Seamless, rapid, and accurate analyses of outbreak genomic data using split *k*-mer analysis

Romain Derelle,[1,7] Johanna von Wachsmann,[2,7] Tommi Mäklin,[2,3] Joel Hellewell,[2] Timothy Russell,[4] Ajit Lalvani,[1] Leonid Chindelevitch,[5] Nicholas J. Croucher,[5] Simon R. Harris,[6,7] and John A. Lees[2,7]

[1]NIHR Health Protection Research Unit in Respiratory Infections, National Heart and Lung Institute, Imperial College London, London W21PG, United Kingdom; [2]European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Genome Campus, Hinxton CB10 1SD, United Kingdom; [3]Department of Mathematics and Statistics, University of Helsinki, Helsinki 00014, Finland; [4]Centre for Mathematical Modelling of Infectious Diseases, London School of Hygiene & Tropical Medicine, London WC1E 7HT, United Kingdom; [5]MRC Centre for Global Infectious Disease Analysis, Department of Infectious Disease Epidemiology, School of Public Health, Imperial College London, London W12 0BZ, United Kingdom; [6]Bill and Melinda Gates Foundation, Westminster, London SW1E 6AJ, United Kingdom

Sequence variation observed in populations of pathogens can be used for important public health and evolutionary genomic analyses, especially outbreak analysis and transmission reconstruction. Identifying this variation is typically achieved by aligning sequence reads to a reference genome, but this approach is susceptible to reference biases and requires careful filtering of called genotypes. There is a need for tools that can process this growing volume of bacterial genome data, providing rapid results, but that remain simple so they can be used without highly trained bioinformaticians, expensive data analysis, and long-term storage and processing of large files. Here we describe split *k*-mer analysis (SKA2), a method that supports both reference-free and reference-based mapping to quickly and accurately genotype populations of bacteria using sequencing reads or genome assemblies. SKA2 is highly accurate for closely related samples, and in outbreak simulations, we show superior variant recall compared with reference-based methods, with no false positives. SKA2 can also accurately map variants to a reference and be used with recombination detection methods to rapidly reconstruct vertical evolutionary history. SKA2 is many times faster than comparable methods and can be used to add new genomes to an existing call set, allowing sequential use without the need to reanalyze entire collections. With an inherent absence of reference bias, high accuracy, and a robust implementation, SKA2 has the potential to become the tool of choice for genotyping bacteria. SKA2 is implemented in Rust and is freely available as open-source software.

[Supplemental material is available for this article.]

Pathogen genomes accumulate sequence variation over time, and mapping these differences to their genomes has proven invaluable for tracking outbreaks and informing public health interventions (Harris et al. 2010; Gardy et al. 2011; Grad et al. 2012; Quick et al. 2016). Genetic distances are a powerful additional source of information for inferring transmission events in outbreaks, allowing transmission events to be comprehensively ruled out when epidemiological links may suggest otherwise (Cori et al. 2018; Wymant et al. 2018). In particular, single-nucleotide polymorphisms (SNPs) make up a substantial portion of the molecular events that generate genetic diversity among pathogenic strains. SNPs are ideal for inferring evolutionary relationships owing to their high resolution, as well as the availability of tractable molecular models for phylogenetics.

From a practical perspective, the identification of SNPs still relies on complex and computationally intensive pipelines usually based on read alignment. Making reliable SNP calling quicker and easier would lower the barriers to routine implementation for transmission tracing and would assist academic scientists wishing to quickly investigate populations of pathogens before launching more involved analyses. As volumes of pathogen genome data continue to grow (Hunt et al. 2024), tools that can make use of large populations without requiring excessive disk space or compute time can make the iterative cycle of investigation much more interactive and accessible. This is particularly the case for low-resource settings, which often have a high burden of bacterial diseases. Genome data are large and complex, even more so when dealing with populations. Unlike tabular data, which are easy to manipulate and plot, genome data have remained relatively undemocratic; their use is still mostly restricted to experts with good computational resources and training.

Identifying SNPs within a population traditionally involves aligning each sample's sequencing reads against a reference genome. This involves multiple algorithms and pieces of software, so the resulting pipelines are often complex and sensitive to parameters and may not necessarily be well tuned to every pathogen, often restricting their usage to users with strong bioinformatics training and resources. Additionally, the choice of reference genome, read-alignment pipeline, and its parameters can have a

[7]These authors contributed equally to this work.
Corresponding author: jlees@ebi.ac.uk

Article published online before print. Article, supplemental material, and publication date are at https://www.genome.org/cgi/doi/10.1101/gr.279449.124. Freely available online through the *Genome Research* Open Access option.

significant impact on SNP identification and can influence downstream analyses (Pightling et al. 2014; Usongo et al. 2018; Bush et al. 2020; Walter et al. 2020). Differences in genomic composition and organization between the input sequences and the chosen reference can generate misalignments, leading to the identification of spurious SNPs (Landan and Graur 2009; Farrer et al. 2013; Hurgobin and Edwards 2017). Without sufficient coverage, this approach is more likely to call the reference variant, a phenomenon known as "soft reference bias" (Paten et al. 2017). Alignment pipelines mitigate these errors using various alignment and SNP filtering criteria (e.g., matching alignment strands), excluding certain genomic regions or selecting a reference genome closely related to the strains of interest (Colquhoun et al. 2021). With typical mapping approaches, SNPs are totally missed when they occur in genomic regions that are either absent from or too dissimilar to the reference genome, a phenomenon known as "hard reference bias."

An alternative approach, and the subject of our work, is to use subsequence probes, namely, *k*-mers to probe variation between samples, which avoids mapping and variant-calling steps entirely. Odd-length *k*-mers can be further divided into two smaller fragments surrounding a variable middle base, forming a structure called "split *k*-mer" (Gardner and Hall 2013; Harris 2018). In this structure, the left and right parts of the *k*-mer serve as local reference points to map the position of the middle base. If the same combination of left–right *k*-mers is found in another strain with a different middle base, homology between the two middle bases can be hypothesized and their difference interpreted as a SNP. The data conversion into split *k*-mers essentially creates independent local references for each genomic position of the strains of interest on the fly, enabling alignment-free and reference-free comparisons between them. Additionally, the split *k*-mers can be mapped to a reference sequence to impose a useful and interpretable ordering and coordinate system on the called variants.

Alignment-free algorithms can entirely avoid both forms of reference bias, so they are well suited to bacterial species in which these biases have a major effect on accuracy. The split *k*-mer approach does not use alignment during construction, so it is robust to diversity from a reference, either avoiding references entirely as in *ska align* or using the reference only as a coordinate system as in *ska map*. It is, however, important to note that cases of high diversity *among* samples being aligned cannot be addressed by SKA: SNPs spaced closer than the *k*-mer length will disrupt exact *k*-mer matches, which becomes more likely the higher the divergence between a pair of samples.

The split *k*-mer approach was first introduced through the kSNP program (Gardner and Hall 2013; Gardner et al. 2015; Hall and Nisbet 2023). We previously implemented an enhanced version of this approach in the SKA ("split *k*-mer analysis") software, hereafter referred to as SKA1. SKA1 offered improvements in efficiency and flexibility and the ability to map detected SNPs to a reference genome (Harris 2018). SKA1 has since been successfully employed in genomic studies of a large range of bacterial pathogens (Becker et al. 2021; Gladstone et al. 2021; Morris et al. 2022; Chew et al. 2023) and has independently been shown to be superior to limited-resolution gene-by-gene approaches for mapping transmission (Maechler et al. 2023). There remains a need for these variant-calling methods that scale to the size of modern data sets and produce high-quality results without needing manual testing and adjustment and that avoid the soft and hard reference bias common in analysis of bacterial populations (Bush et al. 2020; Colquhoun et al. 2021; Valiente-Mullor et al. 2021).

We have therefore re-engineered and improved our implementation to develop SKA2, which we designed to be faster, more flexible, and easier to maintain. SKA2 is intentionally designed as a tool that is quick and easy to use, although it still requires some command line expertise. Academics can use SKA2 to quickly analyze data and test hypotheses. In public health and clinical settings, SKA2 can rapidly determine whether an isolate is from an outbreak quickly and without the need for dedicated high-performance computing. At the same time, users have access to a method that is as accurate as the more complex, detailed analyses for the particular question they have.

Methods

Split *k*-mer encoding

A split *k*-mer is broadly defined as a sequence of length k, containing one or more contiguous unspecified ("wildcard") bases that can match with any character (Harris 2018). Throughout this paper, we use a more specific definition, which is an odd length *k*-mer in which the middle base is allowed to vary (i.e., the only wildcard position is the middle one). Other choices, including having the wildcard position as the final or first base, would be possible, but including the base within the *k*-mer ensures only SNPs can be tagged, as a variable base at the end may be the start of a larger genetic event. We therefore opt for the middle base following the precedent of SKA1. This spaced seed pattern has also been proven to be optimal for lossless filtering of single-base errors (Kucherov et al. 2005; Břinda 2013). For example, with $k = 11$, the split *k*–mer would be XXXXX-XXXXX, where X = {A, C, G, T} and "-" is any base. The methods used in SKA2 are demonstrated with an example with $k = 11$ in Figure 1.

To count the split *k*-mers in an input FASTA file, we read each new base in turn and use the following commonly used mapping to two bits: {A, C, G, T} = {00, 01, 11, 10} (Drezen et al. 2014). This can be efficiently converted into the upper and lower half of the split *k*-mer using bitshift and masking operations, rolling through the input string rather than rebuilding the split *k*-mer at every position. To match the machine word size, for $k \leq 31$, the known bases are packed into 64 bits; for $k \leq 63$, the known bases are packed into 128 bits. The default is to consider both of the integer representations of the *k*-mer and its reverse complement and keep only the smallest, which is consistent irrespective of which strand the *k*-mer was actually counted on. If the strand is made to be consistent between samples ahead of time (e.g., single-stranded viruses or reference sequences), this step can be skipped.

We store the middle base in a single byte and thereby support IUPAC uncertainty codes. Any observation of the same split *k*-mer contributes equally to the uncertainty in the middle base (i.e., a single duplicate observation adds that base as a possibility), and the specific counts of different middle bases are ignored. Because of the split *k*-mer necessarily containing an even number of fixed bases, it is possible for the fixed part of a split *k*-mer to be its own reverse complement (a palindrome). In this case, the strand of the middle base is ambiguous, so we keep both the observed base and its complement as observations using uncertainty codes.

The fundamental data structure we use in SKA2 for each input sequence is a hash map (dictionary), with the split *k*-mers as the keys and with their middle bases as the values for the entry. For example, using $k = 11$ and ignoring the reverse-complement sequences for simplicity, the sequence CTAGCTCACAAGT would have the dictionary entries {CTAGCCACAA:T; TAGCTACAAG:C, AGCTCCAAGT:A}.

