1-1-2020

Algorithmic methods for large-scale genomic and metagenomic data analysis

Quang Tran

ALGORITHMIC METHODS FOR LARGE-SCALE GENOMIC AND METAGENOMIC DATA ANALYSIS

by

Quang Minh Tran

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Major: Computer Science

The University of Memphis

August 2020

ACKNOWLEDGMENTS

This accomplishment would not have been possible without the assistance of some special people I've worked with during my time at the University of Memphis. I would like to express my gratitude to: My advisor Dr. Vinhthuy Phan for his guidance and support throughout this work. My committee members: Dr. Ramin Homayouni, Dr. Thomas Watson, Dr. Vasile Rus, for their insight and patience. My labmates: Nam Sy Vo, Shanshan Gao, Kevin Okello, Diem-Trang Pham, Eric Hicks, for all their useful comments and suggestions.

I'd also like to thank: The Computer Science department's administrative staff greatly helped me navigate the logistics of the degree and all the paperwork. Rhonda Smothers, Corinne O'Connor and Lyndsey Rush: Thank you for your constant support. Eric Spangler for his help with installing the tools and troubleshoots of running jobs on the research high performance computing cluster. The Department of Computer Science at the University of Memphis for awarding me a Graduate Assistantship so that I could complete my dissertation.

During my PhD journey, I was fortunate enough to have some exciting internships where I gained some valuable insight into real problems. To my mentors: Jian Yin, Bayo Lau, Betty Ulitsky, Hugo Lam, and Alexej Abyzov. Thank you for investing so much time and effort into working with me. To my colleagues: Lai Xu, Roger Liu, Yunfei Guo, Arijit Panda, Milovan Suvakov, Shobana Sekar, Taejeong Bae, Vivekananda Sarangi. I am also grateful for the great friendship during those summers.

And finally, thank you deeply to my family for their unconditional love, patience, and continued support during my study.

ABSTRACT

Tran, Quang Minh. PhD. The University of Memphis. August 2020. Algorithmic methods for large-scale genomic and metagenomic data analysis. Major Professor: Dr. Vinhthuy Phan.

DNA sequencing technologies have advanced into the realm of big data due to frequent and rapid developments in biologic medicine. This has caused a surge in the necessity of efficient and highly scalable algorithms.This dissertation focuses on central work in read-to-reference alignments, resequencing studies, and metagenomics that were designed with these principles as the guiding reason for their construction.

First, consider the computing intensive task of read-to-reference alignments, where the difficulty of aligning reads to a genome is directly related their complexity. We investigated three different formulations of sequence complexity as viable tools for measuring genome complexity along with how they related to short read alignments and found that repeat measures of complexity were best suited for this task. In particular, the fraction of distinct substrings of lengths close to the read length was found to correlate very highly to alignment accuracy in terms of precision and recall. All this demonstrated how to build models to predict accuracy of short read aligners with predictably low errors. As a result, practitioners can select the most accurate aligners for an unknown genome by comparing how different models predict alignment accuracy based on the genomes complexity. Furthermore, accurate recall rate prediction may help practitioners reduce expenses by using just enough reads to get sufficient sequencing coverage.

Next, focus on the comprehensive task of resequencing studies for analyzing genetic variants of the human population. By using optimal alignments, we revealed that the current variant profiles contained thousands of insertion/deletion (INDEL) that were constructed in a biased manner. The bias is caused by the existence of many theoretically optimal alignments between the reference genome and reads containing alternative alleles at those INDEL locations. We examined several popular aligners and showed that these aligners could be divided into groups whose alignments yielded INDELs that either

strongly agreed or disagreed with reported INDELs. This finding suggests that the agreement or disagreement between the aligners called INDEL and the reported INDEL is merely a result of the arbitrary selection of an optimal alignment. Also of note is LongAGE, a memory efficient of Alignment with Gap Excision (AGE) for defining geneomic variant breakpoints, which enables the precise alignment of longer reads or contigs that potentially contain SVs/CNVs while having a trade off of time compared to AGE.

Finally, consider several resource-intensive tasks in metagenomics. We introduce a new algorithmic method for detecting unknown bacteria, those whose genomes have not been sequenced, in microbial communities. Using the 16S ribosomal RNA (16S rRNA) gene instead of the whole genome's information is not only computational efficient, but also economical; an analysis that demonstrates the 16S rRNA gene retains sufficient information to allow us to detect unknown bacteria in the context of oral microbial communities is provided. Furthermore, the main hypothesis that the classification or identification of microbes in metagenomic samples is better done with long reads than with short reads is iterated upon, by investigating the performance of popular metagenomic classifiers on short reads and longer reads assembled from those short reads. Higher overall performance of species classification was achieved simply by assembling short reads.

These topics about read-to-reference alignments, resequencing studies, and metagenomics are all key focal points in the pages to come. My dissertation delves deeper into these as I cover the contributions my work has made to the field.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table **Page**

Chapter 1

Introduction

The past decade marked impressive progress in DNA sequencing technologies, due to the advent of next-generation sequencing (NGS) platforms which cut the cost for sequencing large genomes, like humans, from millions to thousands of dollars [1]. These advances have lead to current sequencing platforms generating increasingly higher volumes of data at faster speeds than ever before. A recent example is the Illumina sequencing machine, which can produce more than a terabyte of data per day [2]. The ability to sequence DNA quickly and cost effectively is providing us with a new ability to understand the biological world around us at a much deeper level. These advancements have shifted research in the fields of genomics, medicine, and computational biology to a new era of possibilities.

Current sequencing platforms are limited despite vast improvements with DNA sequencing data [3, 4]. DNA sequences are often released with low error rates (1-2%) as sets of short unordered reads, which greatly limits the ability to accurately reconstruct the initial sequence, due to its complex and highly repetitive nature. These challenges complicate downstream analysis (e.g. clinical applications). Single-molecule sequencing (SMS) technology has been able to address some of these challenges [5, 6, 7]; however, they operate with larger sequencing error rates (10-15%) and are still prohibitively expensive.

While $genome$ is the complete set of genes or genetic material present in a cell or organism, $metagenome$ usually refers to the collection of microbial genomes present in a sample. This dissertation presents several methods designed to leverage both genomic and metagenomic data features, in order to guarantee efficiency and scalability.

Genomic data analysis has completely revolutionized how genetic variants are studied with the advent of sequencing technologies making it possible to survey variants or mutations across an entire genome within a reasonable amount of time. The technology

itself has undergone a series of paradigmatic shifts and keeps improving in terms of speed, accuracy, and cost-efficiency. This technological breakthrough promises to transform the field of biomedicine by opening new opportunities for studying genetic variants and mutations at the population level. It will be crucial to implement well-designed computational algorithms to fully capitalize on this innovations.

In metagenomic data analysis, samples sometimes contain hundreds or even thousands of various microbes. The human body carries more than a thousand various organisms, so sequencing produces short reads corresponding to a mixture of various genomes, since there are more microbial cells than human cells. There is no single reference with which these reads can be compared. Although we have reference genomes for numerous microorganisms, unknown species may form a large portion of samples, which results in a very challenging time figuring out which microbial species are present or even how many various genomes there are.

The algorithms presented in this dissertation ultimately serve to answer the following concept: Given reads sequenced from a genome, what can we conclude about that genome? Multiple directions are possible to target this question, depending on various points of view on vast different data and techniques. On one hand, if reads are sequenced from a known reference, we have a resequencing study. Techniques can be used such as read-to-reference mapping and variant profile comparison. On the other hand, if reads are from an organism with an unknown reference, de novo assembly must be used to study these unknown organisms. Metagenomic studies focus on reads represented by a mixture of genomes. Besides the techniques used for each of the above research categories, data comprehension is also important, since its characteristics can significantly affect end results.

We present algorithms relevant to read-to-reference alignment performance predictions, resequencing studies, and metagenomic applications.

1.1 Contributions

The contributions of this dissertation are presented below.

- We investigated the extent to which the complexity of genomic sequences affects the performance of short read aligners. We demonstrated that a proper measure of sequence complexity was essential in studying the relationship between alignment performance and the abundance of repeats in genomes. This finding suggests a novel approach to selecting aligners for new genomes and has great potential for reducing experimental cost.
- We have demonstrated that the current insertion/deletion (INDEL) variants of the human population profile constructed and curated by the 1000 Genome Project exhibits a bias at certain INDEL locations. These locations can be identified by counting the number of optimal alignments between reads containing alternative alleles to the reference genome at those locations. The bias is essentially an effect of either short-read aligners or variant callers having to choose one out of many equally theoretically optimal alignments.
- We developed a memory-efficient implementation – LongAGE – based on the classical Hirschberg algorithm. We demonstrated an application of LongAGE for resolving breakpoints of SVs embedded into segmental duplications on Pacific Biosciences (PacBio) reads that can be longer than 10Kbp. Furthermore, we observed different breakpoints for a deletion and a duplication in the same locus, providing direct evidence that such multi-allelic copy number variants (mCNVs) arise from two or more independent ancestral mutations.
- We proposed a method for detecting unknown bacteria in environmental samples. Our approach is unique in its utilization of short reads only from 16S rRNA genes rather than from entire genomes. We showed that short reads from 16S rRNA genes retain sufficient information for detecting unknown bacteria in oral microbial communities.

- We compared performance of popular metagenomic classifiers on short reads and longer reads, which are assembled from the same short reads. When using a number of popular assemblers to assemble long reads from the short reads, we discovered that most classifiers made fewer predictions with longer reads and that they achieved higher classification performance on synthetic metagenomic data. On real metagenomic data, we observed a similar trend that classifiers made fewer predictions. This suggested that they might have the same performance characteristics of having higher precision while maintaining the same recall with longer reads.

1.2 Outline

This dissertation is structured as follows. Chapter 2 presents machine learning algorithms for predicting performance of short-read aligners using Genome Complexity. Chapter 3 presents analysis of Optimal Alignments that unfolds aligners bias in existing variant profiles. Chapter 4 presents LongAGE, a memory-efficient implementation of Alignment with Gap Excision (AGE). Chapter 5 presents Union-Find like algorithms for detecting unknown microbes along with oral microbial applications. Chapter 6 presents Read Assembly algorithms for producing longer reads as an input to improve species classification in metagenomics. Chapter 7 concludes the dissertation with a brief description of on-going and future work.

Chapter 2

Algorithms for Predicting Short-read Performance

2.1 Background

Advances in next-generation sequencing (NGS) technologies have fostered an active development of computational methods to align short reads to reference genomes [8, 9, 10, 11, 12, 13, 14, 15]. The alignment of short reads to reference genomes plays a critical role in many important applications and workflows that utilize NGS data such as assembling genomes, genotyping, profiling metagenomics samples and measuring gene expression. To find the best performing aligners, a conventional approach is necessary to compare different aligners on selected genomic datasets and pick the best ones based on their overall performance [16, 17]. While this approach can identify high-performing aligners with *studied* genomes, it is uncertain how accurately the aligners will perform on new genomes. In other words, the most reliable way to find the best aligners for a new genome has been simply trying different aligners on that particular genome and picking the best ones. But experimenting with different software configurations and parameters to find the best aligner for a specific genomic dataset is often time consuming. As a result, researchers often adopt a well-known aligner and presume that it is appropriate enough for any type of genomes.

Yu *et al.* [18] evaluated several aligners and reported that long repeats seriously degraded their performance. Algorithmically, the abundance of repeats makes it difficult for aligners to map reads to correct chromosomal locations. Genetic variants and sequencing errors result in repeats that are approximately matched. Consequently this complicate mapping and alignment of short reads even further. Efforts have been taken to identify regions of genomes that are difficult to map reads accordingly. Lee and Schatz [19] assigned a mappability score to each genomic position and were able to identify up to 14% of the human, mouse, fly and yeast genomes that would be difficult to analyze with short reads. Although mappability scores get improved with longer reads, Li *et al.* [20]

showed that the diminishing return was about 200; reads longer than 200 would not improve significantly mappability of the human genome.

Although the negative effect of repeats on the performance of aligners have been widely observed, no serious effort has been taken to study and exploit the relationship between repeat abundance and aligner performance. The reason for this might be partly due to the lack of a appropriate formulation of sequence complexity for the specific purpose of studying short-read alignment. Complexity of sequences has been studied extensively. Lempel and Ziv [21, 22] formulated the notion LZ-complexity and related it to how much sequences can be compressed. Nan and Adjeroh [23] explored several measures of complexity and found that the *linguistic complexity* (LC) was useful in its ability to help identify known biologically relevant relationships. Recently, Becher et. al [24] introduced the I-complexity, which is defined in terms of discrete logs of longest common prefixes of consecutive sorted suffixes. The authors demonstrated that the I-complexity was close to the LZ-complexity. Furthermore, LZ-complexity and I-complexity of a sequence are related to the number of different substrings and thus the number of repeats of the sequence. Other studies [25, 26, 27] introduce interesting ways to visualize and characterize complexity of genomes in terms of k-mer frequencies and spectra.

2.2 Methods

2.2.1 Formulations of Sequence Complexity

Lempel-Ziv complexity: The LZ-complexity [21, 22] measures the degree of randomness in sequences and as such it can be used to compress sequences effectively. The LZ-complexity is defined as the number of different patterns in a sequence when it is scanned from left to right; we used the version introduced by Lempel and Ziv in 1978 [22]. For example, the sequence **ACTACGTT** has complexity 6 because there are 6 different patterns (A, C, T, AC, G, TT) when the sequence is scanned from left to right. The manner of left-to-right scanning *does not rewind*, which means, for example, ACT is

a substring but it is not considered as one of the different patterns accounted by the complexity measure. We normalized the LZ-complexity by dividing it by the maximum number of patterns a sequence of given length could possibly get.

This complexity has broad applications and in particular provided meaningful interpretations in a biomedical context [28].

I-complexity: Becher and Heiber, [24], introduced this measure to account for the number of different substrings of a sequence. For example, the different substrings of the sequence above are: A, C, G, T, AC, CT, TA, CG, GT, TT, ACT, CTA, TAC, ACG, CGT, GTT, ACTA, CTAC, TACG, ACGT, CGTT, ACTAC, CTACG, TACGT, ACGTT, ACTACG, CTACGT, TACGTT, ACTACGT, CTACGTT, ACTACGTT. All repeats of a substring are counted only once. The I-complexity is not exactly the number of different substrings, but it does account for it. It is defined as follows:

$$I(g) = \sum_{i=1}^{|g|} \log_4(LCP[i] + 1) - \log_4(LCP[i] + 2)$$

where LCP is the array storing the lengths of the longest common prefixes of consecutive sorted suffixes of the sequence g. It was shown [24] that for a DNA sequence s, $\frac{LZ(s)}{8} \leq I(s) \leq LZ(s)(\log_4 |s| + 1)$.

Linguistic complexity: Troyanskaya et al. [29] introduced the linguistic complexity and used it to study complexity profiles of prokaryotic genomic sequences. Nan et al. [23] found that the linguistic complexity was a good measure of biological sequences due to its ability to help identify biological relationships. The linguistic complexity of a sequence g is defined as the ratio of the number of distinct substrings in g to the maximum possible number of distinct substrings in g. Mathematically, let $f(x)$ be the number of occurrences of a substring x in g. Then,

$$LC(g) = \frac{|\{x : f(x) > 0\}|}{\binom{|g|}{2}}$$

Repeat complexity We define the repeat complexity of a sequence g as the ratio of all repeats of length k to all substrings of length k in g:

$$R_k(g) = \frac{\sum_{x \in g: f(x)>1, |x|=k} f(x)}{|g| - k + 1}$$

where $f(x)$ denotes the number of occurrences of x in g. The number of substrings of length k in g is $|g| - k + 1$. More generally, we can define the repeat complexity of a chromosome c_i as part of a genome g as

$$R_k(c_i) = \frac{\sum_{x \in c_i: F(x)>1, |x|=k} f(x)}{|c_i| - k + 1}$$

where $F(x)$ denotes the number of global occurrences of x in g. Note that the occurrences of each repeat x in c_i are counted over the entire genome g, i.e. across all chromosomes, not just in chromosome c_i.

These complexity measures are *constant* in the sense that the complexity of a sequence is always the same. This works fine for text compression since the degree of compressibility does not rely on external parameters. In contrast, the alignment of reads to genomes depends on external parameters introduced by sequencing technologies. In particular, the length of reads can affect greatly the performance of alignment.

In contrast with I-complexity and LZ-complexity, R_k is not a *constant*; a different value of k results in a different complexity value.

2.2.2 Calculation of Repeat Complexity

The LZ, I and LC complexity can be computed efficiently. The repeat complexity of a sequence g, $R_k(g)$, can also be calculated optimally in linear time and space. To accomplish this, we rely on two special data structures called the suffix array (SA) and array (LCP). The suffix array of a sequence stores indices of sorted suffixes of the sequence. The lcp array stores the lengths of the longest common prefixes of consecutive

entries in the suffix array. SA and LCP can be constructed efficiently in linear time and space [30, 31].

The total number of repeats of length k can be counted by traversing the LCP array once while keeping track of intervals that correspond to occurrences of repeats. Specifically, the number of repeats of length k is equal to $\sum_{[i,j]\in I}(j-i+2)$, where I is the set of intervals $[i,j]$'s in LCP such that:

1. $LCP[u] \geq k$ for $i \leq u \leq j$
2. $LCP[i-1] < k$ unless $i = 1$
3. $LCP[j+1] < k$ unless $j = |g|$

This procedure is correct because each interval $[i,j]$ satisfying these properties corresponds uniquely to all occurrences of exactly one repeat of length k. This repeat is precisely the prefix of length k shared by all suffixes in SA in positions specified by this interval. Thus, the number of occurrences of this repeat is exactly $j-i+2$. Summing over all such intervals accounts for all occurrences of repeats of length k in g.

The same procedure can be modified slightly to compute, $R_k(c_i)$, the repeat complexity of a chromosome as part of a genome g. To do so, concatenate all chromosomes into a whole genome, g, to which the same procedure can be applied. Then, while traversing through suffixes in each interval $[i,j]$ that satisfies the three properties, add $(j-i+2)$ to the counter of the chromosome that contains each suffix.

2.3 Results

2.3.1 Short-read Aligners and Their Performance

To investigate the relationship between short-read alignment performance and sequence complexity, we considered many existing aligners and multiple aspects of alignment performance [32, 8, 9, 10, 11, 12, 13, 14, 15, 33]. These approaches employ various heuristics for building indexes of genomes (e.g. q-grams, FM index, hash table) for efficient search of exact matches between reads and genomes. These exact matches, also known as *seeds*, are extended to align reads to genomes. After a careful evaluation,

we narrowed down to only aligners that could finish aligning large read datasets of an entire genome on a high-performance cluster within several days without stalling or crashing in any instances. With this requirement, we were left with five aligners: Bowtie2 [11], BWA-SW [12], CUSHAW2 [13], SeqAlto [10], and Smalt [15]. These five aligners were used in our experiments to study how sequence complexity were correlated to short-read alignment performance. Representative performance of each aligner was obtained from default parameters provided by its software package. We are *not* interested in the best performance of each aligner, but rather in *the correlation* between each aligner performance and sequence complexity.

Performance of a short-read aligner can be defined in different ways. In this work, we considered three different aspects of alignment performance:

1. **Precision**, which is the percentage of correctly mapped reads out of all reads that are aligned,
2. **Accuracy**, which is the percentage of correctly mapped reads out of all reads, and
3. **Coverage**, which is the portion of the genomic sequence covered by mapped reads. We want to distinguish coverage from *sequencing depth*, which is the *expected* number of times aligned reads would cover the genome under the assumption that reads are uniformly distributed. A sequencing depth is typically much larger than 1 (e.g. 50x) , whereas actual coverage by mapped reads is at most 1.

All performance measures have range between 0 and 1 with 0 being the worst and 1 being the best.

2.3.2 Genomic data

Genomic sequences were obtained from The European Bioinformatics Institute (EBI) and the National Center for Biotechnology Information (NCBI) databases. More

information is available in the Supplement[1]. We utilized the SAMtools package [34] to simulate reads, with various realistic parameters, from a diverse dataset with 100 genomic sequences, which are chromosomes and long contigs from bacteria, plants, and eukaryotes. "N" bases were removed from these genomic sequences because they were not real contents and constituted false long repeats that inappropriately affected the true complexity. For each genome, 2x (expected) coverage of reads at lengths 50, 75, and 100 were generated using the $wgsim$ program. Single-end reads were generated with sequencing error rates 0.5%, 1%, and 2%; mutation rates varying from 0.1% to 1%. By default, 15% of mutations are indels.

To measure coverage, we also created a second dataset consisting of real reads from 54 genomic sequences, including 24 chromosomes of *homo sapiens* (humans), 20 chromosomes of *glycine max* (soybean), and 10 chromosomes of *zea mays* (corn). Reads are obtained at http://sra.dnanexus.com. *Zea mays* data has id SRR801164. *Glycine max* data has id SRR596509. *Homo sapiens* data has id ERR251193.

2.3.3 Determining a Suitable Measure of Complexity

We investigated the suitability of the Lempel-Ziv complexity (LZ), I-complexity, linguistic complexity (LC) and repeat complexity (R_k) for studying the performance of short-read aligners. A suitable measure of complexity must have high correlation to different aspects of short-read alignment performance, namely accuracy, precision and coverage. Correlation for an aligner's performance is done as follows. The aligner is used to align reads to genomic sequences in the dataset and the performance (accuracy, precision, coverage) for each sequence is measured. Thus, for each aligner and complexity measure, we have two sets of data representing two variables: P, the performance of genomic sequences, and C, the complexity of those genomic sequences. Linear correlation was used to correlate these two variables. Linear correlation between two variables is quantified in the Pearson correlation coefficient R, whose value is

[1] http://github.com/vtphan/repeat-complexity

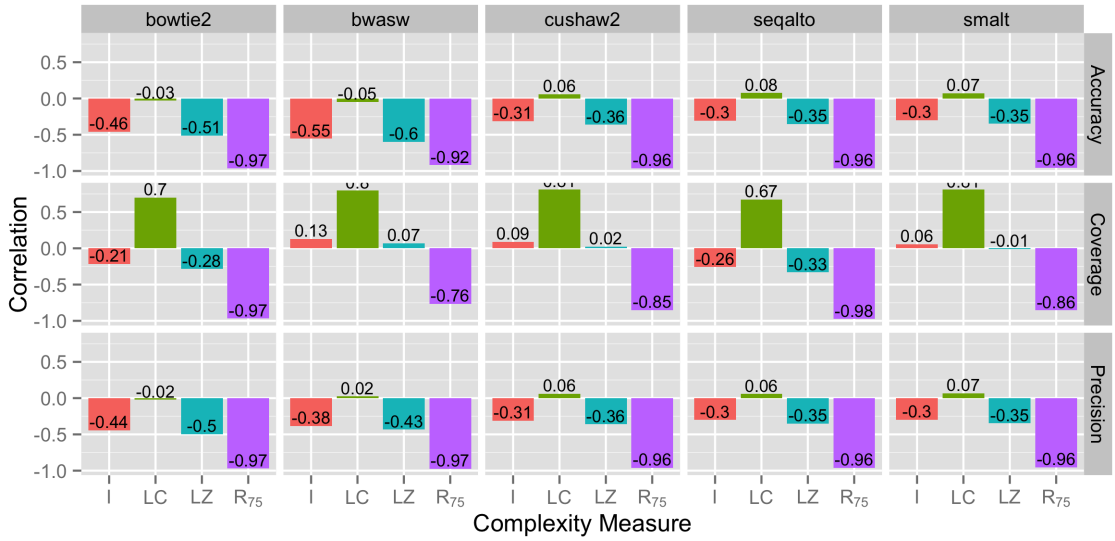

Fig. 2.1: Correlation between performance of short-read aligners and complexity measures. Correlation is measured by linear coefficient, which ranges between -1 and 1. R_{75} has the highest correlation with performance (accuracy, coverage, and precision) across all aligners.

between -1 and 1. If $R = 0$, there is no correlation between the two variables. If R is 1 (or -1), the two variables are maximally positively (or negatively) correlated. Generally, $R \geq 0.75$ is considered a high correlation.

