Genome Biology

SOFTWARE **Open Access**

SPUMONI 2: improved classification using a pangenome index of minimizer digests

Omar Y. Ahmed[1*], Massimiliano Rossi[2], Travis Gagie[3], Christina Boucher[2] and Ben Langmead[1]

*Correspondence: oahmed6@jhu.edu

[1] Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA
[2] Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL, USA
[3] Faculty of Computer Science, Dalhousie University, Halifax, NS, Canada

Abstract
Genomics analyses use large reference sequence collections, like pangenomes or taxonomic databases. SPUMONI 2 is an efficient tool for sequence classification of both short and long reads. It performs multi-class classification using a novel sampled document array. By incorporating minimizers, SPUMONI 2's index is 65 times smaller than minimap2's for a mock community pangenome. SPUMONI 2 achieves a speed improvement of 3-fold compared to SPUMONI and 15-fold compared to minimap2. We show SPUMONI 2 achieves an advantageous mix of accuracy and efficiency in practical scenarios such as adaptive sampling, contamination detection and multi-class metagenomics classification.

Keywords: Pangenome, Indexing, Classification, Minimizer

Background

Read classification is a component of many sequencing data analyses, such as taxonomic classification [1–3], host sequence depletion [4, 5], and adaptive sampling of nanopore reads [6, 7]. Databases holding reference sequences are growing rapidly [8, 9], enabling pangenomic methods that use a collection of related genomes as the reference, rather than a single genome. We previously described SPUMONI [10], a method for rapid binary classification of nanopore reads against a pangenome reference. SPUMONI builds on the r-index [11, 12], a compressed index that grows with the amount of distinct sequence in the reference pangenome. It uses the MONI algorithm [13] to compute *matching statistics* (defined below) as input to its classification decision. Though SPUMONI was faster and used less memory than a minimap2-based approach, we since sought ways to extend its functionality to (a) analyze short as well as long reads, (b) perform multi-class as well as binary classification, and (c) scale efficiently to larger pangenomes.

We present SPUMONI 2, which uses a minimizer scheme to digest and reduce both the pangenome reference and the input reads. By changing the alphabet to consist of all possible minimizers, rather than all possible bases, SPUMONI 2 queries are faster than

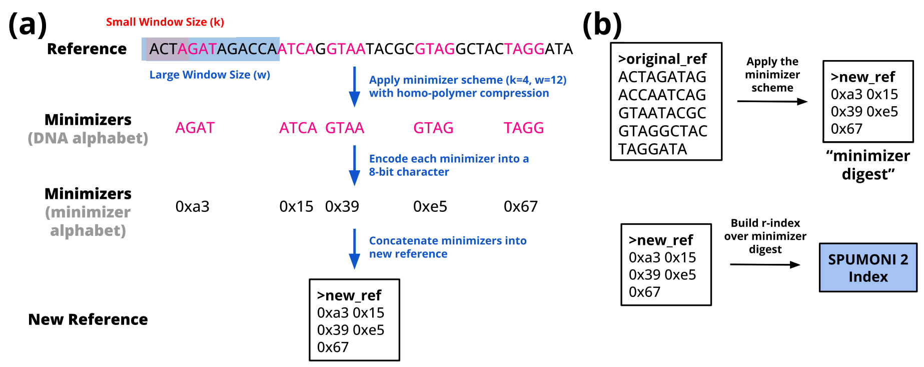
(a)
Small Window Size (k)
Reference ACTAGATAGACCAATCAGGTAATACGCGTAGGCTACTAGGATA
Large Window Size (w)
Apply minimizer scheme (k=4, w=12) with homo-polymer compression
Minimizers (DNA alphabet) AGAT ATCA GTAA GTAG TAGG
Encode each minimizer into a 8-bit character
Minimizers (minimizer alphabet) 0xa3 0x15 0x39 0xe5 0x67
Concatenate minimizers into new reference
New Reference >new_ref 0xa3 0x15 0x39 0xe5 0x67
(b)
>original_ref ACTAGATAG ACCAATCAG GTAATACGC GTAGGCTAC TAGGATA
Apply the minimizer scheme
>new_ref 0xa3 0x15 0x39 0xe5 0x67
"minimizer digest"
Build r-index over minimizer digest
SPUMONI 2 Index

Fig. 1 **a** Shows the procedure used by SPUMONI 2 to digest a reference into a smaller reference of concatenated minimizers. In practice, SPUMONI 2 would also include the reverse complement of each sequence prior to applying the minimizer scheme. **b** After generating this minimizer digest which is typically smaller than the original reference, SPUMONI 2 builds an *r*-index over the minimizer digest which in turns leads to a smaller index

SPUMONI's. Moreover, the approach of SPUMONI 2 works in any situation where a read that truly originates from the reference pangenome yields longer exact substring matches than a read that does not. SPUMONI 2 works with long or short reads and does not require the user to have foreknowledge of what *k*-mer length is capable of distinguishing true from false hits.

Both SPUMONI and SPUMONI 2 build upon the MONI algorithm [13] for computing matching statistics. The *matching statistics* of a pattern string with respect to a collection of reference strings is an array storing the length of the longest prefix of the pattern's i^{th} suffix that occurs in the reference. For faster queries, SPUMONI 1 and 2 compute an approximation of matching statistics called *pseudomatching lengths* (PMLs). See the "Methods" section for formal definitions.

Minimizers [14] are a form of locality-sensitive hashing, applied to windows of a string. A minimizer scheme defines a small window size (k) and a large window size (w); within each length-w window, the constituent k-mer with the minimal hash value is chosen as the minimizer. They have been used to create small genomic and pangenomic indexes [1, 5]. In some settings, minimizers are used to *digest* a long sequence, replacing it with a concatenation of its minimizers (with equal-minimizer runs collapsed). This can, for instance, reduce the size of assembly graph representations [15, 16].

SPUMONI 2 first creates a minimizer digest with configurable small (k) and large (w) window sizes, then builds the *r*-index over the digest. When indexing the digest, SPUMONI 2 considers the alphabet to consist of all possible choices of minimizer (i.e., all possible k-mers) [15, 16]. That is, it uses a minimizer alphabet rather than a DNA alphabet. This poses no problem to the *r*-index, which can adapt to any discrete alphabet. This combined strategy of indexing the minimizer digest and using a minimizer alphabet (see Fig. 1), has the effect of reducing index size, memory footprint, and query time.

