Fast and easy metagenomic data analysis with FastDeMe

Sander Vermeulen

ABSTRACT

The introduction of sequencers such as Illumina's MiSeq and HiSeq boosted our understanding of microorganisms significantly. Metagenomic sequencing of clinical samples is a promising new diagnostic method, but requires new computational tools for proper processing. FastDeMe is a fast, easy to learn, plug and play Linux command line tool for clinical metagenomic data analysis. It features trimming, removal of host sequencing reads, taxonomic classification and resistome profiling. Compared to MGMapper, FastDeMe finishes 22 to 29 times faster when identifying taxonomy while retaining accurate results. Furthermore, FastDeMe was able to identify 50 out of 50 antibiotic resistance genes in a synthetic metagenomic sample when coverage ≥ 10 with minimal false positives (2-8).

Contents

1. INTRODUCTION

1.1 Clinical microbiology before Next Generation Sequencing

Culturing of bacteria has been the staple of (clinical) microbiology ever since the very beginning. Traditional diagnostic methods, such a microscopy, Gram stains, antigen detection, antibiotic susceptibility testing and biochemical tests all rely on successful cultivation of pathogens. However, not all bacteria are easy to culture. For example, *Mycobacterium tuberculosis* has a long replication cycle of 20-24 hours [1]. Visible colonies only appear after about one week of cultivation, thus making accurate diagnosis take a while. Additionally, some pathogens such as *Treponema pallidum*, *Mycobacterium leprae* and *Rickettsia felis* need cell culture or even living animals to be isolated and/or to propagate. Although the following discussed techniques are better for routine diagnosis, culturing emerging pathogenic bacteria is still a necessary model for developing faster diagnosis methods [2].

To reduce diagnosis time and make diagnosis of hard to cultivate bacteria easier, molecular methods have been developed such as broad-range PCR and 16S rDNA sequencing. Because of the high sensitivity of PCR, pathogens can still be detected in samples with very low quantity, making it potentially ideal for difficult to cultivate bacteria. For dormant diseases like chronic Lyme arthritis molecular methods are able to detect the very low concentration of pathogens in bodily fluids, whereas culture probably wouldn't give any results. Determination after amplification with PCR is often done using the sequence of 16S rDNA, most commonly sequenced using Sanger sequencing. However, PCR is subject to DNA contamination, even when appropriate precautions are followed. Additionally, PCR is not ideal when the material contains multiple species of bacteria. [3] [2]

1.2 Next Generation Sequencing and modern clinical microbiology

With the advent of next generation sequencers (NGS) such as Illumina's MiSeq and HiSeq and Pacific Biosciences' Sequel, from several up to hundreds of genomes can be sequenced with a single run. Whereas Sanger sequencing requires specific primers targeting the 16S rDNA are needed of a monoculture, NGS does not suffer from this problem since whole genomes are sequenced. Most NGS experiments begin with library preparation, in which DNA is fragmented. This is done because most modern sequencers can only sequence fragments of 100 to 1000 bases. Exceptions are Oxford Nanopore's MinION and the aforementioned Sequel from Pacific Biosciences, which are capable of generation reads of >200 kb. However, these sequencers are at the moment not routinely used in clinical settings, because of the cost, lower read quality and low throughput. [3]

The introduction of NGS made the analysis of material containing multiple bacterial species easier, eliminating the need of using axenic bacterial cultures for accurate diagnosis. However, a great deal of data is generated using metagenomic sequencing and already established methods and computational tools for single genome sequencing analysis cannot be used for metagenomic data analysis.

Just like normal sequencing analysis, metagenomic analysis cannot be done in one easy step, but requires multiple steps to (pre-)process the data before any real results can be obtained. Common pre-processing steps include trimming and quality control and removal of host DNA. Automated trimming is essential since modern sequencers produce millions of reads and make errors, which will be removed using automated pre-processing steps. The next step depends on the type of dataset, but reads can either be mapped against nucleotide or protein or kmer-databases using various methods or assembled into contigs using specialized assemblers. The followup analysis steps depend on what the goal is, but a few examples are pathogen identification, antibiotic resistance gene recognition and SNP calling.

Metagenomics might seem daunting for biologists not at home in this relatively new field. Furthermore, the vast majority of software available for metagenomics research makes use of the Linux command line, and must be installed with instructions and dependencies that vary from

program to program. To make metagenomics research more accessible web-based applications such as MGMapper [4] from the DTU have been created. However, these tools are often very slow and have limited functionality and require the sequence reads to be uploaded, something not always allowed because sequencing human patient material will also result in contamination with host DNA, and sharing human genetic data is not allowed by the GDPR, furthermore the files are very large and uploading large quantities of data is not always possible in all settings.

Development of these tools is paramount in the fight against antibiotic resistance in bacteria. The final report of *The Review on Antimicrobial Resistance* (AMR), a series of papers commissioned in July 2014 by UK Prime Minister David Cameron, explores what should be done to avoid the predicted 10 million deaths/year due to untreatable infections should we continue with our current policies. One proposed measure is to make it mandatory to diagnose before prescribing antibiotics, limiting the use of unnecessary or ineffective antibiotics. Rapid diagnosis using NGS has the potential to give medical professionals the tools to achieve this, without compromising patient safety by withholding treatment for multiple days. [5]

Introducing FastDeMe, an easy and fast all-in-one pipeline for the classification of bacteria and antibiotic resistance genes in clinical metagenomic samples. The pipeline is command line based, but has been made to be as simple as possible to install. No installation is required, after downloading the required files, scripts and databases the pipeline is ready to be used on most Linux distributions.

Additionally, FastDeMe has been made with modularity in mind. Every implemented module (trimming and QC, detection and removal of host DNA, taxonomic classification and resistome profiling) can be turned on and off depending on the user's needs. Users with Python knowledge can easily add additional modules, such as assemblers, MLST and virulence detection.

The current features of the program include trimming and QC, detection and removal of host DNA, taxonomic classification and resistome profiling, which can be run independent from each other if desired.

