Bioinformatics, 38, 2022, i153–i160
https://doi.org/10.1093/bioinformatics/btac223
ISCB/ISMB 2022

ReadBouncer: precise and scalable adaptive sampling for nanopore sequencing

Jens-Uwe Ulrich[1,2,3,*], Ahmad Lutfi[1,3], Kilian Rutzen[4] and Bernhard Y. Renard [1,2,*]

[1]Hasso Plattner Institute, Digital Engineering Faculty, University of Potsdam, 14482 Potsdam, Germany, [2]Bioinformatics Unit (MF1), Robert Koch Institute, 13353 Berlin, Germany, [3]Department of Mathematics and Computer Science, Free University of Berlin, 14195 Berlin, Germany and [4]Genome Sequencing Unit (MF2), Robert Koch Institute, 13353 Berlin, Germany

*To whom correspondence should be addressed.

Abstract

Motivation: Nanopore sequencers allow targeted sequencing of interesting nucleotide sequences by rejecting other sequences from individual pores. This feature facilitates the enrichment of low-abundant sequences by depleting overrepresented ones in-silico. Existing tools for adaptive sampling either apply signal alignment, which cannot handle human-sized reference sequences, or apply read mapping in sequence space relying on fast graphical processing units (GPU) base callers for real-time read rejection. Using nanopore long-read mapping tools is also not optimal when mapping shorter reads as usually analyzed in adaptive sampling applications.

Results: Here, we present a new approach for nanopore adaptive sampling that combines fast CPU and GPU base calling with read classification based on Interleaved Bloom Filters. ReadBouncer improves the potential enrichment of low abundance sequences by its high read classification sensitivity and specificity, outperforming existing tools in the field. It robustly removes even reads belonging to large reference sequences while running on commodity hardware without GPUs, making adaptive sampling accessible for in-field researchers. Readbouncer also provides a user-friendly interface and installer files for end-users without a bioinformatics background.

Availability and implementation: The C++ source code is available at https://gitlab.com/dacs-hpi/readbouncer.

Contact: jens-uwe.ulrich@hpi.de or bernhard.renard@hpi.de

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

During the last decade, the invention of nanopore sequencing instruments has democratized DNA sequencing in various aspects (Leggett and Clark, 2017; Mikheyev and Tin, 2014). For example, the small MinION devices of Oxford Nanopore Technologies (ONT) provide the possibility to sequence a sample at the place of its origin, without the need to ship the sample to a laboratory (Runtuwene *et al.*, 2019; Sim and Chapman, 2019). This point-of-care sequencing ability makes nanopore sequencing attractive for applications such as pathogen detection in a clinical setting and in the field (Mongan *et al.*, 2020; Quick *et al.*, 2016). It also can shorten the time to detect pathogens or antimicrobial resistance (AMR) genes when using it for point-of-care testing.

While the size of the device and the easier and faster sample preparation are clear advantages, nanopore sequencing still lags the base quality of sequencing-by-synthesis instruments. However, recent improvements in base-calling algorithms showed per read accuracy exceeding 90% (Rang *et al.*, 2018; Wick *et al.*, 2019). ONT even claims to boost per read accuracy up to 99% with their latest R10.4 pore version (https://nanoporetech.com/accuracy). Another exciting feature of ONT's instruments is sequencing DNA molecules in a targeted fashion. Oxford Nanopore provides an Application Programming Interface (API) that enables receiving electrical currents, measured while the molecule transverses the pore (Loose *et al.*, 2016). These signals can be translated into sequence space and analyzed in real-time. An uninteresting DNA molecule located in a pore can be ejected by sending an 'unblock' message back to the control software. This message leads the sequencer to reverse the voltage across the pore, causing the molecule to exit the pore in the reverse direction. The primary requirement for such a live depletion system is that the software making ejection decisions can keep up with the sequencing speed for up to 512 nanopores that concurrently sequence DNA molecules on a MinION sequencer.

Two recent publications describe the implementation of such systems for specific settings. Payne *et al.* combined Oxford Nanopore's Guppy base caller (Wick *et al.*, 2019) with the minimap2 read aligner (Li, 2018) in their Readfish workflow to make ejection decisions after mapping the reads to a reference genome in real-time. Kovaka *et al.* (2021) skipped the base-calling step and performed ejection decisions directly on nanopore current signals. While the latter is designed to run on a general-purpose CPU, it cannot handle large human size reference genomes. In contrast, Readfish can handle larger references but needs additional software like DeepNano-blitz (Boža *et al.*, 2020) or Oxford Nanopore's Guppy graphical processing units (GPU) basecaller for real-time base calling.

Furthermore, the usage of minimap2 (Li, 2018) for read classification is not optimal. In their study, Payne *et al.* showed that only 83% of target reads were correctly classified for rejection after 0.8 s of sequencing. Marquet *et al.* (2022) observed the same issue when they tried to deplete all human host reads from vaginal samples with

ONT's adaptive sampling option. Using the depletion method supported by MinKNOW, 25% of human reads could not accurately be rejected by the software, wasting many resources on sequencing uninteresting reads. Further, missed mappings to repetitive regions of the reference genome can lead to delayed classifications when longer parts of the DNA molecule must be sequenced to make a rejection decision. Both lower sensitivity and classification delay will cause decreased enrichment of clinically relevant sequences of undetected pathogens or antibiotic resistance markers.

This study introduces *ReadBouncer* as a new tool for nanopore adaptive sampling that combines state-of-the-art base-calling software with the DREAM index (Dadi *et al.*, 2018; Piro *et al.*, 2020). Readbouncer facilitates both GPU base-calling with ONTs Guppy as well as CPU base-calling with DeepNano-blitz (Boža *et al.*, 2020). Its Interleaved Bloom Filter (IBF) data structure allows for fast querying of hashed k-mers on large sequence datasets resulting in an improved read classification strategy. Within an integrated workflow, Readbouncer uses IBFs to classify base-called DNA fragments for ejection and finally communicates the decision to the sequencing control software.

We first investigate our read classification approach by comparing it to other software tools used for read classification in a nanopore adaptive sampling context. ReadBouncer shows the best accuracy, recall, F1-Score and Matthews correlation coefficient (MCC) among all tools on a simulated and a real-world dataset, while having almost the same precision and specificity as the best competitor. Furthermore, our tool also has the smallest reference sequence index size and peak memory usage.