Finally, SPUMONI 2 includes a new *sampled document array* structure, allowing for multi-class classification by relating the runs in the *r*-index to a representative document in the input collection. The user provides a list of genomes along with a class assignment for each at construction time. The sampled document array is small — growing linearly with the size of the index — but allows the user to infer class membership during the computation of matching statistics. Though the sparsity of the structure adds uncertainty to these assignments, this can be compensated for by aggregating assignments along the read.

In short, SPUMONI 2 is a new method and software tool that creates small pangenomic indexes from large collections of genomes. SPUMONI 2 combines the compression of the *r*-index and the adjustable sparsity of minimizers to greatly reduce the index size; for example, indexing 10 human genome sequences along with their reverse complements in 4.2 GB. The classification test used by SPUMONI 2 yields high binary classification accuracy on both short and long reads. Additionally, SPUMONI 2 enables multi-class classification through a new *sampled document array* structure that scales linearly with the amount of distinct sequence. Finally, when comparing SPUMONI 2 to a minimap2-based approach for ONT adaptive sampling in a mock community scenario, SPUMONI 2 is 15 times faster using an index more than 68 times smaller.

Results

Method overview

Like SPUMONI, SPUMONI 2 classifies reads according to the lengths of the matching statistics (MSs) computed with respect to an index of reference sequences. In particular, SPUMONI 2 computes an empirical distribution of "null" matching statistics at index construction time, identifying the longest null MS occurring at least a certain number [five] of times. SPUMONI 2 divides the read into non-overlapping windows of 150 symbols each, starting at the left-end of the read, and classifies the read as "present" in the database if a majority of the windows have a maximum MS longer than the null threshold. As with SPUMONI, SPUMONI 2's default mode actually computes pseudomatching lengths (PMLs), a fast approximation of matching statistics (MSs) that have similar discriminatory power. Further details are given in "Methods" section.

Minimizer digestion

To assess the impact of minimizer digestion, we measured the size of the SPUMONI 2 index when indexing a collection of *Eschericha coli* strains using two minimizer schemes and two alphabets. We chose to index 500 *E. coli* strains from the RefSeq database [17].[1] A minimizer scheme is parameterized by a short window size (k) and a long window size (w). We assessed the (k, w)-pairs of (4, 8) and (4, 16). The pair (4, 8) was chosen to yield a digested index with a similar size (r) compared to the undigested index. The pair (4, 16) was chosen to allow us to measure the impact of minimizer sparsity on index size. We assessed two choices of alphabet: a minimizer alphabet, where each possible 4-mer is a distinct alphabet symbol (with k=4, there are 256 possible k-mers which can be encoded in a 8-bit character), and a typical DNA alphabet with 4 nucleotide symbols. The digested reference was obtained by scanning the reference sequence, computing the minimizers, collapsing equal-minimizer runs, then concatenating the minimizers, either leaving them in the original DNA alphabet or converting them to the minimizer alphabet in the process (Fig. 1).

[1] We give the accessions numbers in the supplemental data.

Table 1 SPUMONI index measurements when built over 500 *E. coli* strains (total size 2.5 GB) using different minimizer schemes and alphabets. Both minimizer schemes used a small window size (*k*) of 4. The forward and reverse complement for each *E. coli* genome was included in the index

w	**No digestion**	***w* = 8**		***w* = 16**	
Alphabet		**Minimizer**	**DNA**	**Minimizer**	**DNA**
n	5.13 billion	1.61 billion	6.78 billion	663 million	2.85 billion
r	65.2 million	31.8 million	64.4 million	17.5 million	35.1 million
n/*r*	78.753	50.603	105.35	37.888	81.251
Index size	232 MB	142 MB	227 MB	73 MB	124 MB

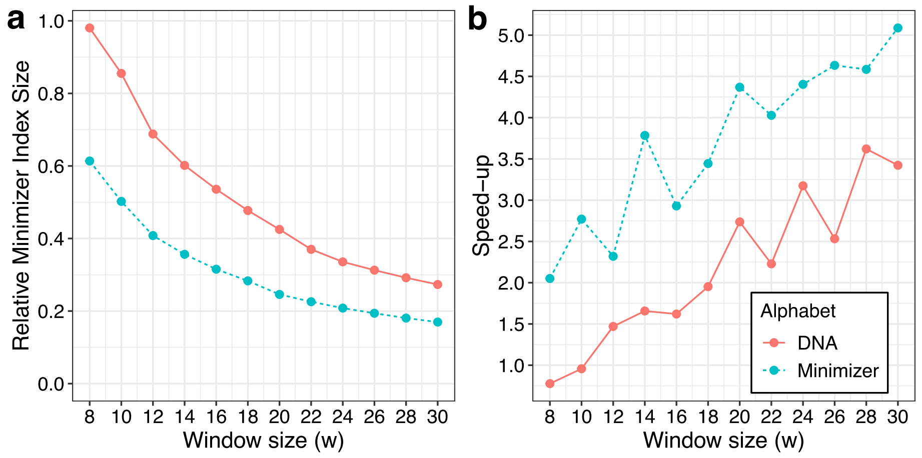

Fig. 2 **a** Shows the relative size of the minimizer-based SPUMONI indexes across a range of large window sizes compared to the index size when indexing the full FASTA file. The dataset indexed is a set of 500 *Escherichia coli* genomes, and the small window size was kept at 4. **b** Shows the speed-up achieved when using the minimizer-based indexes to query 1 million short E. coli reads [18] against our index compared to querying against an index over the full FASTA file

We measured both the length of the digested reference (n) and the number of runs in its Burrows-Wheeler Transform (r), which is the main determinant of index size.

We observed that the compressed indexes of the digested sequences continued to have large compression ratio (n/r), indicating that compressed indexing is still effective on minimizer-digested strings (Table 1). In the case of the (4, 8) minimizer scheme, the value of n after digestion was greater than than the n before digestion (6.78 versus 5.13 billion), but digestion had little effect on r (64.4 after digestion versus 65.2 million before). Consistent with expectations, index size was smaller for sparser minimizer schemes.