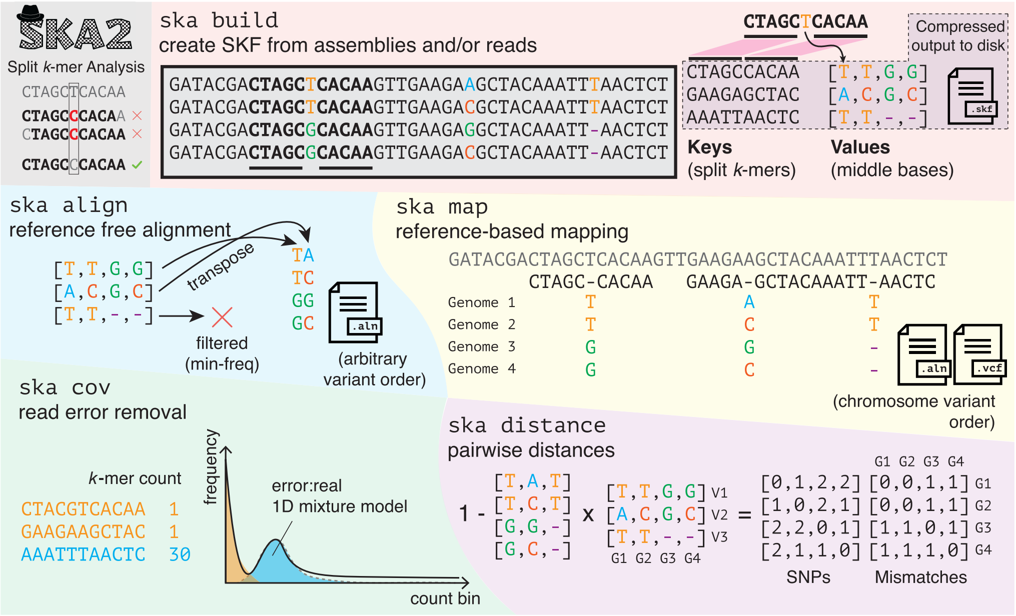

Figure 1. Overview of functions and methods in SKA2. Split *k*-mers allow matching variant positions, whereas contiguous *k*-mers mismatch any variation. *ska build* creates split *k*-mer dictionaries from input sequence data. The example shows four sequences that are aligned and on the same strand for clarity, but in real input data, neither is necessary. Split *k*-mers are used as keys, and their middle bases are stored in lists. This dictionary is compressed using snappy to make split *k*-mer files (SKFs). *ska align* makes reference-free alignments with no coordinate system by writing out the middle bases, applying filters on the frequency of missing data, constant sites, and ambiguous sites. *ska map* makes reference-based mappings as ALN or VCF, with the same coordinate system as the reference. In both modes, the conserved sites are also written out but are not shown for clearer visualization. *ska cov* counts *k*-mers and fits a mixture model to find a threshold for count when using reads as input to ska build. *ska distance* calculates SNP distances and mismatches between samples by multiplying the middle base matrix by its transpose. The cluster_dists.py script can be run on this distance matrix to make phylogeny, single-linkage clusters with a provided threshold, and a Microreact visualization. Operations to merge, delete samples and split *k*-mers, and write out the contents of SKFs are also implemented but are not shown.

Creating split *k*-mers from short-read sequencing data

When using FASTQ files as input, ska assumes these have been generated from error-prone short-read sequencing. We implement two filtering options to prevent sequencing errors from entering the split *k*-mer dictionary. The first is a minimum quality score for reported read quality (default = 20), which can be applied at the middle base only ("middle") or across the entire split *k*-mer ("strict", the default).

The second is a minimum observation count (default = 5). Keeping track of *k*-mer counts, even in relatively small sequencing experiments, takes >90% of the runtime, so filtering on the minimum observation count is an important optimization target. For simulated 150 bp paired-end reads from *Mycobacterium tuberculosis* at 60× coverage, a baseline of keeping counts in a dictionary takes 99 sec and uses 2.4 GB of memory. We evaluated numerous alternative approaches and found that a two-stage approach, using a blocked-Bloom filter to remove singletons and then a dictionary to count *k*-mers appearing at least twice was the fastest, the least memory-consuming, and the most accurate insofar as it gave no false positives or false negatives with the default count threshold. This is similar to the approach taken by bifrost (Holley and Melsted 2020). The hashes in the Bloom filter are calculated using ntHash (Mohamadi et al. 2016); the filter uses a word-aligned block with 12 bits per block (~1% FPR) and is 2^{27} blocks wide (~200 million bits) (Qiao et al. 2014). For the same simulated sample, this approach takes 46 sec and uses 1.6 GB of memory.

To select a suitable minimum *k*-mer count we implemented a coverage model (*ska cov*) that fits a mixture model to the count frequency distribution of contiguous *k*-mers. There are two advantages of a model-based approach over a simpler method such as choosing the minimum count bin: The model is guaranteed to have a single global minimum and can be fit to find a single threshold even with noisy data.

We used an observation model that is a mixture of error *k*-mers and real *k*-mers. The error component probability density is a Poisson count with a mean of one; the real *k*-mer probability density is a Poisson count with a mean of the coverage *c*. Their mixture with *w* as the proportion of error *k*-mers has likelihood

$$\mathcal{L}(w, c) = \prod_{i=1}^{i_{\max}} x_i \cdot \left(w \cdot \frac{e^{-1}}{i!} + (1 - w) \cdot \frac{c^i e^{-c}}{i!} \right),$$

where the datum x_i is the number of split *k*-mers counted *i* times. We fit *w* and *c* by maximizing the likelihood. We do this using the BFGS algorithm (Fletcher 1970) and deriving analytic expressions for the gradient $\frac{\partial \mathcal{L}}{\partial w}$, $\frac{\partial \mathcal{L}}{\partial c}$. The count cutoff is selected by picking the first integer value *i* for which the responsibility of the non–error component is greater than the error component. Users can select either this threshold (which minimizes the chance of including error *k*-mers) or the minimum count bin from the output (which increases sensitivity at low coverages). An example of observed coverage counts and the *k*-mer model fit is shown in Supplemental Figure 1.

Split *k*-mer genotyping

To create an alignment, the multiple dictionaries from each sample are merged into a single dictionary. The values in this dictionary are then lists of the middle bases for each sample, using a gap character "-" if the split *k*-mer was not observed in that sample.

We implement two operations on this merged dictionary: append (to add a single sample) and merge (to combine two merged dictionaries). Using these operations with the same algorithm as a parallel merge sort, it is possible to support parallel construction and merging. As the merge operation is slower than append, this only gives a speedup when construction of the initial dictionaries is slow, namely, with read data. With two threads on a data set of 28 samples with 3.2 million split *k*-mers, our approach yields a speedup of 1.7×. The *ska build* command combines this merge with the encoding for multiple samples as described above. By doing this, we minimize the memory and disk requirements and simplify the command line use. It is also possible to parallelize the build over samples using a program such as GNU parallel, followed by a single merge. This process gives close to a 100% efficient speedup, at the expense of increased memory usage and more temporary disk space.

In the default, reference-free mode, *ska build*, each aligned column is a list of the middle bases in this merged array. Therefore, creating the output alignment simply consists of writing the transpose of the dictionary values. Options to filter the sites are available: a minimum number of observations (i.e., nonmissing middle bases); remove any ambiguous sites; replace any ambiguity with "N"; no constant sites; and allow "-" in otherwise constant sites (useful for low coverage samples). These columns are in an arbitrary order, as in the hash table, and do not represent a physical position in the chromosome.

In the reference-based mode *ska map*, the algorithm iterates over the split-*k*-mers in the input reference sequence, searching for each one in the merged dictionary. If a match is found, the middle base at that position is written; otherwise, a missing base "-" is output. Extra logic is needed to write the matching parts of the split *k*-mers, keep track of multiple chromosomes, and mask mapped but repeated split *k*-mers with N (if requested by the user). We have implemented rigorous unit testing to ensure this process works correctly. Output as either FASTA or VCF is supported.

Split *k*-mer files and their supported operations

We save split *k*-mer files (SKFs) to disk for reuse, addition, or alternative filtering. These consist of the list of split *k*-mer keys, an array of the middle bases, the count of samples in which each split-*k*-mer has occurred, the sample names, and some metadata (*k*-mer size, version, and number of bits used for *k*-mer keys). To generate these, we serialize the split *k*-mer dictionaries and compress them using snappy, which adds minimal time to read/write operations and achieves a compression ratio of ∼10×. Using SKFs, we support merging with another SKF (*ska merge*), deleting named samples (*ska delete*), and deleting specified split *k*-mers or refiltering the split *k*-mers (*ska weed*). The number of *k*-mers command (*ska nk*) can be used to output the contents of SKFs to the terminal, including the total number of split *k*-mers and the split *k*-mer observed in each sample, which can be a useful quality-control metric (e.g., elevated in the case of contamination). The same command can optionally output, in text form, the split *k*-mer sequences and the middle bases of all samples, which can be used for processing in other bioinformatic tools.

Sample distances and clustering

We implemented the *ska distance* command, which iterates over every pair of samples to calculate the number of SNPs different between the samples. For each matching split *k*-mer between a pair of samples, if the middle base matches, then zero is added to this distance; otherwise, one is added to the distance. For ambiguous middle bases, a probability vector is set using the IUPAC codes, with the probability equally divided among possible middle bases. For example, if a sample has S as a middle base (either a C or a G), we give a 50% weight to each one. The probability of a match between the samples at that split *k*-mer is then given by multiplying the probability vectors of each sample. For example, if two samples had the middle bases S and Y, this would give corresponding vectors for A, C, G, T, (0, 0.5, 0.5, 0) and (0, 0.5, 0, 0.5), yielding a match probability of 0.25. For assemblies and small numbers of repeats, this is reasonable but will be less effective than a full variant caller that tracks the exact count and quality scores of each observation, rather than dividing the probability equally between the possible bases. The number of missing sites is also reported as mismatches. To fully replicate and extend the functionality of SKA1, we also wrote a downstream Python script that clusters samples using single linkage at a given SNP threshold, makes a neighbor-joining tree with RapidNJ (Simonsen et al. 2008), and creates a visualization of the clusters, tree, and single-linkage graph. This is displayed interactively using Microreact (Argimón et al. 2016); an example is shown in Supplemental Figure 2.

Comparison to SKA1

We designed SKA2 so that the majority of SKA1 functions would also be available, but a full list of differences is given in the software documentation. The code runs automated tests and ensures at least 90% test coverage of the code. Each version is automatically compiled and so can be installed as a static binary, using the Rust toolchain to compile locally, or via Bioconda. All tests in this paper used version 0.3.5.