We also compare ReadBouncer with Readfish and ONT's MinKNOW software using a playback run of a whole human genome sequencing experiment to evaluate its adaptive sampling performance. In this comparison, we demonstrate that ReadBouncer outperforms the other tools in a targeted sequencing experiment. ReadBouncer results consistently show more sequenced bases for target references and significantly shorter mean read length of off-target or rejected nanopore reads. These results indicate that ReadBouncer can make faster and more reliable rejection decisions than Readfish and MinKNOW. ReadBouncer's source code and installer files for Windows and Linux are freely available as a Git repository (https://gitlab.com/dacs-hpi/readbouncer) under GNU General Public License 3 (GPL-3.0).

2 Materials and methods

2.1 Read classification

With the current nanopore sequencing speed of 450 nucleotides per second, an adaptive sampling approach ideally makes ejection decisions within 2 s after sequencing of a DNA molecule has started. This requires fast base calling and rapid and reliable classification of read fragments smaller than 500 nucleotides. Readfish (Payne *et al.*, 2021) uses the long-read alignment tool minimap2 (Li, 2018) for this purpose. Although being fast and accurate for long error-prone nanopore reads, the alignment approach poses some challenges when working with short error-prone fragments of less than 500 nucleotides. For optimal enrichment of low abundance genomic regions, we need to make reliable rejection decisions as fast as possible. Payne *et al.* showed in their study that it takes about 360 nucleotides for minimap2 to align 90% of those reads correctly. That means, if we want to get higher enrichment, we need to improve the classification sensitivity for the same read length. Mappings are also hard to use when there is no good quality reference sequence available for an organism that is the depletion target such as non-model organisms. In such scenarios, one would try to use the reference sequence of a closely related species for read classification. Mapping reads to the reference of a closely related species would fail to find numerous reads that we would aim to eject from the pore.

All these findings motivated us to seek a different, fast classification strategy. To our knowledge, the fastest current sequence comparison algorithms use k-mer-based approaches, where a DNA sequence is divided into small overlapping substrings of size k. One approach, known as MinHash (Broder, 1997; Ondov *et al.*, 2016), computes a hash value for every k-mer of a sequence and stores the smallest hash values within a data structure called a sketch. The same procedure is applied to the second sequence, and the number of hash values present in both sketches gives an accurate approximation of the identity between the two sequences. Although this works well for sequences of similar size, it fails for sequence containment tests, where one sequence is much smaller than the other one, which is the case when we want to check if a nanopore read is part of a reference genome.

A better approach for testing if the set of k-mers of a reference genome contains the k-mers of a read is using Bloom Filters (Bloom, 1970; Koslicki and Zabeti, 2019). A Bloom Filter simply is a bitvector of size n and a set of h independent hash functions. To insert a k-mer into a Bloom Filter, the bit positions that correspond to the h hash values of the k-mer are set to 1, and a k-mer is considered present in the Bloom Filter if all h positions return a 1 during the lookup phase. In our case, we would insert all k-mers of a reference genome into the Bloom Filter and lookup for the k-mers of a nanopore read in that Bloom Filter.

The biggest problem of k-mer-based approaches is choosing the correct parameter value for k, which is always a tradeoff between sensitivity and specificity in the presence of sequencing errors. Larger values for k will result in more specific read classification results but will also fail to find many reads from the reference genome when the number of sequencing errors is high. When trying to classify nanopore reads with error rates of about 10%, the value for k will hardly become bigger than 13. The number of different k-mers of size 13 is combinatorially defined by $4^{13} = 67,108,864$, which is much too small when working with human-sized genomes that compose about 3 billion k-mers of size 13. To overcome this issue, we divide the reference genome into overlapping fragments of size m and construct a separate Bloom Filter for each fragment. However, querying one read against each of the Bloom Filters separately reduces the performance of the Bloom Filter approach. Thus, we decided to use IBF as proposed by Dadi *et al.* (2018) to index the reference genomes.

An IBF combines several Bloom Filters (bins) in one single bitvector. The IBF can be divided into several subvectors, each having the size of the number of bins. Since one bin in the IBF corresponds to one fragment of the reference sequence, the size of each subvector corresponds to the number of fragments. In Figure 1, for example, we divided the reference sequence into three overlapping fragments, each corresponding to one bin of the IBF. Thus, each subvector in the IBF consists of 3 bits. The ith bit of every subvector belongs to the Bloom Filter bin of fragment F_i. When inserting a k-mer from fragment F_i into the IBF, we compute all h hash values, which point us to the corresponding subvectors SV_j and then simply set the ith bit of this subvector to 1.

When querying a read p against the IBF in order to check if it maps to any of the fragments, every k-mer of that read is matched against the IBF. That means we first retrieve the h subvectors SV_j and apply a logical AND to them, resulting in the required binning bitvector indicating the membership of the k-mers in the bins. The example in Figure 2 visualizes this process. Here, the read consists of four 7-mers, for which we have to calculate the three hash values that point us to the corresponding subvectors SV_j, as can be seen in particular for the 7-mer *CAGGATT*. A logical AND of these three subvectors gives us the binning bitvector for that 7-mer. In our example, the binning vector 010 for *CAGGATT* tells us that this 7-mer only matches fragment F_2. Applying this procedure to every 7-mer of the read gives us four binning bit vectors. Finally, we only need to sum up the 1-bits in the binning vectors for every fragment, which gives us the number of matching 7-mers of the read for every fragment. Thus, instead of computing h hash values for every Bloom Filter separately, we only need to compute the h hash values once, which poses a significant reduction in computing time to investigate the membership of a k-mer in every Bloom Filter. This method enables us to quickly count the number of matching k-mers between the reference genome and a specific nanopore read. The challenge is to

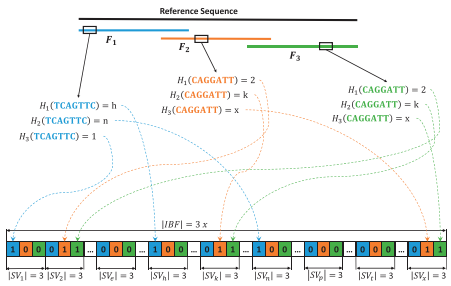

Fig. 1. Example of an IBF construction. In the first step, we subdivide the reference sequence into three overlapping fragments. Then, for each k-mer of the differently colored fragments, all three hash values have to be calculated. The resulting hash values determine the subvector SV_j in which the corresponding bit is set to 1. For example, the second hash function for k-mer *CAGGATT* from fragment F_3 returns *k*. Hence, we set the third bit of subvector SV_k to 1. In this way, the three Bloom Filters for the three fragments are combined in an interleaved fashion. Since we have three fragments in our example, the length of every subvector is three, and the length of the IBF is $3x$, where x is the defined length for every Bloom Filter of the three fragments