As seen in Fig. 2a, SPUMONI 2's index decreased in size as w increased. Also, query speed-up was nearly always greater than 1 (i.e., faster than the original index) and also increased with w, though with some variability. The minimizer-alphabet indexes (green) always outperformed DNA-alphabet indexes (red) with respect to both index size and speed-up.

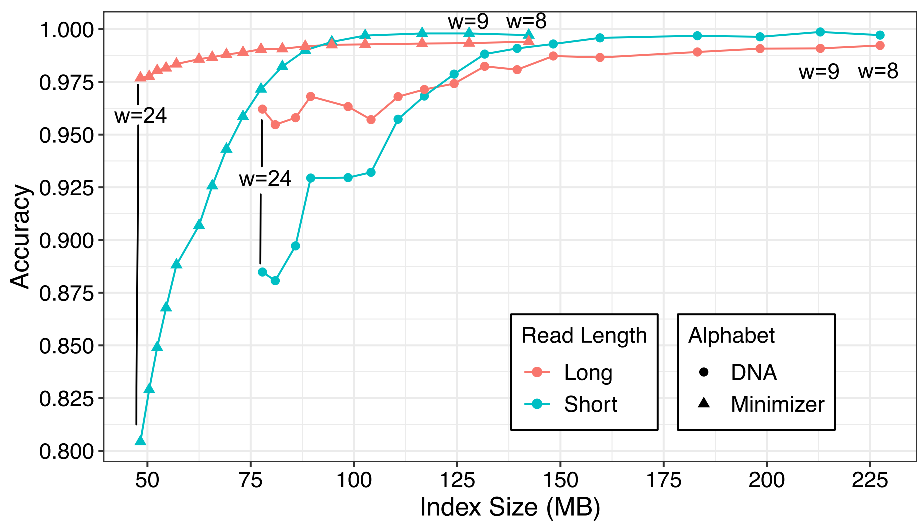

Fig. 3 Shows SPUMONI's binary classification accuracy for indexes of different sizes using different minimizer types. The index was built over 500 *Escherichia coli* genomes and used minimizer schemes where the large window size ranged from every integer value from 8 to 24. The read set consisted of simulated ONT (mean length = 9000 bp, 95% accuracy) [19] and Illumina reads (150 bp, 99% accuracy) [18] from *E. coli* and Human. The goal was to classify whether the read was from *E. coli* or Human

Efficient short and long-read binary classification

To assess how minimizer digestion impacts classification accuracy, we ran SPUMONI 2 with various large window sizes (w), using either DNA or minimizer alphabets. We found that the minimizer alphabet allowed SPUMONI 2 to attain high accuracy — as high as the highest accuracy achieved by the DNA alphabet — while yielding a smaller index (Fig. 3). While accuracy decreased as the minimizer scheme became sparser (i.e., moving from right to left on the plot), the decrease was less drastic for long reads compared to short reads. For both short and long reads, we setting w to a value between 10 and 12 decreased index size with only a minor impact on accuracy. Given these results, we set SPUMONI 2's default minimizer scheme to use $w = 11$, which will be used in future results.

Adaptive sampling classification using SPUMONI 2

As a more challenging test case for minimizer digestion, we assessed SPUMONI 2's classification accuracy in an adaptive sampling setting. More specifically, we replicated the mock-community simulation experiment from the original SPUMONI study [10]. We compared SPUMONI to a minimap2-based approach similar to Readfish [7]. Classification tools were run on four "batches" consisting of non-overlapping windows of 180 bp from the reads, starting from the read's left end. Once the tool can make a classification decision, typically after examining the first or second batch, we record the decision and assess accuracy. Experimental details are given in the "Methods" section.

SPUMONI 2 was able to index both pangenomic databases in about half the space as SPUMONI 1, and achieved a 2-fold speed-up with respect to classification time (Table 2). Because we also measured performance of SPUMONI 2 with minimizer digestion disabled (labeled SPUMONI 2[a]), we can conclude that these improvements were due to minimizer digestion. Compared to minimap2 on the Mock Community dataset,

Table 2 Adaptive sampling simulation using SPUMONI 1, SPUMONI 2 and minimap2. SPUMONI 1 indexes the full input database, while SPUMONI 2 indexes the minimizer-digested sequences of the database using the minimizer alphabet. The "SPUMONI 2 [a]" gives measurements for SPUMONI 2 with minimizer digestion disabled. Batches of 180 bp (0.4s) of data are delivered in each batch, and the goal is to decide whether to eject the read or not. Four batches were considered in the analysis which corresponds to 720 bp. The mock community dataset of ONT reads (SRX7711546) consists of reads from 7 microbial species and 1 yeast species. The goal is to retain the yeast reads and eject the microbial reads. For the human microbiome study, bacterial reads from the microbiome were obtained the following SRA accession (SRX6602475) and human reads were simulated [19] from the CHM13 reference

Scenario	**Mock community**				**Human microbiome**			
Goal	**Retain yeast, eject microbial**				**Retain microbial, eject human**			
Index database	**7 microbial species (n=5867 genomes)**				**Human (n=10 genomes)**			
Tool	**SPUMONI 1**	**SPUMONI 2[a]**	**SPUMONI 2**	**minimap2**	**SPUMONI 1**	**SPUMONI 2[a]**	**SPUMONI 2**	**minimap2**
Sensitivity	97.38	97.62	95.32	97.90	99.24	95.07	97.08	99.56
Specificity	90.77	97.38	97.06	97.85	94.30	99.97	99.12	99.97
Index size	1.54 GB	1.54 GB	0.74 GB	50.9 GB	10.2 GB	10.2 GB	4.21 GB	65.5 GB
Peak memory	1.62 GB	1.62 GB	0.80 GB	8.40 GB	11.0 GB	11.0 GB	4.56 GB	9.99 GB
Time (s)	367.39	362.89	193.53	2957.56	1628.7	1747.12	732.63	3070.0

[a] Running SPUMONI 2 without minimizer digestion (i.e., similar to SPUMONI 1 but using new classification approach)

SPUMONI 2 was about 15 times faster with an index more than 68 times smaller (0.74 GB versus 50.9 GB). On the Human Microbiome dataset, the SPUMONI 2's index was about 15 times smaller than minimap2's, and SPUMONI 2 was about 4 times faster at classification. The SPUMONI 2 pangenome index comprising of 10 human genome sequences and their reverse complements fit in about 4.2 GB.

