6-4-2015

Fast and Sensitive Genome-Hashing Software and its Application in Using NGS as a Detection Agent for Bacterial Presence in Oral Metagenomic Samples

Paul Michael Gontarz
University of Missouri-St. Louis, paul_gontarz@yahoo.com

Fast and Sensitive Genome-Hashing Software and its Application in Using NGS as a Detection Agent for Bacterial Presence in Oral Metagenomic Samples

A Dissertation

By

Paul Gontarz

M.S. Chemistry, University of Missouri - Saint Louis, 2012
B.S. Chemistry with Emphasis in Biochemistry, Lindenwood University, 2010
B.S. Mathematics, Lindenwood University, 2010

A Thesis Submitted to the Graduate School at the University of Missouri-St. Louis
in partial fulfillment of the requirements for the degree
Doctor of Philosophy in Chemistry

June 2015

Advisory Committee

Chung Wong, Ph.D.
Chairperson

James Bashkin, Ph.D.

Cynthia Dupureur, Ph.D.

Michael Nichols, Ph.D.

Abstract

Fast and Sensitive Genome-Hashing Software and its Application in Using NGS as a Detection Agent for Bacterial Presence in Oral Metagenomic Samples

Paul Gontarz, M.S., University of Missouri, St. Louis, MO, USA

Chair of Committee: Dr. Chung Wong

Next generation sequencing has increased the throughput of sequenced DNA into the range of billions of nucleotides sequenced per day. With the increased speed of DNA sequencing and the short length of reads produced by next generation sequencers, a significant challenge has been created in quickly and accurately assembling the hundreds of millions of short reads created by modern sequencing instruments into their full genomic sequences. With the increase in throughput in next generation sequencing and the decrease in time and cost to perform DNA sequencing, novel applications for DNA sequencing are being considered. Among them is a methodology by which DNA sequencing can be used as a diagnostic or detection tool for bacterial infection or presence.

Here, the implementation, characteristics, and deployment of a novel, genome-hashing alignment algorithm for quickly performing reference-based alignment is described. This algorithm, SRmapper, is shown to be between two-fold to eight-fold faster than a current and popular alignment algorithm, BWA, while retaining a similar fraction of reads aligned to human reference genome. SRmapper demonstrates a capability to align approximately 150 billion nucleotides per processor day on an Intel

Xeon 2.8GHz processor to the human genome while using approximately 2.5GB of RAM. SRmapper is demonstrated to be able to perform both single-end and pair-end alignment and tolerates a higher number of discrepancies between reads and the reference sequence than BWA.

Using SRmapper as an alignment tool, a method to detect *Mycobacterium tuberculosis* (TB) in metagenomic samples containing many different bacteria is described. This method utilizes the construction of a novel uniqueness genome for TB containing only the regions of the TB genome not similar to any other bacterial species in the oral metagenome. Alignment of simulated and real metagenomic samples demonstrate the effectiveness of the uniqueness genome in the detection of TB and discover TB contamination in samples from the 1000 genomes project. Finally, the uniqueness genomes methodology is expanded to all genomes within the oral metagenome, and preliminary evidence is provided demonstrating that next generation sequencing can detect the presence of multiple simultaneously via alignment using SRmapper.

Dedication

To my wife, Elena; my daughter Anya; my brother, Peter, and my parents, Ken and Vivian: All my love and thanks.

Acknowledgements

I would first like to acknowledge the guidance, direction, work, and patience of my advisor, Dr. Chung Wong in guiding me through my time in the graduate program at the University of Missouri-St. Louis. I would like to thank Dr. Wong for both giving me the projects I worked on and for his ideas and direction as well as the numerous suggestions he gave to ensure that these projects moved in the correct direction. Furthermore, I would like to thank Dr. Wong for his willingness to give me the freedom to pursue my ideas, even when they proved to be incorrect, and his aid in developing the ones that had potential. Being allowed to struggle at times while having someone to fall back on when I could not solve problems on my own has been very beneficial in my development. Next, I would like to thank those serving on my dissertation committee: Dr. James Bashkin, Dr. Cynthia Dupureur, and Dr. Michael Nichols. To Drs. Dupureur and Nichols - thank you for suggestions, advice, and constructive critique during the defense of my dissertation proposal. To Dr. Bashkin, thank you for your suggestions and encouraging words during the oral presentation at my dissertation proposal. I would also like to acknowledge and thank my lab mates Elizabeth Hood and Andrew Lutes for their input and questions on my projects. I would also like to thank the numerous undergraduate and high school students I have had the pleasure and privilege of working with, advising, and supervising. I especially would like to thank Jennifer Berger for her valuable input and suggestions in the implementation and optimization of SRmapper. I would also like to thank Aaron Wilkerson for all the parallel work he did on strain specific features in TB. Specifically, I would also like to thank Barry Hykes, Kelsey Delph, and Peter Gontarz.

I would like to thank UMSL and the UM department of Chemistry and Biochemistry, the Graduate Dissertation Fellowship, the UM research board, and the UM research award for providing the funding necessary to support me during my graduate work. Without this support, none of this work could have happened. I would also like to express my deepest gratitude to the University of Missouri Bioinformatics Consortium for providing the computational resources necessary to make the project feasible.

Outside of UMSL, I would first and especially like to acknowledge my wife, Elena Vasilieva, and all the time she has spent, the encouragement she has given, and the love she has shown to me. I would also like to thank my daughter, Anya; and I hope that when you are old enough to be able to read this, you see this and know that just seeing you means the world to me. Dad loves you and always will. Peter, watching you may be my greatest motivation. Your work ethic is an inspiration to me, and I am sure that you will achieve great things. It is not possible for me write here all the ways you inspire me. Finally, to my parents Ken and Vivian: thank you so much for always pushing me and encouraging me and reminding me to "never waste the gifts God has given you." I do my best.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF PSEUDOCODES

Chapter I

An Introduction to DNA, DNA Sequencing, and the Potential Applications of DNA Sequencing

1.1 Deoxyribonucleic Acid, Its Properties, and Early Sequencing Means

The field of studies on DNA is possibly the fastest moving field in all of biochemistry. That deoxyribonucleic acid (DNA) functions as the sole genetic material or information molecule for all living cells was initially established over 70 years ago by Hershey and Chase by radiolabeling amino acids and nucleic acids in viruses and observing which was injected into cells in order to replicate the virus (Hershey & Chase, 1952). That DNA is the genetic material for all cells, whether prokaryotic or eukaryotic, has been confirmed since, and is accepted as common knowledge by both those inside and outside the scientific community. In the less than 75 years that the scientific community has understood the function of DNA at its most basic level, its critical importance has been demonstrated by the amount of work performed to understand the chemistry and biochemistry of its structure, means of replication, and use in providing the blueprint for the primary structure or sequence of every protein in every cell. A year after DNA was demonstrated to be the genetic material, Watson and Crick - building on information provided by Rosalind Franklin - determined DNA to be comprised of a double helix with the two helical strands running in opposite directions and being linked by hydrogen bonding of purines to pyrimidines (adenosine to thymine and guanine to cytosine) thereby establishing both an explanation for Chargaff's Rules and a potential means of DNA replication (Watson & Crick, 1953; Wilkens et al., 1953; Chargaff et al., 1952). By 1965, the concept of the triplet code for ribonucleic acid (RNA) - that in protein synthesis, three bases of DNA are transcribed into RNA and translated into a distinct amino acid dependent on the identities of the three bases - had been established with Nirenberg and his colleagues providing the RNA triplet code for the majority of

NUCLEOTIDE SEQUENCES OF RNA CODONS

UpUpU ***UpUpC***	Phe	***UpCpU*** ***UpCpC***	Ser	***UpGpU*** UpGpC	Cys	***UpApU*** ***UpApC***	Tyr
UpUpA ***UpUpG***	Leu	UpCpA ***UpCpG***	Ser	***UpGpA*** UpGpG	Nonsense* or Trypt	***UpApA*** ***UpApG***	Nonsense†
CpUpU ***CpUpC***	Leu or Nonsense*	***CpCpU*** ***CpCpC***	Pro	CpGpU ***CpGpC***	Arg	***CpApU*** ***CpApC***	His
CpUpA ***CpUpG***	Leu	***CpCpA*** CpCpG	Pro	***CpGpA*** CpGpG	Arg	***CpApA*** ***CpApG***	Glu-NH_2
ApUpU ***ApUpC***	Ileu	***ApCpU*** ***ApCpC***	Thr	***ApGpU*** ***ApGpC***	Ser	***ApApU*** ***ApApC***	Asp-NH_2
ApUpA ***ApUpG***	Met	***ApCpA*** ***ApCpG***	Thr	***ApGpA*** ApGpG	Arg. or Nonsense*	***ApApA*** ***ApApG***	Lys
GpUpU GpUpC	Val	***GpCpU*** GpCpC	Ala	***GpGpU*** GpGpC	Gly	***GpApU*** ***GpApC***	Asp
GpUpA GpUpG	Val	GpCpA GpCpG	Ala	GpGpA GpGpG	Gly	***GpApA*** GpApG	Glu

Table 1.1: The Triplet Code for RNA Translation to Protein Sequence. Of the 64 possible RNA triples, Nirenberg correctly identified the amino acid formed by 57 triples and deduced that UGA, UAA, and UAG could be termination codons (marked nonsense in the table). (Table adapted from Nirenberg et al., 1965).

RNA triplets (**Table 1.1**) (Nirenberg et al., 1965). This information lead to the establishment of the central dogma of molecular biology: DNA is transcribed into RNA which is translated into proteins, and proteins cannot be converted to DNA (Crick, 1970).

With the acceptance of the central dogma came the somewhat naïve belief that since DNA coded for every protein in an organism and protein function defined organisms, simply knowing the DNA sequence, or genome, of an organism could be used to understand every aspect of the organism. Although it has been determined that the composition of an organism is far more complicated than a simple conversion of the genes in an organism to the proteins encoded by them with factors such as DNA methylation and RNA interference (RNAi), for example, each inhibiting part of the process by which DNA is used to eventually synthesize proteins (Robertson & Jones, 2000; Fire et al., 1998). However, knowledge of the DNA sequence of an organism can provide a tremendous amount of information about that organism and does define much of the form and function of that organism. The first method to reliably sequence DNA was discovered in the early 1970s and used DNA repair to add radiolabeled nucleotides to the 3' end of DNA single strands in bacteriophage λ (Wu, 1970). This methodology was modified and improved by Fredrick Sanger in 1977 to quickly and accurately sequence DNA by the use of radiolabeled primers and dideoxynucleotides to inhibit chain elongation in a DNA strand complementary to the template strand (Sanger et al., 1977). In the same year, Maxam and Gilbert also devised a method to sequence DNA by base-specific cleavage with dimethyl sulfate being used to methylate guanine and adenine and hydrazine being used to cleave cytosine and thymine (Maxam & Gilbert, 1977). Both Sanger sequencing and Maxam-Gilbert sequencing determined DNA sequence by

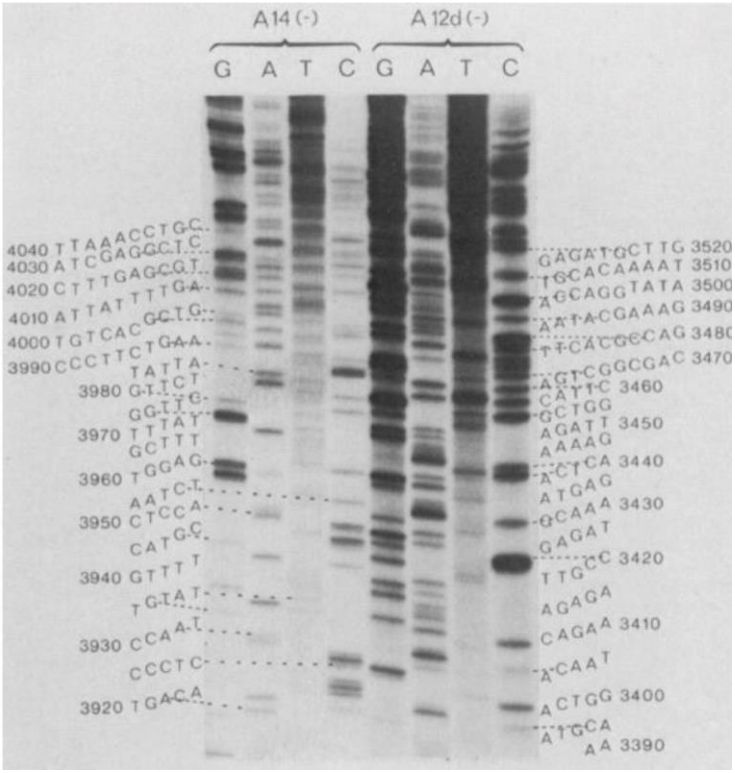


Figure 1.1: Sanger Sequencing By Chain Terminating Inhibitors Using Gel Electrophoresis to Separate Fragments. In Sanger sequencing, a radiolabeled primer was elongated using the DNA polymerase reaction and the four standard nucleotides in four different lanes. In each lane, an additional chain terminating version of one of the nucleotides containing a dideoxyribose was added in a low concentration allowing for random termination of the sequence at various positions where the naturally occurring version of the dideoxyribonucleic acid would normally be incorporated. The fragments of different length were resolved by gel electrophoresis. The DNA sequence could then be read bottom to top. (Figure adapted from Sanger et al., 1977).

separating DNA fragments using gel electrophoresis (**Figure 1.1**). Variants of both methods are still commonly used today; however, due to the more tedious analysis required for Maxam-Gilbert sequencing, Sanger sequencing and its subsequent improvements, such as fluorescently labeled nucleotides and the various forms of sequencing by synthesis, have become the dominant method for sequencing DNA.

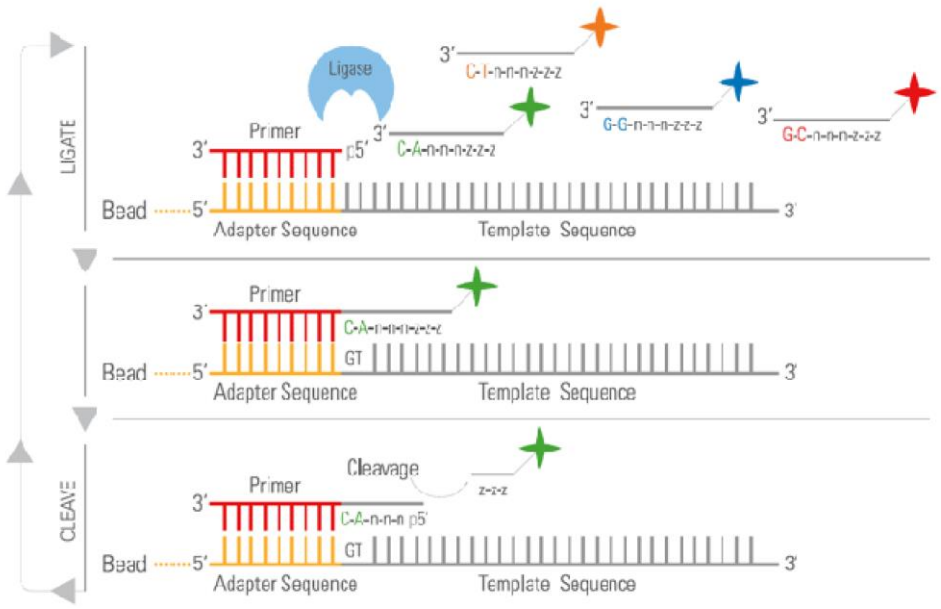
1.2 Next Generation Sequencing

1.2.1 Next Generation Sequencing Platforms and Methodology

In the less than forty years that fast and reliable methods for DNA sequencing have existed, there has been a continuous growth in the rate of DNA sequencing and a continual decrease in the cost of DNA sequencing. Early advances to the field of DNA sequencing included a shift from radiolabeled DNA fragments being sequenced on gel slabs via electrophoresis to fluorescently labeled nucleotides being used to perform sequencing in capillary tubes (Woolley and Mathies, 1995). Next generation sequencing (NGS) has moved away from sequencing by chain termination and into methods such as sequencing by synthesis for companies such as Pacific Biosciences, 454 Life Sciences, Roche, Illumina, Solexa, and Life Technologies and sequencing by ligation for Applied Biosystems (Pareek et al., 2011; Liu et al., 2012; Quail et al., 2012). Specifically, in the case Life Technologies and their Ion Torrent instruments, sequencing is performed in microwells, and addition of a nucleotide to a growing strand of DNA is detected by a change in pH due to the release of hydrogen ions in the DNA polymerization reaction (Rothenberg et al., 2011). A drawback to this method is that for a homopolymer sequence - a sequence with the same nucleotide repeating multiple times - correctly converting the change in pH or electrical charge to the correct number of nucleotides added in the

homopolymer sequence can be difficult (Rothenberg et al., 2011). In the case of Life Sciences and 454 pyrosequencing, sequencing by synthesis is utilized with DNA polymerization detected by the release of pyrophosphate (Margulies et al., 2005). This pyrophosphate is converted into ATP by ATP sulfurylase, with the synthesis of ATP being detected by firefly luciferase (Ronaghi et al., 1996). Solexa, which was purchased by Illumina still utilizes fluorescent dyes in its sequencing. All variants of this dye-based method utilize four different fluorescent dyes to identify the identity of an incorporated nucleotide and reversibly chain terminating nucleotides where the chain terminating functional group can be cleaved after the identity of the incorporated nucleotide is determined (Guo et al., 2008). Pacific Biosciences instrumentation applies a similar approach but, instead of creating clusters of identical sequences like Illumina, identifies the addition of a single nucleotide to a single strand of DNA using a single DNA polymerase enzyme (Eid et al., 2009). Pacific Biosciences achieves this feat by fixing the DNA polymerase to the bottom of a sequencing well with a volume in the range of 10^{-21} liters and utilizing a detector that only measures fluorescence at the bottom of the well (Levene et al., 2003). In contrast to the sequencing by synthesis methods employed by the above listed platforms, Applied Biosystems utilizes a method termed SOLiD (Sequencing by Oligonucleotide Ligation and Detection). In the SOLiD method, after a primer is hybridized to the template sequence, an eight nucleotide fragment in which the first two bases are complementary to the template strand, the next three bases are able to bind any sequence, and the final three bases are fluorescently labeled hybridizes to the template sequence and is ligated to the primer (McKernan et al., 2009). Upon cleavage of the last three bases, the resulting 5 nucleotide fragment is available for ligation to another

A



B

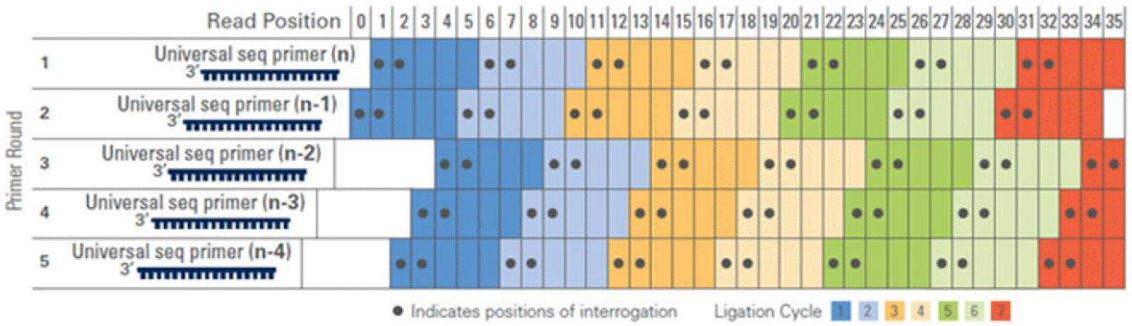


Figure 1.2: SOLiD Sequencing Technology. (A) In sequencing by ligation, a primer hybridizes to the adapter sequence on the template sequence. Next, eight nucleotide oligomers with various natural nucleotides at positions 1 and 2 compete to hybridize the template sequence with the oligomer with a complementary sequence to the template at its 1 and 2 positions successfully hybridizing. The 8-mer is ligated to the primer followed by cleavage of the last 3 nucleotides and the fluorescent dye leaving a 5-mer that is available for ligation. (B) The process from A is repeated a total of 7 times for each

primer. The black dots represent positions where the sequence is determined for each primer. After the final round of ligation, the hybrid sequence is washed off, and a new primer is hybridized to the adapter sequence of the template. This primer is offset from the first so that different bases in the sequence can be determined. A total of five primers each with 7 rounds of ligation are analyzed. This process results in each base being sequenced twice thereby reducing the number of errors in base calling. For example, read position 17 is determined by both the ligation involving primer 1 and primer 5. (Figure adapted from www.appliedbiosystems.com/.)

8 nucleotide fragment. This process of hybridization and ligation is repeated for 7 to 10 cycles. Five rounds of priming are performed in total with the primer being offset by one nucleotide in each round of priming. This results in the identity of each base being determined twice and reduces the number of incorrectly sequenced nucleotides.

1.2.2 Comparison of Next Generation Sequencing Instrument Output

Although most NGS techniques produce sequences, also called reads, shorter than those produced by Sanger or Maxam-Gilbert sequencing, they also produce a much higher quantity of sequenced DNA at a much lower cost due to massive parallelization and small reaction vessels. Whereas Sanger and Maxam-Gilbert sequencing only determines the identity of the last nucleotide in each oligonucleotide formed, techniques such as sequencing by synthesis and sequencing by ligation further reduce the cost of sequencing by determining the identity of many bases in each fragment. Each sequencing method has advantages and disadvantages depending on which of five variables in sequencing is considered. These five variables are read length, nucleotides sequenced, time, cost, and sequencing accuracy (Pareek et al., 2011; Liu et al., 2012; Quail et al., 2012). From **Table 1.2** and **Table 1.3** it can be determined that all of these factors vary depending on the goal of the sequencing operation. Instruments are typically split into two categories, one for large scale sequencing projects and one for smaller scale sequencing projects. Illumina instruments tend to have the lowest sequencing costs and highest outputs in their sequencing classes while retaining a moderate read length and accuracy. However, they also tend to have the longest times for sequencing in their class. Ion Torrent instruments tend to have fast sequencing times and moderate read lengths and costs but a somewhat limited output and higher error rate. 454 pyrosequencing produces

Sequencer	454 GS FLX	HiSeq 2000	SOLiDv4	Sanger 3730xl
Sequencing mechanism	Pyrosequencing	Sequencing by synthesis	Ligation and two-base coding	Dideoxy chain termination
Read length	700 bp	50SE, 50PE, 101PE	50 + 35 bp or 50 + 50 bp	400~900 bp
Accuracy	99.9%*	98%, (100PE)	99.94% *raw data	99.999%
Reads	1 M	3 G	1200~1400 M	—
Output data/run	0.7 Gb	600 Gb	120 Gb	1.9~84 Kb
Time/run	24 Hours	3~10 Days	7 Days for SE 14 Days for PE	20 Mins~3 Hours
Cost/million bases	$10	$0.07	$0.13	$2400

Table 1.2: Comparison of the Poperties of Large-Scale Sequencing Instruments. The read length, accuracy, output, sequencing time, and cost are compared for 454 GS FLX, Illumina HiSeq2000, and Applied Biosystems SOLiDv4. (Table adapted from Liu et al., 2012).

Platform	Illumina MiSeq	Ion Torrent PGM	PacBio RS	Illumina GAIIx	Illumina HiSeq 2000
Instrument Cost*	$128 K	$80 K**	$695 K	$256 K	$654 K
Sequence yield per run	1.5-2Gb	20-50 Mb on 314 chip, 100-200 Mb on 316 chip, 1Gb on 318 chip	100 Mb	30Gb	600Gb
Sequencing cost per Gb*	$502	$1000 (318 chip)	$2000	$148	$41
Run Time	27 hours***	2 hours	2 hours	10 days	11 days
Reported Accuracy	Mostly > Q30	Mostly Q20	<Q10	Mostly > Q30	Mostly > Q30
Observed Raw Error Rate	0.80 %	1.71 %	12.86 %	0.76 %	0.26 %
Read length	up to 150 bases	~200 bases	Average 1500 bases****	up to 150 bases	up to 150 bases

Table 1.3: Comparison Of the Ion Torrent PGM and Pacific Bio RS to Various Illumina Instruments. The five sequencing characteristics as compared in Table 1.2 are compared for Ion Torrent, Pacific Biosciences, and Illumia. (Table adapted from Quail et al., 2012).

longer reads with a high accuracy in a reasonably short amount of time but also has a limited output and a very high sequencing cost for NGS. ABI SOLiD sequencing produces extremely accurately sequenced reads at a low cost and high output. However, it also produces the shortest reads among the NGS instruments and has some of the longest sequencing times. Finally, Pacific Biosciences instruments produce reads over 10,000 base pairs long at a moderate cost but have error rates approximately 10x higher than Illumina or Ion Torrent incorrectly determining the identity of over 10% of nucleotides sequenced (Data not shown).

1.3 Analysis of NGS Data

Before the advent of NGS, the challenges of analyzing sequencing data and the usefulness of computers were already being discussed (Staden, 1979). With the advent of NGS, the challenges associated with analyzing and interpreting sequencing data took as big a step forward as sequencing had (Ng & Kirkness, 2010). The amount of publically available DNA sequencing data has grown exponentially over the last six years. As of May, 2015, there were over two petabases (Pb) of publically available sequencing data from next generation sequencing instruments available on the joint sequencing database of the National Center for Biotechnology Information (NCBI), the European Bioinformatics Institute, and the DNA databse of Japan (Agarwala et al., 2015). These two quadrillion bases of publically available, raw sequencing data are housed in the sequence read archive (SRA) (**Fig. 1.3**) (http://www.ncbi.nlm.nih.gov/sra) and contain the results of whole genome shotgun sequencing, transcriptome sequencing, 16s RNA gene sequencing, and other types of experiments. Additionally, DNA sequencing instruments continue to increase in speed, accuracy, and read length while decreasing in

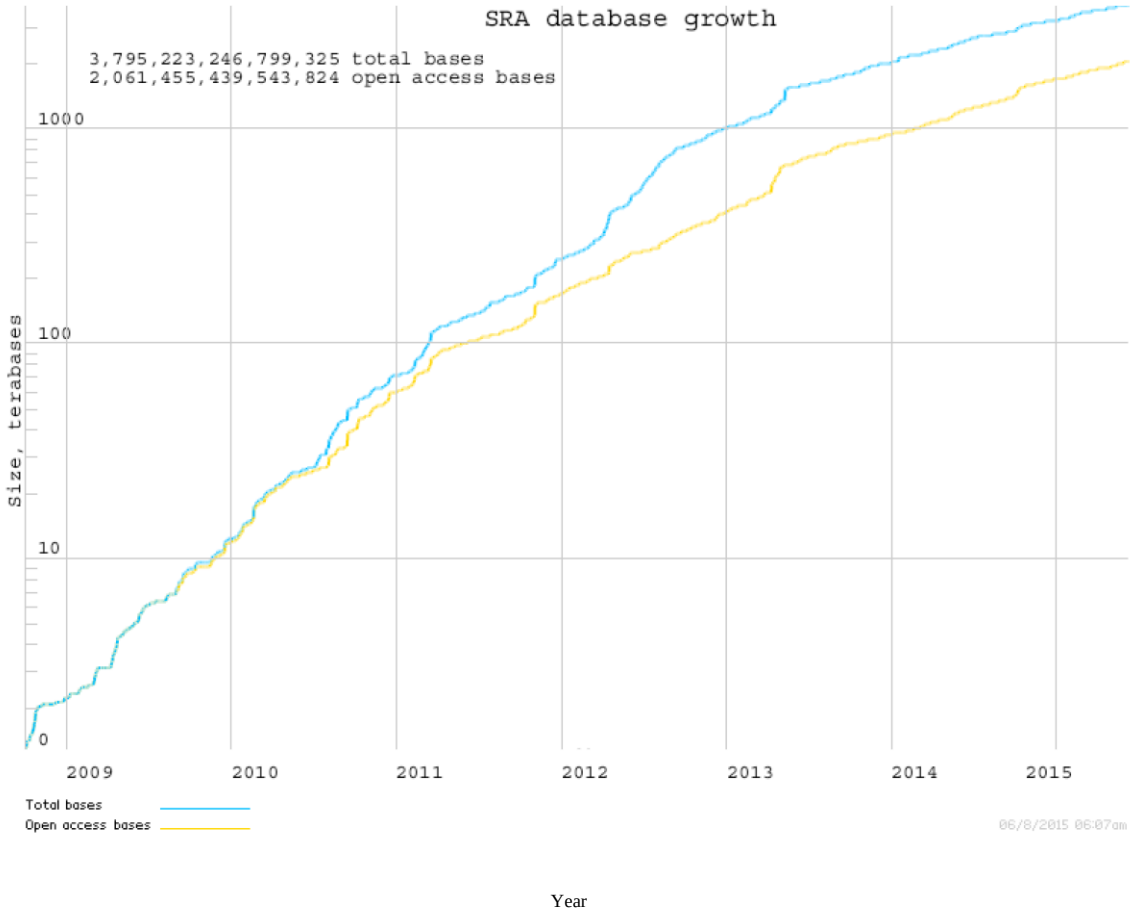


Figure 1.3: SRA Database Size. The SRA database has grown exponentially since its creation. Open access bases are publically available for download. There are an additional 1.7 quadrillion bases of sequenced DNA requiring special permissions to access due to its sensitive nature. Image adapted from http://www.ncbi.nlm.nih.gov/Traces/sra/

sequencing cost. DNA sequencing instruments such as Illumina's HiSeq2000 and Genome Analyzer II, ABI's SOLiD 4, and Roche's GS FLX titanium can generate gigabases (Gb) per day. In the case of the HiSeq2000, a single experiment can produce up to 600 Gb of sequencing data in approximately eleven days. The cost and time of sequencing a human genome has dropped from the often quoted three billion dollar or dollar per base, figure of the human genome project, which took ten years, to well below $10,000 with a $1000 sequenced human genome expected before the end of the decade (Lander et al., 2001) (www.genome.gov/sequencingcosts/).

The ability to analyze the sequencing data created by next-generation sequencing instruments is complicated by many factors including the random locations of sequences gathered, the massive amounts of data generated, the relatively short sequence lengths of the reads that are produced by next-generation sequencing instruments, and occasional errors made by sequencing instruments (Ng & Kirkness, 2010). In contrast to the BAC to BAC method employed in the human genome project, whole genome shotgun sequencing samples from across an entire chromosome or chromosomes simultaneously (Lander et al., 2001). Using the Illumina HiSeq2000 as an example, reads produced by NGS instruments range from as short as 35 base pairs (bp) long up to 150 bp long (Liu et al., 2012). Additionally, errors in base calling, the determination of the identity of a base, in a sequence can be as high as 1.5% for Illumina instruments although the percent error rate in base calling has decreased with advances in technology. Performing a sequencing experiment designed for a 30x coverage of the human genome would result in 90 Gb of DNA being sequenced. Assuming a read length of 100 bp, slightly under one billion reads would be formed in the sequencing of a human genome with no information in the

raw sequencing data about what portion of the genome any of those reads originated from. The task of assembling these reads into the complete genomic sequence of an individual can be likened to the assembly of a one billion piece puzzle where the edges on some of the pieces have to be smoothed down (the equivalent of sequencing errors) so the pieces will fit together correctly.