2. IMPLEMENTATION

2.1 System requirements

FastDeMe uses ~30 GB RAM for a run and requires a recent Linux distribution. No other dependencies are required apart from those already available in nearly all Linux distributions

2.2 General design

FastDeMe is designed for fast and easy screening of metagenomic samples from humans or animals. It accepts FASTQ files as input (uncompressed or gzipped) and supports single- and paired end data. FastDeMe can be run multi-threaded, and trimming and QC will use up to 16 cores. Output depends on which options are chosen by the user. A schematic view of the steps is presented in Figure 1.

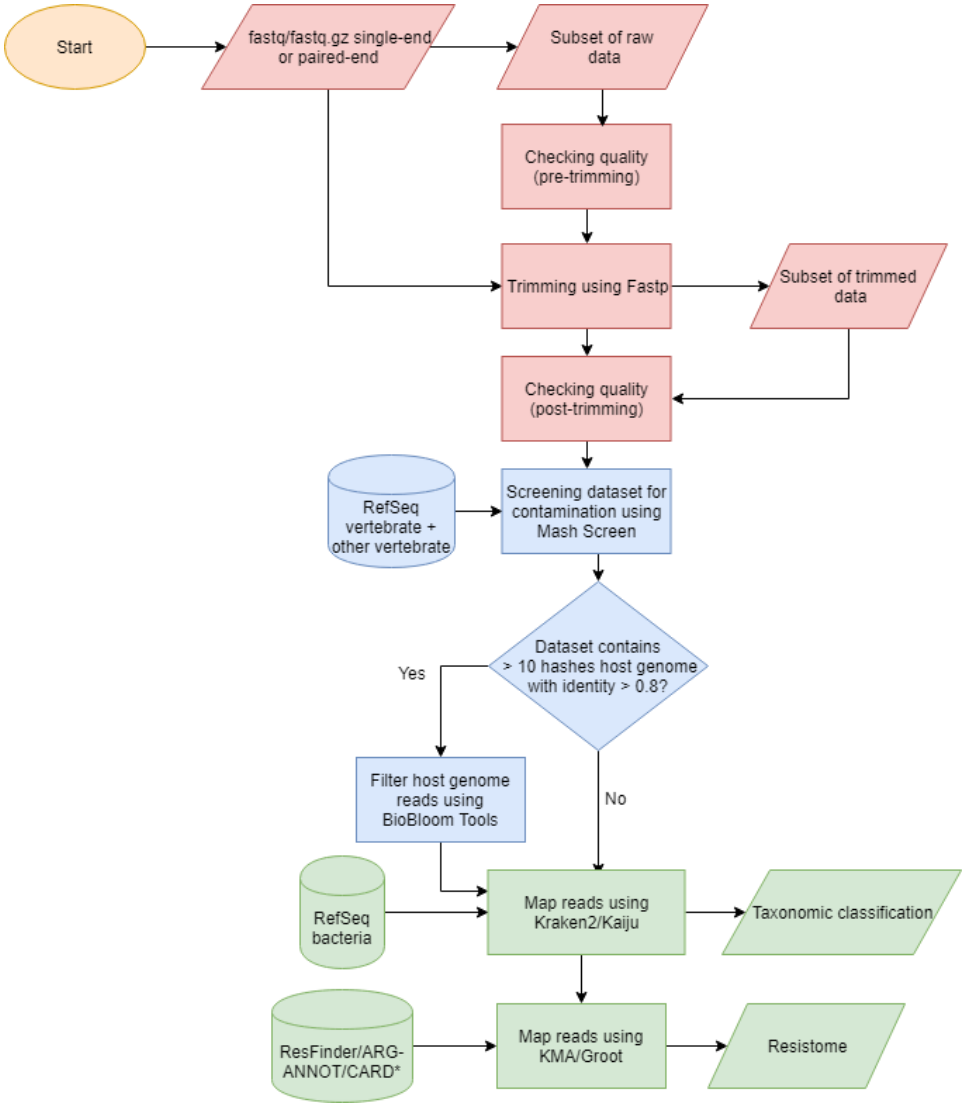

Figure 1: Schematic overview of FastDeMe's pipeline. Red, the preprocessing module, includes QC of raw data, trimming with Fastp and QC of post-trimming data. To reduce time spent on QC, the top 25000 reads of both pre- and post-trimming files are extracted and used for QC. Blue, the host filter module, includes screening with Mash Screen and filtering with BioBloom Tools. To reduce the time of screening, a 25000 read file is used that was made during trimming. Filtering will proceed when Mash Screen reports > 10 hashes with and identity of > 0.8. In case Mash Screen detects nothing, filtering will be skipped. Next, taxonomic classification with Kraken2 and/or Kaiju will be performed, followed by resistome profiling using Groot and/or KMA.

2.3 Trimming and QC

Trimming is done with fastp [7], the QC is calculated using Python. QC is performed pre- and post-trimming, so the results of the trimming step can be inspected by the user. To save time, the first 25000 reads of the input file(s) and trimmed file(s) are extracted, which will be used for the QC. During trimming, reads with less than 80% of the average read length of the input will be removed, as well as adapters and reads with a PHRED score less than 20.

2.4 Screening and host DNA removal

FastDeMe has the ability to quickly detect and remove host DNA from metagenomic samples in order to have as little contamination as possible further down the line which would severely affect quantitative assessment of e.g. the resistome. Screening the input files is done with Mash's [8] screen

option against the RefSeq (vertebrate and other vertebrate) database. Mash screen makes use of the MinHash technique, originally created to quickly estimate the resemblance between websites or, in this case, the input and the RefSeq database. While the MinHash technique is ideal for estimating similarity, it is of limited use for quantifying which genomes are there. When only using MinHash, the similarity between the input metagenome and database would be very low since the database contains a significant number of genomes that are not present in the metagenome, however this problem was mititgated in Mash v2 with the screen option,that specifically keeps a count of observed kmers. Since the Mash compatible made RefSeq database is only 100 MB, a hash table of the RefSeq database can be stored in memory. K-mers of the metagenome can be compared to the hash table of the RefSeq database, and matches can be stored. This way a quick estimation can be made of the composition of the metagenome. [9]

If more than 10 hashes match with a single genome in the database and the identity of the hashes is greater than 0.8, the genome will be filtered out using BioBloom tools' BioBloomCategorizer [10]. BioBloomCategorizer does a similar job as aligners such as BWA [11] and Bowtie2 [12], but makes use of bloom filters, a probabilistic data structure. Bloom filters are able to quickly determine whether a value is present in a set, but can only tell for certain if a value is definitely not in a set. Positive results have a chance to be false positives, but this can be reduced by increasing the size of the filter, by default the false positive rate is 0.0075. BioBloomCategorizer will also remove less host reads compared to BWA and Bowtie2., but the tremendous speed advantage was deemed more important for this specific pipeline.

