Published online 3 July 2020

Nucleic Acids Research, 2020, Vol. 48, No. 14 ***7681–7689***
doi: 10.1093/nar/gkaa566

STing: accurate and ultrafast genomic profiling with exact sequence matches

Hector F. Espitia-Navarro[1,2,3], Aroon T. Chande[1,2,3], Shashwat D. Nagar[1,2], Heather Smith[4,5], I. King Jordan[1,2,3] and Lavanya Rishishwar[1,2,3,*]

[1]School of Biological Sciences, Georgia Institute of Technology, Atlanta, GA 30332, USA, [2]PanAmerican Bioinformatics Institute, Cali, Valle del Cauca 760043, Colombia, [3]Applied Bioinformatics Laboratory, Atlanta, GA 30332, USA, [4]School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA and [5]Department of Mathematics and Computer Science, Davidson College, Davidson, NC 28035, USA

Received April 13, 2020; Revised June 16, 2020; Editorial Decision June 22, 2020; Accepted July 01, 2020

ABSTRACT

Genome-enabled approaches to molecular epidemiology have become essential to public health agencies and the microbial research community. We developed the algorithm STing to provide turn-key solutions for molecular typing and gene detection directly from next generation sequence data of microbial pathogens. Our implementation of STing uses an innovative *k*-mer search strategy that eliminates the computational overhead associated with the time-consuming steps of quality control, assembly, and alignment, required by more traditional methods. We compared STing to six of the most widely used programs for genome-based molecular typing and demonstrate its ease of use, accuracy, speed and efficiency. STing shows superior accuracy and performance for standard multilocus sequence typing schemes, along with larger genome-scale typing schemes, and it enables rapid automated detection of antimicrobial resistance and virulence factor genes. STing determines the sequence type of traditional 7-gene MLST with 100% accuracy in less than 10 seconds per isolate. We hope that the adoption of STing will help to democratize microbial genomics and thereby maximize its benefit for public health.

INTRODUCTION

Molecular typing entails the identification of distinct evolutionary lineages (i.e. types) within species of bacterial pathogens; it is an essential element of both outbreak investigation and routine infectious disease surveillance (1,2). Multilocus sequence typing (MLST) was developed as the first sequence-based approach to molecular typing in 1998 (3). Initially, MLST schemes relied on Sanger sequencing of PCR amplicons from fragments of 7–9 housekeeping genes spread throughout the genome. Briefly, each new distinct sequence of a housekeeping gene (locus) that is characterized in a given species, is uniquely identified by an integer number. The combination of the 7–9 numbers denotes an allelic profile which is uniquely tagged with an integer number that corresponds to a sequence type (ST). While this approach truly revolutionized molecular epidemiology, it is time consuming and costly compared to current next generation sequencing (NGS) methods. Nevertheless, MLST remains widely used for molecular typing, particularly in light of valuable legacy data relating STs to epidemiological information.

Public health agencies increasingly couple NGS characterization of microbial genomes with downstream bioinformatics analysis methods to perform molecular typing. The overhead associated with the bioinformatics methods that are used for this purpose, in terms of both the required human expertise and computational resources, represents a critical bottleneck that continues to limit the potential impact of microbial genomics on public health. This is particularly true for local public health agency laboratories, which are typically staffed with microbiologists who may not have substantial bioinformatics expertise or ready access to high-performance computational resources. In light of this ongoing challenge, our group is working to develop turn-key solutions to genome-enabled molecular epidemiology, including both molecular typing and the detection of critical antimicrobial resistance (AMR) and virulence factor (VF) genes. Methods of this kind must be easy to use, computationally efficient, fast, and most importantly, highly accurate.

We previously developed stringMLST as an alternative approach to genome-enabled molecular typing of bacterial pathogens (4). stringMLST relied on *k*-mer matching between NGS sequence reads and a database of MLST allele sequences, thereby eliminating the need for the sequence quality control, genome assembly, and alignment steps that the first generation of genome-enabled typing algorithms

*To whom correspondence should be addressed. Tel: +1 678 938 0844; Email: lavanya.rishishwar@gatech.edu

used. It proved to be accurate and fast for traditional MLST schemes, but it did not scale well to larger genome-scale typing schemes, such as ribosomal MLST (rMLST) or core-genome MLST (cgMLST), which are increasingly used in molecular epidemiology (1,5). Here, we present our new approach to this problem – STing. The STing algorithm is distinguished from its predecessor in several important ways: the efficiency of its code base, the underlying data structure that it uses, and the scope of its applications. These innovations provide for superior accuracy and performance compared to both stringMLST and other widely used programs for genome-enabled molecular typing.

MATERIALS AND METHODS

STing overview

Given an input sequence read file from a microbial isolate, STing can accurately identify the specific sequence type (ST), e.g. multilocus sequence type or its variants, for the isolate, and what genes of interest are present in its genome. STing accomplishes these tasks by using an exact *k*-mer matching and frequency counting paradigm. STing is implemented in C++ and utilizes two libraries: the SeqAn library (6) for the Enhanced Suffix Array (ESA) (7) data structure and the gzstream library (https://www.cs.unc.edu/Research/compgeom/gzstream/) for working with gz files. Additionally, STing is prepackaged with an R script for visualization of the results and a Python script for downloading database sequences from PubMLST. The ESA data structure is used for *k*-mer look-up and comparison purposes. ESAs are a lexicographically sorted array-based data structure, which represent space-efficient implementation of the Suffix Trees data structure. For a given set of sequences with a total length of n base pairs (summation of lengths of all sequences), an ESA index can be constructed in linear time $O(n)$. ESAs can also be queried for *k*-mer matches (or substring matches) in linear time. Given a *k*-mer of length k, we can determine its presence/absence in the database in $O(k)$ time and find all of its z occurrences in $O(k + z)$ time. While Suffix Trees achieve the same time complexity for index construction and *k*-mer lookup, they take five times more storage space than ESAs. An efficient implementation of a Suffix Tree can use up to 20 bytes per input database character, whereas an equivalent ESA consumes 4 bytes per input database character. Using ESAs for *k*-mer lookup and comparison allows STing to efficiently scale with large sequence databases. The STing algorithm is divided into three steps: (i) database indexing, (ii) sequence typing and (iii) gene detection (Supplementary Figure S1). See the online Supplementary Data and Supplementary Notes for detailed descriptions of the inputs, outputs and algorithms used.