With the obvious challenge posed by assembling the raw data generated by NGS experiments into a complete genome, dozens of algorithms have been created with the goals of quickly, efficiently, and accurately assembling raw NGS data (Ng & Kirkness, 2010; Miller et al., 2010; Ruffalo et al., 2011). A few of these algorithms attempt assembly of these sequences *de novo* – a process that describes assembly of a genome from reads using no outside information. *De novo* assembly is complicated by repetitive elements in genomes such as that of humans (Miller et al., 2010). Assembly of repetitive elements is difficult if the length of the repetitive element is longer than that of the sequences generated by sequencing instruments since the precise location of the sequence within the repeat region is unknown as is the total length of the repeat. The use of pair-end sequencing, where both ends of a longer DNA fragment with known linker length are sequenced has made the spanning of certain shorter repeat elements possible; however, *de novo* assembly remains extremely challenging, memory-intensive, and vastly slower when compared to the alternative that will be discussed subsequently. Due to these drawbacks, *de novo* sequencing is not often used for the sequencing of genomes larger than those of bacteria. Several popular algorithms for *de novo* assembly include Velvet (Zerbino and Birney, 2008), ABySS (Simpson et al., 2009), ALLPATHS-LG (Gnerre et

al., 2011), Edena (Hernandez et al., 2014), Fermi (Li, 2012), and SPAdes (Bankevich et al., 2012).

In contrast to *de novo* assembly, reference based assembly - also called alignment or mapping - is a relatively fast, simple, and less memory intensive processes. Reference based assembly constructs genomic sequences by mapping - also called aligning - reads to a pre-existing reference sequence. For alignment to be successful in constructing the genomic sequence of a virus, bacteria, gene, or genome, the reference being used must be similar to the one being sequenced. Thus, reference based assembly is most feasible for sequencing the genome of a species whose genome has previously been sequenced or for sequencing different strains of a bacterial genome. As the number of organisms whose genomes have been sequenced has increased, reference based assembly has become increasingly useful. Because of this, dozens of alignment algorithms employing various strategies have been devised and implemented. These tools fall into three main categories: read-hashing algorithms, reference-hashing algorithms, and Burrows-Wheeler transform (BWT) algorithms (Burrows & Wheeler, 1994; Ferragina & Manzini, 2000). Read-hashing algorithms are less popular than the other algorithm types and include MAQ (Li et al., 2008), mrFast (Alkan et al., 2009), mrsFast (Hach et al., 2010), SHRiMP (Rumble et al., 2009), and ZOOM (Lin et al., 2008). Between BWT algorithms and genome-hashing algorithms, BWT algorithms have been more favored recently. Among BWT methods are bowtie (Langmead et al., 2009), bowtie2 (Langmead and Salzberg, 2012), BWA (Li & Durbin, 2009; Li & Durbin, 2010), segemehl (Hoffman et al., 2009), and SOAP2 (Li et al., 2009). Genome-hashing algorithms include BFAST (Homer et al, 2009), MOM (Eaves & Gao, 2009), MOSAIK (Lee et al., 2014), PASS (Campagna et al.,

2009), ProbeMatch (Kim et al., 2009), SHRiMP-2 (David et al., 2011), SOAP (Li et al., 2008), STAMPY (Lunter & Goodson, 2011), WHAM (Li et al., 2011), and our algorithm, SRmapper (Gontarz et al., 2013).

BWT algorithms have been more favored recently due to their general superior performance to genome-hashing algorithms. This is due to the fact that in general, BWT methods require less memory usage and perform the task of alignment more quickly than genome-hashing methods while retaining similar sensitivities. For example, among the above listed algorithms, the only ones that can align to the human genome using a computer with 4GB of RAM are BFAST and our algorithm, SRmapper. Among the BWT-based methods, bowtie and BWA both can be run on a computer with 4GB of RAM and are both approximately an order of magnitude faster than BFAST. This makes the above listed tools valuable to those requiring an algorithm that can be used on small memory computers. Even for those with access to large memory machines, the efficient use of memory is desirable if performance is not affected. In this regard, BWT methods again traditionally held the advantage. Before the introduction of SRmapper, BFAST was among the fastest hashing algorithms and was still an order of magnitude slower than BWA and bowtie. Although the sensitivity of BFAST is somewhat higher than that of BWA, the large discrepancy in speed between the two algorithms has made BWA a more popular choice. Between BWA and bowtie, BWA is much more flexible in both the number and types of discrepancies between reads and the reference allowed. Bowtie does not support insertion or deletion detection while BWA does, and bowtie only searches for up to 3 mismatches between reads and the reference genome.

1.4 Future Applications of DNA Sequencing Detection and Diagnostics of TB and Other Bacteria

With the decrease in cost of DNA sequencing and the increase in sequencing speed and output, the possibility of NGS eventually being used a diagnostic tool for bacterial infection is very likely. The hypothetical possibility of using NGS has already been suggested and some of the ethical issues discussed (Voelkerding et al., 2009; Dunne et al., 2012; Desai & Jere, 2012; Biesecker et al., 2012). Currently, however, NGS is not routinely used as a diagnostic tool for bacterial infection but rather is more commonly used to provide diagnostic information in outbreaks of diseases (Sherry et al., 2013; Whitney et al., 2014; Octavia et al., 2015). Alternatively, NGS and analysis of NGS data has been used to develop primers for polymerase chain reaction (PCR) based or real time PCR (RT-PCR) based diagnostics (Fournier et al., 2014). Depending on the circumstances, RT-PCR based diagnostics can rapidly detect bacteria with specificity ranging from genus level diagnostics to strain level diagnostics (Marshall, 2004; Hung et al., 2012). Although RT-PCR has proved effective in diagnostics, questions remain about the feasibility of using NGS especially in an environment without specialists to perform sequencing and analysis as well as the time required to perform analysis. For a recent outbreak of *E. coli*, Sherry et al. noted that even with the rapid sequencing time via Ion Torrent, there was still a five day turnaround time after a positive culture was established (Sherry et al., 2013). For most bacterial infections, a diagnostic tool with a five day turnaround time and the requirement of specialists to perform analysis is not practical for a hospital or clinical setting. For NGS to be used as a diagnostic tool, the analysis required would have to be performed in a much faster mannerism and either be

automated or simple enough that a nonspecialist could perform the analysis. Additionally, the requirement of a positive culture, isolation, or enrichment for a bacterium before sequencing, as suggested necessary by Köser et al., lessens the impact of NGS as a diagnostic tool, although the additional insights that possibly could be provided by NGS, such as drug resistance patterns, could offset the positive culture drawback (Köser et al., 2012).

One bacterial species that could potentially lend itself well to diagnosis by NGS is *Mycobacterium tuberculosis* (TB). TB is a slow-growing, Gram-positive bacteria that causes over 1 million deaths annually (Gey et al., 2001) (www.who.int/tb/en/ and www.cdc.gov/tb/). The World Health Organization (WHO) estimates that 1.5 million people died of TB in 2013 including over 500,000 with HIV and 80,000 children. Compounding the difficulty in treating TB is the fact that due to the long treatment times required to eliminate TB, treatment compliance is poor, and this has led to very high rates of drug resistance in TB. In 2013 alone, there were nearly half a million cases of multi drug resistant TB (MDR-TB) with over 200,000 people dying from MDR-TB. Multidrug resistant TB is defined as TB strains that cannot be treated by the two most common TB drugs, Rifampin and Isoniazid. The nearly 50% mortality rate in MDR-TB cases is significantly higher than the overall mortality rate for TB. In addition to MDR-TB, there is a growing number of extensively drug resistant TB strains (Ford et al., 2012).

Although the dangerous nature of TB makes it an important disease to be able to reliably diagnose, it is the slow-growing and highly drug resistant properties of TB that make it attractive as a target for detection or diagnosis via NGS. Although RT-PCR methods exist for providing a preliminary diagnosis of TB and can even detect some drug

resistance patterns, the WHO notes that culture-based methods are still the most reliable method to positively diagnose TB infection (Drobniewski et al., 2012). Although known mutations in the TB genome resultant in drug resistance are recorded at the TB database (www.tbdb.org), drug resistance diagnostics via RT-PCR are somewhat limited due to an incomplete knowledge of the genetic origins of all drug-resistance patterns (Galagan et al., 2010). Confirmation of TB infection by culture-based methods requires between 2 and 12 weeks to establish a positive diagnosis and drug resistance patterns. Thus, NGS could theoretically provide additional information that cannot be obtained by RT-PCR by comparing TB in a new infection to strains with known drug resistance patterns even if the genomic origin of resistance is unknown in a much shorter time than is required for a diagnosis via culturing. Due to the slow-developing nature of TB, isolation or enrichment, although not ideal, may be permissible in using NGS as a detection agent or diagnostic tool.

Perhaps the Holy Grail of bacterial diagnostics, a single test with the ability to quickly, accurately, and inexpensively test for the presence of any and every infectious bacterial species would revolutionize the field of bacterial diagnostics. In theory, DNA sequencing provides the possibility of such a test. Since every bacterial species and subspecies has a distinct genome, sequencing the genome of an infectious agent has the ability to resolve its species and strain from all other bacterial species in the metagenome of that species - the total genomic environment of a microbiome including the DNA of the species of interest, the DNA of all other species in the same microbiome, and the DNA of the host organism if applicable. However, NGS does not directly sequence an entire genome but rather sequences small fragments of a genome which then require

assembly. With the short length of DNA sequences generated by NGS, it is sometimes impossible to determine the genomic origin of certain reads due to similarities between bacterial species. This is further compounded by the necessity to allow for the possibility of any combination of species within a metagenome. For a fast and inexpensive single test meant to identify every bacterial species, enrichment of a sample for a particular species would be prohibited due to failure to meet the goals for the test. Thus, for there to be any possibility of using NGS as a detection agent for any bacterial species or strain, a method to reliably determine the genomic origins of short sequences from a list of hundreds or thousands of bacteria would be required. Simply finding alignments to a species within the metagenome by mapping would not be sufficient to detect the presence of that bacteria due to the aforementioned reasons, and *de novo* construction of genomes would be severely limited due to the possibly large number of originating species creating reads and assumedly low coverages of each species in a sample containing many species.

1.5 Thesis Scope and Overview

In this thesis, the implementation and deployment of a new genome-hashing alignment algorithm, SRmapper, is described. This algorithm utilizes a probabilistic model to determine the number of discrepancies permissible between reads from NGS instruments and a reference genome. The alignment speed, sensitivity, and accuracy of this algorithm is measured and compared to one of the most popular alignment algorithms, BWA, using a combination of real and simulated NGS sequencing data. This algorithm is then applied in the formation of a technique by which TB can be detected in oral metagenomic samples using NGS by the construction of a uniqueness genome - a

genome consisting of only the portions of the TB genome that are not found to be similar to portions of any other species within the oral metagenome. The usefulness of the uniqueness genome in increasing the sensitivity and selectivity of NGS as a detection agent for TB is demonstrated by the analysis of both simulated oral metagenomic samples and real metagenomic samples. Finally, preliminary studies demonstrating the expansion of the uniqueness genome methodology to all genomes within the oral metagenome are reported. These studies suggest that uniqueness genomes can be built for all species in the oral metagenome and that these uniqueness genomes can be used to detect bacterial species in the oral metagenome even when only a small portion of the bacterial genome is sequenced.

Chapter II

SRmapper: A Fast and Sensitive Genome-Hashing Alignment Algorithm for NGS Analysis

2.1 Background

SRmapper is a genome-hashing alignment algorithm designed with the goals of demonstrating that genome-hashing methods can outperform BWT algorithms in terms of speed and memory usage while retaining sensitivity comparable to other popular alignment algorithms. Additionally, another goal of developing an in house algorithm was to have software that was easily manipulatable for extensions out of theoretical work and into application-based work. SRmapper operates by performing an indexing of the reference genome one time and writing the index to file. In subsequent alignment to the reference genome, reads are anchored to the reference using the index created earlier. Extension of the alignment then occurs in a base by base manner checking for mismatches. A limit for discrepancies between the reference genome and each aligned read is calculated using a probabilistic model based on the reference genome length, the read length, and the desired quality of alignment to the reference genome. The conceptualization of SRmapper began in March of 2011. The first working skeleton versions of the code were implemented in June of 2011 as a hash table only alignment tool. phred scoring and SNP detection were incorporated by July, 2011. As of August, 2011, indexing had been made able to store the entire human genome in systems with 4GB of memory. By the end of September, 2011 SRmapper could align against the full human genome using 2.5GB of RAM. The first publically available version of SRmapper was released in August, 2012 along with a suite of software for testing its performance in comparison to BWA. The main components of SRmapper's source code are the indexing algorithm named "buildindex.cpp" and the alignment algorithm named "align.cpp". These files and others used for testing SRmapper are available at

http://umsl.edu/~wongch/software.html along with updated versions of the SRmapper software suite.

2.2 Construction of Reference Sequence Indexes by SRmappers Buildindex Algorithm

SRmapper requires input files for generation of indexes be given in a file or multiple files of fasta (.fa) or multifasta (.mfa) format. SRmapper's buildindex program builds an index from the reference genome or sequences being used as a reference to which reads are aligned. This index is in the form of a hash table whose characteristics and creation will be subsequently discussed in detail. SRmapper's buildindex algorithm additionally builds a compressed 2-bit per base reference sequence. The full pseudocode for buildindex is provided at the end of this chapter as **SRmapper Buildindex Pseudocode**.

2.2.1 SRmapper Buildindex Input Format and Files

SRmapper's buildindex algorithm indexes one or more fasta or multifasta files as a reference sequence **(Fig 2.1)**. Buildindex is invoked by running the command:

buildindex { <Ref1.fa> <Ref2.fa> … <RefN.fa> } <index.sqn> [options]

where Ref1.fa through RefN.fa are the reference fasta or multifasta files and index.sqn is the name given to the index being built. The option -N can be appended to the end of the command to treat nonstandard nucleotides as random bases which will be discussed later. By default, this option is turned off. SRmapper requires files be of fasta format and attempts to validate correct formats before attempting to create an index. Fasta files have the format of having one or more header lines, which usually give information on source of the sequence, starting with either a '>' or a ';' character followed by one or more lines

A

```
>Chromosome1
AAAAAAAACGTCGACAGTACGTACGTGACGAGACGTA
ACTACTGGCATTCAGCTAGTAGCTAAAACAAACCATG
CCCCCCCNNNNNNGATCACATCTGATACGTAACCCCG
```

B

```
>Chr1
ACGTAGCGATGAGTATATATCAGTATACGATCGATTT
CATTCATCTACTTTTAAGCCAGTGCAGTGTGGGCATT
CCCCAGACCGTTTTTAGCAGACAGTACCACATAACTA
>Seq1
TACTACTACTACTACTAACTATTTGGAGACGCGCCCAG
TCTCCATGCATTCAG
>Genome3
CATTGGGTACCTGATCATTCGATGCGGACGACGTAGT
ACACCAAAGGTGGTGCATTCAGGTACCATATGGCGAA
GGTAGATACAGCTAGCTGGATCG
```

Figure 2.1: Example References in fasta and multifasta file formats. (**A**) Single fasta (.fa) format. Single fasta files have one or more header lines which always start with a '>' and provide information about the reference sequence followed by one or more lines of sequence. (**B**) Multifasta (.mfa) format. Multiple fasta files are comprised of two or more fasta sequences concatenated to each other. Each sequence always contains one or more header lines which start with a '>' followed by one or more lines of sequence. There are no rules imposed on which fasta sequences can be merged to form a .mfa file. Thus, sequences may come from different chromosomes, different contigs from the same genome, or different genomes or references.

of sequence containing only single letter nucleotide codes. Multifasta files have the same format as fasta files except that after the last line of sequence from the first reference, multiple additional references can be included using the same format for fasta files.

2.2.2 Hashing Terminology

A hash function is a function that maps pieces of data of variable size to data of fixed size. The input data are called hash keys, and the output data are called hashes or hash values. Usually, in addition to creating data of fixed size, the hashing function has the goal of storing data in a format that allows for quick lookup of the data by storing the data in a hash table. **Fig. 2.2** demonstrates an example of hashing and the creation of a hash table. The data being looked up is stored in buckets that are located by their hash values. For a hash table to be most efficient in terms of looking up values in buckets, the hash table must be balanced. This means that there are no empty buckets or overflow. Overflow occurs when there are more entries than can be stored in one bucket. Multiple entries in one bucket require more checking to determine which anchoring position is correct. An effective hash table must also be comprehensive. This means that every possible key must have a corresponding hash value and that every location in the genome must be stored in the hash table. Finally, a hash table ideally would be minimal meaning that it uses as little memory as possible while still meeting the previous criteria of balance and comprehensiveness.

2.2.3 Prehashing Routine to Determine Key Length

SRmapper's indexing algorithm first determines the length of the reference genome or reference sequences by scanning through and counting the number of nucleotides in the reference. Next, the indexing algorithm determines the number of

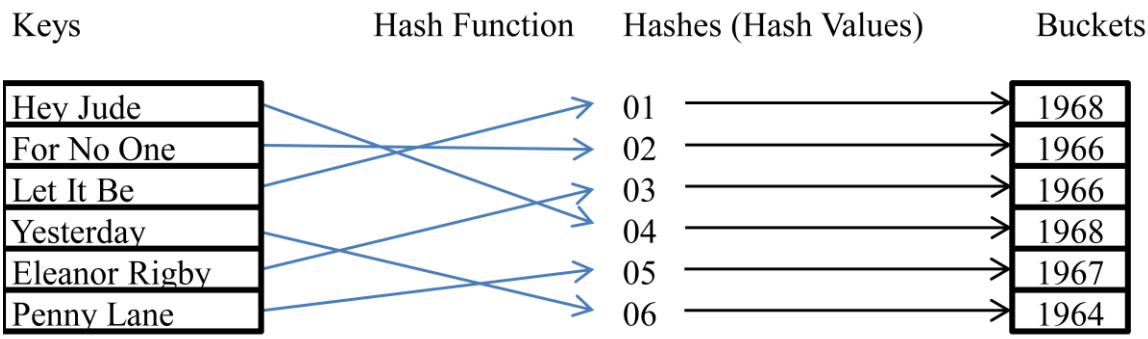


Figure 2.2: Hashing Terminology. Hashing is a process by which data is converted from variable size to a fixed size via a hash function. In this example, several popular songs by "The Beatles" are hashed. The original song titles serve as keys. The hash function converts these keys into hashes or hash values. One common application of hashing is the hash table - a data structure that can serve as a means to quickly look up information. In a hash table, the hashes point to buckets which store some piece or pieces of information related to the key as entries in a bucket. In this example, the release year of each song is stored as an entry in the bucket for each hash. To determine the year of release for a given song, a computer would hash the song title using the hash function. It would then lookup the release year by looking in the bucket that is pointed to by the hash value. In the case of a list containing many songs, finding the year by using a hash table is faster than scanning through the songs until the correct one is found.

bases, D, that will be used in the index to form each word or key. For a reference sequence R nucleotides long, D is determined by:

$$D(R) = floor(\log_4 R) \quad \text{Eq 1}$$

where floor denotes the rounding down of base-4 logarithmic value of the reference length, R. The value of D is chosen with the goal of creating an index that has few buckets that have multiple entries while still being as memory efficient as possible. Since there are four different nucleotides, there are 4^D possible different keys of length D. By choosing D using equation 1, $4^D \approx R$, and if each position in a reference of R is used as the start of a word, there are approximately R entries in the buckets since R is much greater than D. Since there are both R entries in buckets and R discrete keys, each bucket has on average one entry thereby forming a hash table that achieves maximum balance while being minimal. In the case of the human genome with reference length $\sim 3 \times 10^9$ nucleotides, $D = 15$. For a viral genome of length 10kb, $D = 6$.

2.2.4 SRmapper Hashing Function

In the case of SRmapper, the hash function takes as input some sequence of D bases and converts them into a base-4 number by assigning the nucleotides A, C, G, and T values of 0, 1, 2, and 3 respectively (**Fig. 2.3 A)**. Non-Standard nucleotides or undetermined nucleotides, N, in a fasta sequence can either be treated as a random nucleotide value or can cause the hashing function to exclude a key if it contains one or more N nucleotides. The hash function next converts the generated base-4 number into its equivalent base-10 value with a fixed size of 4 bytes. Originally this was performed by calculating the value of each base-4 number. For example, a sequence of TCC was assigned a base-4 value of 311 and the base-10 value was determined by calculating

A

Position 6543210 → 6543210
Sequence TCAGTTC → 3102331

B

$(3102331)_4$

	A-0	C-1	G-2	T-3
0	0	**1**	2	3
1	0	4	8	**12**
2	0	16	32	**48**
3	0	64	**128**	192
4	**0**	256	512	768
5	0	**1024**	2048	3072
6	0	4096	8192	**12288**

$(13501)_{10}$

Figure 2.3: The hashing function of SRmapper. (**A**) The hashing function employed by SRmapper first takes D bases and converts them into a base-4 number of length D by converting A to 0, C to1, G to 2, and T to 3. The thin and thick underscores highlight how the 'C' in position 5 and the 'T' in position 2 are converted into their base-4 equivalents while retaining their position. (**B**) The base-4 value generated in panel 'A' is converted into a base-10 value by utilizing a preformed table to reduce the number of calculations

necessary to perform the conversion. The left column denotes the position number while the top row denotes the base and its base-4 identity. The values in the grid are the base-10 values for every possible base-4 digit for D=7. The bold entries in the table trace the conversion process of each base-4 digit into its equivalent base-10 value. The thin and thick underscored numbers highlight the conversion of the ‘1’ in the 5-position and the ‘3’ in the 2-position. The base-10 conversion is the sum of the bolded numbers.

$3 \times 4^2 + 1 \times 4^1 + 1 \times 4^0$. Performing this operation hundreds of millions of times for a 14 digit number in the case of the human genome proved to be sufficiently inefficient to warrant a faster method. The original method of fully calculating the base-4 to base-10 conversion was replaced by first calculating every possible conversion for $a \times 4^b$ for $0 \leq a \leq 3 \; and \; 0 \leq b \leq D$ (**Fig 2.3 B**). This proved to increase the speed of the indexing algorithm by an order of magnitude and was also employed in the alignment algorithm which will be discussed later. The location of where in the reference sequence or genome the sequence of D bases started at is stored in buckets in the hash table.

2.2.5 SRmapper Indexing Function and Formation of the Hash Table

For each key, SRmapper's indexing algorithm creates a bucket that stores the genomic locations of each key from the reference. For the human genome where D=15, it was impossible to store all the keys for buckets in memory on a system with 4GB of RAM since each key required 4 bytes of memory to point to a bucket and $4 \frac{bytes}{bucket} \times 4^{15} buckets = 4GB$ (in a system with 4GB of RAM, 1GB is usually devoted to the operating system). An additional 4GB of RAM would be required for each actual bucket to store a genomic location of the keys.

This issue was circumvented in two ways. First, a bucket was only declared, loaded into memory, if its key was found in the reference genome (**Fig 2.4 A**). Second, only 4^{D-1} keys were considered at one time. This was accomplished by making four passes through the reference genome in building the index. On the first pass, a key was only considered if it started with 'A'; on the second pass, a key was only considered if it started with 'C', and so on.

Additionally, it was determined that creating a bucket that held five reference locations minimized memory (**Fig 2.4 B**). Although the index was designed such that each key would be found one time on average for any genome, in the case of the human genome, keys that occurred tended to occur multiple times due to the repetitive nature of the human genome. To guard against overflow, every bucket must also point to the location of another bucket to catch reference locations of overflow keys (**Fig 2.5**). A traditional bucket contains memory for one entry and a pointer to an extra bucket to catch overflow. This requires 8 bytes of memory per entry: 4 bytes for the entry itself, and 4 bytes to point to an overflow bucket. These overflow buckets can be chained together indefinitely to catch the rare large overflow. However, every time another overflow occurs, the chain of buckets must be traced to its end to declare an empty bucket which is also time-consuming. Thus, by creating buckets that could store up to five entries, the length of the chain to be traversed in large overflows was greatly shortened. By always having an average key occurrence of one, many keys happened to not occur in the human genome since a few keys occurred many times. This greatly increased the memory effectiveness of not declaring a bucket until its key was found in the genome.

For bacterial genomes, the amount of memory required does not need to be as carefully managed since the genomes are smaller. Thus, although declaring a bucket that can hold five genomic locations is not ideal for memory efficiency in a genome without many repetitive regions, the small size of bacterial genomes results in the additional memory usage being inconsequential.

2.2.5.1 Initial Hash Table Formation Algorithm

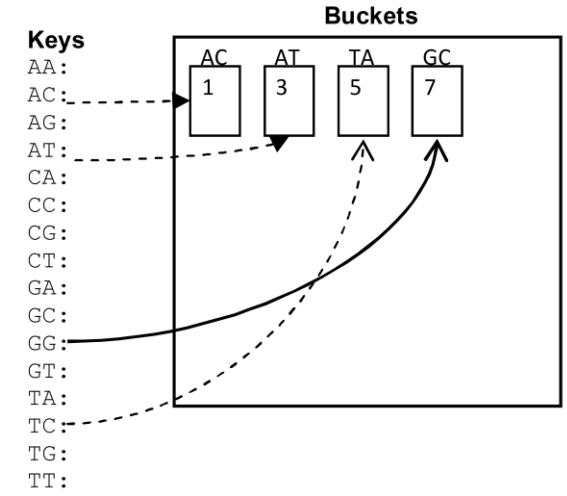
The initial indexing algorithm that constructed the hash table read the first D bases (1 through D) of the reference to form a key. This key was passed through the hash function and the reference location stored in its corresponding bucket. The algorithm then read the next base of the reference in and formed a key from bases 2 through D+1 and again hashed the key and stored the location in the corresponding bucket. This created approximately R entries into the hash table. For the human genome of R≈3,000,000,000, this would require the formation of 3 billion entries, and with each entry requiring at least 4 bytes of RAM, at least 12GB of RAM would be required to store all the entries or ~3GB of RAM for a quarter of the entries that started with each base. Adding in the 1GB of RAM required for the keys resulted in at least 4GB of RAM being required even without allocating for overflow. Thus, storage of every location entry for every key was abandoned as it would require the usage of more memory than the 3GB of RAM available on a system with 4GB of RAM.

2.2.5.2 Modified Hash Table Formation Algorithm

To reduce the amount of memory used in forming the hash table, a method that formed a comprehensive index yet did not create as many entries was required. The method chosen used bases 1 though D, then D+1 through 2D and so on to form keys instead of 1 though D, 2 though D+1, etcetera (**Fig 2.4 A**). This resulted in an index that only created one entry for every D bases. For the case of the human genome with D=15, this reduced the number of entries by a factor of 15 from roughly 3,000,000,000 to roughly 200,000,000. By utilizing this strategy, SRmapper's buildindex algorithm was able to index the entire human genome while using approximately 2.3GB of RAM, well under the maximum available memory on a system with 4GB of total RAM.

A

ACATTAGCATGAGACT



B

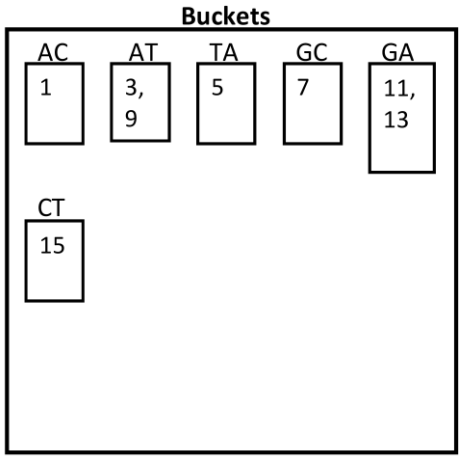


Figure 2.4: Formation of the SRmapper Index Using Buildindex. (**A**) To create the index for the reference sequence ACATTAGCATGAGACT, SRmapper first determines key length, D, which in this example is floor($\log_4(16)$)=2. The indexing algorithm, buildindex, then scans through the reference sequence two bases at a time. As buildindex scans through the reference sequence, it encounters the sequences AC, AT, TA, GC, etc and creates buckets for each of them. In general, buildindex creates a bucket when it

encounters a sequence for the first time. As buildindex scans through the reference, it records the location in the reference in each corresponding bucket. In practice, the reference is scanned through four times with only sequences starting with ‘A’ being evaluated in the first pass, sequences starting with ‘C’ being evaluated in the second pass, etc. Although not shown in the figure, this four scan process is discussed in section 2.3.5. The underline bases highlight the indexing of GC which starts at base number seven in the reference sequence. The solid arrow points to the bucket that is formed for GC, and the 7 inside the bucket denotes the location in the reference sequence where GC was located. (**B**) The complete index for the reference sequence ACATTAGCATGAGACT. Only the buckets that have entries are ever created to reduce memory usage. The buckets corresponding to keys AT and GA have two entries each. Each bucket can store up to five entries as discussed in section 2.2.5. (Adapted from Gontarz et al., 2013).

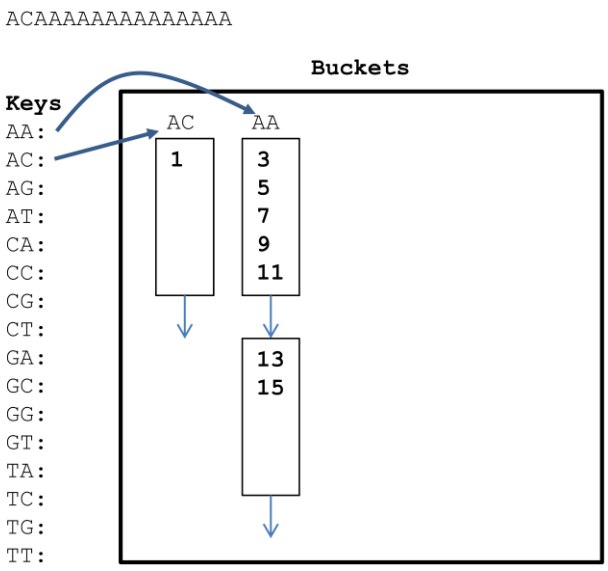


Figure 2.5: Overflow in Indexing. For the example reference sequence ACAAAAAAAAAAAAA simulating a poly-A tail, a key length of 2 is used. The 14 A nucleotides in a row result in 7 entries for the bucket corresponding to AA. Since each bucket can contain a maximum of five entries, after the fifth entry is read into the first bucket for AA, there is no more room in that bucket for entries. When the sixth AA is encountered (reference sequence position 13), a new bucket is created to handle the overflow and a pointer from the first bucket for AA records the location in memory of the second AA bucket. The bucket for AC demonstrates that even when only one entry is required for a bucket, enough memory is allocated for five entries. Additionally, all buckets contain a pointer. As long as additional buckets are not needed for overflow, the location in these pointers is set to NULL and no overflow buckets are created. (Adapted from Gontarz et al., 2013).

