Journal Title Here, YEAR, pp. 1–9
DOI added during production
Published: Date added during production
Paper

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, 6100 Main St, 77005, TX, United States
[2]Department of Civil and Environmental Engineering, Rice University, 6100 Main St, 77005, TX, United States
[3]Department of Bioengineering, Rice University, 6100 Main St, 77030, TX, United States
[4]Ken Kennedy Institute, Rice University, 6100 Main St, 77005, TX, United States

[*]Corresponding author. 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 detection 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 for a single species, 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 signature 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's analysis of nearly 15,000 S. enterica genomes yielded over 200 candidate signatures in 5 minutes. Seqwin provides an open-source solution for the long-standing need for scalable microbial signature discovery and diagnostic assay design.

Availability and Implementation: Seqwin is freely available for academic use (`https://github.com/treangenlab/Seqwin`) and can be installed via Bioconda. Benchmarking datasets, outputs, and scripts are available on Zenodo `https://doi.org/10.5281/zenodo.19176444`.

Contact: treangen@rice.edu, xw66@rice.edu

Supplementary Materials: Provided as separate PDF and data files.

Keywords signature sequences, minimizer graph, pathogen detection

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]. Clinical studies in the late 1980s, such as the detection of HIV-1 DNA in infants and adults [2], first demonstrated its utility in patient diagnosis. These early studies laid the foundation for PCR-based infectious disease diagnostics. Signature sequences are genomic regions that allow accurate identification and classification of microorganisms within specific taxonomic groups. DNA signature sequences were originally defined to be sequences both 100% conserved over all target genomes and unique compared to all non-target genomes [3]. However, over time the definition has been relaxed in signature detection tools and is now more commonly defined as a sequence that is highly sensitive (largely present in most target genomes), and highly specific (largely absent in or divergent to closely related non-target genomes).

Early computational methods for signature discovery mainly focused on microarray and TaqMan assay design, and they usually required signatures to be perfectly conserved in all target genomes [4, 5, 6, 7, 8]. Approaches such as YODA [4] and ProDesign [5] use exhaustive searches combined with various filtering criteria, while Insignia [6, 7] and CaSSiS [8] leverage maximal unique matches (MUMs) identified via a suffix tree data structure. While all of these methods were important advances in microbial signature discovery, they suffer from several limitations. First, tools developed for the design of microarray probes typically produce short signatures ($\leq$50 bp) [4, 5], which are often insufficient for modern targeted sequencing assays. Second, many of these methods were developed before the NGS era, when genomic data was considerably limited. When Insignia pioneered open-source microbial signature discovery, only

Received: Date Month Year. **Revised received:** Date Month Year. **Accepted:** Date Month Year
© The Author(s) YEAR. Copyright and licence statements to be updated by the publisher during production.

tens to hundreds of genomes were available for a given microbial target. Consequently, these tools were not designed to handle modern genomic databases at terabyte-to-petabyte scales. Furthermore, due to their reliance on exact matches across all target genomes, these methods are highly vulnerable 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 [9] relies on BLAST-based [10] genome subtraction methods, 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 abandon all-versus-all sequence alignment in favor of ultrafast k-mer-based strategies or clever filtering techniques. Fur [11, 12] combines targeted genomic subtraction and stringent intersection methods to discover unique genomic regions. However, it struggles with highly divergent target genomes and/or similar background genomes, and may produce few or no candidate signatures. Methods such as Neptune [13], KmerGO [14], KEC [15] and Unikseq [16, 17] use alignment-free k-mer filtering and subsequently combine k-mers into longer signatures through sequence assembly or similar strategies. However, they all require substantial memory to store k-mers from all input genomes. NAUniSeq [18] is a recent k-mer-based method that automates the selection of target and non-target taxa, reporting identified k-mers directly as signatures without further extension. As a result, it typically requires a larger k (e.g., 100 bp), and all output signatures are fixed at that length. CURED [19] uses a similar k-mer filtering strategy and reports restriction sites on the output k-mers. While some of these tools do not explicitly use the term "signature", they computationally address the same problem using target and non-target genome groups as input. Table 1 summarizes these tools, including their publication years and approaches. 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 flexible algorithm for genomic signature discovery using pan-genome minimizer graphs. Unlike previous minimizer graph-based methods that require each minimizer to be present in all input genomes [20, 21], Seqwin allows the inclusion of minimizers from any of the 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 memory-efficient workflow. 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 signatures. In related work, mdBG constructs a minimizer-space de Bruijn graph for long-read assembly [22]; in contrast, Seqwin builds minimizer adjacency graphs (edges reflect consecutive minimizers along genomes) for comparative analyses.

Due to its efficient and flexible design, Seqwin performs well at identifying signatures from large collections of microbial genomes despite sequence variations, a critically important capability given the heterogeneity and rapid expansion of contemporary genomic databases. This allows Seqwin to operate effectively across diverse contexts, including wastewater monitoring and low-microbial-biomass clinical settings, while minimizing the impact of incomplete or low-quality genome assemblies. As a result, Seqwin provides a scalable and computationally efficient framework for microbial pathogen detection, enabling the rapid identification of genomic signatures for a wide range of downstream applications.