Some features of SKA1 were not implemented in SKA2. The maximum *k*-mer size is 63 (to fit into a 128-bit integer, the maximum native size supported by Rust) rather than unlimited; our testing showed that longer *k*-mer sizes did not yield improvements (Fig. 2). The *annotate* function was not implemented, as its functionality is covered by BEDTools (Quinlan and Hall 2010). The *unique* function can be parsed from the output of the dictionary and was not made into a separate function. The *type* function has been superseded by *k*-mer tools such as StringMLST (Gupta et al. 2017) and BioHansel (Labbé et al. 2021).

SKA1 has the option, through *ska merge*, to invert the keys and values in the merged dictionary. In this format, each unique pattern of middle bases appears only once in the keys; the corresponding split *k*-mers, in a vector of the values. For closely related samples, this is effective at reducing redundancy; the above example has only 1163 unique patterns from 3264141 split *k*-mers, so the output compresses to 34 MB. We did not implement this in SKA2 as the gain over the current compression approach is marginal and increases processing time.

Split *k*-mer simulations

We simulated a mutated sequence along a single branch using the Gillespie algorithm (Gillespie 1977), following a similar approach to that taken in phyloSim (Sipos et al. 2011). We used a GTR + Gamma + Invar rate model plus short indels, with parameters previously estimated for *Streptococcus pneumoniae* (Lees et al. 2018), using reference ATCC 7000669 (Spn23F) as the starting sequence (Croucher Vernikos et al. 2011b). We tracked the exact expected split *k*-mers for each substitution in the simulation and calculated power as the proportion of these detected by running *ska align* between the starting sequence and the mutated sequence. We calculated power requiring an exact match of the split *k*-mer and the correct middle base, as well as also a more relaxed mode that just checks the flanking sequence, tolerating ambiguity at repeat

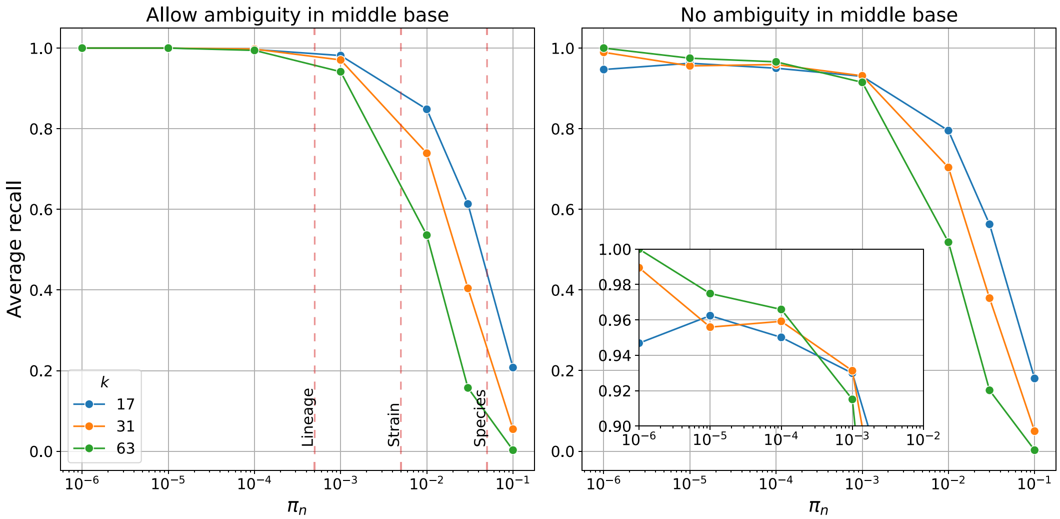

Figure 2. Average recall of SKA2 in simulations across increasing sequence divergence between a pair of sequences (π_n or SNPs per site). Lines show recall using different split *k*-mer lengths *k*. (*Left*) Recall when allowing ambiguous bases, showing typical divergence thresholds used to define species, strain, and lineage boundaries. (*Right*) Recall when requiring exact matches of the middle base, with *inset* showing recall over the within-lineage range.

sequences. Indels are ignored in the power calculation but may reduce power if they interrupt a split *k*-mer.

Outbreak simulations

We simulated outbreak clusters of *S. pneumoniae* and *M. tuberculosis* using Transphylo v1.4.10 (Didelot et al. 2017) with simulation parameters provided in the Supplemental Methods. Starting from the root genomes listed in Supplemental Table 1, we then simulated sets of mutations along these phylogenies using phastSim (De Maio et al. 2022). We choose a mutation rate of 10 and one mutation per genome per year for *S. pneumoniae* and *M. tuberculosis*, respectively, which correspond to upper estimates of the mutational rate of these two species (Chewapreecha et al. 2014; Menardo et al. 2019). For *S. pneumoniae*, we also simulated indels of 100 bp at a rate of 0.5 indel per year to mimic accessory genome changes as in (Lees et al. 2019). phastSim command lines are provided in the Supplemental Methods. SNPs and indels were then randomly inserted in genome assemblies using a custom Python script to recreate the genome of each sample composing the outbreaks. We finally simulated Illumina HiSeq 2500 paired-end sequencing reads of 150 bp from the modified genome assemblies at a 60× coverage using ART (Huang et al. 2012).

We used the variant-calling pipelines Snippy v4.6.0 (https://github.com/tseemann/snippy) and BactSNP v1.1.0 (Yoshimura et al. 2019), both with default parameters. The command lines of the "BWA + BCFtools" read-alignment pipeline are provided in the Supplemental Methods. Briefly, reads were aligned using BWA-MEM v0.7.17 (Li 2013), SNPs were called using BCFtools v1.10.2 (Li 2011) and then filtered using the following criteria: a minimum of 10× coverage, supported by at least one read forward and one read reverse, and a minimum allele frequency of 0.75. SNP alignments obtained from all read-alignment pipelines were finally trimmed to only retain positions with a maximum of 10% missing data, considering degenerate nucleotides as missing. Runtime and memory consumption were measured on an Intel Xeon platinum 8360Y CPU with 10 GB RAM. The parameters and commands used in the simulations are listed in the Supplemental Methods.

False-positive and false-negative SNPs were identified by comparing the positions of the inferred SNP alignments to those of the expected SNP alignment. Phylogenetic analyses were performed using IQtree2 v2.2.2.7 (Minh et al. 2020) under the GTR + ASC model and near-zero branches collapsed into polytomies ("--polytomy" option). Distances between phylogenetic trees were computed using the R package TreeDist v2.7.0 (Smith 2022).

Recombination analysis

We used *ska build*, with both $k = 17$ and $k = 63$, followed by *ska map* with options to mask repeats and any ambiguous bases to create an alignment of 240 samples against the Spn23F reference (Croucher et al. 2009). We used Gubbins v3.3.1 (Croucher et al. 2015), turning off gap filtering (--filter-percentage 100.0), with otherwise default parameters (no outgroup, RAxML as tree builder, five iterations). We compared the density of the recombinations across the genome to the originally published analysis using Phandango (Hadfield et al. 2018), which includes these data as a default example set.

Genome data

Twenty-eight *Listeria monocytogenes* samples from a study of bacterial meningitis, taken from the largest PopPUNK cluster, were used for testing and comparison with SKA1 (Kremer et al. 2017; Lees et al. 2018, 2019). The 288 *E. coli* genome assemblies used for the online analyses were extracted from the GenomeTrackr database (Allard et al. 2016), and all correspond to a single strain in the PopPUNK database (Lees et al. 2019). Their accession numbers are provided in Supplemental Table 2. For the Gubbins analyses, 240 *S. pneumoniae* genomes from the PMEN1 strain were used (Croucher et al. 2011a).

Results

SKA2 can be used to genotype SNPs in sets of populations using *ska align*. These variants are unordered so are only suitable for constructing phylogenies or transmission detection. The *ska distance* tool can also be used to create possible transmission clusters by applying a SNP cutoff. In outbreak settings, this mode is highly accurate and can be used rapidly with either or both reads (60× speedup) and assembly data (190× speedup) as input. Despite the simple approach to SNP genotyping, in its primary use case of outbreak settings, it outperforms traditional read mapping approaches that dominate this type of analysis. In the first section, we show that this holds within related bacterial samples (corresponding to a typical strain or sequence-type cluster definition), and then in the second section, we use simulated outbreak data to compare read mapping approaches. This approach also avoids the need for selection of an appropriate reference, which requires expertise, and introduces hard- and soft-reference bias, which lowers the sensitivity of variant calling.

The mapping approach of *ska map* gives mapped variants a chromosome coordinate. Although in small sample sets of possible transmission this has less power than *ska align*, mapping has advantages in which the order and reference-based functional interpretation of variants needs to be kept, which we demonstrate in an example for recombination detection in the third section below. For large sample sets in which diversity may be larger, *ska map* remains scalable and robust.

Tools to investigate shared *k*-mer content *ska nk* on a per-sample basis are useful quality control and can rapidly determine the diversity of a population. Using *ska weed* to remove or keep chosen sequences can also rapidly be used to detect the presence of genetic elements such as transposons. In the final section, we show these methods can be used to create scalable SKFs for mapping with large and continuously growing sample collections.

Compared with SKA1, SKA2 showed improved runtimes and smaller file sizes, a major objective of our reimplementation. For a 3 Mb bacterial assembly (*L. monocytogenes*), the build process of SKA2 takes 0.1 sec on average, producing a SKF 16 MB in size. In SKA1, the process takes 2.8 sec and produces a SKF 30 MB in size. For 28 samples of *L. monocytogenes* from the same strain, the SKF is 38 MB and takes 5 sec to create. In SKA1, these files are 870 MB and take 194 sec to create and merge (~40× speedup). For a set of simulated reads, SKA2 build takes 46 sec and uses 1.4 GB of memory. In SKA1, this takes 120 sec (~2.5× speedup) and 0.6 GB memory. A major advantage of SKA2 over SKA1 is therefore the optimized and streamlined combined build and merge step, as well as the resulting smaller files. This increase in speed is owing to a faster dictionary implementation, optimized file parsing, and use of a faster serialization process. Parallelization is not directly supported in SKA1.

Split *k*-mers accurately and quickly genotype closely related samples

Before using SKA2 to analyze bacterial populations, we first used two-sample simulations to evaluate the power and limitations of the split *k*-mer genotyping approach. We first note that more complex variation, such as insertions, deletions (indels), rearrangements, or copy number variation, is ignored by SKA2 entirely. However, as typically only SNP calls are used in reconstructing evolutionary history (Lees et al. 2018), owing to their mostly vertical inheritance, and the existence of corresponding effective models of molecular evolution, we focus on these variants entirely.