2.2.6 Output and Storage of the SRmapper Indexing Algorithm

The indexing algorithm creates as output three separate files for storage of the index and its properties. It also creates a fourth binary file comprised of a compressed reference sequence. These files are stored with the goal of being minimal in size and quickly loadable into memory. The first file that is created is a binary file that stores as a list how many times each key is found and where to look up the key locations in a second binary file that lists all the key locations (**Fig 2.6**). Both these files store their values as 4-byte binary values. The first file is given the extension .sqn and the second file is given the extension .sqn.val. The third file contains index properties such as the reference size, the value of D, the references used along with their individual lengths, and the number of entries in each quarter of the index. This third file dictates how the index is loaded into memory in alignment since it is only created one time. It is given the extension .sqn.hdr. Finally, a binary version of the reference is created by taking the two bit representation of each base: 00 for A, 01 for C, 10 for G, and 11 for T. Thus, 4 bases are compressed into their 1 byte binary value. For example, the sequence CAGT is represented as 01001011. This file is given the extension .sqn.bfa.

2.3 Alignment of Short Reads from NGS to Reference Sequences Using the SRmapper Align Algorithm and Probabilistic Model

SRmapper's alignment algorithm requires files being used for alignment be in the .fastq format. Short reads from NGS experiments are aligned one at a time by first anchoring them to the reference using the index created by SRmapper's buildindex algorithm. Alignment is then extended by directly comparing the remaining nucleotides

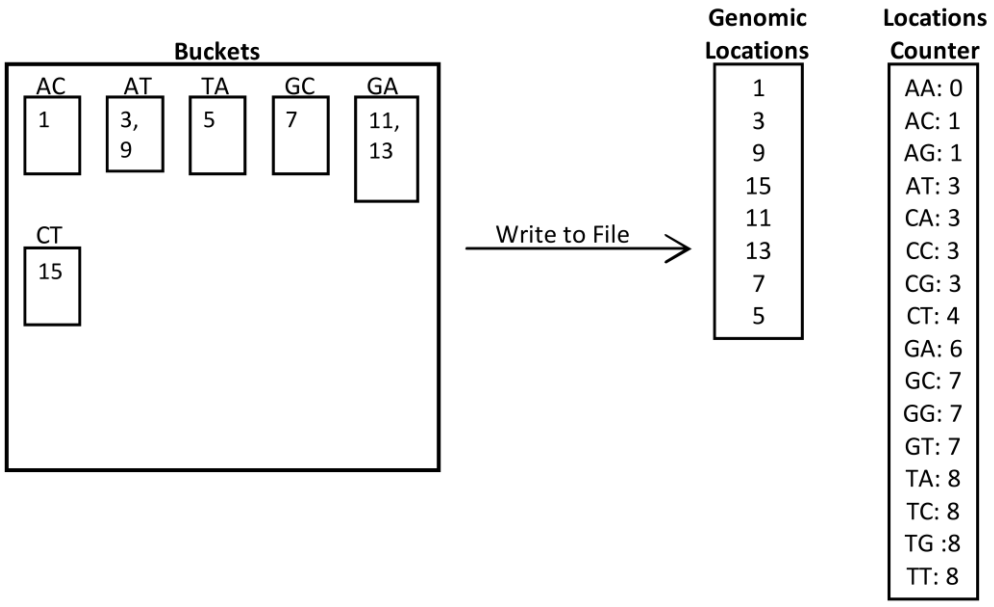


Figure 2.6: Storage of the SRmapper Index. The completed index from **Fig 2.4** forms two files that can be quickly loaded into memory due to their structure. The index stored for permanent record by the creation of two files. A counter is used to track how many genomic or reference locations have been written to file. Writing to file starts by checking how many entries are in the first key, AA. Any entries are written to the genomic locations file which has the extension .sqn. The total number of written entries are stored in the locations counter file that was given the extension .sqn.val. Since there are no entries for AA, nothing is written to the locations file, and 0 is written to the counter file. This process is then repeated for the next key, AC. Since there is an entry for AC, it is written to the locations file, the counter for how many entries have been written to file is increased by one, and the number of the current number of entries written to file is written to the counter file. This process is carried until all possible keys have been processed. (Adapted from Gontarz et al., 2013).

from the read to the reference genome and allows SRmapper to align reads with discrepancies between the reference sequence and the reads. SRmapper's alignment algorithm uses a probability-based determination to set a default upper limit of variants between the reference and the read by determining the likelihood an alignment occurs by chance based on the length of the read, the length of the reference, and the number of discrepancies in the alignment. This allows SRmapper to determine the quality of an alignment. The probability threshold can be modified at run time or a set number of mismatches can be chosen. SRmapper does not limit how long reads can be but does set a default maximum expected length of 1000 bp. This value can also be modified by users at run time. SRmapper does not require reads to be of the same length unlike certain other alignment algorithms. This allows users to trim low quality ends off reads without causing SRmapper to fail. However, SRmapper does require reads to be at least D bases long for anchoring to occur. SRmapper's alignment algorithm has the ability to perform both single-end alignment and pair-end alignment. The Align pseudocode is available at the end of the chapter as **SRmapper Align Pseudocode**.

2.3.1 SRmapper Align Input Format and Files

SRmapper's alignment algorithm requires the index files created by the buildindex algorithm and input sequence files to be of the fastq format. The alignment algorithm is invoked by the following command:

align <index.sqn> { <Reads1.fastq> <Reads2.fastq> … <ReadsN.fastq> }
<alignment.sam> [options]

where index.sqn is the index formed by buildindex and Reads1.fastq through ReadsN.fastq are the input files from sequencing experiments in fastq format.

Alignment.sam is the output file that stores the alignments in the SAM format which will be discussed later. The options that are available for users to customize alignment are as follows:

- -p [int]: -p allows the user to choose some integer number ‘p’ alignments to display in the output file. Only alignments that have the highest quality score are written to file. By default one alignment is printed per aligned read. In the case of a higher number of alignments of equal quality than p, p alignments from the set of alignments with the highest quality are chosen at random to be printed and their quality, or confidence of being the correct alignment, is set to 0. In the case where greater than one alignment but fewer than p alignments of equal quality are found for a read, all alignments are printed with their quality set to 0.
- -m [int]: -m allows the user to bypass SRmapper’s probabilistic model of determining how many mismatches should be allowed between the reference and an alignment and set some maximum integer number ‘m’ mismatches to be allowed in alignment.
- -a [int]: -a allows users to stop searching for alignments with a certain number of mismatches if ‘a’ alignments have already been found containing that number of mismatches. This prevents the exhaustive search in repeat regions but affects only a small portion of alignments. By default, ‘a’ is set at 5.
- -r [int]: -r modifies the maximum read length SRmapper expects. By default, this value is set at 1000.
- -q [int]: -q allows users to set the minimum quality that is allowed for alignment. q is scaled logarithmically with $q = -\log(confidence)$.

- -g [ull]: -g allows users to set a length for the reference genome other than the one that saved in the .sqn.hdr file. This can be useful if only aligning to a portion of a multi-chromosome genome but wanting quality scores that reflect alignment against the whole genome.
- -s [int]: -s allows users to determine how many entries from a single key to be checked as anchors for alignment. By default, 's' is set at 100. Setting s to -1 searches all entries for a key. This policy of limiting the number of entries searched per key affects less than 0.5% of reads but increases alignment speed by an order of magnitude.
- -P: -P sets alignment to pair-end mode. In pair-end alignment, input files are entered as { <file.1.1.fastq> <file1.2.fastq> … <fileN.1.fastq> <fileN.2.fastq> } where files ending in .1.fastq contain one end of the pair and files ending in .2.fastq contain the other end of the pair.
- -i [int]: -i sets the maximum insert size allowed between the two ends in a pair-end alignment. By default, this value is set at 1000 bp. Thus, alignments where the two pairs are located more than 1000 bases apart are not considered valid.
- -f [str]: Specifying -f followed by a file name allows users to define a file where reads that SRmapper could not find alignments for can be dumped out in fastq format so that other alignment algorithms can attempt to align these unaligned reads.
- -d: Specifying the -d option stores additional information about the alignment results to an separate file for downstream processing. This file is automatically given the same prefix as <alignment.sam> but is given the suffix '.sam.data'.

.fastq, also sometimes abbreviated .fq, files contain one or more, but usually many, reads produced by sequencing instruments along with information giving each read a name and information on the quality of each base call - a measure of how confident the sequencer was in identifying a base in the sequence (**Fig 2.7**). Specifically, the format for each read in a .fastq file are four lines in the following format: the first line is always started by the '@' character followed by a sequence identifier. The second line contains the sequence of the read. The third line is always started by the '+' character and can either be blank afterwards or contain the same identifying string of characters from the first line. Finally, the fourth line contains the base quality for each sequenced nucleotide from the second line. In the case of pair-end sequencing, two files are created. The first file contains all the reads containing one end of the sequence, and the second file contains the other end of the sequence. The .fq format requires that the reads are kept in the same order in both files and that both pairs are present for a read to be included.

2.3.2 SRmapper Align Prealignment Routines

When invoked, SRmapper's alignment algorithm performs several checks and processes before performing any alignment to prevent crashes or errors. The alignment algorithm first determines whether valid usage options have been given and what options are being used. It then attempts to open the index files and checks that they are in the correct format. Finally, it attempts to create output files to ensure write capabilities. If all the checks pass, align proceeds to calculate probability scores for alignments of various read lengths and mismatches, builds a alignment probability table, loads the index into memory, and aligns the reads from the .fq file or files.

2.3.2.1 Determination of Alignment Probabilities

```
@Read.1
AGCAGACCTACCTACTGACCC
+
&#$B#$1A7^^D<49AA>>?=
@Read.2
AGTGAGGTATATACCTANNNN
+Read.2
#@%&@#72&:;AC761$####
@Read.3
ACCCCTAGCATCGATGTTACCGATCGAGC
+
1%##@57DDDCA#@%&@#72A339%#$$#
```

Figure 2.7: The .fastq File Format. Reads in the .fastq format have the same general four line structure. The first line provides the name of the read and always starts with the '@' symbol. The second line contains the sequence of the read. Ambiguous bases are represented by an 'N' and are sometimes trimmed off a read before alignment. The third line always starts with a '+' symbol and optionally repeats the read name from the first line. The fourth line provides the quality or confidence in each base call made by the sequencing instrument. Reads in the .fastq format do not all need to be of the same length although some alignment algorithms do require all reads to be the same length. SRmapper is tolerant of a wide variety of read lengths.

To determine the number of mismatches permissible between a short read and the reference, the alignment algorithm in SRmapper uses probability function determines the likelihood an alignment can be generated by chance with some number of mismatches, M, between a read of some length, L, and a reference of length R. If a high enough number of mismatches are allowed in an alignment considered valid, any read could align to a reference based on random chance. To calculate this probability of a read aligning by chance, two assumptions are made about the reference. First, it is assumed for all integers j and k with $j \neq k$, the identity of nucleotide j (N_j) and nucleotide k (N_k) in the reference sequence are independent of each other. Next, it also assumed that for any j, there is an equal probability that N_j is A, C, G, or T. The probability function for generating alignments by random chance can be determined as follows. The probability that a particular base from the read will match a random base in the reference is 1/4. For a read L nucleotides long, there are 4^L distinct combinations for the sequence of that read. Thus, there is a $1/4^L$ chance that a perfect alignment with no mismatches occurs at a particular location in the reference sequence since the likelihood of any base occurring at a given location is assumed to be equal. Were there allowed to be one mismatch between itself and the reference, there would be L possible locations where that mismatch could occur since it could occur at any base. Since there is one match and three possible mismatches at each location on the read, there are 3L combinations that can result in an alignment with one mismatch. The probability of an alignment with one mismatch is then $3L/4^L = 3^1C_1^L/4^L$ where C_a^b represents the number of combinations possible to choose *a* elements from a set of *b* elements provided the order of choice is not considered. $C_a^b = \frac{b!}{a!(a-b)!}$. It

can be seen that for b=L and a=1, $L = \frac{L!}{1!(L-1)!}$ since $L! = L(L-1)!$. For an alignment with two mismatches, there are L locations where the first mismatch can occur and L-1 locations where the second mismatch can occur. Thus, there are L(L-1)/2 combinations to have two mismatches in read of length L. The division of L(L-1) by two occurs since for any *j* and *k*, a read with mismatches in its alignment at N_j and N_k is not considered distinct from a read with mismatches at N_k and N_j. Since there are three possible mismatches at each location *j* and *k*, for any *j* and *k*, there are 3^2 distinct sequences with mismatches at *j* and *k*. Thus the probability of an alignment occurring with two mismatches is $\frac{3^2 L(L-1)}{2} \Big/ 4^L$ or $3^2 C_2^L \Big/ 4^L$ We can again see that $\frac{L(L-1)}{2} = n_2^L$ since $\frac{L!}{2!(L-2)!} = \frac{L(L-1)(L-2)!}{2(L-2)!} = \frac{L(L-1)}{2}$. In general, this principle holds, and the probability of a read aligning with M mismatches is then $3^M C_M^L \Big/ 4^L$. The probability of an alignment with M or fewer mismatches is then

$$P(L,M) = \sum_{i=0}^{M} \left(3^i C_i^L \Big/ 4^L \right) \qquad \text{Eq 2}$$

The probability that an alignment does not occur randomly at a specific location is then 1-P(L,M). For a reference of length R, the probability of an alignment not occurring at any location is the cumulative probability of an alignment not occurring at each of the possible locations within the reference. This is given by:

$$P(L,M,R) = \big(1 - P(L,M)\big)^R = \left(1 - \sum_{i=0}^{M} \left(3^i C_i^L \Big/ 4^L \right)\right)^R \qquad \text{Eq 3}$$

The probability that a read then does align by chance somewhere over the genome is $1 - P(L, M, R)$. The value for this probability is reported as a phred score. Phred score has evolved from its initial use as a measure of experimental error to become the standard method for reporting alignment quality in alignment algorithms. In the case of SRmapper, phred score is used to estimate the probability of a read aligning by chance to the reference genome with up to a certain number of mismatches. Phred scores are related to the probability in that they are the negative logarithmic value of the probability scaled up by a factor of 10. Thus, alignment quality reported by SRmapper is the phred scaled value of equation 3:

$$phred(L, M, R) = -10log\left(1 - \left(1 - \sum_{i=0}^{M}\left({}^{3^i C_i^L}/_{4^L}\right)\right)^R\right) \qquad \text{Eq 4}$$

Although it is not plausible to raise some number to the 3 billionth power as would be necessary in calculating phred scores for alignments to the human genome, an excellent approximation can be made as follows: if a=1 and b=$-\sum_{i=0}^{M}\left({}^{3^i C_i^L}/_{4^L}\right)$, we can rewrite equation 3 as $1 - (a + b)^R$ which can be expanded to $1 - (a^R + Ra^{R-1}b + \cdots + b^R)$. However, since b is extremely small in all cases except where the alignment qualities are very low, terms beyond $Ra^{R-1}b$ are so small that they can be ignored. Thus, equation 3 becomes $P(L, M, R) = 1 - R\sum_{i=0}^{M}\left({}^{3^i C_i^L}/_{4^L}\right)$ and equation for can be transformed into

$$phred(L, M, R) = -10\log\left(1 - \left(1 - R\sum_{i=0}^{M}\left({}^{3^i C_i^L}/_{4^L}\right)\right)\right) =$$

$$-10\log\left(R\sum_{i=0}^{M}\left({}^{3^i C_i^L}/_{4^L}\right)\right) \qquad \text{Eq 5}$$

Equation 5 can be readily calculated and is used in SRmapper's quality scoring function. **Figure 2.8** provides an example demonstrating that the approximation of equation 4 by equation 5 closely models the phred scores from equation 4. When SRmapper is run without the -m option being specified, the maximum number of mismatches, M_m, is chosen such that a phred score of 30 is obtained. This phred score correlates to a minimum of a 99.9% chance of an alignment of a read to the reference is not generated spuriously. The cutoff value for phred score can be adjusted by setting the -q option and choosing a different minimum phred.

2.3.2.2 Creation of the Probability Table for alignment

Instead of calculating probabilities of alignment for each read to determine the maximum number of mismatches between the read and reference, SRmapper creates a table of probabilities and phred scores for every possible read length up to the maximum read length and for every number of mismatches producing a phred score equal to or greater than the default minimum phred score or the phred score specified by the user. This results in the calculation of a few thousand probabilities (1000 bp default max read length multiplied by the number of mismatches allowed) over the course of aligning all the reads from a file. In contrast, calculating phred scores on the fly for each read would result in the calculation of hundreds of millions or billions of phred scores depending on how many reads were present in the alignment file. Thus, calculating and storing the phred scores in advance significantly reduces the amount of calculations made in alignment.

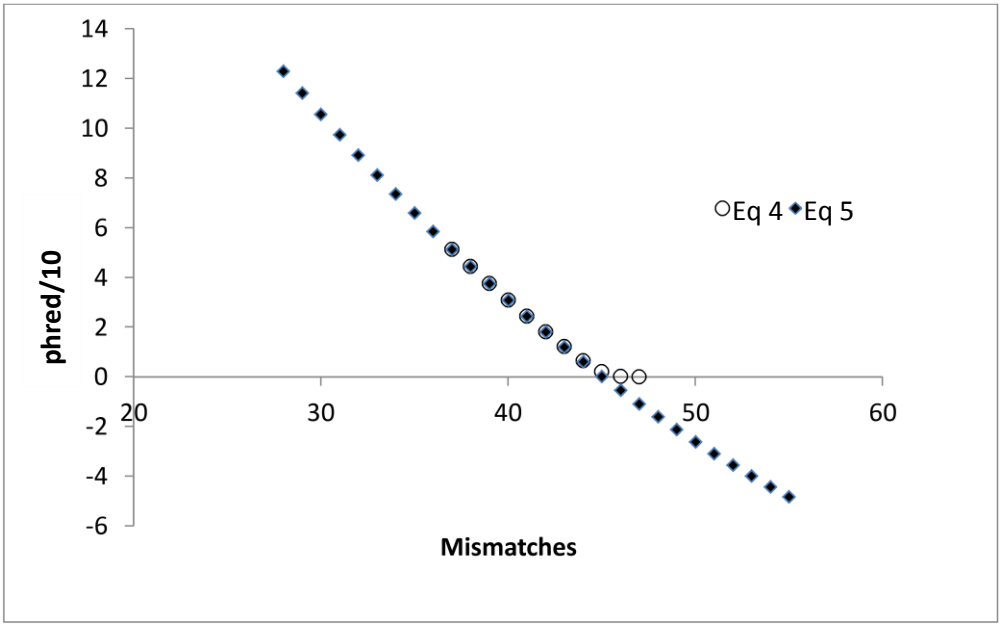


Figure 2.8: Approximation of Eq 4 by Eq 5. The phred score for a 100 bp read being aligned to a reference of length 3 billion to simulate the human genome are determined as the number of mismatches between the reference and read increases. For all number of mismatches for which phred/10=1, equation 4 is closely modeled by equation 5. Since the default minimum phred allowed in an alignment is 30, and this value is often even set higher, equation 5 always models equation 4 accurately under user conditions. Note that calculating values for equation 4 not plausible above certain phred scores. (Adapted from Gontarz et al., 2013).

2.3.2.3 Loading of the Index into Memory

Due to the way the index is written to file in binary format during the indexing step, the alignment algorithm can load the index in a ready-to-use format simply by reading in blocks of the index files, and no formatting of the index is necessary to make the index quickly accessible and usable. Loading of the index starts by reading from the .sqn.hdr file to determine the index parameters such as reference length or lengths, key length, and number of locations stored in each quarter of the index. Next, the requisite amount of memory to store the binary reference genome is determined from the reference length and allocated and the binary reference is loaded into memory. The number of keys is determined from key size, memory is allocated for one quarter of the keys, and those keys are loaded into memory. Finally, the memory to store number of entries in the first quarter of the index is loaded based off the values from in .sqn.hdr file and the entries are loaded.

2.3.3 Sequence Alignment by the SRmapper Align Algorithm

SRmapper requires reads to be in .fastq format and in base space (SOLiD reads in color space first must be converted into base space before alignment). The SRmapper alignment strategy uses the seed-and-extend strategy somewhat similarly to the strategy employed by SSAHA (Ning et al., 2001) and Stampy (Lunter et al., 2011) although the methods for seeding and extension are both different from the previously mentioned algorithms. In the first step of alignment, SRmapper takes the first D bases from the read and passes them through the same hashing function as is used by the indexing algorithm. SRmapper uses the entries from the key established by processing the D bases to establish possible alignments (**Fig 2.9**). In the second step of alignment, the bases not

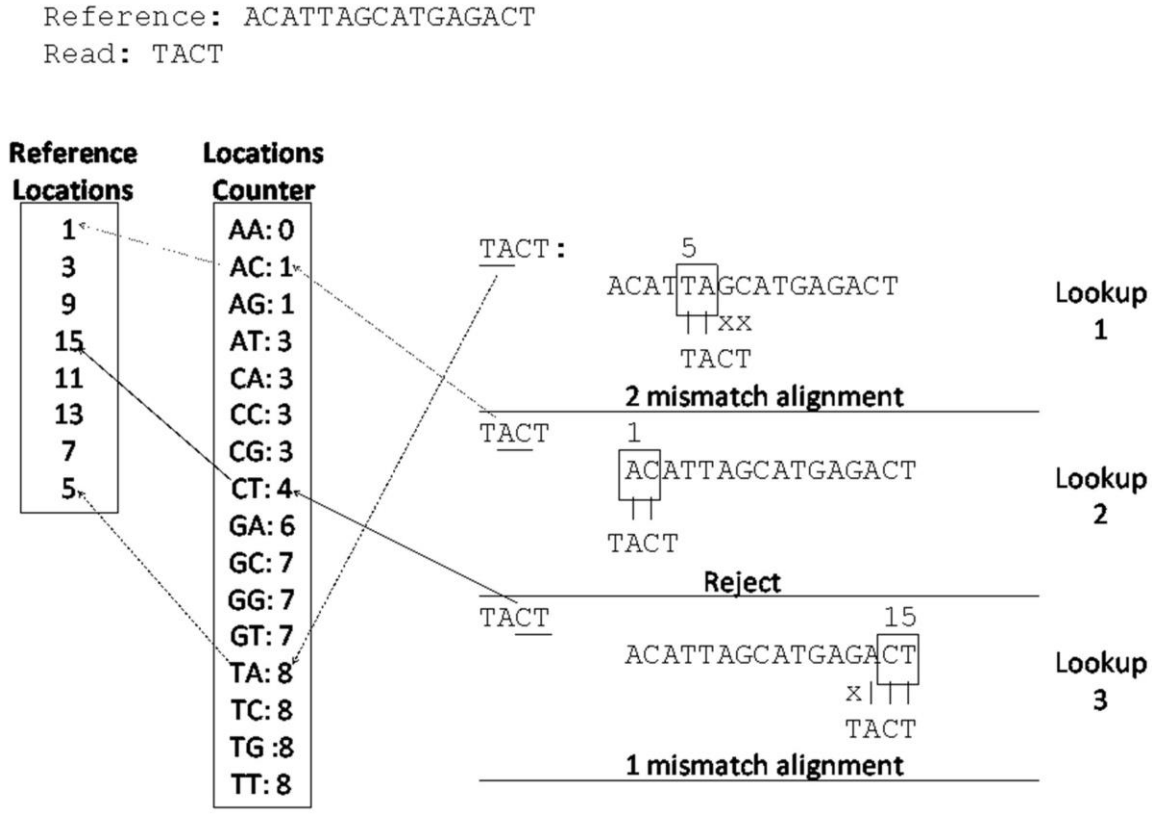


Figure 2.9: Alignment of a Short Read to a Reference Sequence by the Align Algorithm of SRmapper. The short read TACT is aligned to the reference sequence ACATTAGCATGAGACT by the align algorithm of SRmapper using the index created by the buildindex algorithm of SRmapper in **Fig. 2.4** and **Fig. 2.6**. The underlined bases in the read denote which bases are used in the first step of alignment - the anchoring step. For the reference above, D=2. Lookup proceeds by taking two bases from the read and determining where they are found in the reference sequence using the locations counter and reference locations. For the first lookup, the first two bases in the read, TA, are used. Lookup proceeds by starting after the counter value from the previous key and ends at the counter value for the current key. Thus, for the key TA, the previous key, GT, is used to determine to start looking in the reference locations list after the 7^{th} number in the list. The counter value for TA is used to determine to stop looking in the reference locations

list after the 8th number in the list. Hence, the first and last number looked up is the 8th number in the reference locations list as denoted by the arrow pointing from the locations counter for TA to the 8th number in the reference locations list. The value of five in the locations list dictates to anchor the read bases TA starting at the fifth base in the reference sequence. The remaining bases are directly compared to the reference sequence using the binary form of the reference (not shown in figure). In the case of lookup 1, subsequent direct comparison determined that two bases in the read formed mismatches with the reference sequence. If the original value for M_m had been higher than 2, the new value for M_m would be set at 2 and the alignment would be considered a valid possible alignment. For the second lookup, bases two and three, AC, are used. The second potential alignment is rejected since the read extends off the end of the reference sequence. The third lookup uses the bases CT. Direct comparisons determines that the candidate alignment has one mismatch. Since this is the lowest mismatch alignment, it would be reported to be the correct one. (Adapted from Gontarz et al., 2013).

aligned by the seeding step are compared to their corresponding bases from the binary form of the reference sequence. Comparison continues until all bases have been aligned, in which case a successful alignment is achieved, or until more discrepancies are found between the read and reference than the maximum number of mismatches, M_m, allowed.

If an alignment is found, M_m is decreased to the number of mismatches found in that alignment to prevent searching for suboptimal, lower quality, alignments. The location of the alignment is stored along with the number of mismatches in the alignment.

After all the entries from the key from bases 1 through D have been checked, steps 1 and 2 of alignment are repeated using bases 2 through D+1. This process is repeated until all L bases of the read have been used in the first step of alignment or until $D(M_m + 2)$ bases have been used in the first step of alignment. The $D(M_m + 2)$ case applies to when an alignment with few mismatches is found for a read and can be justified as follows: if an alignment with no mismatches exists, then using the first 2D bases to create keys from bases 1 through D, 2 through D+1, …, D+1 through 2D as seeds on the index containing only non-overlapping segments of the reference will guarantee that the seed will be found in the index. If there is an alignment with one mismatch, the slowest alignment scenario would be for the mismatch to occur at the D^{th} base so that a key containing no mismatches could not occur until bases D+1 through 2D are used to form the key. This requires that all keys until the one generated by 2D+1 through 3D be used since the index is non-overlapping. This concept extends as the number of mismatches allowed in alignment increases.

The benefit to limiting the number of keys searched is as follows: since the majority of alignments have few to no mismatches, the proper alignment or alignments

will be found using the first few keys. For read of length 150 bp aligning to the human genome (D=15) with no mismatches, only the keys from 1 to D through D+1 to 2D need to be used and keys from 2D+1 to 3D through 9D+1 to 10D do not need to be considered. This means that only D keys need to be considered instead of 9D keys in this case saving a significant amount of time in alignment.

After finding alignments on the forward strand of the reference sequence, the reverse read is generated to search for alignments on the reverse strand. When the reverse complementary sequence is generated, M_m is not reset to its original value. Instead the current value of M_m is retained based on any changes to it that have been made to M_m due to alignments to the forward strand. This prevents suboptimal alignments being searched for on the reverse strand. Alignment steps 1 and 2 are repeated through keys generated by bases $D(M_m+2)$ as per the alignment to the forward strand. Alignments with M_m or fewer alignments are retained as with the forward strand and written to file.

After all reads have been aligned to the first quarter of the index, the first quarter of the index is removed from memory, and the second quarter of the index is loaded. All the reads are again aligned using the 2^{nd} quarter of the index to create anchors for the alignments. M_m is retained for each read to prevent generating suboptimal alignments for each read through the different quarters of the index. After alignments from the 2^{nd} quarter are written to file, the same process is repeated for the 3^{rd} and 4^{th} quarter of the index.

2.3.4 Alignment Output and Storage

Since SRmapper only aligns to a quarter of the reference at a time, it cannot store all the possible alignments generated from each quarter of the index in memory until after

all alignment is finished because it is not feasible to store all possible alignments in memory if there are tens or hundreds of millions of reads. Instead, SRmapper's alignment algorithm stores candidate alignments in four temporary files, one for each quarter of the index, until all four quarters of the index have been used for seeding alignments. These temporary files are kept as small as possible and only store essential information about the alignments such as location, quality, and strand. If the maximum permitted number of equal quality alignments per quarter is reached, M_m is decreased by one mismatch. After these temporary files are generated and all four quarters of the index have been used, the candidate alignments from the temporary files are compared to each other and the alignment or alignments with the highest quality are chosen. The final output for the alignment is presented in the Sequence Alignment/Map (SAM) format (**Fig 2.10**). For single-end alignment, SRmapper can display one or more alignments of equal quality. For pair-end alignment, only the highest quality alignment is printed. SRmapper does not report reads that failed to align in the SAM output file unlike most alignment algorithms. Instead SRmapper allows users the option to output the reads that could not be aligned in a .fastq file so that they can attempt to align the other reads with slower alignment algorithms. In terms of the SAM output file for the reads that can be aligned, a very specific format for output is followed so that downstream applications can be used to analyze alignment output. Specifically, the format that is required are header lines specifying the names of the reference sequences and their lengths followed by a '*tab*' delimitated line for each aligned or unaligned reads with the following fields:

1. Query name: The name of the sequence that was aligned.

2. FLAG sum: the name of bitwise flags containing information on the alignment such as strand, single-end alignment versus pair-end alignment, and number of segments. A more detailed description will be provided later.
3. Reference sequence name: The name of the reference sequence to which the read aligned. This name will be the same name as one of the header lines.
4. Alignment position: The position of where the first base aligns to the forward strand. In the case of a read that aligns to the reverse strand, the location of the where the last base aligns is reported as this translates to the location of the first base on the leading strand.
5. Alignment quality: the phred score calculated by SRmapper's alignment algorithm. In the case of multiple reads aligning with the same phred score, a phred score of 0 is reported signifying no confidence on which alignment is correct.
6. CIGAR string: Base by base information on the alignment. M, I, and D represent matches and mismatches, insertions, and deletions respectively. For SRmapper, the 6^{th} field will always list the read length followed by 'M'.
7. Pair reference: The name of the reference sequence to which the read's pair aligned in pair-end sequencing. In the case of single-end alignment, this field is filled with an asterisk. In the case of pair-end alignment, this field is filled with an '='.
8. Pair alignment position: The location on the reference to which the read's pair aligned in pair-end sequencing. In the case of single-end sequencing, this field is filled with an asterisk.

9. Observed template length: Mandatory field that denotes the total length over which the read spans for reads partially aligned to multiple reference sequences. This field is always reported as ‘0’ by SRmapper.
10. Segment sequence: The sequence of the aligned read. This string is the same as the second line of each read in the .fastq file.
11. Base quality: The quality of the bases in the alignment. This string is the same as the fourth line of each read in the .fastq file except that they are each increased by 33 to convert them into ASCII text.