For precision and accuracy, we used simulated reads; this is necessary because we needed to know correct locations of reads in the sequences. For chromosomal coverage, we used real reads to capture the true underlying distribution of reads in chromosomes.

First, we observed that the I and LZ complexity measures showed moderate to low negative correlation with precision, accuracy and coverage across all five short-read aligners (Bowtie2, BWA-SW, CUSHAW2, SeqAlto, and Smalt). The LC complexity showed moderately high positive correlation with chromosomal coverage, but almost no correlation at all with accuracy and precision. This is shown in Figure 2.1.

We also found that R_{75} correlated highly or very highly to all three aspects of performance, precision, accuracy and coverage, across all five aligners. The strong negative correlation implies that the more repeats of length 75 there are in these genomic

sequences, the worse these short-read aligners perform. This finding suggests that with an appropriate choice of k, repeat complexity, R_k, can be the most suitable measure of complexity for the purpose of studying the performance of short-read aligners.

2.3.4 Finding an Optimal k for Repeat Complexity R_k

Given a dataset with reads of certain length, an optimal choice of repeat length (k) yields the highest correlation between repeat complexity (R_k) and the performance of aligning short reads to reference genomes. Shown in Figure 2.1, R_{75} strongly correlated with all three performance measures across all aligners. To investigate the effect of different repeat lengths, we correlated $R_{25}, R_{50}, R_{75}, R_{100}$ and R_{125} to the performance of aligning reads of length 100 to reference genomic sequences. This is shown in Figure 2.2. We could see that although all R_k's had moderate to high correlation with performance, R_{75} had the strongest negative correlation across all three performance measures and across all aligners.

We suspected that the reason that $k = 75$ yielding the highest correlation might be closely related to the fact that in these datasets read length was 100. In fact, we found that the optimal value of k, which gave an R_k with highest correlation to performance, varied slightly for different read lengths. For example, for datasets with read length 50 and 75, we found that R_{50} had the highest correlation with performance compared to R_{25}, R_{75}, R_{100} and R_{125}, whereas for datasets with read length 100, as reported early, R_{75} was found to have the highest correlation. Table 2.1 shows the optimal R_k at a given read length 50, 75, or 100. The Pearson correlation coefficient r's across all five aligners are mostly in the range of -0.95 to -0.98.

2.3.5 The Effect of Sequence Quality and Sequencing Errors

Strong correlation between R_k and alignment performance found in experiments reported in Figures 2.1, 2.2 and Table 2.1 was found on simulated reads with default sequencing errors (to measure precision and accuracy) and real reads of high quality (to measure coverage). To measure the effect that sequencing errors had on the correlation

Table 2.1: Correlation between R_k and precision or accuracy at read lengths 50, 75 and 100. For read lengths 50 and 75, R_{50} was chosen. For read length 100, R_{75} was chosen.

	Corr. with Precision			Corr. with Accuracy		
read length	50	75	100	50	75	100
Bowtie2	-0.97	-0.95	-0.89	-0.98	-0.95	-0.89
BWA-SW	-0.98	-0.96	-0.94	-0.74	-0.80	-0.85
CUSHAW2	-0.98	-0.98	-0.95	-0.98	-0.98	-0.95
SeqAlto	-0.97	-0.97	-0.95	-0.98	-0.97	-0.96
Smalt	-0.96	-0.97	-0.95	-0.97	-0.97	-0.95

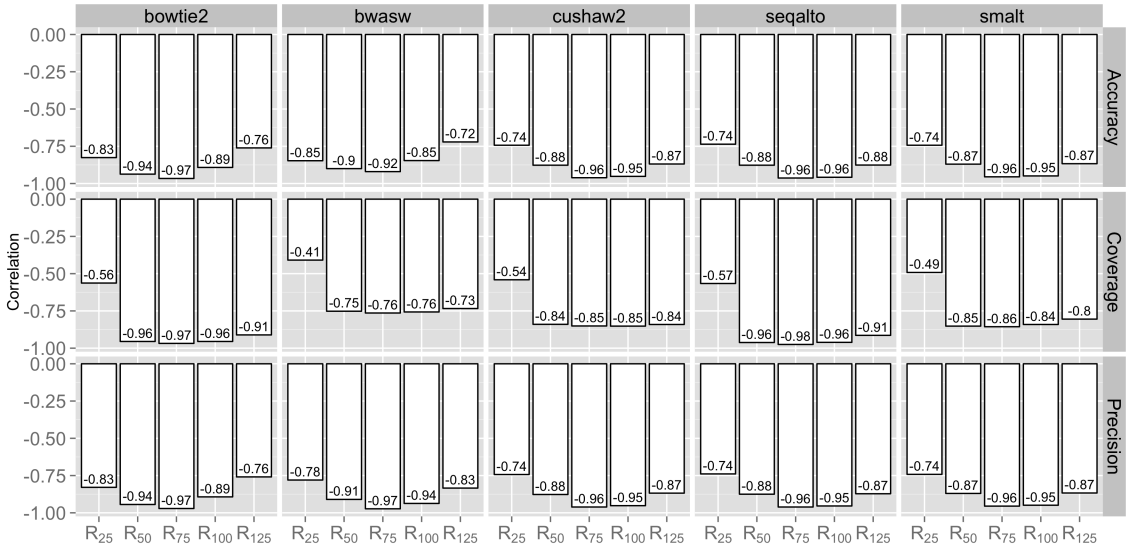

Fig. 2.2: Correlation between performance of short-read aligners and R_k on read datasets of length 100. Compared to the other, R_{75} correlated most strongly to performance (accuracy, precision, coverage) across all aligners.

Table 2.2: Correlation between R_k and precision or accuracy at different sequencing error rates (0.5%, 1% and 2%).

	Corr. with Precision			Corr. with Accuracy		
base error rate	0.5	1.0	2.0	0.5	1.0	2.0
Bowtie2	-0.96	-0.98	-0.97	-0.96	-0.98	-0.97
BWA-SW	-0.95	-0.96	-0.97	-0.96	-0.98	-0.97
CUSHAW2	-0.95	-0.96	-0.96	-0.95	-0.96	-0.96
SeqAlto	-0.96	-0.95	-0.96	-0.96	-0.95	-0.96
Smalt	-0.96	-0.96	-0.96	-0.96	-0.96	-0.96

between alignment performance and repeat complexity, we simulated reads of length 100 using SAMtools [34] at different sequencing error rates ranging from 0.5%, 1% and 2%.

We found that sequence quality or sequencing errors did not affect the correlation between repeat complexity and aligner's performance in terms of precision and accuracy very much. As summarized in Table 2.2, at each sequencing error rate, we found the correlation between the best R_k and alignment performance in terms of precision and accuracy were all very strong at between -0.95 and -0.98.

2.3.6 Prediction of Performance based on Repeat Complexity

To build a model for an aligner, first, select a set of sequences with diverse complexity. Next, choose an appropriate value of k and R_k, where k is similar to read length. Then, measure alignment performance and complexity for each sequence in the dataset, producing a list $(p_1, c_1), \cdots, (p_m, c_m)$, where p_i and c_i are, respectively, the performance value and complexity value of sequence i. Having constructed such a linear regression model, to predict the aligner's performance for an unknown sequence, one computes the complexity of the sequence, uses the linear equation obtained from the model to interpolate alignment performance. To reduce variability, we employed 2-fold cross validation with repeated random subsampling by repeating training and testing 100 times.

Table 2.3: Prediction error of linear models for different aligners.

	Bowtie2	BWA-SW	CUSHAW2	SeqAlto	Smalt
Pre	0.00093	0.00073	0.00067	0.00068	0.00064
Acc	0.00099	0.00105	0.00066	0.0007	0.00069
Cov	0.01746	0.03973	0.03011	0.01832	0.02652

Table 2.3 shows prediction errors of linear models for each performance level precision (Pre), accuracy (Acc) and coverage (Cov) of all aligners.

We observed that prediction errors, across all five aligners, were very small for precision and accuracy. Across all aligners, prediction errors were less than 0.01% for both precision and accuracy. For coverage, prediction errors were still between 1 and 4%.

Bowtie2's and SeqAlto's models had lower errors than the other aligners' models. A closer examination of aligners' correlation with performance (Figures 2.1 and 2.2)

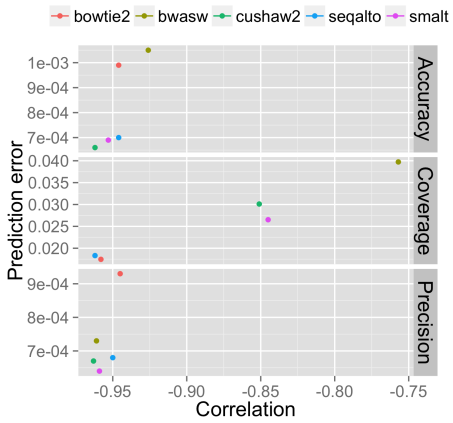

Fig. 2.3: Correlation versus prediction error. Stronger (negative) correlations between alignment performance and repeat complexity lead to smaller prediction errors.

shows that across all 3 performance measure, especially coverage, these two aligners had as high as or higher correlations compared to the others. More generally, as depicted in Figure 2.3, we observed that higher correlations between performance and repeat complexity would lead to more accurate predictions of performance based on complexity.

2.3.7 Selecting Aligners based on Repeat Complexity

There are many factors involved in adopting a computational tool for a job. Software quality, reliability, maintenance and sustainability are among important reasons that affect the adoption of a piece of software. Within the scope of this work, we focus strictly on the performance of short-read aligners. Even within the realm of performance, there are different aspects, among which precision, accuracy and coverage are three important ones. With respect to choosing an aligner, a simple approach, e.g. [16, 17], is simply to choose the best performing aligner on a well-chosen dataset. The assumption is that a high-performance aligner on a known dataset will *similarly* perform highly on a new dataset.

Our finding on the correlation between alignment performance and repeat complexity suggests a novel approach to choosing aligners: based on complexity of the genomes of interest. To illustrate this, suppose that we would like to determine the best

Table 2.4: Average performance of aligners on the two datasets with simulated reads (to measure precision and accuracy) and real reads (to measure coverage).

	Precision	Accuracy	Coverage
Bowtie2	0.997	0.989	0.909
BWA-SW	0.997	0.995	0.867
CUSHAW2	0.998	0.997	0.913
SeqAlto	0.998	0.992	0.898
Smalt	0.997	0.997	0.923

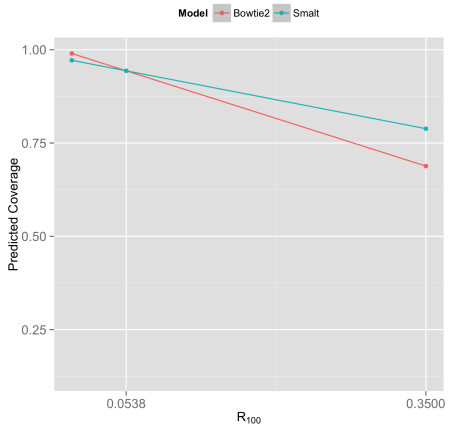

Fig. 2.4: Linear models of coverage performance. Bowtie2's and Smalt's models are, respectively, defined by the equations (i) $y = -0.8617x + 0.9898$ and (ii) $y = -0.5234x + 0.9716$. All sequences in our datasets have repeat complexity (R_{75}) less than 0.35.

aligner among our tested five based on their performance under our datasets. The average performance of the five aligners under the datasets is summarized in Table 2.4. Based on the average performance of aligners in these datasets, one could easily recommend Smalt as the overall best performer. In terms of coverage, Smalt's performance is higher than that of any other tested aligner. Although the average performance on these two datasets suggests that Smalt will perform well on new datasets, it does not reveal much about the characteristics that make Smalt performs well. In other words, although it is less probable, it is entirely possible that Bowtie2 might have higher performance in terms of coverage than Smalt on a new genome, even though on our dataset Bowtie's average coverage (0.909) is lower than Smalt's (0.923).

In fact, a close look at linear models of Smalt and Bowtie2, as shown in Figure 2.4, reveals that neither one of these is an absolute winner. The best choice depends on the repeat complexity of genomic sequences in the new dataset. Specifically, for sequences with repeat complexity less than 0.0538, Bowtie2 is predicted to yield higher coverage, and for sequences with complexity higher than 0.0538, Smalt is predicted to yield higher coverage.

2.4 Discussion

Researchers have long known that the abundance of repeats is detrimental to the performance of computational methods in NGS experiments and extra cost and care must be used to obtain satisfactory results. For instance, to sequence genomes known to have a high number of repeats, large numbers of reads (e.g. sequencing depth larger than 50x) must be used to ensure high coverage (closer to 1) and extra care must be exercised to assemble long contigs. But beyond this intuitive understanding and practical rules of thumb, no formal analysis has been proposed to study the relationships between the complexity of genomic sequences in terms of abundance of repeats and the performance of computational methods.

An important contribution of this work is that it assists researchers in understanding better aligners' performance on new genomes and consequently make more informed decisions on selecting appropriate aligners for a specific genome. Determining if aligner A performs better than aligner B can be very hard; A can give slightly better result than B for some sequences but slightly worse for others. The more relevant question is "*Will A perform better than B for a specific genome*?" The answer can be obtained experimentally: tweak and experiment with parameters of A and of B, and use each set of parameters to align millions of reads to that genome. The problem with this approach is that it is impractical. Each run of millions of reads is already time consuming (both in execution time and manual work to process the results). Attempting many runs with different parameters to compare A and B properly for each specific genome is simply not

practical. As a consequence, in practice, each research group tends to adopt an aligner that is deemed to be good enough and use it for all of their data.

We demonstrated how R_k could be used to select aligners that would be likely to perform well on unknown genomic sequences. Bowtie2 and SeqAlto had good performance on tested datasets and at the same time they had high correlation between performance and repeat complexity. Thus, their performance should be similarly high with unknown sequences. In contrast, one should be cautious in choosing aligners that perform well on tested datasets, but have lower correlation with R_k.

Accurate prediction of alignment performance can have an impact on experimental designs and analyses that are based on NGS data, since short-read alignment is an essential component of many important computational tasks such as genome assembly, genotyping, and gene expression measurement. In particular, performance measures such as accuracy and coverage might help reduce experimental costs. Accuracy of alignment (the percentage of correctly aligned reads out of all reads) gives a hint to how much reads are wasted by an aligners. Thus, an accurate prediction of alignment accuracy can help researchers estimate better an appropriate amount of reads needed for an experimental design. Further, alignment coverage (the percentage of genomes covered by aligned reads) can also help researchers estimate how many reads will be needed to cover most parts of genomes. In other words, this work opens up opportunities for further investigations into how to make better prediction of alignment performance and how to use such accurate prediction to actually reduce experimental costs.

The notion of repeat complexity being directly connected to the degree of difficulty of genomic computational analysis is analogous to the notion Shannon entropy [35] and Lempel-Ziv complexity [21] being directly related to the degree of randomness and compressibility of texts. R_k might be used to correlate with performance of other computational problems. For problems that require complex algorithmic strategies, the

correlation might not be very strong, and the notion of sequence complexity might need to be modified. But, this work shows that such a task is possible.

Chapter 3

Optimal Alignment Algorithms for Aligners' Bias in Existing Variant Profiles

3.1 Introduction

The International HapMap Project and 1000 Genomes Project [36, 37] produced over 10 million single nucleotide variations (SNV) and approximately one million insertion/deletion (INDEL) of the human population. This resource has been utilized to develop a nearly complete map of haplotypes of the human genome [38] and to discover the great extent to which diseases are affected by human genetics. Various approaches for detecting variants have been developed [39, 40, 41, 42, 43, 44, 45, 46, 47]. These variant callers often rely on external tools which align short reads to a reference genome to detect genetic variants. For example, the popular variant caller framework GATK [39] often used an external aligner known as BWA-SW [12] to align reads to reference genomes.

Although methods of aligning reads to genomes are diverse, they are essentially based on two important steps: finding seeds (which are exact matches between a substring of a read and substrings of the genome) and extending seeds into full alignments. Further, the extension of seeds into a full alignment often utilizes a technique based on the local pairwise sequence alignment [48]. Variant callers utilized alignments produced from aligners to call genetic variants that are different from the reference genome. In essence, each difference (substitution or gap) in a correct alignment results in a variant call (SNP or INDEL). Unfortunately, the basic algorithm of pairwise alignment does not account for multiple optimal alignments, each of which might result in different variant calls. From the theoretical point of view, each of the optimal alignments is equally likely to be the correct biological alignment. Thus, the choice of one optimal alignment over another is purely arbitrary.

In this chapter, we demonstrate that many popular aligners can be divided into two groups. The first group of aligners produce alignments that would result in INDEL calls that agree with those reported in existing variant profiles, such as the resources curated by

the 1000 Genomes Project. The second group of aligners produces alignments and INDEL calls that *disagree* with those reported in existing variant profiles. This finding implies that thousands of INDELs that have been reported in public resources were constructed based on algorithmic bias of alignment strategies. This source of bias adds to the list of biases in variant calling caused by sequencing technologies or coverage [49, 50]. It presents a problem for researchers who presume existing resources of human genetic variants as a gold standard for studying genetic variants.

3.2 Methods

Methods that determine genetic variants from NGS data by and large rely on computational methods that align short reads to reference genomes and detect differences between them. The task of aligning short reads to genomes consists of two separate steps: (1) mapping reads to correct chromosomal locations and (2) aligning reads correctly to those chromosomal locations. A read can be correctly mapped and incorrectly aligned. Misalignment at a correct chromosomal location can affect the determination of insertion-deletion variants (INDEL). An INDEL is represented in the form $x_1|x_2|\cdots|x_k$, which means that at that location the string x_1 appears in the reference genome, and any of $x_1, x_2 \cdots x_k$ can appear in another genome at that location.

To see how a read can be correctly mapped and incorrectly aligned, consider an example, in which the read TCAGG is correctly mapped to the genome at location p, and that the substring starting at this location of length 8 is TCACACAG. Depending on the model of alignment, there are two or three different *optimal* alignments:

```
TCACACAG        TCACACAG        TCACACAG
T--CA--G        TCA----G        T----CAG
```

The first alignment results in 2 INDEL calls: TCA|T at location p and ACA|A at location $p + 4$. The second alignment results in an INDEL call ACACA|A at location $p + 2$. And the third alignment results in an INDEL call TCACA|T at location p.

In an alignment model where gap extensions and openings are equally penalized, these three alignments are all optimal because the gaps in each alignment equate to a deletion of 4 bases. In a model such as the *affine gap* model, in which a gap opening is penalized more than a gap extension, however, there are only two optimal alignments (the second and third) because the first alignment would be penalized more than the other two. So, even in the more sophisticated affine gap model, there can be multiple optimal alignments, resulting in different INDEL calls. And if an aligner picks one of these based on some algorithmic bias, this bias will end up in a biased calling of INDEL.

The goal of this work is to examine known INDEL locations and determine if those locations permit multiple optimal alignments. Further, for INDEL locations that permit multiple optimal alignments, we aim to examine the possibility that they were constructed in a biased manner based on biased alignments of many popular short-read aligners.

3.2.1 Pairwise Alignment

The mechanism by which aligners can create a biased alignment can be seen more easily by an examination of the basic pairwise alignment algorithm [48]. Although different alignment methods have different ways to speed up the mapping of reads to genomes, e.g. using an FM index or a hash table, the alignment itself is essentially the same formulation of optimal pairwise alignment, based on dynamic programming.

In a simple alignment model with no penalty for gap opening, an optimal alignment between $x = x_1 \cdots x_n$ and $y = y_1 \cdots y_m$ is found by constructing a matrix M, in which $M[i, j]$ is the score of an optimal alignment between $x_1 \cdots x_i$ and $y_1 \cdots y_j$, for $1 \leq i \leq n$ and $1 \leq j \leq m$. With $M[i, 0] = i$ and $M[0, j] = j$, the matrix M is constructed based on the following relation:

$$M[i,j] = \max \begin{cases} M[i-1, j-1] + match(x_i, y_j) \\ M[i-1, j] + \epsilon \\ M[i, j-1] + \epsilon \end{cases} \tag{3.1}$$

where $match(x_i, y_j)$ is the cost of substituting x_i for y_j and ϵ is the cost of deleting x_i or inserting y_j.

In the affine gap model, finding an optimal alignment between x and y depends on the computation of three matrices M, X, and Y. Here, $M[i, j]$ is the score of an optimal alignment between $x_1 \cdots x_i$ and $y_1 \cdots y_j$, where x_i is aligned with y_j. $X[i, j]$ is the score of an optimal alignment in which x_i aligns with a gap. And, $Y[i, j]$ is the score of an optimal alignment in which y_j aligns with a gap. The computation of the three matrices can be done based on the following relations:

$$M[i,j] = \max \begin{cases} M[i-1, j-1] + match(x_i, y_j) \\ X[i,j] \\ Y[i,j] \end{cases} \tag{3.2}$$

$$X[i,j] = \max \begin{cases} M[i-1, j] + (\epsilon + \rho) \\ X[i-1,j] + \epsilon \end{cases} \tag{3.3}$$

$$Y[i,j] = \max \begin{cases} M[i, j-1] + (\epsilon + \rho) \\ Y[i,j-1] + \epsilon \end{cases} \tag{3.4}$$

where ϵ is the cost of inserting or deleting a base, and ρ is the cost of inserting or deleting the first base (i.e. the penalty for gap opening).

3.2.2 Constructing all Optimal Alignments

In Equations 3.1-3.4, when there exist more than one ways to achieve a maximal value, the choice adopted by an alignment algorithm will be arbitrary. Further, each arbitrary choice of maximal value of each step will lead to a specific optimal alignment. Thus, given the existence of more than one maximal cases to choose from in Equations 3.1-3.4, there will necessarily be multiple optimal alignments, which all have the same alignment scores despite being slightly different from one another.

To construct all optimal alignments under the non-affine gap model after the matrix M is filled, one starts from the entry with the highest cost and retraces all steps at which optimal decisions (as specified in Equation 3.1) are made. The following procedure constructs all optimal alignments in the non-affine model, after the matrix M is computed:

1: Find (i, j) such that $M[i, j]$ is maximum.