Contamination detection in assemblies

We expected that SPUMONI 2 would be able to quickly scan for the presence of contaminants in human genome assemblies. NCBI, which curates the GenBank and RefSeq databases, uses a series of filtering protocols to identify contamination, e.g., by using BLAST [20] to align contigs against common contaminants like *Eschericha coli* and yeast.

We built the SPUMONI 2 index over a pangenome of *Eschericha coli* and yeast genomes (Additional file 1: Table S1). The SPUMONI 2 index was 104 MB, 24 times smaller than the total size of the input FASTA files (2.5 GB). We used the contigs from human assemblies as input to SPUMONI 2 to identify contigs with long pseudomatching lengths (PMLs) with respect to the contaminant pangenome. SPUMONI 2 found four contigs in one human assembly [21] where the 25th percentile of the PML distribution was 2 or greater (Fig. 4a). When comparing these four to all the other contigs, we observed a stark difference in distribution of PMLs. Figure 4b uses minimap2 alignments to confirm that these contigs have large substrings matching with high identity to the contaminant database. These were reported to the authors of the assembly [21], who verified these findings and removed the contigs from the v2.0 assembly as of December 17, 2021.

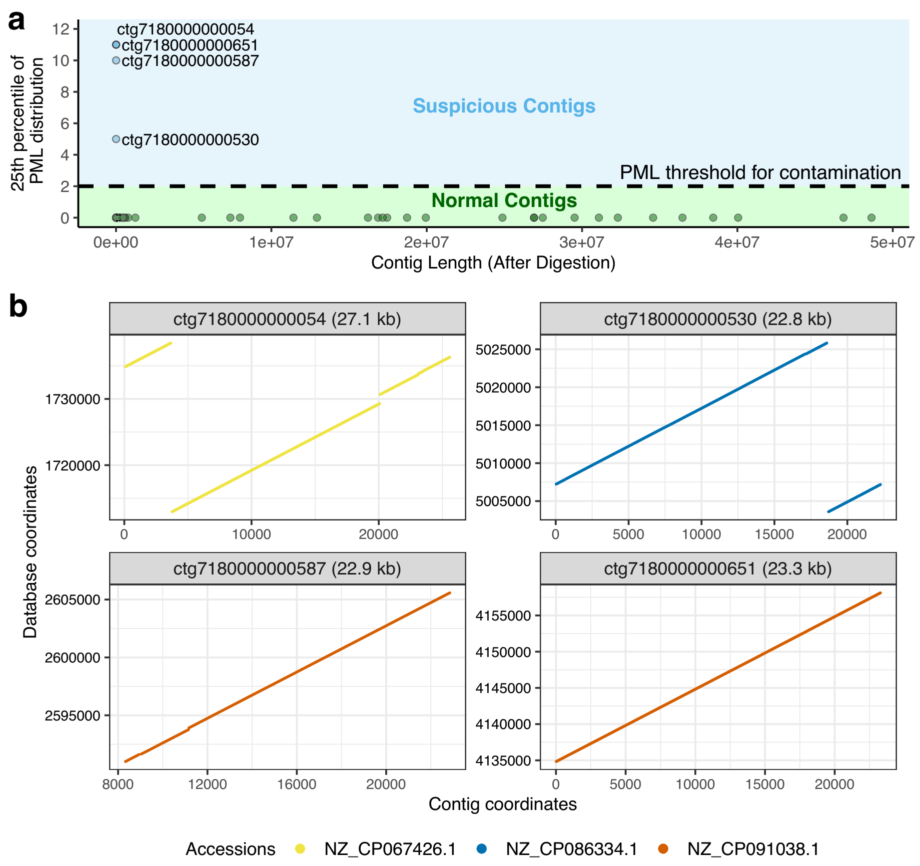

Fig. 4 **a** Scatter plot of the 25th percentile of the PML distribution for each contig in a human assembly [21] with respect to a SPUMONI 2 index of contaminants. Contigs where the 25th percentile of its PML distribution is 2 or greater are labeled "Suspicious" and if it is less than 2, the contig is labeled "Normal." **b** dot plots showing high-scoring local alignments found with minimap2 [5] for the four suspicious contigs versus sequences in the contaminant pangenome. The suspicious contigs were reported and moved from the assembly in December, 2021

We compared SPUMONI 2's speed to that of BLAST+, which is used by NCBI for assembly filtering. SPUMONI 2 was 4.3 times faster while identifying the same suspicious contigs (Additional file 1: Table S3).

Small multi-class classification using the sampled document array

We assessed SPUMONI 2's sampled document array in a multi-class classification setting. As SPUMONI 2 computes matching statistics at each position of the read, it queries the sampled document array to obtain a class label for one document (sequence) containing the current match. The process involves uncertainty, since the particular class reported by the sampled document array may be just one among many that contain the match (see the "Sampled document array" section in the methods for details). We hypothesized that, by aggregating the classes reported over the course of all the steps of the algorithm, we can infer the correct class of origin.

We used SPUMONI 2 to construct a pangenomic index over all of the genomes in RefSeq for eight different microbial species (details in Additional file 1: Table S2). We

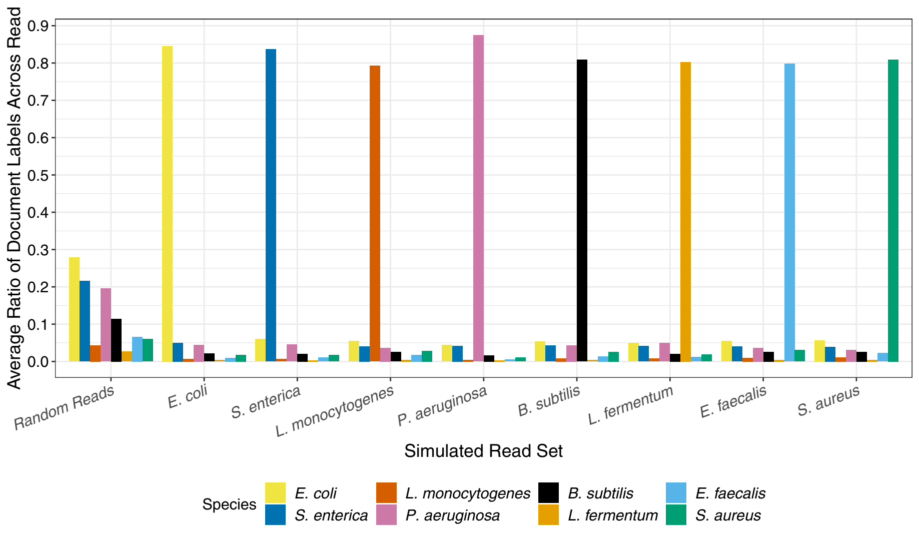

Fig. 5 Average ratio of class labels found at the read level when matching Illumina reads (150 bp) simulated from eight microbial species. The SPUMONI 2 index consisted of over 6000 reference genomes for the eight microbial species

observed that the sampled document array increased the size of the index from 793 MB to 920 MB, a 16% increase.