As SKA2 is alignment-free, it is unaffected by soft reference bias (more likely to call the reference base) (Colquhoun et al. 2021). *ska align* is also robust to changes in sequence content, so it is unaffected by hard reference bias (missing accessory genes). However, split *k*-mers can still miss SNP variant calls, causing false negatives, for the following reasons:

- When using assemblies as input, variants closer than half the split *k*-mer length to a chromosome or contig boundary cause the flanking bases to be too short to be enumerated in the dictionary. When using reads as input, a similar problem occurs at the ends in the case of very low coverage data.
- Two or more SNPs within the half *k*-mer arm length (i.e., $(k-1)/2$) of one another cause a mismatch in the flanking bases.
- Similarly, indels or rearrangements can interrupt split *k*-mers.
- Repeated split *k*-mers cannot be uniquely mapped, so their middle bases contain ambiguity from all observations of the split *k*-mer.

We also note that doubly (or more) mutated sites, occurring more frequently at larger distances, only count the most recent variant. In simulations, it is possible to detect this as an error from the reference sequence, but it is not an issue with variant-calling methods themselves.

Unremoved sequencing errors in reads (owing to poor filtering) or assembly errors may introduce false-positive variant calls prior to input to the SKA2 data structure, but split *k*-mers themselves introduce effectively zero further false-positive variant calls. We therefore focused on false negatives here. False positives are assessed further in the outbreak simulations below.

We used simulations to evaluate the relative importance of these effects across evolutionary distances (Fig. 2, left panel). We found that the major effect on calling accuracy was variants occurring closer than the *k*-mer distance. Indel rate had a negligible effect on power, as it was no more likely to interrupt a split *k*-mer than a nearby SNP (Supplemental Fig. 3). SKA is intended to be used for closely related samples and shows good power in this range. Within a lineage (divergence $\lesssim$ 0.0005 SNPs per site, in some species referred to as a clone), the recall is >99%. Outbreaks are often much more closely related than this, as many species acquire only a few SNPs per year (Duchêne et al. 2016), showing that SKA2 is a reliable genotyping tool for this purpose.

Within a bacterial strain (divergence $\lesssim$ 0.005 SNPs per site), this drops slightly to ~90%, and across an entire bacterial species (divergence $\gtrsim$ 0.05 SNPs per site), the recall drops further, to between 10% and 50%. When ignoring repeats, defined here as any split *k*-mer with multiple observations, shorter *k*-mers had consistently higher power as they are proportionally less susceptible to multiple close SNPs. We therefore recommend that SKA2 is used only within bacterial strains and, for this purpose, with a short *k*-mer length.

The main effect on accuracy is the resolution of repeats (Fig. 2, right panel), although we note that this is a strict criterion, as most phylogenetic software allows uncertainty in alignments. Requiring exactly resolved repeats results in an average recall to ~95%, although using longer *k*-mers improves this up to ~98%. Split *k*-mers show the archetypal sensitivity/specificity tradeoff for length: At small divergences, longer split *k*-mers are preferred as they are more specific, owing to uniquely mapping more sequences by spanning shorter repeats; at larger divergences, smaller split *k*-mers are preferred as they are more sensitive, owing to mapping more nearby SNPs.

SKA2 outperforms read-mapping approaches in simulated outbreak analyses

Having confirmed that split *k*-mers are most effective for closely related examples, we next evaluated the performance of SKA2 in simulated outbreaks, comparing SKA2 to traditional read-alignment methods. We analyzed simulated outbreaks of *S. pneumoniae* (12 samples, 87 SNPs) and *M. tuberculosis* (30 samples, 38 SNPs). Each simulated outbreak was replicated five times, leading to 20 simulated outbreaks per species. We generated sequences at the tips of each expected phylogeny from the transmission series and then created simulated short-read sequences from this truth set of full-length sequences. We tested the effect of reference bias by using references of increasing divergence from the root genome (ATCC_700669 and H37Rv for *S. pneumoniae* and *M. tuberculosis*, respectively). We compared SKA2 (at k = 31) to the popular read-alignment tool Snippy, the hybrid assembly mapping pipeline BactSNP, and a custom-made SNP-calling pipeline referred hereafter as BWA + BCFtools. We used SKA2 in both reference-free mode (align) and its reference-based mode (map).

For *S. pneumoniae* outbreaks, read-alignment pipelines showed increasing numbers of false-negative SNPs when more distant outbreaks to the reference genome were analyzed, highlighting the impact of differential genome composition between the analyzed outbreaks and the reference genome (Fig. 3). In contrast, *ska align* showed relatively stable numbers of missed SNPs, regardless of the origin of the outbreak: SKA2 missed similar numbers of SNPs to read-alignment pipelines when the analyzed outbreak corresponds to the reference genome (here from the strain ATCC 700669), but it missed significantly fewer when outbreaks isolates were much less similar to the reference genome. None of the SNP inference methods produced false-positive SNP when outbreaks derived from the ATCC 700669 reference genome were analyzed. However, Snippy, BWA + BCFtools, and BactSNP produced on average 89.1, 6.4, and 0.2 false-positive SNPs, respectively, when more distant outbreaks were analyzed. The low specificity of Snippy has already been characterized in benchmark studies (Yoshimura et al. 2019; Falconer et al. 2022). In contrast, SKA2 did not produce any false-positive SNPs in any of these analyses. We then compared the phylogenetic trees obtained from all these SNP alignments to the tree obtained from the set of simulated SNPs. We used the clustering information distance from the TreeDist package (Smith 2020), which is a measure of how similarly two trees group tips. Except for outbreaks derived from the ATCC 700669 genome, the phylogenetic trees obtained from the BactSNP and SKA2 SNPs were found to be more similar to the expected tree than those derived from SNPs inferred by Snippy and the BWA + BCFtools read-alignment pipeline.

For *M. tuberculosis*, which is characterized by a highly conserved genome across lineages (Comas et al. 2013; Gagneux 2018), SKA2 still demonstrated similar or superior performance compared with read-alignment methods, as illustrated in Figure 3. The numbers of false-negative SNPs were found to be similar between all methods, and BactSNP and SKA2 were the only methods to not produce any false-positive SNPs (on average, 7.6 and 0.1 false-positive SNPs generated by Snippy and BWA + BCFtools in non-H27Rv outbreaks). The phylogenetic trees obtained from BWA + BCFtools, BactSNP, and SKA2 SNPs were all found to be similarly distant to the expected tree, whereas those obtained from SNPs inferred by Snippy were found to be more distant.

Using *ska map* gives very low numbers of false positives. We investigated the small number of false positives and found them to be present in tandem repeats. In cases in which the flanks of a split *k*-mer can be mapped in multiple overlapping places, it is ambiguous whether the mapped base should be overwritten by the conserved flank or the middle base (for consistency of reporting, the code always chooses the latter option). Filtering on repeat regions removes these false positives but increases false negatives. For a purely phylogenetic or transmission analysis, the lower recall (which drops further with more distant references) is less desirable. However, when using many samples, the ability to map them one at a time can still be useful; with *ska align*, the entire intersection of split *k*-mers must always be kept, which leads to large memory use with diverse samples to keep low-frequency (and typically unused) split *k*-mers. In such sample sets, using the default of 80% presence can also lead to loss of *k*-mers in *ska align*, which can be usefully retained by *ska map*.

We repeated the SKA2 analyses to assess the impact of split *k*-mer size on its SNP detection. Using *k*-mer sizes ranging from 21 to 61 nucleotides, in increments of 10, we observed similar numbers of identified SNPs between *k* = 21 and *k* = 51 for *S. pneumoniae*, followed by a sharp decrease at k = 61 (Supplemental Fig. 4). For *M. tuberculosis*, we observed a slight increase in the number of identified SNPs from *k* = 21 and *k* = 31 followed by a plateau and again a sharp decrease at *k* = 61 (Supplemental Fig. 4). Then, we evaluated the performance of SKA2 in low sequence coverage settings, from 60× coverage used in the previous set of analyses down to 10× coverage, in decrements of 10. We observed a sharp decrease in the sensitivity of SKA2 at 20× coverage when filtering settings are not adjusted (average sensitivity of 0.93 to 0.43 and of 0.96 to 0.31 for the *S. pneumoniae* and *M. tuberculosis* outbreaks, respectively) (Supplemental Fig. 5). We used *ska cov* to determine a more sensitive minimum count threshold (Supplemental Table 3). This resulted in changing from the default of five to three and restored the sensitivity of SKA2 to

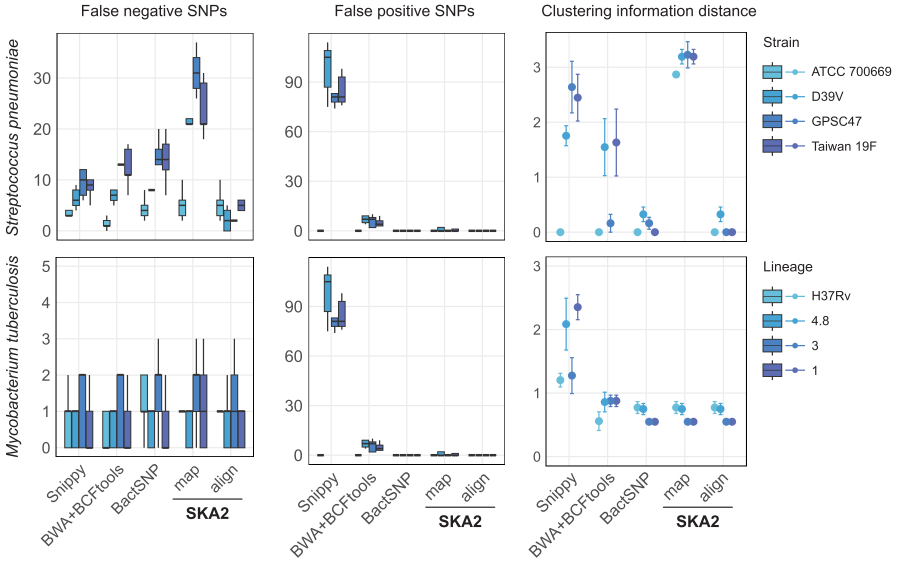

Figure 3. Results obtained from the analyses of simulated outbreaks showing recall (false negatives), false positives, and clustering information distance from the four different tools. "map" and "align" refer to the SKA2 functions used to generate SNP alignments. References of increasing distance (darker blue) from the source of the outbreak were used to evaluate reference bias. The error bars in the CI distance plots correspond to the 95% confidence interval calculated from 10 values (two phylogenies were obtained from each SNP alignment using two independent maximum-likelihood runs). The numbers 4.8, 3, and 1 in the legend correspond to the names of *M. tuberculosis* lineages.