2: **return** $Trace(M, i, j)$

Algorithm 1: $Trace(M, i, j)$

1: **if** $i < 0$ or $j < 0$ **then**
2: **return** $\emptyset$
3: **if** $M[i, j] == M[i-1, j-1] + match(x_i, y_j)$ **then**
4: $m \leftarrow Trace(M, i-1, j-1)$
5: Append (x_i, y_i) to each alignment in m
6: **if** $M[i, j] == M[i-1, j] + \epsilon$ **then**
7: $i \leftarrow Trace(M, i-1, j)$
8: Append $(x_i, -)$ to each alignment in i
9: **if** $M[i, j] == M[i, j-1] + \epsilon$ **then**
10: $d \leftarrow Trace(M, i, j-1)$
11: Append $(-, y_j)$ to each alignment in d
12: **return** $m \cup i \cup d$

As described in Algorithm 1, the call $Trace(i, j)$ returns all optimal alignments ending at x_i and y_j. Trace is done by identifying whether each of the three conditions in Equation 3.1 is optimal. If the condition is optimal, Trace is called recursive to obtain all optimal alignments starting at that entry. By induction, the three recursive calls return all possible optimal alignments just before x_i and y_j. Then, the algorithm correctly returns the union of all of the optimal alignments ending at x_i and y_j.

To construct all optimal alignments under the affine gap model, the process is similar. After the matrices M, X, and Y are filled, one starts from the entry of M with maximum value and retraces all the steps at which optimal decisions (as specified in Equations 2,3,4) are made. The only technical difference is that we need to specify the appropriate matrix (either M, X, or Y) in each recursive call.

3.3 Experimental Design

We hypothesize that the usage of an aligner to detect variants will result in incorporating the aligner's bias into the construction of a variant profile. Specifically, this bias will exhibit itself at INDEL locations that have multiple optimal alignments. In our analysis the reference human genome, obtained from NCBI, build GRCh37, and the known variant profile obtained from the Integrated Variant Set release from the 1000 Genomes Project Consortium, we found that among 1,442,639 INDEL locations, 6,685 of them had multiple optimal alignments.

To demonstrate that many of these INDELs were created based on the bias of some alignment algorithms, we set out to reverse engineer the process of determining these INDELs based on various alignment algorithms. In the reverse engineering process, we create a set of reads $\mathcal{R}$ that bear alternative alleles from INDEL locations with more than one optimal pairwise alignments and use each aligner to align these reads to the reference genome. The alignment of each read in $\mathcal{R}$ to the correct INDEL location gives rise to a variant call. By recording the number of variant calls that agree with the known variant profile, we can compare the aligners' degrees of agreement with known variant profiles and detect aligners' bias, if there is any. Specifically, the process works as follows:

1. Suppose that the INDEL location i has two known alleles: A and $ACGA$, where A is in the reference genome, and $ACGA$ is an alternative allele.
2. Suppose the reference genome g is represented as xAy (g_i is A).
3. Let u be a suffix of x, and v be a prefix of y. (Both presumably have length k). In other words, u and v are k-substrings of the genome that are on the left and the right of the allele A.
4. We will create a string $r = uACGAv$. The string r is presumed to be the substring of another genome that differs from the reference genome at the exact location i with allele $ACGA$. We varied the length of u and v between 25

and 50. Thus, the length of the read r is around 50 to 100. (The actual length is equal to the length of u or v plus the length of the INDEL allele at location i.)

5. Now if we align r to the reference genome, and if r is correctly mapped to location i, then two possible optimal alignments can be observed:

```
uA---v          u---Av
uACGAv          uACGAv
```

6. Of these two optimal alignments, the one on the left resulted in the variant $A|ACGA$, which agrees with the known profile. The other alignment resulted in a variant call at location $i-1$ that is different from the known profile. In general, there can be many optimal alignments but only one of them results in a variant call that agrees with the known variant profile.
7. If there are multiple alternative INDEL alleles at location i, each string r is created for each alternative allele.
8. Let $\mathcal{R}$ be the set of strings r's that are constructed as we have described. For each INDEL location with more than one optimal pairwise alignments, there are exactly 10 reads with length between roughly 50 to 100, as described above, yielding a 10x coverage at those INDEL locations. To test whether the existing variant profile consists of INDELs that might have been constructed based on a bias alignment method, we employed several popular short-read aligners to all strings in $\mathcal{R}$.

3.4 Results

To map and align reads in $\mathcal{R}$ to the reference genome, we considered several popular aligners: Bowtie2 [11], BWA-SW [12], CUSHAW2 [13], Smalt [15], SRmapper [51], SHRiMP2 [8], RazerS [52], GASSST [32], SeqAlto [10], Masai [14], and Soap2 [53]. Most aligners employed a seed-and-extend strategy, which first finds exact matches (seeds) between reads and the genome, and then extend such seeds to full alignments between reads and the genome. While these aligners adopt a wide range of

algorithmic techniques in building indexes to facilitate efficient seed finding, the extension phase of their methods is based on the basic local alignment strategy, which is described in Section 3.2. We eliminated four aligners SeqAlto, Masai, Soap2, and SRmapper, due to their inability to map reads in $\mathcal{R}$ to their correct positions. Possible reasons include: (1) reads in $\mathcal{R}$ are relatively short and aligners might have been designed to work effectively with long reads, and (2) these reads might have been mapped to multiple chromosomal locations, and these aligners might have decided not to map any of them due to such confusion. For BWA, we used the BWA MEM version that is designed to work with both short and long reads.

3.4.1 Analysis of INDELs with Multiple Optimal Alignments

The set of reads $\mathcal{R}$ surrounding known INDEL locations were aligned by all aligners to the reference genome. For each aligner, we recorded the percentage of reads in $\mathcal{R}$ that the aligner was able to map to their correct locations. By design, each read covers a specific INDEL. A read is mapped correctly if it overlaps with the INDEL location that it is supposed to covers. Given that a read is mapped correctly, the alignment between the read and the genomic region gives rise to a unique variant call at that INDEL location. If there are more than one optimal pairwise alignments, the choice of which optimal alignment depends on the specifics of each alignment algorithm. As a result, the resulting variant call may or may not be the same with the reported variant profile that was created based on a different alignment algorithm.

Table 3.1: Percentage of correct mapping, actual and expected alignment by aligners

Aligners	Correct mapping %	Actual agreement %	Expected agreement %	p-value
Bowtie2	96	99	30	0.0000546
BWA	93	99	30	0.0000550
SHRiMP2	97	99	31	0.0001491
RazerS	88	75	31	0.0001631
CUSHAW2	97	70	31	0.0000562
GASSST	91	8	17	0.0015892
Smalt	96	5	31	0.0003852

As shown in Table 3.1, most aligners were able to map most reads in $\mathcal{R}$ to their correct INDEL locations with mapping percentages range from 88% to 97%. Mapping a read to its correct INDEL location means that the read is mapped to a chromosomal location that overlaps the INDEL that the read was designed to cover. A correct mapping of a read does not mean that the alignment of the read to this location will yield a variant call that agrees with (or matches) the known variant profile. When there are multiple optimal alignments between a read and genomic fragment, each optimal alignment results in a different INDEL call. An alignment agrees with the existing information, if it produces an INDEL that is the same as the existing known INDEL. Table 3.1 reveals that these aligners can be divided into 3 groups:

1. Aligners whose correctly mapped reads (to INDEL locations with multiple optimal alignments) are aligned in high agreement with the known variant profile, about 99% in agreement. These aligners include Bowtie2, BWA (MEM version), and SHRiMP2;
2. Aligners whose correctly mapped reads are aligned in moderate agreement with the known variant profile (between 70-75%). These include RazerS and CUSHAW2; and
3. Aligners whose correctly mapped reads are aligned in high disagreement with the known variant profiles (less than 10%). These include GASSST and Smalt.

To analyze if there exists alignment bias in reported variant profiles, we compare an aligner's degree of agreement with reported variant profiles to the expected agreement if the algorithmic choice happens by chance. Suppose that at INDEL location i, there are n_i optimal pairwise alignments (under the affine-gap model), then the probability p_i that an aligner produces an alignment that yields a call in agreement with the known variant profile is $\frac{1}{n_i}$. The expected number of agreed calls is also $\frac{1}{n_i}$. Summing over all events, we find that the expected number of instances that agree with the known variant profile is

$\sum_{i=1}^{N} \frac{1}{n_i}$, where N is the number of INDEL locations with multiple pairwise alignments that the aligner can map correctly reads in $\mathcal{R}$ to.

The last column of Table 3.1 shows the expected percentage of agreement by each aligner ($\frac{1}{N}\sum_{i=1}^{N} \frac{1}{n_i}$). We can see that across all aligners, there is a significant different between the expected percentage of agreement and the actual agreement. For example, with Bowtie2, the expected percentage of alignment is 30% compared to the actual percentage of agreement, which is 99%. This vast difference between the expected and actual degree of agreement suggests that variant calls at these INDEL locations were obtained by alignment algorithms that were very similar to those aligners in the first groups (Bowtie2, BWA, SHRiMP2) whose actual percentage of agreement is more than 3 times the expected percentage of agreement. To compute the likelihood of this difference, we calculated the probability that the difference between the actual agreement and expected agreement (as happened by chance) is as much as or even more extreme than what we observed. This p-value can be bounded by the Chebyshev-Cantelli's inequality, $P(X - \mu < \lambda) < \frac{\sigma^2}{\sigma^2+\lambda^2}$, where λ is the observed difference between actual and expected agreement, μ and σ are the expected agreement and its variance. As described above, $\mu = \sum_{i=1}^{N} \frac{1}{n_i}$. Further, $\sigma^2 = \sum_{i=1}^{N} \frac{1}{n_i}(1 - \frac{1}{n_i})$. The very small p-values shown in the last column of Table 3.1 suggest that the difference in actual and expected agreement is extremely unlikely caused by chance.

3.4.2 Characterization of INDEL Complexity

The existence of multiple optimal alignments giving raise to different INDEL calls is an inherent problem. We have demonstrated that in many cases there are more than one theoretically optimal alignment, each of which has the same chance of being biologically correct. It is important to note that there is no correct optimal alignment among all possible optimal alignments: they are all optimal and thus equal probability of being the correct alignment. In other words, it does not matter which optimal alignment an aligner chooses and a variant caller utilizes the aligner's result, there must be inevitably some

bias. The only way to cope with this is for an aligner to report all optimal alignments and for a variant caller to derive all *alternative possibilities* of INDELs from these optimal alignments. This is tedious and not being done in practice. Existing variant profiles do not report alternative possibilities of INDELs; they only report one.

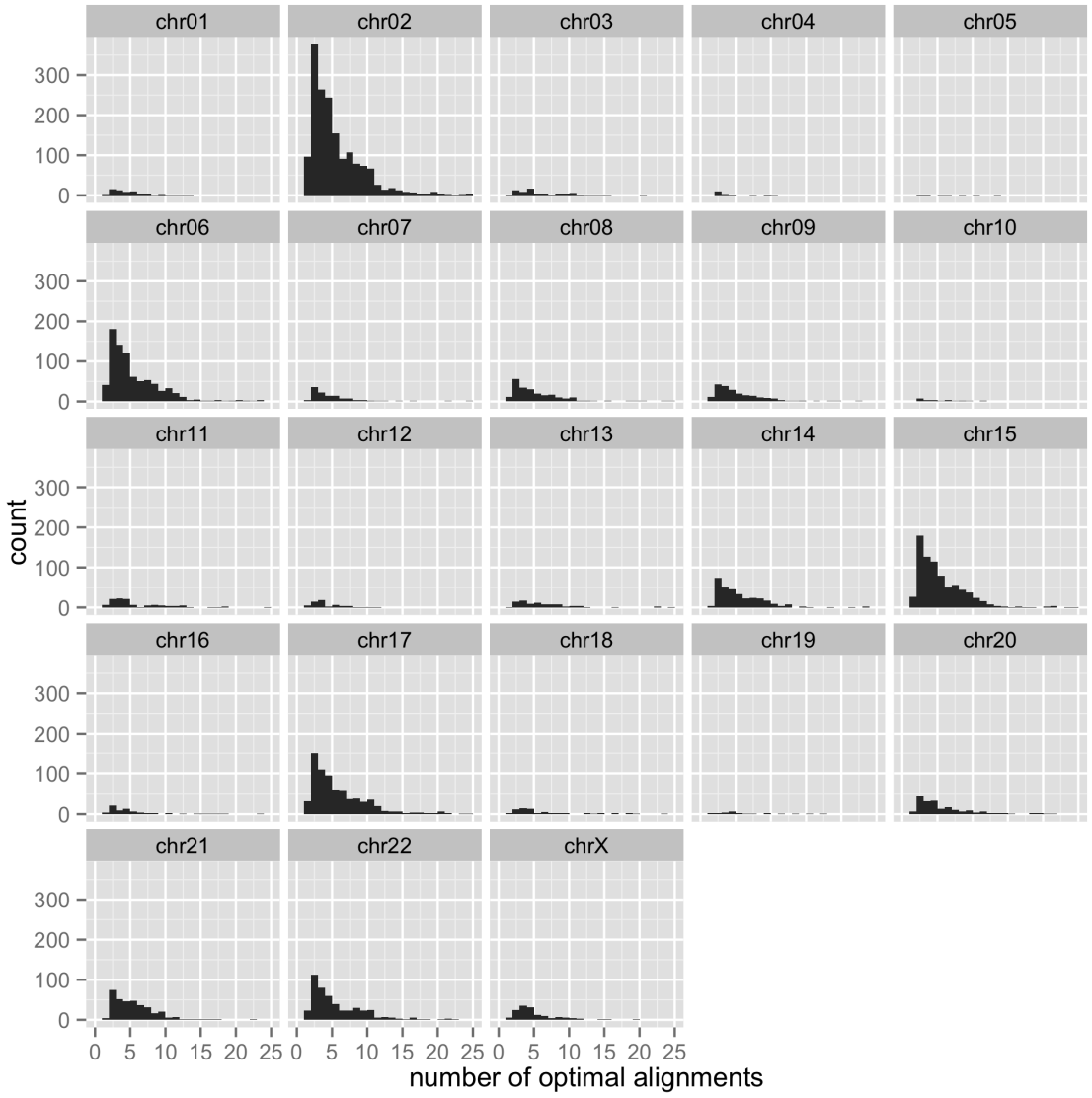

Fig. 3.1: Distribution of INDEL complexity across human chromosomes

Thus, it is useful to examine known INDEL locations and characterize the extent to which they are affected by multiple optimal alignments. We define the *complexity of each INDEL location* as the number of optimal alignments that can be had when reads

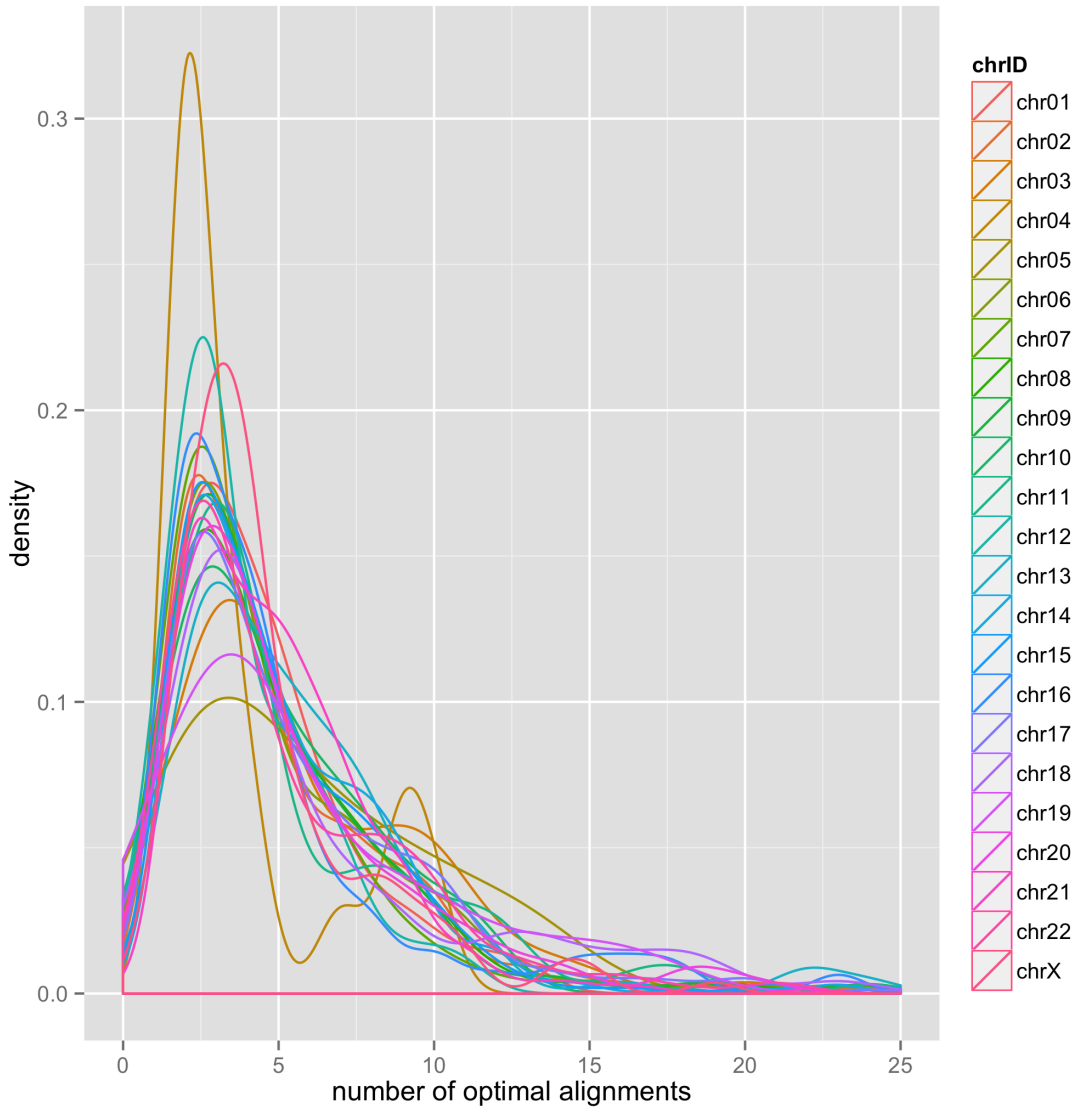

Fig. 3.2: Density of INDEL complexity across human chromosomes

bearing alternative alleles are aligned (under the affine-gap model) to the reference genome at this location. Figure 3.1 shows the distribution of INDEL complexity across human chromosomes. We observed that chromosome Y has no INDEL with multiple optimal alignments. Further, a closer examination of the density of INDEL complexity on all chromosomes, as shown in Figure 3.2, suggests that these distributions are very similar, with the peak occurs at around 3. A majority of these INDELs have 3 multiple optimal alignments. Additionally, chromosomes 2, 6, 15, and 16 stood out with the most number

of INDEL locations with multiple optimal alignments. Larger chromosomes do not necessarily have more complex INDELs. For example, compared to the others, chromosome 1 has fewer INDELs with multiple optimal alignments.

3.5 Discussion

The accuracy of calling variants can be improved by increasing coverage (i.e. using more reads) and realigning reads that overlap INDEL locations. But we argue that neither increasing coverage nor realigning reads around INDELs can help resolve the problem caused by multiple optimal alignments. Increasing reads can reduce the damaging effect of sequencing errors, which occur independently across reads. While realigning reads around an INDEL as described by Li [54] can achieve a better multiple alignment of reads aligned to the INDEL, the multiple alignment is still biased as it is based on one of the optimal pairwise alignments. For instance, recall the example given earlier, in which the read TCAGG is correctly mapped to the genome and is aligned to the genomic sequence TCACACAG. As we showed earlier, there are multiple optimal pairwise alignments. Let us supposed that many reads are aligned to this region. It is possible (due to different chromosomal positions), the alignments of some reads might look like the first alignment in this example; the alignments of some other reads might look like the second alignment; and the alignments of the rest might look like the third alignment. The goal of realigning reads [54], which were pairwise aligned, is to obtain a consistent multiple alignment of reads. The result of such realignment would be an adoption of the same alignment for all reads aligned to this region to obtain a high quality call. But the adopted multiple alignment is still based on one of the three optimal pairwise alignments. As such, the realignment of reads still produces biased results.

We have demonstrated that the current INDEL profile constructed and curated by the 1000 Genome Project exhibits a bias at certain INDEL locations. These locations can be identified by counting the number of optimal alignments between reads containing alternative alleles to the reference genome at those locations. The bias is essentially an

effect of either short-read aligners or variant callers themselves having to choose one out of many equally theoretically optimal alignments. There is no obvious way to "standardize" this phenomenon by designating one optimal alignments as the "canonical" one. As such, it seems the only way to deal with this is reporting all optimal alignments and consequently reporting all alternative INDEL calls as the result of those alignments.

If this phenomenon is not addressed, there can be potential serious problems relating to the analysis and study of INDEL. For example, certain alignment techniques will result in wrong calls at those INDEL locations. Case in point is Smalt, which was able to map 96% of the reads, but very few of the alignments produced the "correct" INDEL calls (as specified by the existing INDEL information). At these location, Smalt was wrong simply because it chooses a different optimal alignment from the one based on which the INDEL was constructed.

Chapter 4

Memory Efficient Algorithms for Alignment with Gap Excision

4.1 Introduction

Recent single-molecule sequencing (SMS) technologies generate very long reads, enabling the capture of multiple variant types including structural and copy number variations (SVs/CNVs). However, precise alignment around SVs is a challenge, because of large gaps in alignment [55, 56, 57]. Previously, Alignment with Gap Excision (AGE) was described as a precise method that uses dynamic programming to solve the problem [58]. However, its application is limited to alignment of short reads/contigs to relatively small genomic regions because its implementation uses matrices, hence it requires vast memory usage. While not intended to be the sole tool used for aligning against a reference genome, its primary purpose is to optimally realign reads around the sites of suspected CNVs.

To address the shortcomings in memory usage, we introduce LongAGE, a memory-efficient implementation of AGE. LongAGE leverages linear space alignment algorithms based on the idea first presented to solve the longest common subsequence problem [59] and several other such algorithms for sequence alignments [60]. We show that LongAGE significantly improves memory usage compared to AGE. This allows users to realign long reads or contigs on a regular compute node, desktop, or laptop.

