UNIVERSITY OF PADUA
School of Engineering
Master Degree in Computer Engineering

Academic year 2018-2019

IMPROVING METAGENOMIC CLASSIFICATION BY BOOSTING REFERENCE K-MERS

Supervisor:
Prof. Matteo Comin

Student:
Davide Storato

Student ID:
1153692

December 2, 2019

ABSTRACT

In this thesis a solution is developed to the problem of improving the metagenomic classification trying to boosting the reference *k*-mers. The purpose of this report is to understand if increasing the information available to the classifier help improving the classification at genus and species level.

ACKNOWLEDGMENTS

I dedicate this space of my work to the people who contributed, with their tireless support, to the realization of the same.

A special thanks to my supervisor Prof. Matteo Comin, for his immense patience, for his indispensable advice, for the knowledge transmitted throughout the drafting of the thesis.

I infinitely thank my parents who have always supported me, supporting my every decision, since the choice of my course of study.

Thanks to my friends for being always present even during this last phase of my studies. Thank you for listening to my outbursts, thank you for all the carefree moments.

A heartfelt thanks to my colleagues with whom I shared the entire university course. It is thanks to them that I have overcome the most difficult moments.

Finally, I dedicate this thesis to my family and to myself, to my sacrifices and to my tenacity that have allowed me to get here.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

DNA	DeoxyriboNucleic Acid
RNA	RiboNucleic Acid
MDA	Multiple Displacement Amplification
DTT	DithioThreitol
NGS	Next-Generation Sequencing
SOLiD	Sequencing by Oligo Ligation Detection
GA	Genome Analyzer
bp	base pair
SBS	Sequencing By Synthesis
OTU	Operational Taxonomic Unit
SVM	Support Vector Machine
GSC	Genomic Standards Consortium
MIxS	Minimum Information about any (x) Sequence checklists
MIGS	Minimum Information about a Genome Sequence
MIMS	Minimum Information about a Metagenome Sequence
MIMARKS	Minimum Information about a MARKer Sequence
US	United States
NCBI	National Center for Biotechnology Information
NLM	National Library of Medicine
NIH	National Institutes of Health
EMBL	European Molecular Biology Laboratory
DDBJ	DNA Data Bank of Japan
RefSeq	Reference Sequence
BLAST	Basic Local Alignment Search Tool
LCA	Lowest Common Ancestor
CHT	Compact Hash Table
BFS	Breadth-First Search
RTL	Root-To-Leaf
PPV	Positive Predictive Value
PCC	Pearson Correlation Coefficient
URL	Uniform Resource Locator
SRA	Sequence Read Archive

TP	True Positive
FN	False Negative
VP	Vague Positive
FP	False Positive
BWT	Burrows-Wheeler transform
FM	Ferragina-Manzini
pp	percentage point

1 INTRODUCTION

It is known that only a small part of the microbial life has been identified. Metagenomic, the direct sequencing and characterization of genes and genomes present in a complex environment, has completely changed the microbiological practices, bypassing the obstacle of pure cultural isolation and cultivation. Metagenomic has shown the possibility of increasing the knowledge of diversity, functions and evolution of the uncultivated part of a sample.

Metagenomic as a field arose in the 1990s after the application of molecular biology techniques to the genomic material directly extracted from microbial assemblies present in different habitats, including the human body. The application of metagenomic approaches allows the acquisition of genetic/genomic information from complex assemblies formed by bacteria, viruses, archaea, fungi and protists. The metagenomic field addresses the fundamental questions of which microbes are present and what they do in the environment sample.

In the mid-2000s, the availability of high-throughput and next-generation sequencing technologies gave a boost to the metagenomic field by reducing the financial and temporal limits imposed by the traditional DeoxyriboNucleic Acid (DNA) sequencing technologies. This progress allows the scientific community to examine microbial communities from different habitats/environments, following the structural community changes in the space and in the time and studying the community responses to environment changes.

In 2012, the publication of the characterization of the microbiome of healthy humans has created great expectations about the microbiome's influence on human health and diseases. With the publication of the Human Microbiome Project results, the metagenomic is emerged as an important field of research in microbiology. In particular, when it comes to the characterization of the microbiome in complex human disorders.

A lot of research in the metagenomic field has been done and many tools have been developed for the classification of unknown sequences.

The metagenomic classification tools look for a correspondence between sequences (generally reads or assembly contigs) and a reference database of microbial genomes to identify the taxonomy of those sequences. In the early days of metagenomic, the best strategy was to use the Basic Local Alignment Search Tool (BLAST) [1] to compare each read with all the sequences in the GenBank. But as the reference database and the sequencing datasets dimensions increase, the use of

BLAST became computationally difficult, leading to the development of new metagenomic classification tools that give results in a short time, although usually with a lower sensitivity than BLAST. Some tools return for each read an assignment, instead other provide the overall composition of the sample. For the matching phase a lot of strategies is used: reads alignment, *k*-mer mapping, use of complete genomes, marker genes alignment and protein sequence alignment. Recent studies have tried to compare the metagenomic classifiers performance based on both accuracy and speed [2, 3], although these studies are limited by their dependence on simulated data.

It is seen that the metagenomic classifiers work very well up to genus level, unlike the species level in which they have the most difficulty in classifying the unknown genomes.
This report explains a method to try to improve the classification at genus and species level by boosting the reference *k*-mers set.
The report is structured as described below.

Chapter 2 provides a definition of metagenomic, the description of the metagenomic project and a brief description of the National Center for Biotechnology Information (NCBI) [4].

Chapter 3 provides a detailed description of the main tool used.

Chapter 4 provides the idea and its implementation details to resolve the task of this report.

Chapter 5 provides a description of the experiments conducted and the results obtained.

Appendix A provides a brief introduction to the use of the Blade Computing Cluster at the Department of Information Engineering and the tables with the results obtained.

2 METAGENOMIC

Metagenomic is the application of modern genomic techniques to the study of communities of microbial organisms directly in their natural environments, bypassing the need for isolation and lab cultivation of individual species [5]. In its approaches and methods, metagenomic goes beyond the individual genome and allows to study all the genomes of a community as a whole and quantifying its diversity in terms of species abundance. The analysis of sequences of an ecosystem is performed without know the microorganisms that the ecosystem contains. For this reason, the metagenomic information allow a more detailed comprehension of the ecological role, the metabolism and the evolutionary history of microorganisms in a given context and how the environment influences the genomic composition of entire species. Combined with the modern sequencers capacity of obtain DNA fragments very quickly, the metagenomic studies can generate huge amounts of data. Many genomic sequences have similarity with those already studied and can be identified through the use of alignment algorithms with high sensitivity. The problem of classifying and determining the origin of a DNA sequence, given a set of known genomes, is common to many fields of the molecular computational biology. The metagenomic allows to assess the abundance of each organism present in a system, as well as the abundance of the genes present within it that encode enzymes and proteins active in metabolic pathways present in the sample being analyzed. Metagenomic has the potential to advance knowledge in a wide variety of fields, such as agriculture, biofuel, biotechnology, ecology and medicine. In clinic field, metagenomic is widely used in the study of oral and intestinal microflora, genetic diseases and neoplasms.

2.1 METAGENOMIC PROJECT

A metagenomic project differs respect to a genomic project in many aspects. In fact, the metagenomic project uses an environment sample that contains a community composed by different species of microorganisms. Many of these microorganisms can not be cultivated in laboratory and so they can not be studied. Since the sample contains many different organisms, it is possible to have DNA contaminations that make genomic sequences difficult to obtain. Often the real scope of metagenomic is not the generation of complete genomes, but understanding the composition of the microorganisms community and the iterations between the community and the environment.

The metagenomic project steps are illustrated in Figure 2.1 and briefly explained below.

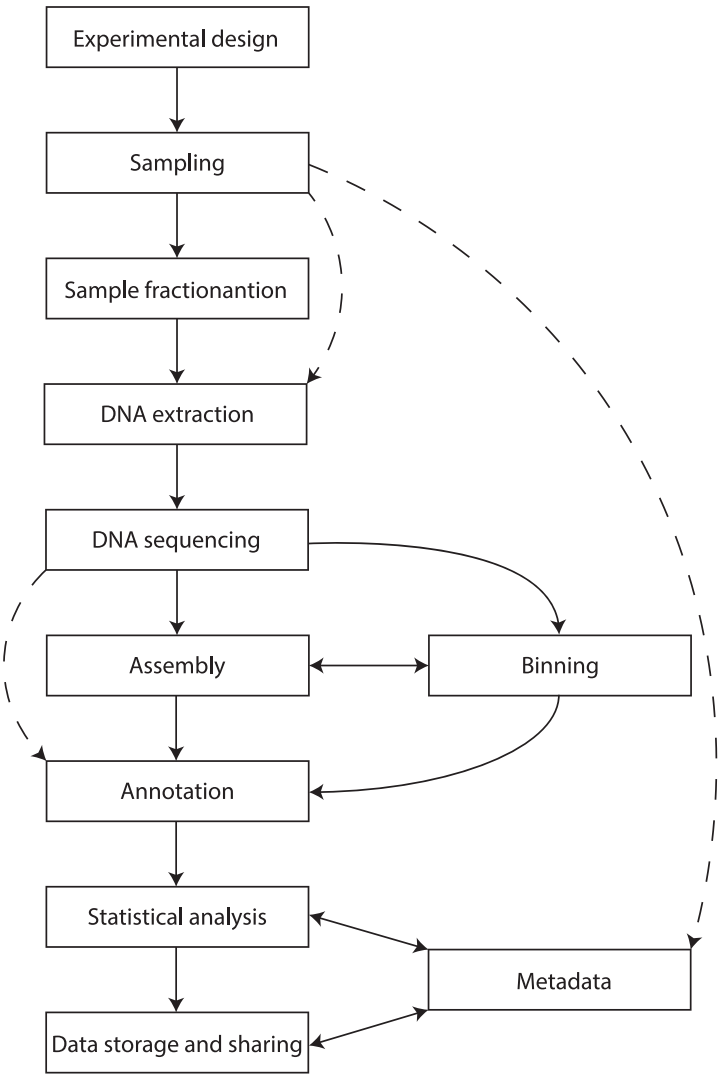

Figure 2.1: Flow diagram of a typical metagenomic project. The dashed arrows indicate steps that can be omitted. (Based on figure in [6].)

Sampling is the first and most crucial step in any metagenomic project addressed to preserve the DNA quality. The DNA extracted should be representative of all the species present in the sample and sufficient amounts of high-quality of nucleic acids must be obtaining for a subsequent use in sequencing and library production. During this phase can be collected the so-called metadata.

Metadata are additional data, strongly dependent on the sample type, that include biochemical data, geographical data and sample-processing data.

After sampling, the physical separation and isolation of the species from the sample is done in order to maximize the DNA production or avoid coextractions of microorganisms or substances that might inter-

fere with the next steps of the metagenomic project. After separation, the DNA extraction is performed.

DNA extraction is the process by which DNA is separated from proteins, membranes and other cellular material contained in the cell from which it is recovered [7].
The DNA extraction generally follows three basic steps:

1. Lyse (break open) the cells.
2. Separate the DNA from the other cell components.
3. Isolate the DNA.

Cell lysis or cellular disruption is a method in which the cell membrane is broken down or destroyed in order to release inter-cellular materials such as DNA, RNA, protein or organelles from a cell [8]. The lysis procedure must be strong enough to fragment the basic material, obtained from the sample, but not too much. If the lysis is too strong, one risk to losing the integrity of the DNA fragments. For a review of the cell lysis methods see [8]. After the lysis procedure, a complex mixture is obtained. This mixture is composed of cellular components such as proteins, lipids, carbohydrates, DNA, RiboNucleic Acid (RNA). These additional components can interfere with the subsequent steps of the metagenomic project. Therefore, the DNA extraction is required. Specific chemical compounds are used to isolate the DNA fragments from the other cellular components which, in water solution, can aggregate with the DNA and create a precipitate.
There are four commonly used extraction procedures for DNA [9]:

1. Organic (variations of phenol/chloroform): use of a multistep liquid chemical process that is labor intensive but produces a high yield and very-clean double-stranded extracted DNA sample.
2. Inorganic Chelex or silica methods: simple and cheap one-tube extraction process in which Mg^{2+} binds to resin beads and yields a single-stranded DNA product.
3. Solid phase extraction methods: simple extraction process in which the DNA binds to paramagnetic or silica beads; example of these methods are Promega's DNA IQ [10], Applied Biosystems' PrepFiler [11] and Qiagen's QIAamp kits [12, 13].
4. Differential extraction: a multistep process used to separate sperm from other cells using DithioThreitol (DTT); used for analyzing biological evidence from sexual assault cases [14].

With some types of sample are possible to produce only a very small amount of DNA, but most sequencing technologies require high amount of data. Therefore, the amplification of such data is required. A widely used amplification is the Multiple Displacement Amplification (MDA) that can produce an amplification of nine order of

magnitude [15, 16]. As happens in any amplification, one can have problems associated with reagent contaminations, chimera formation and sequence bias. These problems depend on the amount and type of the initial sample and the number of amplification rounds needed to produce the sufficient amount of DNA bases. Once the DNA extraction phase ends, the sequencing phase starts.

Sequencing is the process of determining the nucleic acid sequence, so in determing the order of nucleotides in DNA strand. It includes any method or technology that is used to determine the order of the four canonical bases of the DNA, i. e., adenine, guanine, cytosine and thymine. The advent of rapid DNA sequencing methods has greatly accelerated biological and medical research and discovery [17].

For decade the Sanger sequencing technology was used in the first-generation sequencers. Sanger sequencing [18] was developed by Sanger and colleagues in 1977. Is a sequencing technique based on the chain-terminating inhibitors method. This method has been so successful at that time thanks to its efficiency and the non-use of expensive and dangerous radioactive reagents as required by the Gilber method [19]. Other advantages that led to the use of sequencers based on the Sanger method are the low error rate, the long read length ($>$ 700 base pairs (bps)) and the large insert sizes. A drawback of these sequencers is the labor-intensive cloning process in its associated bias against genes toxic for the cloning host [20] and the overall cost. These tools have made an important contribution to the completion of the "Human Genome Project" [21].

Although first generation technologies have been fundamental to success in the Human Genome Project, this project has led to an important push for the development of new sequencing platforms. The Next-Generation Sequencing (NGS) stand out from the first-generation tools for the parallel analysis and the high throughput. They are also more advantageous in terms of costs. The most used NGS tecnologies are Roche 454 System, AB SOLiD System and Illumina HiSeq System.

Roche 454 was the first commercially successful next generation system. This sequencer uses pyrosequencing technology. Instead of using dideoxynucleotides to terminate the chain amplification, pyrosequencing technology relies on the detection of pyrophosphate released during nucleotide incorporation. The read length of Roche 454 was initially 100-150 bps in 2005, 200000 and more reads and could output 20 Mb per run [22, 23]. In 2008 454 GS FLX Titanium system was launched; through upgrading, its read length could reach 700 bps with accuracy 99.9% after filter and output 0.7 G data per run within 24 hours. In late 2009 Roche combined the GS Junior a bench top system into the 454 sequencing system which simplified the library preparation and data processing, and output was also upgraded to 14 G per run [24, 25]. In 2013 Roche announced the disposal of the project. The main advantages of the 454 systems are the read length and the high

speed for every single run, in only 10 hours the machine can complete the entire sequencing process. However, the reagents used by this technology for the sequencing procedure are more expensive than the other two technologies. In addition to that, the reads produced are subject to errors.
The Sequencing by Oligo Ligation Detection (SOLiD) sequencer was introduced in 2006. The sequencer adopts the technology of two-base sequencing based on ligation sequencing. On a SOLiD flow cell, the libraries can be sequenced by 8 base-probe ligation which contains ligation site (the first base), cleavage site (the fifth base) and 4 different fluorescent dyes (linked to the last base) [23]. The pair bases are identified with a color code. The advantage of SOLiD is the extreme accuracy of the procedure thanks to the multiple analysis of each base. The read length of the first SOLiD sequencers was of 35 bps and the output data was of 3 G of data represented in color-scheme. In 2010, the SOLiD 5500xl sequencing system was released. This sequencer improved the read length to 85 bps and the output to 30 G. As mentioned, SOLiD allows a high read accuracy, reaching a value of 99.99%. However, this level of accuracy occurs at the expense of the read length which is extremely limited. The most recent SOLiD sequencer run takes 7 days to complete and generates around 4 T of raw data.
In 2006 the Genome Analyzer (GA) was released by Solexa. In 2007 the company was purchased by Illumina. The company continues the evolution of this sequencer. The GA sequencer adopts the technology of Sequencing By Synthesis (SBS). Since the first versions, the sequencer distinguished itself for the high throughput. The first GA released by Solexa generates in output 1 G of data. In 2009, thanks to improvements in polymerase, buffer, flow cell and software the output increased around 50 G per run. In early 2010, Illumina releases HiSeq 2000. This sequencer uses the same sequencing techniques used by the previous GA. The output generated was around 200 G per run and later incremented to around 600 G. The read length is between 50 and 200 bps with the possibility of single-end or paired-end sequencing. Respect to the other technologies analysed, the HiSeq system is characterized by the high throughput and by the lower cost of the reagents used during the procedure. This happens at the expense of the length of the reads and the accuracy, which in this system is confirmed on 98% [21]. Obtained the reads from the DNA extracted, the assembly phase starts.

Assembly is the process of combining sequence reads into contiguous traits of DNA, called contigs, in order to reconstruct the original sequence. The assembly process, as illustrated in Figure 2.2, starts with the reads obtained by the sequencing phase. With these reads the assembly process produces the contigs, based on sequence similarity between reads. The consensus sequence for a contig is either based on the highest quality in any given reads at each position or based

on majority rule (i. e., the most frequently encountered nucleotide at each position). Two contigs can be linked into a larger not continuous DNA sequence, called scaffold, if the paired reads are present in two different contigs.

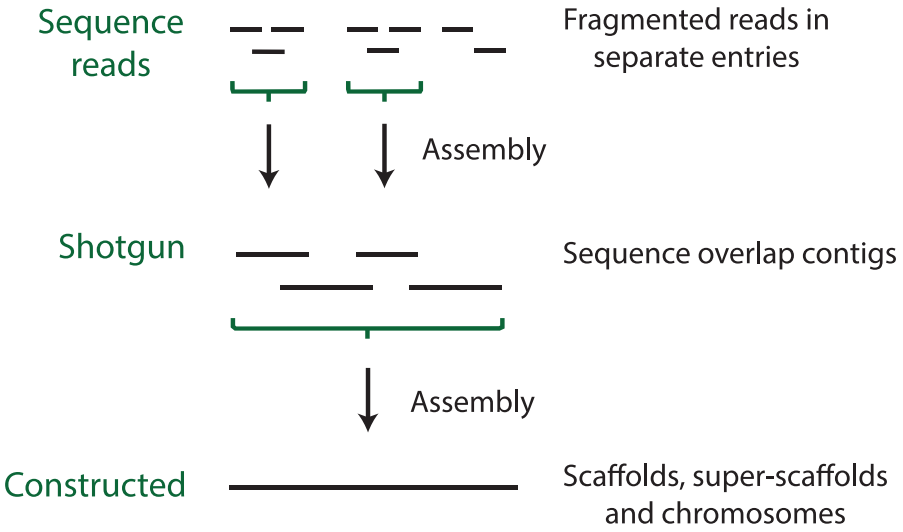

Figure 2.2: Example of an assembly process.

For the sequence assembly, two strategies can be employed:

- reference-based assembly.
- de novo assembly.

The reference-based assembly aligns reads and/or contigs to a known reference genome of a closely releated organism. As one can guess, this strategy works well if the reads/contigs are closely related to the reference genome sequences. If the reference-based assembly founds differences between the sample and the reference genome, this leads to fragmentations of the assembly or the non-coverage of the divergent regions.
The de novo assembly assembles short reads to create full-length (sometimes novel) sequences without using a known reference genome. These types of assembler require a larger amount of computational resources than the reference-based assembler, just see the assembler Velvet [26] or SOAP [27] requirements. For handle a large amount of data, a class of assembly tools was developed based on the de Bruijn graphs [28, 29].
However, assembly remains a very expensive computational problem because the reads can contain errors or have low quality, DNA repetitions and non-covered regions that make gaps.

After assembly or sequencing phase the binning can be done. *Binning* is the process of sorting DNA sequences into groups that represent a genome or genomes from related organisms. Since the amount of reads contained in a metagenomic sample is high and the incomplete

nature of the obtained sequences make it hard to assemble individual genes [30] (much less recovering the full genome of each organism) binning helps to identify reads or contigs with certain groups of organisms designated as Operational Taxonomic Units (OTUs) [6]. (An OTU is an operational definition used to classify groups of closely related individuals.)
Modern binning techniques employ two types of information:

1. Compositional information.
2. Similarity information.

The composition-based binning makes use of the fact that genomes have conserved nucleotide composition (e. g., a certain CG composition or a particular abundance distribution of the k-mers) and will be also reflected in sequence fragments of the genomes. The composition-based binning can be divided in two procedure types: supervised and unsupervised.
The supervised procedure classify DNA fragments against models trained on classified reference sequences. Examples of supervised approaches are the Bayesian classifiers [31] and the Support Vector Machine (SVM) based phylogenetic classifier Phylopythia [32].
The unsupervised method clusters the metagenomic fragments without the need to train models on reference sequences database. This procedure includes self-organizing maps [33] and the program TETRA [34]. The similarity-based binning makes use of the fact that unknown sequences of DNA might encode to a gene and the similarity of this gene with known genes (present in a reference database) can be used to bin the sequence.
For a review of the methodologies, advantages, limitations and challenges of various methods available for binning see [35] or [36].
Some compositional-based binning algorithms are MEGAN [37], Phylopythia [32], S-GSOM [38], PCAHIER [39, 40] and TACOA [40]. Purely similarity-based binning algorithms are IMG/M [41], MG-RAST [42], MEGAN [37], CARMA [43], SOrt-ITEMS [44] and MetaPhyler [45]. There are also hybrid binning algorithms that use both composition and similarity, such as PhymmBL [46] and MetaCluster [47].

After the sequencing, assembly or binning the annotation phase starts. *Annotation* is the process of identifying the locations of genes and all of the coding regions in a genome and determining what those genes do. An annotation is a note added by way of explanation or comment. Once a genome is sequenced, it needs to be annotated to make sense of it.
For DNA annotation, a previously unknown sequence representation of genetic material is enriched with information such as genomic position, regulatory sequences, repeats, gene names and protein products. This annotation is stored in genomic databases.
In metagenomic, the annotation process can be taken two pathways:

1. If the objective of the study are the reconstructed genomes and assembly has produced large amount of data (minimum contigs length required is 30000 bps), then is preferable to use existing pipelines for genome annotation; for example RAST [48] or IMG ER [49].

2. If annotation can be preformed on a entire community and relies on not assembled reads or short contigs, then is more useful use annotation tools specifically developed for metagenomic analyses then tools for genomic annotation.

In general, metagenomic annotation process has two steps:

1. Feature prediction, where it is identified the features of interest (genes).

2. Functional annotation, where it is assigned putative gene function and taxonomic neighbors. An example of metagenomic functional annotation workflow is shown in Figure 2.3.

Currently, metagenomic annotation relies on classifying sequences to known functions or OTUs based on homology searches against available annotated data.