Since the bloom filters for each genome are quite big, only a select group of the most common clinical hosts from the vertebrate and other vertebrate RefSeq databases are selected to be included in the standard database download. When the program encounters a bloom filter that is not included in the user's database, the bloom filter will be downloaded automatically after which filtering will resume, unlike existing tools, which only support removal of human genetic information.

2.5 Taxonomic classification

Taxonomic classification is performed using Kaiju [13], Kraken2 [14] and Bracken [15]. From these three programs, Kaiju and Kraken2 are taxonomic classifiers, while Bracken is a standalone abundance estimator for Kraken2. Abundance estimation is already included in Kaiju.

The reason users can choose between two taxonomic classifiers is because both have a different approach, and depending on the dataset either Kaiju or Kraken2 might work better. Kaiju classifies reads using a database, in this case the RefSeq bacterial and viral database, filled with annotated protein-coding genes of microbial genomes using the Burrows-Wheeler Transform. Since Kaiju uses a protein database, the reads have to be translated to amino acids in all six reading frames. The translated reads are split at stop codons and sorted by length. The longest fragment will be queried against the database. The longest maximum exact match (MEM) is retained. Once finished, the taxon ID will be retrieved from the database sequence with the longest MEM against the fragment. In case multiple taxa have the exact MEM length, Kaiju finds the lowest common ancestor (LCA). [13]

Contrary to Kaiju, Kraken2 uses the DNA sequence from the reads directly. Kraken2's RefSeq bacteria database consist of k-mers and the LCA of all bacteria who possess that k-mer. Reads are queried against the database, and k-mers are assigned to their nodes in the taxonomic tree, creating a "path" to the read origin organism. When a read doesn't match with a k-mer in the database, it is left unclassified. See figure 2 for further clarification. [14]

Kraken2 has trouble correctly predicting on species-level for well-populated clades with low diversity between species. Instead of guessing species, the LCA will be reported. An accurate indication of total abundance can only be made when all reads stranded above species level are taken into account. So, Kraken2's raw output cannot be used for abundance estimation, especially on species or strain

level. Bracken was developed to estimate abundance from Kraken2's output using a Bayesian probability method. [15]

Kaiju suffers from the same problem as Kraken2, but an abundance estimator is already included in the main program.

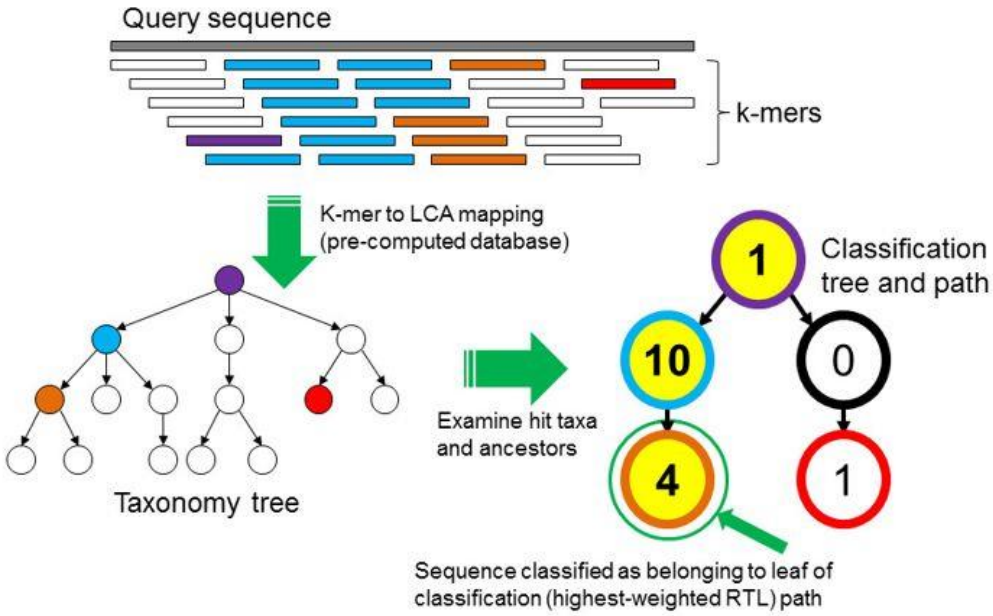

Figure 2: Kraken's classification algorithm. Each k-mer in a query sequence is mapped to the lowest common ancestor (LCA) of the genomes containing that k-mer in the database. Taxa associated with a query sequence's k-mers form a subtree in the general taxanomy tree, which will be used for classification. In this subtree, every node has a weight depending on how many k-mers mapped for the node's taxon. The path which has the highest number of mapped k-mers is the classification path, of which the latest node will be used for classification of the query sequence. [14]

Choosing the best classifier for the dataset depends on the needs of the user and the dataset itself. Kraken2 is faster, but works best for samples in which the majority of species are present in the database. Kaiju works best for environmental samples, were the majority of species are unknown. Since protein sequences are more conserved than DNA sequences, Kaiju might be able to classify these unknown species in an higher taxonomic rank, instead of being left unclassified with Kraken2. Additionally, Kaiju is more tolerant to sequencing errors since the resulting protein might be the same. [14]

2.6 Resistome profiling

For the purpose of this pipeline, Groot [16] and KMA [17] where incorporated. Groot is fast, accurate and most importantly, a resistome profiler. Most tools of this nature aren't designed for resistome profiling, e.g. determining antibiotic resistance genes in metagenomic samples, but rather for single genomes. The main drawback of most resistome profiling tools is the given that different antibiotic resistance genes can be very similar, thus creating a database with reference sequences that are almost identical. This can result in ambiguous alignments, unaligned reads or mis-annotated reads. One proposed solution was to cluster similar reference sequences, but this lead to a loss of information since gene subtypes would be clustered together [18]. Another related solution is to simply collapse

annotations of gene subtypes into one annotation [19]. However, the ability to detect gene subtypes is important since phenotypic activity depends on subtype for some genes [20]

Groot uses variation graphs, a non-linear data structure, to encode reference sequences. This data structure has already been successfully implemented in tools for variant calling [21] [22]. Variation graphs reduce redundant sequences while maintaining relevant information for classification. The graph is indexed using MinHash, and reads are aligned using nearest-neighbour search and hierarchical local alignment.