Genomic data for sequence typing

We used 1050 Illumina sequencing read sets of isolates from four bacterial species (*Campylobacter jejuni*, *Chlamydia trachomatis*, *Neisseria meningitidis* and *Streptococcus pneumoniae*) retrieved from the PubMLST (https://pubmlst.org/)/EBI ENA (https://www.ebi.ac.uk/ena) database to execute the experiments (Supplementary Data). Using the isolate metadata available on PubMLST, we selected 40 samples from the four species (10 samples each) for the MLST comparative test, and 20 samples of *N. meningitidis* isolates for the larger typing schemes (rMLST and cgMLST) comparative test. We selected these two datasets so as to capture the diversity of the most common STs of each species in the PubMLST database and preferred recently sequenced isolates. For the large-scale accuracy test, we used a dataset of 1000 samples of *N. meningitidis* isolates.

Computational environment

We used a machine provided with RedHat Linux OS, 24 cores and 64GB of RAM to perform the experiments described in this study.

MLST comparative test design

To measure the performance of our application on the traditional seven loci MLST analysis, we compared STing (v0.24.2) in two execution modes, fast and sensitive, along with six applications able to perform sequence typing (stringMLST (4), MentaLiST (8), Kestrel (9), SRST2 (10), ARIBA (11) and Offline CGE (Supplementary Tables S1 and S2). These applications can be classified into five groups depending on the strategy (algorithmic paradigm) used to predict the sequence types of whole genome sequencing data samples from bacterial isolates: *k*-mer, *k*-mer + alignment, mapping, mapping + local assembly and assembly (Supplementary Table S3). For the Offline CGE application, we used the script runMLST.py (https://github.com/widdowquinn/scripts/blob/master/bioinformatics/run_MLST.py), an offline implementation of the original alignment-based MLST method from the Center of Genomic Epidemiology (12). This implementation uses multithreaded BLAST searching for the MLST analysis, as opposed to STing, which is a single thread application. For a fair comparison between STing and the Offline CGE/DTU implementation, we modified the script runMLST.py to use only one thread for BLAST searches (https://doi.org/10.5281/zenodo.3604226). For each application, we measured the accuracy in terms of the percentage of alleles correctly predicted from the total samples analyzed and the performance in terms of average run time and average peak RAM required to analyze each of the 40 samples in the dataset. We reported the average run time and average maximum RAM as the average of three executions of each application per sample analyzed. Kestrel requires the generation of a *k*-mer counts file before it can be run to predict STs. For this purpose, we used the application KAnalyze (13) (v2.0.0) with the parameters as described (9). We reported the average run time of Kestrel as the sum of the average times of KAnalyze and Kestrel for processing each sample and the average RAM consumption as the maximum average peak of RAM consumed by the two applications on each sample. Since the Offline CGE application requires complete assemblies to predict STs, we assembled each isolate read sample using the application SPAdes (14) (v3.13.0) with default parameters. We reported the average runtime as the sum of the aver-

age times of SPAdes and Offline CGE to process each sample and the average RAM consumption as the maximum average peak of RAM consumed between the two applications during the analysis of each sample. The commands used with each application tested are listed in the supplementary material (Supplementary Table S3).

Large-scale MLST accuracy test design

To measure the accuracy of our application using the MLST scheme on a large-scale dataset, we ran STing in fast mode on 1000 samples of *N. meningitidis*. We measured the accuracy in terms of the percentage of STs correctly predicted from the total samples analyzed and the performance in terms of average run time and average peak of RAM required to analyze each of the samples. We reported the average run time and average maximum RAM as the average of five executions of the application per sample analyzed.

Limit of detection, and performance on single and multicore environment test design

We evaluated the minimum sequencing depth required for correctly predicting STs on whole genome sequencing samples from bacterial isolates. We retrieved 1,872 assemblies of *Chlamydia trachomatis* ($n = 133$), *Campylobacter jejuni* ($n = 581$), *Neisseria meningitidis* ($n = 725$) and *Streptococcus pneumoniae* ($n = 433$) with known MLST information from the GenBank (https://www.ncbi.nlm.nih.gov/genbank/) database (Supplementary Data). Then, we simulated Illumina paired-end reads—HiSeq 2500, 2 × 150 bp, 500 bp of average fragment length, with 10 as the fragment size standard deviation – from each genome at seven sequencing depths (1×, 3×, 5×, 10×, 20× and 40×) using the software ART (15) (v2.5.8). We executed STing (fast mode) on each generated sample to measure the accuracy in terms of the percentage of correct STs and alleles predicted from the total samples at each sequencing depth. We also evaluated the performance of STing in multicore environments. We executed 20 parallel instances of STing to analyze the 1872 samples and measured the average time required to process the complete dataset at each sequencing depth.

Large-scale sequence typing schemes comparison test design

To evaluate the scalability, accuracy, and performance of our application on large-scale sequence typing schemes, we compared STing (fast and sensitive modes) on 20 samples of *N. meningitidis* against other sequence typing applications using the rMLST (loci = 53) and the cgMLST (loci = 1605) schemes. We used three applications (stringMLST, SRST2, and Offline CGE) for rMLST and three applications (stringMLST, MentaLiST and Offline CGE) for cgMLST; these applications were able to execute the sequence typing analysis successfully using these larger schemes. For each application and typing scheme, we measured the accuracy in terms of the percentage of correct allele predictions from the total alleles of the tested samples and the performance in terms of the average of run time and maximum RAM required to process each sample from the dataset.