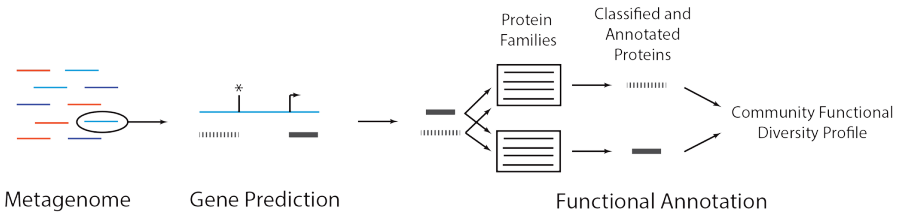

Figure 2.3: A metagenomic functional annotation workflow. A metagenome (colored lines, left) can be annotated by subjecting each reads to gene prediction and functional annotation. In gene prediction, various algorithms can be used to identify subsequences in a metagenomic read (blue line) that may encode proteins (gray bars). In some situations, coding sequences may start (arrow) or stop (asterisk) upstream or downstream the length of the read, resulting in partial gene predictions. Each predicted protein can then be subject to functional annotation, wherein it is compared to a database of protein families. Predicted peptides that are classified as homologs of the family are annotated with the family's function. Conducting this analysis across all reads results in a community functional diversity profile [50].

Obtained all possible information from the genes, the statistical analysis starts. The *statistical analysis* aims to reduce the overall variability of the data by removing systematic errors. This is followed by identification of the genes that are differentially abundant between the studied experimental conditions using statistical models. The statistical analysis starts with the quantified gene abundances and aims

to pinpoint the specific differences between the studied microbial communities.
The statistical analysis of gene abundances, in metagenomic data, is separated into two main steps:

1. Normalization, that aims to remove unwanted variation, such as differences in sequencing depth between samples and other forms of noise that systematically affect genes or samples.

2. Identification of differentially abundant genes, that applies statistical models to identify the genes that significantly change between experimental conditions.

Once all the information and DNA strands have been processed, the obtained data need to be stored. Then the data storage phase and, subsequently, the data sharing phase start. The *data sharing* of metagenomic data requires a level of organization and collaboration to provide metadata and centralized services as well as sharing of both data and computational results. In order to enable sharing of computed results, a standardization is necessary. These is currently piked up by the Genomic Standards Consortium (GSC). Once this has been achieved, researchers will be able to download intermediate and processed results from any one of the major repositories for local analysis or comparison. A suite of standard languages for metadata is currently provided by the Minimum Information about any (x) Sequence checklists (MIxS) [51]. This is a term to describe the Minimum Information about a Genome Sequence (MIGS), the Minimum Information about a Metagenome Sequence (MIMS) and the Minimum Information about a MARKer Sequence (MIMARKS) [51] and contains standard formats for recording environmental and experimental data.
The *storage* of all metagenomic data is piked up by the United States (US) NCBI.

2.2 NATIONAL CENTER FOR BIOTECHNOLOGY INFORMATION

The NCBI is part of the US National Library of Medicine (NLM), a branch of the National Institutes of Health (NIH). NCBI assumed responsibility for the GenBank DNA sequence database in October 1992. NCBI has the task of building the database from sequences submitted by individual laboratories and by data exchange with the international nucleotide sequence databases, European Molecular Biology Laboratory (EMBL) [52] and the DNA Data Bank of Japan (DDBJ) [53]. The NCBI major databases include GenBank (an open access, annotated collection of all publicly available nucleotide sequences and their protein translations), PubMed (bibliographic database for the biomedical literature) and Reference Sequence (RefSeq) database (a public database of nucleotide

and protein sequences with corresponding feature and bibliographic annotation [54]). All these databases are available online through the Entrez search engine. Entrez is a molecular biology database and retrieval system developed by the NCBI that presents an integrated view of biomedical data and their interrelationships (Figure 2.4) [55].

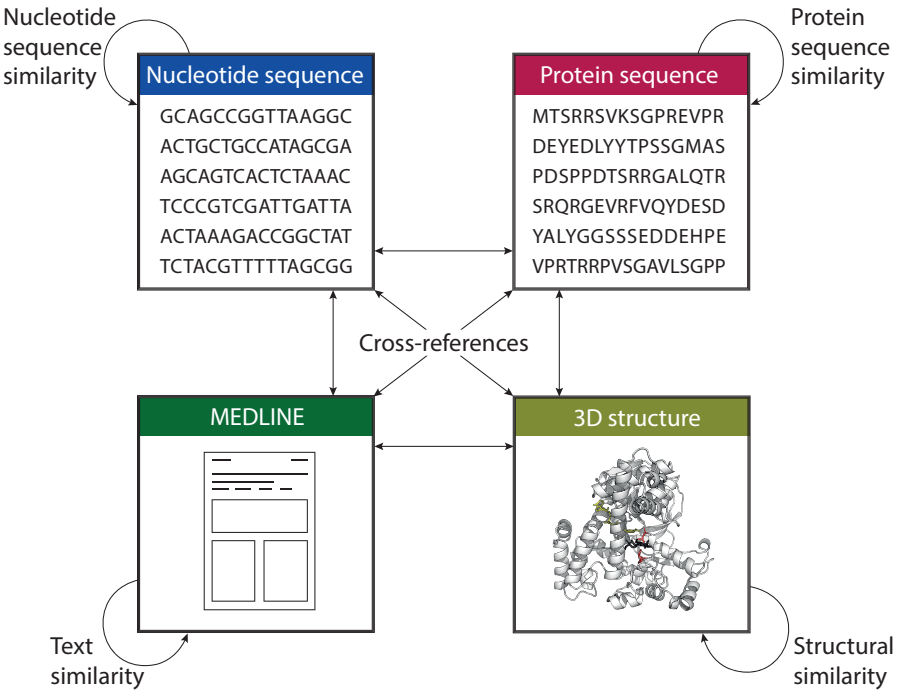

Figure 2.4: Different classes of documents available for browsing in Entrez are linked by both intrinsic cross-reference information and computed relationships [55].

In addition to the databases described above, the NCBI includes a taxonomy database. Taxonomy is the process of naming and classifying organisms, such as animals and plants, into groups within a larger system, according to their similarities and differences. The classification of organisms has various hierarchical categories, also called taxonomy ranks. These categories gradually shift from being very large and including many different organisms to very specific and identifying single species. An example of taxonomy ranks is shown in Figure 2.5.

The NCBI Taxonomy database is a curated set of names and classifications for all of the organisms that are represented in GenBank. The NCBI taxonomy maintains a phylogenetic taxonomy. In a phylogenetic classification scheme, the structure of the taxonomic tree approximates the evolutionary relationships among the organisms included in the classification. Like genome database, the taxonomy database can be queried using the Entrez search engine.

In addition to store and update the genome and taxonomy databases, the NCBI develops bioinformatics tools. The most famous is BLAST.

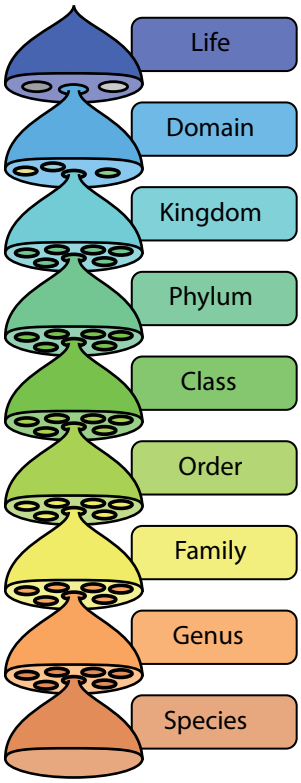

Figure 2.5: The basic scheme of modern classification [56].

BLAST uses a local alignment method, namely uses a subset of a sequence and attempts to align it to subset of other sequences.

3

TOOL DESCRIPTION

The main tool used in this work is the Derrick E. Wood, Jennifer Lu and Ben Langmead's software Kraken 2 [57].

Kraken 2 is a new version of Kraken [58] that aims to reduce the time and the amount of memory necessary to build the database from reference sequences (~85% [57]) and less time and memory usage during the classification phase.
As mentioned on [59], Kraken 2 differs from Kraken in the following features:

1. Only minimizers of the k-mers in the query sequences are used as database queries. Similarly, only minimizers of the k-mers in the reference sequences in the database's genomic library are stored in the database. We will also refer to the minimizers as l-mers, where $l \leq k$. All k-mers are considered to have the same Lowest Common Ancestor (LCA) as their minimizer's database LCA value.

2. Kraken 2 uses a Compact Hash Table (CHT) that is a probabilistic data structure. This means that occasionally, database queries will fail by either returning the wrong LCA, or by not resulting in a search failure when a queried minimizer was never actually stored in the database. By incurring the risk of these false positives in the data structure, Kraken 2 is able to achieve faster speeds and lower memory requirements. Users should be aware that database false positive errors occur in less than 1% of queries, and can be compensated by the use of confidence scoring thresholds.

3. Kraken 2 has the ability to build a database from amino acid sequences and perform a translated search of the query sequences against that database.

4. Kraken 2 utilizes spaced seeds in the storage and querying of minimizers to improve classification accuracy.

5. Kraken 2 provides support for "special" databases that are not based on NCBI's taxonomy. These are currently limited to three popular 16S databases.

In Figure 3.1 is shown the algorithm and data structure differences between Kraken 2 and Kraken 1.

The CHT mentioned above uses a fixed-size array with 32 bit hash cells. Each cell stores a key-value pair. In a cell, the number of bits

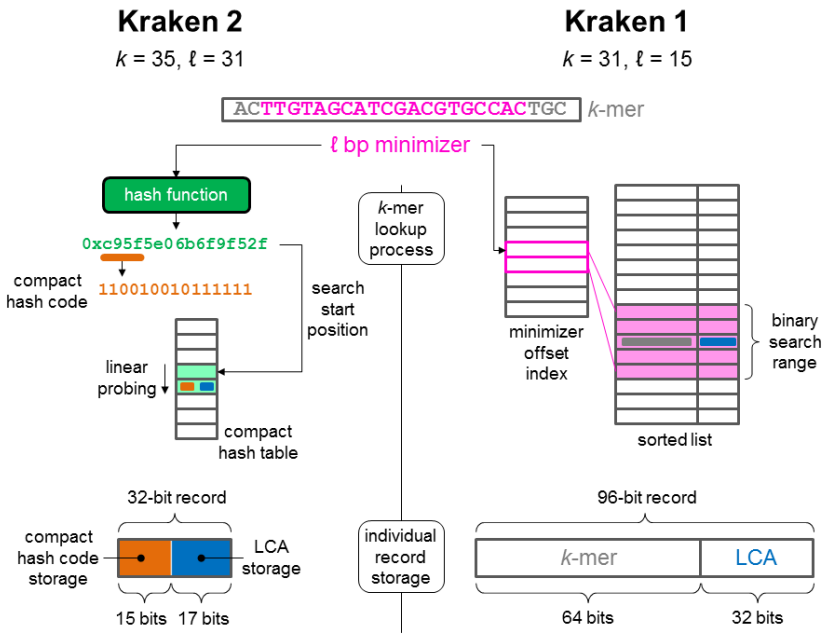

Figure 3.1: Algorithm and data structure differences between Kraken 2 and Kraken 1 [60].

used to store the key-value pair will vary. It depends on the number of bits necessary to represent all unique taxonomy ID numbers (values) found in the reference sequences. In Figure 3.2 is shown the content and partition of a CHT cell.

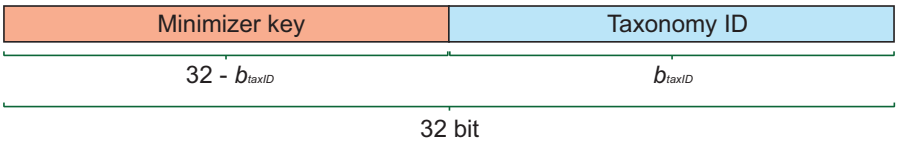

Figure 3.2: The Compact Hash Table (CHT) cell. The most significant bits of the minimizer key is inserted in the $32 - b_{taxID}$ most significant bits of the CHT cell. 32 is the dimension (in bit) of one CHT cell and b_{taxID} is the number of bits necessary to store the internal taxonomy ID of the LCA associated to the minimizer.

For the full running, Kraken 2 needs the BLAST software because it uses *dustmasker* [61] and *segmasker* [62] functions. These functions are used by Kraken 2 to mask the low-complexity sequences. A low-complexity sequence is simple repeats (i. e., ATATATATAT) or regions that are highly enriched for just one letter (e. g., AAACAAAAAAA-GAAAAAAC). Protein segments with only a few amino acids are also considered to be low complexity (e. g., PPCDPPPPPKDKKKKDDGPP).

Once Kraken 2 is installed, using the `install_kraken2.sh` script, the user can use `kraken2-build` and `kraken2` main scripts. With the first script the user can build the Kraken 2's database from one or more databases of reference sequences. The second script classifies

the user input file containing the unknown sequences (reads). Before classifying it is necessary the construction, and therefore the presence, of the database.

3.1 DATABASE BUILDING

Database building is the most time consuming Kraken 2's phase and is strongly dependent on the amount of data to be processed. Kraken 2 has two database building types: *standard* and *custom*. For both types is necessary to download the taxonomy data and the reference sequences.

The *standard database* building type builds the database from the reference genomes, downloaded from the NCBI RefSeq database. The Kraken 2's reference genomes database includes the archaeal, bacterial, viral and human (GRCh38) [63] genomes and the UniVec_Core of the UniVec database [64]. The last two databases are downloaded to make the classification of reads containing the human genome easier and for improve the precision of reads containing vector sequences. Another reason why Kraken 2 downloads these two more databases than Kraken is for the lesser use of memory and the shorter building time.
To build a standard database with Kraken 2, the `kraken2-build --standard --db $DBNAME` command is used, where the `$DBNAME` must be replace with the database name or location.

The *custom database* building type creates the Kraken 2's database from the reference sequences databases chosen by the user. This is useful when the user wants to classify from a specific database, he has not sufficient amount of memory to build the standard database or he wants to add reference genomes not from the NCBI database.
First, the user has to download the NCBI taxonomy with the `kraken2-build --download-taxonomy --db $DBNAME` command. After having downloaded the taxonomy, the user can choose which database to download with the `kraken2-build --download-library $REFDBNAME --db $DBNAME` command, where `$REFBDNAME` can be one of the following databases:

- *archaea*: RefSeq complete archaeal genomes/proteins.
- *bacteria*: RefSeq complete bacterial genomes/proteins.
- *plasmid*: RefSeq plasmid nucleotide/protein sequences.
- *viral*: RefSeq complete viral genomes/proteins.
- *human*: GRCh38 human genome/proteins.
- *fungi*: RefSeq complete fungal genomes/proteins.
- *plant*: RefSeq complete plant genomes/proteins.

- *protozoa*: RefSeq complete protozoan genomes/proteins.
- *nr*: NCBI non-redundant protein database.
- *nt*: NCBI non-redundant nucleotide database.
- *env_nr*: NCBI non-redundant protein database with sequences from large environmental sequencing projects.
- *env_nt*: NCBI non-redundant nucleotide database with sequences from large environmental sequencing projects.
- *UniVec*: NCBI-supplied database of vector, adapter, linker, and primer sequences that may be contaminating sequencing projects and/or assemblies.
- *UniVec_Core*: A subset of UniVec chosen to minimize false positive hits to the vector database.

If the user wants to add sequences not present in the previous list or not from the NCBI database, he can do it with the `kraken2-build --add-to-library $FILE --db $DBNAME`, where `$FILE` is the file to add in the database. Kraken 2 requires this file has the following characteristics:

- The file containing the genomes must be in a FASTA format (multi-FASTA is allowed).
- Each sequence's ID must contain either an NCBI accession number or an explicit assignment of the taxonomy ID in *kraken:taxid* format (see Example 3-1).

In Example 3-1 is shown a sequence format of a known adapter sequence in taxonomy 32630 ("synthetic construct") to add to Kraken 2's database.

Example 3-1

>sequence16|kraken:taxid|32630 Adapter sequence
CAAGCAGAAGACGGCATACGAGATCTTCGAGTGACTGGAGTT...

■

Once the user downloads/adds the necessary files, for build the database he must uses the `kraken2-build --build --db $DBNAME` command.

The database (both standard and custom) building process is shown in Figure 3.3 and is divided in three steps:

1. *Step 1*: creates sequence ID to taxonomy ID map.
2. *Step 2*: estimates required capacity.
3. *Step 3*: builds database files.

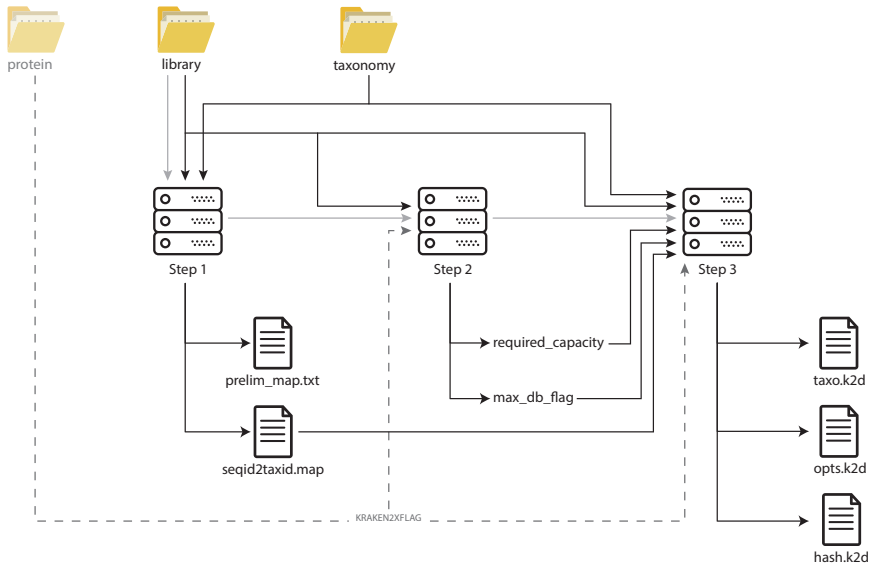

Figure 3.3: Kraken 2 database building steps.

As mentioned above, before the database building, the user needs to download the necessary files. When Kraken 2 starts the download of the files from the NCBI database, it downloads the *assembly_summary.txt* files for each genome types. This file contains all the information relating to the files that belong to the selected database, such as the sequence file name, the taxonomy ID, the scientific name, the file path and other information. Kraken 2 downloads and parses the sequence files and, if enabled, removes the low-complexity sequences. The low-complexity sequences are known to occur in a lot of different organisms and are typically less informative for sequences alignment. Kraken 2 uses the *dustmasker* [61] and the *segmasker* [62] tools for masking the low-complexity sequences from nucleotide and protein sequences respectively. Using low-complexity sequences masking can help prevent false positives in Kraken 2's results. If the user does not want to install BLAST or not want to mask low-complexity sequences, he can use the `--no-masking` option.

Downloaded the taxonomy and reference sequences and created the necessary files, *step 1* starts. This step has the task of create the sequence ID to taxonomy ID map. The first step is to search for *prelim_map.txt* files in the directories containing the downloaded genome databases. These files are concatenated and the result is stored in the *prelim_map.txt* file in the taxonomy directory. Subsequently, Kraken 2 divides the *prelim_map.txt* file in two temporary files *seqid2taxid.map.tmp* and *accmap_file.tmp*. These two temporary files contain the reference sequences taxonomy ID list and the accession number list respectively. Created the latest two files, Kraken 2 verifies the presence of the **.accession2taxid* files in the taxonomy directory. If there are, Kraken 2 creates a new file containing the sequence ID and

the taxonomy ID obtained from *accmap_file.tmp* and **.accession2taxid* files. Once obtained the sequence ID to taxonomy ID map, the map is stored in the *seq2taxid.map* file and the step 1 ends.

Ended the step 1, the *step 2* starts. This step has the task of estimate the CHT capacity. Firstly, Kraken 2 estimates then number of distinct minimizers in the reference sequences for selected values of k, l and s, where k and l are the k-mer and minimizer length respectively. Kraken 2 default values for k, l and s are 35, 31 and 7 respectively. The parameter s indicates the number of positions, from the second position from the right, of the minimizer that are masked. The parameter is part of the spaced k-mers approach (a similar concept of spaced seeds). With these approach is verified an improvement in the reads classification ability [65]. To obtain the estimate of the number of distinct sequences in the reference database, Kraken 2 uses a form of zeroth frequency moment estimation [66] that creates a small set structure (Q) implemented with a traditional hash map. In Q is added the distinct minimizers that verify the following inequality:

$$h(m) \mod F < E \tag{3.1}$$

where $h(m)$ is the hash code of the minimizer m, F is the section range (must be a power of two) and E is the maximum quantification hash code. E must be much smaller than F, Kraken 2 default values of E and F are 4 and 1024 respectively. Subsequently, Kraken 2 estimates the total number of distinct minimizers (D) with the following equation:

$$D = |Q| \cdot \frac{F}{E} \tag{3.2}$$

with $|Q|$ is the number of distinct minimizers that satisfied Equation 3.1. Obtained D, Kraken 2 computes the amount of memory to be allocated, in byte, with the following formula:

$$\frac{4 \cdot D}{0.7} \tag{3.3}$$

where 4 is the number of bytes of a CHT cell, D is the value computed with Equation 3.2 and 0.7 is the amount, in percentage, of cells occupied in the CHT. The CHT is not completely filled for efficiency reasons and due to the low probability of errors. The amount of memory is store in the *max_db_flag* variable and the step 2 ends.

Ended the step 2, the *step 3* starts. This step has the task of create all the database files that will be used in the classification phase. Firstly, Kraken 2 generates the internal taxonomy tree. To do this, Kraken 2 begins by taking the sequence ID to taxonomy ID mapping data from *seqid2taxid.map* file and saves the content in a hash map. Subsequently the internal taxonomy tree is builded. This representation is different from the one provided by the user (in this case by the NCBI). In fact, firstly Kraken 2 finds a minimal set of nodes. This set consists of all

the nodes that have both themselves and their ancestors a non-zero taxonomy ID. The taxonomy structure of the nodes contained in the set is maintained as it is in the user-provided taxonomy. At this point, Kraken 2 assigns to each node previously found an increasing taxonomy ID according to the Breadth-First Search (BFS) approach starting from the root of the tree, with ID equal to 1. An example of internal taxonomy ID is shown in Figure 3.4.

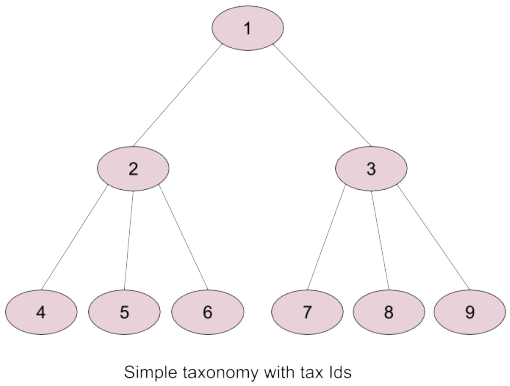

Figure 3.4: An example of Kraken 2's reduced internal representation of the taxonomy with sequential ID numbering via breadth-first search [57].

The internal taxonomy representation leads to the following advantages:

- Provides a guarantee that ancestor nodes will have smaller internal ID numbers than their descendants [57].
- Space reduction for store the taxonomy ID, thus increasing the space for the CHT hash code with a probability reduction of errors (hash table collision).
- Simplifies the LCA computation of two nodes as their taxonomy IDs give information on how deep they are in the tree.