Groot's database is a combination of the ResFinder [23], ARG-ANNOT [24] and CARD [25] databases. Duplicate sequences have been removed, and the remaining sequences were clustered at 90% identity.

KMA is, like Groot, a tool specifically developed for the detection of antibiotic resistance genes. KMA uses k-mer based alignment and a novel scoring scheme dubbed ConClave. First off, a fast heuristic mapping is performed to assess if a query doesn't look like anything in the database or if a query might look like something in the database. Since antibiotic resistance genes often represent a minimal part of an entire metagenome, quickly filtering everything that definitely doesn't match the database speeds up alignment significantly. KMA's database is a hash map of indexed k-mers, which will be aligned against queries passing the heuristic mapping. The ConClave scoring was added since KMA is designed to support redundant databases. The ConClave scoring system allows queries to align to several k-mers, out of which the best match will be chosen. [17]

The KMA database consist of the ResFinder [23] database.

2.7 Software versions

For the initial release of FastDeMe, the following versions of the incorporated third-party software are used:

Fastp: 0.19.6, Mash: 2.1, BioBloom Tools: 2.1.2-5-g8a47, Kaiju: 1.6.3, Kraken2: 2.0.7, Bracken: 2.2, Groot: 0.7 and KMA: 1.1.7.

3. METHODS

All the following verifications and benchmarks were performed with paired-end files, with read lengths of 150 bp.

3.1 Verifying taxonomic classification

To verify the accuracy of FastDeMe, the Mock Bacteria Archaea Community (MBARC-26) was used [6]. This community contains 23 bacterial and 3 archaeal species, isolated from soil, aquatic, human, bovine and frog samples. Since archaea are not included in the Kaiju or Kraken2 databases, these were excluded in the results. Three different subsets were made by randomly extracting 10 million reads from the MBARC-26 paired-end files. Next, the three subsets were analyzed using MGMapper and FastDeMe . For FastDeMe, the trimming, screening, Kraken2 and Kaiju modules were turned on in paired end mode. MGMapper's settings can be found in Table 1 and Table 2 (Supplementary data), which were set to as closely resemble FastDeMe's settings as possible.

Coefficient of variation was calculated by determining the absolute difference between reference abundance and found abundance, and dividing this number with the reference abundance. This was done for all 23 bacteria. The average of these numbers was calculated, and this average number multiplied by 100 to get a percentage.

3.2 Verifying resistome profiling

To verify the results of Groot and KMA, the MBARC-26 dataset was combined with 25 randomly chosen pairs of very similar resistance genes from the ResFinder database resulting in 50 genes. Using ArtificalFastqGenerator [26], random 150 bp paired-end reads were made, with a mean coverage of 1, 10, 20, 30, 40 and 50. To ensure the whole resistance gene is included evenly in the reads, 1000 random nucleotides were added to the front and back of each gene. The MBARC-26 files and the read files from the genes were combined using Unix's cat tool. These files can be found in the supplementary data. Next, the files were analyzed using FastDeMe's Groot and KMA modules. For the verification of Groot, we used the standard 90% clustered ResFinder database constructed with the default hash size (128) and k-mer size (7). Groot report was set with the -c option on 0.2. KMA was constructed with the ResFinder database and a k-mer size of 16 (default).

For the purposes of this benchmark, we considered all detected antibiotic resistance genes, except the manually added genes, as a false positive (FP) hit. If a program failed to find one or multiple of the manually added genes, we considered this a false negative (FN). Genes that were correctly detected were counted as true positives

3.3 Benchmarking runtime

Runtime was only compared between MGMapper and the taxonomic classification module of FastDeMe. The online KMA tool from DTU does not report start and end time, so accurate runtimes could not be obtained. However, since FastDeMe also uses KMA for resistome profiling, runtimes are expected to be very similar.

For benchmarking runtime between MGMapper was configured according to Table 1 and Table 2 (Supplementary data) and FastDeMe was ran with the following settings: --threads 4, --kaiju, --trimming, --screening. Since MGMapper's dedicated server uses 4 threads to run the pipeline, FastDeMe was also set to use 4 threads to allow for a fair comparison. Additionally, the screening/host filtering module was turned on, which MGMapper doesn't support in the pipeline. The three generated MBARC-26 subsets and a real metagenomic sample from chickens were used for benchmarking runtime.
Walltime of FastDeMe was obtained using Bash's time command, walltime of MGMapper was taken from the output webpage.

4. RESULTS

4.1 Community profiling

To analyse the performance of FastDeMe's taxonomic classification module using Kaiju [13] and Kraken2 [14], we used the MBARC-26 mock community [6] and compared it to the results of a similar tool, MGMapper [4]. The community profiling results are shown in Figure 3.

Each bar in Figure 3 represents abundance in percent of each species found in the dataset. This is the average abundance from the three subsets, abundance for each species and subset individually can be found in Table 3 (Supplementary data). Kraken2 was the only method that could successfully detect all bacterial species in the sample. *Staphylococcus pyogenes* could not be detected by Kaiju and MGMapper, and additionally *Escherichia coli* and *Salmonella bongori* were not detected by MGMapper.