Gene detection test design

We evaluated the ability of STing to predict the presence/absence of sequences of interest in NGS read samples by detecting antimicrobial resistance (AMR) genes and virulence factor (VF) genes in simulated Illumina read datasets. We retrieved 71 assemblies from the GenBank database that correspond to 25 species listed in the World Health Organization's priority list of antibiotic-resistant bacteria and tuberculosis (16) (Supplementary Data). Then, we simulated Illumina paired-end reads – HiSeq 2500, 2 × 150 bp, 500 bp of average fragment size, with 10 as the fragment size standard deviation – from each genome at 20× and 40× sequencing depth, using the software ART (15) (v2.5.8). For the AMR gene detection test, we used 1434 AMR genes available in the Comprehensive Antibiotic Resistance Database (17) (CARD, v2.0.2) (Supplementary Data). For the VF gene detection test, we used 1443 genes from the Virulence Factor Database (18) (VFDB, release date 22 March 2019) (Supplementary Data). In both tests, we first defined presence/absence of each gene in each genome using BLASTn (19) (v2.2.28+) as a ground-truth for assessing STing's performance. To perform a fair comparison with STing's gene detection utility, which is based on exact pattern matching, we defined a cutoff of 100% for identity and query (gene) coverage in BLASTn to consider a gene as present in a genome, i.e. if the gene is perfectly contained in the genome. Then, we built databases with STing for each gene set of interest (CARD and VFDB) and executed the respective gene detection analysis on each genome-derived read set at each sequencing depth, using a threshold of 100% for gene coverage to consider a gene as present in a sample. Finally, we evaluated the performance of detection in terms of sensitivity, specificity, precision and accuracy, which are defined as follows:

$$\text{Sensitivity} = \frac{TP}{TP+FN};\ \text{Specificity} = \frac{TN}{TN+FP}$$
$$\text{Precision} = \frac{TP}{TP+FP};\ \text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN};$$

where TP = true positives, TN = true negatives, FP = false positives and FN = false negatives.

RESULTS

STing uses exact *k*-mer matching for ultrafast classification

The STing algorithm breaks down (*k*-merizes) NGS reads into *k*-mers and then compares read *k*-mers against an indexed reference sequence database (Figure 1). Prior to any read processing, STing indexes the reference sequence database using the enhanced suffix array (ESA) (7) data structure; this enables the efficient representation of entire sequences, as opposed to other *k*-mer based methods that employ *k*-merized sequences stored in hash tables. The ESA data structure allows for a single sequence index, independent of *k*-mer size, whereas alternative approaches such as hash tables necessitate independent databases for each *k*-mer size. Moreover, the ESA data structure facilitates rapid exact *k*-mer matching between input reads and the indexed database.

The *k*-mer search strategy is based on exact pattern matching and comprises a read filtering step. For each read,

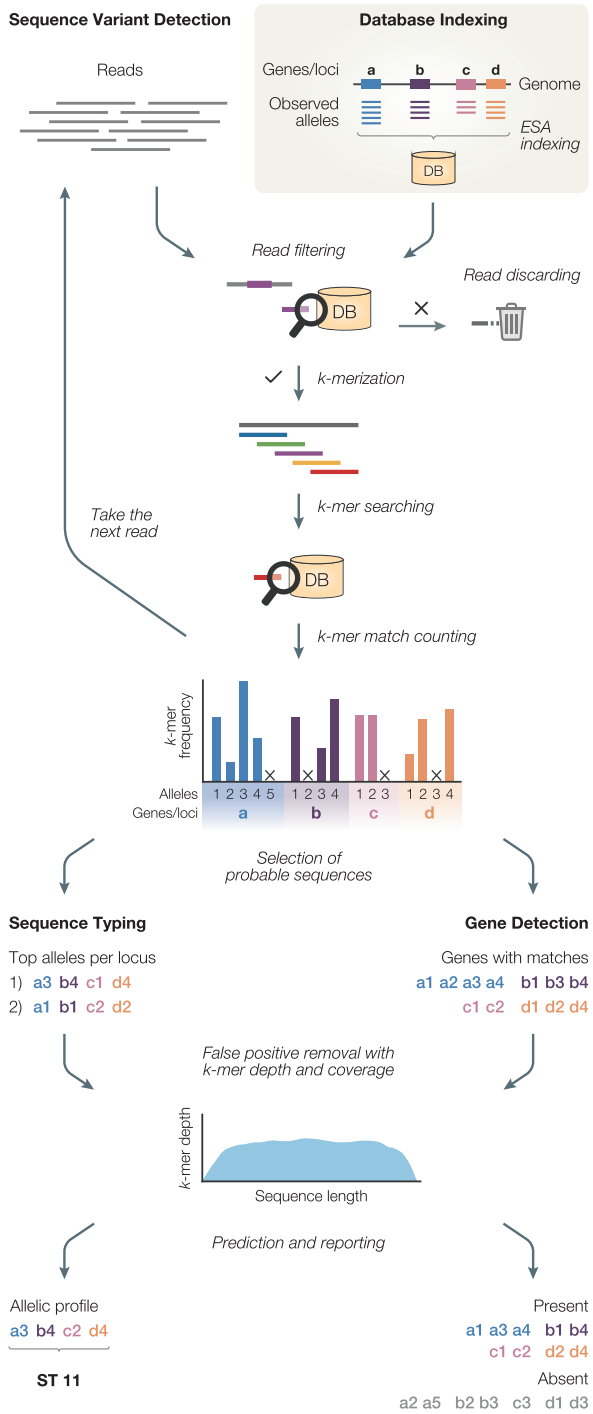

Figure 1. Schematic representation of the STing algorithm. The STing algorithm is comprised of two main phases: 1) Database indexing (shaded box) – user supplied reference sequences (allele or gene sequences) are transformed into an enhanced suffix array index for rapid *k*-mer searching during the sequence variant detection phase and 2) Sequence variant detection – reads are *k*-merized and each *k*-mer is searched for within the database. For each *k*-mer match in the database, a table of frequencies is maintained and updated for the matched sequence. These frequencies are then utilized to select candidate alleles/genes present in the samples analyzed. False positive alleles/genes are filtered out by calculating and analyzing *k*-mer depth and sequence length coverage from the selected candidate sequences. Lastly, predictions of allelic profile and ST and presence/absence of genes, are made and reported. A more detailed flowchart of the algorithm can be seen in Supplementary Figure S1.