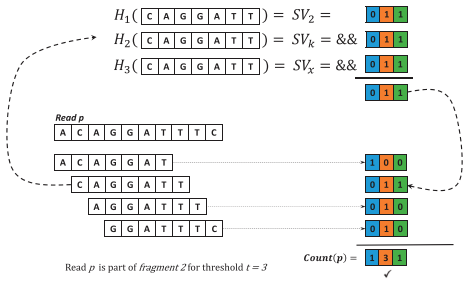

Fig. 2. Finding the correct fragment for a given read *p*. For each k-mer of read *p*, we calculate the three hash values using the same hash functions as for the IBF construction. We use the resulting hash values to find the corresponding subvectors of the IBF. The sub bitvectors are combined with a bitwise AND to a binning bitvector. For all set bits in the binning vectors of the k-mers, we increment the counter of the corresponding bin in a counting vector. Bins whose counter is greater than or equal to a given threshold t are considered to contain the read *p*. In this example, we show the calculation of the binning bitvector for the 7-mer CAGGATT. Using the same three hash functions as for the IBF construction in Figure 1, we get the subvectors SV2, SVk and SVx. We combine these three subvectors via logical AND to get the binning bitvector. The same procedure is applied to the other three 7-mers, and with the resulting four binning bitvectors, we can calculate the number of matching 7-mers of read *p* with each fragment. If at least three 7-mers match against one fragment, we accept the read as a match with the reference sequence

define a threshold value for the number of matching k-mers required to accept a certain nanopore read as a match against a fragment and thus as a match with the reference genome. In our example in Figure 2, we consider the read matching fragment F_2 because three of the four 7-mers match with that fragment. In general, the best threshold value depends on the length of the nanopore read and the expected sequencing error rate. We will describe our method for determining this value in the next section.

2.2 Optimal bitvector size

In a first step, ReadBouncer produces overlapping fragments of the given reference sequences, e.g. 100 000 nucleotide long fragments with an overlap of 500 base pairs. Each of those fragments represents a single bin in the IBF. The constituting k-mers of each fragment are hashed using three different hash functions, and the bits of the corresponding index positions in the IBF are set to one (Fig. 1). Then, ReadBouncer automatically calculates the optimal IBF size in bits (Bits_{IBF}) based on the following equations.

$$\text{Bits}_{\text{IBF}} = n_{\text{frag}} \times \text{Bits}_{\text{SBF}}, \tag{1}$$

where n_{frag} is defined as the number of fragments with maximum size F and Bits_{SBF} as a single Bloom filter size for a single fragment. Let $\max_{\text{kmer}}$ be the maximum number of k-mers for a fragment of size F, and k-mer size k be defined as

$$\max_{\text{kmer}} = F - k + 1. \tag{2}$$

To calculate the optimal size for the IBF, we use the formula for finding the false positive (FP) rate in an IBF as proposed by Dadi *et al.* (2018)

$$p = \left(1 - \left(1 - \frac{1}{\text{Bits}_{\text{SBF}}}\right)^{h \times \max_{\text{kmer}}}\right)^{h}. \tag{3}$$

Then the optimal size of a single Bloom filter can be calculated by resolving the formula for Bits_{SBF}:

$$\text{Bits}_{\text{SBF}} = \left\lceil \frac{-1}{(1-r)^{\frac{1}{h \times \max_{\text{kmer}}}} - 1} \right\rceil, \tag{4}$$

where $r = p^{\frac{1}{h}}$, h is the number of used hash functions and p a predefined FP rate. ReadBouncer implicitly uses three hash functions and a maximum FP rate of 0.01 to minimize the number of false matches between the query sequence and a single bin of the IBF.

2.3 Minimum number of k-mer matches

During the read classification step, the k-mers of every read are hashed with the same three hash functions, and the number of matching k-mers for every bin is calculated as visualized in Figure 2. We accept a read as part of the reference sequence if the number of matching k-mers is greater than or equal to a given threshold t for at least one bin. We calculate the threshold using the expected sequencing error rate e and the definition of a $(1-\alpha)$ confidence interval of the number of erroneous k-mers as recently provided by (Blanca *et al.*, 2022). They first defined the expected number of erroneous k-mers as follows:

$$E[N_{\text{err}}] = L \times q. \tag{5}$$

For a given read r with length $\text{len}(r)$ and k-mer length k, we denote the number of k-mers of read r as $L = \text{len}(r) - k + 1$, and q is defined by $(1-(1-e)^k)$. In a second step, they show that the variance for the number of erroneous k-mers can be calculated by

$$\begin{aligned}\text{Var}(N_{\text{err}}) = L(1-q)\left(q\left(2k + \frac{2}{e} - 1\right) - 2k\right) \\ + k(k-1)(1-q)^2 \\ + \frac{2(1-q)}{e^2}\left((1 + (k-1)(1-q))e - q\right).\end{aligned} \tag{6}$$

Finally, they define the $(1-\alpha)$ confidence interval by:

$$E[N_{\text{err}}] \pm z_\alpha \sqrt{\text{Var}(N_{\text{err}})}, \tag{7}$$

with $z_\alpha = \phi^{-1}(1 - \frac{\alpha}{2})$, where we denote ϕ^{-1} as the inverse of the cumulative distribution function of the standard Gaussian distribution. Based on the calculation of the confidence interval for the number of erroneous k-mers, we define our threshold for the minimum number of matching k-mers for read r as:

$$\min[N_{\text{match}}] = L - (E[N_{\text{err}}] + z_\alpha \sqrt{\text{Var}(N_{\text{err}})}). \tag{8}$$

We classify a read as a match if the number of matching k-mers is bigger or equal to $\min[N_{\text{match}}]$ for at least one bin in the IBF. ReadBouncer per default calculates this threshold for a 95%-confidence-interval, an expected sequencing error rate of 10%, and k-mer length 13. However, these values as well as the fragment size are adjustable via configuration parameters of the command line or graphical user interface (GUI).

2.4 Workflow

The workflow of our tool consists of two consecutive parts. First, we build one or more indexes of the given reference sequence

dataset, which can be used as target or depletion filters. These indexes can be used directly in the second part of the workflow or stored on the computer hard disk for later usage. The construction of this index, for which we apply IBFs, is explained in further detail in Section 2.1. The second part of our tool is the live-depletion or target-enrichment task (Supplementary Fig. S2). Here, ReadBouncer initially loads the indexes and waits for the nanopore device to start sequencing. Immediately after sequencing has begun, the sequencer streams raw electrical currents for every single molecule from every single sequencing pore of the flow cell to our integrated Read-Until client. Oxford Nanopore provides this functionality via an API of its MinKNOW control software (https://github.com/nanoporetech/minknow_api), which allows our Read-Until client to receive the raw data while the molecule traverses through the pore. The client is implemented in C++ and communicates with the MinKNOW control software via gRPC remote procedure calls (https://github.com/grpc/grpc).