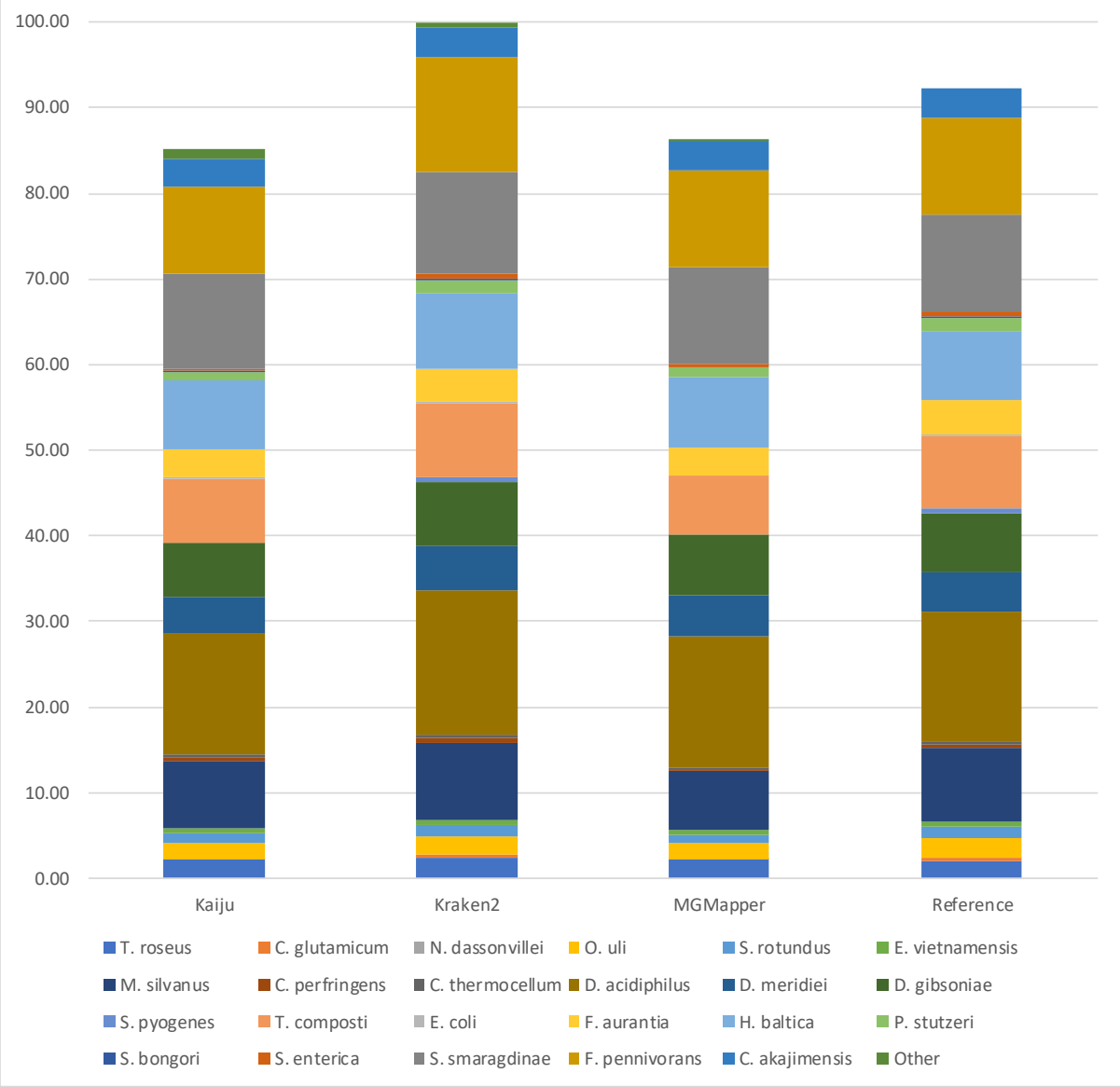

Figure 3: Benchmarking of Kraken2, Kaiju and MGMapper with the MBARC-26 mock community. Each column represents bacterial abundance in percent found with every method and the reference abundance, each color equals one bacterial species.

MGMapper detected the least false positives, with a total abundance of 0.01-0.02%. Kaiju detected the most false positives at 1.13%.

Kraken2's had the lowest coefficient of variation (CV), meaning Kraken2's results were closest to the real abundance percentages. Pearson correlation of all methods was high ($R^2 > 0.98$), meaning every method was quite effective at correctly measuring relative abundance.

Since we took 3 subsets of 10 million reads from the main MBARC-26 files, we wanted to calculate the CV between the subsets to verify that the variation is not dependent on the sequences in the tested sets. The 10 million reads proved to be a large enough sample size, since the CV between the subsets was low, with 0.49%, 0.12% and 0.16% for Kaiju, Kraken2 and MGMapper respectively, suggesting that results from community profiling with all three methods are repeatable.

4.2 Resistome profiling

Benchmarking the performance of the resistome profiling module using Groot [16] and KMA [17] was done with a custom benchmarking dataset, made with 25 pairs of very similar genes extracted from the ResFinder database as described in the methods section

The results show some significant differences between Groot and KMA. Groot was only able to detect 21-22 antibiotic resistance genes out of 50, as shown in Figure 4. KMA was able to detect all resistance genes, apart from one instance (Figure 5).

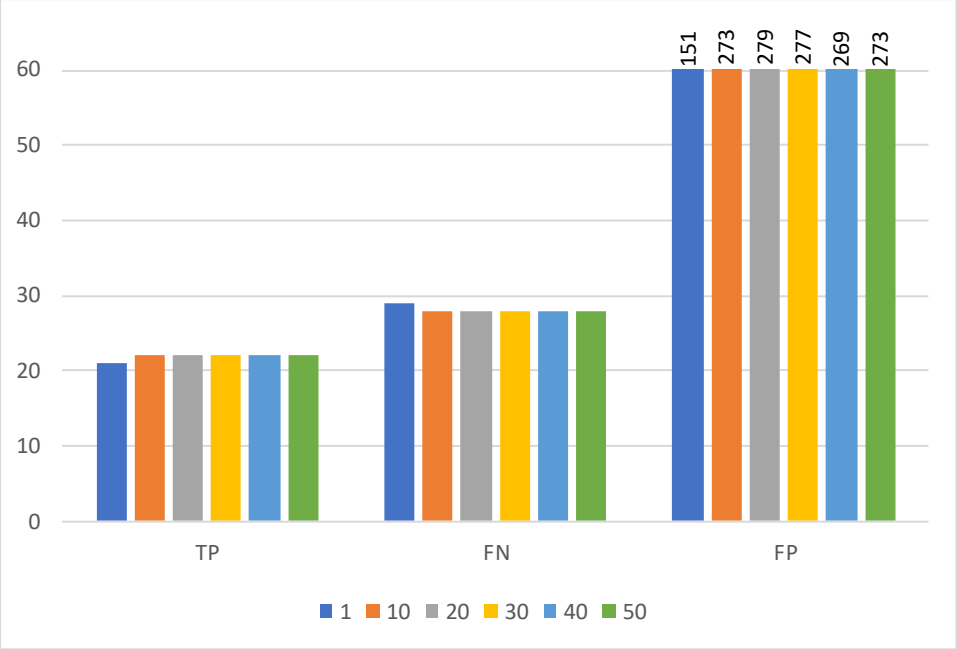

Figure 4: Benchmark results of Groot. Each individual bar represents a coverage value, and each cluster represent the number of true positives (TP), false negatives (FN) or false positives (FP) found in the benchmark dataset.