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

Name	Year [1]	Approach
Insignia [6, 7]	2009	Suffix tree
SigSeekr [9]	2015	BLAST
Neptune [13]	2017	k-mer assembly
KmerGO [14]	2020	k-mer assembly
KEC [15]	2021	k-mer assembly
Fur [11, 12]	2024	Sequence filtering pipeline
NAUniSeq [18]	2024	k-mer filtering
Unikseq [16, 17]	2025	k-mer assembly
CURED [19]	2026	k-mer filtering
Seqwin	2026	Minimizer graph

[1] Year of the latest publication.

Methods

Seqwin's algorithm includes four main steps (Figure 1a). A more detailed description of each step can be found in Supplementary Note 1.

1. Generate a minimizer sketch for each input genome and build a weighted pan-genome minimizer graph (Figure 1b; a more constrained graph was described by Coombe et al. (2020) [20]).
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; Figure 1c).
4. Choose a representative sequence for each low-penalty subgraph as the signature (Figure 1d), and calculate its "conservation" (sensitivity) and "divergence" (specificity) scores by aligning the sequence to all target and non-target genomes with BLAST [10] (Supplementary Note 2).

Signatures with both high conservation and divergence are considered top candidates. Signatures with high divergence scores can still be found in non-target genomes, but only at low nucleotide identity. This reduces the chance of picking up mobile genetic elements (MGEs), which have been shown to be highly problematic as signature sequences [23] (see Supplementary Notes 1 and 2 for details).

Generation of minimizer sketches

Seqwin first computes minimizers [24] for all input genomes with btllib [25] (version 1.7.3, with k-mer length $k = 21$ and window size $w = 200$ by default), including target genomes and non-target genomes. Each genome may contain one or more contigs (sequences) of unknown orientation (strand). For each sequence in a genome, a set of minimizers is sampled from its k-mers to form the minimizer sketch of that sequence.

We set $k = 21$ by default to lower the collision rate for reliable signature detection, which is consistent with common genome sketching defaults for assembled bacterial genomes [26]. For viral

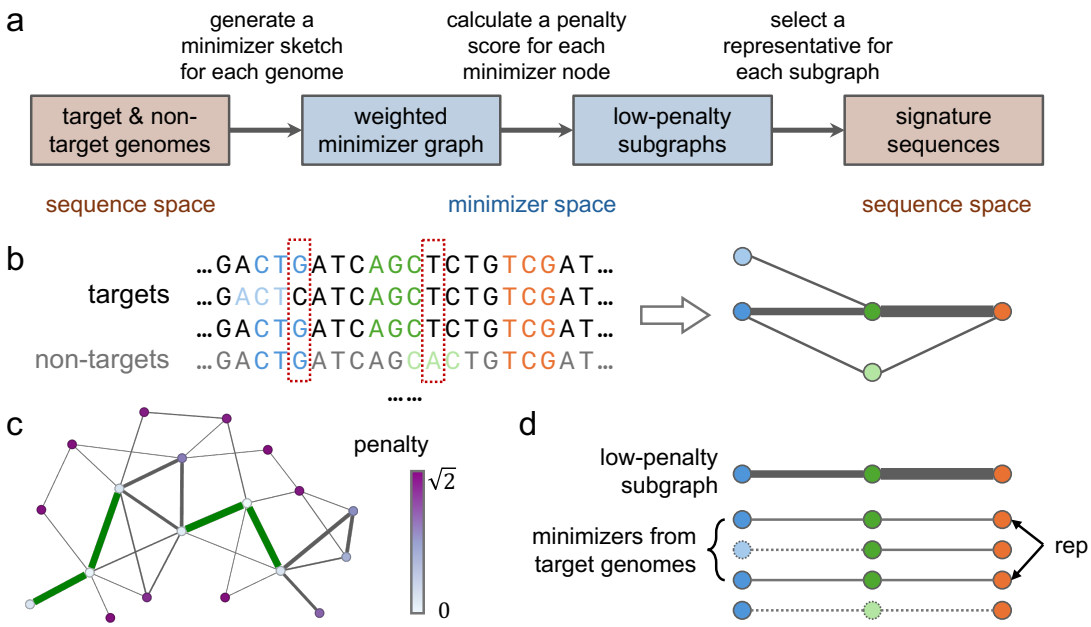

Figure 1 Overview of the Seqwin workflow. **a** Seqwin first compresses all input genomes into a weighted pan-genome minimizer graph. It then extracts connected low-penalty subgraphs and selects a representative for each subgraph in minimizer space. It finally converts each representative back to sequence space as the signature. **b** Graph construction (a toy example). Each unique minimizer is colored. Input genomes may yield different sets of minimizers (light blue and light orange) 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, light green and dashed lines are minimizers and edges not included in this subgraph). The most common minimizer ordering is selected as the representative (rep), and the corresponding genomic sequence becomes the signature (e.g., the first or third sequence in panel **b**).