Received raw signal data get pushed onto a base-calling queue, and a separate thread takes raw signals of each read from the queue and sends it to the chosen base-calling algorithm, which translates the electrical currents into a nucleotide string. The user can choose GPU base-calling with ONT's Guppy basecaller for which we integrated a guppy client that communicates with a guppy basecall server. In addition, we integrated DeepNano-blitz (Boža *et al.*, 2020) for the base-calling step, which is fast enough to perform the base-calling in real-time, even on CPUs.

Base called reads get pushed to the classification queue if the read length is bigger than or equal to 200 nucleotides, and another thread takes each read from that queue and passes it to the classification framework. Otherwise, the thread marks this read as 'pending' and waits for the following data chunk to be base called and concatenates the base called sequences of the read until the minimum read length has been reached. The minimum read length of 200 nucleotides ensures higher confidence in the classification of the reads. In practice, this read length requirement will lead to most reads having about 360 nucleotides length, which corresponds to two data chunks sent by the MinKNOW software. The read classification thread then queries the read sequence against the loaded IBF indexes as described in more detail in Section 2.1. Based on the classification, reads can either be marked for a rejection or continue further sequencing. If a read was not classified for rejection on a first try, we mark it as *once_seen* and wait for further sequencing data to try further classification attempts of that read. After the read has reached a maximum read length of 1500 bp we stop trying to make ejection decisions and mark the read for continued sequencing as usual. Reads that have been classified for rejection or continued sequencing are finally pushed to the response queue and no further data chunks of that read are sent by the control software.

The last thread takes the classified reads from the response queue, and our Read-Until client sends response messages back to the MinKNOW control software for each read. The client sends an unblock message for reads that could be matched to the IBF, telling the sequencer to eject the corresponding DNA molecule. A *stop_further_data* message is sent to the control software for reads that were not classified for rejection. This message tells MinKNOW to continue sequencing the corresponding DNA molecule and send no additional chunks of data for that read.

3 Results

In this study, we show how adaptive sampling benefits from our improved read classification approach. Therefore, we designed experiments that specifically focus on the evaluation of this approach when applied to both adaptive sampling strategies, depletion and targeted sequencing. In a first step, we compare ReadBouncer to minimap2 (Li, 2018), which is used for classification by Readfish, and the pan-genomics matching tool SPUMONI (Ahmed *et al.*, 2021), which is proposed as an alternative to minimap2 in targeted nanopore sequencing pipelines. Here, we assess all three tools on simulated and real reads from a recently published microbial mock community (Nicholls *et al.*, 2019). In a second experiment, we compare ReadBouncer with Readfish in an adaptive sampling setting using the playback feature offered by Oxford Nanopore's MinKNOW software to replay an already completed sequencing run. We assess both tools by targeting chromosomes 21 and 22 in a human whole genome sequencing run, looking at their ability to correctly filter out all other human nanopore reads. Here, we do not compare against SPUMONI because there exists no adaptive sampling pipeline integrating SPUMONI for read classification.

We perform all experiments for classification performance assessment on a laptop with a 2.8 GHz Intel Core i7-7700HQ CPU and 16 GB of memory with an Ubuntu 20.04 OS installed. For the classification evaluation, we run each tool with a single thread for runtime comparisons and record the wall clock time and peak resident set size (RSS) reported by the individual tools or GNU time 1.7.

3.1 Evaluating read classification

3.1.1 Experimental setup

During a nanopore targeted sequencing experiment with ONT's ReadUntil functionality, the sequencing device transmits electrical current data via the MinKNOW control software to ReadBouncer. This data is received as chunks, representing a maximum of 0.4 or 0.8 s of sequencing, depending on the MinKNOW configuration. Since a DNA molecule translocates through the pore at a speed of about 450 bases per second, 0.4 s of sequencing represents about 180 bases of data. In the following experiments, we mimic the situation where a chunk represents 0.4 s of sequencing data received and base called immediately by an adaptive sampling tool. Since we aim to make rejection decisions as early as possible while still being able to classify most of the reads correctly, we want to assess the classification accuracy of the 3 tools after two chunks of data, which correspond to 360 nucleotides or 0.8 s of sequencing. In this section, we assume that base-calling has already been performed. For a fair comparison, we set up all experiments in such a way that all three tools, minimap2, SPUMONI and ReadBouncer, attempt to classify reads based on the 360 bases long read prefix. In practice, all reads, both simulated and real reads, were cut to only the first 360 bases. ReadBouncer then hashes all k-mers of these 360 bases and compares the hash values to a prebuilt IBF of the depletion target references to make classification decisions.

We use the software's default settings for the SPUMONI approach, which means splitting the prefix into substrings of 90 nucleotides each for further read classification. SPUMONI also needs a prebuild index of the references but has to include the reverse complement of the depletion target references. SPUMONI matches the substrings against this positive index and a null index, consisting of the reverse sequences of the positive index. Finally, classification decisions are made by using a Kolmogorov-Smirnov test.

For the minimap2-based approach, we evaluate two different parameter settings. First, we mimic the read classification of Readfish by using the mappy Python interface (https://pypi.org/project/mappy/) for minimap2. Here, we align the read prefixes with the *map-ont* settings, which are the same settings used by Readfish and correspond to k-mer size of 15. Since the choice of the k-mer size has an impact on the classification performance, we also aligned the read prefixes using a k-mer size of 13 in a second experiment to ensure a fair comparison with ReadBouncer.

To evaluate the three tools, reads correctly classified as belonging to the depletion target are considered true positives (TP), while reads falsely classified for depletion are called FP. Consistently, reads that are correctly not classified as depletion target are considered true negatives (TN), and reads belonging to the depletion target but not classified for depletion are called false negatives (FN). We calculate the classification accuracy, precision, recall, specificity and F1-score for all three approaches based on those considerations. Since we assume an imbalanced number of sequenced reads between depletion and enrichment targets, we also report the MCC in every experiment.