Figure 5 shows that when querying simulated Illumina reads [18] from eight different microbial species against a pangenomic database, we find that between 80.7 and 88.0% of the classes reported by the sampled document array at the read level match the true class of origin. For long ONT reads, we see a similar pattern where the true class is in the majority, ranging from 53.8 to 66.3% of the labels across the eight classes (Additional file 1: Fig. S1).

When queried with randomly-generated reads consisting of independent and uniform draws from the DNA alphabet, no class label exceeded 28.2% frequency for short reads (Fig. 5). We observed a tendency for some classes to be reported more often than others (e.g., *E. coli*), likely because of differences in the total number of indexed bases of each class (Additional file 1: Table S2).

Discussion

SPUMONI 2 is a read classification tool optimized for classifying both short and long reads against a pangenomic database. By combining minimizer digestion with the *r*-index, SPUMONI 2 compresses the pangenome more efficiently than SPUMONI, allowing it to handle larger databases. Compared to minimap2, SPUMONI 2's index is far smaller (1/68th of minimap2's in the case of our mock community experiment) and it performs classification up to 15 times faster. While SPUMONI 2's computational efficiency comes at the expense of some classification accuracy, this can be a favorable trade in situations where the pangenome is very large (e.g., minimap2's index of 10 human genome assemblies is over 50 GB) or where reaction time is important, e.g., in an adaptive sampling setting where the algorithm has to keep up with DNA strings moving through many hundreds of pores simultaneously.

SPUMONI 2 includes a novel sampled document array structure whose space usage achieves the same $O(r)$ bound as the overall r-index. This allows our matching-statistics-based classification method to work in multi-class classification settings, expanding its applicability to metagenomics classification.

While the sampled document array was effective in the long-read classification scenario studied here, it is limited by the fact that it records only one "document" (class) per r-index run boundary. A more general approach would additionally allow us to query all classes containing the current match, not just the class corresponding to the run boundary. Such approaches exist for uncompressed indexes, but more work is needed to adapt these to work in compressed space and time. This will be particularly important for short-read classification, where we cannot aggregate as many document array queries.

Since classification decisions are ultimately influenced by the particular choice of hash function, certain sequences might be systematically more or less likely to be represented in the digest. While we do not expect this to favor any particular biologically meaningful class of sequence, any potential bias could be addressed by choosing many hash functions with random inputs and repeating the analysis.

A related point can be made about how the minimizer scheme deals with tandem repeats. When the repeated unit is repeated perfectly and is small compared to the large window size, the minimizer sequence will remain the same over the entire span of the repeat. Our method will compress the consecutive string of equal minimizers to a single minimizer, losing any notion of the tandem repeat's length. In the future, it will be important to study alternatives to our compression scheme that are able to retain more information about tandem repeats and their lengths.

Finally, we demonstrated a novel application of SPUMONI 2 to detecting contaminants in genome assemblies. This represents another application where, as reference databases continue to grow, SPUMONI 2 will be well positioned to leverage the additional assemblies with minimal impact to index size or classification speed. On the other hand, larger genomic databases have been shown to affect k-mer based approaches due to decreasing levels of k-mer specificity [22].

Conclusion

We present SPUMONI 2 as an efficient tool for sequence classification for both short and long reads with respect to large reference collections. Its usage of minimizers along with the r-index yields better compression compared to SPUMONI as well as contributing to a 3-fold speedup. In addition, it incorporates a novel data-structure called the sampled document array that enables efficient multi-class classification of reads against pangenomes. We have shown a series of biological analyses such as adaptive sampling, contamination detection and multi-class metagenomics classification where SPUMONI 2 provides a strong balance between accuracy and efficiency. In conclusion, SPUMONI 2 is an efficient solution for read classification with respect to large genomic databases with a wide array of potential applications.

Methods

Compressed indexes

Given a text $T[1..n]$ of length n, the Burrows-Wheeler transform (BWT) [23] is a reversible permutation of T's characters such that BWT[i] is the character preceding the i^{th} lexicographical suffix of T. The BWT of a repetitive text will have long runs of the same character. The symbol r denotes the number of maximal same-character runs in the BWT.

The r-index [24] is a compressed index consisting of the run-length encoded BWT, plus auxiliary structures enabling rapid queries for counting the number of times a substring occurs in T, as well as for locating all offsets in T where a substring occurs. Only $O(r)$ space is needed overall. More details can be found in the study of Gagie et al [11].

Matching statistics and MONI

The matching statistics array $\mathsf{MS}[1..m]$ is defined for a pattern $P[1..m]$ with respect to a text $T[1..n]$. The element $\mathsf{MS}[i]$ equals the length of the longest prefix of P's i^{th} suffix that occurs in T. Using the r-index, Bannai et al. [25] proposed a two-pass algorithm for computing MS using an additional "thresholds" structure. The algorithm's first pass iterates over P from right to left, attempting to use the LF mapping at each step to extend the match to the left by one character. If the algorithm reaches a row j and finds that match cannot be extended because the next character of P mismatches $\mathsf{BWT}[j]$, the algorithm skips ("jumps") either up or down to the next BWT run that does start with the next character of P. The direction of the jump is determined by the threshold, which indicates whether jumping up or down yields a longer match. Given the steps taken through the BWT in the first pass of the algorithm, the second pass uses a random-access data-structure over T to compute the exact matching statistic lengths for MS. More details can be found in the study of Rossi et al [13].