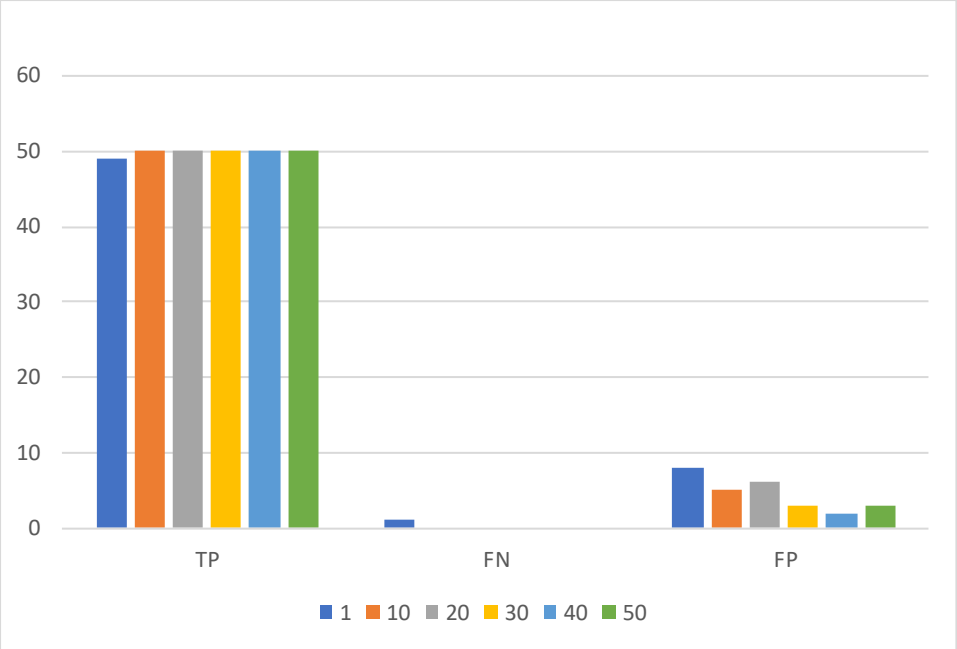

Figure 5: Benchmark results of KMA. Each individual bar represents a coverage value, and each cluster represent the number of true positives (TP), false negatives (FN) or false positives (FP) found in the benchmark dataset.

Figure 4 shows that Groot produces a large number of false positives (151-279), which is caused by the low setting of the -c parameter, which dictates the coverage cutoff for reporting antibiotic resistance genes. Adjusting the -c parameter did decrease the number of false positives at the cost of true positives (results not shown).

In contrast to Groot, KMA reports between 2-8 false positives (Figure 5). At higher coverages (30-50) the amount of false positives drop compared to the lower coverages (1-20).

Runtimes for MBARC-26 subset 1, 2 and 3 were 297, 299 and 316 minutes respectively using MGMapper and 11 minutes in all cases for FastDeMe. Runtimes using the chicken metagenomic sample was 1528 minutes for MGMapper and 67 minutes for FastDeMe.

5. DISCUSSION

FastDeMe was designed to bring users a fast and easy to learn command line tool for the analysis of clinical metagenomic samples. It features trimming using Fastp [7], automatic host species detection and removal of host reads using Mash [8] and BioBloomTools [10], taxonomic classification using Kaiju [13] and/or Kraken2 [14] + Bracken [15] and resistome analysis using Groot [16] and KMA [17] in a download-and-use package, without having to install anything as all dependencies are included in the software package. Additionally, special care was given to the modularity of the program. Everything can be ran independent from each other, saving the user time by not having to, for example, trim the data when the data is already trimmed by the user beforehand. Users with some Python programming skills can also add their own modules without too much hassle should they need extra analysis methods.

Both Kraken2 and Kaiju performed better than MGMapper in estimating the abundance of species in the MBARC-26 subsets. Average coefficient of variation (CV) values amounted to 21.72, 11.28 and 29.05 for Kaiju, Kraken2 and MGMapper respectively. MGMapper reported the least false positives, with an average abundance of only 0.02%. Kaiju reported 1.13% on average, and Kraken2 0.49%. Correlation of all methods to the reference was high, with all samples scoring $R^2 > 0.98$.

Groot didn't perform as well in our benchmark compared to results found by Rowe et al. [16]. Groot was only able to detect 22 out of 50 resistance genes at best. Additionally, Groot reported a large number of false positive genes, which is caused by the low setting of the -c option of Groot report. This setting is responsible for the coverage cutoff for reporting antibiotic resistance genes, which is set at 0.97 for default instead of 0.2. However, we found that leaving this setting on default did not only eliminate a large number of false positives, but also removed true positives. For our benchmark, anything over 0.2 removed true positives so we chose to show the setting which reported the most true positives.

It should be noted that our benchmark is a worst case scenario, since 25 pairs of very similar genes were selected. It is possible that Groot does not handle datasets like our benchmark very well, but performs as expected with datasets where only one representative of very similar alleles of resistance genes is present.

KMA detected all antibiotic resistance genes in the benchmark, apart from the sample with a coverage of 1 where one false negative was reported. The highest number of false positives found was 8, again with the coverage benchmark set with a coverage of 1. Due to the dramatically better performance of KMA compared to Groot, we strongly recommend users to use KMA until we found a possible explanation for the disappointing results of Groot.

For taxonomic classification, FastDeMe is 22 times faster than MGMapper when using a real metagenome and up to 29 times faster using the synthetic metagenome. Keep in mind that only the runtime of MGMapper was taken into account, as jobs do not always start instantly if the DTU servers are busy, therefore waiting times may be significantly longer. Moreover, if the computer running FastDeMe can run more than 4 threads the gap will widen further.