Kraken 2 during the internal taxonomy ID assignment creates a internal taxonomy ID to external taxonomy ID map so the user can have understandable results.
Completed the internal taxonomy construction, Kraken 2 initializes and populates the CHT. To do this, Kraken 2 scans every genome in the reference database. All the genomes that have a taxonomy ID can be inserted in the CHT (as Kraken 2 can compute, if necessary, the LCA), while the genomes without taxonomy ID are not processed. For each genome G, Kraken 2 computes its minimizer and, one by one, tries to insert the key computed for minimizer M ($h(M)$) in CHT with the taxonomy ID T associated. The minimizer key $h(M)$ is obtained from the finalization function of MurmurHash3 [67]. The key $h(M)$ is inserted in the $32 - b_{taxID}$ most significant bits of the CHT cell, where

32 is the dimension of one CHT cell and b_{taxID} is the number of bits necessary to store the internal taxonomy ID of the LCA associated to the minimizer M. b_{taxID} is computed by choosing the number of left shifts of 1 that gives the smallest power of two necessary to store the number of nodes (than the biggest internal taxonomy ID) of the Kraken 2's taxonomy tree.
If during the insertion of the key $h(M)$ not result its presence in the CHT, then the $< h(M), T >$ pair is added to the CHT, indicating that T is the LCA of the minimizer M. If $h(M)$ is in the CHT with taxonomy ID T^*, then the LCA is updated with the LCA of T and T^*. An example of insertion in the CHT is shown in Figure 3.5.

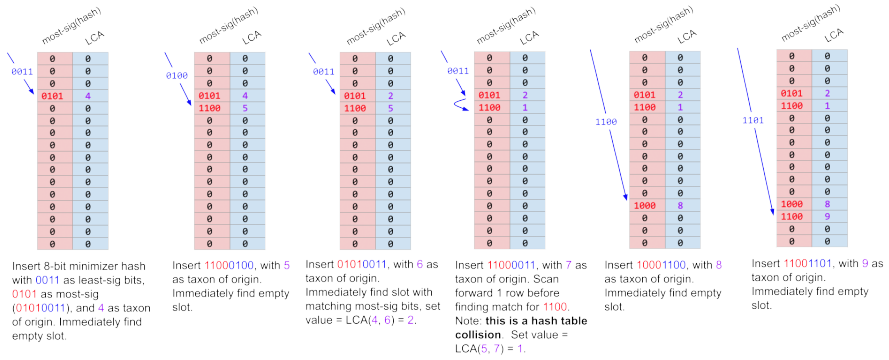

Figure 3.5: Sequential examples of Kraken 2's insertion of minimizer/LCA pairs into a compact hash table [57].

Once all the reference genomes (and so all the minimizers) are processed, all the LCAs are correctly set for each minimizer and step 3 ends. Just before finishing, step 3 produces the following essential files:

- *hash.k2d* containing the minimizer to taxon mappings.
- *opts.k2d* containing the information about the options used to build the database.
- *taxo.k2d* containing the taxonomy information used to build the database.

With the end of step 3 the database building process is complete.

After building the database, the user can reduce the disk space occupied by the database using the `kraken2-build --clean $DBNAME` command. This command removes intermediate and useless files from the database directory `$DBNAME`.

3.2 CLASSIFICATION

For classify an unknown sequence (read) S, Kraken 2 finds, for each k-mers in S, its minimizer and, if it is distinct from the previous

minimizer, uses it as key for querying the CHT. For probe the CHT, Kraken 2 computes the key $h(M)$, then linearly scans the table starting from position $h(M) \mod |T|$ for a matching key, where $|T|$ is the number of cells in the table. If during the querying of the CHT there is a match with a key in the table, Kraken 2 considers the LCA value associated with the key as the k-mer's LCA. An example of querying the CHT is shown in Figure 3.6.

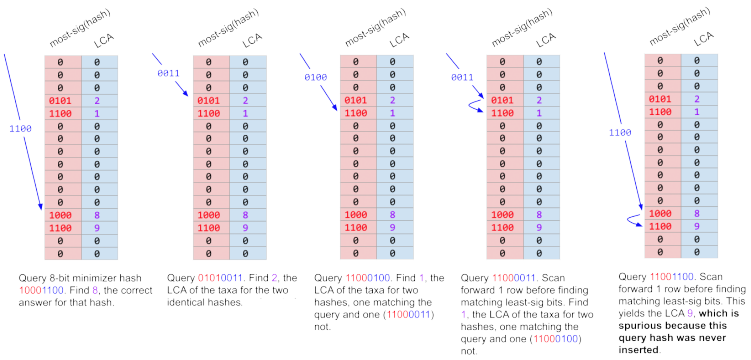

Figure 3.6: Sequential examples of Kraken 2's querying of a compact hash table with a minimizer [57].

After performing the sequence analysis, its LCA and their ancestors (in a taxonomy tree shape) form what the authors of [58] call a *classification tree*. This is a pruned tree used to classify S. In fact, this tree contains all the nodes, and their ancestors, found in the sequence classification. The classification tree nodes are weighted with the number of minimizers (therefore k-mers) that are mapped to that LCA node. Subsequently, a score for each Root-To-Leaf (RTL) path is computed by summing all the weights of the nodes in the selected path. The path with the highest score is called by the authors of [58] the *classification path*. To sequence S is assigned the LCA of the deepest node in the classification path. If there is more than one leaf, then their LCA is chosen. An example of the sequence classification algorithm is shown in Figure 3.7.

If the Kraken 2's database is reduced by the user during the database building phase, only the minimizers with hash code smaller or equal than a maximum allowable hash code and the minimizers from k-mers with not ambiguous nucleotide code are searched in the CHT.

3.3 PROTEIN CASE

In order to classify protein sequences, firstly Kraken 2X builds the protein database in the same manner that Kraken 2 does for nucleotide (see Section 3.1). The only differences are the masking tool used (segmasker) and the alphabet used to represent the basic unit (i. e., amino acids). In fact, protein alphabet uses 20 characters to repre-

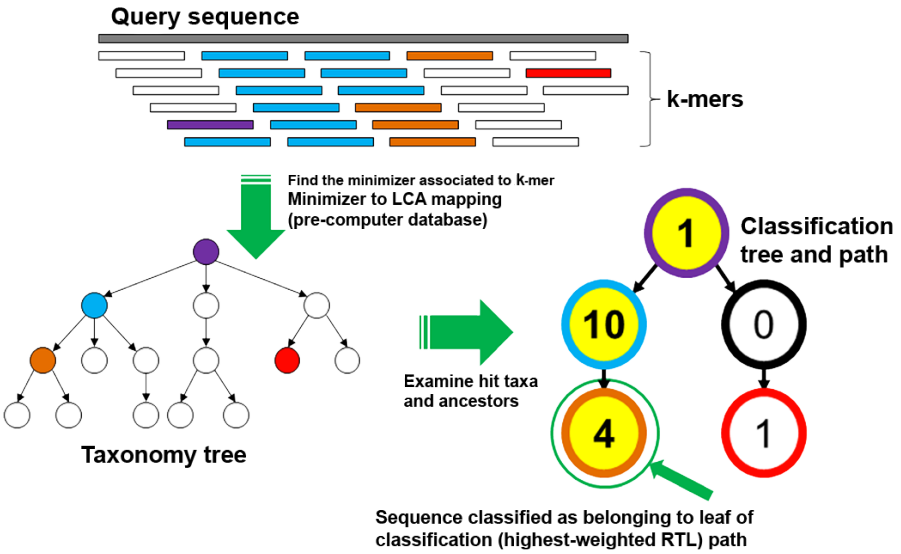

Figure 3.7: The Kraken 2 sequence classification algorithm. (Based on figure in [58].)

sent the amino acids. Kraken 2X reduces this alphabet to 16 using the 15-character alphabet of Solis [68] plus a single additional value, to represent the selenocysteine and pyrrolysine and a translation termination (stop codons). The minimizer computation is the same used for the nucleotide sequence; the only differences are the not computation of the reverse complements and the default parameters are k=15, l=12 and s=0.

When searching against a protein minimizer database, Kraken 2X translates all six reading frames of the input query DNA sequences into the reduced amino acid alphabet. Minimizers from all six frames are pooled and used to query the CHT, and therefore all contribute to the Kraken 2X classification of a query sequence.

4

IMPLEMENTATION DETAILS

The objective of this work is improving the reads classification. This goal is obtained by trying to increasing the number of classified reads and incrementing the number of reads classified to species level. To achieve these objectives, Kraken 2 was equipped with memory on the results of the previous classifications. To obtain this memory effect a data structure was added to Kraken 2 so it can store which taxonomy ID is associated to minimizers not present in its database. From here on Kraken 2 with the additional data structure is called *Kraken 2 plus*. This additional data structure is a simple not ordered map that stores a minimizer-taxonomy ID pair. This map is implemented with the `unordered_map` class, member of the `Container` C++ library. An unordered map is an associative container that contains key-value pairs with unique key. The choice of this structure is due to the fact that search, insertion and removal of elements have average constant-time complexity [69]. Internally, the elements are not ordered in any particular order, but are organized in buckets. Which bucket an element is placed into depends entirely on the hash of its key. This allows a fast access to the single element, once the hash is computed, it refers to the exact bucket where the element has been inserted.

The additional map is managed with the `AdditionalMap` class, defined in the `additional_map.h` file and implemented in the `additional_map.cc` file. The class contains the data member `ump`, the unordered map that stores the minimizer-taxonomy ID pair. The key and the value are 64 bit unsigned integer (`uint64_t`). This choice was made to keep all the information obtained during the classification.

The `AdditionalMap` class contains the following function members:

- `void ReadFile(const char *filename).`
  This method is used to populate the additional map with the data contained in the `filename` file given in input. This file stores, for each line, the key-value pairs contained in the additional map spaced by a tab character. The method reads each line in the file and the elements in the line is saved in the additional map using the private method `Add(uint64_t minimizer, taxid_t tax_id).`

- `void AddPair(uint64_t minimizer, taxid_t tax_id, Taxonomy & taxonomy).`
  This method is used to add a key-value pair to the additional map. The inputs of the method are the minimizer not present in the Kraken 2's database and not present in the additional map,

the taxonomy ID obtained after the read classification and the Kraken 2's taxonomy generated in the database building phase. When the method is called, firstly, it verifies if the minimizer is in the additional map or not. If is present, the associated taxonomy ID is returned and the LCA of the returned taxonomy ID and the input taxonomy ID (`tax_id`) is computed. After that, the taxonomy ID of the minimizer `minimizer` is updated with the computed LCA. Instead, if the minimizer `minimizer` is not in the additional map the `minimizer-tax_id` pair is added to the map.

- `taxid_t GetTax(uint64_t minimizer)`.
  This method returns the taxonomy ID associated to the minimizer `minimizer`. To do this, the minimizer is searched in the additional map. If it is present the taxonomy ID value is returned otherwise, gives back zero.

- `size_t GetSize()`.
  This method returns the number of pairs in the additional map.

- `void WriteMap(const char *filename)`.
  This method writes the additional map content to the file `filename`. Firstly, the method verifies the map size. If it is empty, the method do nothing. Otherwise, for each pair between the key and value a tabular character is added and after the pair a new line character is added.

- `bool IsEmpty()`.
  The method verifies if the additional map does not have pairs saved. If is the case, the method returns true. Otherwise, it returns false.

- `void Add(uint64_t minimizer, taxid_t tax_id)`.
  This *private* method adds the `minimizer-tax_id` pair to the additional map. To do this, the `emplace` `unordered_map`'s member function is used. It was chosen because it automatically creates and adds to the map the key-value pair if the key is not present in the map.

The additional map file is saved in the Kraken 2's *taxonomy* directory in the *add_hash.k2d* file.

For run Kraken 2 plus one must use the script `kraken2-plus`. This script is very similar to `kraken2` script with some additional options to allow the user a variety of use combinations according to his needs. The additional options are the following:

- `--build-new-map`.
  This option allows the user to remove the old additional map file, if created before, and creates a new empty additional map.

If *max-iteration* is greater than one, the map is maintained and populated during the subsequent classifications. If the option is not used the additional map file is maintained and, before starting the population of the map with the new read file, Kraken 2 plus reads the additional map file in the taxonomy directory. The data contained in the file is used by Kraken 2 plus during the population of the map or during the read classification.

- `--max-iteration`.
  This option allows the user to choose the number of times Kraken 2 plus does the classification for populate the additional map. The default value is 1 (i. e., when the option is not used).

- `--disable-additional-map`.
  This option allows the user to not update/create the additional map for classify the read file. In case the additional map is not created before, Kraken 2 plus classifies only with its database, so it does the classification that does `kraken2` script.

- `--disable-classification`.
  This option allows the user to create and populate the additional map or update the map, if a file exists, without execute the last classification of the read file. The last classification only uses the database and the additional map, without update the last data structure.

Once the user runs `kraken2-plus` script with the correct options, the additional map population starts. An empty unordered map is initialized and, if the file *add_hash.k2d* exists and is not empty, the map is populated with the data containing in the *add_hash.k2d* file using the method `ReadFile`. Otherwise, the map remains empty. After the initialization of the additional map, the classification for populate the map starts. Figure 4.1 shows the additional map population steps. During the read classification its minimizers are computed one at a time and, for each of them, the CHT is queried (1). If it returns a taxonomy value equal to zero, then the additional map is queried if it is not empty (2). If the additional map is empty, this means that the minimizer is not in the Kraken 2's database and no taxonomy ID has been assigned to it or is the first time the minimizer is found. In that case, the minimizer is added to a temporary list of not taxonomy assigned minimizers (3). Instead, if the database or additional map querying returns a taxonomy ID not equal to zero, then the taxonomy ID count is updated (4). Once the read is classified, then obtained the deeper LCA (5) in the classification path (as explained in Section 3.2), is verified if it is classified at species level or below (6). If it is, then the minimizers in the list are added to the additional map using the `AddPair` method with key the minimizer and value the taxonomy ID obtained by the read classification. If the minimizer is in the additional

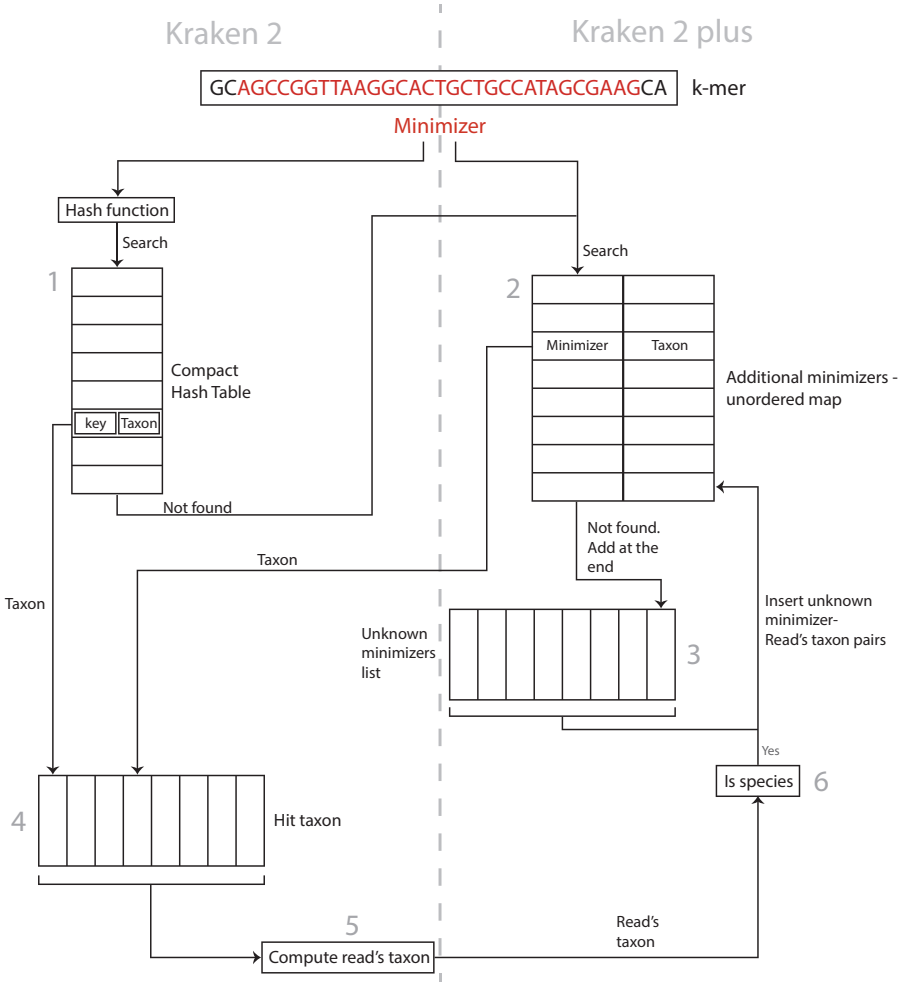

Figure 4.1: Schema of the additional map population steps.

map the LCA of the input and stored taxonomy IDs value is saved. Instead, if the taxonomy ID obtained after the read classification is in a level above the species the minimizers are not added and the list is emptied.

This procedure is repeated for all the reads in the input file. Once all the reads are classified, the additional map content is saved in the *add_hash.k2d* file using the `WriteMap` method.

Once the population of the additional map ends, another classification starts. This classification only uses the Kraken 2's database and the additional map to classify the reads in the same input file but without updates the additional map content. After this classification, `kraken2-plus` ends and generates the same output files as `kraken2` script.

5 EXPERIMENTS AND RESULTS

In this chapter the performance of Kraken 2 plus is analyzed. Firstly, there is a description of the strain exclusion experiment, the datasets and the accuracy measures used to evaluate Kraken 2 plus. Follows an analysis of the obtained results.

5.1 STRAIN EXCLUSION EXPERIMENT

The strain exclusion experiment's data were generated as done by the Kraken 2's author in [57]. Specifically, the generation of these data occurs in the way explained below.
It stars by downloading the reference genomes and the taxonomy from the NCBI's database (October 2019 release). The reference genomes are generated from the archaeal, bacteria and viruses genomes, downloaded from the `ftp://ftp.ncbi.nlm.nih.gov/genomes/archive/old_refseq/Bacteria/all.fna.tar.gz` and `ftp://ftp.ncbi.nlm.nih.gov/genomes/Viruses/all.fna.tar.gz` URLs respectively. From the NCBI's taxonomy the *taxdump.tar.gz* and the *gi_taxid_nucl.tar.gz* files are downloaded. These files contain the taxonomy tree nodes data and the GenBank identifier of nucleotide record to taxonomy ID association respectively. Obtained the taxonomy data, for each NCBI's library (bacteria and viruses) a set containing all the nucleotide and taxonomy data is generated from the complete genomes, excluding plasmids and the 2nd/3rd chromosomes, so to have a list containing only one entry for each genome. From this set a subset is builded that contains both two sister sub-species taxa and two sister species taxa present in the set of reference genome. The subset is ordered by genus, then by species and lastly by strain taxonomy ID. From this subset the first n elements are extracted, $n = 40$ for the bacteria and $n = 10$ for the viruses. These elements will be the origin strains for the strain exclusion experiment. The selected bacteria and viruses origin strains are listed in Table A.1 and Table A.2 respectively. At this point a reference genomes set is created taking all the data downloaded from the NCBI and removing the origin strains previously chosen. Once the reference genomes set is created, Mason 2 [70] is used for simulate 100 bps paired-end Illumina sequence data from the origin strains, with a number of simulated fragments for each strain (f) chosen by the user. Specifically, the Mason 2's *mason_simulator* command is used with default options for the simulation of the sequence's errors. That means that the generated sequences contain a error rate of 0.4% mismatches, 0.005% insertions and 0.005% deletions. The reads obtained from the

origin strains are concatenated in a single file and the truth file of the simulated reads is generated for each downloaded library.

5.2 DATASETS

Different metagenomic datasets are used to compare and evaluate Kraken 2 plus's accuracy.

10 datasets were created using the origin strains obtained from the strain exclusion experiment explained in Section 5.1. Of these datasets, 7 are builded by varying the number of contained reads; precisely 50000, 3125000, 6250000, 12500000, 25000000, 50000000 and 100000000. The other 3 have the same number of reads (100000000) but the mismatch error rate varies; precisely 2%, 5% and 10% are the error rates chosen.

In addition to the datasets above, 74 real read datasets is created using data from the NCBI's Sequence Read Archive (SRA). To build the datasets, firstly the *SraRunInfo.csv* file is downloaded from the NCBI's site. To obtain this file the following query is used in the search bar of the site: *("2015/1/1"[PDAT] : "2015/9/23"[PDAT]) AND ("Bacteria"[Organism] OR "Bacteria"[Organism] OR "bacteria"[All Fields]) AND ("biomol dna"[Properties] AND "strategy wgs"[Properties])*. On the result page the SRA link is followed and the list of items is returned. From this page the *Send to:* menu's *File* option is selected. Subsequently, the *RunInfo* option in the drop down list is selected and the *Create File* button is clicked. At the end of these steps the *SraRunInfo.csv* file is created and downloaded. Once the download finishes, 74 SRA file names are selected from the Centrifuge [71] experiment. A script downloads the files from the NCBI's SRA database and with the `fasterq-dump` tool, from the NCBI's SRA Toolkit, the FASTQ data from the SRA-accessions are extracted. When the data extraction is finished, a truth file for each SRA dataset is generated. After the truth files creation, the datasets generation process finishes. The selected real datasets are listed in Table A.3.

5.3 EVALUATION MEASURES

The measures used to comparing and evaluating Kraken 2 plus with the other tools are: the sensitivity, the PPV, the F-measure and the PCC. To compute these measures, firstly the number of reads that belong to the following categories must be counted. These categories are:

- True Positive (TP).
- False Negative (FN).
- Vague Positive (VP).
- False Positive (FP).

To compute such numbers the user must has the truth file of the dataset to classify, the classifier's result and the taxonomy rank to which the user want to conduct the evaluation . Once the user has these data the count of the number of reads in each category can be done.

A read belongs to the TP category if its taxonomy classification rank is the same or is a descendant of the truth rank. A read belongs to the FN category if the classifier fails to classify the sequence. A read belongs to the VP if its taxonomy classification rank is an ancestor of the truth rank. Lastly, a read belongs to FP if its classification is incorrect; that is, it is not in the true taxonomy of origin, it is not an ancestor or a descendant of the truth rank.

Once these values are computed, the evaluation measures mentioned above can be calculated as explained below.

The *sensitivity* or *recall* is computed as the proportion of the number of reads correctly classified among the total number of reads classified.

$$\begin{aligned} sensitivity &= \frac{number\ of\ reads\ correctly\ classified}{number\ of\ reads\ classified} \\ &= \frac{TP}{TP + VP + FN + FP}. \end{aligned} \tag{5.1}$$

The *Positive Predictive Value (PPV)* or *precision* is computed as the proportion of the number of reads correctly classified among the number of positive calls.

$$\begin{aligned} PPV &= \frac{number\ of\ reads\ correctly\ classified}{number\ of\ positive\ calls} \\ &= \frac{TP}{TP + FP}. \end{aligned} \tag{5.2}$$

In the PPV computation the VP reads are excluded, since it is not certain to which truth taxonomy ID the taxonomy ID given by the classifier belongs.