0.89 and 0.93 at 20× coverage for the *S. pneumoniae* and *M. tuberculosis* outbreaks, respectively, which corresponds to sensitivity levels obtained by Snippy at the same coverage. These analyses also revealed a small rate of false-positive SNPs inferred by SKA2 at 30× and lower coverages owing to unfiltered sequencing errors, although, on average, fewer than 0.5 false-positive SNPs per simulated outbreak (Supplemental Fig. 5). These spurious SNPs might be caused by differential distribution among samples of split-*k*-mers corresponding to duplicated genomic regions owing to stochastic variations of coverage.

We also recorded runtimes and maximum memory consumption of SKA2, SKA1, and read-alignment pipelines in these outbreak analyses. SKA2 was found to be by far the fastest method: two- to fourfold faster than SKA1, 14- to 20-fold faster than the read-alignment pipelines Snippy and BWA + BCFtools, and 60-fold faster than the hybrid pipeline BactSNP (Table 1). Despite the necessity to store all split *k*-mers in memory, SKA2 also displayed modest maximum memory consumption values, which were similar to those observed with SKA1 and other variant-calling pipelines. Finally, the SKFs produced by SKA2 were, on average, only 24 and 53 MB in size per outbreak for *S. pneumoniae* and *M. tuberculosis*, respectively (31 and 66 MB for SKA1 output files), whereas the BAM files produced by the read-alignment pipelines were, on average, 91 and 185 MB in size per sample for *S. pneumoniae* and *M. tuberculosis*, respectively. SKA2 disk usage was therefore found to be 40–100 times lower than traditional read-alignment.

With much longer genomes, such as those from eukaryotes, SKA2's resource use has linear scaling with the number of *k*-mers, which is a function of both genome length and diversity. For example, to align two typical human genomes (3 Gb length, $\pi_n = 0.001$ SNP density) requires 34 min CPU, 230 GB memory, and a 9 GB disk for the SKF. The memory use and requirement for assembled genomes as input therefore make SKA2 a less appealing choice for these organisms.

We finally recorded runtimes and maximum memory usages of SKA2 using a real-world data set composed of FASTQ files from 100 isolates of *S. pneumoniae* belonging to the IC1 cluster (Croucher et al. 2014). By repeating the analyses (*ska build* at $k = 31$ and *ska align* functions) in increments of 10 isolates, we found that SKA2 runtimes increased nearly linearly with the number of isolates (Fig. 4). In contrast, and as expected, its maximum memory usage was found to be dependent on the total number of split *k*-mers extracted from the sets of isolates.

SKA2's map can be used to effectively detect recombination with assemblies as input

We show above that for phylogenetics, using *ska align* is superior to the reference-based approach of *ska map*. However, the all-versus-one approach of *ska map* offers several key advantages: a unique comparison point based on which all variants are characterized, some functional interpretation by using the annotation of the reference, a well-defined coordinate system, computational scalability that can be trivially parallelized, and flexibility as new samples can be added to an already processed data set.

For some purposes, ordering and spacing variants along the chromosome are necessary. These are typically for uses in which the actual physical molecule structure is important to maintain local linkage disequilibrium. One example is recombination detection, which typically finds windows of locally elevated variant density, which are more likely the result of transfer from a relative than from local hypermutation (Croucher et al. 2015; Didelot and Wilson 2015; Mostowy et al. 2017). Another is genome-wide association studies, in which local linkage disequilibrium is used to define distinct signals (Lees et al. 2016).

Anecdotally, a popular use of SKA1 has been to create ordered SNP alignments against a reference genome for use in recombination detection. This mode of input generation is also officially supported in the Gubbins package (Croucher et al. 2015). Given that a limitation of split *k*-mers is that SNPs closer than the *k*-mer length will be missed, we tested that this is still sufficient to identify regions of elevated SNP density and produce correct results with recombination detection methods,

Using 240 genome assemblies as input, SKA2 can rapidly create an input alignment against a reference chromosome (56 sec, 3.6 GB RAM), ∼190× faster than using snippy on all samples (178 min). Using this as the input to Gubbins gives similar recombination signals to the original results (Supplemental Fig. 6), which were based on short-read mapping (similar to the BWA + BCFtools pipeline). As in the original study, the major peaks spanning prophage ϕMM1-2008, the mobile element ICESpn23FST81, and the *cps* operon were all detected. The only major locus not detected was the *psrP* gene. This gene mostly consists of short two to three amino acid serine-containing repeats, and hence, this low complexity repeat sequence that is longer than the split *k*-mer length cannot be mapped by SKA2 (Fig. 2). This repeat is longer than 63 bases, so even using the maximum $k = 63$ is unable to recover these SNPs.

Split *k*-mers can be filtered to allow "online" outbreak analysis with large numbers of samples

Noting the computational efficiency of SKA2, in particular with respect to file sizes, we then tested the suitability of SKA2 for "online" (or serial) analysis in which genomes are added to the existing in batches, rather than the entire data set being reanalyzed. This is the dominant mode by which genome data now arrive for bacterial pathogens, and supporting analysis in this

Table 1. Runtimes obtained on one CPU and maximum memory consumption in the analyses of simulated outbreaks

	S. pneumoniae (n = 12)		*M. tuberculosis* (n = 30)	
	Runtime (min)	Memory (MB)	Runtime (min)	Memory (MB)
BactSNP	59.6	1955	330.1	2059
Snippy	20.1	338	90.9	1434
BWA + BCFtools	19.8	**323**	77.4	**656**
SKA1 align	2.3	424	20.1	865
SKA2 align	**1.0**	443	**5.5**	893

Values represent averages over 20 replicates. (n) Number of samples per outbreak. The lowest values are highlighted in bold.

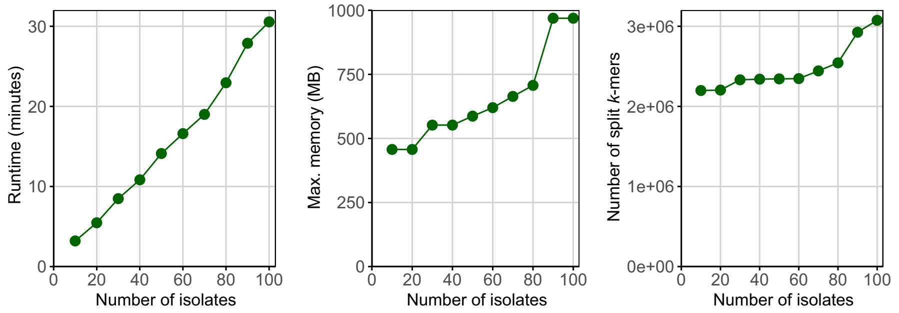

Figure 4. Empirical scaling of SKA2 computational efficiency using increasing block sizes from 100 isolates of the *S. pneumoniae* IC1 cluster. The numbers of split *k*-mers represent the total numbers of split *k*-mers contained across all samples.

manner will lower the computational burden, which is particularly relevant for democratizing access to rapid outbreak analysis. We also consider these results in the context of making "reference" SKFs available on a website (such as pathogen.watch) so users can use them to contextualize their new genomes. To be used for interactive internet tools, file sizes ≲10 MB are desirable. To efficiently store large amounts of data, we propose reducing the size of SKFs by filtering split *k*-mers using the *ska weed* function. Here, we tested the power lost using an online approach by using iterative analyses of 288 *Escherichia coli* genome assemblies, in which the SKF generated from the new genomes is merged to the previous SKF and then filtered to remove constant or rare split *k*-mers. In absence of ground truth, we considered in these analyses the numbers of SNPs obtained from standard SKA2 analyses (i.e., without split-*k*-mer filtering) as reference.

Initially, we added *E. coli* genome assemblies in random batches of 10, employing a split *k*-mer filtering based on consistently missing split *k*-mers (i.e., removing any rare split *k*-mers; parameter min-freq set to 0.8). The number of SNPs found at each iteration closely matched the results obtained through standard SKA2 analyses (with a maximum difference of 13 SNPs out of 16,905 SNPs at 100 genomes). Simultaneously, the size of SKFs was substantially reduced, ranging from twofold to 6.4-fold reductions after filtering in the first and last iterations, respectively (Fig. 5), reaching a file size of 212 MB for 288 genomes. For comparison, the 288 unprocessed and gzipped genome files represent a combined 454 MB of data. Similar outcomes were observed when adding genomes in batches of 50 and when starting with a data set of 200 or 250 genomes followed by additions in batches of five or one. These analyses affirm that SKA2 would be suitable for use in an online mode and even suitable for use in a web browser tool, as it can proficiently store substantial genomic data with negligible information loss, facilitating the analysis of ongoing outbreaks.

We repeated all these analyses using a stricter split *k*-mer filtering based on the presence–absence of SNPs, removing any constant or gap only sites (filters no-ambig-or-const and no-gap-only-sites). We observed a more substantial reduction in SKF sizes (ranging from 100-fold to 400-fold size reductions across all analyses), reaching 12–14 MB for 288 genomes, but the numbers of missed SNPs were significant and increased in each iteration (Fig. 5). Consequently, the filtering based on SNPs would only be suitable for the analysis, storage, and sharing of final data sets, for example fixed SNP-based typing schemes (Hawkey et al. 2021).

We also compared the pairwise SNP distances calculated by *ska distance* between the default SKA2 analysis and both online

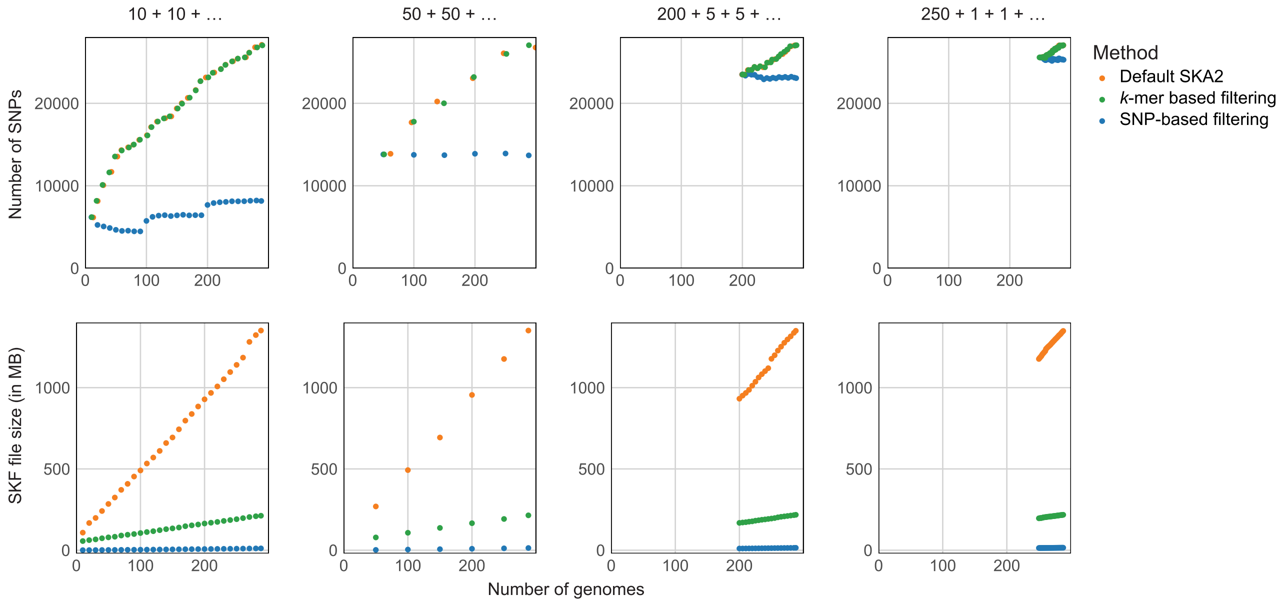

Figure 5. Online analyses of *E.coli* genomes. The three different genome addition strategies mentioned in the main text are displayed from *left* to *right*. Units of the *x*-axes (number of genomes) are identical across the six plots, and units of the *y*-axes (number of SNPs and SKF sizes) are identical within plots on the same line. "*k*-mer-based filtering" refers to the filtering based on missing split-*k*-mers, and "SNP-based filtering" refers to the filtering based on presence-absence of SNPs. Points corresponding to the number of SNPs (*upper* panels) obtained from default SKA2 analyses and after *k*-mer-based filtering were jittered to avoid overlapping.