a single central *k*-mer is initially compared against the sequence database. Reads are only fully *k*-merized if there is a match between the central *k*-mer and the database. If there is no match, the read is discarded. For schemes containing a small set of target loci, the vast majority of reads are discarded at this step. This filtering step results in substantial savings in terms of both the number of reads that need to be *k*-merized and the number of database search steps. Then, each *k*-mer from reads that pass the filter is searched within the indexed database. Upon a match, the algorithm updates the table of *k*-mer frequencies. Once all the reads from the input dataset are processed, the table of *k*-mer frequencies is used to perform sequence typing or gene detection.

In sequence typing mode, the algorithm calls the most probable allele sequences for each locus present in the sample. For each locus in the sequence typing scheme, STing selects the best alleles by taking the sequences with the highest *k*-mer frequency. STing forms the allelic profile with the alleles called for each locus. Finally, the sequence type (ST) of the sample is assigned by looking up the allelic profile in a profile table stored in the reference database. This table associates previously characterized allelic profiles with distinct STs. Sequence typing can be run in fast and sensitive modes. In fast mode, STing calls alleles based solely on the highest *k*-mer frequencies. In sensitive mode, STing calls alleles and removes false positives using *k*-mer depth and coverage information calculated for the top N ($N = 2$ by default) most probable sequence selected for each locus of the typing scheme. *k*-mer depth is defined as the number of *k*-mer matches that cover each nucleotide of the called allele, and coverage is defined as the percentage of the allele sequence that is covered by *k*-mer matches.

In gene detection mode, STing selects the most probable sequences to be present in the sample by taking the sequences with at least one *k*-mer match. Then, the algorithm calculates *k*-mer depth and sequence coverage information for each of the selected sequences. Finally, the algorithm defines a gene as present if its coverage is greater than or equal to a threshold defined for this purpose (75% by default); this threshold can be modified by the user.

STing outperforms other state of the art sequence typing programs

We compared STing to six of the most widely used programs for genome-enabled molecular typing, including its predecessor stringMLST. Details on the algorithmic approaches and data structure used by each of these programs are provided in Supplementary Table S1 (2). Our criteria for selecting the programs included the ability to perform sequence typing analysis on whole genome sequencing (WGS) data, using either Illumina raw reads or genome assemblies. Additionally, we preferred the most recent version of programs that are widely used and that implement different algorithmic paradigms for sequence typing. We selected six programs that fall into five different algorithmic paradigms: (i) stringMLST and (ii) MentaLiST which implement the *k*-mer paradigm based on *k*-mer match frequencies (4,8); (iii) Kestrel that uses a hybrid paradigm of *k*-mer frequencies plus a dynamic programming-based local alignment (9); (iv) SRST2 that utilizes read mapping to reference se-

quences to avoid full genome assembly (10); (v) ARIBA, a pipeline that uses read mapping to clusters of closely related alleles followed by a local assembly of reads mapped to each cluster (11) and (vi) Offline CGE, an offline implementation (https://doi.org/10.5281/zenodo.3604226) of the first alignment-based sequence typing method developed by the Center for Genomic Epidemiology (CGE) that requires full genome assembly (12).

The programs were evaluated for accuracy in terms of the percentage of correct allele predictions, speed in terms of average run time, and efficiency in terms of average maximum RAM consumption. STing was run in the fast and sensitive modes for the traditional housekeeping MLST scheme and two larger-scale typing schemes, rMLST and cgMLST. Allele databases for all three typing schemes were accessed from the PubMLST database (https://pubmlst.org/). STing's fast mode uses a *k*-mer matching only strategy, whereas the sensitive mode includes an additional step to exclude false positive calls based on gaps in *k*-mer coverage. Comparisons were performed for 10 samples each across four species that are widely used in MLST and accordingly have diverse MLST databases: *C. jejuni*, *C. trachomatis*, *N. meningitidis* and *S. pneumoniae*. STing shows 100% accuracy, in both the fast and sensitive modes, as well as the fastest run time and lowest memory use of any program for MLST (Figure 2A). The results of the same comparisons are broken down for each of the four individual species in Supplementary Figure S2. STing shows the highest accuracy, speed, and efficiency for the four programs that are capable of genome-enabled rMLST typing (Figure 2B). Programs that show an 'X' in these comparisons were unable to run for a variety of reasons related to their initial design, the runtime, and database indexing limitations. The program MentaLiST shows marginally higher accuracy, run time, and efficiency for cgMLST compared to STing (Figure 2C). However, the utility of MentaLiST, which was designed specifically for cgMLST, is limited by the size of the database that can be indexed. For that reason, it could not be run on the latest rMLST database available from PubMLST. The rMLST and cgMLST allele prediction errors seen for STing occurred for one of two reasons: (i) the correct allele was marginally lower in allele coverage compared to the predicted allele, or (ii) the allele in the isolate was a novel allele not present in the allele database used by STing. The coverage error gets resolved when STing is run in the sensitive mode (see Figure 2B); however, the error based on novel alleles persists.