The *F-measure* is computed as the harmonic mean of sensitivity and PPV.

$$\begin{aligned} F1 &= \frac{2 \cdot sensitivity \cdot PPV}{sensitivity + PPV} \\ &= \frac{2 \cdot TP}{2 \cdot TP + VP + FN + 2 \cdot FP}. \end{aligned} \tag{5.3}$$

The *Pearson Correlation Coefficient (PCC)* measures the linear correlation between two variables X and Y. This measure returns values between -1 and 1. Where 1 is total positive linear correlation, 0 is no linear correlation and -1 is total negative linear correlation. In metagenomic is used to evaluate the species abundance. The PCC is computed in the following manner:

Given paired data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ consisting of n pairs, PCC is defined as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x}) \cdot (y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}. \tag{5.4}$$

Where: n is the sample size, x_i, y_i are the individual sample points indexed with i and $\overline{x}$ and $\overline{y}$ is the sample mean of x and y respectively. The sample mean of x (analogously of y) is computed as: $\overline{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$. Rearranging Equation 5.4 provides the single-pass algorithm formula for calculating the PCC:

$$r_{xy} = \frac{\sum_{i=1}^{n} x_i \cdot y_i - n \cdot \overline{x} \cdot \overline{y}}{\sqrt{\sum_{i=1}^{n} x_i^2 - n \cdot \overline{x}^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2 - n \cdot \overline{y}^2}}. \tag{5.5}$$

Where: n, x_i, y_i, $\overline{x}$, $\overline{y}$ are defined as above.
In this work the (x_i, y_i) pair contains the classifier's taxonomy ID assigned and the truth taxonomic ID of sequence i respectively.

5.4 RESULTS

Kraken 2 plus is compared with the tools briefly described in Table 5.1. For each tool its database is builded using the strain exclusion

Table 5.1: Metagenomic classifiers used for the strain exclusion experiment.

TOOL	BRIEF DESCRIPTION
Centrifuge [71]	Taxonomic classifier using database compressed with BWT and FM index.
CLARK [72]	Taxonomic classifier using in-memory k-mer search of metagenomic reads against a specific taxonomy level database built from completed genomes.
Kraken [58]	Taxonomic classifier using in-memory k-mer search of metagenomic reads against a database built from multiple genomes.
Kraken 2 [57]	Taxonomic classifier using in-memory minimizer search of metagenomics reads against a database built from multiple genomes.
KrakenUniq [73]	Taxonomic classifier using in-memory unique k-mer search of metagenomic reads against a database built from multiple genomes.

genomes without the origin strains. The classifiers are run in the Blade Computing Cluster using 16 threads. The tools are compared using the evaluation measures explained in Section 5.3 and are splitted in different groups as to better understand the performance without overcrowding the graph. The analysis starts with a specific case where Kraken 2 plus's performance is studied at genus and species level for bacteria and only species level for viruses. After, is analyzed the performance as the number of reads in the dataset changes. Follow the performance analysis as the mismatch errors rate varies for the 100000000 reads dataset. This method of analysis was chosen because the behavior of Kraken 2 plus and the other tools is very similar for all the cases studied and to not overcrowd the results section. Finally, the performance of Kraken 2 plus is analyzed with the real datasets described in Section 5.2. To not overcrowd the graphics and the results table, the evaluation measures means was computed and Kraken 2 plus is compared with Kraken and Kraken 2. All the evaluation's graphics and tables can be found in Section A.4.

READS FROM ORIGIN STRAINS As seen in Figure 5.1, at genus level Kraken 2 plus with bacteria gets a sensitivity improvement of at least 1 percentage point (pp) respect to the best of other tools (Kraken 2). This sensitivity improvement has led to a worsening of the PPV

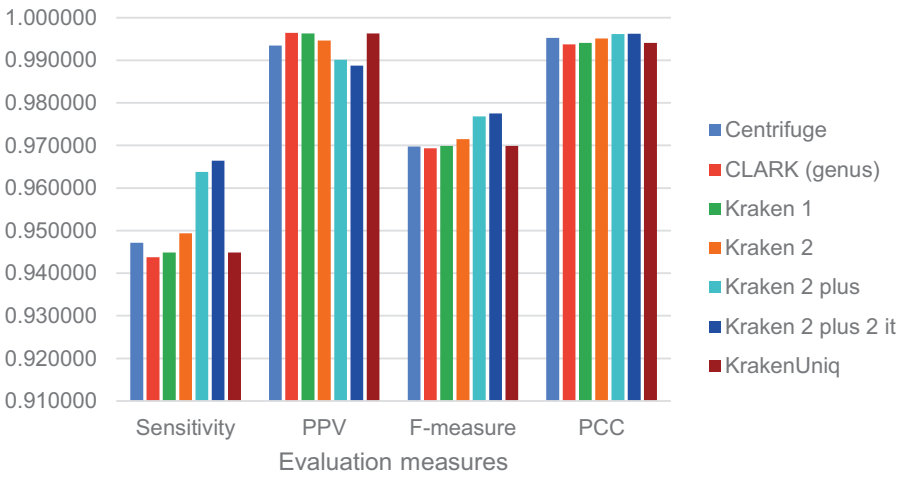

Figure 5.1: Bacteria evaluation at genus level on the 6250000 reads dataset.

of at least 0.3 pps respect to the worst tool (Centrifuge). Despite this worsening of the PPV, Kraken 2 plus has the best F-measure values with an increment of at least 0.5 pps respect to Kraken 2. Moreover, Kraken 2 plus obtains the best result in genus level abundance (PCC) with an improvement of at least 0.1 pps respect to Centrifuge.

As seen in Figure 5.2, at species level Kraken 2 plus with bacteria gets the best results only in sensitivity with improvement of at least 1 pp. While for all the other evaluation measures Kraken 2 plus gets worst results respect to the other tools, in particular Centrifuge. This

worsening at species level can be due to the fact that the new minimizers (not present in the Kraken 2's database) found are classified at genus level or above.

From Figure 5.1 and Figure 5.2 we note that Kraken 2 plus gets better

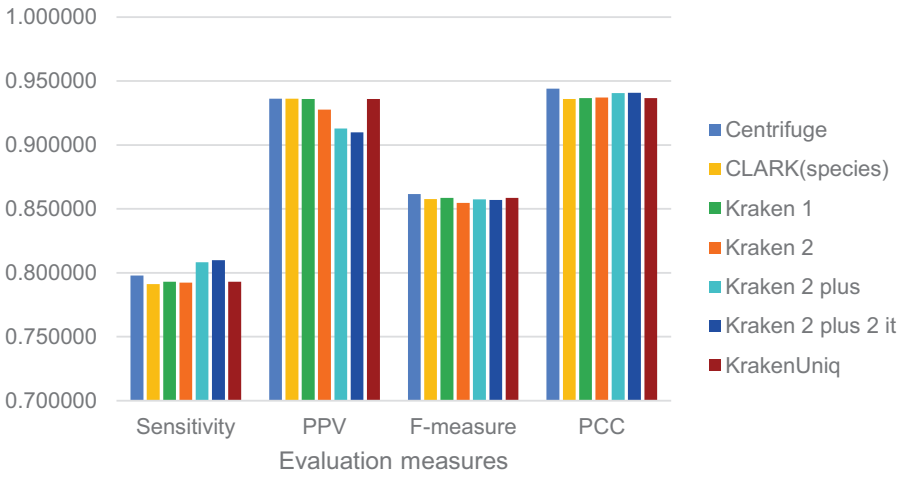

Figure 5.2: Bacteria evaluation at species level on the 6250000 reads dataset.

results than Kraken 2 in all the evaluation measures except for the PPV.

For knowledge of the reader, Kraken 2 plus's performance are analyzed for viruses at species level. As seen in Figure 5.3, Kraken 2 plus gets the best performance in all the evaluation measures; with noticeable improvement of sensitivity, F-measure and PCC and a slightly improvement of PPV. In summary, Kraken 2 plus gets excellent results

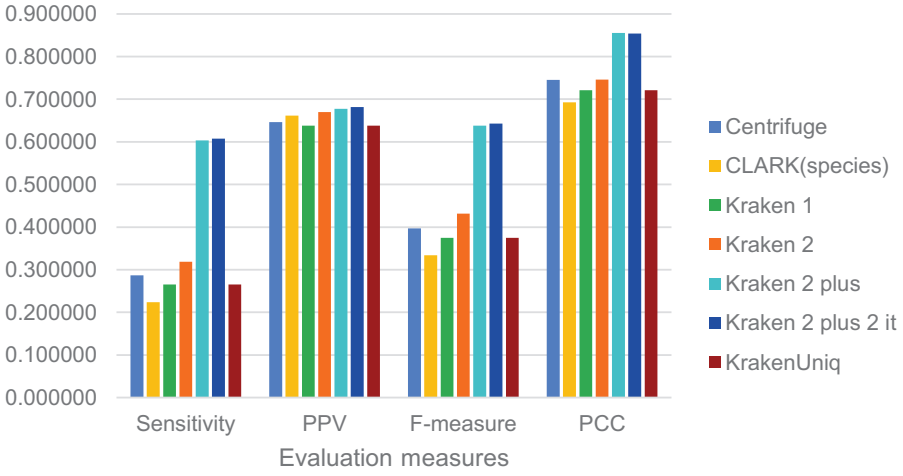

Figure 5.3: Viruses evaluation at species level on the 6250000 reads dataset.

with the viruses improving all the evaluation measures and gets good improvements with the bacteria except for the PPV.

The Kraken 2 plus's results, at genus level, obtained with bacteria varying the number of reads in the dataset are now analyzed. As seen

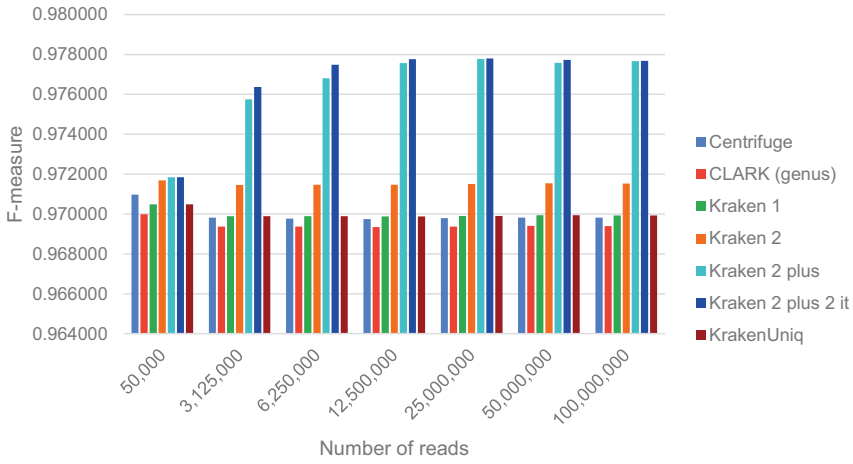

Figure 5.4: Bacteria F-measure evaluation at genus level.

in Figure 5.4, Kraken 2 plus gets better F-measure values than the other tools as the number of reads increases; obtaining improvements up to almost 1 pp. This improvement is given mainly from the increasing of the sensitivity (Figure A.2).

Regarding the PCC, as seen in Figure 5.5, Kraken 2 plus gets the best results for all the datasets, except for the 50000 reads dataset where Centrifuge goes better, with improvement of at least 0.1 pps. It can be

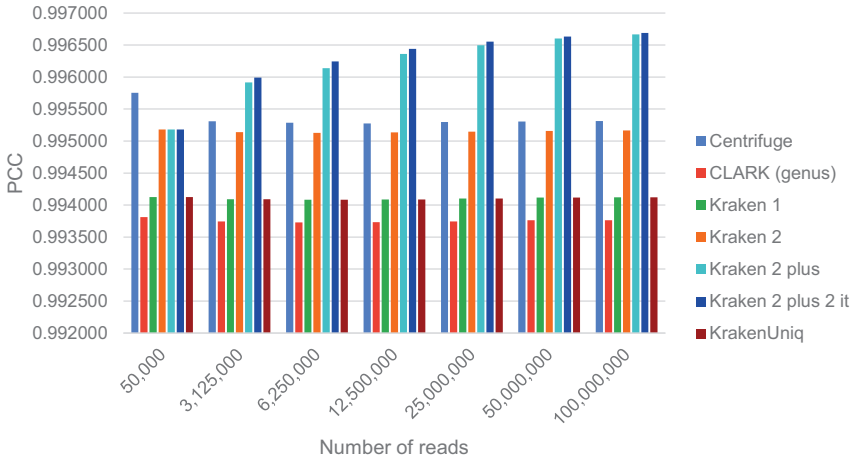

Figure 5.5: Bacteria PCC evaluation at genus level.

guessed that the greater the amount of data that Kraken 2 plus has available the better its classification results.

The results obtained by Kraken 2 plus with the 100000000 reads dataset as the mismatch errors rate changes are now analyzed. The default mismatch error rate values are added to have a point of comparison. As seen in Figure 5.6, Kraken 2 plus gets the best F-measure values in all the cases. We note a gap increasing (to 25 pps) between

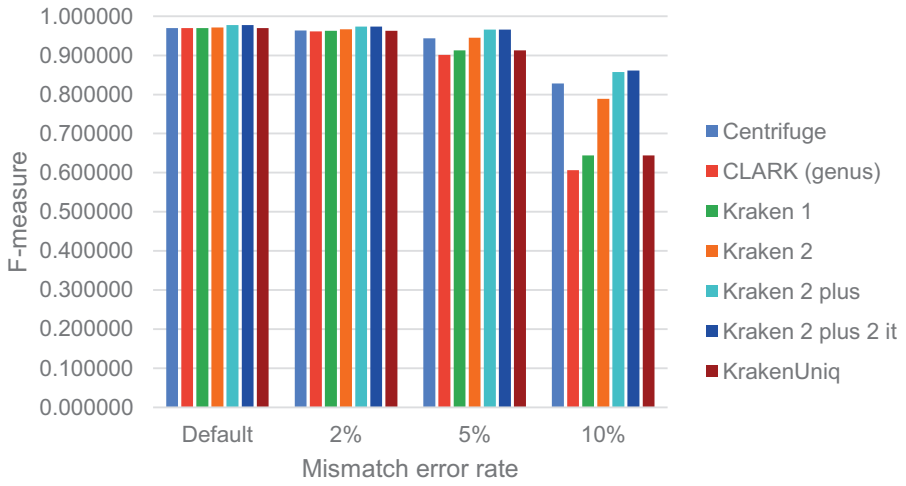

Figure 5.6: Bacteria F-measure evaluation at genus level with mismatch errors.

Kraken 2 plus and the other tools as the mismatch errors rate increases. The same applies to the PCC, as can be seen in Figure 5.7, where there is a smaller gap increasing than the F-measure (to 2.4 pp). As expected,

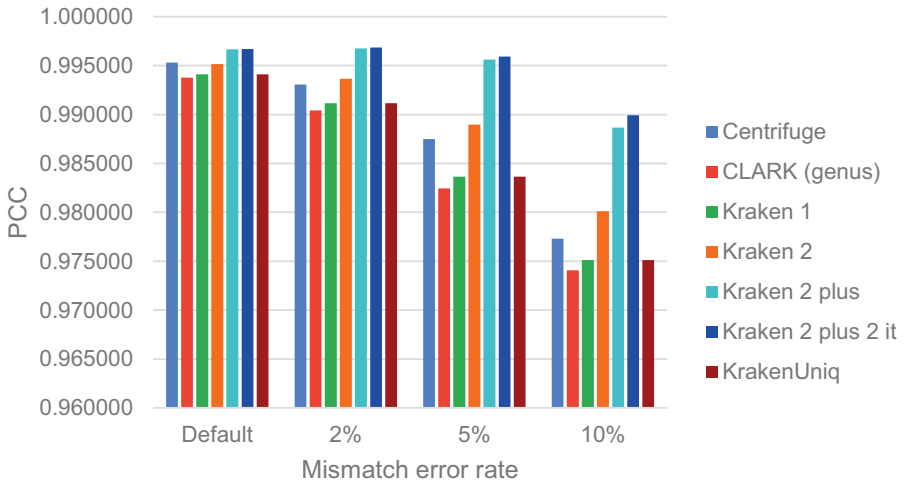

Figure 5.7: Bacteria PCC evaluation at genus level with mismatch errors.

the change in the mismatch error rate leads to a worsening in all tools performance. However, Kraken 2 plus is the tool that has less suffered from the presence of errors in the dataset.

REAL DATASETS Kraken 2 plus was tested with the real datasets generated as explaned in Section 5.2. As seen in Figure 5.8 and Table 5.2 Kraken 2 plus gets the best F-measure value with an improvement of about 0.1 pp. This improvement is due to the increase in sensitivity of about 0.3 pp. Also for these datasets Kraken 2 plus gets the worst PPV, with a decrease of about 0.1 pp.

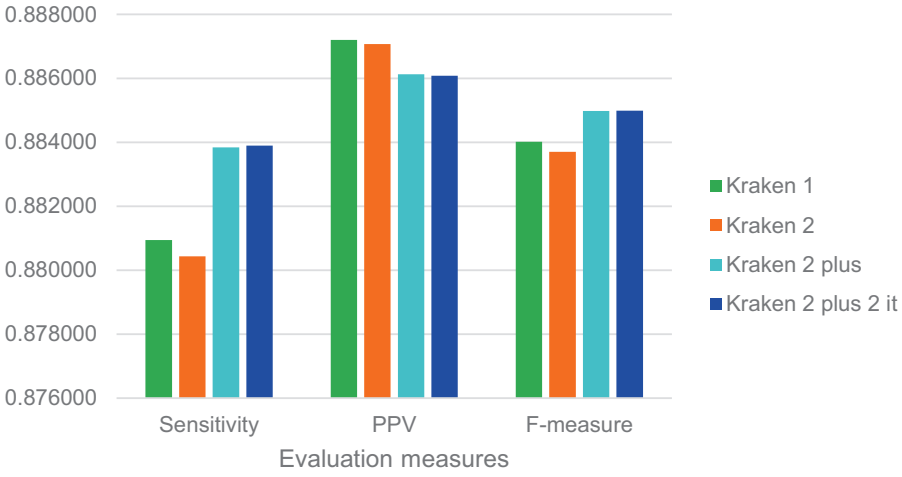

Figure 5.8: Real datasets evaluation measures means at genus level.

Table 5.2: Evaluation measures means at genus level on real datasets.

TOOL	SENSITIVITY	PPV	F-MEASURE
Kraken 1	0.880941	0.887200	0.884017
Kraken 2	0.880434	0.887072	0.883705
Kraken 2 plus	0.883841	0.886128	0.884982
Kraken 2 plus 2 it	0.883899	0.886080	0.884988

The same applies to the species level, as can be seen in Figure 5.9 and Table 5.3.

Table 5.3: Evaluation measures means at species level on real datasets.

TOOL	SENSITIVITY	PPV	F-MEASURE
Kraken 1	0.866725	0.874150	0.870388
Kraken 2	0.866397	0.874088	0.870203
Kraken 2 plus	0.869713	0.873062	0.871383
Kraken 2 plus 2 it	0.869796	0.873010	0.871398

It is noted that, with the real datasets, the tools performance are lower than those with datasets generated from the origin strains. This is due to the fact that with 10% of the datasets Kraken 2 plus

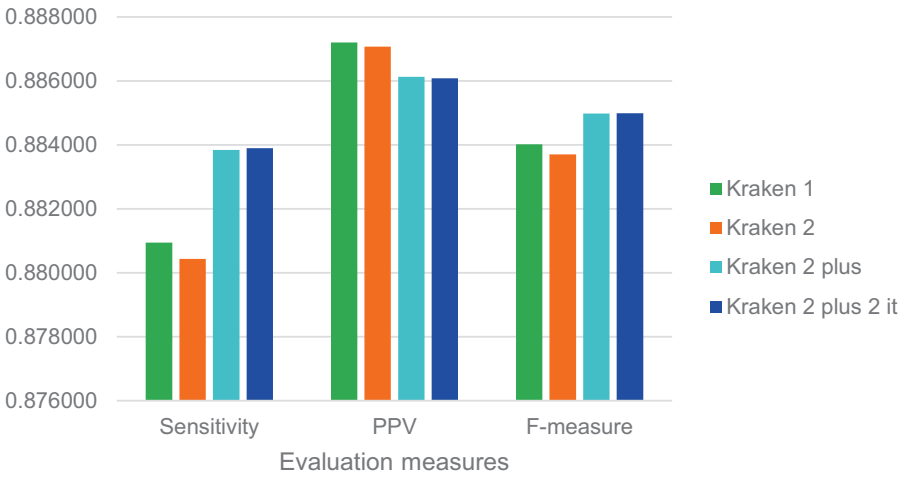

Figure 5.9: Real datasets evaluation measures means at species level.

and the other tools correctly classify only a small amount of reads. Probably due to the fact that the reads contained in these datasets are very different from the sequences used to build the databases or the taxonomy ID is not present in the databases. Despite this, Kraken 2 plus gets classification improvements respect to Kraken and Kraken 2, as can be seen in Table 5.4.

Table 5.4: Evaluation measures means at genus level on ERR915393 real dataset.

TOOL	SENSITIVITY	PPV	F-MEASURE
Kraken 1	0.968126	0.990403	0.979138
Kraken 2	0.970141	0.994990	0.982408
Kraken 2 plus	0.992824	0.993408	0.993116
Kraken 2 plus 2 it	0.993064	0.993642	0.993353

EXECUTION TIME AND MEMORY USAGE The execution time and memory usage of each tool during the datasets classification are now analyzed. For the analyzed cases that use the datasets generated from the origin strains the execution time and memory usage means are computed to not overcrowding the graphics.
Regarding the execution time, as expected, Kraken 2 plus has an increase in classification time of at least double respect to Kraken 2 with all analyzed datasets as can be seen in Figure 5.10 and Figure 5.11. If the reader is interested in the specific data of each case analyzed, see Figure A.30, Figure A.31 and Figure A.32. This increment is due to the population of the additional map.

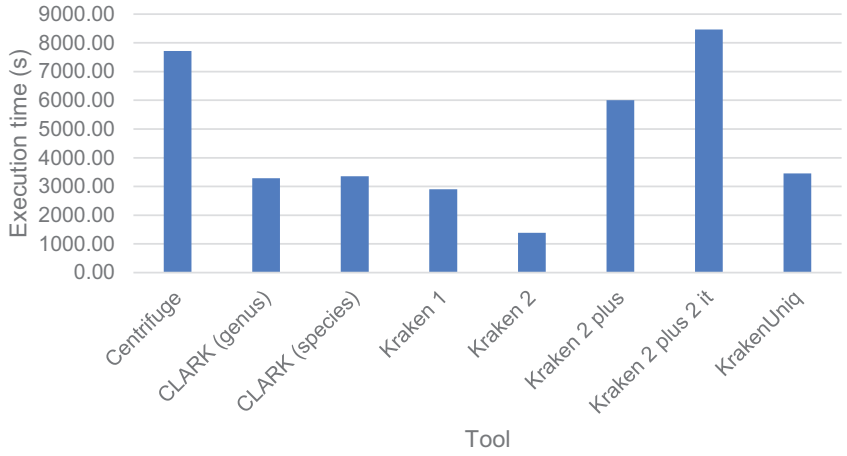

Figure 5.10: Tools execution time means on datasets obtained from the origin strains.

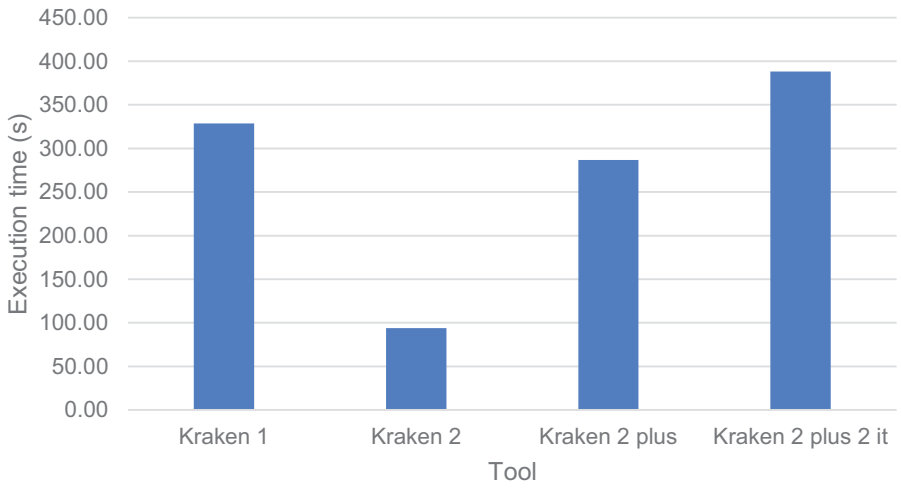

Figure 5.11: Kraken tools execution time means on real datasets.