4.2 Methods

4.2.1 Memory-efficient Implementation

Given two sequences to be aligned of length N,M: $X = x_1x_2x_3...x_N$ and $Y = y_1y_2y_3...y_M$, with $\omega = \max(M, N)$. Let $P_0 = 0$, $P_i = P_{i-1} \oplus \phi(a_\xi, b_\tau)$ denote the optimal score for the left flank (ξ_*, τ_*) and $Q_\omega = 0$, $Q_j = Q_{j+1} \oplus \phi(a_\chi, b_\psi)$ denote the optimal score for the right flank (χ_*, ψ_*), where ϕ is defined to be the maximum sum of

values (aligning x to y, or either x or y to a gap "-") of up-to the aligned pairs. The AGE algorithm is summarized as follows:

$$\begin{aligned} \max_{i,j} \quad & \{P_i + Q_j\} \\ \text{s.t.} \quad & 0 \leq i < j \leq \omega \\ & P_0 = 0 \\ & Q_\omega = 0 \end{aligned} \tag{4.1}$$

Recall that the AGE algorithm uses matrices to compute the best score (BS) of aligning n and m nucleotides at the $5'$-ends and $N - n$ and $M - m$ nucleotides at the $3'$-ends is $M^L(n, m) + M^R(n + 1, m + 1)$, where M^L is the maximum in the leading submatrix $[0, n] \times [0, m]$ and M^R is the maximum in the trailing submatrix $[n + 1, N + 1] \times [m + 1, M + 1]$:

$$BS = \max(M^L(n, m) + M^R(n + 1, m + 1)) \tag{4.2}$$

We reckon that M^L and M^R are values of P_i and Q_j respectively. To reduce memory usage, we can use a single array (α, β) for each matrix:

$$P_i = \max_{\substack{0 \leq \xi \leq n \\ 0 \leq \tau \leq m}} \{\alpha_\tau + \phi(x_\xi, y_\tau)\} \tag{4.3}$$

$$Q_j = \max_{\substack{n+1 \leq \chi \leq N+1 \\ m+1 \leq \psi \leq M+1}} \{\beta_\psi + \phi(x_\chi, y_\psi)\} \tag{4.4}$$

AGE's matrices implementation finds the alignment of two sequences by tracing back to a cell having the best local alignment. However, with our linear-space

implementation, we have to walk through the sequences once again to find the best local alignments that sum up to the best score.

$$BS = \max \left\langle \sum_{i,j=0,0}^{k-1,l-1} \phi(u_i \rightarrow u_{i+1}) + \sum_{i=0}^{k-1} \phi(u_i \rightarrow u_{i+1}) \right\rangle \tag{4.5}$$

$$\max_{1 \leq j < k \leq n \text{s.t.} P_j = 1 \wedge Q_k = 1} j + n - k + 1 \tag{4.6}$$

Our main implementation is summarized in two steps:

- Compute the maxima scores using the linear-space algorithms using the detail implementation outlined by [60].
- Reconstruct pair-wise alignments based on the maxima scores (the second round of the same procedure of finding the maxima scores).

It is well known that CNVs and SVs can have homologous and identical sequences around their breakpoints [61]. Several optimal alignments exist with the same maxima scores because of identical sequences at SV breakpoints [62], differences in alignments result from shifting along the identical sequences. By common convention LongAGE returns the left-shifted solution. LongAGE reduces the space usage from $\theta(NM)$ to $\theta(\max(N, M))$, while increasing computation time by at most four times. As mentioned in [61], there are homology sequences around breakpoints of SVs, where it happens that several optimal alignments can occur when aligning sequences containing SVs [62]. Therefore, we might have several alignments with the same maxima scores, however their actually alignments are slightly different. Additionally, Hirschbergs method reduces the space use from $\theta(NM)$ to $\theta(N)$ while possibly doubling the worst case time needed for the computation.

4.2.2 Resolving Breakpoints of mCNVs using Long-reads

Figure 4.1 shows an application using AGE alogrithm to resolve breakpoints for haplotypes with deletion and tandem duplication (one case of mCNVs) in a family

Table 4.1: Memory usage in megabytes and run time in seconds of AGE and LongAGE in controlled experiments on aligning two sequences with various variant lengths. Benchmarks were made on an Intel Xeon(R) Gold 6148 Processor (27.5M Cache, 2.40 GHz) with 192 GB of memory.

Tools	Memory usage (megabytes)						
	1Kbp	2Kbp	4Kbp	8Kbp	16Kbp	32Kbp	1Mbp
AGE	550.83	600.85	700.90	901.04	1,301.21	2,101.68	⊘
LongAGE	2.71	2.92	3.13	3.55	3.62	5.55	113.29
Tools	Running time (seconds)						
	1Kbp	2Kbp	4Kbp	8Kbp	16Kbp	32Kbp	1Mbp
AGE	5.05	5.55	6.57	8.37	12.03	19.27	⊘
LongAGE	18.92	20.72	22.80	23.77	32.06	50.63	1159.61

(parents and a child). Our method for resolving breakpoints of mCNVs using long-read data went as follows:

Identify SVs of interest (Figure 4.1A): Aligned Illumina HigSeq short-reads [63] in BAM format are available for three trios from the Genome in a Bottle (GIAB) Consortium. The coverage was $100\times$ for the parents and $300\times$ for the child. CNVs were discovered in children using CNVnator [64] with default options and 1Kbp bins. We then genotyped CNVs in corresponding parents using the same bin size. CNVnator returned estimated copy number (CN) for each member of the trio. Applying the condition: $(0.5 \leq$ CN(in one parent) $\leq 1.5)$ and $(2.5 \leq$ CN(in the other parent) $\leq 3.5)$ and $(1.5 \leq$ CN(in child) $\leq 2.5)$ for each GIAB trio, we obtained two candidate mCNVs. The candidate mCNV in the Ashkenazim trio was likely a false positive as no PacBio reads supported deletion and duplication in that region. The other mCNV in the Chinese trio was around 20Kbp in length and contained a deletion in the father (HG006) and a duplication in the mother (HG007) (Figure 4.1B).

Analyze long-reads containing SVs (Figure 4.1C): NGMLR [57] was used to map the GIAB Mt Sinai PacBio reads of the Chinese son (HG005) [63] to the Human Reference GRCh38, where the option was "$-x \quad pacbio$". Using SAMtools [65], we

extracted reads from regions of interest, which are chromosomal coordinates where coordinate intervals $[L - 40\text{Kbp}, R + 40\text{Kbp}]$, where L and R refer to the left and right breakpoint coordinates from read depth (RD) analysis. Extracted reads were realigned to the reference genome around the breakpoints using LongAGE with either "$-indel$" or "$-tdup$" which specify alignment that is expected to have indels or duplications in the read sequence respectively. However, it should be noted that until recently long-reads have had high error rates [66], hence our use of a lower gap opening penalty "$-go = -1$".

Rectify SV breakpoints (Figure 4.1C): Realigned reads were grouped based on which haplotype (deletion or duplication) had better support. For the best alignment, we required that: (i) the ranges of excised regions from LongAGE's alignment roughly match the breakpoints of mCNV; (ii) every flank of a read should be fully aligned from the beginning to the first breakpoint and from the second breakpoint to the end of the read; (iii) its score is at least 500 more than for the alignment in the alternative mode ("$-indel$" for "$-tdup$" and vice versa). We assembled the above-selected reads into two contigs using a long-read assembler wtdbg2 [67] and then aligned those contigs with the same parameters to precisely resolve the breakpoints.

4.3 Results

To study the trade-off between memory usage and running time, we created a synthetic dataset of SVs with lengths varying from 1Kbp to 32Kbp, and one of 1Mbp length. Inspired by [55, 56], we randomly generated coordinates of a synthetic deletion of a certain length, then created the sequence of SV allele by joining left and right flanks of 10Kbp in length total. We then aligned the created sequence of SV allele against the regions in the reference from the 5-end of the left flank to 3-end of the right flank. We perform alignment with AGE and LongAGE on each pair of such sequences for all lengths of synthetic SVs.

Table 4.1 summarizes run time and memory usage of AGE and LongAGE by Valgrind [68] on all pairs of synthesized sequences. In LongAGE, memory usage grows

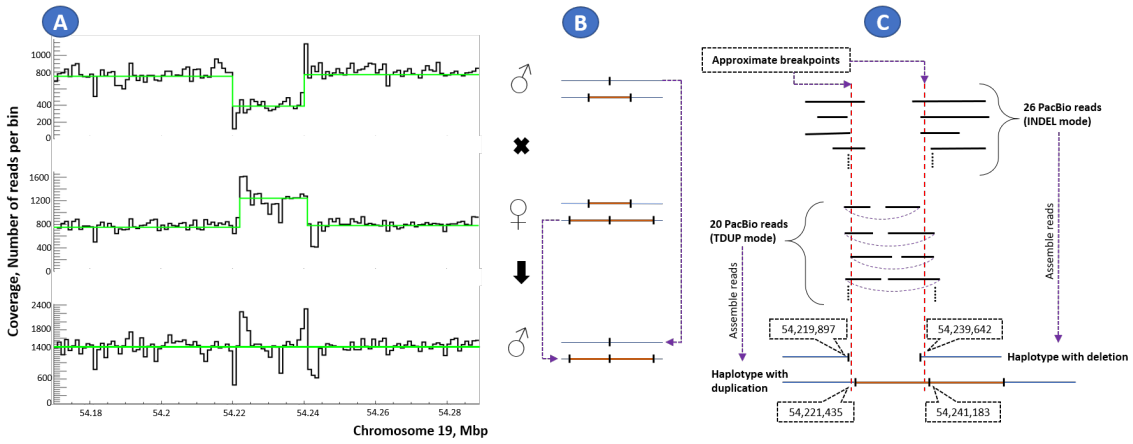

Fig. 4.1: Defining breakpoints of mCNV on chromosome 19 in Chinese Trio from GIAB. (A) Read depth signals from top to bottom corresponding to father (HG006), mother (HG007), and son (HG005), (B) Haplotypes with deletion and duplication are passed down from both parents to son, (C) Haplotypes with tandem duplication and deletion were assembled by haplotype-assigned PacBio reads. Breakpoints of the deletion and duplications are different.

linearly, while computation time is 2.6 to 3.7× longer than AGE, which is expected under Hirschbergs method. Given 192GB of memory on a Gold 6148 Processor workstation, AGE failed to align sequences of 1Mbp due to the lack of memory allocation. LongAGE completed in nearly twenty minutes and only needed a maximum of 114 megabytes for the task.

Thousands of deletion and duplication polymorphisms larger than 1Kbp in human genomes, called copy number variations (CNVs), can impact phenotypes by causing gene dosage and structure to vary among individuals [69]. Some rare and de novo CNVs have well-known roles in diseases; however, many CNVs are multi-allelic (mCNVs) where their structural alleles have been rearranged multiple times in their ancestors. The origin of such events is not fully understood due to difficulties in resolving their breakpoints with short reads, as the breakpoints are often embedded in segmental duplications. To demonstrate the applicability of LongAGE, we resolved breakpoints of reciprocal deletion and duplication with long homologies around breakpoints in the Chinese Trio sequenced by the GIAB Consortium. Such events have been previously described by [64] and were

hypothesized to occur from a single non-allelic homologous recombination (NAHR) mentioned in [55, 56].

First, we identified a copy number neutral region on the Human Genome GRCh38 of mCNV ($chr19{:}54,219,999\text{-}54,241,000$) with possible deletion and duplication haplotypes in a child using Illumina HiSeq short-read data [63] (Figure 4.1A). Then, assuming the two (deletion and duplication) haplotypes are present in the child (Figure 4.1B), we locally realigned PacBio long-reads with LongAGE using both INDEL (for alignment with deletion) and TDUP (for alignment with tandem duplication) modes. Next, by comparing alignments in each mode, we selected reads likely to be supported by deletion and tandem duplication. Breakpoints can be imprecise due to sequencing errors/homologies, yet roughly match those identified from RD analysis. We obtained 26 deletion-supporting reads, and 20 duplication-supporting reads. We then assembled these reads into two contigs, which aligned to the reference (by LongAGE in appropriate mode) with a high percent identity of over 98%. We observed that deletion breakpoints are left-shifted compared to duplication breakpoints for 1538bp and 1541bp for the left breakpoint and the right breakpoint respectively (Figure 4.1C). Such a shift suggests that the deletion and duplication occurred ancestrally from two different events.

4.4 Discussion

We have presented LongAGE, a memory-efficient implementation of AGE. Even when aligning megabase-long sequences, LongAGEs memory footprint is less than hundreds of megabytes, while it is at most four times slower than AGE in terms of running time. The tool facilitates the resolution and standardization of SV breakpoints in highly repetitive regions at a single base pair. It is capable of refining read alignment once a read has been heuristically mapped to a particular genomic location that is expected to contain an SV. In addition, with AGE acting as a generalized algorithm, the software is compatible for use in various biological studies that primarily rely on alignments and it can even be extended into several different versions for various purposes.

4.4.1 LongAGE's Features

LongAGE[1] has the same features of AGE [58], which aligns indel (insertion/deletion), tdup (tandem duplication), inv (inversion) modes. For more complex SVs, one can create a sudo SV sequence, then use the tool for aligning the sudo sequence to the reference for resolving the breakpoints. LongAGE's modes are as follows:

- INDEL mode ("-indel") assume alignment has deletion or insertion (default)
- TDUP mode ("-tdup") assume alignment has tandem duplication
- INV mode ("-inv") assume alignment has inversion; tries alignment over the left and right breakpoints; reports the best alignment

LongAGE core software was developed in C++, and source code is free for noncommercial use and available at https://github.com/Coaxecva/LongAGE.

4.4.2 Best Practice for Resolving SV Breakpoints

Methods that determine the breakpoints of genetic variants from sequencing data rely by and large on computational methods that align reads to reference genomes and methods that detect where the consensus of the starting and ending differences occur. Reads can be correctly mapped and incorrectly aligned. Misalignment at a correct chromosomal location can affect the determination of a variant's breakpoints. Moreover, many SVs/CNVs have breakpoints that are often embedded in segmental duplications, which makes the breakpoints are difficult to resolve.

Our goal was to standardize the breakpoint resolution process of SVs/CNVs such that (1) the methods used to resolve the breakpoints produce precise and consistent results, (2) the adoption of the process for resolving breakpoints is standardized such that it can be applied more easily with different tools, and (3) the process can be extended by various variant types (insertion, deletion, and inversion) and genome context.

The task for resolving SV/CNV breakpoints consists of these steps:

1. **Identify regions of interest in SVs/CNVs:**

[1] for help option, Help: $ ./long_age_align

Given aligned short-reads in BAM format, we call CNVs using CNVnator [64]. There are five steps for calling CNVs using CNVnator: extract reads with specified chromosomes/sequences, generate a read depth histogram with chosen bin size, compute statistics, partition read depth signal, and then call CNVs. Note that choosing bin size is important since bin size directly affects the resolution at which CNVs can be called. These locations where CNVs are called are regions of interest.

2. **Extract long-reads mapped around the regions of SVs/CNVs:**
   For long read data, we recommend using Minimap2 [70] or NGMLR [57] for read-to-reference alignment. Aligned long reads are then extracted using SAMtools [65] based on the regions of interest. Of the several extracted unique reads that overlap a region of interest, we only select reads that are longer than the length of expected CNVs, for further analysis. High read coverage would be preferable for both long and short read data [71, 72], since we are able to collect more reads.
3. **Analyze long-reads containing SVs/CNVs:**
   Selected long reads are realigned (by AGE/LongAGE) to the reference around the sites of interest. A site of interest is the chromosomal coordinate; and coordinate intervals are often extended to the right and left some thousands of bps, e.g. [L - 10k, R + 10k], where L and R refer to the left and right breakpoint coordinates of a called CNV. An appropriate mode is applied upon the suspected type of CNVs (deletion or duplication or inversion).
4. **Rectify SV/CNV breakpoints:**
   Reads are once again selected based on the quality of their alignments, which often have similar lengths between excised regions and SVs, have relatively long left and right flanks compared to the read length, and are fully aligned. These reads are assembled to contigs using a long-read assembler wtdbg2 [67]

or SPAdes [73]. We then align those contigs to sites of interest using AGE/LongAGE. We need this step since read-to-reference alignments often have some wrinkles that only show approximate breakpoints. By assembling reads to contigs, and precisely aligning contigs to sites of interest, we would achieve high confident breakpoints.

4.4.3 Resolved Breakpoints of CNVs on Chromosome 1 of GIAB HG005

To apply the best practice on a genome-wide study for resolving numbers of CNV breakpoints, we used the tools previously mentioned for defining breakpoints of CNVs on chromosome 1 in Chinese Trio from the Genome in a Bottle (GIAB) Consortium in the son (HG005). Short and long read data were downloaded from the GIAB Consortium FTP [63].

The human genome remains unassembled, unmapped, and poorly characterized by as much as 10% if not more [74]. These missing regions are annotated as multi-megabase heterochromatic gaps and are found primarily near centromeres and on the short arms of the acrocentric chromosomes. Therefore, from a set of around 300 CNVs called on the chromosome 1 using Illumina HiSeq short-reads we discarded a significant number of CNVs that fall in these genomic gap regions. During the process of resolving the breakpoints, we have also observed that some of these chosen CNVs consists of several SV events; such events should be further investigated at a later date. We solely reported resolved CNV breakpoints with high confidence. Table 4.6 shows both the original CNV coordinates and lengths called by CNVnator (with bin size = 1000) and the resolved ones. The majority of coordinates and lengths are very close, but some are slightly shifted from less than 50 bps to several hundred bps.

One example of resolving CNV breakpoints is the case where we have a deletion type CNV length of 6000bp (chr1:1581001-1587000). An example of selecting support reads for the deletion CNV is shown in Table 4.2 and Table 4.3. Its resolved breakpoints are also reported in Table 4.6.

In addition, we have also provided read identifiers used in the main paper in Table 4.4 and Table 4.5 that are either deletion (DEL) or duplication supporting reads. The criteria for selecting supporting reads here are that (1) the excised regions on the reference have a length similar to the expected length of the SV, (2) the left and right flanks get fully aligned, and that (3) the alignment score is relatively high.

Table 4.2: Selecting support reads at regional coordinates chr1:1581001-1587000; AS: Alignment Score, ER: Excised Regions, ILF: Identic Left Flank, IRF: Identic Right Flank

ReadID	Length	Alignment score	Excised regions	ILF (%)	IRF (%)	Selected
m54016_171228_072100/52625559/1491_29039	27548	3053	6061	9140 (85%)	1847 (87%)	yes
m54015_180106_132909/52822972/4704_28388	23684	4412	6411	9146 (88%)	3528 (86%)	yes
m54015_171028_225633/63832573/0_17986	17986	106	822	166 (82%)	255 (83%)	no
m54016_171217_185748/73794316/0_22134	22134	4874	35	952 (87%)	15168 (86%)	no
m54016_171216_223846/35586629/0_17705	17705	4040	101	3454 (87%)	8988 (87%)	no
m54016_171216_223846/38666568/12786_33162	20376	153	11425	216 (84%)	161 (89%)	no
m54016_171102_063154/61801329/0_17044	17044	3434	6131	8931 (86%)	4121 (86%)	yes
m54016_171101_202423/29819626/0_16643	16643	127	893	141 (85%)	258 (86%)	no
m54015_171111_220010/19136785/15253_27703	12450	3968	6194	9143 (88%)	962 (90%)	yes
m54016_171227_005214/9634778/0_12079	12079	115	328	104 (90%)	142 (85%)	no
m54016_171216_223846/17564577/5589_23006	17417	80	16525	49 (96%)	110 (85%)	no
m54016_171225_110102/37094119/348_12938	12590	2957	6316	8616 (87%)	1404 (79%)	yes
m54015_180106_032201/31654073/73_22781	22708	166	957	272 (86%)	182 (86%)	no
m54016_171101_202423/48694189/0_16028	16028	4004	171	2932 (85%)	11258 (87%)	no
m54016_171214_075421/68223427/789_25057	24268	109	14801	102 (88%)	71 (93%)	no
m54016_171102_063154/64356785/1350_12981	11631	70	15386	59 (91%)	70 (92%)	no
m54015_171029_090401/48759165/0_6589	6589	1880	150	104 (89%)	5568 (87%)	no
m54016_171124_053133/56885340/8141_26611	18470	3571	87	8212 (86%)	6083 (84%)	no
m54015_171112_081009/18154101/8688_16033	7345	186	3264	253 (90%)	91 (89%)	no
m54015_171029_090401/65340146/0_14161	14161	102	1832	165 (85%)	66 (87%)	no
m54015_180106_032201/24904525/47305_55131	7826	1906	285	3349 (89%)	2450 (87%)	no
m54015_180106_032201/24904525/36751_47260	10509	97	794	105 (88%)	52 (100%)	no
m54015_180106_032201/39649469/71_10958	10887	82	8777	110 (87%)	49 (94%)	no
m54016_171122_124350/11337971/0_7941	7941	75	1937	113 (85%)	141 (84%)	no
m54016_171101_101302/36110957/0_6928	6928	1672	89	1053 (86%)	5065 (86%)	no
m54015_171112_081009/14811298/6198_18038	11840	2744	5	7309 (87%)	1483 (86%)	no

Table 4.3: Selecting support reads at regional coordinates chr1:1581001-1587000; AS: Alignment Score, ER: Excised Regions, ILF: Identic Left Flank, IRF: Identic Right Flank (continue)

ReadID	Length	AS	ER	ILF (%)	IRF (%)	Selected
m54016_171216_022308/36111269/15629_32679	17050	175	850	353 (85%)	204 (87%)	no
m54015_180106_132909/14746013/29780_35471	5691	54	19807	27 (100%)	50 (91%)	no
m54016_171227_005214/53215669/19115_35718	16603	103	20922	135 (87%)	50 (94%)	no
m54016_171214_075421/70910447/27533_34739	7206	2767	50	2522 (89%)	3937 (90%)	no
m54015_171222_210605/15401756/10269_22769	12500	156	908	317 (85%)	222 (85%)	no
m54016_171214_075421/70910447/0_27486	27486	209	926	347 (86%)	231 (88%)	no
m54016_171214_075421/38601194/0_21263	21263	114	9573	61 (92%)	196 (86%)	no
m54016_171102_063154/8454644/14430_14723	293	46	102	70 (88%)	16 (100%)	no
m54016_171101_101302/36110957/6973_9352	2379	81	4839	63 (93%)	41 (98%)	no
m54016_171102_063154/74449519/34019_43745	9726	118	939	167 (89%)	180 (81%)	no
m54015_171227_113709/15205161/41674_43761	2087	943	13	1647 (91%)	252 (91%)	no
m54015_171227_113709/15205161/12065_41628	29563	190	852	200 (91%)	234 (88%)	no
m54015_171111_220010/66519305/0_714	714	31	17709	19 (100%)	12 (100%)	no
m54016_171102_063154/31327019/0_5742	5742	74	2229	62 (93%)	45 (98%)	no
m54015_171028_024134/56558247/0_9962	9962	232	2187	417 (86%)	230 (86%)	no
m54016_171216_022308/73335329/1623_12658	11035	1681	2009	761 (85%)	5282 (86%)	yes
m54016_171102_063154/28049559/8763_17294	8531	48	4835	40 (93%)	28 (93%)	no
m54016_171123_091238/7012674/0_6209	6209	180	914	362 (86%)	241 (84%)	no
m54016_171101_101302/67109356/0_14848	14848	2409	2023	1023 (89%)	5257 (88%)	yes
m54016_171123_091238/40108875/0_17230	17230	2445	1022	6333 (88%)	1378 (85%)	no
m54015_180106_032201/33424178/0_22991	22991	5283	181	7256 (87%)	11357 (86%)	no
m54016_171225_005142/51315226/0_6422	6422	2213	80	2171 (87%)	3499 (88%)	no
m54016_171122_124350/30343569/0_14067	14067	3011	162	2771 (86%)	6402 (88%)	no
m54016_171217_185748/71106772/1007_6739	5732	78	5710	108 (89%)	69 (88%)	no
m54016_171216_022308/70845026/6186_19468	13282	133	795	236 (85%)	270 (86%)	no
m54015_171222_004930/31589129/0_4431	4431	1238	204	1746 (87%)	1986 (87%)	no
m54016_171216_223846/52822936/14670_27368	12698	1262	610	1444 (85%)	4361 (85%)	no

