvedev, P. The minimizer Jaccard estimator is biased and inconsistent. *Bioinformatics* **38**, i169–i176, DOI: 10.1101/2022.01.14.476226 (2022).

## Acknowledgements

The authors thank Dr. Adam Phillippy for valuable feedback and suggestions. This work has been supported in part by NIH grants R21-AI190938 and P01-AI152999, NSF awards IIS-2239114 and EF-2126387, and the National Library of Medicine Training Program in Biomedical Informatics and Data Science [T15LM007093 to B.K.].

## Author contributions statement

M.X.W., B.K., L.B.S. and T.J.T conceived the project, M.X.W., M.G.N, S.Z. and T.J.T conceived the experiments. M.X.W. developed the code and conducted the experiment. M.X.W. and T.J.T drafted the original manuscript. All authors reviewed the manuscript.

## Data Availability

NCBI accessions and metadata of all genomes used in this study can be found in Supplementary Table. All signature sequences generated in this work are available on Figshare `https://doi.org/10.6084/m9.figshare.30311311.v1`[37].

## Code Availability

Source code, installation guide and usage of Seqwin are available on GitHub: `https://github.com/treangenlab/Seqwin`. Signature sequences generated in this study can be reproduced with Seqwin version 0.2.0[53], available on Zenodo: `https://doi.org/10.5281/zenodo.17459714`.

## Competing interests

M.X.W, M.G.N, and T.J.T are co-inventors on a provisional patent application that includes algorithms described in this manuscript. The remaining authors declare no competing interests.