3.1.2 Simulated mock community

In the first dataset, we consider simulated ONT-like reads derived from the identical genomes of the ZymoBIOMICS High Molecular Weight DNA Mock Microbial community (ZymoMC) (Nicholls *et al.*, 2019). This mock community consists of seven bacterial species—*Enterococcus faecalis*, *Listeria monocytogenes*, *Bacillus subtilis*, *Salmonella enterica*, *Escherichia coli*, *Staphylococcus aureus* and *Pseudomonas aeruginosa*—as well as *Saccharomyces cerevisiae*. We use PBSIM2 (Ono *et al.*, 2021) to simulate Oxford-Nanopore-like reads (R9.4 pores) from Zymo Mock Community references at varying levels of mean read accuracy: 80%, 85%, 90%, 95% and 98%. Furthermore, we simulated proportions of reads from each genome in such a way to mimic a scenario where only 2.16% of reads originate from *S.cerevisiae* (Supplementary Fig. S3). The goal here is to enrich *S.cerevisiae* sequences by correctly classifying bacterial reads, which we would aim to eject from the pores in a real nanopore sequencing run. This can be considered as a depletion-only experiment, where a priori only the depletion references are known, but not the enrichment targets. Therefore, we build an index of the seven bacterial reference genomes and query all bacterial and yeast reads against the index. Consistent with our definition in Section 3.1.1, we consider correctly classified bacterial reads TP, while yeast reads found in the index are considered FP. In addition, we define bacterial reads that are missed to be found by a tool in the index as FN, and yeast reads that are not found in the index are considered TN.

On all read accuracy levels, ReadBouncer consistently demonstrates best accuracy, recall, precision, F1-scores and MCC (Supplementary Table S5). Figure 3 visualizes recall and specificity for the three tools across various read accuracies. It can be observed that recall improves with increasing read accuracy for all three tools while specificity stays almost unchanged. On all read accuracy levels, ReadBouncer demonstrates slightly but consistently better recall (sensitivity) than SPUMONI, while both tools outperform minimap2. Minimap2 is the only tool that shows 100% specificity, but ReadBouncer comes close to 100% as well. SPUMONI lags a bit behind the specificity scores of the other two tools. It can be seen that ReadBouncer is the best performing tool for this read classification task. It combines high recall (sensitivity) with high specificity. The other two tools either have high recall but lower specificity or high specificity but lower recall scores.

3.1.3 Real mock community

Next, we applied our method to real nanopore reads from a Zymo Mock Community (NCBI BioProject PRJNA742838). After sample preparation, we sequenced the mock sample on a MinION flowcell (FLO-MIN106) with v. R9.4.1 pores (Supplementary Section S1). Obtained Fast5 files were base called with DeepNano-blitz using a recurrent neural network size of 48. For better comparison with minimap2, we first build a separately obtained minimap2 mapping as a gold standard. Therefore, we filter out all reads shorter than 2000 base pairs and trim the first 360 nucleotides from each read since we use these bases for later classification. Then, we mapped the trimmed reads with standard ONT settings to the ZymoMC reference genomes and only reads with a mapping quality score bigger or equal to 30 are considered confidently mapped. From these mapped reads, the trimmed 360 nucleotide long prefixes are used for the read classification by the three tools again. Proportions of reads from each genome are similar to the simulated experiment with 2.27% of reads from *S.cerevisiae* (Supplementary Fig. S4). In this experiment, we also measure the peak RSS and index size in GigaByte and the throughput for each of the tools in reads classified per second.

Results in Table 1 show that ReadBouncer achieves better accuracy, recall and F1-score than SPUMONI and minimap2, which both have similar results for those three measures. Minimap2 has slightly better precision and specificity than ReadBouncer. While SPUMONI has almost the same precision as ReadBouncer and minimap2, it shows significantly less specificity. These results are consistent with those for the simulated datasets in section *Simulated Mock Community* and show that ReadBouncer outperforms the other tools on read classification for short nanopore reads.

Another important aspect is the amount of main memory a tool needs to hold the reference index needed for read classification. Using the seven bacterial reference genomes of the Zymo Mock Community as depletion target (reference index), ReadBouncer shows the smallest maximum memory consumption measured as Peak RSS. It only needs 0.099 GigaBytes (GB) of main memory, in contrast to 0.251 GB consumed by minimap2 with k-mer size 13. Furthermore, ReadBouncer has the smallest index file size (0.047 GB) of all three tools. In addition to the smallest memory footprint, ReadBouncer also achieves the highest classification throughput. We can classify 5967 reads per second with our approach compared to 5632 reads per second by minimap2 (k-mer size 15) and 1102 reads per second achieved by SPUMONI. These results show that ReadBouncer can correctly classify more reads and is computationally more efficient than other state-of-the-art tools used for nanopore adaptive sampling.

3.2 Adaptive sampling evaluation

In our live experiment, we assess our read classification based on IBFs in a targeted adaptive sampling setup. For this purpose, we downloaded a bulk Fast5 file of a human whole-genome sequencing experiment provided via the Github page of Readfish (https://github.com/LooseLab/readfish). Such a bulk Fast5 file (Payne *et al.*, 2019) allows the playback of the whole sequencing run for testing if the ReadUntil functionality is working correctly. Oxford Nanopore's MinKNOW software simulates an already finished sequencing run without the need for a physical sequencing device when performing a playback run. Compared to the original sequencing run, read signals are reported at the same time point after starting the run. Unblocking a read does not cause MinKNOW to finish sending signals for that read during a playback run. It just breaks the read when receiving an unblock message for the read and creates a new read identifier but continues to send signals of the same original read. Here, we compare ReadBouncer and Readfish using both

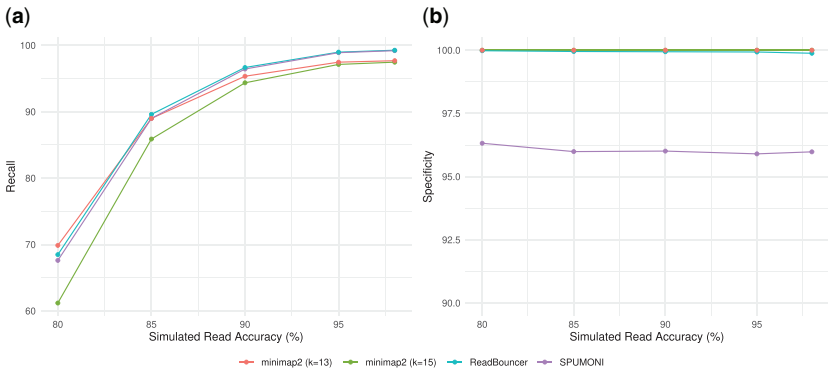

Fig. 3. Visualization of (**a**) recall and (**b**) specificity with varying simulated read accuracies for ReadBouncer, minimap2 and SPUMONI

Table 1. Comparing ReadBouncer, SPUMONI and minimap2 across various metrics on a real Zymo Mock Community dataset consisting of seven bacterial species and *S.cerevisiae*