The FLAG score from the second field of the SAM output is the sum of the following flags that apply to SRmapper:

1 - The read has multiple segments. This field as applicable in pair-end alignment

2 - Both segments are properly aligned. Since SRmapper does not consider a pair-end alignment valid unless both pairs align, this flag will always be set in SRmapper pair-end alignment.

4 - Sequence not aligned. Since SRmapper does not print unaligned sequences, this flag is never set.

8 - Pair not aligned. Since SRmapper does print pair-end alignments where one or more pairs is not aligned, this flag is never set.

16 - Sequence aligns to the reverse strand. This flag applies to both single-end and pair-end alignments.

A

```
Ref:        ACTCAGGCAGGTACGATACCGATACGAAAT

Read1               AGGTACCATCC
Read2.1       TCAGGCACG
-Read2.2                         GATAGGAAA
```

B

```
@SQ        SN:Ref    LN:30
Read1       0  Ref   9 50 11M * 0   0 AGGTACCATCC *
Read2.1    99  Ref   3 40  9M * = 21 TCAGGCACG   *
-Read2.2  147 Ref 21 40  9M * =  3 GATAGGAAA   *
```

Figure 2.10: The SAM Output Format. (**A**) Example alignment of three reads to a reference sequence by an alignment algorithm. Read1 is a single end alignment while Read2.1 and -Read2.2 are two mates in a pair-end alignment with the sequence of -Read2.2 being reverse complemented for clarity. (**B**) The SAM output file for the alignments in A as would be created by SRmapper. The first line is a header line as denoted by the '@SQ' tag. 'SN' signifies a field for the name of the reference being used, and 'LN' signifies a field for the length of the reference. Each alignment is represented by a single line of text composed of 11 fields in the SAM output format. The first field is the sequence name. The second is the bitwise FLAG. The third field is the reference sequence to which the read aligned. The fourth field is the location of the leftmost base in the alignment. The fifth field is the phred quality score (Note that in this example, no quality measurements were made but rather an arbitrary phred score was created for clarity). The sixth field is the CIGAR string, which is always represented by SRmapper as the length of the read followed by an 'M'. The seventh field is always reported as an asterisk by SRmapper. The eighth and ninth fields are the reference and reference location of the alignment of the mate in a pair-end alignment. For single-end alignment,

these fields are both recorded as 0. In pair-end alignment, the eighth field is represented by an '=' denoting both mates aligned to the same reference as reported in field three. The tenth field is the sequence of the read aligned. The eleventh field is the same quality string as in the fourth line of the .fastq description of the read with each value incremented by 33 to convert into ASCII text. In this example, no quality information was provided about the bases in the reads, so the field is marked with an asterisk.

32 - Pair aligns to the reverse strand. This flag applies only to pair-end alignment.

64 - First read in a pair.

128 - Second read in a pair.

In the case of pair end alignment, there is an extra step in producing the alignments. Alignment proceeds in the same manner as in single-end alignment until after the temporary files have been created. Each mate in a pair-end read is first aligned as a single-end read, and all the possible single-end alignments are stored in the temporary files previously described except that four temporary files are created for the forward pairs, and four temporary files are created for the reverse pairs. Then, possible pair-end alignments are generated by attempting to find two candidate single-end alignments that form a proper pair. Two single-end alignments form a proper pair-end alignment if they align reasonably close to each other on the reference sequence. The distance between two pairs is considered acceptable if it is smaller than the maximum insert size allowed by SRmapper. This value is by default set to 1000 bp but can be modified by the user by using the -i option to more accurately reflect the maximum insert size between the two ends of a pair-end read if the maximum insert size of the sequencing technology is known. In contrast to single-end alignment in which only reads of maximal phred score are selected, pair-end alignment retains possible alignments that have a lower phred score. This allows more possibilities to find an acceptable pair which is considered desirable since two possible single-end alignments with higher phred scores do not constitute a proper pair-end alignment unless the single-end alignments are sufficiently close to each other. Thus, although the individual phred scores for the single-end alignments may not be as high as other candidate alignments, forming a proper pair

governs whether a candidate single-end alignment is valid or not. In the case where multiple, proper pair-end alignments are found, the pair-end alignment with the fewest combined mismatches is chosen as the correct alignment. The phred score that is reported in pair-end alignment is the score for a read which has a length equal to the sum of the lengths of the two mates in the pair and a number of mismatches equal to the sum of the number of mismatches in the two mates. If two or more pair-end alignments of equal quality are found, one is randomly chosen and the reported phred score is set to zero as in single-end alignment.

2.3.5 Miscellaneous Implementations to Increase Alignment Speed

In addition to the $D(M_m+2)$ key search discussed above, SRmapper imposes two additional intuitive steps which result in a combined increase in alignment speed by over an order of magnitude. The first intuitive step is to decrease the number of mismatches allowed, k, by one if some set number of alignments containing k mismatches are found. By default, this value is set to five alignments per quarter of the index but can be modified by the user at runtime using the -a option. This modification does not affect the number of confident alignments found, those that have a phred score greater than 0, since confident alignments only occur when a single alignment with the highest phred score is found and this modification only affects reads that have multiple alignments of equal phred score.

The second intuitive step is to limit the number of entries looked for in each bucket. This step increases speed while slightly reducing the number of confident alignments found since not considering all entries in a bucket takes less time but may cause some alignments to be missed. This policy was created due to the large number of

entries found in a select few buckets when the human genome is used as a reference. Low-complexity regions in the human genome such as polyA-tracts, short interspersed elements (SINES), long interspersed elements (LINES), and dGdC islands result in a small number of keys with a very high number of entries. By default, SRmapper only considers up to the first 100 entries in any particular bucket, although this policy can be relaxed using the -s option. The default setting results in less than 1 in 25,000 buckets being affected and retains more than 99.5% of reads that would be aligned without this limit still being aligned. However, imposing this restriction increases the speed of aligned by more than a factor of 9x and is therefore considered a very reasonable exchange.

2.4 Results

SRmapper was extensively and directly compared to BWA since it has been one of the most popular alignment algorithms since its inception in 2009 and is most similar to SRmapper in the amount of memory used and contains similar functionality. Hence, most of the testing of performance of SRmapper is performed against BWA. However, by using comparisons between BWA and other algorithms, SRmapper has also been implicitly compared to several other popular alignment algorithms.

2.4.1 Indexing Reference Sequences

Both SRmapper and BWA require that the reference sequence or sequences only be indexed one time since they are written to file after being created. SRmapper is somewhat more flexible than BWA in that it can index multiple reference files simultaneously whereas BWA requires multiple reference files to be first concatenated into a single multireference (.mfa) file. In terms of indexing the human genome, SRmapper can take separate .fa input files for each chromosome or a single .mfa file

containing all the chromosomes whereas BWA can only use the single .mfa file. To index the human genome, SRmapper required 2350 s (seconds) using an Intel Xeon 2.8GHz processor. Using the same processor, BWA required 8100s to index the human genome meaning that SRmapper is approximately 3.5x faster in terms of indexing. SRmapper's index for the human genome requires slightly more disk space than BWA's index. The two algorithms require 5.4GB and 4.3GB, respectively, to index the human genome. However, since disk space is relatively inexpensive compared to memory and processor time, the size of the index is not a major concern. Admittedly, since the index only needs to be built once, the time taken for indexing is also not a serious issue within reason. For smaller references, such as bacterial genomes, indexing requires only a few seconds and is even less of a concern.

2.4.2 Comparison Between SRmapper and BWA Using Real and Simulated Sequencing Datasets

2.4.2.1 Real Datasets and Software

To compare the performance of SRmapper in comparison to BWA, several sets of sequencing data were download from the SRA (http://www.ncbi.nlm.nih.gov/sra). Datasets were chosen to reflect a variety of sequencing conditions but all came from human sequencing studies since aligning to human genomes tends to be among the most difficult tasks for an alignment algorithm due to the size of the human genome. Specifically, the datasets chosen were SRR002787, which contained 5.88M single-end reads with each read being 32 bp long; SRR006150, which contained 13.18M pair-end reads with each mate being 51 bp long; SRR020477, which contained 2.04M pair-end reads with each mate being 76 bp long; and SRR539393, which contained 2.25M pair-

end reads from the Illumina HiSeq2000 with each mate being 101 bp long. SRR006150, SRR020477, and SRR539393 were also evaluated as single-end reads to increase the number of datasets evaluated for single-end alignment and yielded 13.18M, 51 bp single-end reads from the forward strand for SRR006150; 4.08M, 76 bp single-end reads from both strands for SRR020477; and 4.50M, 101 bp single-end reads from both strands for SRR539393. All files were downloaded in the .sra (Short Reads Archive) format and were subsequently converted to the .fastq format for alignment using the SRA Toolkit's fastq-dump command. The SRA toolkit is a free resource from the National Center for Biotechnology Information available for download at http://www.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software. All comparisons to BWA were performed using BWA version 0.5.8c (http://sourceforge.net/projects/bio-bwa/) and SRmapper 0.1.2 (http://www.umsl.edu/~wongch/software.html) although since the time SRmapper was compared to BWA, small improvements have been made to both algorithms.

2.4.2.2 Alignment Conditions and Measures of Aligner Speed and Reads Aligned

Every attempt was made to produce fair comparisons between SRmapper and BWA. Comparisons were made attempting to use as the same alignment conditions for each algorithm and also to use each algorithm as similarly to what was intended in its implementation. This meant for BWA, the -o 0 option was invoked on all tests to disable gapped alignment and the -A options was invoked to prevent Smith-Waterman alignment. Apart from this, two separate comparisons were made between SRmapper and BWA - one in which BWA's default parameters were chosen for alignment and another in which SRmapper's default parameters were chosen for alignment. This was implemented by

allowing one algorithm to use its default mismatch settings and specifying the number of mismatches the other algorithm used to match the number of mismatches allowed by the algorithm whose default parameters were being used. Both BWA and SRmapper utilize seeding restrictions by default although the restriction methods are somewhat different. SRmapper limits entries searched per bucket as described above, and BWA restricts the number of mismatches permitted early in the alignment. Since these were intended measures to increase aligner performance, they were not modified as a comparison in which the algorithms were handicapped simply for meeting the purpose of an identical comparison is less meaningful than a comparison where the algorithms were used as intended.

Since both algorithms are run from a linux terminal, the 'time' command was prepended to the command to invoke either SRmapper or BWA in order to determine alignment time. SRmapper achieves alignment over one step whereas BWA achieves alignment in two distinct steps using two distinct commands. In the first step, BWA aligns reads to its suffix array in stores the suffix array alignments in binary form, and in the second step, it converts these alignments into the same text-based, SAM-formatted alignments that SRmapper uses. Thus, the overall time for BWA to perform alignment is the sum of its alignment and reformatting command. SRmapper performs both its alignment and formatting in one command. To measure the percentage of confidently aligned reads by each alignment algorithm, another algorithm was created that scanned through the SAM files generated by each alignment algorithm to count the number of confidently aligned reads. Determining confidently aligned reads is preferable to determining aligned reads since a read that is not confidently aligned is usually not used

in downstream analysis. For the purposes of determining whether a read was confidently aligned, two fields in the SAM file were checked for each alignment. First, the FLAG field was checked to ensure the '4' flag, specifying that a read was not mapped, was not set. Unlike SRmapper which does not report unmapped reads in its SAM file, BWA writes unmapped reads to the SAM file and sets the '4' flag. Second, the fifth field, phred score, was checked to filter out alignments that had a phred score of zero signifying an alignment with no confidence.

2.4.2.3 Results of Comparing SRmapper to BWA on Real Datasets

For all four datasets tested using single-end alignment, SRmapper ranged from being 2.1x to 39.0x faster than BWA (**Table 2.1**), (**Table 2.2**). Excluding SRR002787 which had extremely short read lengths, SRmapper varied from 2.1x to 8.7x faster than BWA depending on the dataset being studied and the alignment conditions set at runtime. At the same time, SRmapper retained similar alignment sensitivity to BWA with each algorithm aligning a similar percentage of reads except for in the case of the very short reads. For these, BWA had a higher sensitivity. These results are not surprising since SRmapper needs to find D bases that have no mismatches in them to anchor the alignment. This is a more significant issue with shorter reads since there are fewer keys to use. Thus if a short read has a few mismatches, SRmapper may be unable to find the alignment for it. Specifically for SRmapper, there are L-D+1 usable keys for a read of length L. Unlike SRmapper, BWA can tolerate a small number of mismatches in its seed sequence although the number of mismatches allowed in the seed sequence greatly increases alignment time. However, in every dataset tested, there was a high overlap of alignments found by both alignment algorithms. By creating software that determined

	Alignment Time(s)	SRmapper Speedup	% Reads Aligned
SRR002787			
BWA	4,404		56.84%
SRmapper	715	6.16x	50.74%
SRR006150			
BWA	10,673		73.49%
SRmapper	2,711	3.85x	72.68%
SRR020477			
BWA	1,482		66.35%
SRmapper	616	2.41x	67.62%
SRR539393			
BWA	2,715		90.0%
SRmapper	1,298	2.09x	90.2%

Table 2.1: Comparison of Alignment Time and Percent Reads Aligned for BWA and SRmapper using the default mismatch Parameters of BWA and Single-End Alignment. The alignment time and percent reads aligned for four datasets were measured using BWA and SRmapper. The datasets were SRR002787, SRR006150, SRR020477, and SRR539393 and contained 5.88M 32 bp reads, 26.28M 51 bp reads, 4.08M 76 bp reads, and 4.49M 100 bp reads respectively. The datasets were all downloaded from the Sequence Read Archive (http://www.ncbi.nlm.nih.gov/sra). The number of mismatches allowed for the above datasets were 2, 3, 4, and 5 mismatches respectively. BWA was run with the option -o 0 to disable gapped alignment. Alignment time for BWA is the sum of the times for BWA aln and BWA samse to run while alignment time for SRmapper is the time for SRmapper align to run.

	Alignment Time(s)	SRmapper Speedup	% Reads Aligned
SRR002787			
BWA	29,023		66.08%
SRmapper	744	39.01x	56.62%
SRR006150			
BWA	13,574		82.18%
SRmapper	3,611	3.76x	84.08%
SRR020477			
BWA	8,112		78.35%
SRmapper	930	8.72x	84.02%
SRR539393			
BWA	11,155		95.6%
SRmapper	1,548	7.21x	91.0%

Table 2.2: Comparison of Alignment Time and Percent Reads Aligned for BWA and SRmapper using the default mismatch Parameters of SRmapper and Single-End Alignment. The alignment time and percent reads aligned for four datasets were measured using BWA and SRmapper. The datasets were SRR002787, SRR006150, SRR020477, and SRR539393 and contained 5.88M 32 bp reads, 26.28M 51 bp reads, 4.08M 76 bp reads, and 4.49M 100 bp reads respectively. The datasets were all downloaded from the Sequence Read Archive (http://www.ncbi.nlm.nih.gov/sra). The number of mismatches allowed for the above datasets were 3, 12, 28, and 40 mismatches respectively. BWA was run with the option -o 0 to disable gapped alignment. Alignment time for BWA is the sum of the times for BWA aln and BWA samse to run while alignment time for SRmapper is the time for SRmapper align to run.

which reads were being aligned, it was determined that in every case, the more sensitive algorithm aligned greater than 99% of the reads aligned by the less sensitive algorithm while aligning a small fraction of additional reads not found by the less sensitive algorithm. Since BWA has been demonstrated to be highly accurate in its alignment and SRmapper always produced similar results to BWA in terms of output, it was qualitatively determined that SRmapper also accurately aligned reads to the reference. A further comparison of the relative accuracies of SRmapper and BWA will be provided subsequently. It is also worth noting that the alignment time for SRmapper was less sensitive to an increase in the number of mismatches than was the alignment time for BWA (**Fig 2.11**). This is due to SRmapper using direct comparison to establish alignments. Regardless of whether an alignment has few or many mismatches, SRmapper will perform a similar number of comparisons. BWA, however, is structured in a way that it requires many more comparisons to find an alignment with a higher number of mismatches due to way searches are performed using BWT methodology. Due to the advantage SRmapper has in aligning reads with a higher number of mismatches, it would be much more suitable than BWA for performing alignment where the reference sequence is not extremely similar to the gene or genome being sequenced. Another way to interpret this data would be to say that if a user wanted to spend some fixed time performing alignment, he or she could obtain more alignments by using SRmapper than BWA as is apparent by comparing the alignment times and percent of reads aligned by comparing the default parameters of BWA against the default parameters of SRmapper (**Fig 2.12**). It is also worth that the additional alignments gathered by SRmapper contain a higher number of mismatches than are allowed by the default parameters of BWA and

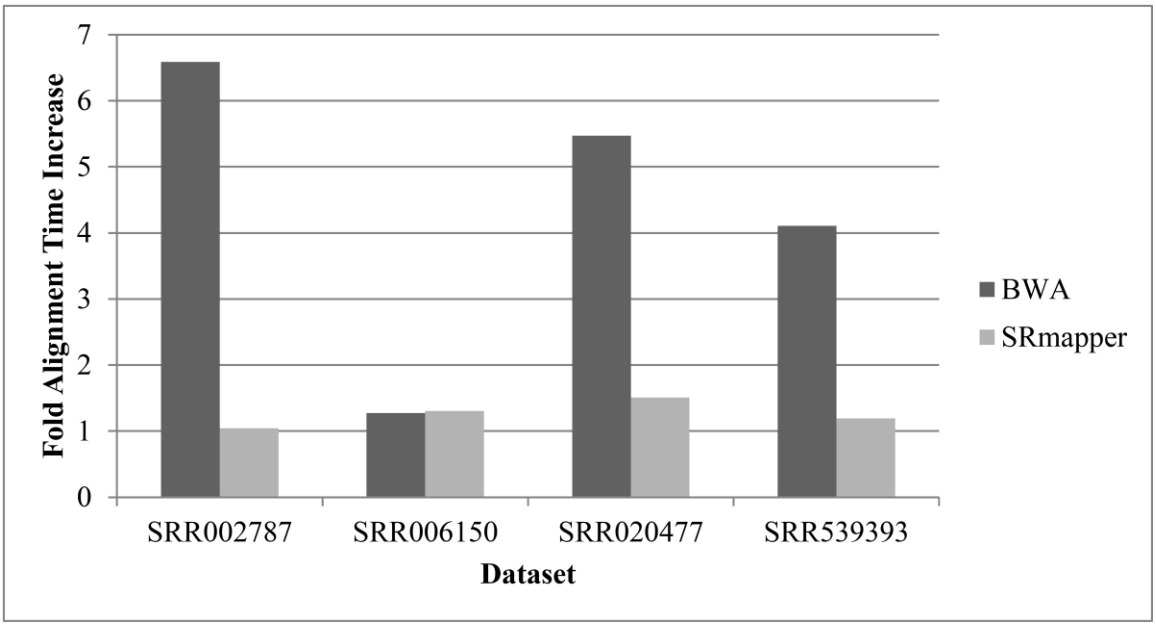


Figure 2.11: Fold Alignment Time Increase by Increasing Mismatches Allowed from BWA Default Parameters to SRmapper Default Parameters. The relative increases in alignment time are shown for BWA and SRmapper as the number of allowed mismatches increased from the default parameters used by BWA to the default Parameters used by SRmapper. Fold alignment time increase is calculated as (algorithm alignment time using SRmapper default mismatch number / algorithm alignment time using BWA default mismatch number).

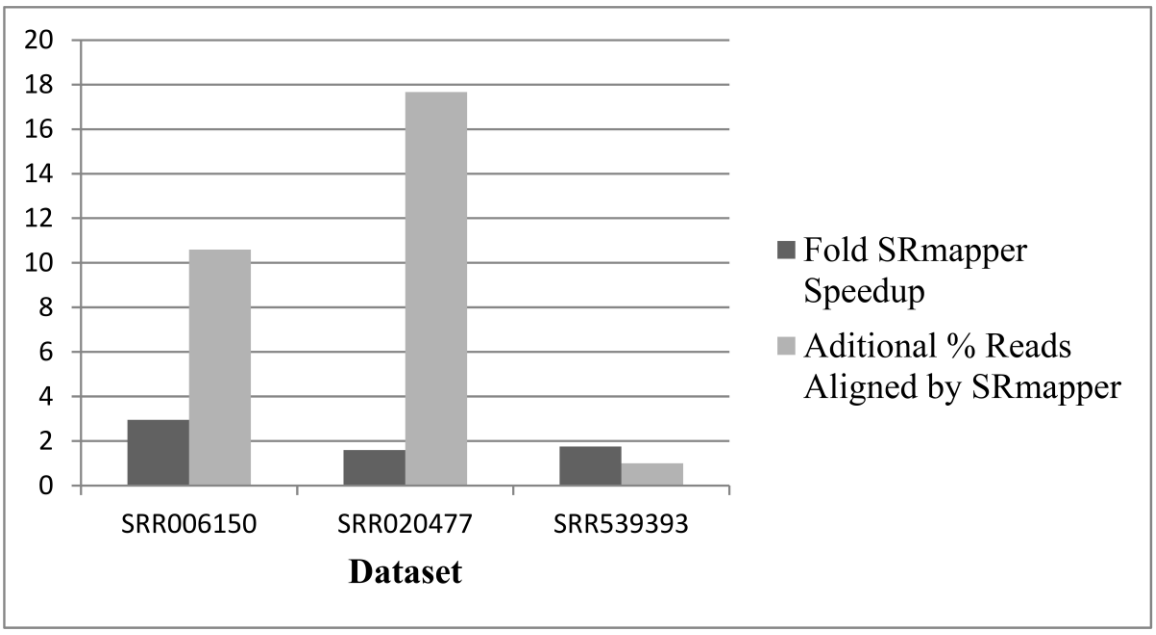


Figure 2.12: Comparison of BWA and SRmapper Alignment Performance with Each Algorithm Using Its Default Parameters. When both algorithms are used with default parameters, SRmapper retains a speed advantage over BWA while aligning a higher fraction of reads. Additional reads aligned by SRmapper are mostly those with a higher number of mismatches between the read and reference. In theory, these alignments with higher numbers of mismatches would produce more useful data in studying variation between two DNA sequences.

that alignments with discrepancies are more useful for studying genetic variation than alignments with no discrepancies. Thus, by using SRmapper, a user could either find more variant positions while still spending a somewhat shorter time aligning sequences than would be required by using BWA, or a user could find a similar number of variant positions in a much shorter time by using SRmapper instead of BWA.

In pair-end alignment, SRmapper retained the same speed advantage of 2x to 8x compared to BWA depending on the dataset being evaluated and the parameters used (**Table 2.3**, **Table 2.4**). In the case of pair-end alignment, BWA has an improved sensitivity compared to single-end alignment. Enabling Smith-Waterman alignment can further improve BWA's sensitivity at the cost of alignment speed, but was not used so as to produce a comparison with as similar settings as possible (Smith and Waterman, 1981). However, even with the improvement in sensitivity that BWA experienced, both algorithms had similar sensitivities on two of the three datasets evaluated. In terms of sensitivity, BWA fared better than SRmapper on the dataset with the shortest reads, but SRmapper had a higher speedup increase when comparing single-end alignment to pair-end alignment (3.85x vs 4.30x and 3.76x vs 4.17x).

It is also worth noting that in order for BWA to perform pair-end alignment, a higher amount of memory is required than in single-end alignment. In fact, this increase in memory usage requires pair-end alignment by BWA be performed on a system with more than 4GB of total memory making SRmapper an attractive choice for those who do not have access to higher memory machines but still wish to perform pair-end alignment. It is also worth noting that as sequencer technology has improved, the length of reads has generally increased. The relative number of experiments generating reads with lengths

	Alignment Time(s)	SRmapper Speedup	% Reads Aligned
SRR006150			
BWA	11,644		64.60%
SRmapper	2,706	4.30x	59.44%
SRR020477			
BWA	1,547		55.76%
SRmapper	676	2.29x	54.90%
SRR539393			
BWA	2,795		84.18%
SRmapper	1,317	2.12x	82.72%

Table 2.3: Comparison of Alignment Time and Percent Reads Aligned for BWA and SRmapper using the default mismatch Parameters of BWA and Pair-End Alignment. The alignment time and percent reads aligned for three datasets were measured using BWA and SRmapper. The datasets were SRR006150, SRR020477, and SRR539393 and contained 13.14M pair-end reads of 51 bp on each end, 2.04M pair-end reads of 76 bp on each end, and 2.25M pair-end reads of 100 bp on each end respectively. The number of mismatches allowed for the above datasets were 3, 4, and 5 mismatches respectively. BWA was run with the option -o 0 to disable gapped alignment. Alignment time for BWA is the sum of the times for BWA aln and BWA sampe to run while alignment time for SRmapper is the time for SRmapper align to run.

	Alignment Time(s)	SRmapper Speedup	% Reads Aligned
SRR006150			
BWA	14,794		73.34%
SRmapper	3,552	4.17x	67.25%
SRR020477			
BWA	8,192		68.43%
SRmapper	973	8.42x	71.53%
SRR539393			
BWA	11,248		88.56%
SRmapper	1,542	7.29x	89.29%

Table 2.4: Comparison of Alignment Time and Percent Reads Aligned for BWA and SRmapper using the default mismatch Parameters of SRmapper and Pair-End Alignment. The alignment time and percent reads aligned for three datasets were measured using BWA and SRmapper. The datasets were SRR006150, SRR020477, and SRR539393 and contained 13.14M pair-end reads of 51 bp on each end, 2.04M pair-end reads of 76 bp on each end, and 2.25M pair-end reads of 100 bp on each end respectively. The number of mismatches allowed for the above datasets were 12, 28, and 40 mismatches respectively. BWA was run with the option -o 0 to disable gapped alignment. Alignment time for BWA is the sum of the times for BWA aln and BWA sampe to run while alignment time for SRmapper is the time for SRmapper align to run.

between 30-50 bp has drastically decreased. In contrast, most sequencing experiments performed using Illumina instruments now produce reads of lengths either 100 bp or 150 bp. Thus, the conditions under which SRmapper performed less favorably than BWA in terms of sensitivity are becoming less prevalent while the conditions under which SRmapper had equal or greater sensitivity are becoming more prevalent.

2.4.2.4 Creation of Simulated Reads and Determination of Aligner Accuracy

To evaluate the accuracy of SRmapper in comparison to BWA, software was developed that simulated the sequencing of reference sequences or genomes. This software allowed for the simulations of reads with known numbers of mismatches, sequencing errors, and insertions or deletions as well as the exact position on the reference sequence from which these reads originated from. This software allowed for a tunable number of simulated reads with a length specified by the user to be created and also allowed the user to specify the rate of mismatches, insertions and deletions, their sizes, and sequencing errors. Reads were simulated from random positions in the genome such that every time a simulated sequencing was performed, a different set of reads would be created. These reads were stored in .fastq format, and the name given to each read in the first line of its .fastq expression was the reference sequence and reference sequence location from which it came.

A second piece of software was created to analyze the results of the alignment of these simulated reads. This software scanned through the SAM files created by either SRmapper or BWA to determine if reads had been aligned to the correct position within the reference sequence. Since the exact position of where the reads originated from was

known, it was possible for this software to determine whether the alignment generated was to the correct position within the reference. Specifically, since the first line in the SAM output format is the sequence name from the first line of the .fastq expression for a read, the alignment can be checked for accuracy by comparing the first field of the SAM formatted alignment for a read with the third and fourth field to determine whether the read aligned to the correct reference sequence and location within that sequence. Finally, this software was designed to only measure the accuracy of confidently aligned reads since these reads are the ones used in downstream analysis. For BWA, this meant checking whether a read was aligned and confidently aligned, and for SRmapper, this meant checking whether a read was confidently aligned.

Additionally, an additional software package made available by the creators of BWA was utilized to generate receiving operating characteristics (ROC) curves. ROC curves can be used to graph the relationship between sensitivity and selectivity where selectivity is the ability to correctly map alignments and provide insight into how changes in the sensitivity of an alignment algorithm affects its selectivity. This tool, wgsim, was downloaded from https://github.com/lh3/wgsim.

2.4.2.5 Results of Comparing SRmapper to BWA to Determine Alignment Accuracy by Using Simulated Reads

Simulations were performed by creating 100,000 reads of length 50 bp or 100 bp using an sequencing error rate of 1.5%, a SNP rate of 0.09%, and a indel rate of 0.01%. Again, the full human genome was chosen as the reference sequence. These parameters were chosen since they were used in the original evaluation of the accuracy of BWA and were chosen in an attempt to create as realistic of a simulation as is possible. As with

evaluations to compare speed and sensitivity, accuracy evaluations were carried out using default mismatch settings for both BWA and SRmapper, and as above, gapped alignment was disabled for BWA. For all of the evaluations performed, both alignment tools aligned the vast majority of the reads correctly, but BWA did show a higher accuracy across all tests (**Table 2.5**, **Table 2.6**). However, as read length increased, the incorrect placement of confidently aligned reads decreased for both algorithms. Even though SRmapper had a somewhat lower accuracy than BWA, its error rate was only 1 in 250 for 100 bp reads that were confidently aligned for reads of 100 bp in length using the default mismatch parameters of BWA. Evaluations using wgsim to create ROC curves on pair-end alignments that used the human genome as the reference sequence similarly revealed that BWA had a somewhat higher selectivity than SRmapper (**Fig 2.13**). It also showed that a large portion of the incorrect alignments generated by SRmapper occurred when there were no mismatches or very few mismatches. However, as the number of mismatches that were allowed increased, the selectivity difference between SRmapper and BWA drastically decreased.

A manual inspection of incorrectly aligned reads revealed two major sources of error for reads being incorrectly placed. The first occurred when sequencer errors or SNPs resulted in reads being created that aligned to the incorrect location with fewer mismatches than the location from which the read originated. This resulted in alignments with a higher phred score being generated for the incorrect alignment position. This source of error affected both SRmapper and BWA. The issue of reads being more similar to an alternative location within the reference than the portion of the reference it originated from is more of a weakness of the entire reference-based assembly method

Simulated Reads	% Reads Aligned	%Correct Alignments
50 bp reads		
BWA	83.55%	99.62%
SRmapper	81.83%	98.79%
100 bp reads		
BWA	87.30%	99.90%
SRmapper	88.07%	99.58%

Table 2.5: Alignment Accuracy of BWA and SRmapper as Determined By the Alignment of Simulated Reads Using the Default Mismatch Settings of BWA. To measure the accuracy of BWA and SRmapper, 100,000 simulated reads were created using in-house developed software. Reads were simulated from the human genome with a 1.5% sequencer error rate, a 0.09% SNP rate, and a 0.01% indel rate with indels ranging in length from 1 to 5 bases. For the 50 bp reads, 3 mismatches were allowed; for the 100 bp reads, 5 mismatches were allowed. The %Reads Aligned was calculated as (reads confidently aligned / total reads). The %Correct Alignments was calculated as (confident reads correctly aligned / reads confidently aligned).

Simulated Reads	% Reads Aligned	%Correct Alignments
50 bp reads		
BWA	83.96%	99.58%
SRmapper	82.30%	98.32%
100 bp reads		
BWA	87.88%	99.82%
SRmapper	89.06%	99.19%

Table 2.6: Alignment Accuracy of BWA and SRmapper as Determined By the Alignment of Simulated Reads Using the Default Mismatch Settings of SRmapper.