modes at the final step with all 288 samples (Supplemental Fig. 7). This confirmed that SNP-based filtering loses too much information and biases distances downward as SNPs are systematically missed. *k*-mer-based filtering showed good correlation with analysis of the full data set, with results improving when the initial batch size was larger.

Discussion

Many popular tools in bioinformatics succeed by doing something relatively simple but doing it well. SKA2 uses exact matching of *k*-mers, which has been the focus of intense optimization efforts in bioinformatics, to create SNP alignments without explicitly performing any alignment. This is very similar to the pseudoalignment approach which has become the dominant form of analysis of RNA-seq data (Bray et al. 2016) and for representing population data (Holley and Melsted 2020; Alanko et al. 2023; Břinda et al. 2023). SKA2 has very similar benefits: speed, ease of use, robustness to structural variation, and reduced reference bias.

In analyses of simulated outbreaks, in which samples were closely related and thus most suited for split *k*-mer analysis, SKA2 showed higher sensitivity compared with mapping approaches that used distant strains as reference genomes. Another major advantage of SKA2 over classical alignment-based methods was that it avoided reference bias or sensitivity to parameters, generating no false positives. SKA2 was also faster than other methods, with runtimes more than an order of magnitude lower than those of read-alignment pipelines and representing at least a twofold improvement over SKA1. We further showed that iterative filtering of split *k*-mers represents a promising approach to track variation when streaming input samples, a potential solution for storing and analyzing the ever-growing amount of publicly available bacterial genome data.

The high accuracy and speed of SKA2 are complemented by its user-friendly workflow, requiring only two commands for standard operation: one for building split-*k*-mer indexes and the other for aligning split-*k*-mers. Additionally, akin to other alignment-free methods, outbreak investigations conducted using SKA2 eliminate the need for analysis steps such as manually masking low-complexity genomic regions (e.g., PE-PPE genes in *M. tuberculosis*) and expert selection of closely related reference genomes. Another practical advantage of SKA2 is the ability to directly and efficiently work with assembly rather than read data or to even combine the two. As sequence assemblies are becoming an increasingly popular archival form for bacterial genomes (Blackwell et al. 2021; Břinda et al. 2023; Hunt et al. 2024), they are more convenient for many users.

Algorithm scalability is of increasing importance to users. With hundreds of thousands of potential comparator genomes in the public domain (Hunt et al. 2024), users cannot afford to “just wait” for computation to complete, and many who would benefit from pathogen genomics do not have easy access to high-performance computing servers. This is also important for public health pathogen genomics, in which quick-turnaround time is essential to allow genomic data to be included in rapid public health decision making. Fast, simple methods that can run on regular laptops are particularly important in resource-limited settings where there is a lack of compute and skilled bioinformaticians to allow local genomic epidemiology in countries where most infectious disease outbreaks and burden are concentrated.

There is a clear potential to further develop SKA2 to accommodate emerging sequencing technologies. Although we did not test SKA2’s performance using sequencing reads with high error rates (e.g., Oxford Nanopore sequencing data), we would expect the current SKA2 error filtering to perform relatively poorly on such noisy data. However, assembling these data for use as input should still be suitable (Sanderson et al. 2023), and it is also expected that advances in this technology mean that high error rate reads will likely only be an issue in the short term. We also only looked at data from single colony picks, but the identification and analysis of samples composed of a mixture of strains from plate sweeps (Mäklin et al. 2020, 2021) or metagenomic samples are becoming more common (Richardson et al. 2023). SKA2 currently only saves the observed middle bases of each split *k*-mer, but a future version could also save their abundances, information that can potentially be used to deconvolute such mixtures. Also emerging are metagenome-assembled genomes (MAGs), which are typically more incomplete and contaminated than are colony pick-generated assemblies (Bickhart et al. 2022).

SKA2 remains under active development. Among future improvements, better compression of the output files in SKA2 is possible, which will be important when tracking variation in very large strain collections. Future versions of SKA2 will use a sparse and phylogenetically compressed (Břinda et al. 2023) data structure for the variant array. With many constant sites, this would be expected to yield an improvement and be usable during construction, too, saving memory and disk usage and making parallelization more effective. Various data structures for compressing *k*-mers also exist and could be applied to the lists of front and back halves of the split *k*-mers (Rahman et al. 2021; Břinda et al. 2023). Runtimes of analyses based on sequencing reads could also be further reduced by limiting the number of reads to be analyzed using a stopping criterion as implemented in other tools (Peterlongo et al. 2017; Derelle et al. 2023) and using a cache-optimized filter (Holley and Melsted 2020). Finally, the inference of variants by SKA2 is limited to SNPs, and indels or structural variants cannot be detected using the current implementation. Such variants could be inferred by repeating the split *k*-mer analysis at various split sizes and matching flanking bases across a middle region of various lengths, similar to haplotyping approaches (Garrison and Marth 2012), or by using split *k*-mers to create a de Bruijn graph as employed in other alignment-free variant-calling methods (Iqbal et al. 2013; Fang et al. 2016; Peterlongo et al. 2017). Finally, SKA2’s execution pipeline, parameters, and downstream programs still require bioinformatics training. In the future, we hope to relax this requirement further with a web-browser-executable version of the code.

We originally released SKA in 2018 and SKA2 in 2022, so we have had time to work with users to improve testing, address unexpected results, and identify edge cases. Creating reliable results with well-chosen defaults, offering the right amount of flexibility in the interface, good documentation, and long-term maintenance potential are also important to users, an important focus of our work that has only been touched upon in this paper. This has culminated in a high-quality implementation that we are confident will continue to deliver accurate results as genomic sequencing of bacteria continues to grow.

Software availability

SKA2 is implemented in Rust and is freely available with an Apache 2.0 license at GitHub (https://github.com/bacpop/ska.rust) and as Supplemental Code. Installation is also possible via crates.io, Bioconda, or directly from source. Documentation, including

API code for reuse in other software, is available at https://docs.rs/ska/latest/ska/. A tutorial is available at https://www.bacpop.org/guides/building_trees_with_ska/. Analysis code used to generate the figures in the paper is available at GitHub (https://github.com/bacpop/ska_simulations and https://github.com/rderelle/compareALI) and as Supplemental Code.

Competing interest statement

The authors declare no competing interests.

Acknowledgments

J.v.W., T.M., J.H., and J.A.L. were supported by the European Molecular Biology Laboratory. T.M. was also supported by the Academy of Finland EuroHPC grant. R.D. was supported by the National Institute for Health and Care Research Health Protection Research Unit in Respiratory Infections, in partnership with the UK Health Security Agency (NIHR200927). N.J.C. was supported by the UK Medical Research Council and Department for International Development (grants MR/R015600/1 and MR/T016434/1). L.C. and N.J.C. acknowledge funding from the MRC Centre for Global Infectious Disease Analysis (reference MR/X020258/1 and MR/T016434/1), funded by the UK Medical Research Council (MRC). This UK funded award is carried out in the frame of the Global Health EDCTP3 Joint Undertaking.

Author contributions: Conceptualization was by R.D., S.R.H., and J.A.L. Data curation was by R.D. and J.A.L. Formal analysis was by R.D. and J.A.L. Funding acquisition was by A.L., L.C., N.J.C., and J.A.L. Investigation was by R.D. and J.A.L. Methodology was by R.D., J.v.W., T.M., J.H., T.R., S.R.H., and J.A.L. Project administration was by J.A.L. Resources were by A.L., L.C., N.J.C., and J.A.L. Software was by R.D., J.v.W., T.M., S.R.H., and J.A.L. Supervision was by J.A.L. Validation was by N.J.C. and J.A.L. Visualization was by R.D. and J.A.L. Writing of the original draft was by R.D. and J.A.L. Reviewing and editing were by all the authors.