6. SUPPLEMENTARY DATA

Table 1: Settings used for MGMapper, as it can be found on the webtool

Minimal read length after trimming	120
Minimum PHRED quality	20
Cutadapt QUALITY_BASE	33
Database id's for best-mode mapping	2
Database id's for full-mode mapping	n/a
FMM value	0.8
MAS value	30
Abundance cutoff	0.01
Unique reads ratio	0.005
Max mismatch ratio	0.01
Min read count	10

Table 2: Sequences of adapters that were removed for MGMapper

Adapter sequences to be removed	AGATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATC TCGTATGCCGTCTTCTGCTTG AATGATACGGCGACCACCGAGATCACACTCTTTCCCTACACGAC GCTCTTCCGATCT AGATCGGAAGAGCACACGTCTGAACTCCAGTCAC AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTA

Table 3: Raw abundance results in percent for each subset and bacterial species.

		SUBSET 1			SUBSET 2			SUBSET 3				
Species	Reference	Pipeline (Kaiju)	Pipeline (Kraken2)	MGMapper	Pipeline (Kaiju)	Pipeline (Kraken2)	MGMapper	Pipeline (Kaiju)	Pipeline (Kraken2)	MGMapper	Reference	Species
T. roseus	2.07	2.16	2.41	2.16	2.17	2.42	2.16	2.16	2.41	2.15	2.07	*T. roseus*
C. glutamicum	0.3	0.07	0.34	0.05	0.08	0.34	0.05	0.08	0.34	0.05	0.3	*C. glutamicum*
N. dassonvillei	0	0	0	0	0	0	0	0	0	0	0	*N. dassonvillei*
O. uli	2.26	1.85	2.14	1.85	1.85	2.14	1.85	1.85	2.14	1.86	2.26	*O. uli*
S. rotundus	1.41	1.13	1.28	1.02	1.13	1.27	1.02	1.13	1.27	1.02	1.41	*S. rotundus*
E. vietnamensis	0.62	0.54	0.67	0.65	0.55	0.67	0.64	0.54	0.67	0.64	0.62	*E. vietnamensis*
M. silvanus	8.56	7.96	9.02	6.88	7.97	9.03	6.89	7.97	9.03	6.88	8.56	*M. silvanus*
C. perfringens	0.42	0.41	0.48	0.21	0.42	0.48	0.21	0.41	0.48	0.21	0.42	*C. perfringens*
C. thermocellum	0.43	0.41	0.5	0.22	0.41	0.5	0.22	0.41	0.5	0.22	0.43	*C. thermocellum*
D. acidiphilus	15.11	14.07	16.81	15.19	14.1	16.83	15.21	14.08	16.81	15.19	15.11	*D. acidiphilus*
D. meridiei	4.61	4.24	5.09	4.84	4.23	5.07	4.82	4.24	5.09	4.84	4.61	*D. meridiei*
D. gibsoniae	6.91	6.38	7.54	7.16	6.38	7.54	7.16	6.38	7.54	7.17	6.91	*D. gibsoniae*
S. pyogenes	0.43	0	0.5	0	0	0.5	0	0	0.5	0	0.43	*S. pyogenes*
T. composti	8.5	7.5	8.64	6.84	7.49	8.62	6.82	7.5	8.65	6.84	8.5	*T. composti*
E. coli	0.18	0.07	0.3	0	0.07	0.3	0	0.07	0.3	0	0.18	*E. coli*
F. aurantia	3.99	3.31	3.83	3.22	3.3	3.82	3.22	3.3	3.82	3.21	3.99	*F. aurantia*
H. baltica	8.16	7.99	8.76	8.17	7.99	8.75	8.17	8.01	8.76	8.18	8.16	*H. baltica*
P. stutzeri	1.55	1.03	1.54	1.25	1.03	1.54	1.25	1.03	1.54	1.25	1.55	*P. stutzeri*
S. bongori	0.14	0.07	0.15	0	0.07	0.15	0	0.07	0.15	0	0.14	*S. bongori*
S. enterica	0.52	0.3	0.56	0.31	0.3	0.57	0.31	0.31	0.57	0.31	0.52	*S. enterica*
S. smaragdinae	11.39	11.17	12.03	11.36	11.16	12.02	11.35	11.15	12.01	11.35	11.39	*S. smaragdinae*
F. pennivorans	11.26	10.16	13.24	11.39	10.16	13.27	11.41	10.17	13.27	11.41	11.26	*F. pennivorans*
C. akajimensis	3.41	3.23	3.54	3.31	3.23	3.53	3.3	3.22	3.53	3.29	3.41	*C. akajimensis*
Other	0	1.13	0.49	0.02	1.13	0.48	0.01	1.13	0.5	0.02	0	*Other*
CV		21.89	11.20	29.09	21.60	11.31	29.03	21.67	11.32	29.03		
R^2		0.993	0.996	0.988	0.993	0.995	0.987	0.997	0.996	0.987		

All benchmark sequence files can be downloaded from http://klif.uu.nl/download/metagenomics_db/

7. REFERENCES

[1] I. Smith, „Mycobacterium tuberculosis Pathogenesis and Molecular Determinants of Virulence," *Clin Microbiol Rev,* vol. 16, nr. 3, pp. 463-596, 2003.

[2] P. Houpikian en D. Raoult, „Traditional and Molecular Techniques for the Study of Emerging Bacterial Diseases: One Laboratory's Perspective," *Emerg Infect Dis,* vol. 8, nr. 2, pp. 122-131, 2002.

[3] R. H. Deurenberg, E. Bathoorn, M. A. Chlebowicz, N. Couto, M. Ferdous, S. García-Cobos, A. M. Kooistra-Smid, E. C. Raangs, S. Rosema, A. C. Veloo, K. Zhou, A. W. Friedrich en J. W. Rossen, „Application of next generation sequencing in clinical microbiology and infection prevention," *Journal of Biotechnology,* vol. 243, pp. 16-24, 2017.

[4] T. N. Petersen, O. Lukjancenko, M. C. F. Thomsen, M. M. Sperotto, O. Lund, F. M. Aarestrup en T. Sicheritz-Pontén, „MGmapper: Reference based mapping and taxonomy annotation of metagenomics sequence reads," *PLoS One,* vol. 12, nr. 5, 2017.