Real-time molecular epidemiology in the post-genomic era with STing

We ran STing for MLST across a range of sequencing depth levels in an effort to assess its applicability on real-time molecular epidemiology by measuring its detection limits and multi-core performance (Figure 3). We simulated 2131 samples at six sequencing depths from genome assemblies of four species (*C. trachomatis*, *C. jejuni, N. meningitidis* and *S. pneumoniae*) retrieved from GenBank. The detection limit test shows that STing can accurately predict the sequence alleles with genomic depths as low as 20× and shows a marginal drop-off in accuracy at lower depth of 10× (Figure 3A). The drop-off in the ST calls is larger than allele calls as a single incorrect allele prediction can lead to incorrect ST call. While STing is designed as a single core application, executing STing in multiple parallel threads leads to a linear decrease in the processing time, requiring <6 min to process all of the 20× and 40× simulated genomes as opposed to over 50 min of processing time when run serially (Figure 3B). This provides a straightforward way to run STing on numerous genome samples as required in scenarios for real-time molecular epidemiology.

Additionally, we evaluated the accuracy and speed of STing for MLST analysis on a larger dataset of 1000 *N. meningitidis* samples obtained from the PubMLST/EBI ENA database. Detailed results of this large-scale test are shown in Supplementary Table S3. When this large-scale analysis was performed, STing uncovered samples that were initially scored as erroneous predictions but turned out to be misannotated in the PubMLST database (Supplementary Table S4). Seven samples that were initially incorrectly predicted were assembled, and the resulting assemblies were aligned to the *N. meningitidis* MLST reference database using BLASTn to determine the corresponding STs. All seven predicted STs were the same as what was predicted by STing, confirming that these samples are misannotated in the PubMLST database. Thus, STing had 100% accuracy predicting the STs for the 1000 samples analyzed.

Antimicrobial resistance and virulence profiling

In addition to molecular sequence typing, STing can also be used for automated gene detection directly from NGS reads. The gene detection mode uses a database of genes of interest indexed as an ESA for rapid *k*-mer searching. We assessed the ability of STing to detect genes by predicting the presence/absence of two types of epidemiologically relevant markers, antimicrobial resistance (AMR) and virulence factor (VF) genes, on simulated read samples.

To generate two indexed databases of genes of interest, we used 1434 AMR genes from the Comprehensive Antibiotic Resistance Database (17) (CARD, https://card.mcmaster.ca/), and 1443 VF genes from the Virulence Factor Database (18) (VFDB, http://www.mgc.ac.cn/VFs/), which were identified as present through BLASTn in 71 assemblies of bacterial pathogens relevant to public health. The assemblies were retrieved from GenBank and correspond to 25 bacterial species from the World Health Organization global priority list of antibiotic-resistant bacteria (16). STing was used to query the AMR and VF databases in NGS datasets simulated from the 71 assemblies (20× and 40× sequencing depth).

STing shows very high accuracy metrics for both AMR and VF detection, along with fast and efficient performance (Figure 4). In the case of AMR detection, STing shows a value of 1.0 for the sensitivity, specificity, and accuracy metrics in both 20× and 40× simulated datasets, with a low number of false positives that slightly impacts the precision metric in the two datasets (Figure 4A). In the case of VF detection, STing shows a value of 1.0 for the specificity and accuracy metrics in both 20× and 40× simulated datasets, and a minimal drop in sensitivity for the 20× dataset and in precision for both datasets due to false positives (Figure