References

Alanko JN, Vuohtoniemi J, Mäklin T, Puglisi SJ. 2023. Themisto: a scalable colored *k*-mer index for sensitive pseudoalignment against hundreds of thousands of bacterial genomes. *Bioinformatics* **39(Supplement_1):** i260–i269. doi:10.1093/bioinformatics/btad233

Allard MW, Strain E, Melka D, Bunning K, Musser SM, Brown EW, Timme R. 2016. Practical value of food pathogen traceability through building a whole-genome sequencing network and database. *J Clin Microbiol* **54:** 1975–1983. doi:10.1128/JCM.00081-16

Argimón S, Abudahab K, Goater RJE, Fedosejev A, Bhai J, Glasner C, Feil EJ, Holden MTG, Yeats CA, Grundmann H, et al. 2016. Microreact: visualizing and sharing data for genomic epidemiology and phylogeography. *Microb Genom* **2:** e000093. doi:10.1099/mgen.0.000093

Becker HEF, Jamin C, Bervoets L, Boleij A, Xu P, Pierik MJ, Stassen FRM, Savelkoul PHM, Penders J, Jonkers DMAE. 2021. Higher prevalence of *Bacteroides fragilis* in Crohn's disease exacerbations and strain-dependent increase of epithelial resistance. *Front Microbiol* **12:** 598232. doi:10.3389/fmicb.2021.598232

Bickhart DM, Kolmogorov M, Tseng E, Portik DM, Korobeynikov A, Tolstoganov I, Uritskiy G, Liachko I, Sullivan ST, Shin SB, et al. 2022. Generating lineage-resolved, complete metagenome-assembled genomes from complex microbial communities. *Nat Biotechnol* **40:** 711–719. doi:10.1038/s41587-021-01130-z

Blackwell GA, Hunt M, Malone KM, Lima L, Horesh G, Alako BTF, Thomson NR, Iqbal Z. 2021. Exploring bacterial diversity via a curated and searchable snapshot of archived DNA sequences. *PLoS Biol* **19:** e3001421. doi:10.1371/journal.pbio.3001421

Bray NL, Pimentel H, Melsted P, Pachter L. 2016. Near-optimal probabilistic RNA-seq quantification. *Nat Biotechnol* **34:** 525–527. doi:10.1038/nbt.3519

Břinda K. 2013. *Lossless seeds for approximate string matching*. Czech Technical University in Prague. doi:10.5281/zenodo.12750412

Břinda K, Lima L, Pignotti S, Quinones-Olvera N, Salikhov K, Chikhi R, Kucherov G, Iqbal Z, Baym M. 2023. Efficient and robust search of microbial genomes via phylogenetic compression. bioRxiv doi:10.1101/2023.04.15.536996

Bush SJ, Foster D, Eyre DW, Clark EL, De Maio N, Shaw LP, Stoesser N, Peto TEA, Crook DW, Walker AS. 2020. Genomic diversity affects the accuracy of bacterial single-nucleotide polymorphism–calling pipelines. *GigaScience* **9:** giaa007. doi:10.1093/gigascience/giaa007

Chew KL, Achik R, Osman NH, Octavia S, Teo JWP. 2023. Genomic epidemiology of human candidaemia isolates in a tertiary hospital. *Microb Genom* **9:** mgen001047. doi:10.1099/mgen.0.001047

Chewapreecha C, Harris SR, Croucher NJ, Turner C, Marttinen P, Cheng L, Pessia A, Aanensen DM, Mather AE, Page AJ, et al. 2014. Dense genomic sampling identifies highways of pneumococcal recombination. *Nat Genet* **46:** 305–309. doi:10.1038/ng.2895

Colquhoun RM, Hall MB, Lima L, Roberts LW, Malone KM, Hunt M, Letcher B, Hawkey J, George S, Pankhurst L, et al. 2021. Pandora: nucleotide-resolution bacterial pan-genomics with reference graphs. *Genome Biol* **22:** 267. doi:10.1186/s13059-021-02473-1

Comas I, Coscolla M, Luo T, Borrell S, Holt KE, Kato-Maeda M, Parkhill J, Malla B, Berg S, Thwaites G, et al. 2013. Out-of-Africa migration and neolithic coexpansion of *Mycobacterium tuberculosis* with modern humans. *Nat Genet* **45:** 1176–1182. doi:10.1038/ng.2744

Cori A, Nouvellet P, Garske T, Bourhy H, Nakouné E, Jombart T. 2018. A graph-based evidence synthesis approach to detecting outbreak clusters: an application to dog rabies. *PLoS Comput Biol* **14:** e1006554. doi:10.1371/journal.pcbi.1006554

Croucher NJ, Walker D, Romero P, Lennard N, Paterson GK, Bason NC, Mitchell AM, Quail MA, Andrew PW, Parkhill J, et al. 2009. Role of conjugative elements in the evolution of the multidrug-resistant pandemic clone *Streptococcus pneumoniae*$^{\mathrm{Spain23F}}$ ST81. *J Bacteriol* **191:** 1480–1489. doi:10.1128/JB.01343-08

Croucher NJ, Harris SR, Fraser C, Quail MA, Burton J, van der Linden M, McGee L, von Gottberg A, Song JH, Ko KS, et al. 2011a. Rapid pneumococcal evolution in response to clinical interventions. *Science* **331:** 430–434. doi:10.1126/science.1198545

Croucher NJ, Vernikos GS, Parkhill J, Bentley SD. 2011b. Identification, variation and transcription of pneumococcal repeat sequences. *BMC Genomics* **12:** 120. doi:10.1186/1471-2164-12-120

Croucher NJ, Hanage WP, Harris SR, McGee L, van der Linden M, de Lencastre H, Sá-Leão R, Song J-H, Ko KS, Beall B, et al. 2014. Variable recombination dynamics during the emergence, transmission and "disarming" of a multidrug-resistant pneumococcal clone. *BMC Biol* **12:** 49. doi:10.1186/1741-7007-12-49

Croucher NJ, Page AJ, Connor TR, Delaney AJ, Keane JA, Bentley SD, Parkhill J, Harris SR. 2015. Rapid phylogenetic analysis of large samples of recombinant bacterial whole genome sequences using Gubbins. *Nucleic Acids Res* **43:** e15. doi:10.1093/nar/gku1196

De Maio N, Boulton W, Weilguny L, Walker CR, Turakhia Y, Corbett-Detig R, Goldman N. 2022. phastSim: efficient simulation of sequence evolution for pandemic-scale datasets. *PLoS Comput Biol* **18:** e1010056. doi:10.1371/journal.pcbi.1010056

Derelle R, Lees J, Phelan J, Lalvani A, Arinaminpathy N, Chindelevitch L. 2023. fastlin: an ultra-fast program for *Mycobacterium tuberculosis* complex lineage typing. *Bioinformatics* **39:** btad648. doi:10.1093/bioinformatics/btad648

Didelot X, Wilson DJ. 2015. ClonalFrameML: efficient inference of recombination in whole bacterial genomes. *PLoS Comput Biol* **11:** e1004041. doi:10.1371/journal.pcbi.1004041

Didelot X, Fraser C, Gardy J, Colijn C. 2017. Genomic infectious disease epidemiology in partially sampled and ongoing outbreaks. *Mol Biol Evol* **34:** 997–1007. doi:10.1093/molbev/msw275

Drezen E, Rizk G, Chikhi R, Deltel C, Lemaitre C, Peterlongo P, Lavenier D. 2014. GATB: genome assembly & analysis tool box. *Bioinformatics* **30:** 2959–2961. doi:10.1093/bioinformatics/btu406

Duchêne S, Holt KE, Weill F-X, Le Hello S, Hawkey J, Edwards DJ, Fourment M, Holmes EC. 2016. Genome-scale rates of evolutionary change in bacteria. *Microb Genom* **2:** e000094. doi:10.1099/mgen.0.000094

Falconer C, Cuddihy T, Beatson SA, Paterson DL, Harris PNA, Forde BM. 2022. Systematic benchmarking of "all-in-one" microbial SNP calling pipelines. bioRxiv doi:10.1101/2022.05.05.487569

Fang H, Bergmann EA, Arora K, Vacic V, Zody MC, Iossifov I, O'Rawe JA, Wu Y, Jimenez Barron LT, Rosenbaum J, et al. 2016. Indel variant analysis of short-read sequencing data with scalpel. *Nat Protoc* **11:** 2529–2548. doi:10.1038/nprot.2016.150

Farrer RA, Henk DA, MacLean D, Studholme DJ, Fisher MC. 2013. Using false discovery rates to benchmark SNP-callers in next-generation sequencing projects. *Sci Rep* **3:** 1512. doi:10.1038/srep01512

Fletcher R. 1970. A new approach to variable metric algorithms. *Comput J* **13:** 317–322. doi:10.1093/comjnl/13.3.317

Gagneux S. 2018. Ecology and evolution of *Mycobacterium tuberculosis*. *Nat Rev Microbiol* **16:** 202–213. doi:10.1038/nrmicro.2018.8

Gardner SN, Hall BG. 2013. When whole-genome alignments just won't work: kSNP v2 software for alignment-free SNP discovery and phylogenetics of hundreds of microbial genomes. *PLoS One* **8:** e81760. doi:10.1371/journal.pone.0081760

Gardner SN, Slezak T, Hall BG. 2015. kSNP3.0: SNP detection and phylogenetic analysis of genomes without genome alignment or reference genome. *Bioinformatics* **31:** 2877–2878. doi:10.1093/bioinformatics/btv271

Gardy JL, Johnston JC, Ho Sui SJ, Cook VJ, Shah L, Brodkin E, Rempel S, Moore R, Zhao Y, Holt R, et al. 2011. Whole-genome sequencing and social-network analysis of a tuberculosis outbreak. *N Engl J Med* **364:** 730–739. doi:10.1056/NEJMoa1003176

Garrison E, Marth G. 2012. Haplotype-based variant detection from short-read sequencing. arXiv:1207.3907 [q-bio.GN]. doi:10.48550/arXiv.1207.3907

Gillespie DT. 1977. Exact stochastic simulation of coupled chemical reactions. *J Phys Chem* **81:** 2340–2361. doi:10.1021/j100540a008

Gladstone RA, McNally A, Pöntinen AK, Tonkin-Hill G, Lees JA, Skytén K, Cléon F, Christensen MOK, Haldorsen BC, Bye KK, et al. 2021. Emergence and dissemination of antimicrobial resistance in *Escherichia coli* causing bloodstream infections in Norway in 2002-17: a nationwide, longitudinal, microbial population genomic study. *Lancet Microbe* **2:** e331–e341. doi:10.1016/S2666-5247(21)00031-8

Grad YH, Lipsitch M, Feldgarden M, Arachchi HM, Cerqueira GC, Fitzgerald M, Godfrey P, Haas BJ, Murphy CI, Russ C, et al. 2012. Genomic epidemiology of the *Escherichia coli* O104:H4 outbreaks in Europe, 2011. *Proc Natl Acad Sci* **109:** 3065–3070. doi:10.1073/pnas.1121491109

Gupta A, Jordan IK, Rishishwar L. 2017. stringMLST: a fast k-mer based tool for multilocus sequence typing. *Bioinformatics* **33:** 119–121. doi:10.1093/bioinformatics/btw586

Hadfield J, Croucher NJ, Goater RJ, Abudahab K, Aanensen DM, Harris SR. 2018. Phandango: an interactive viewer for bacterial population genomics. *Bioinformatics* **34:** 292–293. doi:10.1093/bioinformatics/btx610

Hall BG, Nisbet J. 2023. Building phylogenetic trees from genome sequences with kSNP4. *Mol Biol Evol* **40:** msad235. doi:10.1093/molbev/msad235

Harris SR. 2018. SKA: split kmer analysis toolkit for bacterial genomic epidemiology. bioRxiv doi:10.1101/453142

Harris SR, Feil EJ, Holden MTG, Quail MA, Nickerson EK, Chantratita N, Gardete S, Tavares A, Day N, Lindsay JA, et al. 2010. Evolution of MRSA during hospital transmission and intercontinental spread. *Science* **327:** 469–474. doi:10.1126/science.1182395