Tool	ReadBouncer ($k = 13$)	SPUMONI	minimap2 ($k = 15$)	minimap2 ($k = 13$)
Accuracy	**94.50**	90.96	89.33	92.33
Precision	99.99	99.89	**100.00**	99.99
Recall	**94.38**	90.85	89.08	92.15
Specificity	99.73	95.87	**99.95**	**99.95**
F1-Score	**97.10**	95.16	94.23	95.91
MCC	**0.52**	0.41	0.39	0.45
Peak RSS (GB)	**0.099**	0.163	0.272	0.251
Index Size (GB)	**0.047**	0.153	0.097	0.090
Throughput (reads per sec)	**5967**	1102	5632	5306

Note: Reads from a nanopore sequencing run are mapped to the eight organisms to generate ground truth. We use only the first 360 nucleotides for classification from those confidently mapped reads to mimic unblock decision-making after 0.8 s of sequencing the individual read. All reads are mapped against the seven bacterial reference sequences to filter out only the bacterial reads. At the same time, we want to keep as much *S.cerevisiae* reads, which corresponds to an enrichment of that organism in an enrichment/depletion experiment. Consistent with the simulated data, ReadBouncer can classify a higher percentage of bacterial reads while having slightly less precision and specificity than minimap2. Our approach also is the computationally most effective one, with the lowest memory footprint and highest classification throughput. Bold values represent the best metrics value achieved over all tools, e.g. ReadBouncer (k=13) shows highest accuracy, F1-Score, recall but also the smallest memory footprint (Peak RSS and index size).

real-time GPU base-calling with ONT's Guppy basecaller and real-time CPU base-calling with DeepNano-blitz. While ReadBouncer integrates DeepNano, we had to use a special git branch of Readfish (https://github.com/LooseLab/readfish/tree/caller_refactor) to facilitate CPU base calling. In all experiments, ReadBouncer and Readfish were run on a separate Ubuntu 18.04 Laptop with 16 GB RAM and Intel Core i7 while GPU live base calling and the playback were performed on an NVIDIA Jetson AGX Xavier. In addition, we compared the results with two MinKNOW adaptive sampling experiments, one using MinKNOW's *target* and the other using MinKNOW's *deplete* method. Both experiments were performed on the NVIDIA Jetson AGX Xavier, too.

In our experiments, we do a playback of a complete human genome sequencing run with the goal to enrich for chromosomes 21 and 22 of the human genome and deplete all other human reads from that run. This setup not only mimics a targeted sequencing approach but also corresponds to the application of sequencing a clinical human blood sample where up to 99% of the reads are human reads that we would want to deplete in order to enrich the number of reads from a pathogenic microbe. We perform playback runs for 60 min on ONT's MinKNOW control software version 4.3.3. To ensure that the vast majority of the sequenced reads are of human origin, we first perform a playback run without adaptive sampling. Reads were base called with Guppy 5.0.14 and mapped with minimap2 to the human Telomere-to-Telomere Consortium ('T2T') CHM13 v1.1 reference assembly (Nurk *et al.*, 2022). From the resulting reads passing the in-built quality filtering of MinKNOW, 99.66% could be mapped to the human reference genome. For the comparison of the tools in an adaptive sampling setting, we first adjust the *break_reads_after_seconds* parameter within MinKNOW to 0.4 s as recommended by the Readfish authors. Since MinKNOW sends data as chunks, this parameter sets the size of one chunk to a maximum of 180 nucleotides. Both tools, ReadBouncer and Readfish, can concatenate the data chunks and perform classification after the receipt of every chunk. For integrated CPU base calling with DeepNano-blitz we used a neural network size of 48 for both tools. For real-time GPU base calling on the NVIDIA Jetson AGX Xavier we used the fast base-calling mode of Guppy 5.0.14 for ReadBouncer and Readfish.

For the evaluation of both tools, we repeat the same playback run for 60 min. In the experiment with CPU base calling, we ran ReadBouncer with default parameters ($fragment\_size = 100,000$ and $kmer\_size = 13$) using three base calling threads and three read classification threads, respectively. The same setting was applied to Readfish with three CPU base calling threads and minimap2 using three threads per default. Since Guppy ensures a higher raw read accuracy, we ran ReadBouncer with $fragment\_size = 200,000$, $kmer\_size = 15$ and $error\_rate = 0.05$ in the GPU base calling experiment. In all experiments, we used chromosomes 21 and 22 as target filter and all other chromosomes as depletion filter in ReadBouncer. Our settings within the Readfish configuration file correspond to the example TOML file in the github repository (https://github.com/LooseLab/readfish/blob/master/examples/human_chr_selection.toml) and aim to target chromosomes 21 and 22 as well, while unblocking all reads that do not map to the targets. For MinKNOW *target* we used chromosomes 21 and 22 as reference and for MinKNOW *deplete* all other chromosomes as reference sequence. Before starting the adaptive sampling experiments, we had to build index files for all three tools. For Readfish and MinKNOW, we created minimap2 index files, which took 103 s on an Intel Core i7 with one thread and peak RSS of 11.68 GB. Building ReadBouncer index files took 478 s on the same system, but needed only 8.62 GB peak RSS. After finishing the playback run, the resulting Fast5 files were basecalled in high accuracy mode with Guppy 5.0.14. All reads in the resulting fastq files were mapped to the human genome reference and mapping statistics were calculated with Readfish's *summary* script. Using a playback run allows a fair comparison of the different approaches since the same sequencing data come from MinKNOW during the same amount of time. Thus, we expect a similar number of on-target reads and on-target bases across all experiments. On the other hand, we expect different numbers of rejected reads while retaining similar number of bases for those reads due to the different lengths of rejected reads caused by different unblock time points. The reason is MinKNOW just splitting a sequenced read into two segments when receiving an unblock message for that read. Thus, the earlier we reject an off-target read, the shorter the read length and the more off-target reads are seen.

The results of all six experiments can be seen in Table 2. Our first observation is that the results for our target chromosomes 21 and 22 are similar for all experiments but the MinKNOW *deplete* experiment. Here, the number of on-target reads is much higher while showing the smallest mean and median read lengths caused by a high number of false rejection decisions. These results suggest that MinKNOW *deplete* is not suitable for targeting single chromosomes of the human genome in an adaptive sampling experiment. On the other hand, MinKNOW *target* shows similar results for chromosomes 21 and 22 when compared to ReadBouncer and Readfish. However, the mean read length of 3629 bp measured for unblocked reads is much higher than those in the ReadBouncer and Readfish experiments, which shows that MinKNOW *target* spends too much time sequencing off-target reads. These experiments show that

Table 2. Comparison of ReadBouncer, Readfish and MinKNOW in a targeted sequencing experiment