genomes, a lower k might be preferable. The default window size ($w = 200$) is selected to maximize the number of signatures longer than 200 bp, which is sufficient for the design of most PCR-based assays (typical amplicon length ~100 bp). For a random minimizer sketch, the expected minimizer density is $2/(w+1)$ under standard assumptions (e.g., $k \ll w$) [27], implying an expected spacing of $(w+1)/2$ positions between consecutive minimizers and an expected span of $w+1$ for three consecutive minimizers (i.e., from the first to the third). Therefore, requiring at least three consecutive minimizers to represent a signature of at least 200 bp motivates $w = 200$ as a reasonable default. Throughout, we treat a run of three consecutive minimizers as the minimum evidence needed to call a signature. In practice, the choice of w reflects a trade-off between memory and resolution: increasing w reduces minimizer density and memory usage but coarsens the resolution for detecting signatures, whereas decreasing w increases density (and memory usage) and enables detection of shorter signatures, at the cost of representing long signatures with more consecutive minimizers and potentially reducing the number of long signatures reported.

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)) [27]. 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 and minimizers are represented by their canonical hash values unless stated otherwise. 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 the selected k-mers and their locations across all windows. 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. The minimizer sketch could also be replaced with other k-mer sketching methods [28, 29], as long as the "local guarantee" holds [27].

The algorithms described in this manuscript (Algorithm 1 and Algorithms S1 to S5) are not necessarily the most efficient implementations, but they have the same behavior as the Seqwin source code (version 0.2.3).

Construction of a weighted pan-genome minimizer graph

This process is described in Algorithm 1. 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

Algorithm 1 Construction of a weighted pan-genome minimizer graph

Require: Set of genomes $\mathcal{A} = \{A_1, \ldots, A_N\}$
Ensure: Weighted undirected graph $G = (V, E, \omega)$

```
V ← ∅, E ← ∅, ω ← empty map
for all A ∈ 𝒜 do
    V_A ← ∅, E_A ← ∅
    for all sequence S in A do
        M ← Sketch(S)  ▷ M is a list of tuples (h, ℓ) (minimizer hash and location, defined in Algorithm S1)
        if |M| < 2 then continue
        for i ← 1 to |M| − 1 do   ▷ minimizers are ordered by location
            h_1 ← M[i].h; h_2 ← M[i + 1].h
            if h_1 > h_2 then (h_1, h_2) ← (h_2, h_1)   ▷ undirected edge
            V_A ← V_A ∪ {h_1, h_2}
            E_A ← E_A ∪ {(h_1, h_2)}
        end for
    end for
    V ← V ∪ V_A
    for all e = (h_1, h_2) ∈ E_A do   ▷ calculate edge weights
        if e ∉ E then
            E ← E ∪ {e}; ω(e) ← 1
        else
            ω(e) ← ω(e) + 1
        end if
    end for
end for
return G = (V, E, ω)
```

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_{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 that a minimizer is more sensitive and specific to the target group.

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

Calculation of penalty threshold

Intuitively, τ_v should be determined by the expected k-mer absence and presence in target and non-target genomes, respectively. 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). τ_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, multiplied by a constant α_v. In practice, α_v defaults to 0.5 and can be tuned by the user to trade off the number of output signatures against their quality (e.g., increase α_v when more signatures are needed). We use the geometric mean so that τ_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 these expectations can be calculated from pairwise Jaccard indices (a proof can be found in Supplementary Note 3). Since the Jaccard indices can be effectively estimated with MinHash sketches using Mash [26], τ_v can be calculated by running Mash on all input genomes.

Another way to estimate the expected values is to use minimizer sketches instead of MinHash sketches. This can be faster because 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 [30], 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 resulting geometric mean is smaller (Equation (4)). Consequently, this provides a practical solution for estimating τ_v defined in Equation (4), particularly when processing large datasets where calculating pairwise Jaccard indices is computationally prohibitive. Seqwin implements this method as a faster alternative for calculating τ_v, along with the unbiased Mash implementation.

Extraction of low-penalty subgraphs

The minimizer graph is filtered to remove edges with negligible weights and isolated nodes (Supplementary Note 4). Disjoint low-penalty subgraphs are then extracted from the filtered graph (Algorithm S2). Each subgraph is a set of connected minimizer nodes whose average node penalty (Equation (3)) does not exceed the threshold τ_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 τ_v. This expansion continues until no more neighboring nodes can be added

Table 2 Benchmarking of Seqwin, Fur, Unikseq, and Neptune on genomes from NCBI Taxonomy