Pseudomatching lengths and SPUMONI

Pseudomatching lengths (PMLs) were proposed in the SPUMONI study as being a quantity similar to matching statistics but more efficient to compute. SPUMONI's computation of PMLs does not require the second pass of the algorithm described above. Instead, each step of the first pass that successfully extends the match using the LF mapping causes a length variable to be incremented by one. When the algorithm reaches a step where it cannot proceed using the LF mapping and has to jump instead, the length variable is reset to 0. The values taken by this variable at each step constitutes the vector of PMLs (PML). The PML values are upper-bounded by the MS values, but we have shown that they are similarly useful for classification of sequences [10].

The SPUMONI index consists chiefly of the r-index (omitting the sampled suffix array) and the thresholds. These structures allow SPUMONI to compute PML in $O(r)$

space. Because the r-index component does not need to include the sampled suffix array, SPUMONI's disk and memory footprint can be substantially smaller than that of the MONI algorithm described in the previous subsection.

Minimizer digestion

SPUMONI 2 computes minimizers by applying a hash function to all k-mers in a larger window of size w, $w > k$. The k-mer with minimal hash value is the minimzer. If the minimizers found in two or more consecutive steps are identical, they are compressed to a single copy of the minimizer. SPUMONI 2 uses the RollingHasher object provided by the Bonsai C++ library [26].

These minimizer sequences are then indexed by using the r-index [11] to build a run-length encoded BWT and thresholds data-structure.

Read classification

The previous SPUMONI method classified reads by accumulating an empirical distribution of positive PMLs (with respect to the reference), and another of null PMLs (with respect to the reverse of the reference) [10]. It used a Kolmogorov-Smirnov statistical test (KS-test) to assess whether the distribution of positive PMLs was overall larger (shifted higher) compared to the null PMLs.

SPUMONI 2 uses a simpler approach; it also accumulates empirical distributions of positive and null PMLs, but it classifies reads by first computing a threshold PML value as a function of the null PML distribution. The null PML distribution is constructed by extracting small reads (substrings) from the reference, reversing the sequence, then computing PMLs for those sequences with respect to the reference. Since the sequences are reversed (not reverse complemented), they serve as random sequences that should not be classified as matching the reference, but they nonetheless share the reference's base distribution. Pooling across simulated reads, we compile an aggregate distribution of null PMLs and compute the threshold as being equal to the largest PML that occurs at least 5 times. Because the process of computing the threshold uses the same parameters as the read-classification process (i.e., same minimizer scheme and alphabet), the choice of threshold is customized to the problem at hand.

Given the threshold PML value, SPUMONI 2 classifies a read by computing the read's positive PMLs with respect to the reference index in many distinct non-overlapping windows of the read. By default, the read is divided into non-overlapping windows of 150 symbols. We found this window size performs well in terms of accuracy across different alphabets (minimizers or DNA). The windows at the end of the read that are less than 150 symbols are grouped together with the previous window. If a majority of the windows have a maximum PML greater than the threshold, the read is classified as matching the reference, otherwise it is classified as not matching.

Adaptive sampling simulation

Nanopore sequencers report a time series representing the amount of current flowing through a pore. The data is split into batches which are delivered to the control software for analysis. Following our experimental setup from the SPUMONI study [10], we simulated 4 batches of 0.4 s of basecalled nanopore data, with each batch consisting of 180 bp each. Batches of this size were shown to be sufficient for binary classification method [5, 7]. We start from base-called data, rather than from raw signal data.

For our adaptive sampling simulation, we simulated two different scenarios where adaptive sampling could be used to enrich for certain reads. The first scenario is a mock community which consists seven bacterial species (*Staphylococcus aureus*, *Salmonella enterica*, *Escherichia coli*, *Pseudomonas aeruginosa*, *Listeria monocytogenes*, *Enterococcus faecalis*, *Bacillus subtilis*) and 1 yeast species (*Saccharomyces cerevisiae*). The goal in this scenario is to enrich for the yeast reads by rejecting bacterial reads. To accomplish this, we built a pangenome reference over all the strains of the seven bacterial species in RefSeq [17] to identify which reads come from bacteria.

The second scenario is focused on the human microbiome where we used a 50:50 mixture of simulated human reads from the CHM13 reference [27], and real nanopore reads from the microbial species in the human gut [28]. The goal in this experiment is to enrich for the microbial reads while removing as many human reads as possible. For this scenario, we built a pangenome reference over 10 human assemblies [27, 29–36] in order to identify human reads that we want to remove.

We use both SPUMONI and minimap2 to classify batches against the pangenome index. A read classified as "present" should be immediately ejected by the pore, so subsequent batches of data are not processed. SPUMONI 2 classifies the current batch of 180 bp using the test described in "Read Classification" methods section. In the case of minimap2, aligning against a pan-genomic database usually yields numerous alignments: a primary and many secondary alignments. We examine these to ensure all are to the same species and, if so, we classify the read as "present." For any batch after the first, we provide the entire base-called read so far to minimap2; i.e. we concatenate the bases from all the batches so far. In this way, minimap2 is redundantly processing the same bases when examining batches beyond the first. This is consistent with minimap2's design; unlike the *r*-index-based algorithms that have the ability to "pause" and "resume" the matching process, minimap2 must be run on a complete read sequence.

The time required for each method is reported in Table 2; this is the total time used by each method to classify the reads across all 4 batches of data.

Sampled document array

SPUMONI 2's sampled document array consists of $2r$ integers encoding the class of the suffixes at the beginning and end of every BWT run. The class labels are computed at index construction time. Note that the class labels can be stored in $\lceil \log_2 c \rceil$ bits, where c is the number of class labels, which willl often be much less than the $\lceil \log_2 n \rceil$ bits required for suffix array entries. An example is shown in Fig. S4 (Additional file 1).

At query time, the sampled document array is queried whenever the algorithm for computing PMLs passes through a suffix at a run boundary. As described in detail in the MONI paper [13], the algorithm proceeds base-by-base starting from the right-hand extreme of the read. Each step falls into one of two cases. When the match can be extended to the left using the LF mapping, this is called "case 1." When the match cannot be extended to the left using the LF mapping, the algorithm uses threshold information to choose a new BWT run to "jump" to. This is "case 2." Importantly, case 2 always results in the algorithm moving to a run boundary. That is, when the algorithm jumps, it jumps either to the beginning of a run (if it jumps down) or the end of a run (if it jumps up).