To measure the accuracy of BWA and SRmapper, 100,000 simulated reads were created using in-house developed software. Reads were simulated from the human genome with a 1.5% sequencer error rate, a 0.09% SNP rate, and a 0.01% indel rate with indels ranging in length from 1 to 5 bases. For the 50 bp reads, 28 mismatches were allowed; for the 100 bp reads, 40 mismatches were allowed. The %Reads Aligned was calculated as (reads confidently aligned / total reads). The %Correct Alignments was calculated as (confident reads correctly aligned / reads confidently aligned).

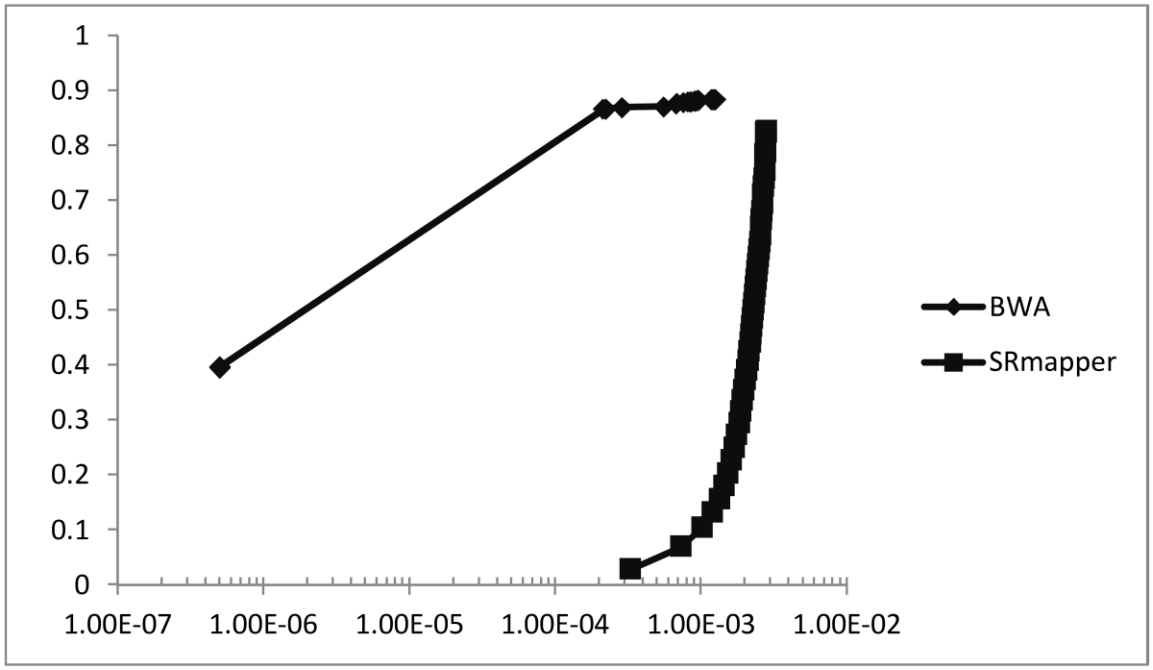


Figure 2.13: ROC Curves for BWA and SRmapper Alignments Using Wgsim to Simulate Reads and Build ROC Curves. Wgsim was used to simulate pair-end reads from the human genome using the suggested settings for simulating reads and accuracy (http://lh3lh3.users.sourceforge.net/alnROC.shtml). The abscissa denotes accuracy while the ordinate denotes fraction reads aligned. As the trace proceeds to the right, the number of mismatches in alignment increases. BWA was run with the option -o 0 in alignment and -A in SAM file creation. The simulation demonstrates that BWA has a higher accuracy than SRmapper but that as mismatches allowed increases, the difference in accuracies drastically decreases. Overall, BWA had an alignment rate of 90% with an error rate around 0.1% while SRmapper had an alignment rate of 85% with an error rate around 0.2%. (Adapted from Gontarz et al., 2013).

than a weakness in SRmapper or BWA. It also reflects a weakness in current sequencing technology since short reads occasionally do not contain enough information to properly determine their correct alignment position. Looking at the conditions chosen to simulate reads from the human genome and determining where mismatches between reads and the reference originate from, it is apparent that sequencing errors are the largest source of discrepancies between reads and the reference. As sequencing technology continues to improve, read lengths will continue to increase and error rates will continue to decrease. Thus, in the future, this first source of error will become even less of a factor than it currently is.

The second cause for reads being incorrectly aligned affected SRmapper more seriously than it affected BWA and was seen in reads that were incorrectly aligned by SRmapper even though they had zero mismatches or very few mismatches compared to the reference sequence. It was determined that SRmapper made incorrect confident alignments on a small fraction of reads had no or few mismatches in alignment due to the repetitive nature of the human genome. In a few cases, SRmapper found a potential alignment to a repetitive region in the first 100 entries searched in a bucket whose key was from the repetitive region but due to the limit of only searching the first 100 entries did not find other alignments, including the correct one. SRmapper, therefore, incorrectly reported a confident alignment. In contrast, BWA uses a different kind of index and alignment strategy that does not have this limitation. However, as the number of mismatches increases, the selectivity of SRmapper does not deviate nearly as much as the selectivity of BWA does. This likely reflects that SRmapper handles higher number of mismatches in alignment better than BWA does. The most plausible explanation is that

the seeding procedure used by BWA strictly limits the number of mismatches that can occur early in the alignment. Thus, when there are a higher number of discrepancies between the read and the reference, BWA is more likely to miss the correct alignment and report an incorrect, confident alignment. In contrast, SRmapper only requires that a stretch of D bases be found with no mismatches and therefore is not affected by reads with mismatches early in the sequence. Thus, the number of incorrectly aligned reads does not drastically increase with SRmapper as it does with BWA when a higher number of mismatches are permitted.

Finally, it can be noted that the ROC curves demonstrate that the maximum number of permitted mismatches determined by the probability function that SRmapper employs does not result in a large number of incorrect alignments. Were the permissible number of mismatches set too high, the ROC curve would reflect a decrease in selectivity since many spurious alignments being generated would result in a high number of incorrect alignments. That this is not seen in the ROC curve for SRmapper validates that the probabilistic model being used works as intended since as the number of mismatches allowed increases, there is a steady increase in the number of alignments being found without a large increase in reads being incorrectly aligned.

Pseudocode 2.1: SRmapper Buildindex.

```
Open and check input files;
Open and check output files;
Set total Reference length=0;
For each reference sequence
. Determine reference length;
. Store reference name and length in .sqn.hdr file;
. Add reference length to total reference length;
Calculate index key length using eq. 1;
Store key length (D), number of keys, and total reference length in .sqn.hdr file;
For keys starting with A,C,G,T
. Set reference location to 0;
. For each reference sequence
. . While not at the end of the reference sequence
. . . Read D bases to form a key;
. . . If key starts with correct base
. . . . Hash the key;
. . . . If key has not been hashed before
. . . . . Create bucket;
. . . . . Store reference location in bucket;
. . . . Else
. . . . . While bucket for the key is full
. . . . . If pointer to next overflow bucket is not NULL
. . . . . Use pointer in bucket to move to next overflow bucket;
. . . . . Else
. . . . . Create new bucket;
. . . . . Set pointer in current bucket to new bucket;
. . . . . Use pointer in bucket to move to next overflow bucket;
. . . . . Store reference location in bucket;
. . . . . Increase reference location by D;
. Set processed keys to 0;
. Set locations written to 0;
. While processed keys is less than 4^(D-1)
. . If there is a bucket for processed keys
. . . While the bucket pointer is not NULL
. . . . While there are reference locations in bucket
. . . . . Print location to .sqn file;
. . . . . Increase locations printed by 1;
. . . . Move to the next bucket;
. . . While there are reference locations in the last bucket;
. . . . Print location to .sqn file;
. . . . Increase locations printed by 1;
. . Write locations printed to .sqn.val file;
. Write locations printed to .sqn.hdr file;
. Remove buckets from memory;
```

SRmapper Buildindex (cont.)

For each reference sequence
. While not at the end of the reference sequence
. . Read four bases;
. . Convert bases into 2-bit per base format;
. . Store binary value for four bases in .sqn.bfa file;
Close all files;
<end>

Pseudocode 2.2: SRmapper Align.

```
Read and parse usage options;
Open and check input files;
Load reference names and lengths from .sqn.hdr file;
Load key length, number of keys, number of entries from .sqn.hdr file;
For reads lengths from 'D' to max read lengths
.  Calculate phred score for 0 mismatches;
.  While phred score is equal to or greater than min phred score
.  .  Increase mismatches by 1;
.  .  Calculate phred score for current number of mismatches;
For 'Key Base'=A,C,G,T
.  Create temp file;
.  Load entries into memory;
.  Load locations counter into memory;
.  For each read in each .fastq file
.  .  Get a read;
.  .  Set Mm by checking phred table with read length;
.  .  Set 'key start' to 0;
.  .  While not at the end of read and not past D(Mm+2)
.  .  .  Form key from D bases starting with 'key start';
.  .  .  If first base in key is the same as 'Key Base'
.  .  .  .  Hash key;
.  .  .  .  Find possible alignments from index;
.  .  .  .  For each possible alignment
.  .  .  .  .  Align remaining bases by direct comparison;
.  .  .  .  .  If alignment mismatches is less than or equal to Mm
.  .  .  .  .  .  Set Mm to alignment mismatches;
.  .  .  .  .  .  If fewer alignments found with Mm mismatches than allowed
.  .  .  .  .  .  .  If alignment location not already found
.  .  .  .  .  .  .  .  Store alignment in temp file;
.  .  .  .  .  .  Else
.  .  .  .  .  .  .  Decrease Mm by 1;
.  .  .  Increase 'key start' by 1;
.  .  Reverse complement the read;
.  .  Set 'key start' to 0;
.  .  While not at the end of read and not past D(Mm+2)
.  .  .  Form key from D bases starting with 'key start';
.  .  .  If first base in key is the same as 'Key Base'
.  .  .  .  Hash key;
.  .  .  .  Find possible alignments from index;
.  .  .  .  For each possible alignment
.  .  .  .  .  Align remaining bases by direct comparison;
.  .  .  .  .  If alignment mismatches is less than or equal to Mm
.  .  .  .  .  .  Set Mm to alignment mismatches;
.  .  .  .  .  .  If fewer alignments found with Mm mismatches than allowed
```

SRmapper Align (cont.)

. . . . If alignment location not already found
. . . . Store alignment in temp file;
. . . . . . Else
. . . . Decrease M_m by 1;
. . . Increase 'key start' by 1;
. Remove index from memory;
. Close temp file;
If performing single-end alignment
. Open all temp files;
. For each aligned read
. . Select alignment(s) with fewest mismatches from temp files;
. . Print alignment in SAM format;
If performing pair-end alignment
. Open all temp files;
. For each aligned pair
. . For each possible alignment for the first mate in pair
. . . For each possible alignment for the second mate in pair
. . . . Determine if the two alignments for a proper pair;
. . . . . If a proper pair is formed
. . . . . If no proper pair stored in memory
. . . . Store proper pair in memory
. . . . . Else
. . . . If proper pair is a better alignment than stored pair
. . . . Store as best pair;
. . Print best pair in SAM format;
Close all temp files;
Delete all temp files;
Close SAM file;
<end>

Chapter III

Development of Detection Methodology for *Mycobacterium Tuberculosis* Using SRmapper and Uniqueness Genomes

3.1 Background

The initial goal of this project was to develop and test a method to detect TB in metagenomic NGS samples isolated from saliva using SRmapper. The initial concept for detection of TB was to filter out the portions of the TB genome which were similar to sequences from other genomes that could be generated in an NGS experiment. The devised concept to filter out the portions of the TB genome which were similar to other genomes was to simulate every possible read that could be created in an NGS experiment involving the oral metagenome and align these reads to the TB genome. This would, in theory, determine every region of the TB genome similar to other genomes, and these regions would be removed from the TB genome thereby leaving the unique portions of the TB genome also referenced as the uniqueness TB genome. To simulate every possible read, all the bacterial genomes from the oral metagenome were downloaded and 100 bp reads were created at every position in the genome. Reads from the human reference genome were simulated in the same manner. Additionally, 46 trillion nucleotides of DNA from the 1000 Genomes Project were aligned to the TB genome to ensure that human variation did not preclude the use of this method. In the process of performing these alignments, it was determined that several samples From the Finnish HapMap portion of the 1000 genomes project were contaminated with TB DNA. Final creation of the uniqueness TB genome resulted in 36% of the TB genome being filtered out. Real and simulated metagenomic datasets were used to demonstrate the usefulness of the uniqueness TB genome in terms of increasing the sensitivity of NGS as means to detect TB and increasing the selectivity of NGS by greatly reducing the rate of false positive alignments.

3.2 Materials and Methods

3.2.1 Computational Resources

Jobs requiring large amounts of computational resources were run on the Lewis cluster at the University of Missouri Bioinformatics Consortium (http://umbc.rnet.missouri.edu/) (UMBC). Jobs demanding internet access, such as downloading of sequences, were run from the head node. All other jobs were run using the Load Sharing Facility (LSF) employed by Lewis by submitting batch jobs using the bsub command or through gocomp for interactive jobs. For using bsub on Lewis, the required usage format is "bsub -J job_name -e output.err -o output.out command [options]." The status of submitted jobs can be monitored using the bjobs command. At the time of use, the LSF allows a user to have up to 48 active processes and up to 44 more processes queued. Less demanding jobs were run locally on an Intel Xeon 2.8GB processor with 4GB of memory.

3.2.2 Genomic Sequences and NGS Sequences

Raw sequence files from the 1000 Genomes Project were downloaded from www.1000genomes.org/data using the aspera client (http://asperasoft.com/software/) for fast download (Altshuler et al., 2012). Oral microbiome data was downloaded from the Human Oral Microbiome Database (www.homd.org) (Dewhirst et al., 2010). Full bacterial genomes were downloaded from the National Center for Biotechnology Information (www.ncbi.nih.nlm.gov). Sequencer data was downloaded from the Sequence Read Archive (www.ncbi.nih.nlm.gov/sra). Human reference genome hg19 was used for generating simulated reads from the human genome.

3.2.3 Simulated Reads

For all experiments where reads were simulated from a genome, 100 base pair reads were created unless otherwise noted. In creating simulated reads to build uniqueness genomes, sequencing features, such as sequencer errors and genomic variance features such as SNPs and indels were not simulated into these reads. For simulated reads used for demonstrating the effectiveness of uniqueness genomes, 100 bp reads were simulated from random locations in the genome of interest using a 1.5% sequencer error rate, a 0.09% SNP rate and a 0.01% indel rate with indel lengths ranging from 1 to 5 nucleotides using in-house software. The number of reads simulated varied depending on simulated sequencing load, depth, and coverage and is described in the results section.

3.2.4 Alignment Settings and Conditions

SRmapper was used for all alignment tests. SRmapper has probability functions built into it that allow for a dynamic number of mismatches between reads and the reference genome to be allowed such that a minimum chance of alignment by random chance is always achieved. Alignment with SRmapper was performed with the –q 6 option to enforce all alignments to have a less than 1 in 10^6 chance of aligning by random chance (align <ref.sqn> { <in.fastq> } <out.sam> -q 6) for the creation of the uniqueness genomes and initial alignment of 1000 genomes data to the tuberculosis full genome and uniqueness genome. In tests where the results of alignment with SRmapper were verified by using another alignment program, BWA was used with its default parameters (bwa aln –f out.sai <in.fa> <in.fastq>, bwa samse –f out.sam <in.fa> <in.sai> <in.fastq>). For testing the effectiveness of the uniqueness genomes, SRmapper was used with either the –q 10 option to require a less than 1 in 10^{10} chance of alignment by random chance or with the –m option to set a specific maximum number of mismatches allowed between

the reference genome and short read sequence with either 0, 5, or 10 mismatches depending on the experiment.

3.2.5 Creation of Uniqueness Genomes

To create the uniqueness genome for tuberculosis, all available sequences from the oral metagenome were selected to be compared to the tuberculosis genome. Each genome within the metagenome of interest was fragmented into simulated reads as described above, and each read was aligned to the reference tuberculosis genome H37Rv as described above. Certain incompletely assembled genomes contained some contigs shorter than the 100 bp read length used for creating the uniqueness genome. These contigs were excluded from alignment to the tuberculosis genome. Simulated reads were created using software developed in house that created every possible read from a genome by scanning through the genome and creating a read starting at every position in the genome. After every simulated read from the oral metagenome had been aligned to the tuberculosis genome, all of the nucleotides in the TB genome to which no nucleotide from the metagenomic reads had aligned were retained. This was performed by developing software which first read through the SAM file and marked every nucleotide as either covered by reads from the oral metagenome or not covered by reads from the oral metagenome. This program then wrote which contiguous regions were covered or not covered. Nucleotides that had any read aligned to them were removed from the genome. The contiguous stretches of nucleotides without reads aligned them listed in the aforementioned file were used to form the uniqueness genome by another developed program that took the list of unique contiguous segments and the original TB genome and created a genome containing only the unique regions (**Fig 3.1**). Each of these regions was

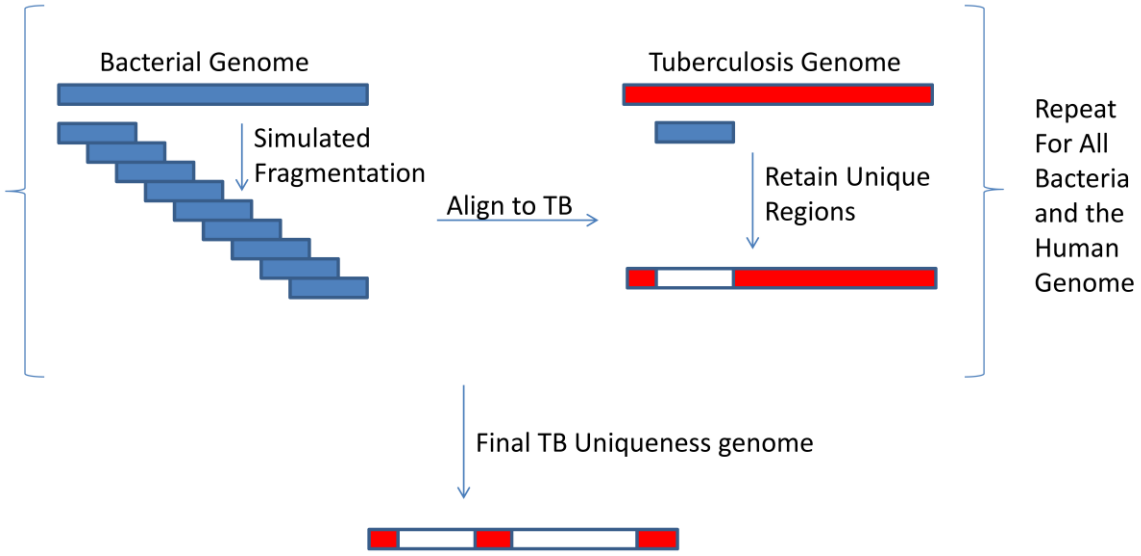


Figure 3.1: Formation of the Uniqueness Genome for TB. To form the uniqueness genome for tuberculosis, every bacterial sequence included in the oral metagenome as well as the human reference genome was fragmented to simulate every possible read that could be generated in an NGS experiment. Simulated reads were chosen to have a read length of 100 bp. The simulated reads from each bacterial genome were aligned to the TB genome (red), and the regions to which no reads aligned were retained while those to which any reads aligned (white) were discarded. After the unique portions of the TB genome were determined, they were each linked together by a sequence of 50 'N' nucleotides to form the final uniqueness genome for TB.

separated by a buffer of 50 ambiguous 'N' nucleotides to prevent alignments being generated spuriously across two different unique regions. The program also generated a partial uniqueness genome forming a genome from the 100 largest unique regions and padding each region with the buffer as described above.

3.2.6 Download and Processing of 1000 Genomes Data

To measure the effect of human variation on the coverage of the TB genome by human short read sequences, the base space reads from the 1000 Genomes Project were aligned to the tuberculosis genome using the standard –q 6 alignment criteria. At the time of download, this represented approximately 46 Tb of human genomic data. This analysis was performed using the Lewis cluster for three reasons due to the scale of the project. First, downloading approximately 46 Tb of human DNA involved downloading over 50 terabytes of compressed sequencing files in approximately 80,000 files. Download speeds on the Lewis servers routinely reached 50MB/s although there were times where download was much slower. In contrast, local download speeds at the University of Missouri, Saint Louis (UMSL) were approximately 5MB/s. Thus, downloading approximately 50TB of information required about 15 days of continuous download time on the UMBC servers versus 150 days of continuous download time at UMSL. Secondly, the amount of sequence data to be aligned to the tuberculosis genome precluded the use of local resources. SRmapper had an alignment rate of approximately 150Gb/day per processor used with additional time needed to interpret SAM files to determine what portions of the TB genome were covered by any of the data from the 1000 genomes project. The alignment of the 1000 Genomes Project data then required

more than 300 processor days with a nearly equal time to initially decompress the sequencing data flies and several weeks more of processor time to analyze the alignment data. Utilizing the Lewis cluster allowed for a peak usage of 48 processors simultaneously depending on availability and download rates. Finally, due to the size of the files being downloaded and processed, nearly a TB of disk space was required at any one time to store a portion of the sequencing data and alignment results even with the deletion of downloaded files and alignment files post-processing. BASH scripting was used to automatically download the 80,000 files, to direct submission to the LSF for decompression, alignment, and post-processing, to monitor LSF usage, and to remove downloaded files and alignment files after post-processing. Pseudocode for the script is provided at the end of the chapter as **1000 Genomes Project Analysis Pseudocode**.

3.2.7 Measuring Loads and Coverage

For alignment to a full genome, Load_F is measured as reads aligned / total reads. Removing portions of a genome to create a uniqueness genome results in the loss of the alignment of reads to the removed portions of that genome. Since reads are theoretically generated at random from locations in a genome, the proportion of aligned reads lost will be, on average, equivalent to the proportion of the genome being removed. Therefore, the load on a uniqueness genome was scaled by whatever proportion of the full genome was removed in creation of the uniqueness genome. This gave a scaled load as follows:

$$\text{Load}_U = \frac{\text{Nucleotides in Full Genome}}{\text{Nucleotides in Unique Genome}} \times \frac{\text{Reads Aligned}}{\text{Total Reads}} \qquad \text{Eq. 6}$$

This should not be confused with sample load which measures the fraction or percent of reads that come from the species of interest. Sample load was defined as:

$$Sample\ Load = \frac{Reads\ in\ Sample\ from\ Species\ of\ Interest}{Total\ Reads\ in\ Sample} \qquad \text{Eq. 7}$$

In all experiments, coverage of a reference genome was defined as follows:

$$\text{Coverage} = \frac{\text{Nucleotides Covered}}{\text{Total Nucleotides in Reference}} \quad \text{Eq. 8}$$

Finally, since different sequencing experiments often contain different sequencing depths, coverage was normalized by dividing the coverage of the full TB genome or the uniqueness TB genome by the number of nucleotides sequenced in an experiment and then scaling the result to produce a value that usually falls between 1 and 1000 as follows:

$$\frac{\%C}{B} = \frac{\%Coverage}{1\ Billion\ Nucleotides\ sequenced} = \frac{Coverage \times 10^9 \times 100\%}{Number\ of\ Nucleotides\ Sequenced} \quad \text{Eq. 9}$$

where $\frac{\%C}{B}$ is the unit the rate at which a genome is covered by nucleotides from a NGS experiment or simulated NGS experiment.

3.3 Results

3.3.1 Creation of the Oral Uniqueness Genomes for Tuberculosis

A diagnostic tool involving NGS for detecting the presence of TB in a sample or a TB infection would be fastest if it did not require time to separate other bacterial cells and human cells from tuberculosis and time to culture TB cells or amplify TB DNA before sequencing. Therefore, the effect of allowing every possible read from all known bacterial species in a NGS experiment from the oral metagenome as well as the human genome to be included in an oral sample was determined by simulation. This was performed by generating every possible read from the oral metagenome as well as every possible read from the human reference genome, build hg19. **Table 3.1** lists the bacteria chosen as the oral metagenome. This produced approximately 6 billion reads that were then aligned to the TB genome H37Rv. This alignment was performed allowing

Abiotrophia defectiva ATCC
Achromobacter xylosoxidans A8
Acinetobacter baumannii AB0057
Actinomyces cardiffensis F0333
Actinomyces georgiae F0490
Actinomyces graevenitzii C83
Actinomyces johnsonii F0330
Actinomyces massiliensis 4401292
Actinomyces massiliensis F0489
Actinomyces naeslundii MG1
Actinomyces odontolyticus ATCC
Actinomyces oris K20
Actinomyces sp-170
Actinomyces sp-171
Actinomyces sp-175
Actinomyces sp-178
Actinomyces sp-180
Actinomyces sp-181
Actinomyces sp-448
Actinomyces sp-848
Afipia broomeae ATCC
Aggregatibacter actinomycetemcomitans D17P-2
Aggregatibacter actinomycetemcomitans HK1651
Aggregatibacter aphrophilus NJ8700
Aggregatibacter segnis ATCC33393
Agrobacterium tumefaciens C58
Alloiococcus otitis ATCC
Alloscardovia omnicolens DSM
Anaerococcus lactolyticus ATCC
Anaerococcus prevotii DSM
Anaerococcus tetradius ATCC
Anaeroglobus geminatus F0357
Arcanobacterium haemolyticum DSM
Atopobium minutum 10063974
Atopobium parvulum DSM
Atopobium rimae ATCC
Atopobium sp-199
Atopobium vaginae DSM
Bacillus anthracis A0248
Bacillus clausii KSM-K16
Bacillus subtilis BSn5
Bacteroidetes bacterium sp-274
Bacteroidetes G-1 sp-272
Bifidobacterium animalis lactis
Bifidobacterium animalis lactis HN019
Bifidobacterium breve CECT
Bifidobacterium breve UCC2003
Bifidobacterium dentium ATCC
Bifidobacterium longum infantis
Bifidobacterium longum longum
Bordetella pertussis Tohama
Bradyrhizobium elkanii 587
Bradyrhizobium elkanii USDA
Brevundimonas diminuta 470-4
Brevundimonas diminuta ATCC
Bulleidia extructa ATCC
Bulleidia extructa W1219
Burkholderia cepacia GG4
Campylobacter concisus 13826
Campylobacter curvus 525.92
Campylobacter gracilis ATCC
Campylobacter gracilis RM3268
Campylobacter rectus ATCC
Campylobacter rectus RM3267
Campylobacter showae ATCC
Campylobacter showae CSUNSWCD
Campylobacter showae RM3277
Candidate TM7 TM7a
Candidate TM7 TM7b
Candidate TM7 TM7c
Capnocytophaga gingivalis ATCC
Capnocytophaga ochracea DSM
Capnocytophaga sp-324
Capnocytophaga sp-326
Capnocytophaga sp-329
Capnocytophaga sp-332
Capnocytophaga sp-335
Capnocytophaga sp-336
Capnocytophaga sp-338
Capnocytophaga sp-380
Capnocytophaga sp-412
Capnocytophaga sputigena ATCC
Cardiobacterium hominis ATCC
Cardiobacterium valvarum F0432
Catonella morbi ATCC
Centipeda periodontii DSM
Chlamydophila pneumoniae TW-183
Comamonas testosteroni KF-1
Corynebacterium diphtheriae NCTC
Corynebacterium durum F0235
Corynebacterium matruchotii ATCC
Corynebacterium urealyticum DSM
Cronobacter sakazakii ATCC
Cryptobacterium curtum DSM
Delftia acidovorans SPH-1
Desulfobulbus sp-041 DSB2
Desulfobulbus sp-041 DSB3
Dialister invisus DSM
Dialister micraerophilus DSM
Dolosigranulum pigrum ATCC
Eggerthella lenta DSM
Eikenella corrodens ATCC
Enterobacter cancerogenus ATCC
Enterobacter hormaechei ATCC
Enterococcus casseliflavus 14-MB-W-14
Enterococcus casseliflavus EC30
Enterococcus durans ATCC
Enterococcus durans FB129-CNAB-4
Enterococcus faecalis DSM
Enterococcus faecalis V583

Enterococcus italicus DSM
Enterococcus saccharolyticus 30 1
Erysipelothrix tonsillarum DSM
Escherichia coli BW2952
Escherichia coli O157
Eubacterium infirmum ATCC
Eubacterium infirmum F0142
Eubacterium limosum KIST612
Eubacterium saburreum DSM
Eubacterium saphenum ATCC
Eubacterium yurii margaretiae
Filifactor alocis ATCC
Finegoldia magna ATCC
Fusobacterium gonidiaformans ATCC
Fusobacterium necrophorum funduliforme
Fusobacterium nucleatum animalis F0419
Fusobacterium nucleatum animalis
Fusobacterium nucleatum nucleatum ATCC25586
Fusobacterium nucleatum polymorphum ATCC
Fusobacterium nucleatum vincentii ATCC
Fusobacterium periodonticum ATCC
Fusobacterium sp-370 F0437
Gardnerella vaginalis ATCC
Gemella haemolysans ATCC
Gemella haemolysans M341.
Gemella morbillorum M424.
Gemella sanguinis M325
Granulicatella adiacens ATCC
Granulicatella elegans ATCC
Haemophilus aegyptius ATCC
Haemophilus ducreyi 35000HP
Haemophilus haemolyticus M21639
Haemophilus influenzae PittGG
Haemophilus parainfluenzae ATCC
Haemophilus parainfluenzae T3T1
Haemophilus sp-851 F0397
Helicobacter pylori B38
Helicobacter pylori India7
Johnsonella ignava ATCC
Jonquetella anthropi DSM
Jonquetella anthropi E3
Kingella denitrificans ATCC
Kingella kingae ATCC
Kingella oralis ATCC
Klebsiella pneumoniae Kp342
Klebsiella pneumoniae NTUH-K2044
Kytococcus sedentarius DSM
Lachnospiraceae bacterium ACC2
Lachnospiraceae sp-082 F0431
Lachnospiraceae sp-107 F0167
Lactobacillus acidophilus NCFM
Lactobacillus brevis ATCC
Lactobacillus buchneri ATCC
Lactobacillus buchneri NRRL
Lactobacillus casei BL23
Lactobacillus catenaforme OT
Lactobacillus coleohominis 101-4
Lactobacillus crispatus ST1
Lactobacillus fermentum IFO
Lactobacillus gasseri ATCC
Lactobacillus iners DSM
Lactobacillus jensenii 1153
Lactobacillus johnsonii NCC
Lactobacillus kisonensis F0435
Lactobacillus oris PB013
Lactobacillus paracasei paracasei
Lactobacillus parafarraginis F0439
Lactobacillus pentosus KCA1
Lactobacillus plantarum plantarum-ATCC
Lactobacillus plantarum plantarum-ST3
Lactobacillus reuteri JCM
Lactobacillus rhamnosus GG
Lactobacillus salivarius UCC118
Lactobacillus vaginalis ATCC
Lactococcus lactis IL1403
Lactococcus lactis KF147
Lautropia mirabilis ATCC
Leptotrichia buccalis ATCC
Leptotrichia buccalis C-1013-b
Leptotrichia goodfellowii F0264
Leptotrichia hofstadii F0254
Leptotrichia shahii DSM
Leptotrichia wadei DSM
Listeria monocytogenes 08-5578
Listeria monocytogenes 4b
Lysinibacillus fusiformis ZC1
Megasphaera micronuciformis F0359
Mesorhizobium loti MAFF303099
Microbacterium sp-186 F0373
Mitsuokella multacida DSM
Mobiluncus mulieris ATCC
Moraxella catarrhalis RH4
Mycobacterium leprae Br4923
Mycobacterium tuberculosis CDC1551
Mycoplasma fermentans JER
Mycoplasma fermentans M64
Mycoplasma genitalium G-37
Mycoplasma hominis ATCC
Mycoplasma orale ATCC
Mycoplasma pneumoniae M129
Neisseria bacilliformis ATCC
Neisseria elongata ATCC
Neisseria flavescens NRL30031
Neisseria gonorrhoeae DGI2
Neisseria gonorrhoeae NCCP11945
Neisseria lactamica 020-06
Neisseria lactamica ATCC
Neisseria meningitidis ATCC
Neisseria meningitidis MC58
Neisseria mucosa ATCC
Neisseria polysaccharea ATCC