Target taxon	# genomes[1]	Tool	# signatures	Median length (bp)	Median conservation	Median divergence	Wall-clock time (s)[2]	Peak memory usage (GB)[3]
C. difficile (ID 1496)	100 (20)	Seqwin	33	271	0.994	0.12	12.2	3.42
		Fur	0	N/A	N/A	N/A	10.8	3.54
		Unikseq	9,074	237	0.975	0.067	1,090	78.8
		Neptune	3,679	327	0.962	0.025	2,450	13.5
	1,000 (167)	Seqwin	227	313	0.992	0.13	34.5	5.14
		Fur	0	N/A	N/A	N/A	68.1	4.21
		Unikseq	10,989	186	0.982	0.032	10,600	565
		Neptune	4,596	212	0.977	0.014	18,700	13.9
	3,995 (667)	Seqwin	156	312	0.993	0.13	70.0[4]	7.60
M. tuberculosis (ID 1773)	100 (20)	Seqwin	1,674	1184	0.994	0.14	18.3	4.29
		Fur	655	372	0.995	0.0087	12.4	6.14
		Unikseq	10,520	199	1.00	0.16	1,130	71.8
		Neptune	1,843	350	0.992	0.039	1,360	13.5
	1,000 (77)	Seqwin	150	312	0.997	0.090	40.0	5.32
		Fur	0	N/A	N/A	N/A	63.2	7.46
		Unikseq	103	172	0.993	0.020	11,700	576
		Neptune	30	301	0.993	0.0086	2,760	14.8
	8,296 (637)	Seqwin	208	350	0.998	0.091	194[4]	11.7
S. enterica subsp. *enterica* (ID 59201)	100 (20)	Seqwin	382	333	0.990	0.048	13.7	3.74
		Fur	0	N/A	N/A	N/A	11.4	6.22
		Unikseq	593	249	0.339	0.017	1,260	77.2
		Neptune	24	800	0.685	0.0036	206	14.7
	1,000 (21)	Seqwin	319	317	0.990	0.053	40.0	5.39
		Fur	0	N/A	N/A	N/A	52.9	4.91
		Unikseq	542	233	0.337	0.021	13,200	624
		Neptune	23	382	0.797	0.0029	2,770	15.6
	14,822 (304)	Seqwin	275	321	0.990	0.049	313[4]	22.3

[1] Including neighboring non-target genomes. The number of non-target genomes is shown in parentheses. Percentage of complete genomes (assembly level at or above chromosome) in each full dataset: 7.43% (*C. difficile*), 12.03% (*M. tuberculosis*), 13.73% (*S. enterica*). Metadata of all genomes can be found in Supplementary Data 2-4.
[2] 20 CPU threads/processes were used for Seqwin, Fur and Neptune. Unikseq only supported a single CPU thread.
[3] Neptune stores k-mers of input genomes as temporary files on disk.
[4] When Seqwin was run on the full datasets, minimizer sketches were used to calculate the penalty thresholds to reduce running time (see Methods and Supplementary Note 5).

or the subgraph reaches a specified maximum size (100 by default). Subgraphs meeting the minimum size requirement (3 by default) are retained, and their nodes are marked used so subgraphs remain disjoint (no node is part of more than one subgraph). Seeds are processed in random order, and the final list of subgraphs is shuffled to balance subgraph sizes in downstream processes (subgraphs created first tend to be larger due to the greedy optimization).

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 1) finding the maximal consecutive occurrence of the subgraph's minimizers in each target genome, and 2) choosing the most common one across all target genomes, weighted by length (Algorithm S3 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 be 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 common minimizer ordering (treating forward and reverse ordering as equivalent) is then selected as the subgraph's representative minimizer ordering. 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 ordering 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. Candidates are evaluated for their sensitivity and specificity, measured as conservation and divergence, respectively (Supplementary Note 2).