Table 4.4: Deletion support reads of mCNV ($chr19{:}54,219,999\text{-}54,241,000$) on chr.19; L: length of the read, AS: Alignment Score, ER: Excised Regions

ReadID	L	AS	ER
m54015_171028_124452/11927686/0_5894	5894	4099	19787
m54015_171028_124452/40502023/4621_21625	17004	13630	19746
m54015_171111_220010/45023851/20779_21717	938	641	19752
m54015_171113_042859/68289266/690_10369	9679	7315	19777
m54015_171222_004930/18416590/1488_30909	29421	23408	19745
m54015_171222_004930/29622358/0_27636	27636	21397	19747
m54015_171222_004930/29622358/27681_40498	12817	10278	19746
m54015_171223_071608/30409028/0_20443	20443	16152	19743
m54015_171229_224813/74514908/759_3476	2717	1908	19745
m54015_180106_032201/47055322/14317_34915	20598	16389	19746
m54015_180106_032201/47055322/34959_51683	16724	13684	19743
m54015_180106_132909/10289779/0_6162	6162	4605	19747
m54015_180106_132909/63176887/0_998	998	761	19844
m54016_171101_000840/38994690/6252_12105	5853	4432	19745
m54016_171102_063154/61145748/32600_38234	5634	3895	19746
m54016_171213_113525/29557030/0_6411	6411	4733	19745
m54016_171216_022308/21954906/0_17701	17701	11750	19750
m54016_171216_122926/53608588/33165_55439	22274	14808	19745
m54016_171217_084820/39649562/6571_26530	19959	14450	19738
m54016_171217_084820/39649562/26575_38922	12347	8846	19746
m54016_171218_050715/58917702/19046_41634	22588	16848	19744
m54016_171220_193020/25428496/0_27254	27254	20042	19743
m54016_171220_193020/25428496/27299_31053	3754	3026	19745
m54016_171225_110102/60555644/0_31094	31094	23467	19743
m54016_171227_005214/18875151/23157_29243	6086	4694	19748
m54016_171227_005214/37552377/24484_33548	9064	5987	19745

Table 4.5: Duplication support reads of mCNV ($chr19{:}54,219,999\text{-}54,241,000$) on chr.19; L: length of the read, AS: Alignment Score, ER: Excised Regions

ReadID	L	AS	ER
m54015_171029_191526/18743862/0_19312	19312	14384	19756
m54015_171029_191526/51184327/20128_30697	10569	3486	19781
m54015_171111_220010/54198833/11478_15750	4272	2553	19761
m54015_171113_042859/5177604/0_19297	19297	8546	19788
m54015_171113_042859/73531488/0_15177	15177	10399	19749
m54015_171229_224813/9371837/0_14340	14340	8706	19742
m54015_180106_032201/8717133/0_13537	13537	10707	19752
m54015_180106_132909/55772079/358_15568	15210	10076	19758
m54015_180106_233853/29164362/0_25277	25277	17556	19755
m54016_171101_000840/66192044/0_12511	12511	9778	19752
m54016_171101_101302/27984790/8800_24418	15618	11350	19754
m54016_171124_053133/16843161/0_15010	15010	10734	19751
m54016_171213_113525/13435607/19595_43374	23779	17932	19745
m54016_171216_022308/31195929/270_12126	11856	3189	19120
m54016_171216_223846/24511180/12804_39691	26887	20343	19754
m54016_171217_084820/12649301/0_12579	12579	9346	19737
m54016_171218_050715/53674260/2515_16122	13607	8035	19775
m54016_171219_231439/17957098/0_31591	31591	20838	19741
m54016_171227_005214/18088266/18129_21100	2971	2430	19759
m54016_171228_072100/19333688/2340_30269	27929	19873	19747

Table 4.6: Resolved breakpoints on chromosome 1 of GIAB HG005 (son); SV type: type of CNVs (deletion or duplication); Est. Coordinates: estimate coordinates called by CNVnator with bin size = 1000; Est. L.: estimate length of the SV produced by CNVnator; Res. Coordinates: resolved coordinates using best practice; Res. L.: resolved length of the SV using best practice

SV type	Est. Coordinates	Est. L.	Res. Coordinates	Res. L.
deletion	chr1:1028001-1031000	3000	chr1:1027773-1031373	3599
deletion	chr1:1581001-1587000	6000	chr1:1581027-1587055	6027
deletion	chr1:9787001-9797000	10000	chr1:9786592-9796634	10041
deletion	chr1:14110001-14113000	3000	chr1:14109813-14112445	2631
deletion	chr1:24194001-24197000	3000	chr1:24193857-24197185	3327
deletion	chr1:24832001-24835000	3000	chr1:24832199-24835023	2823
deletion	chr1:25767001-25769000	2000	chr1:25767787-25769385	1597
deletion	chr1:27852001-27854000	2000	chr1:27852202-27854291	2088
deletion	chr1:47158001-47160000	2000	chr1:47157819-47159974	2154
deletion	chr1:54627001-54630000	3000	chr1:54626598-54630293	3694
deletion	chr1:75377001-75383000	6000	chr1:75377846-75383253	5406
deletion	chr1:82660001-82662000	2000	chr1:82660275-82661887	1611
deletion	chr1:84052001-84059000	7000	chr1:84052242-84058947	6704
duplication	chr1:86612001-86632000	20000	chr1:86611770-86631274	19503
deletion	chr1:91459001-91461000	2000	chr1:91459306-91460671	1364
deletion	chr1:93823001-93826000	3000	chr1:93822816-93825701	2884
deletion	chr1:109644001-109649000	5000	chr1:109644537-109648774	4236
deletion	chr1:112149001-112162000	13000	chr1:112149168-112162082	12913
deletion	chr1:152583001-152617000	34000	chr1:152583066-152615265	32198
deletion	chr1:156557001-156559000	2000	chr1:156556912-156559144	2231
deletion	chr1:174827001-174833000	6000	chr1:174827418-174832707	5288
deletion	chr1:175102001-175108000	6000	chr1:175101345-175107500	6154
deletion	chr1:184846001-184852000	6000	chr1:184845609-184851667	6057
deletion	chr1:194482001-194485000	3000	chr1:194481201-194485183	3981
deletion	chr1:197531001-197534000	3000	chr1:197531378-197534142	2764
deletion	chr1:198335001-198341000	6000	chr1:198335523-198341149	5625
deletion	chr1:218009001-218015000	6000	chr1:218009146-218015253	6106
deletion	chr1:236385001-236388000	3000	chr1:236385786-236388030	2243
deletion	chr1:242964001-242966000	2000	chr1:242964469-242965914	1444
deletion	chr1:245974001-245977000	3000	chr1:245974207-245976531	2323
deletion	chr1:246520001-246522000	2000	chr1:246519928-246521874	1945
deletion	chr1:247687001-247693000	6000	chr1:247687149-247693207	6057
deletion	chr1:247888001-247894000	6000	chr1:247888190-247894398	6207

Chapter 5

Algorithms for Detecting Unknown Microbes

Quantification and identification of microbial genomes based on next-generation sequencing data is a challenging problem in metagenomics. Although current methods have mostly focused on analyzing bacteria whose genomes have been sequenced, such analyses are, however, complicated by the presence of unknown bacteria or bacteria whose genomes have not been sequence. We propose a method for detecting unknown bacteria in environmental samples. Our approach is unique in its utilization of short reads only from 16S rRNA genes, not from entire genomes. We show that short reads from 16S rRNA genes retain sufficient information for detecting unknown bacteria in oral microbial communities. In our experimentation with bacterial genomes from the Human Oral Microbiome Database, we found that this method made accurate and robust predictions at different read coverages and percentages of unknown bacteria. Advantages of this approach include not only a reduction in experimental and computational costs but also a potentially high accuracy across environmental samples due to the strong conservation of the 16S rRNA gene.

5.1 Motivation and Related Work

Although the main objective of metagenomics analysis focuses on profiling known bacteria, it is complicated by the presence of unknown bacteria (or those without sequenced genomes). To the best of our knowledge, only MicrobeGPS [75] provides a basic analysis of unknown bacteria in how they are similar to known bacteria. It does not address the scenario where unknown bacterial genomes are vastly different from already-sequenced reference genomes.

To address this challenge, this work focuses on identifying and quantifying unknown bacteria in microbial communities. In this context, *unknown bacteria* are those whose genomes have not been sequenced. Given short reads from a microbial community that contain genomic materials from known and unknown bacteria, the method works by

(i) first separating reads from known bacteria and unknown bacteria, and then (ii) clustering reads from unknown bacteria into multiple clusters; each cluster represents a hypothetical unknown bacterium. Importantly, the method utilizes only reads from 16S rRNA genes as a means to accomplish these tasks. Due to its high conservation, historically, the 16S rRNA gene has been used as a marker for taxonomic and phylogenetic analyses ([76, 77]). In the context of metagenomics, whose analyses depend on only short reads and not entire genes, the 16S rRNA gene was recently used as a means to construct functional profiles of microbial communities [78].

Using the 16S rRNA gene instead of whole genome information is not only computational efficient but also economical; Illumina indicated that targeted sequencing of a focused region of interest reduces sequencing costs and enables deep sequencing, compared to whole-genome sequencing. On the other hand, as observed by [75], by focusing exclusively on one gene, one might lose essential information for advanced analyses. We, however, will provide an analysis that demonstrates that at least in the context of oral microbial communities, the 16S rRNA gene retains sufficient information to allow us detect unknown bacteria.

5.2 Uniqueness of the 16S rRNA Genes

Using the 16S rRNA gene as marker instead of the whole genome for identification and profiling bacterial communities potentially can lose a lot of information. On the other hand, this gene is highly conserved, which means that using it as the marker is more advantageous than using the whole genome since the reference gene in our database is less likely to be different than the gene in bacteria collected from environmental samples. Our analysis with a dataset that consists of 889 bacteria in the Human Oral Microbiome database suggests that the use of the 16S rRNA gene as marker is justified because there is a sufficient amount of information in this gene among different bacteria to help distinguish these bacteria. Consequently, the use of the 16S rRNA gene as

marker to distinguish bacteria enjoys both the advantageous characteristics of the gene and having sufficient information required for the task.

To analyze the effectiveness of using the 16S rRNA gene as marker, we quantify the uniqueness of the gene among the set of 16S rRNA genes in bacteria of interest. To be precise, let $G = \{g_1, g_2, \cdots, g_n\}$ be the set of 16S rRNA genes of bacteria of interest. Define $U(k, g_i, g_j)$ to be the number of k-mers in g_i that are not in g_j or g_j^{rc} divided by $|g_i| - k + 1$, where g_j^{rc} is the reverse complement of g_j. Note that $0 \leq U(k, g_i, g_j) \leq 1$. In particular, $U(k, g_i, g_j)$ being 1 means that all k-mers in g_i do not occur in g_j or g_j^{rc}. Thus, when $U(k, g_i, g_j) = 1$, it is likely that reads much longer than k coming from g_i will not be mistakenly mapped to g_j. Further, for each g_i, define

$$U(k, g_i) = \min_{1 \leq j \leq n, j \neq i} U(k, g_i, g_j)$$

Thus, the *uniqueness score*, $U(k, g_i)$, is a conservative measure of uniqueness of g_i in the whole set G. The closer $U(k, g_i)$ is to 1, the more unique it is, and the more likely that reads much longer than k from g_i will not be mistakenly mapped to any other gene g_j in G.

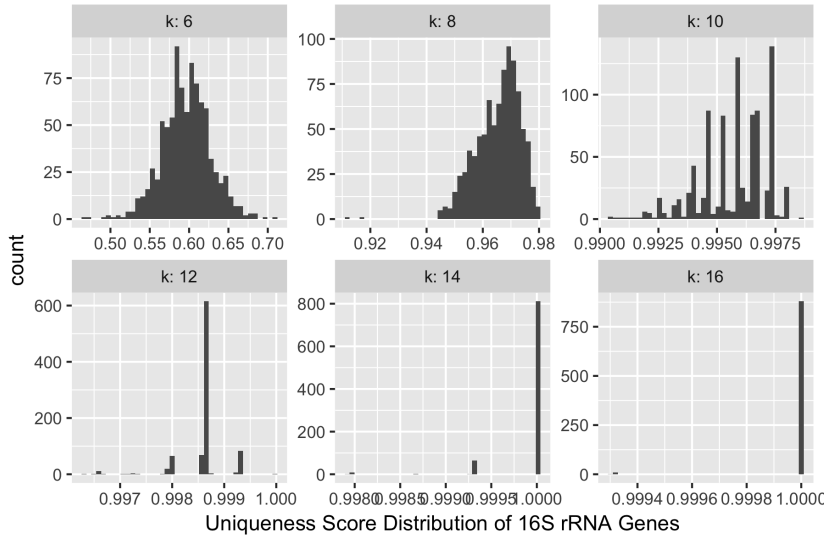

Fig. 5.1: Distributions of $U(k, g_i)$ of 16S rRNA genes suggest that k-mers longer than 16 can effectively be used to distinguish bacteria in the human oral microbiome.

Figure 5.1 shows, for different values of k, the distributions of $U(k, g_i)$ of 889 16S rRNA genes obtained from the Human Oral Microbiome database. We can see that the distribution of $U(6, k_i)$ peaks at around 0.58; i.e. around 88 genes have uniqueness scores at approximately 0.58. When $k = 8$, most genes have uniqueness scores at around 0.97. When $k = 16$, most genes have uniqueness scores at 1. When $k \geq 18$, we observed that all genes have uniqueness score of 1. This means for each gene in G, we can distinguish it with other genes using 18-mers. It also means that given reads produced by current technologies (e.g. ≥ 10), it is likely that reads that come from some gene g_i will not be mistakenly mapped to any gene other than g_i.

5.3 Method

5.3.1 Overview

Our method for identifying unknown bacteria from short reads that come from 16S rRNA genes of all bacteria (including known and unknown bacteria) in an environmental sample works as follows:

1. Reads are first roughly assigned to known bacteria. This is done by aligning those reads to the collection of already-sequenced 16S rRNA genes of known bacteria. The alignment process can be done using a good aligners such as Bowtie2 [79], BWA-MEM[80], Soap2 [81], RandAL[82]. We used Bowtie2 due to the efficiency and flexibility of the software package. The aligner works by creating an index $\mathcal{R}$ of reference 16S rRNA genes, which come from known (already-sequenced) bacterial genomes.
2. Reads that are not mapped to $\mathcal{R}$ are presumed to belong to 16S rRNA genes of unknown bacteria. We used SAMtools [65, 77] to collect unmapped reads from the results of Bowtie2. At this point, it is possible and actually expected that (i) some reads that belong to unknown bacteria have been mistakenly mapped to $\mathcal{R}$, and (ii) some reads that belong to the 16S rRNA gene of some known

bacteria are mistakenly not mapped to $\mathcal{R}$. Thus, the set of unmapped reads, $\mathcal{U}$, contain both false positives and false negatives.

3. The unmapped reads, $\mathcal{U}$, are then clustered into distinct clusters. Each cluster represents a hypothetical unknown bacterium. An additional post-processing step can be applied to (i) remove clusters with too few reads as they do not possess sufficient information and (ii) split large clusters that might contain reads belong to more than one bacteria. At this point, it is possible that (i) multiple clusters can represent the same unknown bacterium and (ii) an unknown bacterium is not represented at all by any cluster. Both cases are not desirable and they both affect the accuracy of predicting the number of unknown bacteria.

5.3.2 Clustering Unmapped Reads

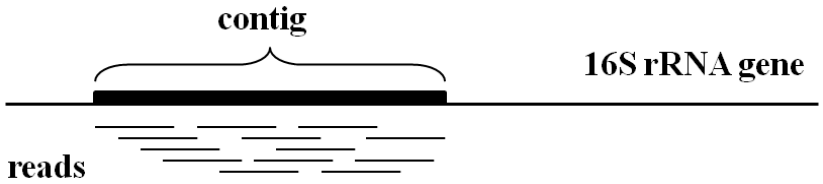

Fig. 5.2: Reads mapped to a contiguous region of a 16S rRNA gene

The clustering procedure described in Step 3 of Section Overview is a critical component of this method. Technically, each cluster is a collection of reads that cover a contiguous genomic region. In other words, if one was to align these reads to the correct genomic region of a 16S rRNA that contains these reads, they would form a contiguous sequence. See Figure 5.2.

We employ the data structure that is similar to a Union-Find data structure[83] to partition unmapped reads in $\mathcal{U}$ into a disjoint set of subsets. Each subset or cluster would represent a contiguous genomic region. This data structure C has following methods:

- MakeSet(x), which creates a singleton set containing the element x.
- Union(x,y), which unions the two disjoint sets that contain x and y.

- Find(x), which finds the set that contains x.
- Clusters(), which returns all disjoint subsets that C maintains.

Algorithm 2: Placing reads into disjoint clusters of overlapping reads

```
1: C ← UnionFind()
2: for each x in U do
3:     C.MakeSet(x)
4: for each x in U do
5:     for each y in U do
6:         if C.Find(x) ≠ C.Find(y) and Overlap(x, y) then
7:             C.Union(x, y)
8: return C.Clusters()
```

These methods can be encapsulated in data structure that is similar to the Union-Find data structure. Given the set of unmapped reads, $\mathcal{U}$, the clustering procedure (as described in Step 3, Section Overview) can be described in Algorithm 2, which is described in an inefficient manner to help understandability; our actual implementation is more efficient. Essentially, the procedure looks at all pairs of unmapped reads and – if they overlap – merges the contigs to which they belong. Since reads can be in either the primary or the complementary strand, the determination of overlapping of two reads must account for this fact. First, given two sequences, define $O(a, b) = HAM(pre(a, k), suf(b, k))$, where $pre(a, k)$ is the k-prefix of a; $suf(b, k)$ is the k-suffix of b; and HAM is the Hamming distance function. Then, the overlapping of two reads x and y is determined as follows: Overlap(x, y) is True and only if

$$\frac{\max(O(x,y), O(x^{rc},y), O(x,y^{rc}), O(x^{rc},y^{rc}))}{\min(|x|,|y|)} \geq \tau$$

where $|x|$ is the length of x; x^{rc} is the reverse complement of x; and τ is an empirically determined parameter.

5.3.3 Post Clustering Processing

Clusters produced by Algorithm 2 are predicted raw representations of different bacteria. Additional processing can be done to improve prediction accuracy. In particular,

two heuristics can be employed. First, clusters containing too few reads should be removed as they do not possess enough information to give sufficient confidence in prediction. Second, clusters with too many reads might contain reads that belong to more than one bacteria. We consider heuristics that decompose graphs into large disjoint clusters representing different bacteria. One of such heuristics is based on a well-studied problem in network analysis: decomposition of graphs into dense subgraphs [84]. To adopt this strategy, we represent the set of unmapped reads in cluster i as a graph, G_i, in which vertices represent reads and edges represent overlapping of read pairs. Specifically, there is an edge (u, v), if and only if Overlap(u, v) is true. As defined in Section Clustering Unmapped Reads , the function Overlap examines the overlapping of reads as well their reverse complements. With this representation, reads within each cluster that belong to different bacteria tend to form dense subgraphs of G_i. These subgraphs are connected with each other by edges that represent the overlapping of similar reads belonging to different bacteria.

5.3.4 Method Evaluation

As clusters returned by Algorithm 2 represent predicted species, the quality of prediction can be quantified in terms of how closely the clusters resemble the set of bacteria that reads belong to. Let $T = \{T_1, \cdots, T_n\}$ be the set of bacteria that unmapped reads belong to and $C = \{C_1, \cdots, C_m\}$ be the set of clusters that our method assigns the reads to. Although there are many different ways the accuracy of clusterings can be evaluated, we chose four different metrics that evaluate clustering quality in different meaningful and complementary ways.

Mutual information is an information-theoretic measure of how similar two joint distributions are. In the context of clustering, the mutual information between two clusterings T and C is defined as

$$MI(T, C) = \sum_{i=1}^{n} \sum_{j=1}^{m} P(i, j) \log \frac{P(i, j)}{P(i)P(j)}$$

where $P(i,j)$ is the probability that a read belongs to both T_i and C_j; $P(i)$ is the probability that a read belongs to T_i; $P(j)$ is the probability that a read belongs to C_j. The Adjusted Mutual Information (**AMI**) [85] of two clusterings is an adjustment of mutual information to account for chance and is defined as follows:

$$AMI(T,C) = \frac{MI(T,C) - E(MI(T,C))}{\max(H(T),H(C)) - E(MI(T,C))}$$

where $E(MI(T,C))$ is the expected mutual information of two random clusterings and $H(T)$ is the entropy of the clustering T. An AMI value of 0 occurs when the two clusterings are random, whereas a value of 1 occurs when C and T are identical.

Rand Index is a common measure in classification problems, where the measure takes into account directly the number of correctly and incorrectly classified items.

$$RI(T,C) = \frac{2(a+b)}{n(n-1)}$$

where a is the number of pairs of reads that are in the same cluster in T and C; and b is the number of pairs of reads that are in different clusters in T and C. The Adjusted Rand Index (**ARI**) was introduced to take into account when the Rand Index of two random clusterings is not a constant value [86]. An ARI value of 0 occurs when two C and T are independent, whereas a value of 1 means C and T are identical.

In addition to AMI and ARI, we also considered two complementary metrics, introduced by [87]: **homogeneity** and **completeness**. A clustering is homogenous if each cluster C_j contains only reads that come from some bacterium T_i. A clustering is complete if all reads that belong to any bacterium T_i are placed into some cluster C_j. These two metrics are opposing in that it is often hard to achieve high scores on both homogeneity and completeness. A few examples might help understand this intuition:

- $T = C$ if and only if both homogeneity are completeness scores are 1. T being

identical to C only occurs when reads in each T_i are placed in exactly one C_j, and all reads in each C_j come only from one T_i.

- Suppose $T = \{\{r_1, r_2\}, \{r_3, r_4\}\}$ and $C = \{\{r_1, r_2, r_3, r_4\}\}$. Then, the completeness score is 1, because all reads that belong to T_1 (and respectively to T_2) are placed in the same cluster in C. On the other hand, the homogeneity score is 0, because reads in the only cluster in C come from different bacteria in T.
- Suppose $T = \{\{r_1, r_2\}, \{r_3, r_4\}\}$ and $C = \{\{r_1, r_3\}, \{r_2, r_4\}\}$. Then, both completeness and homogeneity scores are 0.

5.4 Experiments

In this section, we report experimental results that show various aspects of accuracy and robustness of this method. Accuracy is measured by four different metrics Adjusted Mutual Information (AMI), Adjusted Rand Index (ARI), Homogeneity and Completeness.

5.4.1 Mock Oral Microbial Communities

Experiments were conducted on 16S rRNA genes obtained from 889 sequences cataloged by the Human Oral Microbiome Database. The lengths of genes vary between 1,323 to 1,656 bases. We simulated mock microbial communities at various settings in order to be able to compare ground truths and predicted values and ascertain the accuracy of the method. Each mock community consists of (A) *known* bacteria, whose 16S rRNA genes were used to filter out known bacteria, and (B) *unknown* bacteria, whose 16S rRNA genes must be identified and separated into different clusters representing different unknown bacteria.