About memory usage, as can be seen in Figure 5.12 and Figure 5.13, Kraken 2 plus uses more memory than Kraken 2, as expected, due to the fact that a new map is added.

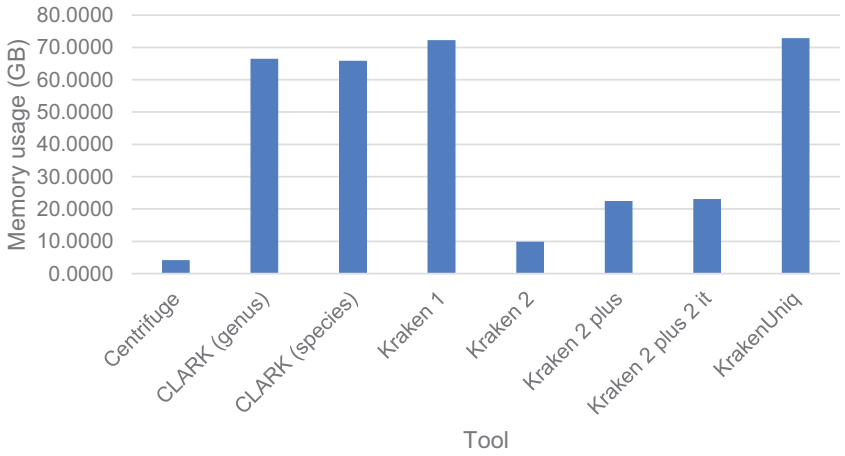

Figure 5.12: Tools memory usage means on datasets obtained from the origin strains.

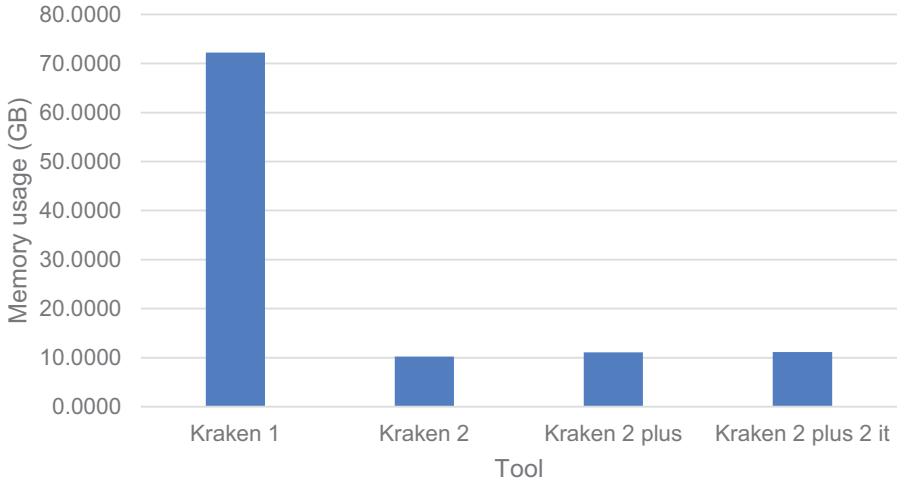

Figure 5.13: Kraken tools memory usage means on real datasets.

The difference in memory usage is strongly affected by the number of unknown minimizers found by Kraken 2 plus during the population of the additional map If the reader is interested in the specific data of each case analyzed, see Figure A.33, Figure A.34 and Figure A.35.

6 CONCLUSIONS

In this thesis work the improvement of metagenomic classification by boosting the reference *k*-mers was analyzed and a resolution approach was studied, through the use of an additional map that equips Kraken 2 with memory from previous classifications. The additional map was implemented and the new tool was tested, together with other tools, executing one or two classification for populate the map. This is done for the purpose to see if the tool returns better results than the other tools and if increasing the number of classifications to populate the map improve a lot or not the final classification.

In general, the proposed solution returns good results for the sensitivity, F-measure and PCC in most of the analyzed datasets. Conversely, this solution returns a worsening in PPVs, probably caused by the increase in the number of classified reads. In fact, is not guaranteed that with an increment of the classified reads there will also be a correct assignments. With the proposed solution it may happen that a read assignment moves to a higher taxonomy level or a propagation of classification errors. Even these facts can reduce the PPV.

The obtained results show that a further classification to populate the additional map does not lead to great improvements in the final classification given the execution times used. Therefore, is sufficient only one classification to populate the additional map.

As possible future developments one could try to increasing PPVs at genus and species level (e. g., using unique *k*-mers), improving the population algorithm to obtain a reduction in the classification time and try to use other data structure (e. g., counting quotient filter [74]) to decrease the tool's memory usage.

APPENDIX

The appendix contains a brief introduction to the use of the Blade Computing Cluster at the Department of Information Engineering and the tables containing the genomes used for the strain exclusion experiment and all the test results obtained.

A.1 RUNNING PROGRAMS ON THE BLADE COMPUTING CLUSTER

As the data quantity and the amount of memory required for the metagenomic classification is a lot, the use of the Blade Computing Cluster is necessary. Blade uses the Sun Grid Engine to manage the queue. This is a queueing system that allows one to run a job according to the requirements specified (it reads the requirements and then queues it based on the priority given by the requirements themselves).

BASIC COMMANDS AND FILE TRANSFER In order to connect to the computing cluster one needs to use `ssh`. This is something that is provided with every major Linux distribution:

```
$ ssh username@login.dei.unipd.it
```

Running this command will then prompt the input of the password and allow to start an `ssh` communication with the server. Here it is possible to access the space on the server's machine and issue commands like the ones required to queue jobs or compile the code. To transfer files from the local machine to the cluster there are a number of alternatives but are used mainly two of them. The first one is the `scp` command, which can be used to transfer files between any two hosts via `ssh`, but is use it to upload files from the local machine. To achieve this, the basic syntax of the command is:

```
$ scp [ options ] source_dir/source_filename username@login.dei.
    unipd.it_host:directory/filename
```

An alternative is using FileZilla which provides an easy-to-use GUI, as shown in Figure A.1. Installation is straightforward by using:

```
$ sudo apt-get install filezilla
```

The interface is self explanatory, there are four fields at the top which are filled with the same information used for the `ssh` command, the only thing to note is that in the Host field it should be specified that we are trying to establish an SFTP connection with the server, by writing: `sftp://login.dei.unipd.it`.

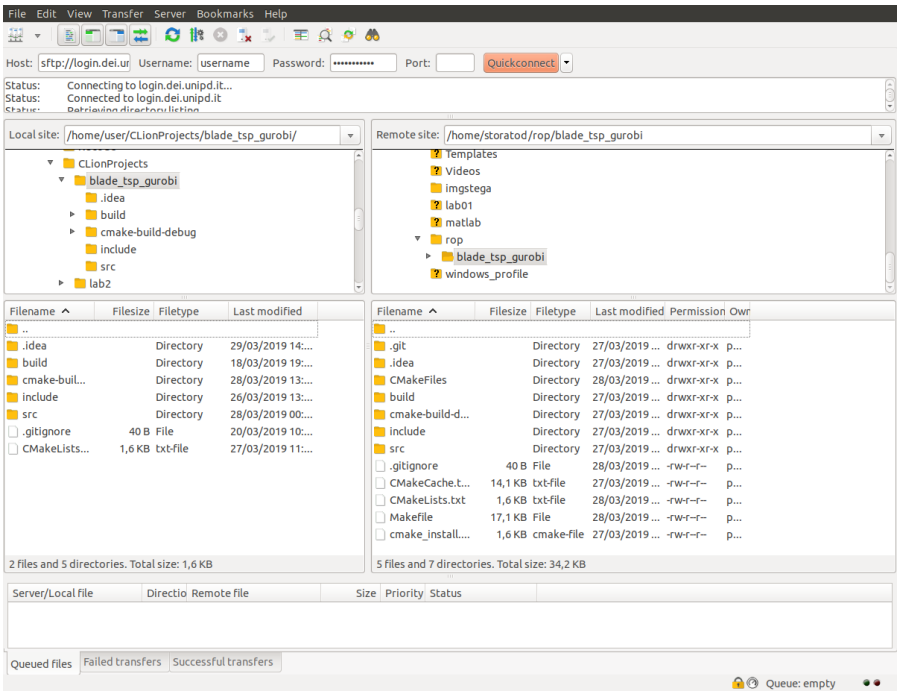

Figure A.1: Interface of FileZilla.

A.2 GENOMES USED FOR THE STRAIN EXCLUSION EXPERIMENT

The tables below (Table A.1 and Table A.2) list the bacteria and viruses genomes used as origin strains for the generation of the simulated reads with Mason 2. These strains are obtained during the strain exclusion data generation explained in Section 5.2.

Table A.1: Bacteria genomes used as origin stains in the strain exclusion experiment.

TAX ID	SCIENTIFIC NAME
706191	Pantoea ananatis LMG 20103
698969	Corynebacterium diphtheriae HC03
1105098	Rickettsia prowazekii str. GvV257
300852	Thermus thermophilus HB8
759913	Streptococcus dysgalactiae subsp. equisimilis AC-2713
401614	Francisella tularensis subsp. novicida U112
272559	Bacteroides fragilis NCTC 9343
1096995	Acinetobacter baumannii BJAB07104
882096	Listeria monocytogenes SLCC5850
863638	Clostridium acetobutylicum EA 2018

Continued on next page

Table A.1 – *Continued from previous page*

TAX ID	SCIENTIFIC NAME
366649	Xanthomonas citri pv. fuscans
1117943	Sinorhizobium fredii HH103
1173064	Anaplasma phagocytophilum str. JM
354242	Campylobacter jejuni subsp. jejuni 81-176
1161918	Brachyspira pilosicoli WesB
1244085	Klebsiella pneumoniae CG43
936153	Enterococcus faecalis 62
591020	Shigella flexneri 2002017
243276	Treponema pallidum subsp. pallidum str. Nichols
374930	Haemophilus influenzae PittEE
1042876	Pseudomonas putida S16
395492	Rhizobium leguminosarum bv. trifolii WSM2304
909420	Neisseria meningitidis H44/76
1392476	Staphylococcus aureus subsp. aureus 6850
257310	Bordetella bronchiseptica RB50
336982	Mycobacterium tuberculosis F11
644042	Lactobacillus plantarum JDM1
138677	Chlamydia pneumoniae J138
402882	Shewanella baltica OS185
634997	Mycoplasma hyorhinis DBS 1050
1053692	Methanococcus maripaludis X1
224326	Borreliella burgdorferi B31
592021	Bacillus anthracis str. A0248
573059	Desulfovibrio vulgaris RCH1
1116391	Paenibacillus mucilaginosus 3016
434271	Actinobacillus pleuropneumoniae serovar 3 str. JL03
956149	Cronobacter sakazakii SP291
290847	Helicobacter pylori 51
386656	Yersinia pestis Pestoides F
1300259	Alteromonas mediterranea UM4b

Table A.2: Viruses genomes used as origin stains in the strain exclusion experiment.

TAX ID	SCIENTIFIC NAME
1070413	Human papillomavirus 140
41856	Hepatitis C virus genotype 1
1087109	Canis familiaris papillomavirus 10
981431	Pseudomonas phage PAK_P3
1156769	Porcine kobuvirus
57579	Adeno-associated virus - 4
12524	Junonia coenia densovirus
11801	Moloney murine leukemia virus
89623	Snow goose hepatitis B virus
1458710	Mycobacterium phage Badfish

A.3 REAL DATASETS USED

The table below (Table A.3) lists the real datasets from sequencing reads of bacterial genomes used for evaluate Kraken 2 plus.

Table A.3: List of real datasets from sequencing reads of bacterial genomes.

SRA FILE	TAX ID	SCIENTIFIC NAME
ERR657992	83333	Escherichia coli K-12
ERR738806	818	Bacteroides thetaiotaomicron
ERR738813	37734	Enterococcus casseliflavus
ERR757411	195	Campylobacter coli
ERR760539	485917	Pedobacter heparinus DSM 2366
ERR760543	222523	Bacillus cereus ATCC 10987
ERR760549	419947	Mycobacterium tuberculosis H37Ra
ERR915393	446	Legionella pneumophila
SRR1183746	1243618	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20110353
SRR1183769	1412472	Salmonella enterica subsp. enterica serovar Enteritidis str. SA19980677
SRR1183771	1412474	Salmonella enterica subsp. enterica serovar Enteritidis str. SA19970769
SRR1183773	1412476	Salmonella enterica subsp. enterica

Continued on next page

Table A.3 – *Continued from previous page*

SRA FILE	TAX ID	SCIENTIFIC NAME
		serovar Enteritidis str. SA20094682
SRR1183775	1412478	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20084824
SRR1183788	1412491	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20094352
SRR1183790	1412493	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA19942384
SRR1183792	1412495	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20123395
SRR1183794	1412497	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA19961622
SRR1183796	1412499	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA19994216
SRR1183799	1412502	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA19982831
SRR1183801	1412504	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20094350
SRR1183803	1412506	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20083456
SRR1183805	1412508	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20092320
SRR1183807	1412510	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20093977
SRR1183809	1412512	Salmonella enterica subsp. enterica
		erovar Enteritidis str. SA20093430
SRR1183811	1412514	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20093421
SRR1183813	1412516	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20094383
SRR1183815	1412518	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20094642
SRR1183817	1412520	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20093543
SRR1183819	1412522	Salmonella enterica subsp. enterica
		serovar Enteritidis str. SA20093538

Continued on next page

Table A.3 – *Continued from previous page*

SRA FILE	TAX ID	SCIENTIFIC NAME
SRR1183821	1412524	Salmonella enterica subsp. enterica serovar Enteritidis str. SA20094079
SRR1183823	1412526	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121825
SRR1183825	1412528	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120213
SRR1183827	1412530	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121004
SRR1183829	1412532	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120776
SRR1183831	1412534	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120685
SRR1183833	1412536	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120544
SRR1183835	1412538	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20111515
SRR1183837	1412540	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121751
SRR1183839	1412542	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120970
SRR1183841	1412544	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120505
SRR1183843	1412546	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120240
SRR1183845	1412548	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120219
SRR1183847	1412550	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120722
SRR1183849	1412552	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120469
SRR1183851	1412554	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121744
SRR1183854	1412557	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121542
SRR1183856	1412559	Salmonella enterica subsp. enterica

Continued on next page

Table A.3 – *Continued from previous page*

SRA FILE	TAX ID	SCIENTIFIC NAME
		serovar Enteritidis str. EC20120677
SRR1183858	1412561	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121672
SRR1183860	1412563	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121746
SRR1183862	1412565	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121671
SRR1183864	1412567	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20111554
SRR1183866	1412569	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120994
SRR1183868	1412571	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121812
SRR1183870	1412573	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20122045
SRR1183872	1412575	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20122031
SRR1183874	1412577	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20122022
SRR1183876	1412579	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121989
SRR1183878	1412581	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121976
SRR1183880	1412583	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20121969
SRR1183882	1412585	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20090530
SRR1183884	1412587	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20090195
SRR1183886	1412589	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20130346
SRR1183888	1412591	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20130348
SRR1183896	1412599	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20100089

Continued on next page

Table A.3 – *Continued from previous page*

SRA FILE	TAX ID	SCIENTIFIC NAME
SRR1183898	1412601	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120051
SRR1183900	1412603	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120580
SRR1183902	1412605	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120590
SRR1183904	1412607	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120686
SRR1183906	1412609	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120734
SRR1183908	1412611	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120773
SRR1183910	1412613	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120917
SRR1183912	1412615	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120925
SRR1183914	1412617	Salmonella enterica subsp. enterica serovar Enteritidis str. EC20120927
SRR1290758	882	Desulfovibrio vulgaris str. Hildenborough

A.4 ADDITIONAL GRAPHICS AND RESULTS TABLES

The section contains the remainder graphics and tables not inserted in the results section (Section 5.4) obtained by executing the tools in Table 5.1 with the datasets described in Section 5.2.

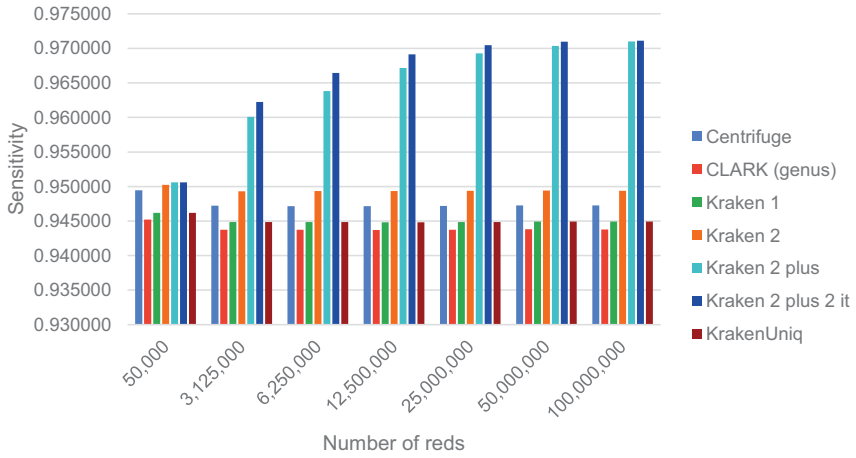

Figure A.2: Bacteria sensitivity evaluation at genus level.

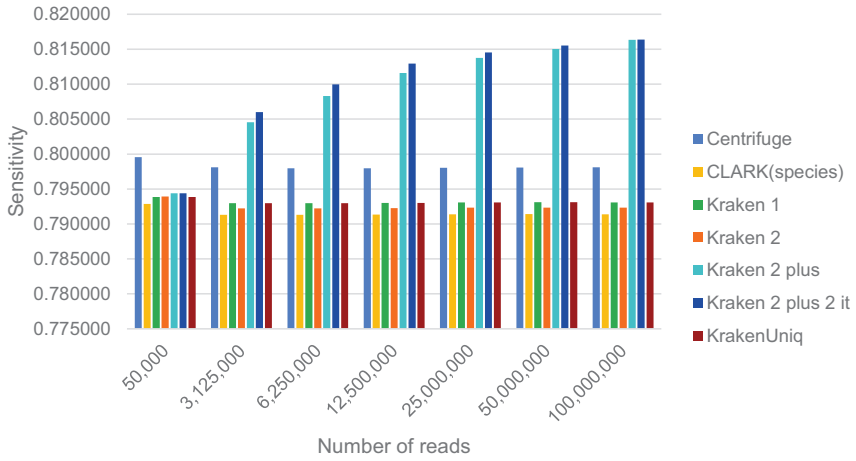

Figure A.3: Bacteria sensitivity evaluation at species level.

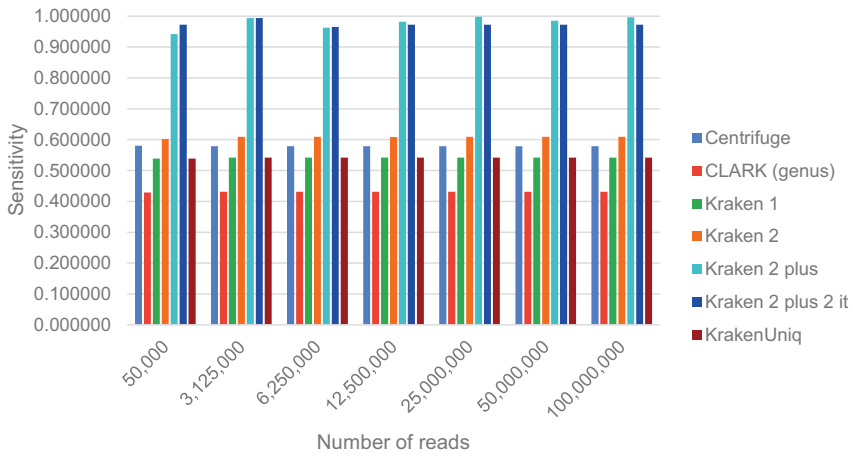

Figure A.4: Viruses sensitivity evaluation at genus level.

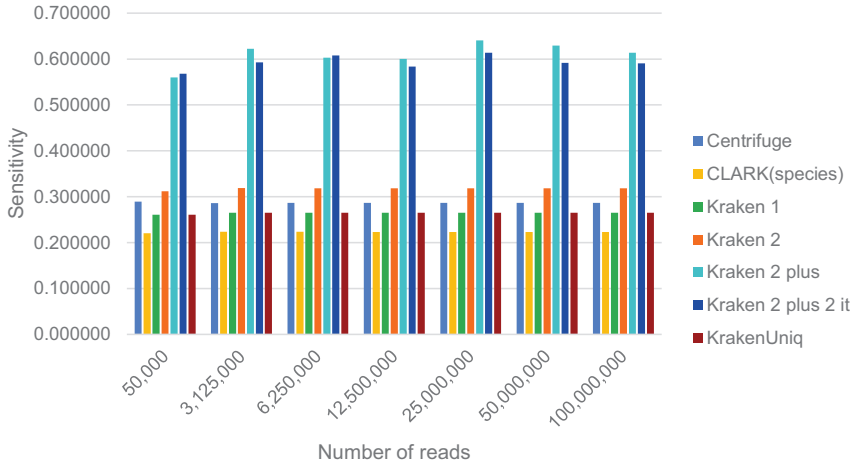

Figure A.5: Viruses sensitivity evaluation at species level.

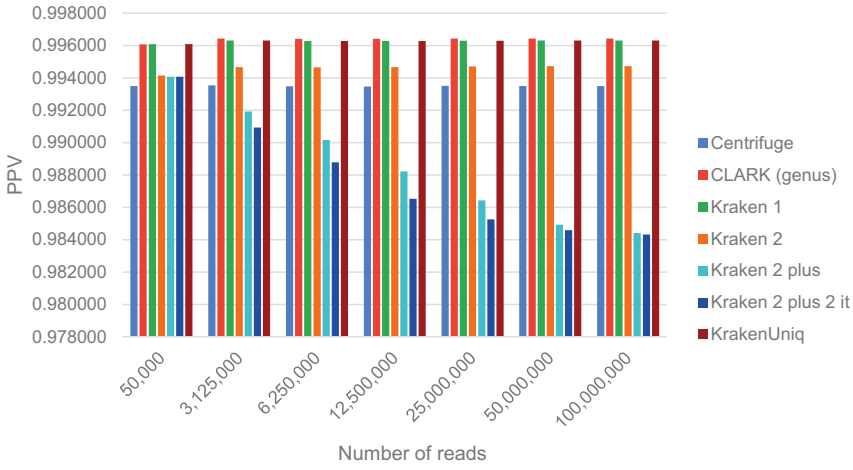

Figure A.6: Bacteria PPV evaluation at genus level.