Neisseria sicca ATCC
Neisseria sp-014 F0314
Neisseria sp-020 F0370
Neisseria subflava NJ9703
Neisseria weaveri ATCC
Ochrobactrum anthropi ATCC
Ochrobactrum anthropi
Olsenella sp-809 F0356
Olsenella uli DSM
Oribacterium sinus F0268
Oribacterium sp-078 F0262
Oribacterium sp-108 F0425
Paenibacillus sp-786 D14
Parascardovia denticolens DSM
Parascardovia denticolens F0305
Parvimonas micra ATCC
Parvimonas sp-110 F0139
Parvimonas sp-393 F0440
Peptoniphilus indolicus ATCC
Peptoniphilus lacrimalis 315-B
Peptoniphilus lacrimalis DSM
Peptoniphilus sp-375 F0436
Peptoniphilus sp-386 F0131
Peptoniphilus sp-836 F0141
Peptostreptococcus anaerobius 653-L
Peptostreptococcus anaerobius DSM
Peptostreptococcus stomatis DSM
Porphyromonas asaccharolytica DSM
Porphyromonas asaccharolytica PR426713P-I
Porphyromonas catoniae F0037
Porphyromonas endodontalis ATCC
Porphyromonas gingivalis ATCC
Porphyromonas gingivalis W83
Porphyromonas sp-279 F0450
Porphyromonas uenonis 60-3
Prevotella bivia DSM
Prevotella bivia JCVIHMP010
Prevotella buccae D17
Prevotella buccalis ATCC
Prevotella dentalis DSM
Prevotella denticola F0289
Prevotella histicola F0411
Prevotella intermedia 17
Prevotella loescheii DSM
Prevotella maculosa DSM
Prevotella maculosa OT
Prevotella marshii DSM
Prevotella melaninogenica ATCC
Prevotella melaninogenica D18
Prevotella micans DSM
Prevotella micans F0438
Prevotella multiformis DSM
Prevotella multisaccharivorax DSM
Prevotella nigrescens ATCC
Prevotella oralis ATCC
Prevotella oris DSM
Prevotella oris F0302
Prevotella oulorum F0390
Prevotella pallens ATCC
Prevotella saccharolytica OT
Prevotella salivae DSM
Prevotella sp-299 F0039
Prevotella sp-302 F0020
Prevotella sp-302 F0323
Prevotella sp-306 F0472
Prevotella sp-317 F0108
Prevotella sp-472 F0295
Prevotella sp-473 F0040
Prevotella tannerae ATCC
Prevotella veroralis DSM
Prevotella veroralis F0319
Propionibacterium acnes SK137
Propionibacterium avidum 44067
Propionibacterium avidum ATCC
Propionibacterium propionicum F0230a
Propionibacterium sp-192 F0372
Proteus mirabilis ATCC
Proteus mirabilis HI4320
Pseudomonas aeruginosa LESB58
Pseudomonas fluorescens Pf0-1
Pseudomonas fluorescens Pf-5
Pseudomonas pseudoalcaligenes KF707
Pseudomonas stutzeri A1501
Pseudoramibacter alactolyticus ATCC
Pyramidobacter piscolens W5455
Pyramidobacter piscolens W5455-(JCVI)
Pyramidobacter piscolens W5455-(TFI)
Pyramidobacter piscolens W5455-(TFI-JCVI)
Ralstonia pickettii 12D
Rhodobacter capsulatus SB1003
Rothia aeria F0474
Rothia dentocariosa ATCC
Rothia mucilaginosa ATCC
Rothia mucilaginosa DY-18
Sanguibacter keddieii DSM
Scardovia inopinata F0304
Scardovia wiggsiae F0424
Selenomonas artemidis F0399
Selenomonas flueggei ATCC
Selenomonas infelix ATCC
Selenomonas noxia ATCC
Selenomonas sp-133 F0473
Selenomonas sp-137 F0430
Selenomonas sp-138
Selenomonas sp-149
Selenomonas sputigena ATCC
Shuttleworthia satelles DSM
Simonsiella muelleri ATCC
Slackia exigua ATCC
Solobacterium moorei F0204
Solobacterium moorei W5408
Staphylococcus aureus JH1

Staphylococcus aureus WW2703 97
Staphylococcus caprae C87
Staphylococcus epidermidis RP62A
Staphylococcus epidermidis W23144
Staphylococcus warneri L37603
Stenotrophomonas maltophilia K279a
Stenotrophomonas maltophilia R551-3
Streptococcus agalactiae NEM316
Streptococcus anginosus CCUG
Streptococcus anginosus SK52
Streptococcus australis ATCC
Streptococcus constellatus pharyngis
Streptococcus cristatus ATCC
Streptococcus downei F0415
Streptococcus gordonii Challis
Streptococcus infantarius infantarius-ATCC
Streptococcus infantis X
Streptococcus intermedius F0413
Streptococcus intermedius JTH08
Streptococcus mitis ATCC
Streptococcus mitis B6
Streptococcus mitis biovar-2 SK95
Streptococcus mitis NCTC
Streptococcus mutans UA-159
Streptococcus oligofermentans AS
Streptococcus oralis ATCC
Streptococcus oralis Uo5
Streptococcus parasanguinis ATCC
Streptococcus parasanguinis II-F0405
Streptococcus peroris ATCC
Streptococcus pneumoniae AP200
Streptococcus pyogenes MGAS10270
Streptococcus pyogenes NZ131
Streptococcus salivarius JIM8780
Streptococcus salivarius SK126
Streptococcus sanguinis SK36
Streptococcus sobrinus TCI-107
Streptococcus sp-056 F0418
Streptococcus sp-058 F0407
Streptococcus sp-066 F0442
Streptococcus sp-070 F0441
Streptococcus sp-071 F0408
Streptococcus vestibularis F0396
Synergistetes sp. SGP1
Tannerella forsythia ATCC
Treponema denticola ATCC
Treponema lecithinolyticum OMZ
Treponema maltophilum ATCC
Treponema medium ATCC
Treponema pallidum Nichols
Treponema socranskii ATCC
Treponema vincentii ATCC
Turicella otitidis ATCC
Variovorax paradoxus S110
Veillonella atypica ACS
Veillonella dispar ATCC
Veillonella parvula ATCC
Veillonella parvula DSM
Veillonella sp-158 F0412
Veillonella sp-780 F0422
Yersinia pestis Antiqua
Yersinia pestis KIM-D27

Table 3.1: The Genomes Chosen to Form the Oral Metagenome. To form the oral metagenome used to create the uniqueness genome for TB, 395 bacterial genomes were downloaded from HOMD. The genus, species, and subspecies names (if applicable) are listed.

discrepancies between the reads and the reference tuberculosis genome as described in the methods by using the -q 6 option in SRmapper.

Allowing discrepancies had a twofold purpose. First, it attempted to compensate for variations which exist within members of a species or between different strains of bacteria. For example, a human whose DNA is slightly different than the human reference genome at some loci may have produce alignments that would not have been found when alignment of simulated reads from the human reference genome was performed allowing no discrepancies. Secondly, allowing discrepancies attempted to compensate for artificial variation that is introduced by the sequencing process. Although sequencing instruments have become more accurate, current NGS instruments still have a non-negligible error rate. This especially applies to Illumina and IonTorrent instruments – the two NGS platforms most envisioned for using this method with SRmapper performing as the alignment algorithm. After the alignment of all possible reads from the human genome to the tuberculosis genome H37Rv, it was determined that the simulated reads from the human genome covered 1.83% of the reference tuberculosis genome (**Table 3.2**). After aligning all the reads from the oral microbiome, approximately 35.5% of the tuberculosis genome was covered by bacterial DNA. The remaining nucleotides formed nearly 5,900 fragments, the largest of those being approximately 11,500 bp in length and covering the proteins PPE5 and PPE6. The 100 largest fragments were used to form a partial uniqueness genome and contain a total of 554,025 nucleotides. When the same process was repeated for the lung microbiome, 39.7% of the tuberculosis genome was covered with the majority of the coverage coming from three species: *Nocardia farcinica*, *Segniliparus rotundus*, and *Segniliparus rugosus*.

Chromosome Name	Chromosomal coverage	Chromosome Name	Chromosomal coverage
1	0.35%	14	0.21%
2	0.34%	15	0.16%
3	0.19%	16	0.26%
4	0.21%	17	0.31%
5	0.20%	18	0.13%
6	0.24%	19	0.40%
7	0.24%	20	0.26%
8	0.20%	21	0.06%
9	0.26%	22	0.10%
10	0.19%	X	0.23%
11	0.19%	Y	0.00%
12	0.18%	M	0.00%
13	0.21%	Cumulative	1.83%

Table 3.2: Coverage of the TB Genome H37Rv by the Human Reference Genome.

To measure the similarity of the TB to the human genome in terms of an NGS experiment, every possible 100 bp read from the human reference genome hg19 was simulated and aligned to the TB genome. The coverage of each individual chromosome was determined as well as the cumulative coverage by all of the chromosomes combined.

Since nearly all of available metagenomic sequencing data was for saliva samples at the time these studies were being performed, these studies focused primarily exclusively on the oral metagenome.

3.3.2 Human Variation Does not Prevent the Use of NGS for Detecting TB

Initial alignment of the 80,000 files containing base space reads resulted in over 92.1% of the tuberculosis genome covered by reads from the 1000 genomes project. However, the vast majority of the coverage of the tuberculosis genome came from 25 samples from the same project (Finnish HapMap project) (**Table 3.3**). In fact, there were only 19 runs that showed more coverage of the TB genome than aligning all reads from the human reference genome (greater than 1.83%). The rest of the data appeared to be consistent with the data gathered from aligning simulated reads from the human reference genome to the TB genome. Although some coverage was expected from the 1000 genomes project data, the amount of coverage was expected to be near to the coverage from aligning the human reference or lower since many of the sequencing studies in the 1000 genomes project are low coverage sequencings. Taken together, this data suggests that a small number of runs were either performed by sampling subjects who were infected with tuberculosis or that there was some other contamination of samples with tuberculosis.

To test this hypothesis, all of the reads from the 1000 genomes project that initially aligned to the TB genome were converted back into .fastq format and aligned to the human reference genome. These reads were split into two categories - those which did align to the human reference genome and those which did not align to the human reference genome. **Figure 3.2** shows the overall workflow from the 1000 Genomes

Percent TB Genome Covered	Sequencing Run	Project	Percent TB Genome Covered	Sequencing Run	Project
28.58%	**ERR016001_1**	FIN	0.85%	ERR015870_2	FIN
25.31%	**ERR016001_2**	FIN	0.79%	ERR015870_1	FIN
22.80%	**ERR015874_2**	FIN	0.77%	ERR013123_1	FIN
22.50%	**ERR018501_1**	FIN	0.77%	ERR015876_2	FIN
22.30%	**ERR015874_1**	FIN	0.71%	ERR013121_2	FIN
19.18%	**ERR015732_1**	FIN	0.70%	ERR015876_1	FIN
19.12%	**ERR018501_2**	FIN	0.66%	ERR013123_2	FIN
15.73%	**ERR013120_1**	FIN	0.63%	SRR006204	PP3
14.05%	**ERR018499_1**	FIN	0.48%	SRR006203	PP3
13.11%	**ERR015872_2**	FIN	0.24%	ERR013122_1	FIN
12.44%	**ERR015732_2**	FIN	0.21%	SRR017041_2	YRI
12.36%	**ERR015872_1**	FIN	0.21%	ERR013122_2	FIN
12.04%	**ERR013120_2**	FIN	0.20%	SRR017041_1	YRI
11.68%	**ERR018499_2**	FIN	0.19%	SRR017034_1	YRI
8.11%	ERR013119_1	FIN	0.19%	SRR023308_1	JPT
5.76%	ERR013119_2	FIN	0.19%	ERR018560_1	GBR
4.25%	ERR016346_1	FIN	0.18%	SRR017034_2	YRI
3.89%	ERR016346_2	FIN	0.18%	ERR006241_1	CEU
2.10%	ERR015479_1	FIN	0.18%	ERR018560_2	GBR
1.57%	ERR015875_2	FIN	0.17%	ERR022461_1	CLM
1.56%	ERR018504_1	FIN	0.17%	SRR017040_2	YRI
1.47%	ERR015875_1	FIN	0.17%	SRR017040_1	YRI
1.45%	ERR015479_2	FIN	0.17%	ERR019495_1	FIN
1.02%	ERR018504_2	FIN	0.16%	ERR006241_2	CEU
1.01%	ERR013121_1	FIN	0.16%	ERR022461_2	CEU

Table 3.3: Datasets from the 1000 Genome Project with the Highest Coverage of the TB Genome. Of the roughly 80,000 reads files downloaded from the 1000 genomes project, the 50 with the highest coverage of the tuberculosis genome are recorded. Of these, only 25 result in coverage higher than 1%, and only 19 result in a higher coverage of the TB genome than when all possible reads from the human reference genome were aligned to the TB genome. Boldfaced datasets were used later to confirm the presence of TB. All datasets originated from the same project, the Finnish HapMap project. Project

abbreviations are as follows: Fin: Finish HapMap population; PP3: Pilot Project 3; YRI: Yoruba HapMap population; JPT: Japanese HapMap population; GBR: England and Scotland HapMap population; CEU: Utah residents with ancestry from Northern and Western Europe; CLM: Columbian HapMap population.

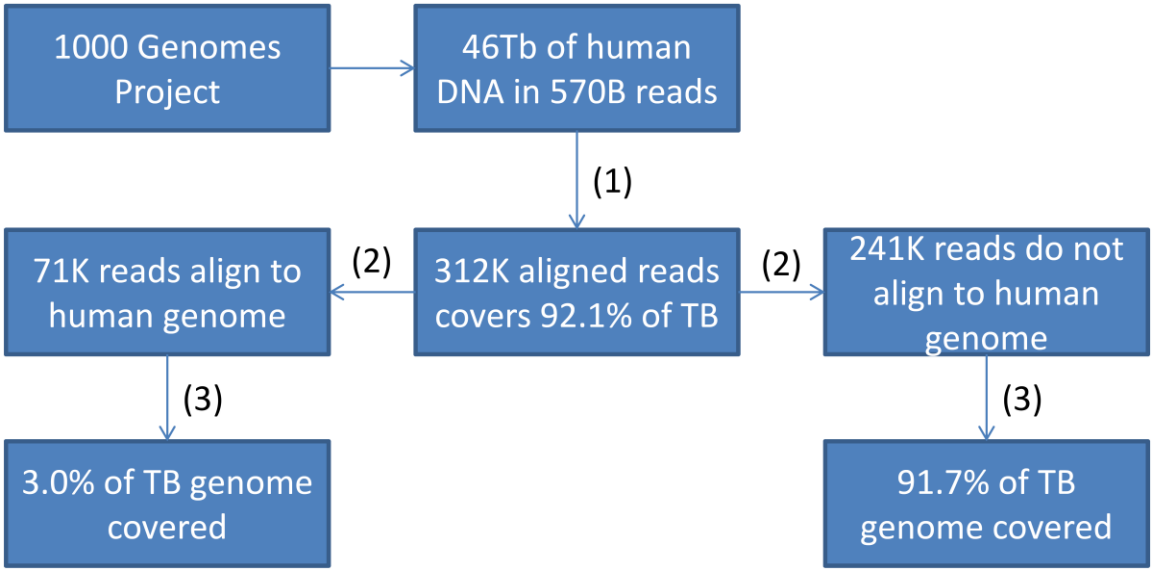


Figure 3.2: Workflow of the Analysis of the 1000 Genomes Data. From the 1000 genomes project, approximately 46Tb of human DNA comprising approximately 570 billion short sequences was analyzed by first aligning them to the full tuberculosis genome (1). Upon determining that the approximately 312,000 reads that aligned to the TB genome covered over 92% of the genome, an attempt was made to align these reads to the human reference genome to confirm their human origin (2). These reads were split into two categories based on whether they aligned to the human reference genome or not. Roughly three-quarters of the reads that were supposedly of human origin could not be aligned to the human genome. Reads from each category were again aligned to the TB genome (3). Those reads which did align to the human reference genome only covered 3.0% percent of the TB genome while the reads which could be aligned to the human reference genome covered 91.7% of the TB genome.

Project data. Approximately three quarters of the reads initially aligned to the TB genome could not be aligned to the human genome by SRmapper. Since only a quarter of supposedly human reads that aligned to the TB genome aligned to the human reference genome, verification that the alignment software (SRmapper) was correctly functioning was performed by using a second alignment tool (BWA). Since the reads were of different length because they came from different files and sequencers, BWA was not told what number of mismatches to allow. Instead, it was run using its default parameters to allow the number of discrepancies the algorithm deemed appropriate. Alignment to the human reference genome of the 1000 genomes project reads that aligned to the TB genome by BWA yielded similar results to those gathered using SRmapper. BWA aligned slightly fewer reads to both the human reference genome and the TB genome with some of the reads aligned by SRmapper being unaligned. These results were as expected since by default BWA does not tolerate as many differences between the reference and read as SRmapper does.

The approximately 70,000 reads that did align to the human genome were again aligned to the TB genome to measure their coverage, and it was determined that they covered only 3.0% of the TB genome. The higher coverage from the reads in the 1000 genome project compared to the coverage by the reference human genome (3.0% versus 1.83%) was attributed to two factors: the variation in human genetic makeup from samples in the 1000 genomes project and the variation of read length in studies from the 1000 genomes project. For the latter factor, it is easily conceivable that reads are more likely to have similarity to multiple genomes over a shorter stretch of sequence than a longer one. Thus, the shorter reads from the human genome project (some as short as 30

bp) have a higher likelihood to be similar to a location within the TB reference genome than long reads would. To verify this, the reads were separated by length, and the general pattern that was observed was that as read length increased, alignment rate decreased. Using the $\frac{\%C}{B}$ notation to normalize the amount of data in each set of reads, it was determined that reads <50 bp in length had an average coverage rate of $2.4 \times 10^{-2} \frac{\%C}{B}$ (percent genome coverage per billion nucleotides sequenced); reads 50 bp-74 bp in length had an average coverage rate of $1.8 \times 10^{-3} \frac{\%C}{B}$; reads 75 bp-99 bp in length had a raw average coverage rate of $6.1 \times 10^{-3} \frac{\%C}{B}$; and reads ≥150 bp in length had an average coverage rate of $3.2 \times 10^{-4} \frac{\%C}{B}$ (**Fig 3.3**). Reads from the 100 bp-149 bp length range contained the Data from the Finnish HapMap project which skewed the average coverage rate to $1.1 \times 10^{-2} \frac{\%C}{B}$. Removal of the 44 samples with the highest coverage (all from the Finnish HapMap project) reduced the average rate to $2.1 \times 10^{-3} \frac{\%C}{B}$, which, as expected, falls between the rates for reads of 75 bp-99 bp and reads ≥150 bp.

The approximately 280,000 reads which did not align to the human were again aligned to the reference TB genome using SRmapper and covered 91.7% of the TB genome. This analysis was repeated using BWA and yielded similar results (data not shown). This data suggests the possibility that certain samples were contaminated with tuberculosis DNA (**Table 3.3**) and that some were possibly contaminated with other bacteria. Although it was not feasible to experimentally verify that these samples were contaminated with TB, further computational evidence is provided later to strengthen this claim.

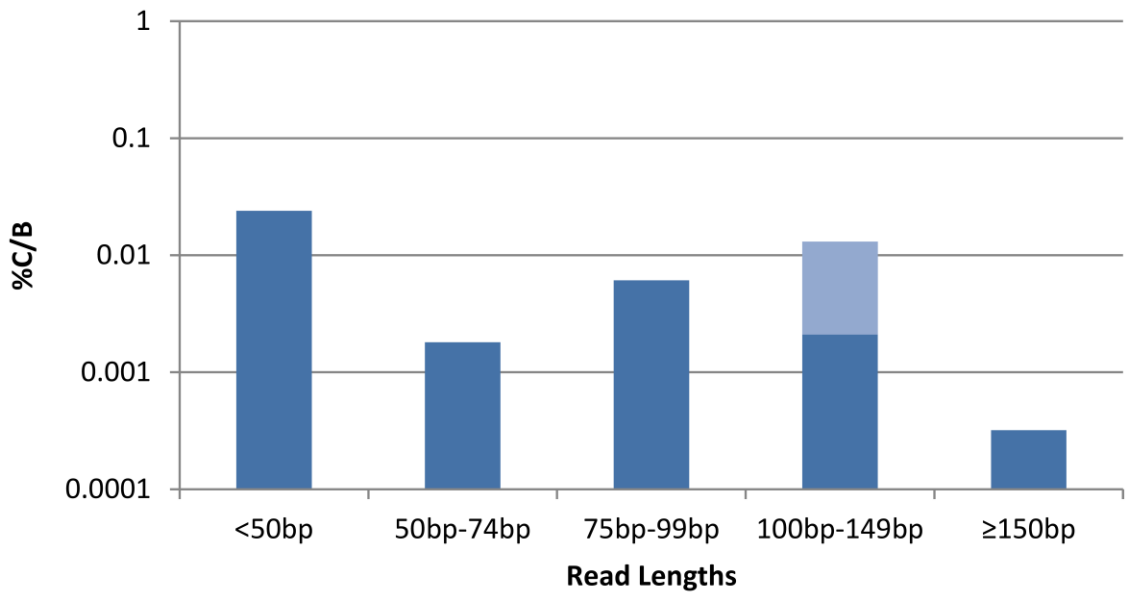


Figure 3.3: The Effect of Read Length on the Coverage Rate of Sequences from the 1000 Genomes Project Aligning to the TB Genome. The coverage rates, $\frac{\%C}{B}$, for reads of different lengths was determined. The reads in the 100 bp-149 bp grouping contained the data from the Finnish HapMap project. The light blue column represents the coverage rate from all data from the 100 bp-149 bp group, and the dark blue column represents the coverage rate if the samples with unusually high coverage from the Finnish HapMap project are filtered out.

If the assumption were accepted that the reads from the 1000 genome project that aligned to TB but not the human reference genome did in fact come from a source other than human DNA, either due to TB infection or contamination, then this result strongly suggests that human variance does not render tuberculosis detection by NGS to be impossible or even difficult on the basis of similarities between the human genome and TB genome from a NGS experiment perspective. This conclusion is drawn due to the fact that even after analyzing over 46T nucleotides of human DNA, only 3.0% of the TB genome shared similarity with the human genome.

3.3.3 Advantages of the Uniqueness Genome Shown Through Simulated Data

To determine whether the uniqueness genomes created for TB aid in the detection of tuberculosis, several sets of simulated data were created to measure the coverage on both the full and unique tuberculosis genome by oral metagenomic samples in samples contaminated with other bacteria. To perform this analysis, the 250 species which demonstrated the highest coverage of the full tuberculosis genome were determined. In each sample, ten of these bacteria were randomly chosen as being present in the simulated sample. Additionally, the samples also included DNA from the human reference genome such that 75% of non-tuberculosis DNA came equally from the 10 species of bacteria, and the other 25% came from the human reference genome. The read locations from these genomes were chosen at random utilizing the same software used to simulate reads from the human genome in Chapter 2 and allowed sequencing errors and SNPs. Simulated reads from TB were also included from random locations in the TB genome to create overall TB sequencing loads ranging from 0% to 5% TB DNA. Sequence depth was also varied from 10M nucleotides up to 1B nucleotides (**Table 3.4**).

TB Load	Sequencing Depth	%Reads Aligned Full Genome	%Reads Aligned Unique Genome
0%	10Mb	0.071	0.002
0%	100Mb	0.1678	0.0001
0%	1Gb	0.12554	0.00022
0.10%	10Mb	0.206	0.059
0.10%	100Mb	0.2462	0.0645
0.10%	1Gb	0.19272	0.06375
1.00%	10Mb	1.094	0.641
1.00%	100Mb	1.0884	0.6388
1.00%	1Gb	1.13759	0.6353
5%	10Mb	5.041	3.194
5%	100Mb	5.2142	3.184
5%	1Gb	5.08449	3.17886

Table 3.4: Comparison of Percentage Reads aligned to the Full TB Genome and Uniqueness TB genome Using Simulated Metagenomic Samples. Between 10M and 1B nucleotides of short reads were simulated by creating simulated reads from 10 random bacteria from the oral metagenome as well as by creating reads from the human reference genome such that 75% of the non-TB reads came from equally from the 10 bacteria and 25% came from the human reference genome. Between 0%-5% of reads were simulated to originate from TB, and the percent of reads that aligned to the TB genome were measured. In each simulation, the same set of simulated reads was aligned to both the uniqueness TB genome and full TB genome to guard against biasing that may have resulted from using two different sets of simulated reads.

In all scenarios where tuberculosis was excluded, the uniqueness genome reduced the percent reads aligned which in these cases can be equated to the false positive alignment rate. This rate was reduced by over an order of magnitude in the 10M nucleotide simulated sample and by over two orders of magnitude in the 100M and 1B nucleotide simulated samples suggesting the usefulness of the unique genome in reducing false positive hits. It should be noted that since roughly 64% of the TB genome was retained in creating the uniqueness genome, approximately 64% of TB reads would be expected to align to the uniqueness genome, and this was seen in all tests. In all samples that contained TB, the alignment rate was well above the levels seen with 0% TB. However, the uniqueness genome did show a higher distinction between the 0% simulation and the 0.1% simulation than the full genome did especially in lower sequencing depths. The percent reads aligned at 0.1% TB ranged from 29.5x-645x the 0% TB coverage values when using the unique genome compared to 1.5x-2.9x for the same experiments using the full genome (**Fig 3.4**). This data could also be interpreted as a comparison between signal and background noise and demonstrated how the uniqueness genome was beneficial in filtering out the background noise that can originate from a sample containing many different bacteria. With TB loads of 1% and 5%, there was a larger difference in percent reads aligned from the 0% TB load simulation such that either the full TB genome or uniqueness TB genome could likely be used to discriminate between the two.

The small coverage of the unique tuberculosis genome in simulated samples not containing tuberculosis was attributed to two factors. The first was that the simulation of sequencer errors and polymorphisms resulted in a small number of alignments that were

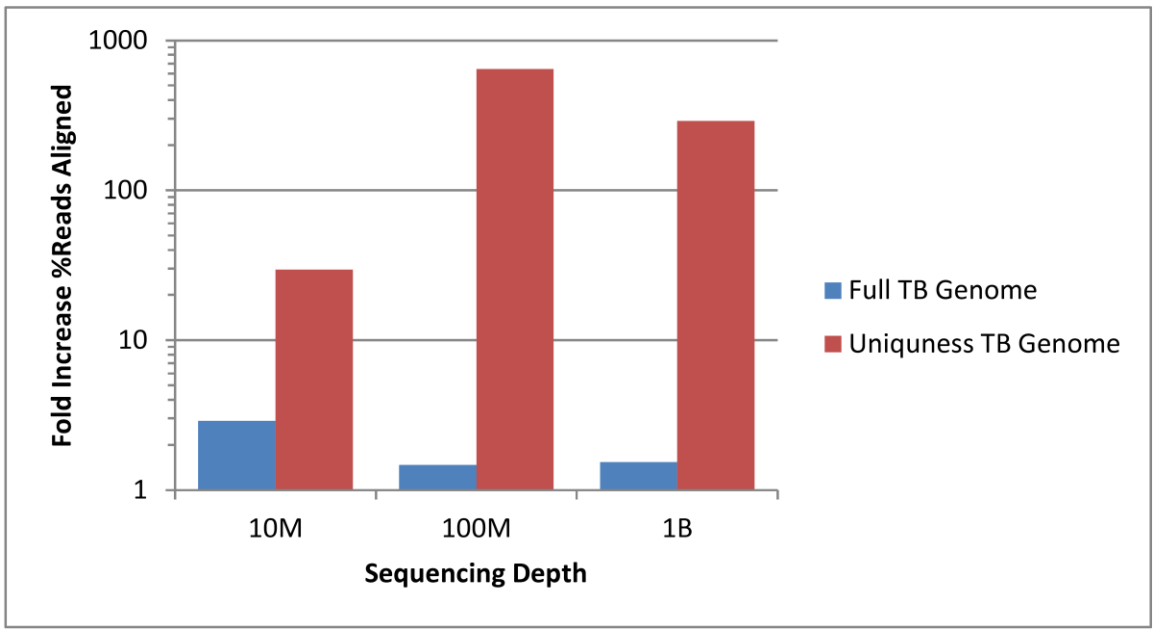


Figure 3.4: Comparison of Percent Reads Aligned at 0.1% TB Load and 0% TB load Using the Full TB Genome and the Uniqueness TB Genome. The percent reads aligned from the simulations in Table 3.4 were used to generate a comparison between percent reads aligned by dividing the percentage of reads aligned for the simulations with a 0.1% TB load by the percentage of reads aligned for the simulations with a 0% TB load. For the full TB genome, there was between a 1.5x to 2.9x increase in reads aligned whereas with the uniqueness TB genome, there was a 29.5x to 645x increase in reads aligned.

not detected in the original formation of the uniqueness genome. In a sense, this was viewed as a positive result since the analysis performed above was performed partially with the intention of observing what the role these sequence errors and polymorphisms play in alignment to the unique genome. The second potential cause of alignment to the uniqueness genome for TB was that elimination of portions of the full tuberculosis genome that were similar to other bacterial sequences could have resulted in a suboptimal alignment site in the full tuberculosis genome becoming the primary alignment site in the unique genome. This could possibly be remedied by performing the alignment of all sequences from the oral metagenome to the uniqueness TB genome multiple times with removal of regions found not to be unique after every round of alignment until no suboptimal alignments were found, but it was believed that the extent of suboptimal alignments did not warrant this time consuming process.