[5] J. O'Niell, „Tackling Drug-Resistant Infections Globally: Final Report and Recommendations," *Ro A,* 2016.

[6] E. Singer, B. Andreopoulos, R. M. Bowers, J. Lee, S. Deshpande, J. Chiniquy, D. Ciobanu, H.-P. Klenk, M. Zane, D. Christopher, A. Clum, J.-F. Cheng, A. Copeland en T. Woyke, „Next generation sequencing data of a defined microbial mock community," *Nature,* 2016.

[7] S. Chen, Y. Zhou, Y. Chen en J. Gu, „fastp: an ultra-fast all-in-one FASTQ preprocessor," *Bioinformatics,* vol. 34, nr. 17, pp. 884-890, 2018.

[8] B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren en A. M. Phillippy, „Mash: fast genome and metagenome distance estimation using MinHash," *Genome Biology,* vol. 17, nr. 132, 2016.

[9] B. Ondov en A. Phillippy, „Mash Screen: what's in my sequencing run?," Genome Informatics , 25 September 2017. [Online]. Available: https://genomeinformatics.github.io/mash-screen/. [Geopend 9 January 2019].

[10] J. Chu, S. Sadeghi, A. Raymond, S. D. Jackman, K. M. Nip, R. Mar, H. Mohamadi, Y. S. Butterfield, G. A. Robertson en I. Birol, „BioBloom tools: fast, accurate and memory-efficient host species sequence screening using bloom filters," *Bioinformatics,* vol. 30, nr. 23, pp. 3402-3404, 2014.

[11] H. Li en R. Durbin, „Fast and accurate short read alignment with Burrows-Wheeler transform," *Bioinformatics,* vol. 25, nr. 14, pp. 1754-1760, 2009.

[12] S. L. Salzberg en B. Langmead, „Fast gapped-read alignment with Bowtie2," *Nature,* vol. 9, pp. 357-359, 2012.

[13] P. Menzel, K. L. Ng en A. Krogh, „Fast and sensitive taxonomic classification for metagenomics with Kaiju," *Nature Communications,* vol. 7, 2016.

[14] D. E. Wood en S. L. Salzberg, „Kraken: ultrafast metagenomic sequences classification using exact alignments," *Genome Biology,* vol. 15, nr. 46, 2014.

[15] J. Lu, F. P. Breitwieser, P. Thielen en S. L. Salzberg, „Bracken: estimating species abundance in metagenomics data," *PeerJ Computer Science,* 2017.

[16] W. P. M. Rowe en M. D. Winn, „Indexed variation graphs for efficient and accurate resistome profiling," *Bioinformatics,* vol. 34, nr. 21, pp. 3601-3608, 2018.

[17] P. T. L. C. Clausen, F. M. Aarestrup en O. Lund, „Rapid and precise alignment of raw reads against redundant databases with KMA," *BMC Bioinformatics,* vol. 19, nr. 307, 2018.

[18] W. Rowe, K. S. Baker, D. Verner-Jeffreys, C. Baker-Austin, J. J. Ryan, D. Maskell en G. Pearce, „Search Engine for Antimicrobial Resistance: A Cloud Compatible Pipeline and Web Interface for Rapidly Detecing Antimicrobial Resistance Genes Directly from Sequence Data," *PLoS One,* vol. 10, nr. 7, 2015.

[19] P. Munk, V. D. Andersen, L. de Knegt, M. S. Jensen, B. E. Knudsen, O. Lukjancenko, H. Mordhorst, J. Clasen, Y. Agerso, A. Folkesson, S. J. Pamp, H. Vigre en F. M. Aarestrup, „A sampling and metagenomic sequencing-based methodology for monitoring antimicrobial resistance in swine herds," *J Antimicrob Chemother,* vol. 72, nr. 2, pp. 385-392, 2017.

[20] K. Bush en G. A. Jacoby, „Updated functional classification of beta-lactamases," *Antimicrob Agents Chemother,* vol. 54, nr. 3, pp. 969-976, 2010.

[21] E. Garrison, J. Sirén, A. M. Novak, G. Hickey, J. M. Eizenga, E. T. Dawson, W. Jones, M. F. Lin, B. Paten en R. Durbin, „Variation graph toolkit improves read mapping by representing genetic variation in the reference," *Nature biotechnology,* vol. 36, pp. 875-879, 2018.

[22] B. Paten, A. M. Novak, J. M. Eizenga en E. Garrison, „Genome graphs and the evolution of genome inference," *Genome Res.,* vol. 27, nr. 5, pp. 665-676, 2017.

[23] E. Zankari, H. Hasman, S. Cosentino, M. Vestergaard, S. Rasmussen, O. Lund, F. M. Aarestrup en M. V. Larsen, „Identification of acquired antimicrobial resistance genes," *Journal of Antimicrobial Chemotherapy,* vol. 67, nr. 11, pp. 2640-2644, 2012.

[24] S. K. Gupta, B. R. Padmanabhan, S. M. Diene, R. Lopez-Rojas, L. Landraud en J. M. Rolain, „ARG-ANNOT, a new bioinformatic tool to discover antibiotic resistance genes in bacterial genomes," *Antimicrob Agents Chemother,* vol. 58, nr. 1, pp. 212-220, 2014.

[25] A. G. McArthur, N. Waglechner, F. Nizam, A. Yan, M. A. Azad, A. J. Baylay, K. Bhullar, M. J. Canova, G. De Pascale, L. Ejim, L. Kalan, A. M. King, K. Koteva, M. Morar, M. R. Mulvey, J. S. O'Brien, A. C. Pawlowski, L. J. Piddock, P. Spanogiannopoulos, A. D. Sutherland, I. Tang, P. L. Taylor, M. Thaker, W. Wang, M. Yan, T. Yu en G. D. Wright, „The comprehensive antibiotic resistance database," *Antimicrob Agents Chemother,* vol. 57, nr. 7, 2013.

[26] M. Frampton en R. Houlston, „Generation of Artificial FASTQ Files to Evaluate the Performance of Next-Generation Sequencing Pipelines," *PLoS One,* vol. 7, nr. 11, 2012.