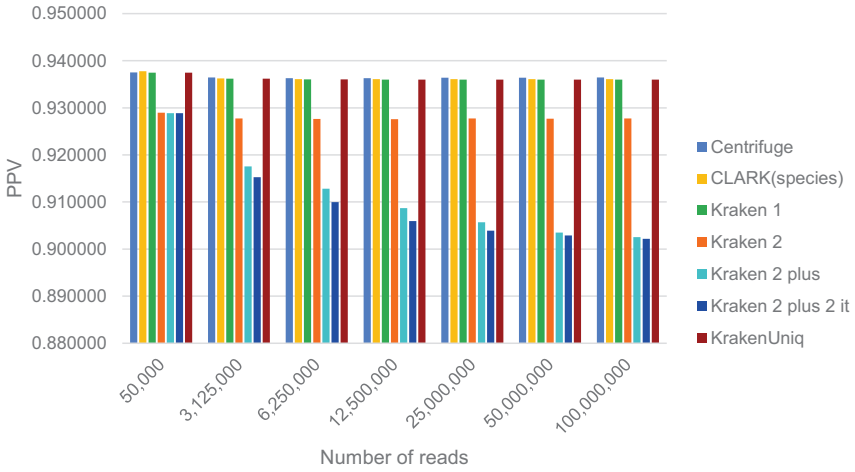

Figure A.7: Bacteria PPV evaluation at species level.

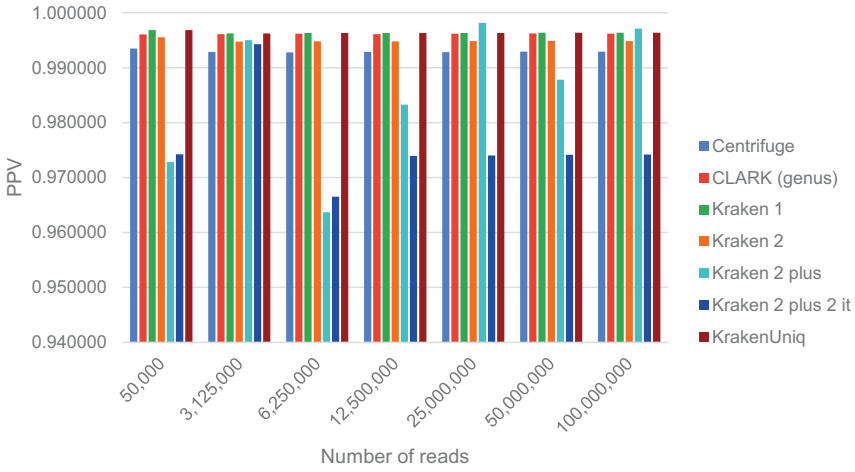

Figure A.8: Viruses PPV evaluation at genus level.

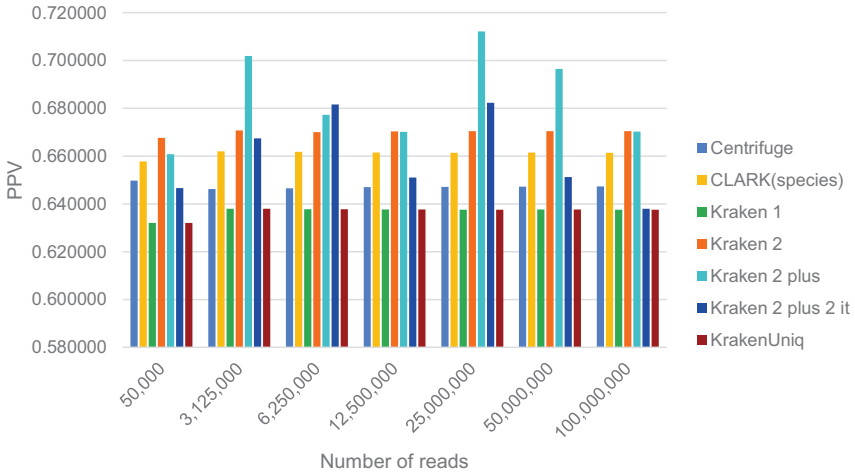

Figure A.9: Viruses PPV evaluation at species level.

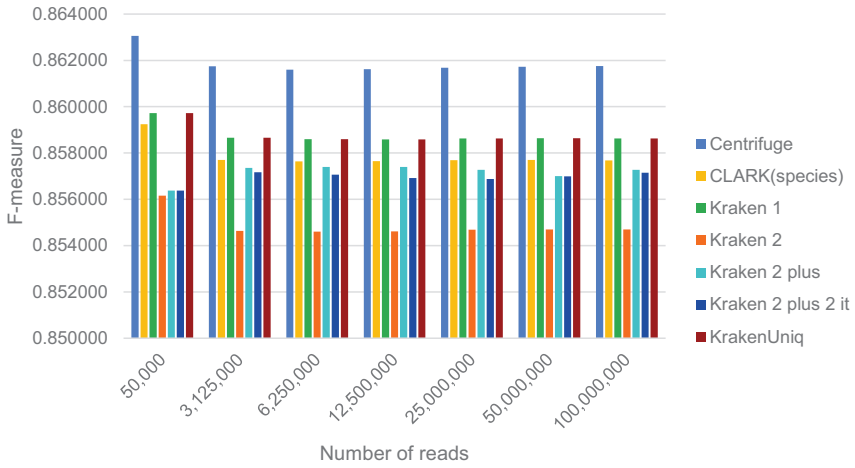

Figure A.10: Bacteria F-measure evaluation at species level.

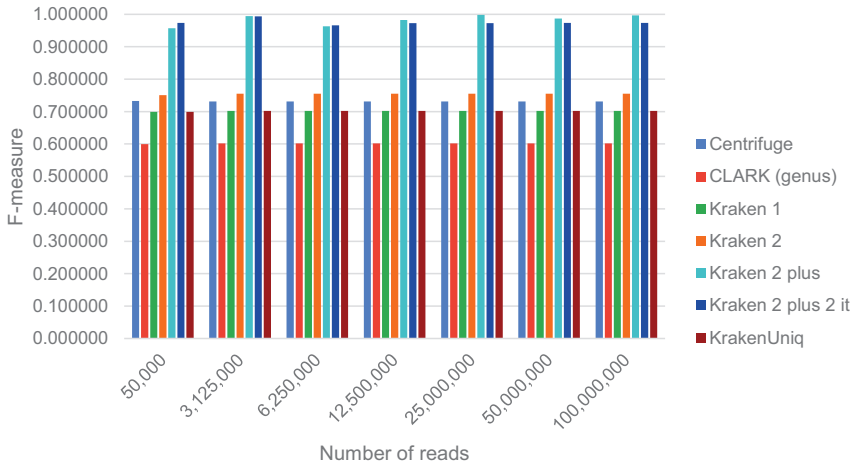

Figure A.11: Viruses F-measure evaluation at genus level.

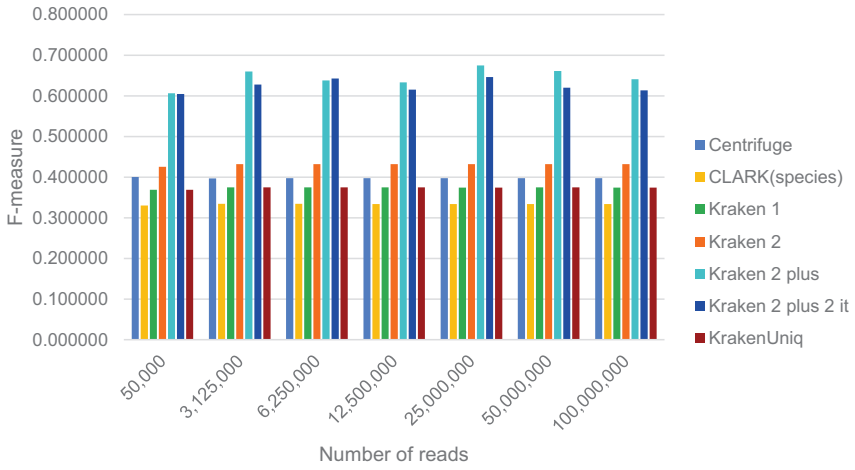

Figure A.12: Viruses F-measure evaluation at species level.

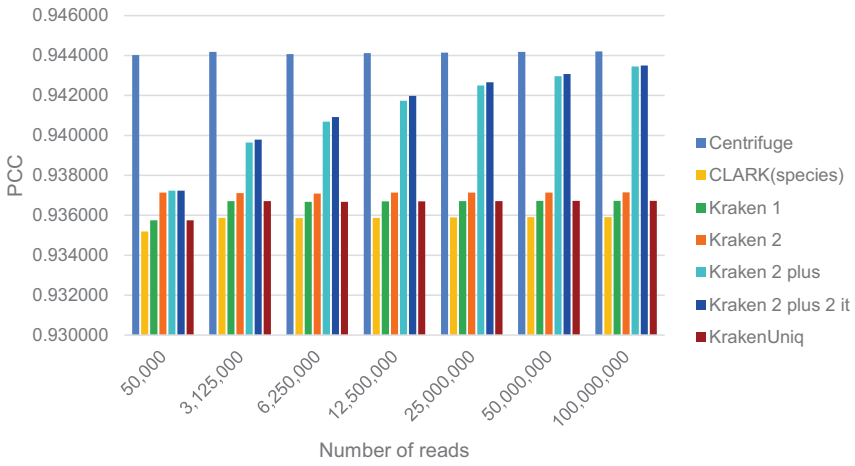

Figure A.13: Bacteria PCC evaluation at species level.

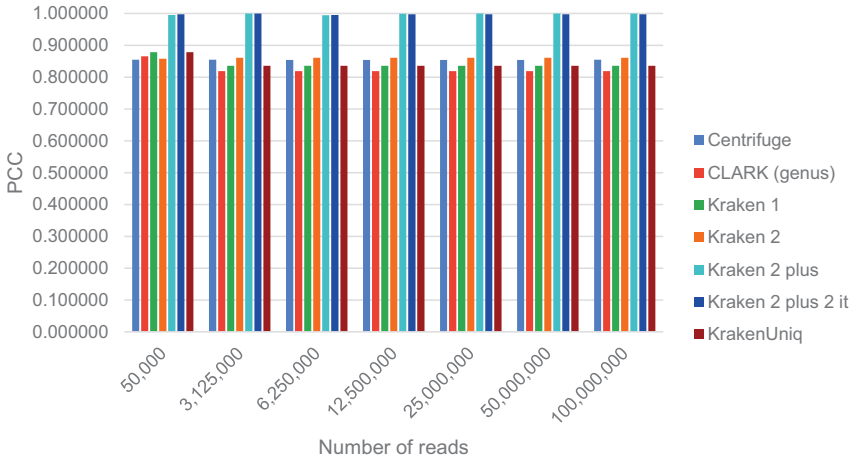

Figure A.14: Viruses PCC evaluation at genus level.

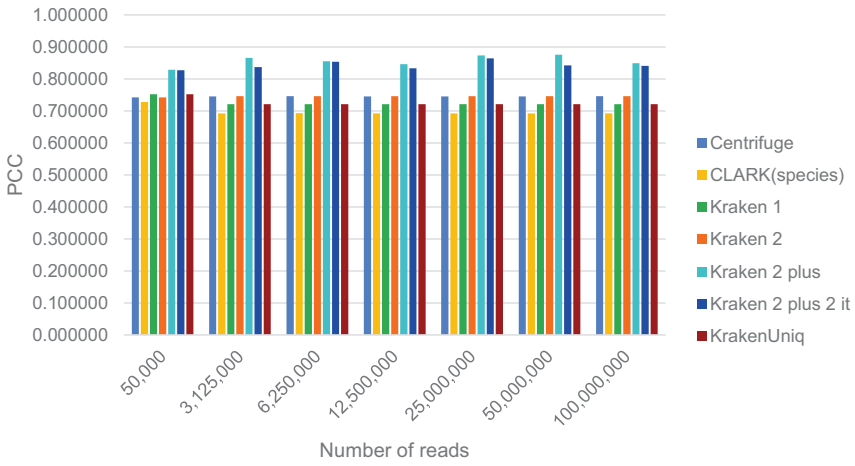

Figure A.15: Viruses PCC evaluation at species level.

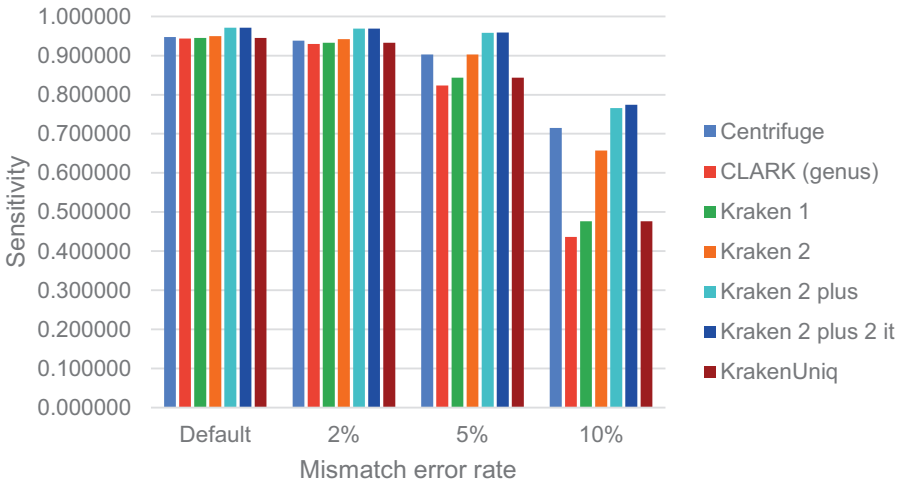

Figure A.16: Bacteria sensitivity evaluation at genus level with mismatch errors.

Figure A.17: Bacteria sensitivity evaluation at species level with mismatch errors.

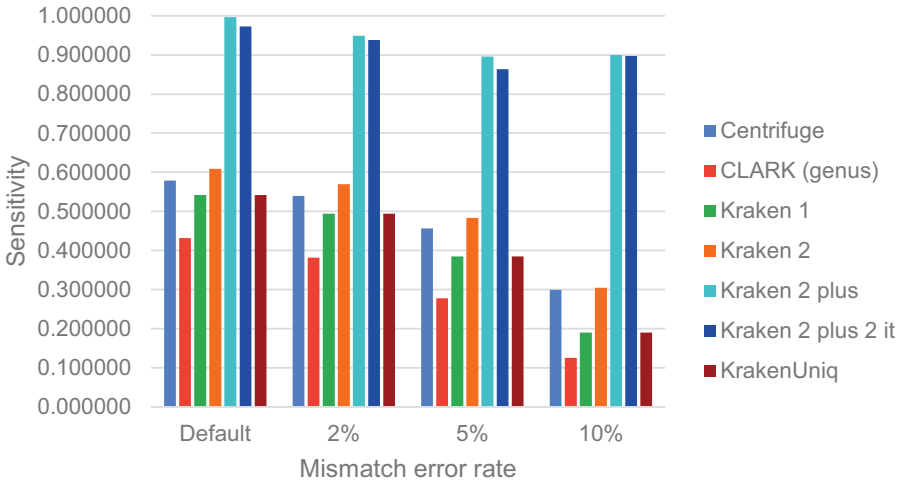

Figure A.18: Viruses sensitivity evaluation at genus level with mismatch errors.

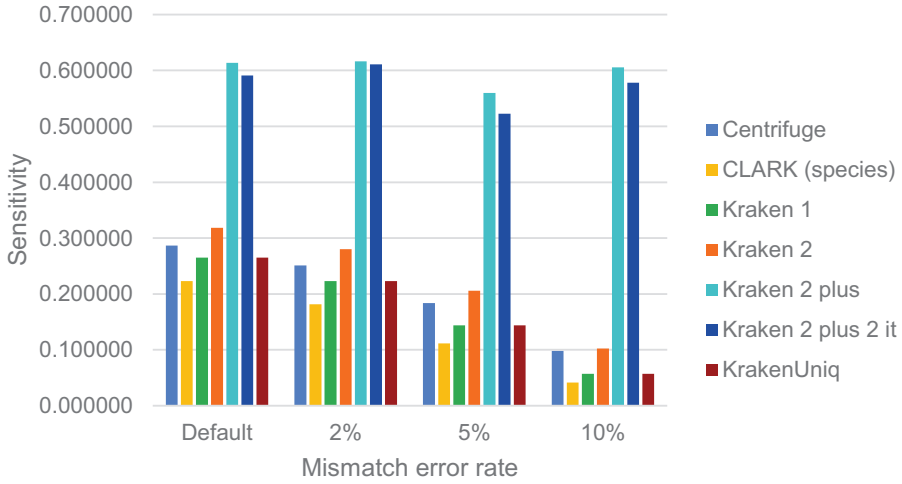

Figure A.19: Viruses sensitivity evaluation at species level with mismatch errors.

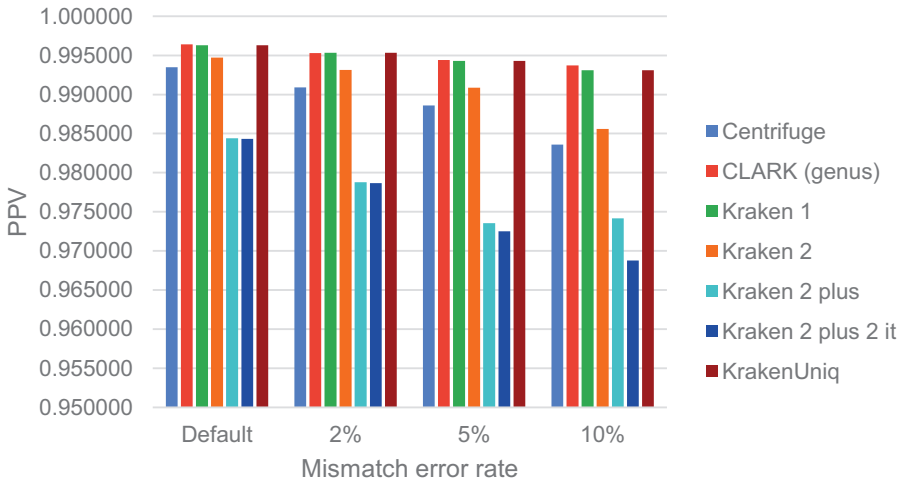

Figure A.20: Bacteria PPV evaluation at genus level with mismatch errors.

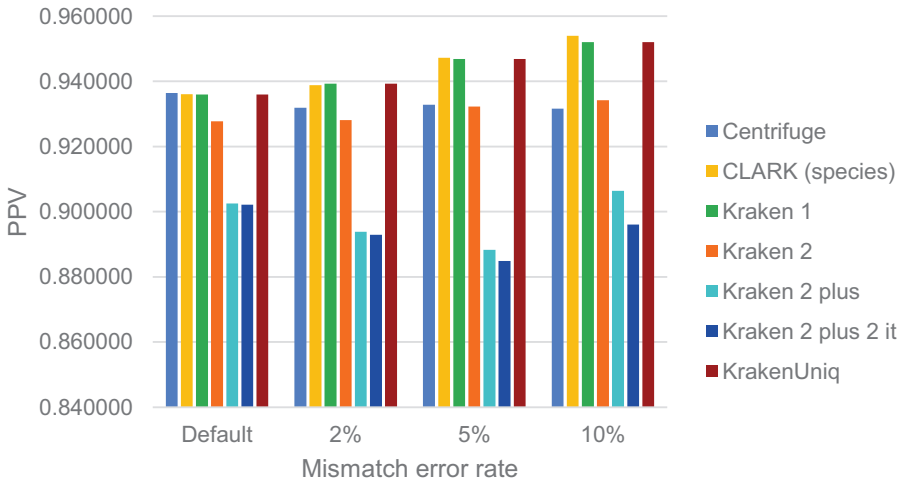

Figure A.21: Bacteria PPV evaluation at species level with mismatch errors.

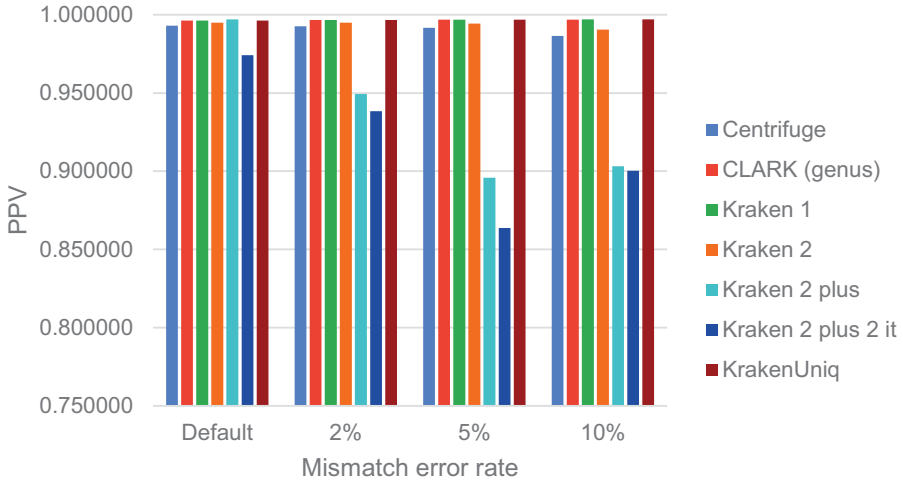

Figure A.22: Viruses PPV evaluation at genus level with mismatch errors.

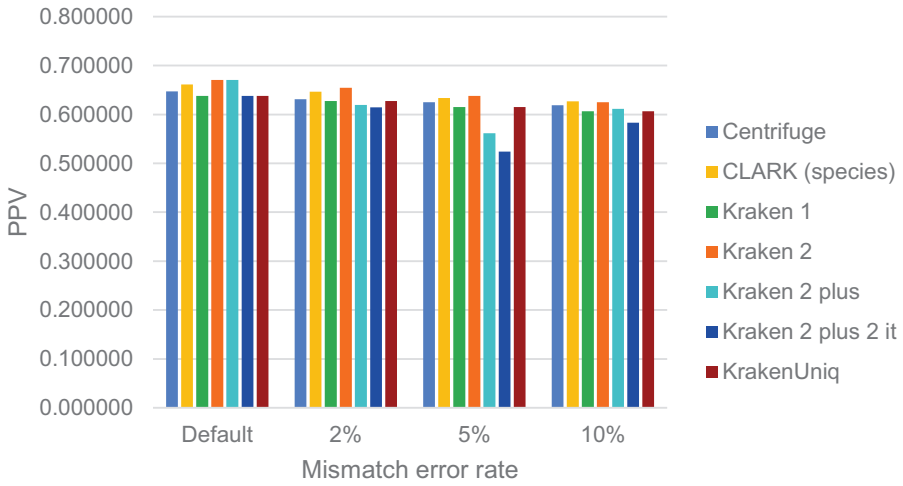

Figure A.23: Viruses PPV evaluation at species level with mismatch errors.

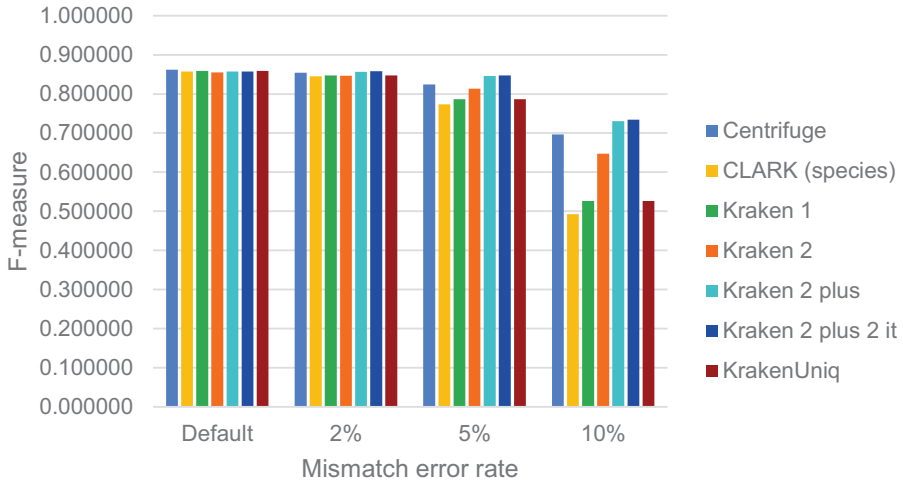

Figure A.24: Bacteria F-measure evaluation at species level with mismatch errors.