The simulated data was also analyzed in terms of $\frac{\%C}{B}$ to determine how using the uniqueness genome affected the coverage of the TB genome in the above simulations (**Table 3.5**). Whereas the percent reads aligned remained similar to the TB load at TB loads at or above 1%, the $\frac{\%C}{B}$ decreased for both the full and uniqueness genomes as sequencing depth increased under almost all simulations. The decrease in $\frac{\%C}{B}$ as sequencing depth increased was expected especially at higher loads of TB. Essentially, this effect was caused by a saturation of the TB genome by sequencing data. In other words, nearly the entire TB genome was covered in simulated samples that contained TB, and all the non-unique regions were covered by reads from other bacteria in the simulated samples that did not contain TB. Once saturated, no amount of increased sequencing depth will lead to a higher coverage since, even though a higher number of reads are

TB Load	Sequencing Depth	%C/B Full TB Genome	%C/B Unique TB Genome
0%	10M	12.270000	0.698874
	100M	9.508000	0.034944
	1B	1.406700	0.006400
0.10%	10M	35.550000	18.980000
	100M	28.427000	20.537000
	1B	20.830400	18.322800
1.00%	10M	238.740000	203.040000
	100M	206.129000	183.647000
	1B	89.704100	83.347300
5.00%	10M	1070.950000	969.660000
	100M	678.196000	619.476000
	1B	99.982200	97.573200

Table 3.5: Comparison of $\frac{\%C}{B}$ Between the Full TB Genome and Uniqueness TB genome Using Simulated Metagenomic Samples. The simulated samples from Table 3.4 were used to compare the differences in $\frac{\%C}{B}$ between the full TB genome and the uniqueness TB genome. $\frac{\%C}{B}$ generally decreases with increased depth and load due to saturation.

aligning, there are no locations in the genome where reads have not been aligned before. However, the increase in depth will be reflected in $\frac{\%C}{B}$ since a higher number of nucleotides were sequenced. For higher loads of TB and higher sequencing depths, $\frac{\%C}{B}$ was similar for both the uniqueness genome for TB and the full genome for TB because the entire genome was covered. The slightly higher $\frac{\%C}{B}$ in the full TB genome is due to the fragments in the uniqueness genome shorter than 100 bp to which reads cannot be aligned. Even with the decrease in $\frac{\%C}{B}$ as sequencing depth and load increases, there is a clear distinction between the abilities of the full TB genome and uniqueness TB genome to detect the presence of TB in a sample. This difference in the ratio of $\frac{\%C}{B}$ for samples containing TB and not containing TB was best seen at the lowest TB loads (**Fig 3.5**). As with measurements of load, the uniqueness genome showed a much higher ability to differentiate between samples containing TB and those not containing TB. Aligning to the full TB genome resulted in an increase in increase in $\frac{\%C}{B}$ ranging from 2.9x-to 14.8x when comparing simulations with 0.1% TB load to those with no TB. For the same simulations, using the uniqueness genome for TB increase $\frac{\%C}{B}$ by a factor of 27.2x-2863x again demonstrating the usefulness of the uniqueness TB genome in reducing background noise. Overall, the results from this first set of simulated data suggested that the uniqueness TB genome proved better able to distinguish tuberculosis from other bacterial species than the full TB genome. It was noted that under most of the above simulated circumstances, use of the full genome may have the ability to detect tuberculosis as well, albeit with more false positive alignments.

Next, it was determined how well the uniqueness genome and full genome for TB

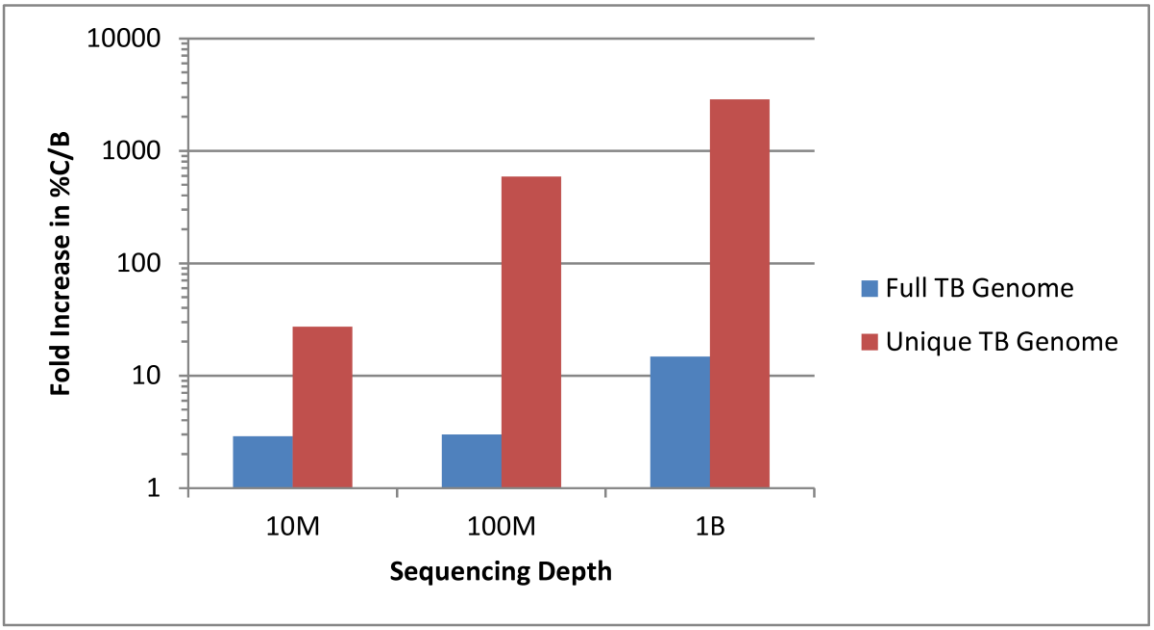


Figure 3.5: Comparison of $\frac{\%C}{B}$ at 0.1% and 0% TB Load Using the Full TB Genome and the Uniqueness TB Genome. The percent reads aligned from the simulations in table 3.5 were used to generate a comparison between $\frac{\%C}{B}$ by dividing $\frac{\%C}{B}$ for the simulations with a 0.1% TB load by $\frac{\%C}{B}$ for the simulations with a 0% TB load. For the full TB genome, there was between a 2.9x to 14.8x increase in $\frac{\%C}{B}$ whereas with the uniqueness TB genome, there was a 27.2x to 2863x increase in $\frac{\%C}{B}$.

could be used to distinguish between TB-positive samples and TB-negative samples at a low sequencing depth and low TB load from a worst case scenario for a false positive detection in which the species from oral metagenome genome most similar to TB were present. To perform this test, the 10 bacteria that showed the highest coverage of the full tuberculosis genome were used to simulated sequencings again such that 75% and 25% of non-TB DNA came from bacteria and the human reference genome respectively. In these samples, there was no TB included in the sample. These were compared to the TB-positive samples that would generate the lowest $\frac{\%C}{B}$ by simulating a load of 0.1% TB and no other alignments being generated by bacteria in the oral metagenome. This simulation for the samples not containing TB was performed a total of five times at depths of 10M and 100M nucleotides sequenced, and the simulation of the 0.1% TB load samples was performed three times so variance due to the random location of simulated sequenced reads from a NGS type experiments could be measured; the 1B nucleotide test was excluded due to a lack of variance due to saturation (**Table 3.6, Table 3.7)**. The average and standard deviation for $\frac{\%C}{B}$ for the samples containing no TB were calculated and compared to the $\frac{\%C}{B}$ in the samples that contained 0.1% TB. Based on the average and standard deviation for the samples with no TB, a z-score was calculated for each of the samples with a 0.1% TB load. In this analysis, it became very clear that at a low TB load (0.1%), a TB-positive sample cannot be distinguished from a worst case scenario for a false positive using the full TB genome. By comparing **Table 3.6** and **Table 3.7** with **Table 3.5**, it was determined that even at a load as high as 1% TB, there is hardly a distinction between a true positive and a false positive result at the low sequencing depth of 10M nucleotides.

RUN	Sequencing Depth	%C/B Full Genome		%C/B Unique Genome	
1	10M	184.42		12.93	
2	10M	193.04		12.87	
3	10M	194.93		10.44	
4	10M	194.73		9.73	
5	10M	180.48		11.77	
		Average : 189.52 Std Dev: 6.643		Average : 11.52 Std Dev : 1.435	
	0.1%TB Load	%C/B	Z-score	%C/B	Z-score
1		22.67	-25.115	19.94	5.847
2		22.67	-25.115	18.01	4.502
3		22.67	-25.115	19.3	5.401

Table 3.6: Worst Case Scenario Simulation for Detecting TB in a Metagenomic Sample Using a Sequencing Depth of 10 M Nucleotides Sequenced. To simulate the most difficult scenario possible for differentiating between a false positive and a true positive, a simulated sequencing of 10M nucleotides was performed using the 10 bacterial species shown to have the highest coverage of the full TB genome. This simulated sequencing was performed 5 times to account for variance that occurs due to the randomness of read generation in NGS. The $\frac{\%C}{B}$ for the alignment of each of the 5 samples was determined for both the full TB genome and the uniqueness TB genome, and their average and standard deviation was calculated. Next, 3 simulated sequencings were performed to simulate a 0.1% TB load with no other sequences aligning to the TB genome. The $\frac{\%C}{B}$ for the alignment of each of the 3 samples was determined, and based on the average and standard deviations calculated earlier, z-scores were determined for each sample.

RUN	Sequencing Depth	%C/B Full Genome		%C/B Unique Genome	
1	100M	100.6		9.482	
2	100M	100.5		9.309	
3	100M	100.5		8.948	
4	100M	101.6		10.002	
5	100M	101.4		9.076	
		Average: 100.9304 Std Dev: 0.510		Average: 9.3634 Std Dev: 0.412	
	0.1%TB Load	%C/B	Z-score	%C/B	Z-score
1		22.326	-153.9	20.664	27.4
2		22.337	-153.9	20.683	27.5
3		22.324	-197.6	20.683	27.5

Table 3.7: Worst Case Scenario Simulation for Detecting TB in a Metagenomic Sample Using a Sequencing Depth of 100 M Nucleotides Sequenced. To simulate the most difficult scenario possible for differentiating between a false positive and a true positive, a simulated sequencing of 100M nucleotides was performed using the 10 bacterial species shown to have the highest coverage of the full TB genome. This simulated sequencing was performed 5 times to account for variance that occurs due to the randomness of read generation in NGS. The $\frac{\%C}{B}$ for the alignment of each of the 5 samples was determined for both the full TB genome and the uniqueness TB genome, and their average and standard deviation was calculated. Next, 3 simulated sequencings were performed to simulate a 0.1% TB load with no other sequences aligning to the TB genome. The $\frac{\%C}{B}$ for the alignment of each of the 3 samples was determined, and based on the average and standard deviations calculated earlier, z-scores were determined for each sample.

In contrast, using the uniqueness genome resulted in a statistically significant difference between a false positive detection of TB and true positive detection of TB even for TB loads as low as 0.1% with a low sequencing depth of 10M nucleotides sequenced. The lowest z-score, 4.5, corresponds to $p<1x10^{-5}$. Extrapolating from the results in Table 3.6 and Table 3.7, for a sequencing depth of 10M nucleotides, p=0.05 would occur at a 0.073% load and p=0.01 would occur at a 0.079% load. By contrast, for the full genome, p=0.05 would occur at a load of 0.884% for a sequencing depth of 10M nucleotides. For a depth of 100M nucleotides, p=0.05 and p=0.01 occur at 0.048% and 0.050% for the uniqueness TB genome, and p=0.05 occurs around a 0.456% load for the full TB genome. It was therefore concluded that the use of the uniqueness genome enhances the sensitivity of this technique by approximately an order of magnitude in regards to the TB load required to confidently detect the presence of TB in a sample.

3.3.4 Confirmation of the Detection of TB in Finnish HapMap Samples Via the Complete and Partial Uniqueness TB Genomes

As noted when studying the effect of human variation on the coverage of the TB genome, several samples, all from the Finnish HapMap study, showed unusually high coverage of the TB genome when compared to other samples from the 1000 Genomes Project leading to the hypothesis that these samples were contaminated with TB DNA. Since there was a general trend that longer read length produced a lower percent coverage per nucleotide sequenced, $\frac{\%C}{B}$, every sample from the 1000 genomes project that contained reads of length 100 bp to 150 bp had determined their $\frac{\%C}{B}$ on the TB genome using –q 6 in alignment, and calculated the average, standard deviation, and z-score on

$\frac{\%C}{B}$ for all the samples. This subset of the 1000 genomes project contained over 21,274 samples and 26.4Tb of sequence. Of these samples, the 38 samples with the highest z-score all originated from the Finnish HapMap project and had z-scores ranging from 1.3 to 35.4 (**Table 3.8**). This corresponds to $\frac{\%C}{B}$ ranging from .46 to 9.58. It is also of note that the non-Finnish HapMap samples with the highest z-scores had sequencing depths between two and three orders of magnitude lower than the highest coverage Finnish HapMap samples. A small enough sample would lead to even one alignment producing an unusually high $\frac{\%C}{B}$, and this exact scenario appears to have occurred. For example, the non-Finnish HapMap sample with the highest z-score had a percent coverage of 0.0028% on the TB genome but also only had about 9 million sequenced nucleotides resulting in a value of 0.33$\frac{\%C}{B}$. This corresponds to a total of 125 bp aligned to the TB genome which corresponds to one read aligning to the TB genome. In a dataset containing over 80,000 samples, it was not surprising that a few samples produced a high coverage rate. The highest $\frac{\%C}{B}$ value from a non-Finnish HapMap sample of at least 100M nucleotides was 0.034$\frac{\%C}{B}$, an order of magnitude lower than the above noted sample with a 0.33$\frac{\%C}{B}$.

To further support the hypothesis that certain Finnish HapMap samples contained TB, the seven pairs of samples with the highest z-scores (ERR013120, ERR015732, ERR015872, ERR015874, ERR016001, ERR018499, and ERR018501) were again aligned to both the full TB genome and the complete uniqueness TB genome under different conditions that allowed fewer mismatches than the initial alignment (**Table 3.9**). In addition, the reads from these samples were also aligned to partial uniqueness genome containing the 100 largest contiguous regions of sequence unique to TB. An earlier

Sample	%C/B	**z-score**	Sample	%C/B	**z-score**
ERR018504_2	0.3616	1.2958	ERR015874	3.7319	13.7577
ERR015479_2	0.4617	1.6660	ERR015732_2	3.8117	14.0528
ERR013119	0.4754	1.7166	ERR018499	4.5330	16.7200
ERR013123_2	0.4845	1.7504	ERR018499_2	4.9188	18.1464
ERR015870_1	0.4920	1.7779	ERR015872_1	4.9383	18.2185
ERR015870_2	0.5288	1.9140	ERR018501	5.0679	18.6978
ERR015875_1	0.5439	1.9697	ERR013120	5.0959	18.8012
ERR018504_1	0.5542	2.0080	ERR015872_2	5.2358	19.3184
ERR013123_1	0.5691	2.0630	ERR016001	5.4455	20.0939
ERR015875_2	0.5797	2.1022	ERR015732_1	5.8767	21.6884
ERR015479_1	0.6694	2.4338	ERR018499_1	5.9164	21.8350
ERR018504	0.7032	2.5589	ERR018501_2	6.8265	25.2004
ERR016346	0.8892	3.2466	ERR013120_2	7.3282	27.0555
ERR016346_2	1.3863	5.0848	ERR018501_1	8.0321	29.6582
ERR016346_1	1.5171	5.5685	ERR016001_2	8.3692	30.9047
ERR015732	1.8524	6.8080	ERR015874_1	9.1924	33.9483
ERR013119_2	1.9011	6.9880	ERR015874_2	9.3977	34.7074
ERR015872	2.0170	7.4168	ERR016001_1	9.4487	34.8962
ERR013119_1	2.6782	9.8615	ERR013120_1	9.5757	35.3658

Table 3.8: 1000 Genomes Project Files with the Highest %C/B. Of the approximately 80,000 samples analyzed from the 1000 genomes project, the 21,274 samples containing reads of length 100 bp-149 bp were analyzed to determine their $\frac{\%C}{B}$. Samples of this length were chosen since the samples from the Finnish HapMap project that demonstrated unusually high coverage of the TB genome were included in this subset of samples. The 38 samples with the highest $\frac{\%C}{B}$ are shown and have z-scores for coverage rates ranging from 1.3 to 35.4. The above samples were all part of the Finnish HapMap project.

alignment of all the possible simulated reads - with no sequencing errors, SNPs, or indels - from the human reference genome to the full TB genome allowing no mismatches in alignment did not produce any alignments from the entire human genome. Thus, it was hypothesized that if the source of coverage of the TB genome from the Finnish HapMap samples were not from TB contamination, reducing the mismatches allowed between the reads and the reference would almost eliminate all coverage of the TB genome. Additionally, it was expected that if the source of contamination was from other bacteria, the uniqueness TB genome would show a significantly lower percent coverage than the full TB genome since the regions where TB was similar to other bacteria were excluded in the uniqueness TB genome.

When the alignments were performed, it was determined that even with no mismatches allowed between the TB genome and the reads from the samples from the Finnish HapMap project, anywhere from 4% to 17.6% of the complete uniqueness TB genome was covered by reads from Finnish HapMap project samples depending on the sample (**Table 3.9**). For the full TB genome, anywhere from 4.5% to 20.4% of the genome was covered by reads with no mismatches allowed, and using the partial TB uniqueness genome containing only the largest 100 fragments of the uniqueness genome, anywhere from 5.8% to 22.0% of the genome was covered. With up to 10 mismatches allowed, anywhere from 25.7% to 48.5% of the full TB genome was covered by reads, and from 22.5% to 43.1% of the uniqueness TB genome was covered, depending on the sample. The results that the full uniqueness genome tended to have a slightly lower coverage than the full TB genome is not inconsistent with the hypothesis that the alignments were generated due to TB contamination due to the fact that some of the

Run	Alignment Conditions	Full TB Genome Coverage	Complete Unique TB Genome coverage	Partial Unique TB Genome Coverage
ERR013120	m 0	5.90%	5.00%	6.80%
	m 5	21.80%	18.80%	23.60%
	m 10	28.70%	24.90%	30.30%
ERR015732	m 0	4.50%	4.00%	5.80%
	m 5	23.20%	20.20%	26.20%
	m 10	33.90%	30.00%	37.00%
ERR015872	m 0	6.80%	5.90%	8.40%
	m 5	20.80%	18.00%	23.10%
	m 10	25.70%	22.50%	27.70%
ERR015874	m 0	13.70%	11.80%	15.40%
	m 5	35.30%	30.70%	37.00%
	m 10	42.00%	37.00%	43.40%
ERR016001	m 0	20.40%	17.60%	22.00%
	m 5	42.80%	37.30%	44.60%
	m 10	48.50%	43.10%	50.30%
ERR018499	m 0	7.20%	6.20%	8.10%
	m 5	21.30%	18.10%	21.80%
	m 10	25.90%	22.40%	26.00%
ERR018501	m 0	10.00%	8.60%	11.50%
	m 5	32.20%	27.80%	34.00%
	m 10	40.50%	35.40%	41.60%

Table 3.9: Coverage of the Full TB Genome, Complete Uniqueness Genome, and Partial Uniqueness Genome by Samples from the Finnish HapMap Project. The seven samples from the Finnish HapMap project that demonstrated the highest coverage of the full TB genome under initial alignment conditions (-q 6) were aligned to the various forms of the TB genome using conditions that allowed fewer mismatches than the -q 6 setting allowed with m 0, m 5, and m 10 representing 0, 5, and 10 mismatches allowed respectively. Each of the runs was comprised of two samples containing both ends of the pair-end reads sequenced in the Finnish HapMap project. The complete uniqueness genome for TB was the form of the uniqueness genome for TB that had been used for all other experiments. The partial TB uniqueness genome was formed by taking the 100 largest contiguous segments of the complete TB uniqueness genome and

concatenating them into a separate uniqueness genome. Due to the complete TB uniqueness genome containing some sequences shorter than the read lengths of the reads in the samples used, certain portions of the complete TB uniqueness genome could not be aligned to but still comprised part of the total length of the complete TB uniqueness genome resulting in a lower coverage of the complete TB uniqueness genome than the full TB genome. The partial TB uniqueness genome did not contain any segments shorter than the read lengths, so all possible positions in the reference were available to be aligned to. This resulted in the partial TB uniqueness genome having similar coverage levels to the full TB genome.

smaller fragments of the complete uniqueness TB genome were too small for reads to align to. However they were included in the calculation for genome length which affects the percent coverage. Using only the largest fragments of the uniqueness genome demonstrated that without these smaller fragments, coverages were at least as high as those for the full TB genome. For zero mismatches, from 5.8% to 22.0% of the partial uniqueness genome was covered, and for 10 mismatches, anywhere from 23.7% to 50.3% of the partial uniqueness genome was covered. These coverages were slightly higher than the coverages for the full TB genome. That the percent coverage for all the forms of the TB genome with no mismatches allowed was much lower than the percent coverage with ten mismatches allowed was not surprising or inconsistent either. With a sequencing error rate of 1.5%, which is reasonable rate for Illumina sequencing, only 22% of reads were expected to have zero sequencing errors for a 100 bp read ($.985^{100}$ likelihood that all bases were correct if each had a 98.5% likelihood of being correctly called by the sequencing instrument) and therefore to be aligned to any of the TB genomes with zero mismatches.

Beyond the levels of coverage, which were far higher than in any simulation including even the worst case scenario tests, the hypothesis that these samples were contaminated with TB was further supported by plotting the percent coverage of the two forms of the uniqueness TB genomes as a function of percent coverage of the full TB genome (**Fig 3.6**). Both plots showed a strong linear correlation with R^2 values over .995 and slopes near to one which would be expected if the aligned reads came from TB since the coverages of the full TB genome and the uniqueness TB genomes would be expected to be approximately equal. That all alignment conditions resulted in almost perfectly

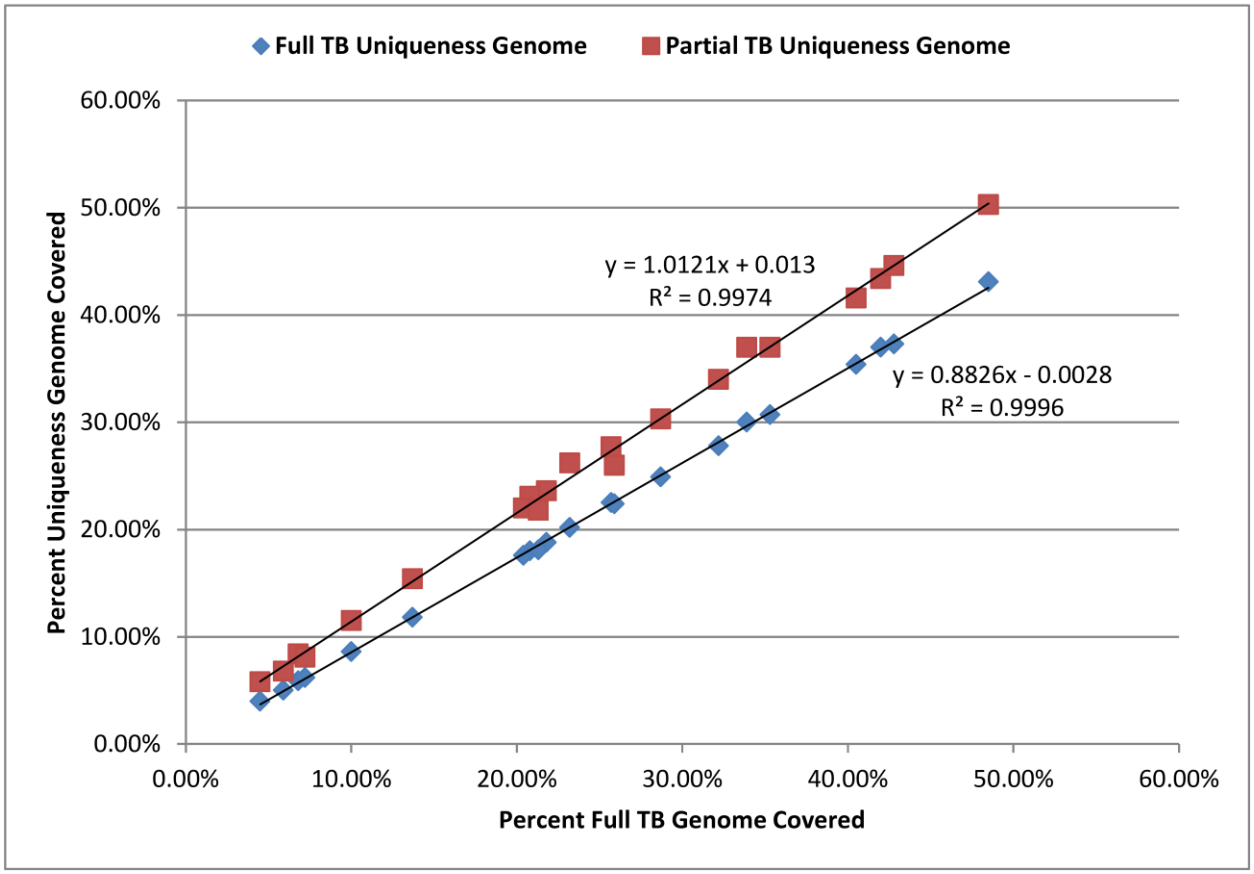


Figure 3.6: Coverage of the Complete and Partial Uniqueness Genomes for TB Versus Coverage of the Full TB Genome. The data from Table 3.9 for all seven samples was plotted on one graph to demonstrate the correlation between coverage of the full TB genome and coverage of the two forms of the uniqueness TB genome. The linear best fit lines, their equations, and R^2 values are also displayed. All samples demonstrated a nearly perfect linear correlation which, when coupled with the high coverage from each sample strongly suggested that the samples contained TB DNA.

linear correlation provided strong evidence that all seven of the samples analyzed contained TB. The full uniqueness genome has a somewhat lower slope due to the aforementioned fact that there are many small fragments of the genome shorter than the length of the reads to which it is impossible for reads longer than these fragments to be aligned to. When the partial unique genome consisting of the 100 largest contiguous stretches of TB DNA was used, the slope was almost exactly 1. The unity between percent coverage in the full TB genome and partial unique TB genome under all alignment conditions served as strong evidence that the source of aligned reads was not contamination by some other bacterial species, and the preservation of alignments with no mismatches strongly argued that the source of coverage was not due to similar regions from the human genome.

Next, the TB load in the above samples was measured. It was expected that if the source of alignments to the TB were primarily from sources other than TB, then the TB load for the unique genome ($Load_u$) would be much lower than the TB load for the full genome ($Load_F$) since the uniqueness genome for TB would greatly reduce the number of alignments coming from human DNA and bacterial DNA from the oral metagenome. In contrast, if the source of alignments to the TB genome were primarily due to TB contamination, it was expected that $Load_u$ would be very similar to the load on the full TB genome. This hypothesis was formed as follows: since a NGS experiment creates sequences randomly across a genome, the expected value for the number of times each base in the genome generates a read is uniform. Thus, eliminating some percent of the genome will eliminate on average that same percent of the reads aligned to the genome.

When the loads were compared using the -m 10 setting for each of the samples, every sample fell very close to 1:1 ratio between $Load_u$ and $Load_F$ (**Fig 3.7**). A plot of this data showed a very strong linear correlation with R^2 well above .99 further supporting the hypothesis of TB presence in the samples.

Finally, as an aside, it was also be noted that the $\frac{\%C}{B}$ in the above samples were well above the $\frac{\%C}{B}$ of the simulated worst case scenario for 1B nucleotides even though the Finnish HapMap project samples were even larger in size (1.5B to 3.2B nucleotides). Due to every piece of available evidence and every result supporting the hypothesis that TB DNA was present in these samples, it was concluded these samples indeed contained TB either due to TB infection or sample contamination and that the uniqueness genomes for TB were a valuable tool for verifying the presence of TB in real samples.

3.3.5 The TB Uniqueness Genome Eliminates or Greatly Reduces the Number of False Positive Alignments in Real Oral Metagenomic Samples

To further measure the usefulness of the uniqueness TB genome in the detection of TB in an oral metagenomic sample, several real oral metagenomic samples were downloaded from the SRA to attempt to either confirm the samples as positive or negative for TB. The samples downloaded were SRR488339, SRR488610, SRR488611, SRR488612, SRR488613, SRR488614, SRR488615, SRR488616, SRR488617, SRR488618, SRR488619, SRR488620, SRR488621, SRR488622, SRR488623, SRR488624, SRR488625, SRR488626, SRR488627, SRR488628, SRR488629, SRR488630, SRR488631, and SRR488632. Analysis of these samples by Leung, et al. by de novo assembly followed by a phylogenomic approach where the most taxonomically informative genes were searched for and a direct search using various databases

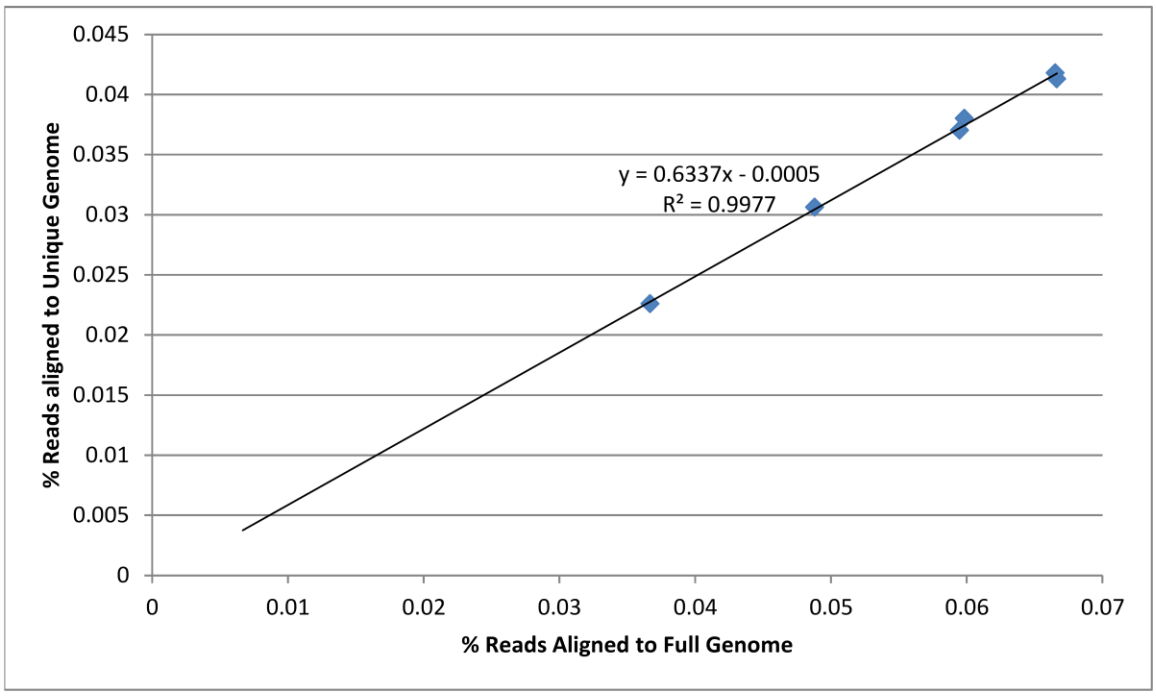


Figure 3.7: Percentage of Reads Aligned to the Complete Uniqueness Genome Versus Percentage of Reads Aligned to the Full TB Genome. The samples from table 3.9 were analyzed to determine the percentage of reads that aligned to either the full TB genome or the complete uniqueness TB genome. The linear best fit was determined, and its equation and R^2 value were displayed. The percentage of reads aligned to the complete uniqueness TB genome was always near 64% of the reads aligned to the full TB genome. Dividing the loads on the uniqueness genome by the ratio of the full genome length to uniqueness genome length (1:0.64) results $load_F$:$load_u$ ratio of almost exactly 1:1 as would be expected for samples containing TB.

demonstrated that all samples were negative for TB (Leung et al., 2012). The reads from these samples were aligned against the full TB and the complete uniqueness TB genome to study whether they would detect TB in the TB-free samples. When the full TB genome was used as the reference for alignment, reads that produced alignments to the TB genome were found in all but two samples (**Table 3.10**). In samples where reads aligned to the TB genome, loads ranged from 0.4% to 0.004% reads aligned. The sequencing depths on these samples were in the range of 50M-100M nucleotides sequenced per sample. Comparing this data to the simulated worst case scenario for a depth of 100M nucleotides, the supposed loads on these samples placed them in the grey area where it would be difficult to determine whether the alignments being generated were truly from TB or from other bacteria since for the worst case scenario simulated tests, $p<0.05$ does not occur until a load of 0.456% and $p<0.01$ requires an even higher TB load. However, when the uniqueness TB genome was used as the reference, no alignments were generated from any of the samples clearly demonstrating that these samples were indeed TB-negative in agreement with the previous work by Leung, et al. (**Table 3.10**). Although the uniqueness genome has been shown to reduce false-positive alignments by a factor of at least 10, complete removal of false-positive alignments was somewhat surprising. The two most likely causes for this phenomenon were that the loads were already very low in most cases and that although there were 24 samples generated, they all came from the same host organism. Due to this second reason especially, it was not nearly as surprising for all samples to behave similarly.