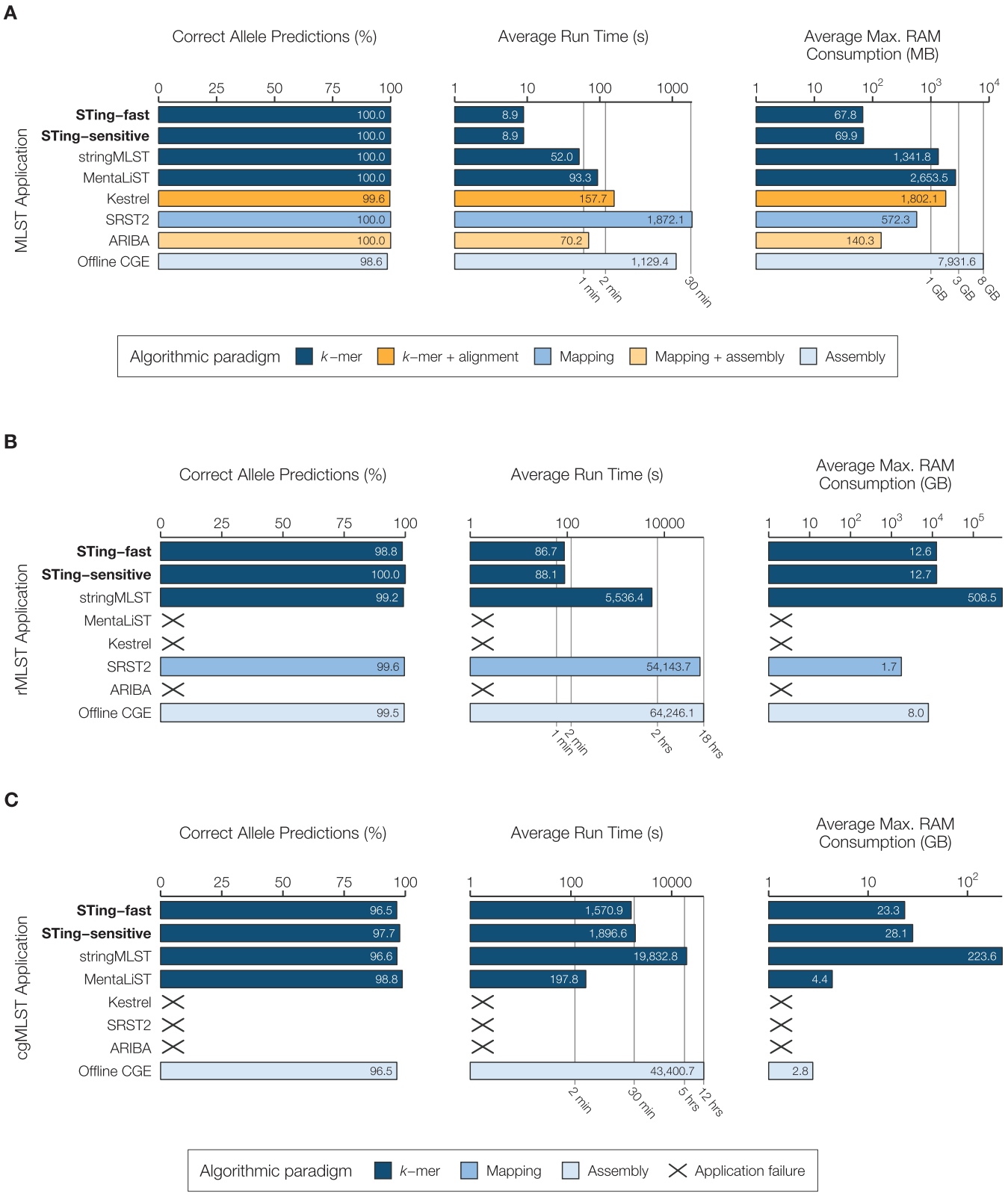

Figure 2. Performance comparison of STing with six other sequence typing applications. The fast and sensitive modes of STing are compared to six other contemporary typing applications to measure the accuracy and runtime performance, using three different typing schemes: (**A**) the traditional MLST (loci = 7) on 40 samples from four bacterial species (10 samples per species: *C. jejuni*, *C. trachomatis*, *N. meningitidis* and *S. pneumoniae*); (**B**) the ribosomal MLST (rMLST) scheme (loci = 53) on 20 samples of *N. meningitidis*, and (**C**) the core genome MLST (cgMLST) scheme (loci = 1605) on 20 samples of *N. meningitidis*. The typing applications are color coded based on the algorithmic paradigms that they utilize for performing sequence typing. Performance is measured in terms of the percentage of correct alleles predicted, the average runtime across each dataset measured in seconds (displayed in log-scale), and average peak RAM utilization across each dataset measured in megabytes (MB) for MLST and gigabytes (GB) for rMLST and cgMLST (both displayed in log-scale).

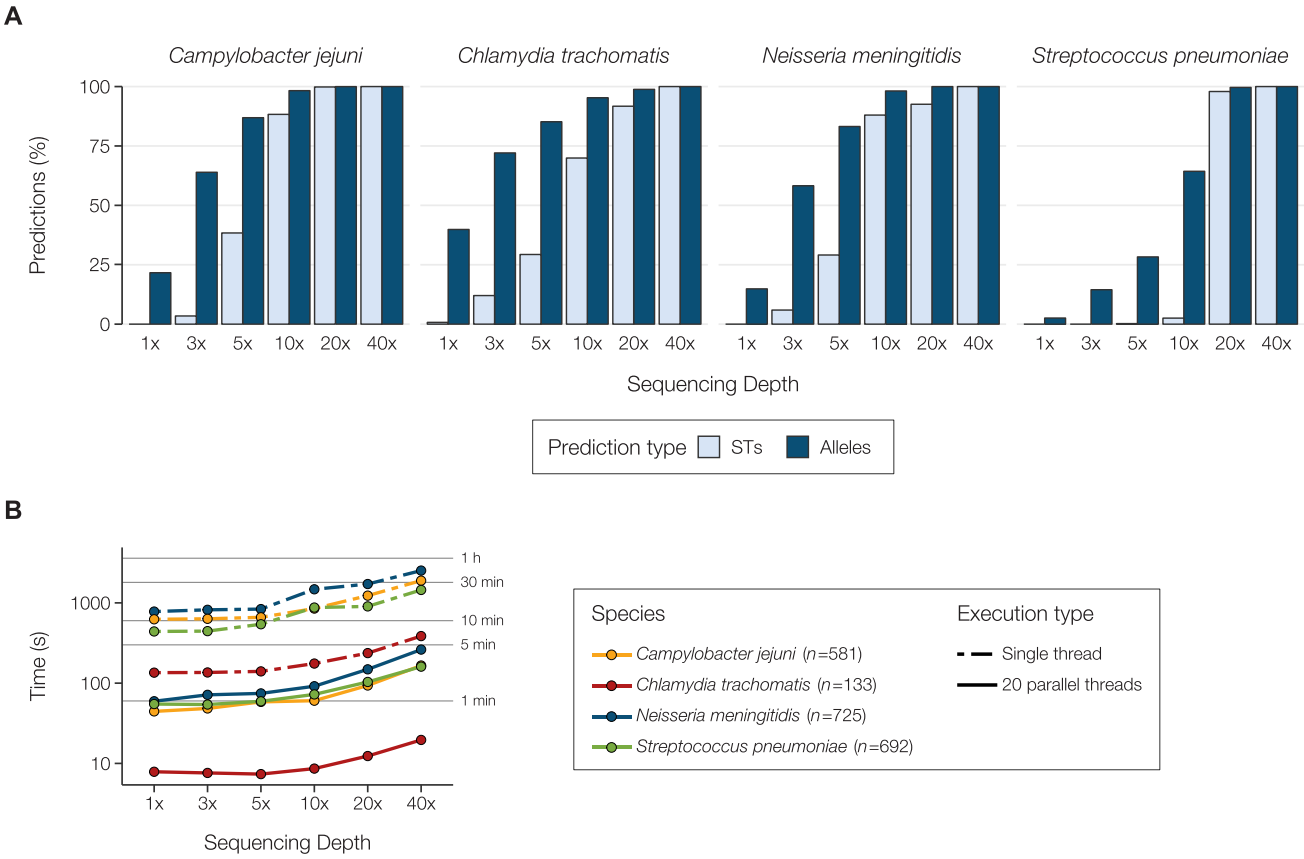

Figure 3. Results of the limit of detection test, and single- and multi-core performance test using MLST scheme. STing's sequence typing utility was run for MLST scheme in the fast mode over 1872 read samples simulated at 6 sequencing depths (1×, 3×, 5×, 10×, 20× and 40×) from assemblies of 4 species: *Chlamydia trachomatis* ($n = 133$), *Campylobacter jejuni* ($n = 581$), *Neisseria meningitidis* ($n = 725$), and *Streptococcus pneumoniae* ($n = 433$). (**A**) Percentage of correct predictions in terms of STs and alleles at different sequencing depths for the four datasets. (**B**) Total time in seconds (displayed in log-scale) required to process the complete dataset for each species at different sequencing depths using a single thread or instance (dashed lines) and 20 multiple threads (solid lines) of the sequence typing utility.