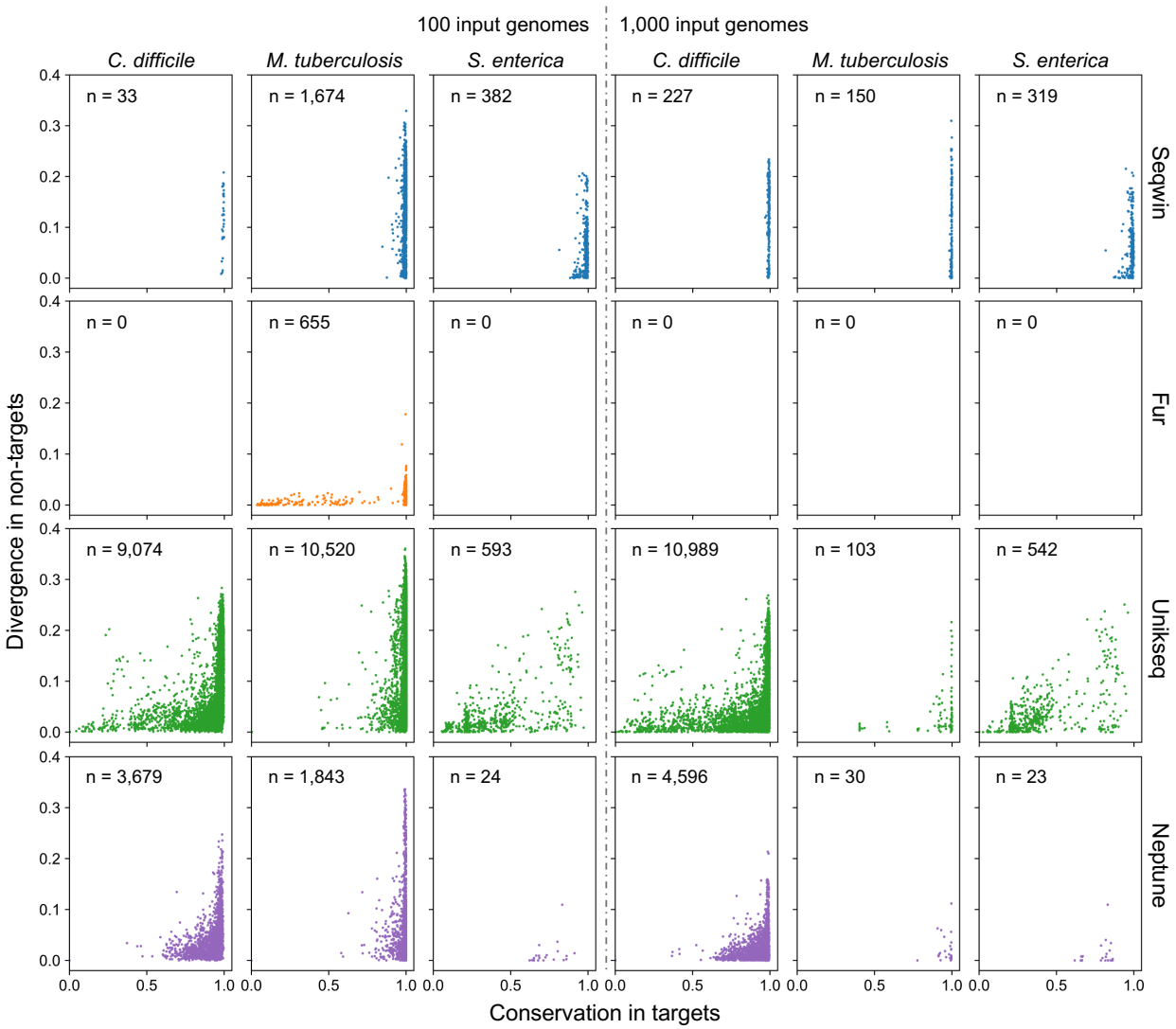

Figure 2 Benchmarking of Seqwin, Fur, Unikseq and Neptune using genomes downloaded from NCBI Taxonomy [31]. Each data point represents a signature sequence, generated by one of the tools using genomes under a pathogenic taxon as targets (e.g., *C. difficile*), and genomes under its neighboring taxa as non-targets. Blue, orange, green and purple represent Seqwin, Fur, Unikseq and Neptune, respectively. The number of output signatures (data points) in each setting is shown in each scatter plot. 100 or 1,000 genomes are sampled from each NCBI full dataset (Table 2), and used as inputs to the tools. See Supplementary Note 2 for the calculation of conservation and divergence.

Results

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

To test Seqwin's ability to identify sensitive and specific signatures, we compared it with other tools. We selected Fur as a fast, memory-efficient method and Unikseq and Neptune as representative k-mer assembly methods (Table 1). Note, NAUniSeq was not included in the benchmark given we were unable to successfully run it on our evaluation dataset due to missing files that caused installation errors.

We first used the dataset published with Fur [11] to benchmark the four tools (Seqwin, Fur, Unikseq and Neptune). The Fur dataset consists of 33 *Escherichia coli* genomes from 6 different strains, labeled A (6 genomes), B1 (14 genomes), B2 (5 genomes), D (2 genomes), E (4 genomes) and F (2 genomes). For each run of each tool, genomes under one of the strains were used as targets (e.g., strain A), and genomes under all other strains (e.g., B1, B2, D, E and F) were used as non-targets. Thus, each tool generated 6 different sets of signatures, summarized in Table S1. Each signature was evaluated for its sensitivity (conservation) and specificity (divergence), with results shown in Figure S2. 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 S3.

Next, we benchmarked all four tools against microbial genomes retrieved from NCBI Taxonomy. 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). Here, "neighboring taxa" are those that diverge from a common ancestral node into separate sister lineages. For example, as the only species in the genus *Clostridioides*, *C. difficile*'s neighbors encompass

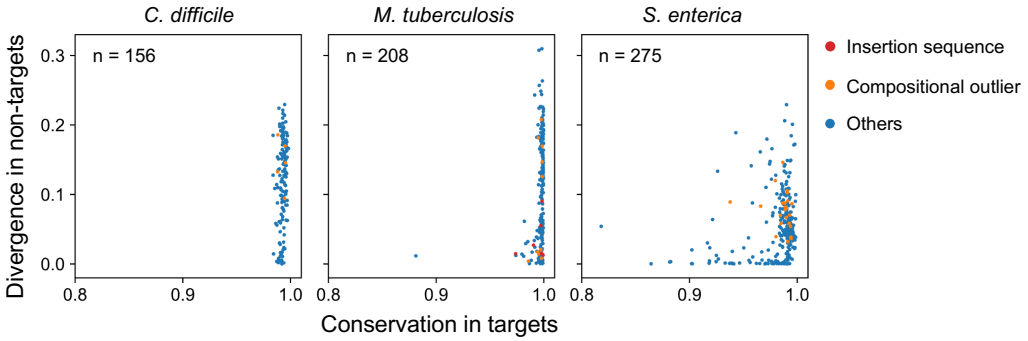

Figure 3 Output signatures of Seqwin using all genomes downloaded from NCBI Taxonomy. Each data point represents a signature sequence, generated by Seqwin using genomes under a pathogenic taxon as targets (e.g., *C. difficile*), and genomes under its neighboring taxa as non-targets. The number of output signatures (data points) in each setting is shown in each scatter plot. Signatures overlapping with potential MGEs are labeled in red and orange, all others are 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.

all other genera under the shared family, *Peptostreptococcaceae*. Genomes from 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 dataset, we sampled 100 or 1,000 genomes as input (targets and non-targets combined), with results shown in Table 2 and Figures 2 and S4. To retain genome diversity, we included at least 20 non-target genomes for each of the 100-genome datasets, since target genomes far outnumbered non-target genomes in the full datasets (Table 2). Metadata of all NCBI genomes can be found in Supplementary Data 2-4.