These mock communities were synthetically created to evaluate various aspects of our method. In our experiments, short reads from 16S rRNA genes were generated using Grinder [88] using parameters for the Illumina sequencing platform. Mean read length was 150 with a standard deviation of 20. Read coverage was between 10x to 100x and the

percentage of unknown bacteria varied from 1% to 16%. To study how one parameter affects the accuracy of the method, we used mock communities in which only that parameter varied while the others were kept constant.

5.4.2 The Affect of Coverage on Prediction Accuracy

First, we examined how the method's accuracy (in terms of completeness, homogeneity, mutual information and Rand index) varied at increasing read coverages. We expected that having more reads means having more information and that would result in an observed increase in accuracy. In this experiment, read coverage in mock communities varied from 10x to 100x. The percentage of unknown bacteria in these communities were kept constant at 8%.

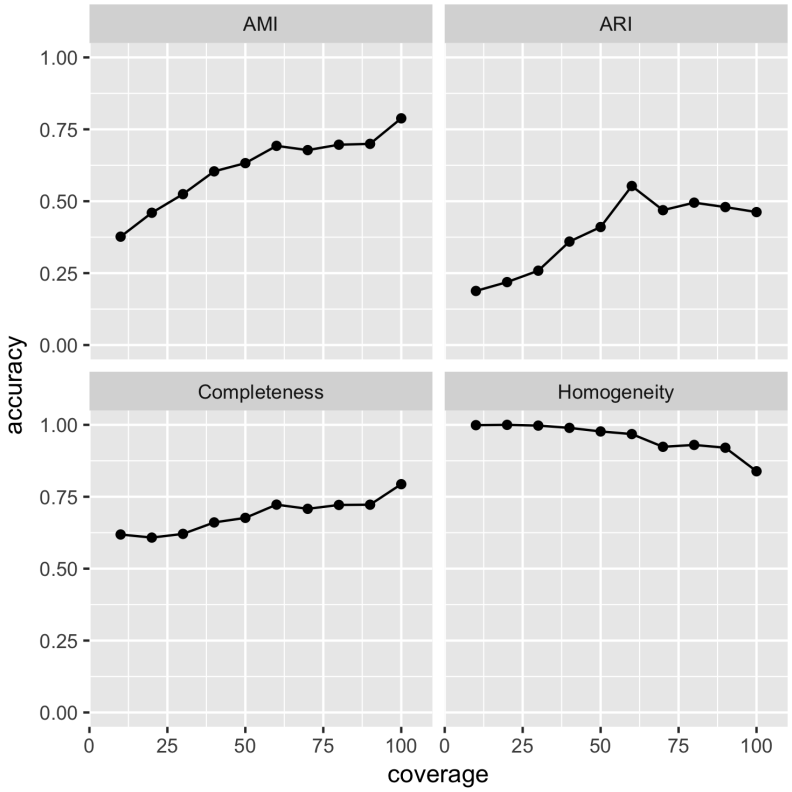

Fig. 5.3: Accuracy of predicting unknown bacteria (measured by four different metrics) at read coverage ranging from 10x to 100x.

Figure 5.3 shows accuracies measured by four different metrics. As expected, prediction accuracy was higher at higher coverage for three of the measures. Additionally,

accuracy values measured by AMI are generally higher than ARI. AMI tells us about the degree of randomness of a predicted clustering compared to the ground-truth clustering, whereas ARI attempts to quantify the item pairs that are in the same and different subsets. Our interpretation of this observation is that while predictions are not random, there are still structural information among clusters or within clusters that our method has not fully exploited.

Further, predictions were homogeneous than complete. This means that (i) a cluster more likely contains only reads that belong to some bacterium, and (ii) reads belonging to a bacterium could be placed in multiple clusters. Observation (i) confirmed that the method worked as it should. To understand observation (ii), note that if reads belonging to a gene do not assemble into a contiguous sequence (due to low or non-uniformity of coverage), then reads belonging to the gene will be placed into multiple clusters.

Finally, as coverage approached 100x, clusters became less homogeneous. This happened because having more reads increased the change of mistakenly placing reads into clusters representing different bacteria. In this experiment, 80x appears to be a good coverage.

5.4.3 The Affect of Unknown Bacteria Concentration

To study the affect of the amount of unknown bacteria has on prediction accuracy, we evaluated our method with mock communities in which percentage of unknown bacteria varied from 2% to 16%, while read coverage was kept constant at 40x with 10 random replicates at each percentage.

The result of this experiment is summarized in the box plot in Figure 5.4. As expected, prediction accuracy (as measured by AMI, ARI and Completeness) tended to decrease with more unknown bacteria. On the other hand, homogeneity were not effected very much. The result shows that accuracy starts dropping dramatically when the

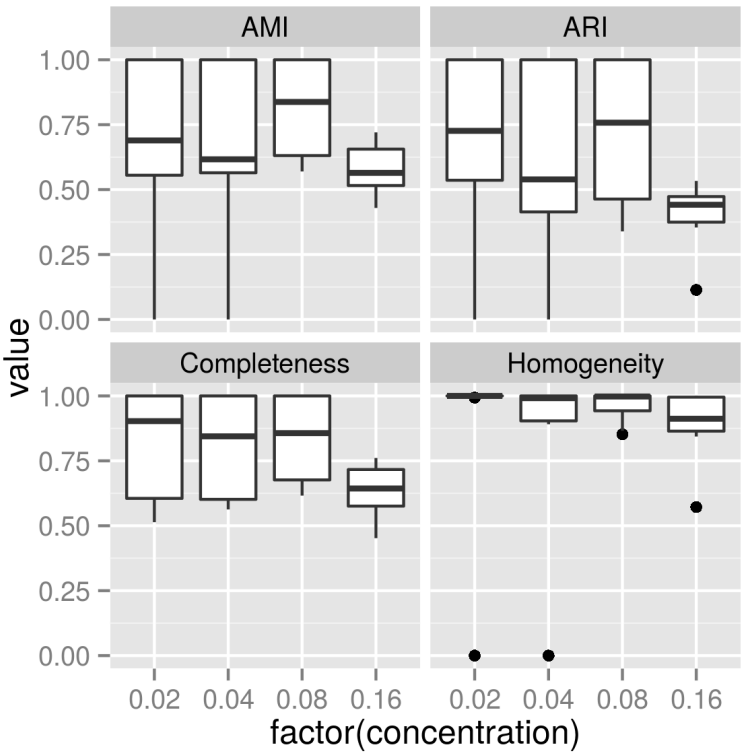

Fig. 5.4: Accuracy of predicting unknown bacteria (measured by four different metrics) at different amount of unknown bacteria.

concentration of unknown bacteria reaches 16%. We hope that future improvements can increase this number.

5.5 Discussion

Although it is known that 16S rRNA genes can be used to distinguished known bacteria, we demonstrated that only *reads* from these genes can be used to predict the number of unknown bacteria in oral microbial communities. Advantages include (i) a reduction in cost and computational processing, and (ii) the high conservation of 16S rRNA genes increases the chance of reference genetic materials being highly similar to those of bacteria in environments, which eliminates multiple sources of errors and challenges.

Chapter 6

Read Assembly Algorithms for Improving Species Classification

6.1 Background

Species classification and profiling is an important problem in metagenomics analysis. Many different computational tasks and workflows depend on the identification and profiling of organisms that are present in metagenomics samples. Examples include studies that assessed the host-microbe interactions in the gut microbiome to gain better insight into human health [89], revealed ecological differentiation of closely related bacteria [90], uncovered the presence of ancient sub-populations of marine bacteria [91], and highlighted extensive intra-species recombination [92, 93].

Methods for classification and profiling of microbial communities are diverse. CLARK [94] uses a database of k-mers that aims to uniquely describes genomic regions of each targeted microbes. GOTTCHA [95] has a different approach to identifying unique genomic regions of targeted microbes by using a combination of empirical data and machine learning methods. Kraken [96] also utilizes k-mers, but builds taxonomic trees that help differentiate closely related microbes. MetaPhlAn2 [97] employs a similar taxonomic approach, but narrows read alignment and its analysis only on a set of around one million markers.

Although short reads have low sequencing errors, their short lengths have been a limitation in the identification of structural variants, sequencing repetitive regions, phasing of alleles and distinguishing highly homologous genomic regions. This limitation can have a serious practical consequence. For example, it might have significantly contributed to the diagnostic gap in patients with genetic disorders [98].

More recent technologies can produce very long reads, but at the expense of having higher costs and much higher error rates [66]. However, longer reads have been found to be more appropriate or better compared to short reads in certain studies. Single-molecule sequencing (SMS) offers exceptionally long reads that enable direct

sequencing of genomic regions that are difficult to sequence with short reads, including long repetitive elements, extreme GC-content regions, and complex gene loci. Similarly, these platforms enable structural variation characterization at previously unparalleled resolution and direct detection of epigenetic marks in native DNA [99]. Similarly, the PacBio sequencing system can capture full-length 16S rRNA sequences [100]. Third-generation nanopore sequencing offers many solutions to the current problems of using whole metagenome sequencing (WMS) for infectious disease diagnostics. It has been successfully utilized for pathogen detection, AMR prediction, and characterization of mixed microbial communities [101].

While long read technologies are more appropriate for certain studies, short read technologies are mature and less expensive. Is it possible to leverage known strengths of short read technologies to garner the high performance of long reads?

In this chapter, we demonstrate that it is possible to improve the performance of species classification in metagenomic applications using long reads that are assembled from short reads. This finding has two major implications. First, it suggests that many existing studies that utilize short reads can benefit from long reads that are assembled from the short reads. Although there is an extra computational cost of assembly and minor modification to the existing workflows, the increase in performance might justify the cost. Second, this finding suggests that there are potential gains in utilizing long-reads technologies in this type of applications. As current long-read technologies have different characteristics from short-read technologies in terms of cost and sequencing errors, the trade-offs between these pros and cons remain to be investigated.

6.2 Results

6.2.1 Experimental Design

The main hypothesis is that the classification or identification of microbes in metagenomics samples is better done with long reads than with short reads. We aim to design a controlled experiment to verify the hypothesis. To achieve this, we evaluated the

ability to detect species in metagenomics samples of several well-known classifiers on several short-reads datasets and *derived* long-reads datasets. The choice of which long-reads datasets are used to compare against which short-reads datasets is an important design decision. If we choose a long-reads dataset produced by a current technology to compare against a short-reads dataset produced by a different technology, the result might be due to differences in technologies rather than in read lengths. As our goal is to examine the impact of read lengths on classification, we chose to use long reads that are derived from the same short reads. These derived long-reads datasets are constructed by assembling short reads from the datasets that are used to evaluate the classifiers' performance. Although this design choice removes the effect of sequencing technologies, it introduces the potential effect of assembling reads on the result. To address this, we evaluated classifiers with different assemblers to remove algorithmic bias on classification performance.

More descriptions of the experimental design can be found in section Methods.

6.2.2 Performance Assessment

Classifiers were evaluated with synthetic and real samples. Although some tools can work on the strain level, we evaluate classification results at species levels since most methods still do not provide strain level identifications. Classification performance of synthetic data was measured in terms of precision, recall, and F1.

$$Precision = \frac{TP}{TP + FP}; Recall = \frac{TP}{TP + FN}; F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Where TP (true positives): the number of correctly classified species, FP (false positives): the number of incorrectly classified species, FN (false negatives): the number of incorrectly classified non-species, by each method.

To access classifiers' performance on real data, we define the overall *pairwise*

similarity of a method c to other methods as

$$\frac{\sum_{i=1,c\neq i}^{n} |S_c \cap S_i|}{\sum_{i=i}^{n} |S_c \cap S_i|}$$

where, S_i is the number of species predicted by method i. This similarity is between 0 and 1. The closer it is to 1, the higher the overall similarity to other methods.

6.2.3 Data

Mende datasets [102] were simulated for Sanger sequencing, pyrosequencing, and Illumina sequencing. For each technology, three metagenomes were simulated to mimic different community complexities 10 species (10s), 100 species (100s), and 400 species (400s). However, the Sanger sequencing, pyrosequencing technolgies seem obsolete/out-of-date. We test our hypothesis on Illumina paired-end raw reads of Mende datasets, which is a very widely used sequencing platform.

To test our hypothesis with the real data, we used the gut microbiome data [103]. The metagenomic shotgun-sequencing data for two samples (ERR2017411, ERR2017412) was downloaded from the European Bioinformatics Institute (EBI) database under the accession code ERP023788.

All data used was paired-end reads within the Illumina platform. Synthetic data consists of 26 million reads for each dataset (10s, 100s, and 400s) with the read length of 75bp. Real data has 17 million reads for each sample with the read length of 90bp. There is a slightly different in read lengths between synthetic data and real data; however, the read lengths from 75bp to 100bp have been reported [104, 105] to produce the same alignment results.

6.2.4 Findings

Using assembled reads, four out of seven classifiers increased their precision by at most double, while maintaining similar recall; see Table 6.1. These four classifiers are Kaiju, CLARK, Kraken and MetaCache. The improvement in performance was most

Table 6.1: Precision, recall, F-1 of species-level classification of four metagenomic classifiers on three synthetic short read datasets, which are, respectively, not assembled and assembled by three assemblers.

		Kaiju			CLARK			Kraken			MetaCache		
		Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
10s	not assembled	.02	1.0	.04	.02	1.0	.05	.03	1.0	.06	.20	1.0	.33
	MEGAHIT	.50	.90	.64	1.0	1.0	1.0	1.0	1.0	1.0	.90	1.0	.95
	MetaSPAdes	.50	.90	.64	1.0	1.0	1.0	1.0	1.0	1.0	.66	1.0	.80
	Ray	.39	.90	.54	1.0	1.0	1.0	1.0	1.0	1.0	.83	1.0	.91
100s	not assembled	.18	.87	.29	.21	.98	.35	.21	.84	.34	.47	.97	.63
	MEGAHIT	.35	.87	.50	.88	.99	.93	.67	.86	.75	.78	.99	.87
	MetaSPAdes	.35	.87	.50	.69	.99	.81	.63	.86	.73	.73	.99	.84
	Ray	.25	.87	.38	.98	.99	.98	.75	.86	.80	.83	.99	.90
400s	not assembled	.84	.88	.86	.95	.99	.97	.95	.83	.88	.91	.97	.94
	MEGAHIT	.88	.88	.88	.99	.99	.99	.98	.84	.90	.97	.99	.98
	MetaSPAdes	.87	.88	.87	.98	.99	.99	.98	.84	.90	.96	.99	.98
	Ray	.95	.85	.90	.99	.99	.99	.99	.84	.91	.99	.98	.99

drastic for smaller datasets. With the dataset 10s, which consisted of 10 species, CLARK, for example, benefited from a 50x increase in precision with the same recall, when reads were assembled by any of the three assemblers. With the dataset 100s, which consisted of 100 species, CLARK benefited from 3x-4x increase in precision with the same recall, when reads were assembled by any of the three assemblers. With the dataset 400s, which consisted of 400 species, CLARK benefited from a 1.04x increase in precision with the same recall. Similarly, other three classifiers benefited from assembled reads. Kraken and MetaCache benefited from increases in both precision and recall with the larger datasets 100s and 400s.

MetaPhlAn2's performance got worse with assembled reads, compared to its performance on unassembled short reads. As seen in Figure 6.1, the overall F1 score is highest when reads were not assembled. We also observed that DUDes and GOTTCHA did not benefit from assembled reads. Figure 6.1 shows that the F1 scores of DUDes and GOTTCHA remained unchanged or decreased slightly with assembled reads. We also

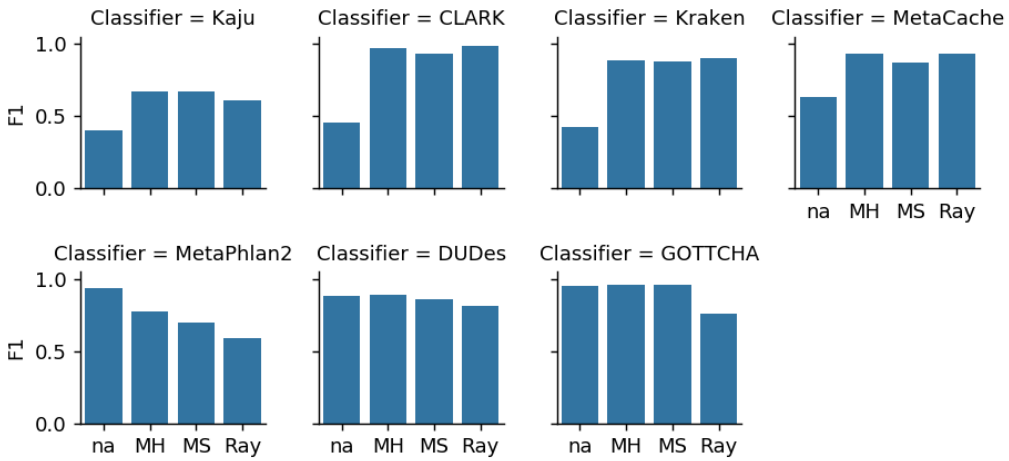

Fig. 6.1: Overall F-1 scores of species-level classification produced with short reads (abbreviated ”na”) and assembled reads by MEGAHIT(abbreviated ”MH”), metaSPdes(abbreviated ”MS”) and Ray

observed that F1 scores were highest when reads were assembled by MEGAHIT and metaSPAdes.

We observed that with synthetic, the majority of classifiers predicted much fewer species when reads were assembled; see Table 6.3. This observation likely explains the drastic observed increase in precision, while maintaining similar recall, across the board with synthetic datasets. With real datasets, we observed a similar behavior that classifiers predicted much fewer species when reads were assembled. Although we could not compute precision and recall with real data, the same trend suggests that as with synthetic data, classifiers should be much more precise when reads were assembled. This increase in precision should, similarly, be drastic when the datasets have much fewer species than the index database that were used by classifiers to classify species.

Additionally, Table 6.4 shows that with real datasets, the overall pairwise similarity decreased with assembled reads. This suggests that with assembled reads, classifiers had a higher chance of showing their uniqueness in predicting species.

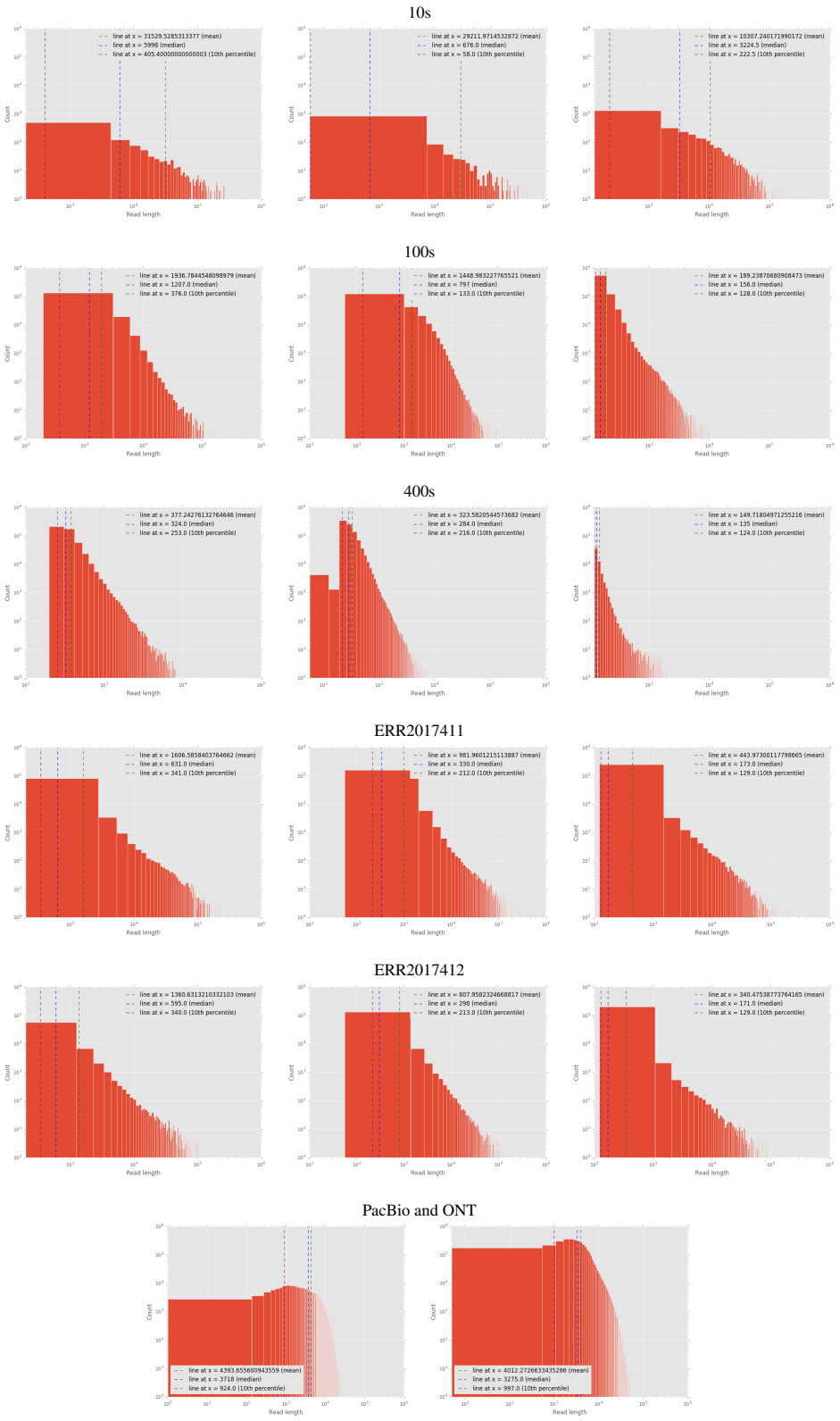

Fig. 6.2: Contig length distribution compared to PacBio and ONT long read length distribution. Contigs were assembled by different assemblers (left to right): MEGAHIT, metaSPAdes and Ray. The bottom subfigures are PacBio (left) and ONT (right) read length distribution.

Table 6.2: Assembly statistics for all assemblers on simulated (10s, 100s, 400s) and real (ERR2017411, ERR2017412) data

Statistics	Dataset	MEGAHIT	metaSPAdes	Ray
Synthetic Data				
number of contigs	10s	1,069	1,156	3,256
largest contig	10s	835,563	1,436,250	294,361
avg contig	10s	31,529.53	29,211.97	10,307.24
n50	10s	131,416	234,206	31,735
number of contigs	100s	156,074	210,765	717,512
largest contig	100s	573,139	190,202	14,995
avg contig	100s	1,936.78	1,448.98	189.24
n50	100s	3,051	2,732	177
number of contigs	400s	488,142	901,182	59,663
largest contig	400s	21,914	13,618	3,367
avg contig	400s	377.24	323.58	149.72
n50	400s	361	319	138
Real Data				
number of contigs	ERR2017411	85,426	165,252	252,974
largest contig	ERR2017411	516,770	394,993	278,191
avg contig	ERR2017411	1,606.59	981.96	443.97
n50	ERR2017411	4,063	2,820	1,620
number of contigs	ERR2017412	67,750	141,689	201,038
largest contig	ERR2017412	212,503	264,186	192,118
avg contig	ERR2017412	1,360.63	807.96	340.48
n50	ERR2017412	2,720	1,816	432

6.2.5 Discussion

In this work, we showed the promising prospect of utilizing long reads in identifying species in metagenomic samples. Long reads, used in this study, are assembled from the same short reads, which were used to compare classification performance. This was performed to remove potential side effects of different sequencing technologies. As future long-read technologies achieve fewer sequencing errors and become less expensive, their use for species classification in metagenomics should be desirable.