4B). STing required between 3.7 and 12.7 s on average, and between 71.8 and 134MB of RAM on average to process each sample of the AMR and VF simulated datasets (Figure 4C).

DISCUSSION

STing is the only program based exclusively on *k*-mer frequencies that provides for two critical analyses in molecular epidemiology: sequence typing and gene detection.

We compared STing to a set of six contemporary applications that use different algorithmic strategies for sequence typing. STing outperformed the other applications in accuracy and efficiency for the classic MLST and the larger rMLST schemes. Our software was very competitive in cgMLST analysis following closely to MentaLiST. Although MentaLiST performed better in cgMLST, a large-scale scheme for which it was specifically designed, we found that this application is limited by the size of the reference database that can be processed. The cgMLST approach is in its infancy and the size of the species-specific databases are small today. However, as more genomes are sequenced and characterized over time, the databases will continue to grow, and MentaLiST will present the same limitation as it currently does with rMLST. Regarding sequence typing, STing was the only application able to perform the analysis using all three of the schemes assessed, while also showing the ability to scale successfully to large genome-enabled typing schemes like cgMLST.

We also showed the ability of STing to be used in real-time molecular epidemiology. STing can provide accurate allele calls at sequencing depths as low as 10× and scales to efficiently analyze large-scale WGS datasets. Moreover, STing is easily parallelizable using multiple instances to reduce the time required to analyze hundreds of genome samples.

In addition to sequence typing, we assessed the ability of STing for gene detection using AMR and VF genes, two types of markers of high relevance in public health. STing was highly accurate and efficient for detecting both AMR and VF genes. More importantly, STing can be run in the gene detection mode to rapidly detect any genes of interest, which extends its utility beyond public health genomics. This could be particularly useful for large scale environmental genomics samples, including amplicon-based and metagenome studies.

STing was developed to provide turn-key solutions for NGS analysis in support of public health. Despite its

A Antimicrobial Resistance

STing	Known: Present, 20x	Known: Absent, 20x	Known: Present, 40x	Known: Absent, 40x
Present	132 TP	3 FP	132 TP	4 FP
Absent	0 FN	101,679 TN	0 FN	101,678 TN

B Virulence Factor

STing	Known: Present, 20x	Known: Absent, 20x	Known: Present, 40x	Known: Absent, 40x
Present	1,166 TP	10 FP	1,170 TP	10 FP
Absent	4 FN	101,273 TN	0 FN	101,273 TN

	A: 20x	A: 40x	B: 20x	B: 40x
Sensitivity:	1.000	1.000	0.996	1.000
Specificity:	1.000	1.000	1.000	1.000
Precision:	0.978	0.971	0.991	0.992
Accuracy:	1.000	1.000	1.000	1.000

C

Test[a]	Seq. Depth	Avg. Time[b] (s)	Avg. RAM[c] (MB)	Total Time[d] (s)
Antimicrobial Resistance	20x	3.7	71.8	200.1
	40x	8.0	72.4	400.4
Virulence Factor	20x	3.9	79.0	460.3
	40x	12.7	134.0	900.4

[a] Type of gene detection test executed
[b] Average time in seconds required to process each genome read sample
[c] Average maximum RAM required to process each genome read sample
[d] Total time in seconds required to process the entire read dataset

Figure 4. Performance comparison of STing's Gene Detection program. STing's gene detection program was run on 71 WHO-designated high-priority bacterial genomes (simulated at read depths of 20× and 40×) using two databases that contained gene annotations for 1434 antimicrobial resistance (AMRs) and 1443 virulence factors (VFs). Confusion matrices for the detection of (**A**) AMR genes from the CARD dataset, and (**B**) VF genes from VFDB dataset are shown. (**C**) The table demonstrates the accuracy and average runtime performance comparison of STing's gene detection at each sequencing read depth.

Feature	STing	stringMLST	MentaLiST	Kestrel	SRST2	ARIBA	Offline CGE
Assembly- & alignment-free	✓	✓	✓				
Standalone	✓	✓	✓				
MLST support	✓	✓	✓	✓	✓	✓	✓
Larger schemes support	✓	✓	✓		✓		✓
Runtime ≤ 1 hour for larger schemes	✓	✓	✓				
Big DBs support (≥ 2^{15} alleles per locus)	✓	✓					✓
k-mer size independent DB	✓				–	–	–
Gene detection	✓				✓	✓	
Confidence information	✓		✓				
Automated DB download	✓	✓	✓		✓	✓	

✓ Feature included
– Not applicable

Algorithmic paradigm: *k*-mer; *k*-mer + alignment; Mapping; Mapping + assembly; Assembly

Figure 5. Feature comparison between STing and the six applications tested for sequence typing.

lightweight computational footprint, STing performs accurate and ultrafast molecular typing and gene detection. We summarize the features and utility of STing compared to related programs for genome-enabled typing in Figure 5. In addition to its superior accuracy and performance, STing is distinguished by its streamlined algorithmic design, its broad applicability across typing schemes, its ability to support large databases, and its broad use as an automated gene detection utility.