Compared to Fur, Seqwin identified more signatures with comparable running time and peak memory (Tables S1 and 2). Fur produced zero signatures in several experiments, most likely due to its stringent search strategy. Compared to Unikseq, Seqwin's output signatures had similar median lengths (Tables S1 and 2) in most settings. Although Unikseq produced more signatures, many of them had lower conservation scores (Figures S2 and 2). This was especially true for *S. enterica* (Table 2 and Figure 2), 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. Neptune's output signatures resembled those of Unikseq, with higher conservation but still less than Seqwin (Table 2 and Figure 2). It also required more running time and peak memory than Seqwin (Tables S1 and 2). Note that Neptune stores k-mers of input genomes as temporary files on disk, which reduces peak memory usage but can increase runtime.

Lists of genomes used in all experiments can be found in Supplementary Data 1-4. Benchmark details can be found in Supplementary Note 5.

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 2). For nearly 15,000 *S. enterica* genomes, Seqwin finished in only 5 minutes using 20 CPU cores and 22 GB peak memory (Table 2). Seqwin also maintained similar signature quantity and quality to those obtained with 1,000 genomes, as shown in Figure 3. Note that the penalty thresholds in these experiments were calculated with minimizer sketches to reduce 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 [32] and the Mobilome Annotation Pipeline (MAP) under MGnify [33], respectively. "Compositional outliers" were potential MGEs identified by MAP (Figure 3), 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 gene products annotated as transposase, integrase or resolvase. In all three experiment settings, less than 10% of the signatures overlapped with predicted MGEs (compositional outliers and insertion sequences). Moreover, only *M. tuberculosis* had signatures overlapping with insertion sequences (Figure 3), and their median divergence was 0.021, which is lower than the median divergence of all *M. tuberculosis* signatures (Table 2). Annotation results of all signatures shown in Figure 3 can be found in Supplementary Data 5-7. Details of the annotation process can be found in Supplementary Note 6.

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 target and non-target genome sets as 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 experimental evaluation, we show that Seqwin consistently achieves higher signature sensitivity and specificity across both small and large-scale datasets while using fewer computational resources than other approaches. Although wet-lab validation was not performed in this study (planned as future work), we envision Seqwin being coupled with established primer and probe design software, such as varVAMP [34], Olivar [35] and PrimalScheme [36], supporting sensitive, specific PCR assay design for clinical and public health surveillance.

However, there are two major open problems we leave for future work. First, Seqwin estimates specificity from the provided non-target genomes and considers all non-targets equally. Consequently, specificity is best estimated when the non-target set is restricted to close phylogenetic neighbors, ideally using well-curated and high-quality genomes. Although Seqwin is flexible and tolerates some noise in the non-target set, including a large number of very distantly related non-target genomes can bias specificity estimation. In practice, it is suggested to perform an additional screening of candidate signatures against a comprehensive background and examine hits outside the intended search space. Taxonomic mislabelling in public databases is another challenge that Seqwin itself cannot resolve. Using a genome-based taxonomy database such as GTDB [37] may alleviate this issue. A potential solution is to weight non-target genomes by sequence similarity, for example using Jaccard indices, so that specificity estimation is driven more strongly by the most relevant near-neighbor genomes. Second, Seqwin is designed to identify single genomic regions that distinguish the target group from the non-target group, rather than combinations of multiple regions that are informative only when considered together. As a result, Seqwin may miss combinatorial signatures in which no individual region is sufficiently conserved across the full target set, even though the union of those regions could still distinguish targets from non-targets. This limitation is relevant when designing assays for antimicrobial-resistant (AMR) strains of a pathogen, as AMR can depend on combinations of mutations or genes across multiple loci rather than on a single diagnostic region [38]. Support for identifying combinatorial signatures is under development and is planned for a future Seqwin update.

In summary, inspired by approaches pioneered by Insignia two decades ago, Seqwin represents a highly sensitive and discriminative computational approach for microbial genome signature discovery. We anticipate Seqwin will enable signature discovery at unprecedented scale and support automated, sensitive, specific PCR assay design for clinical and environmental applications.

Conflicts of interest

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.

Funding

This work was 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.].

Data availability

NCBI accessions and metadata of all genomes used in this study can be found in Supplementary Data. Benchmarking datasets, outputs, and scripts are available on Zenodo `https://doi.org/10.5281/zenodo.19176444`.