Hawkey J, Paranagama K, Baker KS, Bengtsson RJ, Weill F-X, Thomson NR, Baker S, Cerdeira L, Iqbal Z, Hunt M, et al. 2021. Global population structure and genotyping framework for genomic surveillance of the major dysentery pathogen, *Shigella sonnei*. *Nat Commun* **12:** 2684. doi:10.1038/s41467-021-22700-4

Holley G, Melsted P. 2020. Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. *Genome Biol* **21:** 249. doi:10.1186/s13059-020-02135-8

Huang W, Li L, Myers JR, Marth GT. 2012. ART: a next-generation sequencing read simulator. *Bioinformatics* **28:** 593–594. doi:10.1093/bioinformatics/btr708

Hunt M, Lima L, Shen W, Lees J, Iqbal Z. 2024. AllTheBacteria: all bacterial genomes assembled, available and searchable. bioRxiv doi:10.1101/2024.03.08.584059

Hurgobin B, Edwards D. 2017. SNP discovery using a pangenome: Has the single reference approach become obsolete? *Biology (Basel)* **6:** 21. doi:10.3390/biology6010021

Iqbal Z, Turner I, McVean G. 2013. High-throughput microbial population genomics using the cortex variation assembler. *Bioinformatics* **29:** 275–276. doi:10.1093/bioinformatics/bts673

Kremer PHC, Lees JA, Koopmans MM, Ferwerda B, Arends AWM, Feller MM, Schipper K, Valls Seron M, van der Ende A, Brouwer MC, et al. 2017. Benzalkonium tolerance genes and outcome in *Listeria monocytogenes* meningitis. *Clin Microbiol Infect* **23:** 265.e1–265.e7. doi:10.1016/j.cmi.2016.12.008

Kucherov G, Noé L, Roytberg M. 2005. Multiseed lossless filtration. *IEEE/ACM Trans Comput Biol Bioinform* **2:** 51–61. doi:10.1109/TCBB.2005.12

Labbé G, Kruczkiewicz P, Robertson J, Mabon P, Schonfeld J, Kein D, Rankin MA, Gopez M, Hole D, Son D, et al. 2021. Rapid and accurate SNP genotyping of clonal bacterial pathogens with BioHansel. *Microb Genom* **7:** 000651. doi:10.1099/mgen.0.000651

Landan G, Graur D. 2009. Characterization of pairwise and multiple sequence alignment errors. *Gene* **441:** 141–147. doi:10.1016/j.gene.2008.05.016

Lees JA, Vehkala M, Välimäki N, Harris SR, Chewapreecha C, Croucher NJ, Marttinen P, Davies MR, Steer AC, Tong SYC, et al. 2016. Sequence element enrichment analysis to determine the genetic basis of bacterial phenotypes. *Nat Commun* **7:** 12797. doi:10.1038/ncomms12797

Lees JA, Kendall M, Parkhill J, Colijn C, Bentley SD, Harris SR. 2018. Evaluation of phylogenetic reconstruction methods using bacterial whole genomes: a simulation based study. *Wellcome Open Res* **3:** 33. doi:10.12688/wellcomeopenres.14265.2

Lees JA, Harris SR, Tonkin-Hill G, Gladstone RA, Lo SW, Weiser JN, Corander J, Bentley SD, Croucher NJ. 2019. Fast and flexible bacterial genomic epidemiology with PopPUNK. *Genome Res* **29:** 304–316. doi:10.1101/gr.241455.118

Li H. 2011. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics* **27:** 2987–2993. doi:10.1093/bioinformatics/btr509

Li H. 2013. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997 [q-bio.GN]. doi:10.48550/arXiv.1303.3997

Maechler F, Weber A, Schwengers O, Schwab F, Denkel L, Behnke M, Gastmeier P, Kola A. 2023. Split k-mer analysis compared to cgMLST and SNP-based core genome analysis for detecting transmission of vancomycin-resistant enterococci: results from routine outbreak analyses across different hospitals and hospitals networks in Berlin, Germany. *Microb Genom* **9:** mgen000937. doi:10.1099/mgen.0.000937

Mäklin T, Kallonen T, David S, Boinett CJ, Pascoe B, Méric G, Aanensen DM, Feil EJ, Baker S, Parkhill J, et al. 2020. High-resolution sweep metagenomics using fast probabilistic inference. *Wellcome Open Res* **5:** 14. doi:10.12688/wellcomeopenres.15639.1

Mäklin T, Kallonen T, Alanko J, Samuelsen Ø, Hegstad K, Mäkinen V, Corander J, Heinz E, Honkela A. 2021. Bacterial genomic epidemiology with mixed samples. *Microb Genom* **7:** 000691. doi:10.1099/mgen.0.000691

Menardo F, Duchêne S, Brites D, Gagneux S. 2019. The molecular clock of *Mycobacterium tuberculosis*. *PLoS Pathog* **15:** e1008067. doi:10.1371/journal.ppat.1008067

Minh BQ, Schmidt HA, Chernomor O, Schrempf D, Woodhams MD, von Haeseler A, Lanfear R. 2020. IQ-TREE 2: new models and efficient methods for phylogenetic inference in the genomic era. *Mol Biol Evol* **37:** 1530–1534. doi:10.1093/molbev/msaa015

Mohamadi H, Chu J, Vandervalk BP, Birol I. 2016. ntHash: recursive nucleotide hashing. *Bioinformatics* **32:** 3492–3494. doi:10.1093/bioinformatics/btw397

Morris AJ, Yau YCW, Park S, Eisha S, McDonald N, Parsek MR, Howell PL, Hoffman LR, Nguyen D, DiGiandomenico A, et al. 2022. *Pseudomonas aeruginosa* aggregation and Psl expression in sputum is associated with antibiotic eradication failure in children with cystic fibrosis. *Sci Rep* **12:** 21444. doi:10.1038/s41598-022-25889-6

Mostowy R, Croucher NJ, Andam CP, Corander J, Hanage WP, Marttinen P. 2017. Efficient inference of recent and ancestral recombination within bacterial populations. *Mol Biol Evol* **34:** 1167–1182. doi:10.1093/molbev/msx066

Paten B, Novak AM, Eizenga JM, Garrison E. 2017. Genome graphs and the evolution of genome inference. *Genome Res* **27:** 665–676. doi:10.1101/gr.214155.116

Peterlongo P, Riou C, Drezen E, Lemaitre C. 2017. *discosnp++*: de novo detection of small variants from raw unassembled read set(s). bioRxiv doi:10.1101/209965

Pightling AW, Petronella N, Pagotto F. 2014. Choice of reference sequence and assembler for alignment of *Listeria monocytogenes* short-read sequence data greatly influences rates of error in SNP analyses. *PLoS One* **9:** e104579. doi:10.1371/journal.pone.0104579

Qiao Y, Li T, Chen S. 2014. Fast bloom filters and their generalization. *IEEE Trans Parallel Distrib Syst* **25:** 93–103. doi:10.1109/TPDS.2013.46

Quick J, Loman NJ, Duraffour S, Simpson JT, Severi E, Cowley L, Bore JA, Koundouno R, Dudas G, Mikhail A, et al. 2016. Real-time, portable genome sequencing for Ebola surveillance. *Nature* **530:** 228–232. doi:10.1038/nature16996

Quinlan AR, Hall IM. 2010. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26:** 841–842. doi:10.1093/bioinformatics/btq033

Rahman A, Chikhi R, Medvedev P. 2021. Disk compression of *k*-mer sets. *Algorithms Mol Biol* **16:** 10. doi:10.1186/s13015-021-00192-7

Richardson L, Allen B, Baldi G, Beracochea M, Bileschi ML, Burdett T, Burgin J, Caballero-Pérez J, Cochrane G, Colwell LJ, et al. 2023. MGnify: the microbiome sequence data analysis resource in 2023. *Nucleic Acids Res* **51:** D753–D759. doi:10.1093/nar/gkac1080

Sanderson ND, Kapel N, Rodger G, Webster H, Lipworth S, Street TL, Peto T, Crook D, Stoesser N. 2023. Comparison of R9.4.1/Kit10 and R10/Kit12 Oxford nanopore flowcells and chemistries in bacterial genome reconstruction. *Microb Genom* **9:** mgen000910. doi:10.1099/mgen.0.000910

Simonsen M, Mailund T, Pedersen CNS. 2008. Rapid neighbour-joining. *Algorithms in Bioinformatics* **5251:** 113–122. doi:10.1007/978-3-540-87361-7_10

Sipos B, Massingham T, Jordan GE, Goldman N. 2011. PhyloSim: Monte Carlo simulation of sequence evolution in the R statistical computing environment. *BMC Bioinformatics* **12:** 104. doi:10.1186/1471-2105-12-104

Smith MR. 2020. Information theoretic generalized Robinson–Foulds metrics for comparing phylogenetic trees. *Bioinformatics* **36:** 5007–5013. doi:10.1093/bioinformatics/btaa614

Smith MR. 2022. Robust analysis of phylogenetic tree space. *Syst Biol* **71:** 1255–1270. doi:10.1093/sysbio/syab100

Usongo V, Berry C, Yousfi K, Doualla-Bell F, Labbé G, Johnson R, Fournier E, Nadon C, Goodridge L, Bekal S. 2018. Impact of the choice of reference genome on the ability of the core genome SNV methodology to distinguish strains of *Salmonella enterica* serovar Heidelberg. *PLoS One* **13:** e0192233. doi:10.1371/journal.pone.0192233

Valiente-Mullor C, Beamud B, Ansari I, Francés-Cuesta C, García-González N, Mejía L, Ruiz-Hueso P, González-Candelas F. 2021. One is not enough: on the effects of reference genome for the mapping and subsequent analyses of short-reads. *PLoS Comput Biol* **17:** e1008678. doi:10.1371/journal.pcbi.1008678

Walter KS, Colijn C, Cohen T, Mathema B, Liu Q, Bowers J, Engelthaler DM, Narechania A, Lemmer D, Croda J, et al. 2020. Genomic variant-identification methods may alter *Mycobacterium tuberculosis* transmission inferences. *Microb Genom* **6:** mgen000418. doi:10.1099/mgen.0.000418

Wymant C, Hall M, Ratmann O, Bonsall D, Golubchik T, de Cesare M, Gall A, Cornelissen M, Fraser C. 2018. PHYLOSCANNER: inferring transmission from within- and between-host pathogen genetic diversity. *Mol Biol Evol* **35:** 719–733. doi:10.1093/molbev/msx304

Yoshimura D, Kajitani R, Gotoh Y, Katahira K, Okuno M, Ogura Y, Hayashi T, Itoh T. 2019. Evaluation of SNP calling methods for closely related bacterial isolates and a novel high-accuracy pipeline: BactSNP. *Microb Genom* **5:** e000261. doi:10.1099/mgen.0.000261

Received April 8, 2024; accepted in revised form September 16, 2024.