DATA AVAILABILITY

Whole genome sequencing samples used for sequence typing, assemblies used for the limit of detection and multicore performance test, and genomes used for gene detection, are listed with accession numbers in Supplementary Data.

The source code of STing is available at https://github.com/jordanlab/STing and the pre-built databases are available at https://github.com/jordanlab/STing_datasets/releases. The modified script implementing the Offline CGE MLST method is available at https://doi.org/10.5281/zenodo.3604226.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

FUNDING

IHRC-Georgia Tech Applied Bioinformatics Laboratory [RF383]. Funding for open access charge: IHRC-Georgia Tech Applied Bioinformatics Laboratory.
Conflict of interest statement. None declared.

REFERENCES

1. Maiden,M.C.J., van Rensburg,M.J.J., Bray,J.E., Earle,S.G., Ford,S.A., Jolley,K.A. and McCarthy,N.D. (2013) MLST revisited: the gene-by-gene approach to bacterial genomics. *Nat. Rev. Microbiol.*, **11**, 728–736.
2. Espitia-Navarro,H.F., Rishishwar,L., Mayer,L. W. and Jordan,I. K. (2019) In: Budowle,B.A.S.S. and Morse,S. (ed). *Microbial Forensics*. 3rd edn. Academic Press.
3. Maiden,M.C., Bygraves,J.A., Feil,E., Morelli,G., Russell,J.E., Urwin,R., Zhang,Q., Zhou,J., Zurth,K., Caugant,D.A. *et al.* (1998) Multilocus sequence typing: a portable approach to the identification of clones within populations of pathogenic microorganisms. *PNAS*, **95**, 3140–3145.
4. Gupta,A., Jordan,I.K. and Rishishwar,L. (2017) stringMLST: a fast k-mer based tool for multilocus sequence typing. *Bioinformatics*, **33**, 119–121.
5. Jolley,K.A., Bliss,C.M., Bennett,J.S., Bratcher,H.B., Brehony,C., Colles,F.M., Wimalarathna,H., Harrison,O.B., Sheppard,S.K., Cody,A.J. *et al.* (2012) Ribosomal multilocus sequence typing: universal characterization of bacteria from domain to strain. *Microbiology*, **158**, 1005–1015.
6. Reinert,K., Dadi,T.H., Ehrhardt,M., Hauswedell,H., Mehringer,S., Rahn,R., Kim,J., Pockrandt,C., Winkler,J., Siragusa,E. *et al.* (2017) The SeqAn C++ template library for efficient sequence analysis: a resource for programmers. *J. Biotechnol.*, **261**, 157–168.
7. Abouelhoda,M.I., Kurtz,S. and Ohlebusch,E. (2004) Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorith.*, **2**, 53–86.
8. Feijao,P., Yao,H.-T., Fornika,D., Gardy,J., Hsiao,W., Chauve,C. and Chindelevitch,L. (2018) MentaLiST - a fast MLST caller for large MLST schemes. *Microb. Genomics*, **4**, e000146.
9. Audano,P.A., Ravishankar,S. and Vannberg,F.O. (2018) Mapping-free variant calling using haplotype reconstruction from k-mer frequencies. *Bioinformatics*, **34**, 1659–1665.
10. Inouye,M., Dashnow,H., Raven,L.A., Schultz,M.B., Pope,B.J., Tomita,T., Zobel,J. and Holt,K.E. (2014) SRST2: Rapid genomic surveillance for public health and hospital microbiology labs. *Genome Med.*, **6**, 90.
11. Hunt,M., Mather,A.E. and Sanchez-Buso,L. (2017) ARIBA: rapid antimicrobial resistance genotyping directly from sequencing reads. *Microb. Genomics*, **3**, e000131.
12. Larsen,M.V., Cosentino,S., Rasmussen,S., Friis,C., Hasman,H., Marvig,R.L., Jelsbak,L., Sicheritz-Ponten,T., Ussery,D.W., Aarestrup,F.M. *et al.* (2012) Multilocus sequence typing of total-genome-sequenced bacteria. *J. Clin. Microbiol.*, **50**, 1355–1361.
13. Audano,P. and Vannberg,F. (2014) KAnalyze: a fast versatile pipelined k-mer toolkit. *Bioinformatics*, **30**, 2070–2072.
14. Bankevich,A., Nurk,S., Antipov,D., Gurevich,A.A., Dvorkin,M., Kulikov,A.S., Lesin,V.M., Nikolenko,S.I., Pham,S., Prjibelski,A.D. *et al.* (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.*, **19**, 455–477.
15. Huang,W., Li,L., Myers,J.R. and Marth,G.T. (2012) ART: a next-generation sequencing read simulator. *Bioinformatics*, **28**, 593–594.
16. Tacconelli,E., Carrara,E., Savoldi,A., Harbarth,S., Mendelson,M., Monnet,D.L., Pulcini,C., Kahlmeter,G., Kluytmans,J., Carmeli,Y. *et al.* (2018) Discovery, research, and development of new antibiotics: the WHO priority list of antibiotic-resistant bacteria and tuberculosis. *Lancet Infect. Dis.*, **18**, 318–327.
17. Jia,B., Raphenya,A.R., Alcock,B., Waglechner,N., Guo,P., Tsang,K.K., Lago,B.A., Dave,B.M., Pereira,S., Sharma,A.N. *et al.* (2017) CARD 2017: expansion and model-centric curation of the comprehensive antibiotic resistance database. *Nucleic Acids Res.*, **45**, D566–D573.
18. Liu,B., Zheng,D., Jin,Q., Chen,L. and Yang,J. (2019) VFDB 2019: a comparative pathogenomic platform with an interactive web interface. *Nucleic Acids Res.*, **47**, D687–D692.
19. Camacho,C., Coulouris,G., Avagyan,V., Ma,N., Papadopoulos,J., Bealer,K. and Madden,T.L. (2009) BLAST+: architecture and applications. *BMC Bioinformatics*, **10**, 421.