Author contributions statement

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

Acknowledgments

The authors thank Dr. Adam Phillippy and Dr. Jingjing Wu for valuable feedback and suggestions.

References

1. K. D. Eisenach, M. Donald Cave, J. H. Bates, and J. T. Crawford. Polymerase chain reaction amplification of a repetitive DNA sequence specific for Mycobacterium tuberculosis. *Journal of Infectious Diseases*, 161(5):977–981, 1990.
2. F. Laure, C. Rouzioux, F. Veber, C. Jacomet, V. Courgnaud, S. Blanche, M. Burgard, C. Griscelli, and C. Brechot. Detection of HTV1 DNA in infants and children by means of the polymerase chain reaction. *The Lancet*, 332(8610): 538–541, 1988.
3. T. Slezak, T. Kuczmarski, L. Ott, C. Torres, D. Medeiros, J. Smith, B. Truitt, N. Mulakken, M. Lam, E. Vitalis, et al. Comparative genomics tools applied to bioterrorism defence. *Briefings in bioinformatics*, 4(2):133–149, 2003.
4. E. K. Nordberg. YODA: selecting signature oligonucleotides. *Bioinformatics*, 21(8):1365–1370, 2005.
5. S. Feng and E. R. Tillier. A fast and flexible approach to oligonucleotide probe design for genomes and gene families. *Bioinformatics*, 23(10):1195–1202, 2007.
6. A. M. Phillippy, J. A. Mason, K. Ayanbule, D. D. Sommer, E. Taviani, A. Huq, R. R. Colwell, I. T. Knight, and S. L. Salzberg. Comprehensive DNA signature discovery and validation. *PLoS computational biology*, 3(5):e98, 2007.
7. A. M. Phillippy, K. Ayanbule, N. J. Edwards, and S. L. Salzberg. Insignia: a DNA signature search web server for diagnostic assay development. *Nucleic acids research*, 37 (suppl_2):W229–W234, 2009.
8. K. C. Bader, C. Grothoff, and H. Meier. Comprehensive and relaxed search for oligonucleotide signatures in hierarchically clustered sequence datasets. *Bioinformatics*, 27(11):1546–1554, 2011.
9. M. Knowles, D. Lambert, G. Huszczynski, M. Gauthier, and B. W. Blais. PCR for the specific detection of an Escherichia coli O157: H7 laboratory control strain. *Journal of Food Protection*, 78(9):1738–1744, 2015.
10. C. Camacho, G. Coulouris, V. Avagyan, N. Ma, J. Papadopoulos, K. Bealer, and T. L. Madden. BLAST+: architecture and applications. *BMC bioinformatics*, 10(1): 1–9, 2009.
11. B. Haubold, F. Klötzl, L. Hellberg, D. Thompson, and M. Cavalar. Fur: Find unique genomic regions for diagnostic PCR. *Bioinformatics*, 37(15):2081–2087, 2021.
12. B. Vieira Mourato, I. Tsers, S. Denker, F. Klötzl, and B. Haubold. Marker discovery in the large. *Bioinformatics advances*, 4(1):vbae113, 2024.