At presence, we have demonstrated that we can leverage the advantage of long reads by assembling short reads that would otherwise be used for species classification. We showed that at least two of the currently popular assemblers can be used for this purpose. We observed that MEGAHIT and metaSPdes produced higher N50s across

Table 6.3: Number of species predicted by each classifiers

	Kaiju	CLARK	Kraken	MetaCache	MetaPhlAn2	DUDes	GOTTCHA
26,666,674 paired-end reads (10s) length of 75bp							
n/a	3553	372	346	50	10	9	10
MEGAHIT	25	10	10	11	5	11	10
MetaSPAdes	31	10	10	15	3	13	10
Ray	36	10	10	12	8	12	10
26,667,004 paired-end reads (100s) length of 75bp							
n/a	3659	394	380	176	87	73	84
MEGAHIT	1258	95	125	108	80	71	84
MetaSPAdes	1328	122	131	115	81	72	84
Ray	2109	86	107	101	86	74	84
26,665,698 paired-end reads (400s) length of 75bp							
n/a	3707	416	405	426	402	282	390
MEGAHIT	2024	403	394	411	370	284	388
MetaSPAdes	2522	405	396	416	375	277	389
Ray	754	398	392	394	10	188	99
17,853,919 paired-end reads (ERR2017411) length of 90bp							
n/a	3654	3140	3638	1071	79	29	37
MEGAHIT	2071	1477	1537	718	29	47	25
MetaSPAdes	2618	1782	1867	797	32	33	25
Ray	2679	1630	1731	515	31	40	23
17,793,507 paired-end reads (ERR2017412) length of 90bp							
n/a	3647	3075	3651	1044	82	48	45
MEGAHIT	1653	1035	1058	611	23	33	26
MetaSPAdes	2312	1387	1423	679	39	42	29
Ray	2192	1203	1297	448	21	21	22

Table 6.4: Pairwise similarity of a method to other methods

	Kaiju	CLARK	Kraken	MetaCache	MetaPhlAn2	DUDes	GOTTCHA
17,853,919 paired-end reads (ERR2017411) length of 90bp							
n/a	0.66	0.69	0.66	0.68	0.65	0.82	0.80
MEGAHIT	0.51	0.63	0.62	0.60	0.76	0.81	0.80
MetaSPAdes	0.53	0.65	0.64	0.63	0.73	0.81	0.81
Ray	0.50	0.64	0.62	0.63	0.74	0.81	0.80
17,793,507 paired-end reads (ERR2017412) length of 90bp							
n/a	0.66	0.69	0.65	0.68	0.71	0.82	0.82
MEGAHIT	0.51	0.63	0.62	0.60	0.76	0.81	0.80
MetaSPAdes	0.53	0.65	0.64	0.63	0.73	0.81	0.81
Ray	0.50	0.64	0.62	0.63	0.74	0.81	0.80

datasets, while Ray had lower N50s. In fact, it failed to assemble reads when the datasets contained 400 species. An quick comparison between metaSPAdes and MEGAHIT assemblers across all the datasets considered in this study confirmed that metaSPAdes performs better for a smaller dataset (10s) while MEGAHIT performs better for larger datasets (100s and 400s).

We think that Kaiju, CLARK, Kraken, and MetaCache benefited from the longer reads because their approach of k-mers as unique markers to distinguish closely related species. On the other hand, MetaPhlAn2, DUDes and GOTTCHA have built-in statistical post-processing procedures that align reads to reference genomes, which appear not benefit from longer reads.

6.3 Methods

Most metagenomic classifiers, including those that we studied in this chapter, consist of two main steps. In the preprocessing step, a classifier utilizes reference genomic sequence of existing species to build an index or reference table. The index or reference table is built only once for a metagenomic environment. In the classification step, the classifier uses the index or reference table to classify metagenomic data, in the form of reads, and make predictions.

Our method interrupts this workflow by modifying the classification step. Before feeding short reads as inputs to a classifier, we assemble them into long reads. Figure 6.3 depicts the process of comparing a classifier's performance on short reads and long reads. Figure 6.3A is the standard workflow of a classifier, in which the classifier takes as input a short reads dataset and outputs species that it predicts to be present in the sample. Figure 6.3B shows a workflow, in which the same short reads are first assembled before feeding to the classifier.

Different methods may have different types of prediction outputs, which can be species label for each read, or predicted species for the entire dataset, or predicted percentages of species in the sample (in case of metagenomics profiling). Profiling

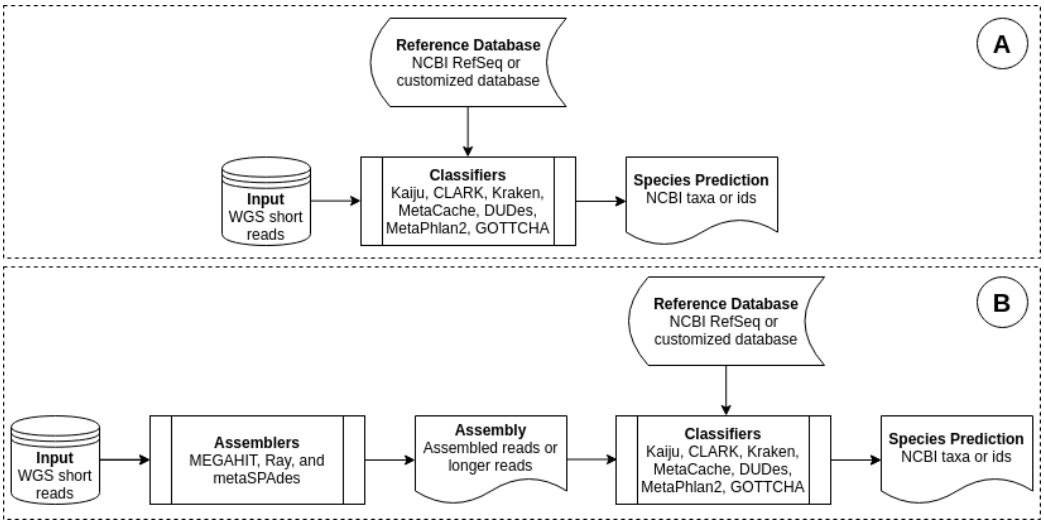

Fig. 6.3: Workflow of metagenomic classification: (A) original workflow, which uses short reads, (B) modified workflow, which uses assembled reads. Metegenomic classifiers are Kaiju, CLARK, Kraken, MetaCache, MetaPhlAn2, DUDes and GOTTCHA. Metagenomic assemblers are MEGAHIT, metaSPAdes, and Ray.

classifiers output a rank separated taxanomic profile with relative abundances, whereas binning classifiers provide sequence identification, length used in the assignment, and taxon as output. These outputs are then converted into species names. As a result, each metagenomic classifier produces a list of species as output.

Each classifier requires a reference database of genomic sequences to classify metagenomic reads into species. We used complete genomes of bacteria archaea, and viruses from NCBI to construct this database for each classifier. We removed species that were labeled unclassified or unknown because they might cause problems for taxonomic prediction [106].

For consistency, we used the NCBI taxonomy database [107] to standardize results from different classifiers. Further, for classifiers that produced strain-level predictions, we converted them to species-level predictions so that the results can be compared consistently across different classifiers.

We compared the outputs of classifiers using default parameters at the species level because not all classifiers still predicted at strain level. Species is a taxonomic rank

more relevant in clinical diagnostics or pathogen identification than genus or phylum. Although some clinical diagnosis and epidemiological tracking often requires identification of strains, genomic databases remain poorly populated below the species level [108]. Evaluation was done in a similar way to [108, 109]. For each classifier, we evaluated predicted species produced with assembled reads and predicted species produced with original short reads.

6.3.1 Classifiers

We evaluated with a set of seven metagenomic classifiers: Kaiju (version 1.7.2) [110], CLARK (version 1.2.6) [94], Kraken (version 1.1.1) [96], MetaCache (version 0.6.1) [111], MetaPhlAn2 (version 2.6.0) [97], DUDes (version 0.08) [112], and GOTTCHA (version 1.0c) [95]. The choice was motivated by recent publications comparing the performance of such tools [108].

Kaiju, CLARK, Kraken, MetaCache are k-mer based methods for metagenomic read classification. CLARK and Kraken were run with the default k-mer size of 31, while MetaCache use 16-mers by default. Kaiju was run in the fastest MEM mode (with minimum fragment length $m = 11$), as well as in the heuristic greedy mode (with minimum score $s = 65$).

On the other hand, both MetaPhlAn2 [97] and DUDes [112] have to use results of read-to-reference mapping from Bowtie2 [79]; however, for some longer contigs (several million bps), Bowtie2 (version 2.3.4.2) crahsed. We used a read mapper designed for both short and long reads: Minimap2 (version 2.17) [70] as an alternative for mapping reads to reference genomes.

While running the classifiers above, we specified the “paired-end reads” option for raw read input as well as the “singleton read” option for assembled read input.

6.3.2 Assemblers

MEGAHIT (version 1.2.9) [113], metaSPAdes (version 3.13.1) [114], Ray (version 2.3.1) [115] have been used to assemble short-reads into contigs. These tools were selected based on their popularity for assembling metagenomic reads [116].

Assemblers have been launched with (mostly) default parameters; taking a pair of FASTQ files that contains raw reads and then producing a single FASTA file that contains assembled reads for each dataset. The file names of assembled reads from MEGAHIT, metaSPAdes, and Ray are "final.contigs", "contigs" and "Contigs" respectively.

Ray parallelizes assembly computations using the Message Passing Interface (MPI) standard, a run agent "mpirun". While metaSPAdes consumes very high memory, MEGAHIT specifies multiple computational threads and optionally a graphical processing unit for improving its runtime. Due to the scope of this work, we do not report the runtime as well as memory usage of the assemblers.

6.4 Summary

We compared performance of popular metagenomic classifiers on short reads and longer reads, which are assembled from the same short reads. Using a number of popular assemblers to assemble short reads, we discovered that most classifiers made fewer predictions with longer reads and that they achieved higher classification performance on synthetic metagenomic data. Specifically, across most classifiers, we observed a significant increase in precision, while recall remained the same, resulting in higher overall classification performance. On real metagenomic data, we observed a similar trend as in the case of synthetic data that classifiers made fewer predictions. This suggested that they might have the same performance characteristics of having higher precision while maintaining the same recall with longer reads.

This finding has two main implications. First, it suggests that classifying species in metagenomic environments can be achieved with higher overall performance simply by assembling short reads. This finding can make a big impact on the many existing studies

that utilize short reads. The modification to their existing workflow is minimal, although there is an extra computational cost of assembling short reads. We showed that a number of existing assemblers could be used for the purpose of assembling short reads into longer contigs for this specific purpose.

Second, this finding suggests that it might be a good idea to consider utilizing long-read technologies in species classification for metagenomic applications. Current long-read technologies tend to have higher sequencing errors and are more expensive compared to short-read technologies. The trade-offs between the pros and cons should be investigated.

Chapter 7

Conclusion and Future Work

In this dissertation, various new algorithms for several computationally intensive tasks of genomic and metagenomic data analysis have been presented. There are still problems in this area wide open for further research. For example, realignment analysis is a very time and resource-demanding computation and in genome-wide studies (e.g. human) it can take days to complete even on high performance clusters. Moreover, as sequencing technologies continue to advance, they constantly produce more data and reads with novel characteristics (e.g. longer lengths, different error profiles), cause new computational challenges, and create exciting research questions. Here we briefly discuss several on-going and future projects related to the work presented in this dissertation.

- **Long-read data and aligner characteristics:** Single-molecule sequencing (SMS) technologies generate data with properties drastically different from that of next-generation sequencing technologies (NGS). SMS generates very long reads with very high sequencing errors. A fundamental characterization of SMS is a fruitful study, since it is not as widely adopted as NGS. Furthermore, the bioinformatics tools required to analyze SMS are not as matured or well understood, hence characterizing sequence alignment methods is an important prerequisite. In an on-going project, we are looking at pair-wise alignments between long-reads and a reference genome from various popular long-read aligners across multiple samples. By measuring the sequencing quality of alignments by multiple aligners, we compute statistics of sequencing error rates and presence the pattern across samples.
- **Genome-wide SVs/CNVs discoveries:** In one chapter of this disertation, a mCNV was discovered in the human genome GIAB data on a chromosome 19 neutral region. This work can be expanded on resolving breakpoints of complex structural variation to other complex loci to understand them. Such highly

complex loci with overlapping CNVs and loci containing multiple classes of structural variation still require locus specific strategies in order to be discovered and typed. This is why complex structural genomics is focused on developing genome-wide approaches to typing and incorporating complex structural haplotypes into association studies. Therefore, we have focused on capturing these variations in larger genome-wide sequence data in an on-going project.

- **Single cell data applications:** High-throughput DNA sequencing technologies have recently lead to many noteworthy advances in single-cell studies. Single-cell sequencing enables comprehensive genome-wide copy number profiling of thousands of cells of various evolutionary stages and lineage. In humans, natural-mutation markers that are created within cells as they proliferate and age are the primary determinants. It is now possible to trace human lineages in normal, noncancerous cells with a variety of data types using natural variations in the nuclear and mitochondrial DNA as well as variations in DNA methylation status. We plan to have an extended version of AGE which can work with single cell data for defining breakpoints based on statistics of CNVs in each cell.

With the rapid advance in sequencing technologies, biomedical data is being generated at an unprecedented rate. Accurate, efficient, and well designed computational algorithms will ultimately boost the success of personalized medicine, which involves a full end-to-end solution, from sample preparation to data interpretation. In addition, it relies on deep domain expertise, as well as highly efficient and accurate computational solutions to transform the gigantic information into meaningful insights.

REFERENCES

[1] E. L. Van Dijk, H. Auger, Y. Jaszczyszyn, and C. Thermes, "Ten years of next-generation sequencing technology," *Trends in genetics*, vol. 30, no. 9, pp. 418–426, 2014.

[2] H. Y. Lam, M. J. Clark, R. Chen, R. Chen, G. Natsoulis, M. O'huallachain, F. E. Dewey, L. Habegger, E. A. Ashley, M. B. Gerstein, *et al.*, "Performance comparison of whole-genome sequencing platforms," *Nature biotechnology*, vol. 30, no. 1, pp. 78–82, 2012.

[3] M. G. Ross, C. Russ, M. Costello, A. Hollinger, N. J. Lennon, R. Hegarty, C. Nusbaum, and D. B. Jaffe, "Characterizing and measuring bias in sequence data," *Genome biology*, vol. 14, no. 5, p. R51, 2013.

[4] M. A. Quail, M. Smith, P. Coupland, T. D. Otto, S. R. Harris, T. R. Connor, A. Bertoni, H. P. Swerdlow, and Y. Gu, "A tale of three next generation sequencing platforms: comparison of ion torrent, pacific biosciences and illumina miseq sequencers," *BMC genomics*, vol. 13, no. 1, p. 341, 2012.

[5] C.-S. Chin, D. H. Alexander, P. Marks, A. A. Klammer, J. Drake, C. Heiner, A. Clum, A. Copeland, J. Huddleston, E. E. Eichler, *et al.*, "Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data," *Nature methods*, vol. 10, no. 6, p. 563, 2013.

[6] J. Eid, A. Fehr, J. Gray, K. Luong, J. Lyle, G. Otto, P. Peluso, D. Rank, P. Baybayan, B. Bettman, *et al.*, "Real-time dna sequencing from single polymerase molecules," *Science*, vol. 323, no. 5910, pp. 133–138, 2009.

[7] E. E. Schadt, S. Turner, and A. Kasarskis, "A window into third-generation sequencing," *Human molecular genetics*, vol. 19, no. R2, pp. R227–R240, 2010.

[8] M. David, M. Dzamba, D. Lister, L. Ilie, and M. Brudno, "Shrimp2: sensitive yet practical short read mapping," *Bioinformatics*, vol. 27, no. 7, pp. 1011–1012, 2011.

[9] C. Alkan, J. M. Kidd, T. Marques-Bonet, G. Aksay, F. Antonacci, F. Hormozdiari, J. O. Kitzman, C. Baker, M. Malig, O. Mutlu, *et al.*, "Personalized copy number and segmental duplication maps using next-generation sequencing," *Nature genetics*, vol. 41, no. 10, pp. 1061–1067, 2009.

[10] J. C. Mu, H. Jiang, A. Kiani, M. Mohiyuddin, N. B. Asadi, and W. H. Wong, "Fast and accurate read alignment for resequencing," *Bioinformatics*, vol. 28, no. 18, pp. 2366–2373, 2012.

[11] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with bowtie 2," *Nature Methods*, vol. 9, no. 4, pp. 357–359, 2012.

[12] H. Li and R. Durbin, "Fast and accurate long-read alignment with burrows–wheeler transform," *Bioinformatics*, vol. 26, no. 5, pp. 589–595, 2010.

[13] Y. Liu and B. Schmidt, "Long read alignment based on maximal exact match seeds," *Bioinformatics*, vol. 28, no. 18, pp. i318–i324, 2012.

[14] E. Siragusa, D. Weese, and K. Reinert, "Fast and accurate read mapping with approximate seeds and multiple backtracking," *Nucl Acids Res*, vol. 41, no. 7, p. e78, 2013.

[15] H. Ponstingl and Z. Ning, "Smalt—a new mapper for dna sequencing reads," *F1000 Posters*, vol. 1, p. 313, 2010.

[16] H. Li and N. Homer, "A survey of sequence alignment algorithms for next-generation sequencing," *Briefings in bioinformatics*, vol. 11, no. 5, pp. 473–483, 2010.

[17] J. Shang, F. Zhu, W. Vongsangnak, Y. Tang, W. Zhang, and B. Shen, "Evaluation and comparison of multiple aligners for next-generation sequencing data analysis," *BioMed research international*, vol. 2014, p. 309650, 2014.

[18] X. Yu, K. Guda, J. Willis, M. Veigl, Z. Wang, M. D. Markowitz, *et al.*, "How do alignment programs perform on sequencing data with varying qualities and from repetitive regions?" *BioData mining*, vol. 5, p. 6, 2012.

[19] H. Lee and M. C. Schatz, "Genomic dark matter: the reliability of short read mapping illustrated by the genome mappability score," *Bioinformatics*, vol. 28, no. 16, pp. 2097–2105, 2012.

[20] W. Li, J. Freudenberg, and P. Miramontes, "Diminishing return for increased mappability with longer sequencing reads: implications of the k-mer distributions in the human genome." *BMC bioinformatics*, vol. 15, p. 2, Jan 2014. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/24386976

[21] A. Lempel and J. Ziv, "On the complexity of finite sequences," *Information Theory, IEEE Transactions on*, vol. 22, no. 1, pp. 75–81, 1976.

[22] J. Ziv, "Coding theorems for individual sequences," *Information Theory, IEEE Transactions on*, vol. 24, no. 4, pp. 405–412, 1978.

[23] F. Nan and D. Adjeroh, "On complexity measures for biological sequences," in *Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE*. IEEE, 2004, pp. 522–526.

[24] V. Becher and P. Heiber, "A linearly computable measure of string complexity," *Theoretical Computer Science*, vol. 438, pp. 62–73, 2012.

[25] B. Chor, D. Horn, N. Goldman, T. Levy, and T. Massingham, "Genomic dna k-mer spectra: models and modalities," *Genome Biology*, vol. 10, no. 10, p. R108, 2009.

[26] S. Kurtz, A. Narechania, J. C. Stein, and D. Ware, "A new method to compute k-mer frequencies and its application to annotate large repetitive plant genomes," *BMC genomics*, vol. 9, no. 1, p. 517, 2008.

[27] N. E. Whiteford, N. J. Haslam, G. Weber, A. Prugel-Bennett, J. W. Essex, C. Neylon, *et al.*, "Visualizing the repeat structure of genomic sequences," *Complex Systems*, vol. 17, no. 4, pp. 381–398, 2008.

[28] M. Aboy, R. Hornero, D. Abásolo, and D. Alvarez, "Interpretation of the lempel-ziv complexity measure in the context of biomedical signal analysis." *IEEE transactions on bio-medical engineering*, vol. 53, no. 11, pp. 2282–2288, 2006. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/17073334

[29] O. G. Troyanskaya, O. Arbell, Y. Koren, G. M. Landau, and A. Bolshoy, "Sequence complexity profiles of prokaryotic genomic sequences: A fast algorithm for calculating linguistic complexity," *Bioinformatics*, vol. 18, no. 5, pp. 679–688, 2002.

[30] T. Kasai, G. Lee, H. Arimura, S. Arikawa, and K. Park, "Linear-time longest-common-prefix computation in suffix arrays and its applications," in *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching*, ser. Lecture Notes in Computer Science, vol. 2089, 2001, pp. 181–192.

[31] J. Kärkkäinen, P. Sanders, and S. Burkhardt, "Linear work suffix array construction," *J. ACM*, vol. 53, pp. 918–936, 2006.

[32] G. Rizk and D. Lavenier, "Gassst: global alignment short sequence search tool," *Bioinformatics*, vol. 26, no. 20, pp. 2534–2540, 2010.

[33] R. Li, Y. Li, K. Kristiansen, and J. Wang, "Soap: short oligonucleotide alignment program," *Bioinformatics*, vol. 24, no. 5, pp. 713–714, 2008.

[34] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and . G. P. D. P. Subgroup, "The sequence alignment/map format and samtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[35] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[36] I. H. . Consortium *et al.*, "Integrating common and rare genetic variation in diverse human populations," *Nature*, vol. 467, no. 7311, pp. 52–58, 2010.

[37] . G. P. Consortium *et al.*, "A map of human genome variation from population-scale sequencing," *Nature*, vol. 467, no. 7319, pp. 1061–1073, 2010.

[38] D. Altshuler, L. D. Brooks, A. Chakravarti, F. S. Collins, M. J. Daly, P. Donnelly, R. Gibbs, J. Belmont, A. Boudreau, S. Leal, *et al.*, "A haplotype map of the human genome," *Nature*, vol. 437, no. 7063, pp. 1299–1320, 2005.

[39] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, *et al.*, "The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data," *Genome research*, vol. 20, no. 9, pp. 1297–1303, 2010.

[40] H. Y. Lam, C. Pan, M. J. Clark, P. Lacroute, R. Chen, R. Haraksingh, M. O'Huallachain, M. B. Gerstein, J. M. Kidd, C. D. Bustamante, *et al.*, "Detecting and annotating genetic variations using the hugeseq pipeline," *Nature biotechnology*, vol. 30, no. 3, pp. 226–229, 2012.

[41] W. Wang, Z. Wei, T.-W. Lam, and J. Wang, "Next generation sequencing has lower sequence coverage and poorer snp-detection capability in the regulatory regions," *Scientific reports*, vol. 1, 2011.