		ReadBouncer				Readfish			
Basecaller	contig	Reads	Bases	Mean	Median	Reads	Bases	Mean	Median
DeepNano	chr21	73	2 208 211	**30 249**	9025	73	2 118 199	29 016	9285
	chr22	92	1 179 449	12 820	6262	91	1 177 699	12 942	5449
	Others	122 745	136 441 510	1112	503	92 527	140 303 151	1516	1310
Guppy	chr21	77	2 189 976	28 441	**9442**	71	2 126 553	29 951	9262
	chr22	83	1 210 472	**14 584**	**7663**	88	1 178 629	13 394	6602
	Others	154 684	140 636 076	907	**479**	140 267	133 484 295	952	877
		MinKNOW *target*				MinKNOW *deplete*			
	contig	Reads	Bases	Mean	Median	Reads	Bases	Mean	Median
	chr21	77	2 099 268	27 263	9170	1425	2 285 420	1604	845
	chr22	105	1 061 911	10 113	3368	468	1 219 518	2606	883
	Others	38 656	140 284 944	3629	520	177 549	132 949 878	**749**	769

Note: Four 60 min playback runs of a whole human genome sequencing experiment were performed using either ReadBouncer or Readfish in combination with either DeepNano CPU base calling or Guppy GPU base calling. The same experiment was repeated with MinKNOW's adaptive sampling functionality in *target* and *deplete* mode. The goal of all experiments was to target chromosomes 21 and 22 while rejecting all other human reads. For chromosomes 21 and 22, highest mean and median read lengths across all experiments are highlighted in bold. For rejected reads, lowest mean and median read lengths across all experiments are highlighted in bold. ReadBouncer and Readfish show consistently better results when using GPU base calling with ReadBouncer having shorter mean and median read lengths for non-target reads regardless of the used basecaller. ReadBouncer outperforms MinKNOW *target* by having longer read lengths for on-target reads and shorter read lengths for off-target reads caused by a better read classification. MinKNOW *deplete* has the worst results of all tools indicated by high numbers of on-target reads with short read lengths caused by lots of false unblock decisions for on-target reads.

ReadBouncer and Readfish outperform the two MinKNOW adaptive sampling strategies.

Comparing ReadBouncer with Readfish, when both tools use the same basecaller, ReadBouncer shows better results regarding median read lengths and the number of bases sequenced. We also see that the choice of the base calling tool has a significant impact on the outcome of the adaptive sampling experiment. Using Guppy GPU base calling for both tools, ReadBouncer and Readfish result in much shorter read lengths for non-target (unblocked) reads. Interestingly, we observe that unblocked reads from the ReadBouncer playback runs have shorter mean and median read lengths than those from the Readfish playback runs. This is also shown in the length distribution plots of unblocked reads for playback runs with Guppy base calling presented in Figure 4.

Fig. 4. Read length distributions of unblocked reads when using (**a**) ReadBouncer or (**b**) Readfish on a 60 min playback run of a whole human sequencing experiment with real-time Guppy GPU base calling. ReadBouncer makes faster rejection decisions than Readfish, which can be observed by shorter read lengths of unblocked nanopore reads

4 Discussion

The idea of adaptive sampling is to selectively sequence individual DNA molecules on nanopore sequencing devices using in-silico methods. This study presents a new tool for adaptive sampling that improves read classification by combining IBFs with k-mer matching statistics. ReadBouncer shows a higher read classification sensitivity than other state-of-the-art classification tools for adaptive sampling while retaining a high specificity. Our tool also improves classification performance and memory usage compared to the other tools. We could observe shorter read lengths of non-target reads in different playback experiments when using ReadBouncer instead of Readfish. In a real experiment, this could mean that ReadBouncer investigates more DNA molecules in the same amount of sequencing time. We developed our tool as an easy-to-install software application with a graphical user interface on Linux and Windows operating systems. In addition, ReadBouncer supports fast CPU basecalling, providing even small sequencing facilities or in-field researchers that typically only have access to low-cost hardware the possibility to use the adaptive sampling feature of the MinION sequencer.

The key benefit of our new tool is the improved read classification. We neither use signal nor sequence space mapping algorithms for read classification compared to other adaptive sampling tools. Instead, our IBF approach uses k-mer counting in Bloom Filters for sequence containment testing, resulting in smaller index files and fewer memory requirements. However, the improved sensitivity comes at the cost of decreased classification speed with increasing reference database size due to our approach of fragmenting the reference genome sequences and using one bin of the IBF per fragment. The fragmentation approach ensures a high classification specificity for nanopore reads with high error rates of approximately 10–15% as observed by the CPU basecaller DeepNano-blitz (Boža *et al.*, 2020). This error rate forces us to use small k-mer sizes such as 13, which entails the need for smaller fragment sizes down to 100 000 nucleotides to avoid too many FP matches. Using real-time GPU base-calling with single raw read accuracies of about 94% allows increasing the k-mer size to 15 and fragment size to 200 000, reducing the number of bins in the IBF by 50%. In the future, we expect to use even fewer fragments per genome and consequentially improve the classification speed for larger genomes as Oxford Nanopore is steadily improving its per-read accuracy. This could also enable the usage of our IBF approach for real-time metagenomics classification of nanopore reads or the construction of pan-genomics indexes that store all different haplotypes of a pathogen in

one IBF, with one haplotype per bin. To further increase performance, combining ReadBouncer and minimap2 could be worthwhile, as the integration of different methods in related fields has demonstrated (Piro *et al.*, 2017).

A second key feature of ReadBouncer is its support for fast and accurate real-time GPU base-calling with ONT's Guppy and real-time CPU base-calling with DeepNano-blitz. This study showed that both approaches show reliable results for a whole human sequencing playback run with the application to target specific chromosomes while rejecting reads belonging to all other chromosomes. Since there are some performance drawbacks of MinKNOW when using a playback run, the measured read lengths of rejected reads can deviate to a real experiment. Other users (https://github.com/sirselim/jetson_nanopore_sequencing) reported much shorter unblocked read lengths on real experiments performed on NVIDIA Jetson AGX Xavier. To ensure reproducibility and fair comparison between tools and to reduce the influence of potential artifacts, we evaluated our tool here on a playback of a well-performed experimental run rather than during run-time of the sequencer. Since a playback run is data from a real sequencing experiment, we do not expect any bias from this comparison but can guarantee a fair comparison between tools. We also do not expect any negative impact on ReadBouncer's classification approach's improved sensitivity by using a playback run.

We expect that ReadBouncer can also contribute to the field of pathogen detection in non-model organisms. Metagenomics sequencing of such samples easily consists of up to 99% host reads that can be depleted with adaptive sampling resulting in an in-silico enrichment of pathogenic reads as shown by other research groups (Marquet *et al.*, 2022; Martin *et al.*, 2022). Here, our CPU-based approach also makes access to adaptive sampling much easier for researchers studying wild living animals in the field. With Nanopores being successfully applied to peptide sequencing (Brinkerhoff *et al.*, 2021), we also see possible modifications of the approach to be useful for targeted protein sequencing.