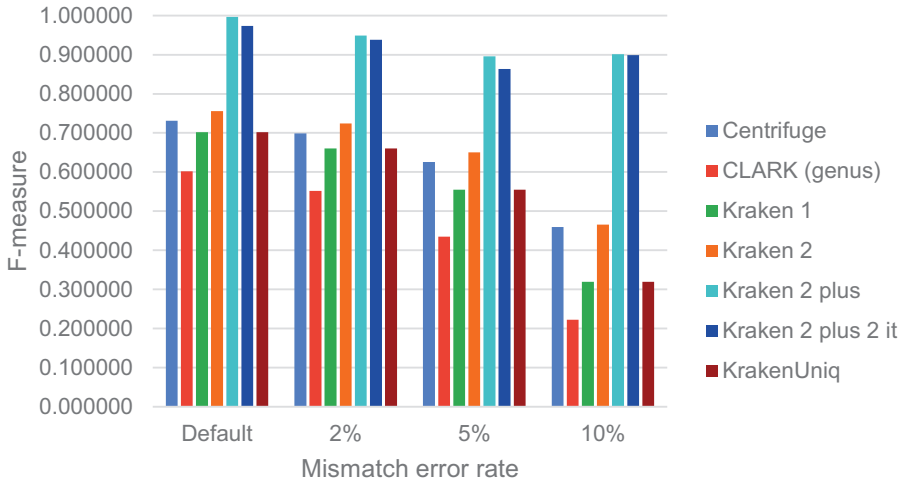

Figure A.25: Viruses F-measure evaluation at genus level with mismatch errors.

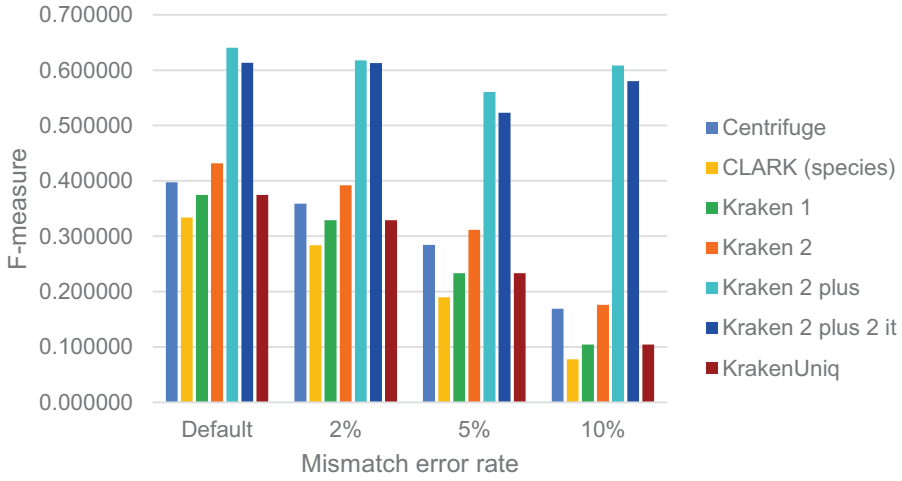

Figure A.26: Viruses F-measure evaluation at species level with mismatch errors.

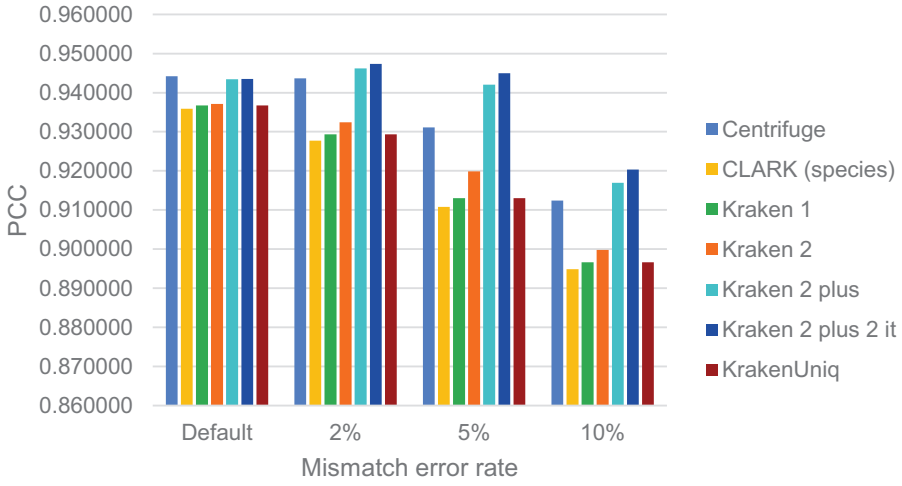

Figure A.27: Bacteria PCC evaluation at species level with mismatch errors.

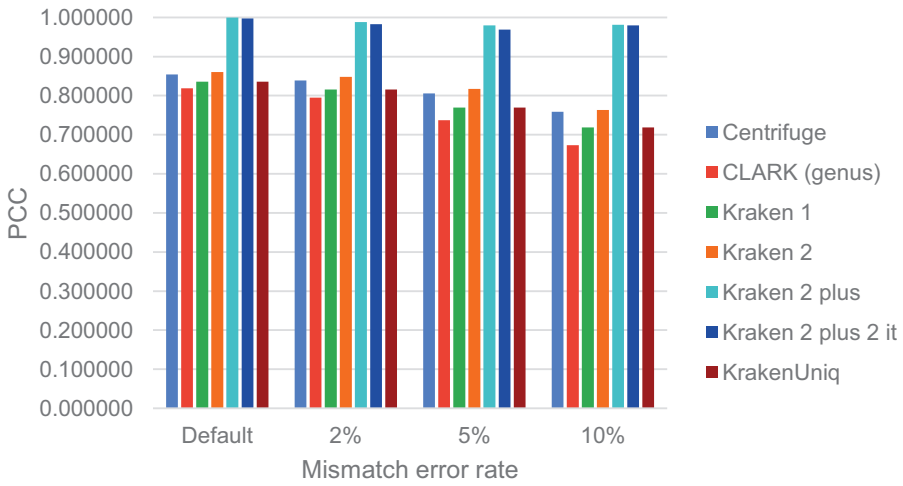

Figure A.28: Viruses PCC evaluation at genus level with mismatch errors.

Figure A.29: Viruses PCC evaluation at species level with mismatch errors.

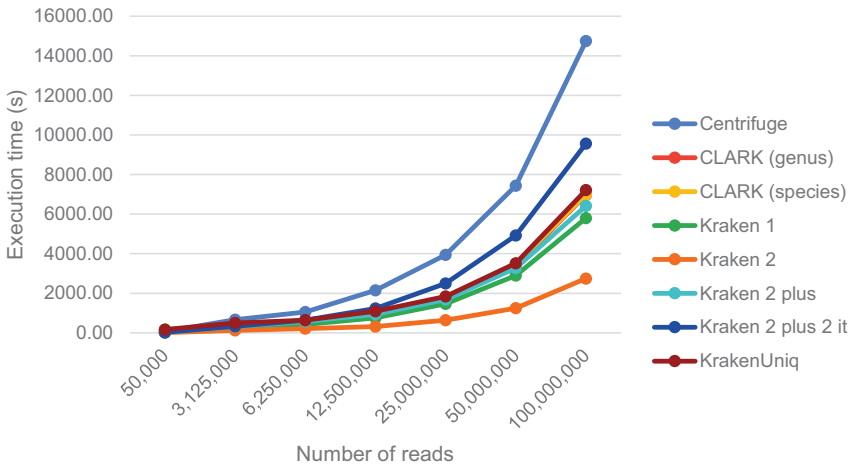

Figure A.30: Execution time obtained varying the number of reads.

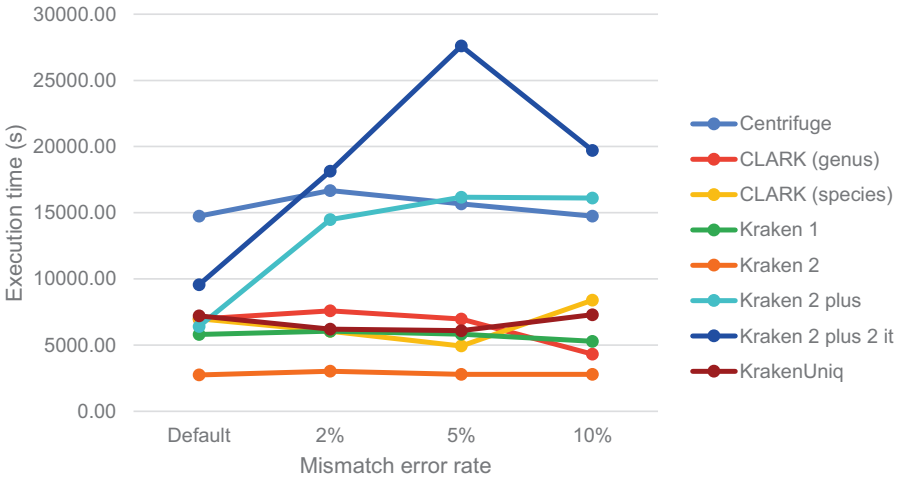

Figure A.31: Execution time obtained varying the mismatch error rate.

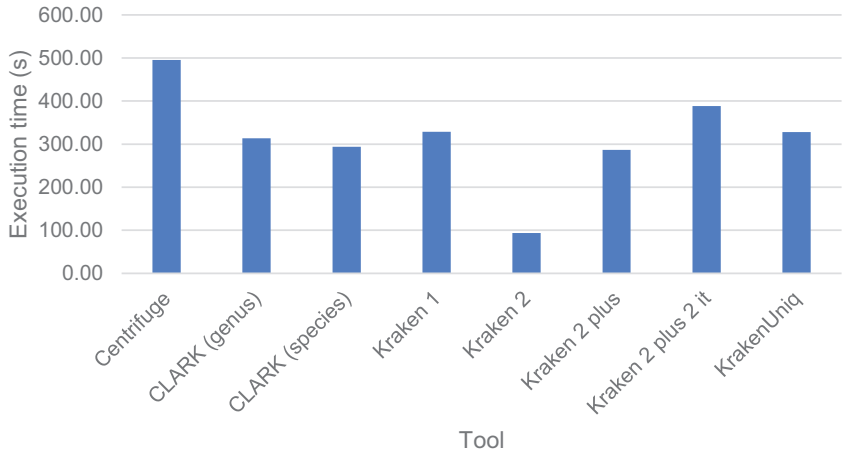

Figure A.32: Execution time with real datasets.

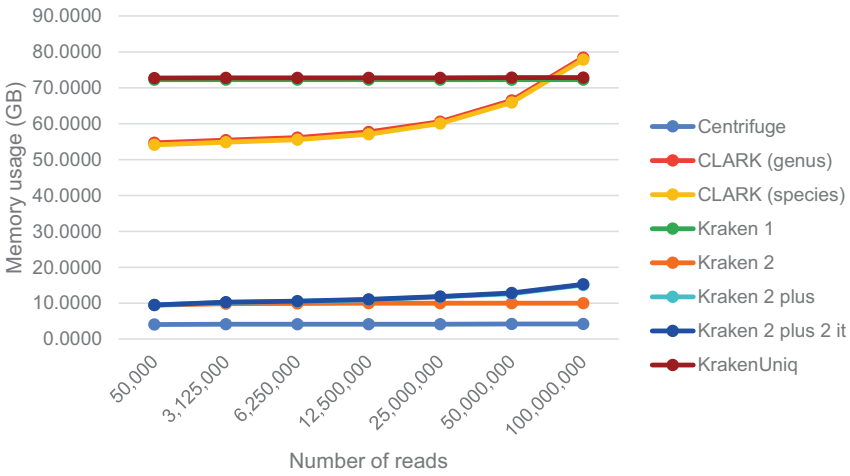

Figure A.33: Memory usage obtained varying the number of reads.

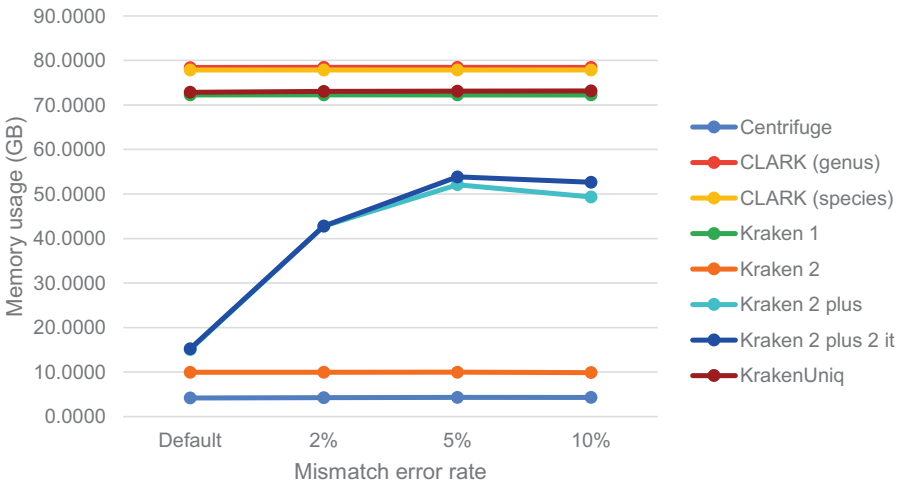

Figure A.34: Memory usage obtained varying the mismatch error rate.

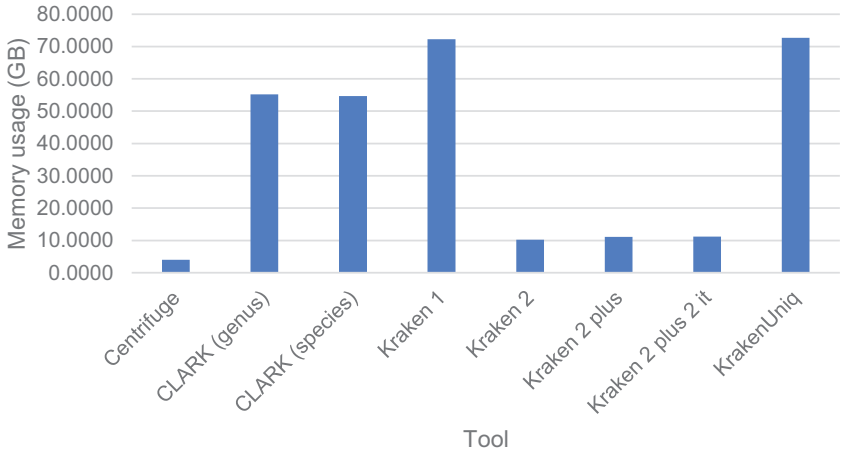

Figure A.35: Execution time with real datasets.

Table A.4: Bacteria (on top) and viruses (on bottom) sensitivity values at genus level obtained varying the number of reads.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.949450	0.947215	0.947169	0.947139	0.947198	0.947247	0.947244
CLARK (genus)	0.945225	0.943750	0.943744	0.943719	0.943754	0.943819	0.943789
Kraken 1	0.946175	0.944840	0.944853	0.944833	0.944871	0.944928	0.944911
Kraken 2	0.950225	0.949301	0.949342	0.949330	0.949366	0.949416	0.949395
Kraken 2 plus	0.950600	0.960094	0.963803	0.967148	0.969280	0.970341	0.971013
Kraken 2 plus 2 it	0.950600	0.962242	0.966447	0.969148	0.970453	0.970967	0.971117
KrakenUniq	0.946175	0.944840	0.944853	0.944833	0.944871	0.944928	0.944911

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.949450	0.947215	0.947169	0.947139	0.947198	0.947247	0.947244
CLARK (genus)	0.945225	0.943750	0.943744	0.943719	0.943754	0.943819	0.943789
Kraken 1	0.946175	0.944840	0.944853	0.944833	0.944871	0.944928	0.944911
Kraken 2	0.950225	0.949301	0.949342	0.949330	0.949366	0.949416	0.949395
Kraken 2 plus	0.950600	0.960094	0.963803	0.967148	0.969280	0.970341	0.971013
Kraken 2 plus 2 it	0.950600	0.962242	0.966447	0.969148	0.970453	0.970967	0.971117
KrakenUniq	0.946175	0.944840	0.944853	0.944833	0.944871	0.944928	0.944911

Table A.5: Bacteria (on top) and viruses (on bottom) sensitivity values at species level obtained varying the number of reads.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.799550	0.798099	0.797955	0.797981	0.798021	0.798089	0.798111
CLARK(species)	0.792850	0.791318	0.791310	0.791330	0.791397	0.791427	0.791394
Kraken 1	0.793875	0.792989	0.792976	0.793011	0.793075	0.793102	0.793079
Kraken 2	0.793925	0.792220	0.792240	0.792277	0.792325	0.792346	0.792333
Kraken 2 plus	0.794400	0.804552	0.808305	0.811603	0.813772	0.815043	0.816338
Kraken 2 plus 2 it	0.794400	0.806001	0.809982	0.812932	0.814523	0.815530	0.816379
KrakenUniq	0.793875	0.792989	0.792976	0.793011	0.793075	0.793102	0.793079

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.289500	0.286277	0.286610	0.286511	0.286557	0.286586	0.286609
CLARK(species)	0.220600	0.223506	0.223394	0.223092	0.223115	0.223207	0.223180
Kraken 1	0.260500	0.265174	0.265302	0.265085	0.265096	0.265159	0.265073
Kraken 2	0.311900	0.318674	0.318594	0.318425	0.318571	0.318512	0.318445
Kraken 2 plus	0.559900	0.622315	0.602976	0.600083	0.640747	0.629426	0.613541
Kraken 2 plus 2 it	0.567900	0.592822	0.607621	0.583407	0.613895	0.591608	0.590771
KrakenUniq	0.260500	0.265174	0.265302	0.265085	0.265096	0.265159	0.265073

Table A.6: Bacteria (on top) and viruses (on bottom) PPV values at genus level obtained varying the number of reads.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.993486	0.993534	0.993481	0.993458	0.993499	0.993497	0.993498
CLARK (genus)	0.996075	0.996423	0.996413	0.996411	0.996424	0.996425	0.996418
Kraken 1	0.996079	0.996303	0.996281	0.996276	0.996294	0.996302	0.996301
Kraken 2	0.994141	0.994661	0.994653	0.994666	0.994703	0.994721	0.994714
Kraken 2 plus	0.994066	0.991928	0.990156	0.988215	0.986420	0.984934	0.984404
Kraken 2 plus 2 it	0.994066	0.990921	0.988776	0.986529	0.985248	0.984593	0.984328
KrakenUniq	0.996079	0.996303	0.996281	0.996276	0.996294	0.996302	0.996301

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.993493	0.992882	0.992781	0.992866	0.992847	0.992929	0.992935
CLARK (genus)	0.996053	0.996084	0.996186	0.996119	0.996172	0.996217	0.996191
Kraken 1	0.996852	0.996241	0.996314	0.996302	0.996315	0.996372	0.996366
Kraken 2	0.995536	0.994732	0.994807	0.994781	0.994828	0.994872	0.994845
Kraken 2 plus	0.972845	0.995018	0.963691	0.983259	0.998156	0.987827	0.997113
Kraken 2 plus 2 it	0.974248	0.994250	0.966532	0.973951	0.974035	0.974149	0.974174
KrakenUniq	0.996852	0.996241	0.996314	0.996302	0.996315	0.996372	0.996366

Table A.7: Bacteria (on top) and viruses (on bottom) PPV values at species level obtained varying the number of reads.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.937531	0.936420	0.936270	0.936288	0.936383	0.936387	0.936426
CLARK(species)	0.937785	0.936234	0.936105	0.936093	0.936107	0.936095	0.936075
Kraken 1	0.937472	0.936188	0.936054	0.935991	0.936003	0.935986	0.935993
Kraken 2	0.928975	0.927732	0.927628	0.927603	0.927716	0.927709	0.927712
Kraken 2 plus	0.928851	0.917575	0.912840	0.908678	0.905689	0.903514	0.902534
Kraken 2 plus 2 it	0.928851	0.915281	0.909964	0.905948	0.903884	0.902888	0.902200
KrakenUniq	0.937472	0.936188	0.936054	0.935991	0.936003	0.935986	0.935993

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.649686	0.646244	0.646539	0.647034	0.647107	0.647207	0.647314
CLARK(species)	0.657722	0.662011	0.661734	0.661482	0.661371	0.661434	0.661368
Kraken 1	0.631975	0.637978	0.637710	0.637695	0.637563	0.637675	0.637602
Kraken 2	0.667594	0.670677	0.670030	0.670321	0.670410	0.670449	0.670398
Kraken 2 plus	0.660727	0.701923	0.677203	0.670131	0.712166	0.696412	0.670195
Kraken 2 plus 2 it	0.646590	0.667355	0.681554	0.650995	0.682310	0.651243	0.637936
KrakenUniq	0.631975	0.637978	0.637710	0.637695	0.637563	0.637675	0.637602

Table A.8: Bacteria (on top) and viruses (on bottom) F-measure values at genus level obtained varying the number of reads.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.970969	0.969822	0.969772	0.969746	0.969796	0.969821	0.969820
CLARK (genus)	0.969984	0.969371	0.969364	0.969349	0.969374	0.969409	0.969390
Kraken 1	0.970486	0.969889	0.969886	0.969873	0.969901	0.969935	0.969925
Kraken 2	0.971687	0.971452	0.971469	0.971470	0.971506	0.971541	0.971526
Kraken 2 plus	0.971847	0.975751	0.976802	0.977568	0.977775	0.977583	0.977663
Kraken 2 plus 2 it	0.971847	0.976371	0.977484	0.977761	0.977795	0.977732	0.977678
KrakenUniq	0.970486	0.969889	0.969886	0.969873	0.969901	0.969935	0.969925

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.732576	0.731234	0.731255	0.731170	0.731162	0.731253	0.731240
CLARK (genus)	0.599706	0.602016	0.602200	0.601963	0.601935	0.602095	0.602049
Kraken 1	0.699091	0.701949	0.702137	0.702012	0.702049	0.702207	0.702174
Kraken 2	0.750374	0.755320	0.755424	0.755247	0.755413	0.755429	0.755397
Kraken 2 plus	0.957277	0.994695	0.963367	0.982795	0.998049	0.986713	0.996907
Kraken 2 plus 2 it	0.973273	0.994002	0.965769	0.973144	0.973189	0.973296	0.973311
KrakenUniq	0.699091	0.701949	0.702137	0.702012	0.702049	0.702207	0.702174

Table A.9: Bacteria (on top) and viruses (on bottom) F-measure values at species level obtained varying the number of reads.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.863060	0.861744	0.861597	0.861620	0.861683	0.861724	0.861754
CLARK(species)	0.859248	0.857698	0.857639	0.857646	0.857691	0.857703	0.857676
Kraken 1	0.859719	0.858659	0.858595	0.858589	0.858632	0.858640	0.858630
Kraken 2	0.856157	0.854638	0.854605	0.854616	0.854692	0.854701	0.854695
Kraken 2 plus	0.856381	0.857355	0.857398	0.857401	0.857274	0.857001	0.857275
Kraken 2 plus 2 it	0.856381	0.857172	0.857067	0.856923	0.856880	0.856988	0.857146
KrakenUniq	0.859719	0.858659	0.858595	0.858589	0.858632	0.858640	0.858630