[42] B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg, "Searching for snps with cloud computing," *Genome Biol*, vol. 10, no. 11, p. R134, 2009.

[43] R. Li, Y. Li, X. Fang, H. Yang, J. Wang, K. Kristiansen, and J. Wang, "Snp detection for massively parallel whole-genome resequencing," *Genome research*, vol. 19, no. 6, pp. 1124–1132, 2009.

[44] K. Ye, M. H. Schulz, Q. Long, R. Apweiler, and Z. Ning, "Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads," *Bioinformatics*, vol. 25, no. 21, pp. 2865–2871, 2009.

[45] C. A. Albers, G. Lunter, D. G. MacArthur, G. McVean, W. H. Ouwehand, and R. Durbin, "Dindel: accurate indel calls from short-read data," *Genome research*, vol. 21, no. 6, pp. 961–973, 2011.

[46] K. Chen, J. W. Wallis, M. D. McLellan, D. E. Larson, J. M. Kalicki, C. S. Pohl, S. D. McGrath, M. C. Wendl, Q. Zhang, D. P. Locke, *et al.*, "Breakdancer: an algorithm for high-resolution mapping of genomic structural variation," *Nature methods*, vol. 6, no. 9, pp. 677–681, 2009.

[47] D. C. Koboldt, K. Chen, T. Wylie, D. E. Larson, M. D. McLellan, E. R. Mardis, G. M. Weinstock, R. K. Wilson, and L. Ding, "Varscan: variant detection in massively parallel sequencing of individual and pooled samples," *Bioinformatics*, vol. 25, no. 17, pp. 2283–2285, 2009.

[48] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[49] A. M. Meynert, M. Ansari, D. R. FitzPatrick, and M. S. Taylor, "Variant detection sensitivity and biases in whole genome and exome sequencing," *BMC Bioinformatics*, vol. 15, no. 1, pp. 1–11, 2014.

[50] N. Rieber, M. Zapatka, B. Lasitschka, D. Jones, P. Northcott, B. Hutter, N. Jäger, M. Kool, M. Taylor, P. Lichter, S. Pfister, S. Wolf, B. Brors, and R. Eils, "Coverage bias and sensitivity of variant calling for four whole-genome sequencing technologies," *PLoS ONE*, vol. 8, no. 6, pp. 1–11, 06 2013.

[51] P. M. Gontarz, J. Berger, and C. F. Wong, "Srmapper: a fast and sensitive genome-hashing alignment tool," *Bioinformatics*, vol. 29, no. 3, pp. 316–321, 2013.

[52] D. Weese, M. Holtgrewe, and K. Reinert, "Razers 3: faster, fully sensitive read mapping," *Bioinformatics*, vol. 28, no. 20, pp. 2592–2599, 2012.

[53] R. Li, C. Yu, Y. Li, T.-W. Lam, S.-M. Yiu, K. Kristiansen, and J. Wang, "Soap2: an improved ultrafast tool for short read alignment," *Bioinformatics*, vol. 25, no. 15, pp. 1966–1967, 2009.

[54] H. Li, "Toward better understanding of artifacts in variant calling from high-coverage samples," *Bioinformatics*, vol. 30, no. 20, pp. 2843–2851, 2014.

[55] H. Y. Lam, X. J. Mu, A. M. Stütz, A. Tanzer, P. D. Cayting, M. Snyder, P. M. Kim, J. O. Korbel, and M. B. Gerstein, "Nucleotide-resolution analysis of structural variants using breakseq and a breakpoint library," *Nature biotechnology*, vol. 28, no. 1, p. 47, 2010.

[56] A. Abyzov, S. Li, D. R. Kim, M. Mohiyuddin, A. M. Stütz, N. F. Parrish, X. J. Mu, W. Clark, K. Chen, M. Hurles, *et al.*, "Analysis of deletion breakpoints from 1,092 humans reveals details of mutation mechanisms," *Nature communications*, vol. 6, p. 7256, 2015.

[57] F. J. Sedlazeck, P. Rescheneder, M. Smolka, H. Fang, M. Nattestad, A. von Haeseler, and M. C. Schatz, "Accurate detection of complex structural variations using single-molecule sequencing," *Nature methods*, vol. 15, no. 6, p. 461, 2018.

[58] A. Abyzov and M. Gerstein, "Age: defining breakpoints of genomic structural variants at single-nucleotide resolution, through optimal alignments with gap excision," *Bioinformatics*, vol. 27, no. 5, pp. 595–603, 2011.

[59] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Communications of the ACM*, vol. 18, no. 6, pp. 341–343, 1975.

[60] K.-M. Chao, R. C. Hardison, and W. Miller, "Recent developments in linear-space alignment methods: A survey," *Journal of Computational Biology*, vol. 1, no. 4, pp. 271–291, 1994.

[61] J. M. Kidd, T. Graves, T. L. Newman, R. Fulton, H. S. Hayden, M. Malig, J. Kallicki, R. Kaul, R. K. Wilson, and E. E. Eichler, "A human genome structural variation sequencing resource reveals insights into mutational mechanisms," *Cell*, vol. 143, no. 5, pp. 837–847, 2010.

[62] Q. Tran, S. Gao, and V. Phan, "Analysis of optimal alignments unfolds aligners bias in existing variant profiles," in *BMC bioinformatics*, vol. 17, no. 13. BioMed Central, 2016, p. 349.

[63] J. M. Zook, D. Catoe, J. McDaniel, L. Vang, N. Spies, A. Sidow, Z. Weng, Y. Liu, C. E. Mason, N. Alexander, *et al.*, "Extensive sequencing of seven human genomes to characterize benchmark reference materials," *Scientific data*, vol. 3, p. 160025, 2016.

[64] A. Abyzov, A. E. Urban, M. Snyder, and M. Gerstein, "Cnvnator: an approach to discover, genotype, and characterize typical and atypical cnvs from family and population genome sequencing," *Genome research*, vol. 21, no. 6, pp. 974–984, 2011.

[65] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, *et al.*, "The sequence alignment/map format and samtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[66] B. Lau, M. Mohiyuddin, J. C. Mu, L. T. Fang, N. Bani Asadi, C. Dallett, and H. Y. Lam, "Longislnd: in silico sequencing of lengthy and noisy datatypes," *Bioinformatics*, vol. 32, no. 24, pp. 3829–3832, 2016.

[67] J. Ruan and H. Li, "Fast and accurate long-read assembly with wtdbg2," *Nature Methods*, pp. 1–4, 2019.

[68] J. Seward, N. Nethercote, and J. Weidendorfer, *Valgrind 3.3-advanced debugging and profiling for gnu/linux applications*. Network Theory Ltd., 2008.

[69] C. L. Usher and S. A. McCarroll, "Complex and multi-allelic copy number variation in human disease," *Briefings in functional genomics*, vol. 14, no. 5, pp. 329–338, 2015.

[70] H. Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinformatics*, vol. 34, no. 18, pp. 3094–3100, 2018.

[71] S. Gao, Q. Tran, and V. Phan, "Understand effective coverage by mapped reads using genome repeat complexity," in *Proceedings of 11th International Conference on Bioinformatics and Computational Biology*, vol. 60, 2019, pp. 65–73.

[72] Q. Tran, S. Gao, N. S. Vo, and V. Phan, "Repeat complexity of genomes as a means to predict the performance of short-read aligners," in *Proceedings of the 8th International Conference on Bioinformatics and Computational Biology (BiCOB)*, 2016.

[73] A. Bankevich, S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski, *et al.*, "Spades: a new genome assembly algorithm and its applications to single-cell sequencing," *Journal of computational biology*, vol. 19, no. 5, pp. 455–477, 2012.

[74] N. Altemose, K. H. Miga, M. Maggioni, and H. F. Willard, "Genomic characterization of large heterochromatic gaps in the human genome assembly," *PLoS computational biology*, vol. 10, no. 5, 2014.

[75] M. S. Lindner and B. Y. Renard, "Metagenomic profiling of known and unknown microbes with microbegps," *PloS one*, vol. 10, no. 2, p. e0117711, 2015.

[76] G. Muyzer, E. C. De Waal, and A. G. Uitterlinden, "Profiling of complex microbial populations by denaturing gradient gel electrophoresis analysis of polymerase chain reaction-amplified genes coding for 16s rrna." *Applied and environmental microbiology*, vol. 59, no. 3, pp. 695–700, 1993.

[77] E. Stackebrandt and B. Goebel, "Taxonomic note: a place for dna-dna reassociation and 16s rrna sequence analysis in the present species definition in bacteriology," *International Journal of Systematic and Evolutionary Microbiology*, vol. 44, no. 4, pp. 846–849, 1994.

[78] M. Langille, J. Zaneveld, G. Caporaso, D. McDonald, D. Knights, J. Reyes, J. Clemente, D. Burkepile, R. Thurberand, R. Knight, R. Beiko, and C. Huttenhower, "Predictive functional profiling of microbial communities using 16s rrna marker gene sequences," *Nature Biotechnology*, 2013.

[79] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with bowtie 2," *Nature methods*, vol. 9, no. 4, pp. 357–359, 2012.

[80] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with bwa-mem," *arXiv preprint arXiv:1303.3997*, 2013.

[81] C.-M. Liu, T. Wong, E. Wu, R. Luo, S.-M. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, *et al.*, "Soap3: ultra-fast gpu-based parallel alignment tool for short reads," *Bioinformatics*, vol. 28, no. 6, pp. 878–879, 2012.

[82] N. S. Vo, Q. Tran, N. Niraula, and V. Phan, "Randal: a randomized approach to aligning dna sequences to reference genomes," *BMC genomics*, vol. 15, no. 5, p. S2, 2014.

[83] B. A. Galler and M. J. Fisher, "An improved equivalence algorithm," *Communications of the ACM*, vol. 7, no. 5, pp. 301–303, 1964.

[84] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal, *A Survey of Algorithms for Dense Subgraph Discovery*. Boston, MA: Springer US, 2010, pp. 303–336.

[85] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2837–2854, 2010.

[86] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[87] A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure." in *EMNLP-CoNLL*, vol. 7, 2007, pp. 410–420.

[88] F. E. Angly, D. Willner, F. Rohwer, P. Hugenholtz, and G. W. Tyson, "Grinder: a versatile amplicon and shotgun sequence simulator," *Nucleic acids research*, p. gks251, 2012.

[89] M. J. Bonder, A. Kurilshikov, E. F. Tigchelaar, Z. Mujagic, F. Imhann, A. V. Vila, P. Deelen, T. Vatanen, M. Schirmer, S. P. Smeekens, *et al.*, "The effect of host genetics on the gut microbiome," *Nature genetics*, vol. 48, no. 11, pp. 1407–1412, 2016.

[90] B. J. Shapiro, J. Friedman, O. X. Cordero, S. P. Preheim, S. C. Timberlake, G. Szabó, M. F. Polz, and E. J. Alm, "Population genomics of early events in the ecological differentiation of bacteria," *science*, vol. 336, no. 6077, pp. 48–51, 2012.

[91] N. Kashtan, S. E. Roggensack, S. Rodrigue, J. W. Thompson, S. J. Biller, A. Coe, H. Ding, P. Marttinen, R. R. Malmstrom, R. Stocker, *et al.*, "Single-cell genomics reveals hundreds of coexisting subpopulations in wild prochlorococcus," *Science*, vol. 344, no. 6182, pp. 416–420, 2014.

[92] E. S. Snitkin, A. M. Zelazny, C. I. Montero, F. Stock, L. Mijares, P. R. Murray, J. A. Segre, *et al.*, "Genome-wide recombination drives diversification of epidemic strains of acinetobacter baumannii," *Proceedings of the National Academy of Sciences*, vol. 108, no. 33, pp. 13 758–13 763, 2011.

[93] M. J. Rosen, M. Davison, D. Bhaya, and D. S. Fisher, "Fine-scale diversity and extensive recombination in a quasisexual bacterial population occupying a broad niche," *Science*, vol. 348, no. 6238, pp. 1019–1023, 2015.

[94] R. Ounit, S. Wanamaker, T. J. Close, and S. Lonardi, "Clark: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers," *BMC genomics*, vol. 16, no. 1, p. 236, 2015.

[95] T. A. K. Freitas, P.-E. Li, M. B. Scholz, and P. S. Chain, "Accurate read-based metagenome characterization using a hierarchical suite of unique signatures," *Nucleic acids research*, p. gkv180, 2015.

[96] D. E. Wood and S. L. Salzberg, "Kraken: ultrafast metagenomic sequence classification using exact alignments," *Genome biology*, vol. 15, no. 3, p. R46, 2014.

[97] D. T. Truong, E. A. Franzosa, T. L. Tickle, M. Scholz, G. Weingart, E. Pasolli, A. Tett, C. Huttenhower, and N. Segata, "Metaphlan2 for enhanced metagenomic taxonomic profiling," *Nature methods*, vol. 12, no. 10, p. 902, 2015.

[98] T. Mantere, S. Kersten, and A. Hoischen, "Long-read sequencing emerging in medical genetics," *Frontiers in genetics*, vol. 10, p. 426, 2019.

[99] A. Ameur, W. P. Kloosterman, and M. S. Hestand, "Single-molecule sequencing: towards clinical applications," *Trends in biotechnology*, vol. 37, no. 1, pp. 72–85, 2019.

[100] W. Pootakham, W. Mhuantong, T. Yoocha, L. Putchim, C. Sonthirod, C. Naktang, N. Thongtham, and S. Tangphatsornruang, "High resolution profiling of coral-associated bacterial communities using full-length 16s rrna sequence data from pacbio smrt sequencing system," *Scientific reports*, vol. 7, no. 1, pp. 1–14, 2017.

[101] L. M. Petersen, I. W. Martin, W. E. Moschetti, C. M. Kershaw, and G. J. Tsongalis, "Third-generation sequencing in the clinical laboratory: Exploring the advantages and challenges of nanopore sequencing," *Journal of Clinical Microbiology*, vol. 58, no. 1, 2019.

[102] D. R. Mende, A. S. Waller, S. Sunagawa, A. I. Järvelin, M. M. Chan, M. Arumugam, J. Raes, and P. Bork, "Assessment of metagenomic assembly using simulated next generation sequencing data," *PloS one*, vol. 7, no. 2, p. e31386, 2012.

[103] Z. Jie, H. Xia, S.-L. Zhong, Q. Feng, S. Li, S. Liang, H. Zhong, Z. Liu, Y. Gao, H. Zhao, *et al.*, "The gut microbiome in atherosclerotic cardiovascular disease," *Nature communications*, vol. 8, no. 1, pp. 1–12, 2017.

[104] V. Phan, S. Gao, Q. Tran, and N. S. Vo, "How genome complexity can explain the hardness of aligning reads to genomes," in *2014 IEEE 4th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)*. IEEE, 2014, pp. 1–2.

[105] Q. Tran, S. Gao, N. S. Vo, and V. Phan, "A linear model for predicting performance of short-read aligners using genome complexity," *BMC bioinformatics*, vol. 16, no. 15, p. P17, 2015.

[106] J. C. Wooley, A. Godzik, and I. Friedberg, "A primer on metagenomics," *PLoS Comput Biol*, vol. 6, no. 2, p. e1000667, 2010.

[107] S. Federhen, "The ncbi taxonomy database," *Nucleic acids research*, vol. 40, no. D1, pp. D136–D143, 2011.

[108] A. Sczyrba, P. Hofmann, P. Belmann, D. Koslicki, S. Janssen, J. Dröge, I. Gregor, S. Majda, J. Fiedler, E. Dahms, *et al.*, "Critical assessment of metagenome interpretationa benchmark of metagenomics software," *Nature methods*, vol. 14, no. 11, p. 1063, 2017.

[109] A. B. McIntyre, R. Ounit, E. Afshinnekoo, R. J. Prill, E. Hénaff, N. Alexander, S. S. Minot, D. Danko, J. Foox, S. Ahsanuddin, *et al.*, "Comprehensive benchmarking and ensemble approaches for metagenomic classifiers," *Genome biology*, vol. 18, no. 1, p. 182, 2017.

[110] P. Menzel, K. L. Ng, and A. Krogh, "Fast and sensitive taxonomic classification for metagenomics with kaiju," *Nature communications*, vol. 7, no. 1, pp. 1–9, 2016.

[111] A. Müller, C. Hundt, A. Hildebrandt, T. Hankeln, and B. Schmidt, "Metacache: context-aware classification of metagenomic reads using minhashing," *Bioinformatics*, vol. 33, no. 23, pp. 3740–3748, 2017.

[112] V. C. Piro, M. S. Lindner, and B. Y. Renard, "Dudes: a top-down taxonomic profiler for metagenomics," *Bioinformatics*, vol. 32, no. 15, pp. 2272–2280, 2016.

[113] D. Li, C.-M. Liu, R. Luo, K. Sadakane, and T.-W. Lam, "Megahit: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph," *Bioinformatics*, vol. 31, no. 10, pp. 1674–1676, 2015.

[114] S. Nurk, D. Meleshko, A. Korobeynikov, and P. A. Pevzner, "metaspades: a new versatile metagenomic assembler," *Genome research*, vol. 27, no. 5, pp. 824–834, 2017.

[115] S. Boisvert, F. Raymond, É. Godzaridis, F. Laviolette, and J. Corbeil, "Ray meta: scalable de novo metagenome assembly and profiling," *Genome biology*, vol. 13, no. 12, p. R122, 2012.

[116] M. Ayling, M. D. Clark, and R. M. Leggett, "New approaches for metagenome assembly with short reads," *Briefings in bioinformatics*, 2019.

Appendix A

Source Code Availability

All the algorithms have been open-sourced in the following repositories on Github.

- Short-read Performance Prection:

  https://github.com/vtphan/shortread-alignment-prediction

- Optimal Alignment Analysis:

  https://github.com/Coaxecva/Indel-Analysis

- Unknown Bacteria Hunter:

  https://github.com/Coaxecva/Unknown-Bacteria-Hunter

- LongAGE:

  https://github.com/Coaxecva/LongAGE

Appendix B

Publications

Journals

1. **Quang Tran**, Alexej Abyzov. LongAGE: Defining Breakpoints of Genomic Structural Variants through Optimal and Memory Efficient Alignments of Long Reads *(Bioinformatics 2020)*
2. **Quang Tran**, Diem-Trang Pham, Vinhthuy Phan. Using 16S rRNA Gene as Marker to Detect Unknown Bacteria in Microbial Communities *(BMC Bioinformatics 2017)*
3. **Quang Tran**, Shanshan Gao, Vinhthuy Phan. Analysis of Optimal Alignments Unfolds Aligners Bias in Existing Variant Profiles. **MCBIOS 2016 Best Paper Award Runner-up** *(BMC Bioinformatics 2016)*
4. Vinhthuy Phan, Shanshan Gao, **Quang Tran**, Nam S. Vo. How Genome Complexity Can Explain the Hardness of Aligning Reads to Genomes *(BMC Genomics 2015)*
5. Nam S. Vo, **Quang Tran**, Nobal Niraula, Vinhthuy Phan. RandAL: A Randomized Approach to Aligning DNA Sequences to Reference Genomes *(BMC Genomics 2014)*

Conference

1. **Quang Tran**, Vinhthuy Phan. Assembling Reads Improves Taxonomic Classification of Species. *The 8th International Conference on Intelligent Biology and Medicine (ICIBM 2020)*
2. Shanshan Gao, **Quang Tran**, Vinhthuy Phan. Understand Effective Coverage

by Mapped Reads using Genome Repeat Complexity. *The 11th International Conference on Bioinformatics and Computational Biology (BICOB 2019)*

3. **Quang Tran**, Shanshan Gao, Nam S. Vo, Vinhthuy Phan. Repeat Complexity of Genomes as a Means to Predict the Performance of Short-read Aligners. *The 8th International Conference on Bioinformatics and Computational Biology (BICOB 2016)*
4. Vinhthuy Phan, Shanshan Gao, **Quang Tran**, Nam S. Vo. How Genome Complexity Can Explain the Hardness of Aligning Reads to Genomes. *The 4th IEEE International Conference on Computational Advances in Bio and Medical Sciences (ICCABS 2014)*
5. Nam S. Vo, **Quang Tran**, Nobal Niraula, Vinhthuy Phan. A Randomized Algorithm for Aligning DNA Sequences to Reference Genomes. *The 3rd IEEE International Conference on Computational Advances in Bio and Medical Sciences (ICCABS 2013)*

Poster

1. **Quang Tran** and Vinhthuy Phan. Assembling reads improves taxonomic classification of species. *ISMB Microbiome Community of Special Interest (COSI) 2020 (ISMB 2020)*
2. **Quang Tran**, Diem-Trang Pham and Vinhthuy Phan. Dimension Reduction Methods for Metagenomic Data. *Memphis DATA 2019-A Data Science Conference (www.memphis-data.org)*
3. **Quang Tran**, Shanshan Gao, Nam S. Vo, Vinhthuy Phan. A Linear Model for Predicting Performance of Short-read Aligners based on Repeat Complexity of Genomes. *UT-KBRIN Bioinformatics Summit 2015 (BMC Bioinformatics 2015)*

4. **Quang Tran**, Vinhthuy Phan. Alignment of Short Reads to Multiple Genomes Using Hashing. *UT-KBRIN Bioinformatics Summit 2014 (BMC Bioinformatics 2014)*
5. Nam S. Vo, **Quang Tran**, Vinhthuy Phan. An Integrated Approach for SNP Calling Based on Population of Genomes. *UT-KBRIN Bioinformatics Summit 2014 (BMC Bioinformatics 2014)*
6. Vinhthuy Phan, Shanshan Gao, **Quang Tran**, Nam S. Vo. Predicting Performance of Short-Read Aligners Based on Genome Complexity. *The 11th MidSouth Computational Biology and Bioinformatics Society (MCBIOS 2014)*

Appendix C

Reviewers' Comments

"The manuscript by Tran and Abyzov is devoted to the important problem of the identification of SVs in sequencing data. Despite the biological importance of this type of genome alterations and their accurate detection still remains a problem. The described approach is a memory-efficient implementation of previously reported AGE. This is important because the utilization of AGE is limited due to memory requirements. The authors demonstrated that the new algorithm is significantly more efficient in memory usage. Although this improvement comes at the cost of longer analysis time, this is not critical. The manuscript is well structured and clearly written. "

— *An Anonymous Reviewer* (Bioinformatics journal)

"The manuscript by Tran and Abyzov introduced a new tool LongAGE, which is a memory-efficient implementation of AGE, to resolve breakpoints of SVs embedded into segmental duplications on the PacBio long read dataset. Defining the precise breakpoint at the basepair resolution is always challenging. LongAGE is based on the Hirschberg algorithm and significantly improves memory usage compared to AGE, which allow users to realign long reads to find the precise breakpoint even on a laptop. With the long reads sequencing being widely employed in research for a variety of applications, this tool will be interesting and useful for the SV research community. "

— *An Anonymous Reviewer* (Bioinformatics journal)

"The paper compares the performance of popular metagenomics classifiers on short reads and longer reads assembled from the short reads. The findings can be very helpful for metagenomics analysis. The paper is very well written and easy to follow. I really appreciate the authors' efforts in comparing not only multiple metagenomics classifiers but also multiple assemblers."

— *An Anonymous Reviewer* (Genes journal)