Another potential use case for adaptive sampling is the real-time detection of antibiotic resistance and virulence genes. In their recently published study, Zhou *et al.* (2021) showed that direct nanopore metagenomics sequencing of human blood samples could detect pathogens in real-time but fails to detect antibiotic resistance genes. They compared direct metagenomic sequencing approach to MinION sequencing of blood culture samples. Using blood cultures, they could deplete human reads to about 65% of all sequenced reads in the corresponding sample, which was sufficient to identify more than 80% of resistance genes after 2 hours of sequencing. We expect that the number of sequenced human host reads can be depleted at a similar rate by using adaptive sampling, which was already shown by Marquet *et al.* This could reduce costs and decrease the time to detect pathogens in human blood samples. In the future, a point-of-care test for antibiotic resistance genes in human patient samples that also avoids shipping the samples to a nearby laboratory could decrease antibiotic drugs' usage and help restrict the development of antibiotic resistance that are a burden to many health care systems all over the world. Besides further sample preparation and sequencing technology improvements, we encourage scientists to set up proof-of-principle studies investigating the potential application of adaptive sampling for real-time AMR gene detection.

Acknowledgements

The authors thank Vitor C. Piro and Tobias P. Loka (HPI) and Knut Reinert (FU Berlin) for valuable discussions and comments on the usage of IBFs and Martin Beer (FLI) for insights into Nanopore sequencing. They thank the Genome Sequencing Unit at Robert Koch Institute for sequencing of the ZymoBIOMICS Mock Community.

Funding

This work was supported by a grant from the BMBF/German Center for Infection Research [TI 06.904—FP2019 to B.Y.R.].

Conflict of Interest: J.-U.U. and B.Y.R. have filed a patent application on selective nanopore sequencing approaches.

References

Ahmed,O. *et al.* (2021) Pan-genomic matching statistics for targeted nanopore sequencing. *iScience*, **24**, 102696.

Blanca,A. *et al.* (2022) The statistics of k-mers from a sequence undergoing a simple mutation process without spurious matches. *J. Comput. Biol.*, **29**, 155–168.

Bloom,B.H. (1970) Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, **13**, 422–426.

Boža,V. *et al.* (2020) Deepnano-blitz: a fast base caller for minion nanopore sequencers. *Bioinformatics*, **36**, 4191–4192.

Brinkerhoff,H. *et al.* (2021) Multiple rereads of single proteins at single–amino acid resolution using nanopores. *Science*, **374**, eabl4381–1513.

Broder,A.Z. (1997) On the resemblance and containment of documents. In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE Comput. Soc. Salerno, Italy, pp. 21–29

Dadi,T.H. *et al.* (2018) Dream-yara: an exact read mapper for very large databases with short update time. *Bioinformatics*, **34**, i766–i772.

Koslicki,D. and Zabeti,H. (2019) Improving minhash via the containment index with applications to metagenomic analysis. *Appl. Math. Comput.*, **354**, 206–215.

Kovaka,S. *et al.* (2021) Targeted nanopore sequencing by real-time mapping of raw electrical signal with uncalled. *Nat. Biotechnol.*, **39**, 431–441.

Leggett,R.M. and Clark,M.D. (2017) A world of opportunities with nanopore sequencing. *J. Exp. Bot.*, **68**, 5419–5429.

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

Loose,M. *et al.* (2016) Real-time selective sequencing using nanopore technology. *Nat. Methods*, **13**, 751–754.

Marquet,M. et al. (2022). Evaluation of microbiome enrichment and host dna depletion in human vaginal samples using oxford nanopore's adaptive sequencing. Scientific reports, 12, 1–10.

Martin,S. *et al.* (2022) Nanopore adaptive sampling: a tool for enrichment of low abundance species in metagenomic samples. *Genome Biol.*, **23**, 11–27.

Mikheyev,A.S. and Tin,M.M. (2014) A first look at the oxford nanopore minion sequencer. *Mol. Ecol. Resour.*, **14**, 1097–1102.

Mongan,A.E. *et al.* (2020) Portable sequencer in the fight against infectious disease. *J. Hum. Genet.*, **65**, 35–40.

Nicholls,S.M. *et al.* (2019) Ultra-deep, long-read nanopore sequencing of mock microbial community standards. *Gigascience*, **8**, giz043.

Nurk,S. *et al.* (2022) The complete sequence of a human genome. Science , **376**, 44–53. https://doi.org/10.1126/science.abj6987.

Ondov,B.D. *et al.* (2016) Mash: fast genome and metagenome distance estimation using minhash. *Genome Biol.*, **17**, 1–14.

Ono,Y. *et al.* (2021) Pbsim2: a simulator for long-read sequencers with a novel generative model of quality scores. *Bioinformatics*, **37**, 589–595.

Payne,A. *et al.* (2019) Bulkvis: a graphical viewer for oxford nanopore bulk fast5 files. *Bioinformatics*, **35**, 2193–2198.

Payne,A. *et al.* (2021) Readfish enables targeted nanopore sequencing of gigabase-sized genomes. *Nat. Biotechnol.*, **39**, 442–450.

Piro,V.C. *et al.* (2017) Metameta: integrating metagenome analysis tools to improve taxonomic profiling. *Microbiome*, **5**, 1–11.

Piro,V.C. *et al.* (2020) ganon: precise metagenomics classification against large and up-to-date sets of reference sequences. *Bioinformatics*, **36**, i12–i20.

Quick,J. *et al.* (2016) Real-time, portable genome sequencing for ebola surveillance. *Nature*, **530**, 228–232.

Rang,F.J. *et al.* (2018) From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. *Genome Biol.*, **19**, 1–11.

Runtuwene,L.R. *et al.* (2019) On-site minion sequencing. *Single Mol. Single Cell Sequencing*, **1129**, 143–150.

Sim,J. and Chapman,B. (2019) In-field whole genome sequencing using the minion nanopore sequencer to detect the presence of high-prized military targets. *Aust. J. Forensic Sci.*, **51**, S86–S90.

Wick,R.R. *et al.* (2019) Performance of neural network basecalling tools for oxford nanopore sequencing. *Genome Biol.*, **20**, 1–10.

Zhou,M. *et al.* (2021) Comprehensive pathogen identification, antibiotic resistance, and virulence genes prediction directly from simulated blood samples and positive blood cultures by nanopore metagenomic sequencing. *Front. Genet.*, **12**, 620009.