13. E. Marinier, R. Zaheer, C. Berry, K. A. Weedmark, M. Domaratzki, P. Mabon, N. C. Knox, A. R. Reimer, M. R. Graham, L. Chui, et al. Neptune: a bioinformatics tool for rapid discovery of genomic variation in bacterial populations. *Nucleic acids research*, 45(18):e159–e159, 2017.
14. Y. Wang, Q. Chen, C. Deng, Y. Zheng, and F. Sun. KmerGO: a tool to identify group-specific sequences with k-mers. *Frontiers in microbiology*, 11:2067, 2020.
15. P. Beran, D. Stehlíková, S. P. Cohen, and V. Čurn. KEC: unique sequence search by k-mer exclusion. *Bioinformatics*, 37(19):3349–3350, 2021.
16. M. J. Allison, R. L. Warren, M. L. Lopez, N. Acharya-Patel, J. J. Imbery, L. Coombe, C. L. Yang, I. Birol, and C. C. Helbing. Enabling robust environmental DNA assay design with "unikseq" for the identification of taxon-specific regions within whole mitochondrial genomes. *Environmental DNA*, 5(5):1032–1047, 2023.
17. M. L. D. Lopez, R. L. Warren, M. J. Allison, L. Coombe, J. J. Imbery, I. Birol, and C. C. Helbing. Conserved Sequence Identification Within Large Genomic Datasets Using 'Unikseq2': Application in Environmental DNA Assay Development. *Molecular Ecology Resources*, 25(7):e70014, 2025.
18. G. K. Sharma, R. Sharma, K. Joshi, S. Qureshi, S. Mathur, S. Sinha, S. Chatterjee, and V. Nunia. Advancing microbial diagnostics: a universal phylogeny guided computational algorithm to find unique sequences for precise microorganism detection. *Briefings in Bioinformatics*, 25(6):bbae545, 2024.
19. E. Theiller, S. Rajagopol, S. Higgins, D. I. Torres, T. Napper, B. E. Galis, A. Dash, E. Qian, L. Hamlette, Q. She, et al. Empowering global disease surveillance with CURED: a tool for rapid identification of unique genomic biomarkers. *mSystems*, pages e01063–25, 2026.
20. L. Coombe, V. Nikolić, J. Chu, I. Birol, and R. L. Warren. ntJoin: Fast and lightweight assembly-guided scaffolding using minimizer graphs. *Bioinformatics*, 36(12):3885–3887, 2020.
21. L. Coombe, P. Kazemi, J. Wong, I. Birol, and R. L. Warren. ntSynt: multi-genome synteny detection using minimizer graph mappings. *BMC biology*, 23(1):367, 2025.
22. B. Ekim, B. Berger, and R. Chikhi. Minimizer-space de Bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer. *Cell systems*, 12(10): 958–968, 2021.
23. E. Afshinnekoo, C. Meydan, S. Chowdhury, D. Jaroudi, C. Boyer, N. Bernstein, J. M. Maritz, D. Reeves, J. Gandara, S. Chhangawala, et al. Geospatial resolution of human and bacterial diversity with city-scale metagenomics. *Cell systems*, 1(1):72–87, 2015.
24. M. Roberts, W. Hayes, B. R. Hunt, S. M. Mount, and J. A. Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20(18):3363–3369, 2004.
25. V. Nikolić, P. Kazemi, L. Coombe, J. Wong, A. Afshinfard, J. Chu, R. L. Warren, and I. Birol. btllib: A C++ library with Python interface for efficient genomic sequence processing. *Journal of Open Source Software*, 7(79):4720, 2022.
26. B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy. Mash: fast genome and metagenome distance estimation using MinHash. *Genome biology*, 17(1):1–14, 2016.
27. H. Zheng, G. Marçais, and C. Kingsford. Creating and using minimizer sketches in computational genomics. *Journal of Computational Biology*, 30(12):1251–1276, 2023.
28. R. Edgar. Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences. *PeerJ*, 9: e10805, 2021.
29. B. Kille, E. Garrison, T. J. Treangen, and A. M. Phillippy. Minmers are a generalization of minimizers that enable unbiased local Jaccard estimation. *Bioinformatics*, 39(9): btad512, 2023.
30. M. Belbasi, A. Blanca, R. S. Harris, D. Koslicki, and P. Medvedev. The minimizer Jaccard estimator is biased and inconsistent. *Bioinformatics*, 38(Supplement_1):i169–i176, 2022.
31. C. L. Schoch, S. Ciufo, M. Domrachev, C. L. Hotton, S. Kannan, R. Khovanskaya, D. Leipe, R. Mcveigh, K. O'Neill, B. Robbertse, et al. NCBI Taxonomy: a comprehensive update on curation, resources and tools. *Database*, 2020:baaa062, 2020.
32. C. P. Cantalapiedra, A. Hernández-Plaza, I. Letunic, P. Bork, and J. Huerta-Cepas. eggNOG-mapper v2: functional annotation, orthology assignments, and domain prediction at the metagenomic scale. *Molecular biology and evolution*, 38(12):5825–5829, 2021.
33. L. Richardson, B. Allen, G. Baldi, M. Beracochea, M. L. Bileschi, T. Burdett, J. Burgin, J. Caballero-Pérez, G. Cochrane, L. J. Colwell, et al. MGnify: the microbiome sequence data analysis resource in 2023. *Nucleic acids research*, 51(D1):D753–D759, 2023.
34. J. Fuchs, J. Kleine, M. Schemmerer, J. Kreibich, W. Maier, N. Battur, T. Krannich, S. Sedaghatjoo, L. Jaki, A. Maks, et al. varVAMP: degenerate primer design for tiled full genome sequencing and qPCR. *Nature Communications*, 16(1):5067, 2025.
35. M. X. Wang, E. G. Lou, N. Sapoval, E. Kim, P. Kalvapalle, B. Kille, R. L. Elworth, Y. Liu, Y. Fu, L. B. Stadler, et al. Olivar: towards automated variant aware primer design for multiplex tiled amplicon sequencing of pathogens. *Nature Communications*, 15(1):6306, 2024.
36. C. Kent, A. D. Smith, J. Tyson, D. Stepniak, E. Kinganda-Lusamaki, T. Lee, M. Weaver, N. Sparks, T. Brier, L. Landsdowne, et al. PrimalScheme: open-source community resources for low-cost viral genome sequencing. *bioRxiv*, pages 2024–12, 2024.
37. D. H. Parks, P.-A. Chaumeil, A. J. Mussig, C. Rinke, M. Chuvochina, and P. Hugenholtz. GTDB release 10: a complete and systematic taxonomy for 715 230 bacterial and 17 245 archaeal genomes. *Nucleic Acids Research*, 54(D1): D743–D754, 2026.
38. F. M. Kaczmarek, F. Dib-Hajj, W. Shang, and T. D. Gootz. High-level carbapenem resistance in a Klebsiella pneumoniae clinical isolate is due to the combination of bla ACT-1 β-lactamase production, porin OmpK35/36 insertional inactivation, and down-regulation of the phosphate transport porin PhoE. *Antimicrobial agents and chemotherapy*, 50 (10):3396–3406, 2006.