When using the sampled document array, the SPUMONI 2 algorithm compiles not only its usual array of PMLs, but also an array of document (class) labels, CA. At position j in the read where the algorithm uses case 2, the label CA[j] equals the sampled document array element corresponding to the run boundary jumped to. For a position j in the read where the algorithm uses case 1, the label CA[j] equals the class observed in the most recent instance of case 2. When the read truly originates from one of the classes in the reference pangenome, we expect the majority of the labels in CA to match the true class. This algorithm is summarized in the form of pseudocode in Fig. S2 (Additional file 1).

Abbreviations

MS	Matching statistics
PML	Pseudomatching lengths

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13059-023-02958-1.

Additional file 1. SPUMONI 2 Supplement. This document contains supplementary tables and figures.

Additional file 2. Review history.

Acknowledgements
This work was carried out at the Advanced Research Computing at Hopkins (ARCH) core facility (rockfish.jhu.edu), which is supported by the National Science Foundation (NSF) grant number OAC 1920103.

Peer review information
Andrew Cosgrove was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Review history
The review history is available as Additional file 2.

Authors' contributions
OA and BL conceived the project and designed SPUMONI 2. OA implemented the SPUMONI 2 software with assistance from BL and MR. All authors contributed to the design of the sampled document array structure. All authors wrote, edited and approved the manuscript.

Funding
OA, MR, TG, CB, and BL were supported by NIH/NHGRI grant R01HG011392 to BL. OA, MR, TG, CB, and BL were supported NSF/BIO grant DBI-2029552 to CB, TG and BL. TG was supported by NSERC grant RGPIN-07185-2020 to TG.

Availability of data and materials
• The SPUMONI 2 software [37] is available from GitHub at https://github.com/oma219/spumoni and is released under a GPL-3.0 license.
• SPUMONI 2 (v2.0.6) [37] used in the manuscript can be found at https://doi.org/10.5281/zenodo.7867076.
• The code used to run the experiments is available at https://github.com/oma219/spumoni-experiments.
• FASTA files for the E. coli genomes assessed in the "Minimizer digestion" and "Efficient short and long-read binary classification" sections are available at https://genome-idx.s3.amazonaws.com/spu2/ecoli_500_dataset.tar.gz [38].
• SPUMONI 2 index files for the mock community pangenomes assessed in the "Adaptive sampling classification using SPUMONI 2" section are available at https://genome-idx.s3.amazonaws.com/spu2/mock_community_ont_index.tar.gz [39].
• SPUMONI 2 index files for the human assembly pangenome assessed in the"Adaptive sampling classification using SPUMONI 2" section are available at https://genome-idx.s3.amazonaws.com/spu2/human_pangenome_ont_index.tar.gz [40].
•SPUMONI 2 index files for the assembly contaminant pangenome used in the "Small multi-class classification using the sampled document array" section are available at https://genome-idx.s3.amazonaws.com/spu2/assembly_contamination_index.tar.gz [41].
• SPUMONI 2 index files, including the sampled document array, for the experiments described in the "Small multi-class classification using the sampled document array" section are available at https://genome-idx.s3.amazonaws.com/spu2/sampled_doc_array_index.tar.gz [42].

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 12 September 2022 Accepted: 3 May 2023
Published online: 18 May 2023

References
1. Wood DE, Lu J, Langmead B. Improved metagenomic analysis with Kraken 2. Genome Biol. 2019;20(1):1–13.
2. Kim D, Song L, Breitwieser FP, Salzberg SL. Centrifuge: rapid and sensitive classification of metagenomic sequences. Genome Res. 2016;26(12):1721–9.
3. Menzel P, Ng KL, Krogh A. Fast and sensitive taxonomic classification for metagenomics with Kaiju. Nat Commun. 2016;7(1):1–9.
4. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012;9(4):357–9.
5. Li H. Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics. 2018;34(18):3094–100.
6. Kovaka S, Fan Y, Ni B, Timp W, Schatz MC. Targeted nanopore sequencing by real-time mapping of raw electrical signal with UNCALLED. Nature Biotechnol. 2021;39(4):431–41.
7. Payne A, Holmes N, Clarke T, Munro R, Debebe BJ, Loose M. Readfish enables targeted nanopore sequencing of gigabase-sized genomes. Nat Biotechnol. 2021;39(4):442–50.
8. Li W, O'Neill KR, Haft DH, DiCuccio M, Chetvernin V, Badretdin A, et al. RefSeq: expanding the Prokaryotic Genome Annotation Pipeline reach with protein family model curation. Nucleic Acids Res. 2021;49(D1):D1020–8.
9. Sayers EW, Bolton EE, Brister JR, Canese K, Chan J, Comeau DC, et al. Database resources of the national center for biotechnology information. Nucleic Acids Res. 2022;50(D1):D20–6.
10. Ahmed O, Rossi M, Kovaka S, Schatz MC, Gagie T, Boucher C, et al. Pan-genomic matching statistics for targeted nanopore sequencing. Iscience. 2021;24(6):102696.
11. Gagie T, Navarro G, Prezza N. Fully functional suffix trees and optimal text searching in BWT-runs bounded space. J ACM (JACM). 2020;67(1):1-54.
12. Kuhnle A, Mun T, Boucher C, Gagie T, Langmead B, Manzini G. Efficient construction of a complete index for pan-genomics read alignment. J Comput Biol. 2020;27(4):500–13.
13. Rossi M, Oliva M, Langmead B, Gagie T, Boucher C. MONI: A Pangenomic Index for Finding Maximal Exact Matches. J Comput Biol. 2022;29(2):169–87.
14. Roberts M, Hayes W, Hunt BR, Mount SM, Yorke JA. Reducing storage requirements for biological sequence comparison. Bioinformatics. 2004;20(18):3363–9.
15. Ekim B, Berger B, Chikhi R. Minimizer-space de Bruijn graphs: Whole-genome assembly of long reads in minutes on a personal computer. Cell Syst. 2021;12(10):958–68.
16. Coombe L, Nikolić V, Chu J, Birol I, Warren RL. ntJoin: Fast and lightweight assembly-guided scaffolding using minimizer graphs. Bioinformatics. 2020;36(12):3885–7.
17. O'Leary NA, Wright MW, Brister JR, Ciufo S, Haddad D, McVeigh R, et al. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. Nucleic Acids Res. 2016;44(D1):D733–45.
18. Holtgrewe M. Mason: a read simulator for second generation sequencing data. Technical Reports of Institut für Mathematik und Informatik, Freie Universität Berlin. 2010.
19. Ono Y, Asai K, Hamada M. PBSIM2: a simulator for long-read sequencers with a novel generative model of quality scores. Bioinformatics. 2021;37(5):589–95.
20. Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, et al. BLAST+: architecture and applications. BMC Bioinformatics. 2009;10(1):1–9.
21. Shumate A, Zimin AV, Sherman RM, Puiu D, Wagner JM, Olson ND, et al. Assembly and annotation of an Ashkenazi human reference genome. Ashkenazi assembly. GitHub. https://github.com/AshkenaziGenome/Assembly/. Accessed Dec 2021.
22. Nasko DJ, Koren S, Phillippy AM, Treangen TJ. RefSeq database growth influences the accuracy of k-mer-based lowest common ancestor species identification. Genome Biol. 2018;19(1):1–10.
23. Burrows M, Wheeler D. A block-sorting lossless data compression algorithm. Technical Report 124. Digital SRC Research Report. 1994.
24. Gagie T, Navarro G, Prezza N. Fully functional suffix trees and optimal text searching in BWT-runs bounded space. J ACM (JACM). 2020;67(1):1–54.
25. Bannai H, Gagie T, Tomohiro I. Refining the r-index. Theor Comput Sci. 2020;812:96–108.
26. Baker DN. Bonsai: Flexible Taxonomic Analysis and Extension. GitHub; 2022. https://github.com/dnbaker/bonsai. Accessed Sept 2021.
27. Nurk S, Koren S, Rhie A, Rautiainen M, Bzikadze AV, Mikheenko A, et al. The complete sequence of a human genome. T2T-CHM13 v1.0 assembly. AWS. https://s3-us-west-2.amazonaws.com/human-pangenomics/T2T/CHM13/assemblies/chm13.draft_v1.0.fasta.gz. Accessed May 2022.