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.400526	0.396784	0.397159	0.397158	0.397215	0.397263	0.397305
CLARK(species)	0.330388	0.334185	0.334025	0.333655	0.333667	0.333778	0.333739
Kraken 1	0.368928	0.374633	0.374714	0.374495	0.374484	0.374565	0.374467
Kraken 2	0.425164	0.432055	0.431848	0.431753	0.431906	0.431859	0.431787
Kraken 2 plus	0.606149	0.659726	0.637938	0.633176	0.674571	0.661226	0.640618
Kraken 2 plus 2 it	0.604696	0.627884	0.642467	0.615350	0.646297	0.619995	0.613448
KrakenUniq	0.368928	0.374633	0.374714	0.374495	0.374484	0.374565	0.374467

Table A.10: Bacteria (on top) and viruses (on bottom) PCC values at genus level obtained varying the number of reads.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.995753	0.995309	0.995285	0.995276	0.995298	0.995305	0.995313
CLARK (genus)	0.993812	0.993746	0.993731	0.993734	0.993744	0.993762	0.993762
Kraken 1	0.994127	0.994092	0.994083	0.994087	0.994101	0.994116	0.994120
Kraken 2	0.995181	0.995139	0.995129	0.995135	0.995148	0.995159	0.995165
Kraken 2 plus	0.995182	0.995917	0.996138	0.996361	0.996499	0.996601	0.996666
Kraken 2 plus 2 it	0.995182	0.995992	0.996244	0.996440	0.996554	0.996634	0.996690
KrakenUniq	0.994127	0.994092	0.994083	0.994087	0.994101	0.994116	0.994120

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.854613	0.854449	0.854281	0.854237	0.854303	0.854323	0.854385
CLARK (genus)	0.865418	0.818611	0.818673	0.818646	0.818718	0.818759	0.818793
Kraken 1	0.878367	0.835279	0.835342	0.835313	0.835370	0.835403	0.835439
Kraken 2	0.857767	0.860560	0.860612	0.860649	0.860726	0.860677	0.860675
Kraken 2 plus	0.994830	0.999962	0.994402	0.999012	0.999991	0.999367	0.999980
Kraken 2 plus 2 it	0.997190	0.999962	0.995364	0.997359	0.997351	0.997334	0.997334
KrakenUniq	0.878367	0.835279	0.835342	0.835313	0.835370	0.835403	0.835439

Table A.11: Bacteria (on top) and viruses (on bottom) PCC values at species level obtained varying the number of reads.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.944028	0.944179	0.944075	0.944125	0.944147	0.944181	0.944203
CLARK (species)	0.935192	0.935872	0.935853	0.935870	0.935894	0.935908	0.935910
Kraken 1	0.935754	0.936706	0.936678	0.936696	0.936705	0.936724	0.936727
Kraken 2	0.937134	0.937111	0.937090	0.937132	0.937136	0.937134	0.937143
Kraken 2 plus	0.937236	0.939643	0.940693	0.941733	0.942504	0.942967	0.943458
Kraken 2 plus 2 it	0.937236	0.939787	0.940921	0.941983	0.942660	0.943074	0.943500
KrakenUniq	0.935754	0.936706	0.936678	0.936696	0.936705	0.936724	0.936727

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	0.741954	0.744939	0.745584	0.745571	0.745527	0.745573	0.745740
CLARK (species)	0.727502	0.692492	0.692554	0.692370	0.692158	0.692201	0.692432
Kraken 1	0.752380	0.720837	0.721086	0.721111	0.720928	0.720863	0.721038
Kraken 2	0.741940	0.745632	0.745860	0.745709	0.745803	0.745749	0.745855
Kraken 2 plus	0.828260	0.865824	0.855394	0.845886	0.873407	0.875420	0.849358
Kraken 2 plus 2 it	0.827058	0.836933	0.853804	0.833081	0.863925	0.842294	0.840689
KrakenUniq	0.752380	0.720837	0.721086	0.721111	0.720928	0.720863	0.721038

Table A.12: Real datasets evaluation measures means at genus level.

TOOL	SENSITIVITY	PPV	F-MEASURE
Centrifuge	0.881150	0.886673	0.883866
CLARK (genus)	0.885083	0.890147	0.887558
Kraken 1	0.880941	0.887200	0.884017
Kraken 2	0.880434	0.887072	0.883705
Kraken 2 plus	0.883841	0.886128	0.884982
Kraken 2 plus 2 it	0.883899	0.886080	0.884988
KrakenUniq	0.880939	0.887200	0.884016

Table A.13: Real datasets evaluation measures means at species level.

TOOL	SENSITIVITY	PPV	F-MEASURE
Centrifuge	0.867786	0.873600	0.870656
CLARK (species)	0.870643	0.876981	0.873757
Kraken 1	0.866725	0.874150	0.870388
Kraken 2	0.866397	0.874088	0.870203
Kraken 2 plus	0.869713	0.873062	0.871383
Kraken 2 plus 2 it	0.869796	0.873010	0.871398
KrakenUniq	0.866724	0.874150	0.870387

Table A.14: Classification execution time (in s) for each classifier as the number of reads varies.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	59.27	669.30	1052.20	2145.24	3936.77	7433.35	14745.77
CLARK (genus)	106.51	320.58	500.30	911.59	1786.05	3396.09	6975.98
CLARK (species)	116.36	321.60	523.61	927.91	1817.10	3474.33	6999.22
Kraken 1	164.83	330.48	423.76	754.65	1466.89	2895.39	5801.03
Kraken 2	11.35	118.78	211.56	321.43	638.26	1247.18	2747.47
Kraken 2 plus	25.44	320.91	570.77	939.07	1687.85	3268.46	6417.73
Kraken 2 plus 2 it	26.51	327.56	634.34	1238.99	2501.33	4919.90	9563.64
KrakenUniq	168.34	502.62	637.69	1094.57	1839.69	3511.36	7214.12

Table A.15: Memory usage (in GB) for each classifier as the number of reads varies.

TOOL	50,000	3,125,000	6,250,000	12,500,000	25,000,000	50,000,000	100,000,000
Centrifuge	4.0355	4.0997	4.1053	4.1282	4.1436	4.1670	4.1871
CLARK (genus)	54.6579	55.3893	56.1325	57.6778	60.5775	66.5089	78.4228
CLARK (species)	54.0860	54.8175	55.5580	57.0732	60.0580	65.9371	77.8522
Kraken 1	72.2466	72.2519	72.2520	72.2518	72.2524	72.2518	72.2519
Kraken 2	9.4466	9.8596	9.8976	9.9653	9.9525	9.9767	9.9811
Kraken 2 plus	9.4658	10.2143	10.4789	10.8964	11.7671	12.5985	15.0666
Kraken 2 plus 2 it	9.4756	10.2633	10.5374	11.0149	11.8550	12.8080	15.2328
KrakenUniq	72.7134	72.7576	72.7660	72.7737	72.7848	72.8043	72.8440

Table A.16: Real datasets execution time (in s) and memory usage (in GB) for each classifier.

TOOL	EXECUTION TIME	MEMORY USAGE
Centrifuge	495.73	4.0291
CLARK (genus)	313.52	55.2437
CLARK (species)	293.61	54.6511
Kraken 1	328.83	72.2428
Kraken 2	93.68	10.2497
Kraken 2 plus	286.72	11.1128
Kraken 2 plus 2 it	388.16	11.1651
KrakenUniq	328.05	72.7239

BIBLIOGRAPHY

[1] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. "Basic local alignment search tool." In: *Journal of molecular biology* 215.3 (1990), pp. 403–410.

[2] Stinus Lindgreen, Karen L Adair, and Paul P Gardner. "An evaluation of the accuracy and speed of metagenome analysis tools." In: *Scientific reports* 6 (2016), p. 19233.

[3] David R Kelley and Steven L Salzberg. "Clustering metagenomic sequences with interpolated Markov models." In: *BMC bioinformatics* 11.1 (2010), p. 544.

[4] *National Center for Biotechnology Information*. 2019. URL: https://www.ncbi.nlm.nih.gov.

[5] Kevin Chen and Lior Pachter. "Bioinformatics for whole-genome shotgun sequencing of microbial communities." In: *PLoS computational biology* 1.2 (2005), p. e24.

[6] Torsten Thomas, Jack Gilbert, and Folker Meyer. "Metagenomics-a guide from sampling to data analysis." In: *Microbial informatics and experimentation* 2.1 (2012), p. 3.

[7] KM Elkins. "Chapter 4-DNA extraction." In: *Forensic DNA Biology. Academic Press, San Diego* (2013), pp. 39–52.

[8] Mohammed Shehadul Islam, Aditya Aryasomayajula, and Ponnambalam Selvaganapathy. "A review on macroscale and microscale cell lysis methods." In: *Micromachines* 8.3 (2017), p. 83.

[9] Per Hoff-Olsen, Bente Mevåg, Eva Staalstrøm, Bente Hovde, Thore Egeland, and Bjørnar Olaisen. "Extraction of DNA from decomposed human tissue: an evaluation of five extraction methods for short tandem repeat typing." In: *Forensic science international* 105.3 (1999), pp. 171–183.

[10] Izet Eminovic, J Karamehic, F Gavrankapetanovic, and B Heljic. "A simple method of DNA extraction in solving difficult criminal cases." In: *Med Arh* 59.1 (2005), pp. 57–8.

[11] Maxim G Brevnov, Hemant S Pawar, Janna Mundt, Lisa M Calandro, Manohar R Furtado, and Jaiprakash G Shewale. "Developmental validation of the PrepFiler™ forensic DNA extraction kit for extraction of genomic DNA from biological samples." In: *Journal of forensic sciences* 54.3 (2009), pp. 599–607.

[12] V Castella, N Dimo-Simonin, C Brandt-Casadevall, and P Mangin. "Forensic evaluation of the QIAshredder/QIAamp DNA extraction procedure." In: *Forensic science international* 156.1 (2006), pp. 70–73.

[13] Susan A Greenspoon, Marco A Scarpetta, Melanie L Drayton, and SA Turek. "QIAamp spin columns as a method of DNA isolation for forensic casework." In: *Journal of Forensic Science* 43.5 (1998), pp. 1024–1030.

[14] K Drobnic. "Analysis of DNA evidence recovered from epithelial cells in penile swabs." In: *Croatian medical journal* 44.3 (2003), pp. 350–354.

[15] Roger S Lasken. *Genomic DNA amplification by the multiple displacement amplification (MDA) method*. 2009.

[16] Thomas Ishoey, Tanja Woyke, Ramunas Stepanauskas, Mark Novotny, and Roger S Lasken. "Genomic sequencing of single microbial cells from environmental samples." In: *Current opinion in microbiology* 11.3 (2008), pp. 198–204.

[17] Sam Behjati and Patrick S Tarpey. "What is next generation sequencing?" In: *Archives of Disease in Childhood-Education and Practice* 98.6 (2013), pp. 236–238.

[18] Frederick Sanger, Steven Nicklen, and Alan R Coulson. "DNA sequencing with chain-terminating inhibitors." In: *Proceedings of the national academy of sciences* 74.12 (1977), pp. 5463–5467.

[19] Allan M Maxam and Walter Gilbert. "A new method for sequencing DNA." In: *Proceedings of the National Academy of Sciences* 74.2 (1977), pp. 560–564.

[20] Rotem Sorek, Yiwen Zhu, Christopher J Creevey, M Pilar Francino, Peer Bork, and Edward M Rubin. "Genome-wide experimental determination of barriers to horizontal gene transfer." In: *Science* 318.5855 (2007), pp. 1449–1452.

[21] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. "Comparison of next-generation sequencing systems." In: *BioMed Research International* 2012 (2012).

[22] *Roche Life Science*. 2019. URL: https://lifescience.roche.com.

[23] Elaine R Mardis. "The impact of next-generation sequencing technology on genetics." In: *Trends in genetics* 24.3 (2008), pp. 133–141.

[24] Susan M Huse, Julie A Huber, Hilary G Morrison, Mitchell L Sogin, and David Mark Welch. "Accuracy and quality of massively parallel DNA pyrosequencing." In: *Genome biology* 8.7 (2007), R143.

[25] *Roche's GS Junior*. 2019. URL: https://www.roche.com/media/releases/med-cor-2012-01-17t.html.

[26] Daniel R Zerbino and Ewan Birney. "Velvet: algorithms for de novo short read assembly using de Bruijn graphs." In: *Genome research* 18.5 (2008), pp. 821–829.

[27] Ruiqiang Li, Yingrui Li, Karsten Kristiansen, and Jun Wang. "SOAP: short oligonucleotide alignment program." In: *Bioinformatics* 24.5 (2008), pp. 713–714.

[28] Jason R Miller, Sergey Koren, and Granger Sutton. "Assembly algorithms for next-generation sequencing data." In: *Genomics* 95.6 (2010), pp. 315–327.

[29] Pavel A Pevzner, Haixu Tang, and Michael S Waterman. "An Eulerian path approach to DNA fragment assembly." In: *Proceedings of the national academy of sciences* 98.17 (2001), pp. 9748–9753.

[30] John C Wooley, Adam Godzik, and Iddo Friedberg. "A primer on metagenomics." In: *PLoS computational biology* 6.2 (2010), e1000667.

[31] Daniel Dalevi, Devdatt Dubhashi, and Malte Hermansson. "Bayesian classifiers for detecting HGT using fixed and variable order markov models of genomic signatures." In: *Bioinformatics* 22.5 (2006), pp. 517–522.

[32] Alice Carolyn McHardy, Héctor García Martín, Aristotelis Tsirigos, Philip Hugenholtz, and Isidore Rigoutsos. "Accurate phylogenetic classification of variable-length DNA fragments." In: *Nature methods* 4.1 (2007), p. 63.

[33] Takashi Abe, Shigehiko Kanaya, Makoto Kinouchi, Yuta Ichiba, Tokio Kozuki, and Toshimichi Ikemura. "Informatics for unveiling hidden genome signatures." In: *Genome research* 13.4 (2003), pp. 693–702.

[34] Hanno Teeling, Jost Waldmann, Thierry Lombardot, Margarete Bauer, and Frank Oliver Glöckner. "TETRA: a web-service and a stand-alone program for the analysis and comparison of tetranucleotide usage patterns in DNA sequences." In: *BMC bioinformatics* 5.1 (2004), p. 163.

[35] Sharmila S Mande, Monzoorul Haque Mohammed, and Tarini Shankar Ghosh. "Classification of metagenomic sequences: methods and challenges." In: *Briefings in bioinformatics* 13.6 (2012), pp. 669–681.

[36] Alice C McHardy and Isidore Rigoutsos. "What's in the mix: phylogenetic classification of metagenome sequence samples." In: *Current opinion in microbiology* 10.5 (2007), pp. 499–503.

[37] Daniel H Huson, Sina Beier, Isabell Flade, Anna Górska, Mohamed El-Hadidi, Suparna Mitra, Hans-Joachim Ruscheweyh, and Rewati Tappu. "MEGAN community edition-interactive exploration and analysis of large-scale microbiome sequencing data." In: *PLoS computational biology* 12.6 (2016), p. e1004957.

[38] Chon-Kit Kenneth Chan, Arthur L Hsu, Saman K Halgamuge, and Sen-Lin Tang. "Binning sequences using very sparse labels within a metagenome." In: *BMC bioinformatics* 9.1 (2008), p. 215.

[39] Hao Zheng and Hongwei Wu. "Short prokaryotic DNA fragment binning using a hierarchical classifier based on linear discriminant analysis and principal component analysis." In: *Journal of bioinformatics and computational biology* 8.06 (2010), pp. 995–1011.

[40] Naryttza N Diaz, Lutz Krause, Alexander Goesmann, Karsten Niehaus, and Tim W Nattkemper. "TACOA–Taxonomic classification of environmental genomic fragments using a kernelized nearest neighbor approach." In: *BMC bioinformatics* 10.1 (2009), p. 56.

[41] Victor M Markowitz, Natalia N Ivanova, Ernest Szeto, Krishna Palaniappan, Ken Chu, Daniel Dalevi, I-Min A Chen, Yuri Grechkin, Inna Dubchak, Iain Anderson, et al. "IMG/M: a data management and analysis system for metagenomes." In: *Nucleic acids research* 36.suppl_1 (2007), pp. D534–D538.

[42] Elizabeth M Glass, Jared Wilkening, Andreas Wilke, Dionysios Antonopoulos, and Folker Meyer. "Using the metagenomics RAST server (MG-RAST) for analyzing shotgun metagenomes." In: *Cold Spring Harbor Protocols* 2010.1 (2010), pdb–prot5368.

[43] Lutz Krause, Naryttza N Diaz, Alexander Goesmann, Scott Kelley, Tim W Nattkemper, Forest Rohwer, Robert A Edwards, and Jens Stoye. "Phylogenetic classification of short environmental DNA fragments." In: *Nucleic acids research* 36.7 (2008), pp. 2230–2239.

[44] M Monzoorul Haque, Tarini Shankar Ghosh, Dinakar Komanduri, and Sharmila S Mande. "SOrt-ITEMS: Sequence orthology based approach for improved taxonomic estimation of metagenomic sequences." In: *Bioinformatics* 25.14 (2009), pp. 1722–1730.

[45] Bo Liu, Theodore Gibbons, Mohammad Ghodsi, Todd Treangen, and Mihai Pop. "Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences." In: *Genome biology* 12.1 (2011), P11.

[46] Arthur Brady and Steven L Salzberg. "Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models." In: *Nature methods* 6.9 (2009), p. 673.

[47] Henry CM Leung, Siu-Ming Yiu, Bin Yang, Yu Peng, Yi Wang, Zhihua Liu, Jingchi Chen, Junjie Qin, Ruiqiang Li, and Francis YL Chin. "A robust and accurate binning algorithm for metagenomic sequences with arbitrary species abundance ratio." In: *Bioinformatics* 27.11 (2011), pp. 1489–1495.

[48] Ramy K Aziz, Daniela Bartels, Aaron A Best, Matthew DeJongh, Terrence Disz, Robert A Edwards, Kevin Formsma, Svetlana Gerdes, Elizabeth M Glass, Michael Kubal, et al. "The RAST Server: rapid annotations using subsystems technology." In: *BMC genomics* 9.1 (2008), p. 75.

[49] Victor M Markowitz, Konstantinos Mavromatis, Natalia N Ivanova, I-Min A Chen, Ken Chu, and Nikos C Kyrpides. "IMG ER: a system for microbial genome annotation expert review and curation." In: *Bioinformatics* 25.17 (2009), pp. 2271–2278.

[50] Thomas J Sharpton. "An introduction to the analysis of shotgun metagenomic data." In: *Frontiers in plant science* 5 (2014), p. 209.

[51] Pelin Yilmaz, Renzo Kottmann, Dawn Field, Rob Knight, James R Cole, Linda Amaral-Zettler, Jack A Gilbert, Ilene Karsch-Mizrachi, Anjanette Johnston, Guy Cochrane, et al. "Minimum information about a marker gene sequence (MIMARKS) and minimum information about any (x) sequence (MIxS) specifications." In: *Nature biotechnology* 29.5 (2011), p. 415.

[52] *EMBL - European Molecular Biology Laboratory - The European Molecular Biology Laboratory*. 2019. URL: https://www.embl.org/.

[53] *Bioinformation and DDBJ Center*. 2019. URL: https://www.ddbj.nig.ac.jp/index-e.html.

[54] Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. "NCBI reference sequences (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins." In: *Nucleic acids research* 35.suppl_1 (2006), pp. D61–D65.

[55] Gregory D Schuler, Jonathan A Epstein, Hitomi Ohkawa, and Jonathan A Kans. "Entrez: Molecular biology database and retrieval system." In: *Methods in enzymology*. Vol. 266. Elsevier, 1996, pp. 141–162.

[56] Wikipedia contributors. *Taxonomy (biology) - Wikipedia*. Online; accessed 07-October-2019. 2019. URL: https://en.wikipedia.org/wiki/Taxonomy_(biology).

[57] Derrick E. Wood, Jennifer Lu, and Ben Langmead. "Improved metagenomic analysis with Kraken 2." In: *bioRxiv* (2019). DOI: 10.1101/762302. eprint: https://www.biorxiv.org/content/early/2019/09/07/762302.full.pdf. URL: https://www.biorxiv.org/content/early/2019/09/07/762302.

[58] Derrick E Wood and Steven L Salzberg. "Kraken: ultrafast metagenomic sequence classification using exact alignments." In: *Genome biology* 15.3 (2014), R46.

[59] *Kraken2*. 2019. URL: https://ccb.jhu.edu/software/kraken2/.

[60] *Derrick Wood Twitter*. 2019. URL: https://twitter.com/DerrickEWood/status/1171427342998278144.

[61] Aleksandr Morgulis, E Michael Gertz, Alejandro A Schäffer, and Richa Agarwala. "A fast and symmetric DUST implementation to mask low-complexity DNA sequences." In: *Journal of Computational Biology* 13.5 (2006), pp. 1028–1040.

[62] John C Wootton and Scott Federhen. "[33] Analysis of compositionally biased regions in sequence databases." In: *Methods in enzymology*. Vol. 266. Elsevier, 1996, pp. 554–571.

[63] Deanna M Church, Valerie A Schneider, Karyn Meltz Steinberg, Michael C Schatz, Aaron R Quinlan, Chen-Shan Chin, Paul A Kitts, Bronwen Aken, Gabor T Marth, Michael M Hoffman, et al. "Extending reference assembly models." In: *Genome biology* 16.1 (2015), p. 13.

[64] *The UniVec Database*. 2019. URL: https://www.ncbi.nlm.nih.gov/tools/vecscreen/univec/.

[65] Karel Břinda, Maciej Sykulski, and Gregory Kucherov. "Spaced seeds improve k-mer-based metagenomic classification." In: *Bioinformatics* 31.22 (2015), pp. 3584–3592.

[66] Philippe Flajolet and G Nigel Martin. "Probabilistic counting algorithms for data base applications." In: *Journal of computer and system sciences* 31.2 (1985), pp. 182–209.

[67] *SMHasher GitHub repository*. 2019. URL: https://github.com/aappleby/smhasher.

[68] Armando D Solis. "Amino acid alphabet reduction preserves fold information contained in contact interactions in proteins." In: *Proteins: Structure, Function, and Bioinformatics* 83.12 (2015), pp. 2198–2216.

[69] *std::unordered_map - cppreference.com*. 2019. URL: https://en.cppreference.com/w/cpp/container/unordered_map.

[70] Manuel Holtgrewe. "Mason: a read simulator for second generation sequencing data." In: (2010).

[71] Daehwan Kim, Li Song, Florian P Breitwieser, and Steven L Salzberg. "Centrifuge: rapid and sensitive classification of metagenomic sequences." In: *Genome research* 26.12 (2016), pp. 1721–1729.

[72] Rachid Ounit, Steve Wanamaker, Timothy J Close, and Stefano Lonardi. "CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers." In: *BMC genomics* 16.1 (2015), p. 236.

[73] FP Breitwieser, DN Baker, and Steven L Salzberg. "KrakenUniq: confident and fast metagenomics classification using unique k-mer counts." In: *Genome biology* 19.1 (2018), p. 198.

[74] Prashant Pandey, Michael A Bender, Rob Johnson, and Rob Patro. "A general-purpose counting filter: Making every bit count." In: *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM. 2017, pp. 775–787.