28. Moss EL, Maghini DG, Bhatt AS. Complete, closed bacterial genomes from microbiomes using nanopore sequencing. ONT microbiome reads (SRX6602475). SRA. https://www.ncbi.nlm.nih.gov/sra/SRX6602475[accn]. Accessed May 2022.
29. Church DM, Schneider VA, Steinberg KM, Schatz MC, Quinlan AR, Chin CS, et al. Extending reference assembly models. GRCh38 assembly. RefSeq. https://doi.org/10.1186/s13059-015-0587-3. Accessed May 2022.
30. Levy S, Sutton G, Ng PC, Feuk L, Halpern AL, Walenz BP, et al. The diploid genome sequence of an individual human. Human Assembly. Refseq. https://doi.org/10.1371/journal.pbio.0050254. Accessed May 2022.
31. Steinberg KM, Schneider VA, Graves-Lindsay TA, Fulton RS, Agarwala R, Huddleston J, et al. Single haplotype assembly of the human genome from a hydatidiform mole. Human Assembly. RefSeq. https://doi.org/10.1101/gr.180893.114. Accessed May 2022.
32. Pendleton M, Sebra R, Pang AWC, Ummat A, Franzen O, Rausch T, et al. Assembly and diploid architecture of an individual human genome via single-molecule technologies. Human assembly. RefSeq. https://doi.org/10.1038/nmeth.3454. Accessed May 2022.
33. Chaisson MJ, Huddleston J, Dennis MY, Sudmant PH, Malig M, Hormozdiari F, et al. Resolving the complexity of the human genome using single-molecule sequencing. Human assembly. RefSeq. https://doi.org/10.1038/nature13907. Accessed May 2022.
34. Zook JM, Catoe D, McDaniel J, Vang L, Spies N, Sidow A, et al. Extensive sequencing of seven human genomes to characterize benchmark reference materials. Human assembly. RefSeq. https://doi.org/10.1038/sdata.2016.25. Accessed May 2022.
35. Huddleston J, Chaisson MJ, Steinberg KM, Warren W, Hoekzema K, Gordon D, et al. Discovery and genotyping of structural variation from long-read haploid genome sequence data. Human assembly. RefSeq. https://doi.org/10.1101/gr.214007.116. Accessed May 2022.
36. Seo J-S, Rhie A, Kim J, Lee S, Sohn M-H, Kim C-U, et al. De novo assembly and phasing of a Korean human genome. Human assembly. RefSeq. https://doi.org/10.1038/nature20098. Accessed May 2022.
37. Ahmed OY, Rossi M, Gagie T, Boucher C, Langmead B. SPUMONI 2: Improved classification using a pangenome index of minimizer digests. 2023. https://doi.org/10.5281/zenodo.7867076.
38. Ahmed OY, Rossi M, Gagie T, Boucher C, Langmead B. SPUMONI 2: Improved classification using a pangenome index of minimizer digests. 500 E. coli genomes used in experiments. 2023. https://genome-idx.s3.amazonaws.com/spu2/ecoli_500_dataset.tar.gz. Accessed Nov 2021.
39. Ahmed OY, Rossi M, Gagie T, Boucher C, Langmead B. SPUMONI 2: Improved classification using a pangenome index of minimizer digests. Mock community pangeome index. 2023. https://genome-idx.s3.amazonaws.com/spu2/mock_community_ont_index.tar.gz. Accessed Nov 2021.
40. Ahmed OY, Rossi M, Gagie T, Boucher C, Langmead B. SPUMONI 2: Improved classification using a pangenome index of minimizer digests. Human pangenome index. 2023. https://genome-idx.s3.amazonaws.com/spu2/human_pangenome_ont_index.tar.gz. Accessed Nov 2021.
41. Ahmed OY, Rossi M, Gagie T, Boucher C, Langmead B. SPUMONI 2: Improved classification using a pangenome index of minimizer digests. Assembly contamination indexes. 2023. https://genome-idx.s3.amazonaws.com/spu2/assembly_contamination_index.tar.gz. Accessed Nov 2021.
42. Ahmed OY, Rossi M, Gagie T, Boucher C, Langmead B. SPUMONI 2: Improved classification using a pangenome index of minimizer digests. Index files containing sampled document array. 2023. https://genome-idx.s3.amazonaws.com/spu2/sampled_doc_array_index.tar.gz. Accessed Nov 2021.

Publisher's Note