3.3.6 Subspecies Level Detection of TB

To demonstrate the ability of NGS to be used to quickly detect the strain of TB in

Run	% Load Full TB Genome	% Load Unique TB Genome
SRR488339	0.0002	0.0000
SRR488610	0.3838	0.0000
SRR488611	0.1647	0.0000
SRR488612	0.1017	0.0000
SRR488613	0.1691	0.0000
SRR488614	0.1798	0.0000
SRR488615	0.2049	0.0000
SRR488616	0.0992	0.0000
SRR488617	0.0704	0.0000
SRR488618	0.0000	0.0000
SRR488619	0.1032	0.0000
SRR488620	0.1742	0.0000
SRR488621	0.1421	0.0000
SRR488622	0.0235	0.0000
SRR488623	0.1451	0.0000
SRR488624	0.0704	0.0000
SRR488625	0.0319	0.0000
SRR488626	0.1764	0.0000
SRR488627	0.0430	0.0000
SRR488628	0.1650	0.0000
SRR488629	0.0292	0.0000
SRR488630	0.0486	0.0000
SRR488631	0.0041	0.0000
SRR488632	0.0000	0.0000

Table 3.10: TB Percent Load as Measured by Using the Full TB Genome and Uniqueness TB Genome as References. Samples sequenced by Leung et al were used as real metagenomic samples to demonstrate the effectiveness of the uniqueness TB genome in reducing false positive alignments. Alignment was performed using the -q 10 setting. Percentage of reads aligned is measured as $\frac{Reads\ Aligned}{Total\ Reads} \times 100\%$.

a sample, 23 additional strains of TB were downloaded and compared to the reference TB genome H37Rv to establish 36,220 locations within the TB genome where at least one strain of TB differed from the reference TB genome. The strains of TB used for this experiment were BTB05-552, BTB05-559, CCDC5079,CCDC5180, CDC1551, CTRI-2, F11, H37Ra, H37RvCO, HN878, KZN1435, KZN4207, KZN4207-v2, KZN605, KZNR506, KZNV2475, R1207, RGTB327, RFTB423, S96-129, UT205, and X122. Each of these strains was aligned to H37Rv using the -m 5 setting, and after all the variant sites were determined, for each strain, a table was generated to record the identity of each nucleotide at the variant site. It should be reinforced that each strain did not differ from the reference in 36,220 locations, but rather that among all 24 strains of TB, 36,220 variant sites were determined. For certain strains that were very similar to each other, there were as few as a couple dozen differences between strains while for other strains that were more distinct from each other, there were several thousand variant locations.

To determine the ability of NGS to detect a specific strain of TB, reads were simulated from a selected TB strain and aligned to the full reference TB genome H37Rv. These alignments were scanned through to search for alignments covered the locations where variants were found. For each variant location, the identity of the aligned base in the reads was determined and compared to the expected base at that location for each strain of TB. This process was repeated for all 23 strains of TB as well as for the reference strain H37Rv. In every sample tested, the strain from which the reads were generated demonstrated the highest level of similarity to the expected sequence. **Figure 3.8** demonstrates the results of determining the percentage of bases in the alignment that matched the expected sequence for each strain for BTB05-552. Although BTB05-552

showed the highest level of similarity, there were only 13 more differences detected in the expected sequence for a different strain S96-129 demonstrating how similar different strains of TB can be to each other. In other cases, the similarities are not nearly as close. For example, the strain most similar to CCDC5180 had over 600 differences from its expected sequence when compared to CCDC5180. For R1207, the amount of difference between strains was even more pronounced with the most similar strain having over 1000 differences from R1207.

Overall, this data provides preliminary evidence that it is possible to quickly identify strains of TB or what strain of TB a new strain is most similar to using NGS and SRmapper. Due to the sensitive nature of strain level detection and the small differences between certain strains, crude samples contaminated with other bacteria may not be usable to detect a specific strain of TB since a very small subset of the TB genome is used to differentiate between strains. Although knowing the identity of a specific strain of TB or which strain of TB a new strain is closest to does not directly determine the drug resistance patterns in that strain, it provides a quick method to give a preliminary suggestion of the drug resistance pattern provided the drug resistance pattern in the strains to which it is similar are known.

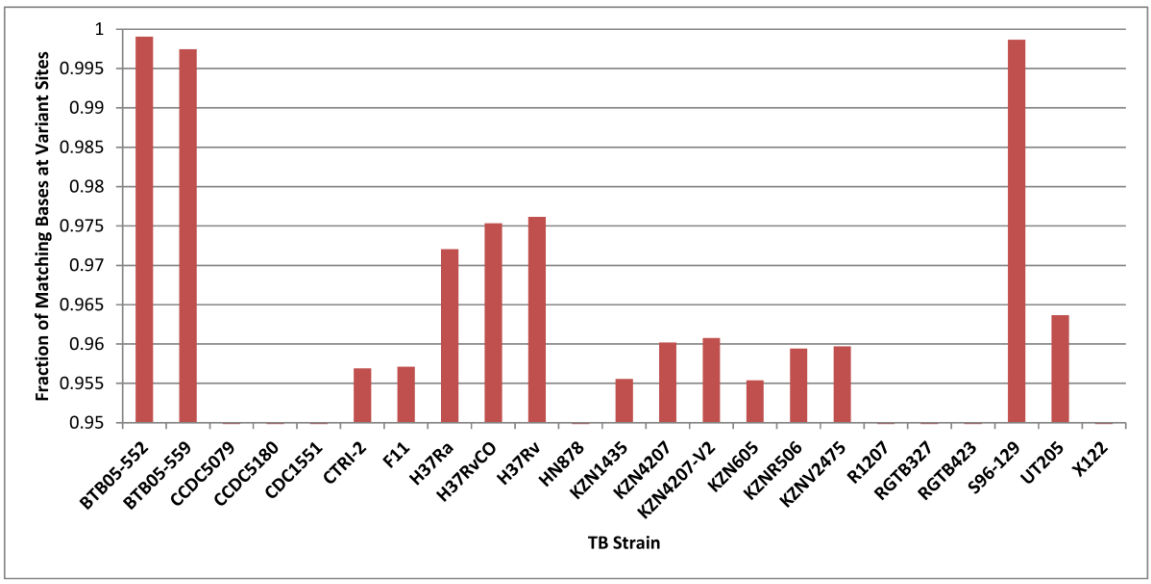


Figure 3.8: Strain level Detection of TB Strain BTB05-552. Simulated reads were generated from BTB05-552 and aligned to reference TB genome H37Rv. The identity of bases aligning to each variant site in the TB genome was determined, and the fraction of those bases that matched the expected sequence of each strain was determined and plotted. Although BRB05-552 had the highest fraction of matching bases at variant sites, S96-129 had only 13 fewer matching bases demonstrating the high levels of similarity between certain strains of TB.

Pseudocode 3.1: 1000 Genomes Analysis

```
Build the SRmapper index for TB;
For each file in 1000 Genomes Project
.  Download files;
.  Check Number of batch jobs via bjobs;
.  While submitted batch jobs > 90
.  .  Wait 10 seconds;
.  Batch submit Processing Script
.  .  Decompress downloaded .sra file to .fastq file;
.  .  Delete compressed .sra file;
.  .  Align .fastq file to TB genome;
.  .  Delete .fastq file;
For each alignment file
.   Determine which nucleotides are covered and not covered;
.  Store SAM file and coverage data in permanent location;

<end>
```

Chapter IV

Extension of the Uniqueness Genome Methodology to Simultaneously Detect Any Species Within the Oral Metagenome Using SRmapper

4.1 Background

Formation of the uniqueness genome for TB demonstrated the effectiveness of the uniqueness genomes method for one species in the oral metagenome. With the results from the formation of the uniqueness TB genome being encouraging, an attempt was made to expand the uniqueness genomes method to all species within the oral metagenome. The goal of this project was to demonstrate the possibility of creating a single uniqueness oral metagenome comprised of the uniqueness genomes of every species in the oral metagenome and to provide preliminary evidence that multiple bacteria can be simultaneously detected using the built-in functionality of SRmapper to align to many reference sequences simultaneously in a nearly time-independent mannerism. Due to the indexing process SRmapper employs, the size of a genome, or metagenome, did not significantly affect alignment time since the dynamic determination of key length ensures that each key occurs an average of once regardless of reference size. Due to similarities between species in genera that contained many species in the oral metagenome, it was unclear before performing any analysis whether the uniqueness genome methodology could be extended to all species in the oral metagenome. Additionally, new analysis tools were required to interpret the results of alignment to the oral metagenome since manual analysis of the results of alignment would be prohibitively slow. Thus, a method involving building contigs from the alignments and aligning them to the RefSeq database using the BLASTn algorithm was devised. Although the early results could not establish a quantitative measure of detection confidence, a qualitative detection was established with only a small coverage of the uniqueness genome of a species necessary to produce BLASTn results suggesting presence in samples tested.

4.2 Methods

4.2.1 Preparation of Species from the Oral Metagenome

Of the species listed in Chapter 3 (Table 3.1), 262 distinct species were chosen to build uniqueness genomes from. The reduction from 395 genomes to 262 genomes represented the choosing of one strain or subspecies from species with multiple sequenced strains available. For species with multiple genomes available, the most complete genome was selected on the basis of the number of contigs formed by the sequencing of that genome. The assembly for a genome was viewed as more complete if there were fewer contigs from that genome. Unlike the TB genome which has been extensively studied and fully assembled, the majority of genomes in the oral metagenome were in an incomplete form consisting of anywhere from a few contigs that do not overlap to several hundred fragments ranging from millions of base pairs to a few hundred base pairs long. For each of these genomes, all contigs were merged into a single sequence in .fasta format by padding between each contig with a short stretch of ambiguous, N, nucleotides to prevent alignments being generated across multiple contigs. Merging contigs into a single fasta files was chosen over concatenating contigs into a multifasta file for ease of determining which species a contig originated from. With several hundred contigs per genome and 262 genomes, building indexes for and aligning to tens of thousands of reference sequences would have put an unnecessary time strain on the project.

4.2.2 Formation of the Uniqueness Genomes for All Species in the Oral Metagenome and of the Oral Uniqueness Metagenome

To form the uniqueness genomes, indexes of each full reference genome from each bacterial species was formed using SRmapper Buildindex. Next, for every species of the 262 used in the oral metagenome, reads were simulated from every possible position to form a fastq file as in Section 3.1.5. The reads from each species were aligned to all other species one at a time excluding the species from which the reads originated from using SRmapper Align with either the -q 6 or -m 5 setting for alignment. After alignment, the portions of each bacterial genome that were aligned to were noted. The fastq and SAM files were then deleted, and the next genome was used to simulate reads. This process was repeated until every bacterial genome from the 262 selected genomes had been used to form reads. Finally, the human reference genome was also used to simulate reads for alignment to the bacterial references. After the covered regions had been determined, they were removed from the bacterial reference genomes. Remaining unique regions were separated by a string of 50 ambiguous nucleotides as described earlier to prevent reads aligning across unique regions. Since alignment using SRmapper Align is roughly time independent of genome size due to the formation of the index, forming one uniqueness metagenome from all the uniqueness genomes in the oral metagenome would result in alignment that is over 200x faster than aligning to each uniqueness genome sequentially since alignment needs to take place once. Thus, all the uniqueness genomes were ordered and listed alphabetically by genus then species and then concatenated into a single uniqueness oral metagenome with each genome numerically named based on its position in the list of genomes from the oral metagenome to facilitate easier downstream analysis to form a single multifasta file containing all the unique portions of every

genome from the oral metagenome. This uniqueness oral metagenome was successfully indexed using SRmapper Buildindex so it could be used in alignment.

4.2.3 Detection of Bacteria from the Oral Metagenome Using the Uniqueness Oral Metagenome

Reads were aligned to the uniqueness oral metagenome using SRmapper Align with the -m 5 option specified to limit the number of mismatches allowed per read to 5. The -d option was also specified to save additional data (number of reads and nucleotides) to aid in calculating load in downstream analysis. Software modified from the original program used in the determination of the fraction of the TB genome covered by reads in alignment was used to simultaneously determine the coverage and load of every genome in the uniqueness oral metagenome. Load was determined by counting the number of reads aligned to a genome within the uniqueness oral metagenome and dividing the number of alignments by the number of reads in the sample, which was stored in the extra data file generated by specifying the -d option in SRmapper Align.

4.2.4 Verification of Oral Metagenomic Bacterial Detection by Using BLASTn

To determine the ability of the uniqueness oral metagenome to detect the presence of bacteria in metagenomic samples, the results of alignment of reads to the uniqueness oral metagenome were analyzed by aligning aligned sequences to a more comprehensive database, namely BLASTn to determine whether the aligned reads originated from any other bacteria including those possibly not listed in the oral metagenome. However, since BLASTn was poorly suited to perform alignment of the short reads from a sequencing experiment or even the subset of reads aligned to a specific uniqueness genome within the uniqueness oral metagenome, aligned reads were first assembled into contigs by

determining the consensus sequence for alignments. The consensus sequence is formed by choosing the nucleotide that aligned most often to each position in the reference genome. Since even the number of contigs formed precluded usage of BLASTn to search for alignments to these contigs, a selection method was employed to choose a subset of contigs for alignment by BLASTn. **Figure 4.1** demonstrates the overall methodology used to verify the presence of bacterial species in a sample.

Simply choosing the longest contigs was tested but rejected due to not demonstrating any correlation between contig lengths and correctly identifying the bacteria aligned to by BLASTn (criteria for determining the correct origin for contigs is described subsequently). It was speculated that long contigs could possibly be biased towards being at least partially formed by false-positive alignments. Due to the random nature of NGS read creation, contigs would be expected to follow a normal distribution. If both true-positive and false-positive alignments were generated in a specific region, contigs from these regions would be expected to be longer than the average lengths of contigs being formed since additional false-positive alignments produced could skew contig length. The method eventually chosen was to use Shannon Entropy to measure the amount of disorder in contigs. Shannon Entropy is defined as

$$H(x) = -\sum_{i=A,C,G,T} P_i \ln P_i \qquad \text{Eq 6}$$

where H denotes Shannon Entropy, and P_i denotes the probability of finding a specific nucleotide at the position. The more disagreement there is for the identity of a base at a specific location, the higher the Shannon Entropy will be at that location. As the amount of disagreement decreases, Shannon Entropy approaches 0 since for the only occurring

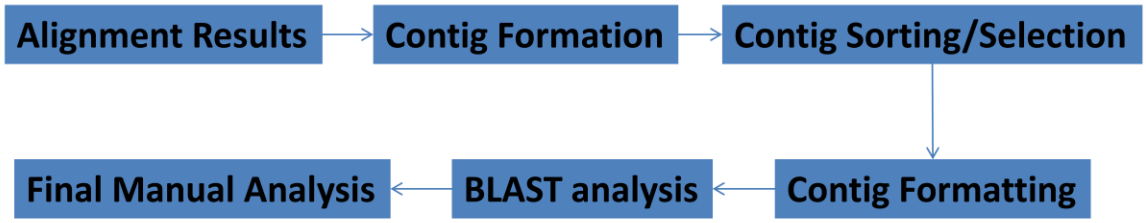


Figure 4.1: Validation of Bacterial Detection by BLASTn Analysis. To validate the results of alignment to the uniqueness oral metagenome using SRmapper, alignments to each species within the oral metagenome were analyzed using a more comprehensive database, BLASTn, to verify that the aligned reads originated from the species to which they were aligned. Since BLASTn cannot handle the large number of alignments formed in NGS analysis, overlapping alignments were first used to form contigs. Since even the number of contigs proved to be too high for analysis using BLASTn, these contigs were sorted by Shannon Entropy and the best 50 or 100 contigs for each species were formatted for upload to BLASTn. After the BLASTn analysis, a final manual analysis was performed to determine whether the BLASTn results indicated a species was present.

nucleotide, P_i=1 and $1 \times \ln(1) = 0$ and $\lim_{P_i \to 0}(P_i \log P_i) = 0$ by L'Hospital's Rule. Thus, for cases where there is no disagreement for the identity of the base, H=0.

Figure 4.2 demonstrates the formation of a consensus sequence and calculation of Shannon Entropy for bases in a contig. Shannon Entropy was used to determine contigs under the hypothesis that if multiple bacterial species were aligning to the same region of DNA, there would be more disorder in the consensus sequence since multiple bacteria are more likely to have somewhat different sequences that align to a region compared to a single bacteria whose reads should all form the same sequence. The contigs with the lowest Shannon Entropy that aligned to each reference within the uniqueness oral metagenome were converted into a format usable in a BLASTn search. Using Shannon Entropy to choose contigs showed a modest correlation between lower Shannon Entropy and correctly identifying the species being searched for. The database used for alignment in BLASTn was changed from the default "Nucleotide Collection (nr/nt)" to "RefSeq" and the maximum matches in the query range was set to 1. Results from the BLASTn search were automatically analyzed by software developed to measure the fraction of the contigs that aligned to various sequences in the BLASTn database. A bacterial species was considered present if a higher fraction of bases from the contigs originating from SRmapper alignments was found to align to the species from which it supposedly originated than any other species in the BLASTn results. A genus was considered present if a higher fraction of bases from the contigs aligned to the originating genus than any other genus in the BLASTn results.

4.2.4.1 Formation of Contigs from Alignment and Using Shannon Entropy to Select Contigs

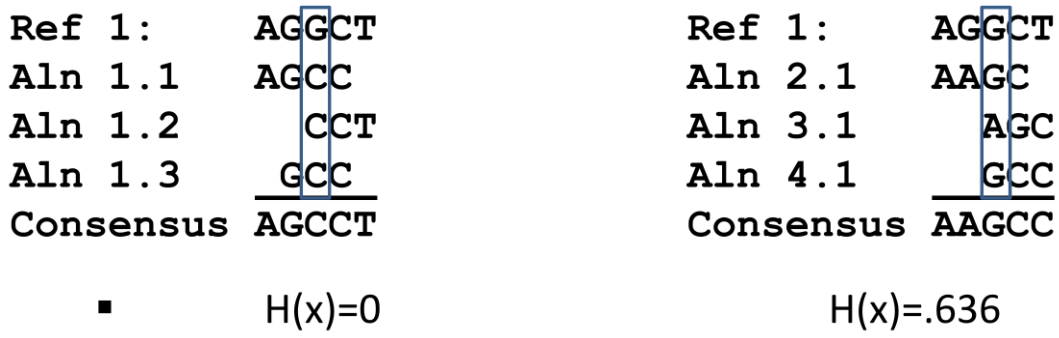


Figure 4.2: Consensus Sequence Formation and Calculation of Shannon Entropy: Two sample alignment scenarios are provided. The scenario on the left demonstrates the expected results of three reads originating from a different strain reference 1 of a bacteria being aligned to reference 1. The consensus sequence is shown below the alignments. The position in highlighted in the blue box demonstrates how the consensus sequence can differ from the reference sequence but still maintain a Shannon Entropy, H(x), of 0 at that position since the is no disorder in the consensus sequence. Cases where one alignment produced a match at a given nucleotide while another alignment produced a gap or deletion at that position would not be considered a zero entropy consensus since different results were obtained for that position. The scenario on the right demonstrates the expected results of three reads each originating from different species being aligned to reference 1. The consensus sequence is again displayed below the alignments. The position highlighted in the blue box demonstrates that for positions where there is disagreement between the alignments, the base occurring most often at that position is chosen to form the consensus sequence. This disagreement leads to a nonzero value for H(x) at the position in the blue box.

Formation of contigs was accomplished by development of in-house software that scans through alignment files to record the number occurrences of each base aligning to a specific location in the reference sequence. To form contigs and determine their Shannon Entropy, the maximum length for each bacterial genome in the oral metagenome was set to 10Mb. Two dimensional arrays were used for each reference to record the number of times each base aligned to every position in a reference genome from the uniqueness oral metagenome. For each reference genome being searched for simultaneously, 50 million integer spaces were stored in memory (10 million bases per reference multiplied by the four different bases and the total number of nucleotides aligned to a base). This required 200MB of memory per reference being scanned. Since scanning through a file was the slowest step in calculating Shannon Entropy, 10 genomes were used in each reference resulting in the use of approximately 2GB of memory for the program to run. Giving each reference sequence in the uniqueness oral metagenome a numerical name (Section 4.1.2) facilitated easy tracking the reference genomes in which alignments were being searched for. After the alignment file had been scanned through and all alignments found and the number of times each base occurred at each location was determined, the Shannon Entropy was calculated for each position to which bases aligned. Two files were created to store the output of calculating the Shannon Entropies for all alignments to the uniqueness oral metagenome. In the first file, the number of times each base aligned to a location was stored along with the Shannon Entropy for that position. In the second file, contiguous stretches of sequence with aligned bases were used to form contigs by choosing the base that most often aligned to each position in the reference. The average Shannon Entropy for each contig was calculated by taking the average of all the Shannon

Entropies from the bases that formed the contig. The reference and location within the reference of the first base in the contig was also stored along with the length of the contig.

A second piece of in-house software was developed to sort through the contigs and select a subset of them to be used in BLASTn analysis. Three options for sorting through the contigs were permitted. The first was to sort by the longest contig length; the second was to sort by lowest Shannon Entropy then contig length if two contigs had the same Shannon Entropy; the third was to sort by a combination of contig length and Shannon Entropy where contigs were sorted by higher values of $\frac{Contig\ Length}{Shannon\ Entropy}$. Additionally sorting could be performed to either sort by strictly listing the best contigs first or by grouping the best contigs for each species so that each species could be searched for in BLASTn. If the option to group contigs by species was chosen, another option allowed for a set number of the best contigs for each species to be chosen since BLASTn could not handle thousands of contigs per species. Even with only selecting the 50 or 100 best contigs per species, there were often too many contigs for BLASTn to perform alignment in the maximum amount of time allowed for a job. Thus, another option was created that allowed for users to split the file to upload to BLASTn into several files each containing a user defined number of the references to which contigs were aligned. This value was usually set to either 10 references or 25 references.

The sorting algorithm originally used was bubble sort due to its simplicity to implement and the belief that a more complex, more efficient sorting algorithm was unnecessary. Bubble sorting, or bubbling, sorts a list by comparing two adjacent values, V_i and V_{i+1}, in the list and swapping them if the larger value is further down in the list.

Next V_{i+1} and V_{i+2} are compared with the same swap being performed and this processes is continued until the end of the list is reached. The list is then scanned through multiple times until no swaps are performed in a scan. At the end of each scan, the largest unsorted value reaches its position in the sorted list by "bubbling" through the list. Bubble sort has an average time required to perform the sort of $O(N^2)$ for a list of N items meaning that as N increases, the time require increases by N^2. For many cases using bubble sort proved satisfactory, but in the cases where several hundred thousand contigs required sorting, the sorting and selection process required over half an hour of time which was deemed too slow.

Bubble sorting was replaced by comb sorting to facilitate faster sorting. Although not as time efficient as mergesort, heapsort, or quicksort, comb sorting was much easier to implement and reduced the time required to perform sorting of lists containing several hundred thousand contigs from over half an hour to less than five seconds. Comb sorting works on the same swap principle as bubble sorting but utilizes a gap between compared items in the list that shrinks every time the list is scanned. Since performing swaps is usually the most time-intensive portion of a sorting algorithm, comb sorting greatly reduces time by quickly moving values that originally are near the end of a list but need to be located near the front of the list and visa-versa. In practice, a shrink factor of 1.3 was used meaning that every time the list was scanned, the size of the gap was divided by 1.3 until a gap size of 1 was achieved.

Output was generated such that output files could be fed directly into BLASTn by using the format of a header line starting with a '>' symbol followed by reference name followed by contigs each being placed on a new line. Starting a line after a contig

sequence with a '>' denoted to BLASTn that a new reference was being used. Thus, multiple references could be loaded into one file for direct submission to BLASTn.

4.2.4.2 BLASTn Analysis on Selected Contigs

For each search using BLASTn, two files forming the results of the BLASTn search were downloaded for analysis. The "Text" file was downloaded for genus and species names for each alignment generated. The "Hit Table(text)" was downloaded for what fraction of the contig bases aligned to the reference sequence. Software was developed to analyze the Text and Hit Table(text) files to determine the percentage of bases from contigs that aligned to each reference sequence in the BLASTn search. The software tracked the BLASTn coverage on the reference sequence for the species from which the reads were aligned, the highest BLASTn coverage from other members of the same genus from which the reads were aligned, and the highest BLASTn coverage from other species outside the genus from which the reads were aligned. These results were reported for each species in the uniqueness oral metagenome. A $BLASTn_{species}$ score and $BLASTn_{genus}$ score were also reported. These scores were defined as follows:

$$BLASTn_{Species} = ln\frac{\%Speceis\ Coverage}{\%nonSpecies\ Coverage+1} \quad \text{Eq 7}$$

$$BLASTn_{genus} = ln\frac{Genus\ Coverage}{\%nonGenus\ Coverage+1} \quad \text{Eq 8}$$

where species coverage was the fraction of the bases from the contigs that BLASTn aligned to the same species that the reads were aligned to by SRmapper, genus coverage was the highest fraction of the bases from the contigs that BLASTn aligned to the species in the same genus as the species that the reads were aligned to by SRmapper, and nonspecies coverage or nongenus coverage were the highest fraction of the bases from the contigs that BLASTn aligned respectively to a species or genus outside that from

which the alignments in SRmapper were generated. This data was plotted against SRmapper alignment coverage of each uniqueness genome to determine whether there was a correlation between BLASTn scores or a cutoff for positive BLASTn scores.

4.3 Results

4.3.1 Uniqueness Reference Genomes Can Be Created for All Species in the Oral Metagenome

The first formation of the uniqueness oral metagenome used the -q 6 alignment setting as had been performed for creation of the uniqueness TB genome. Assuming an average genome length of 5Mb, this project required the alignment of 34Tb of bacterial DNA (5Mb per bacteria multiplied 100 bp per read multiplied by 262 different bacteria each aligning to the other 261 bacteria in the oral metagenome) and 78Tb of human DNA (3Gb genome multiplied by 100 bp per read multiplied by 262 genomes to align against). The first formation of the uniqueness oral metagenome resulted in the formations of uniqueness metagenomes for many species which left a very small fraction of the bases available for use (**fig 4.3**). Roughly a third of the species present had more than 30% of the bases from their genomes covered by other species; nearly 10% had over 80% of their bases covered by other species, and over 5% had less than 10% of their genome available for use. In these cases, this meant a relatively small portion of the genome was available for usage as the uniqueness genome for that species. This was attributed to setting too lose a constraint on what was defined to be similar between two different genomes. In the case TB, much of the genome was not covered by reads from other genomes due to the fact that TB only had one other bacterial species, M*ycobacterium leprae*, in its genus included in the oral metagenome. Bacteria from outside the genus were

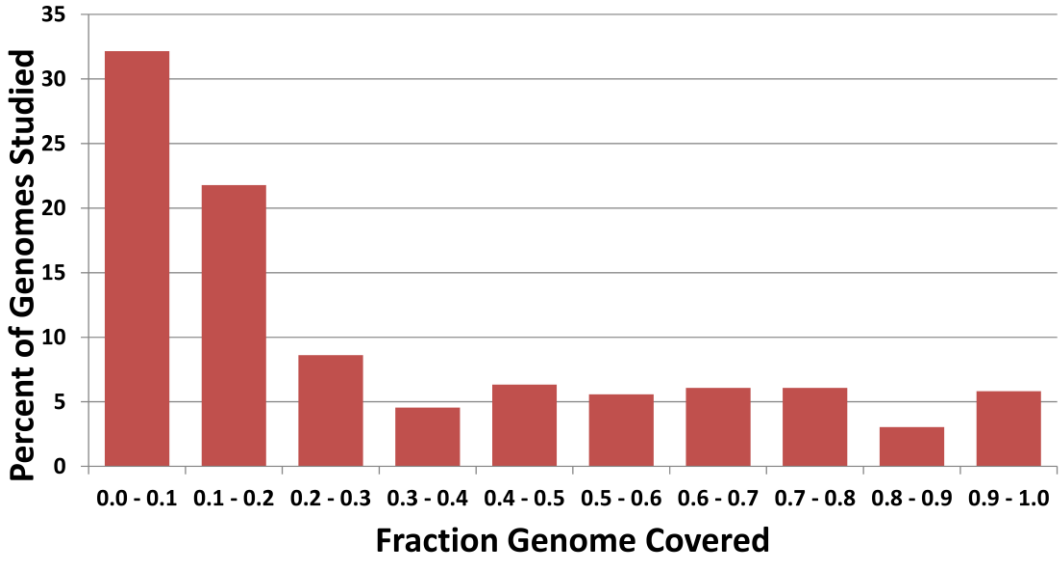


Figure 4.3: Initial Construction of the Uniqueness Oral Metagenome. Uniqueness genomes for each of the species in the oral metagenome were generated using SRmapper with the -q 6 option. The distribution of the fraction of each genome covered is displayed. Fraction covered denotes what portion of each genome is not unique to that particular species and is removed in the formation of the uniqueness genome for that species.

expected to demonstrate lower coverage of the TB genome due to their genomes being less similar to the TB genome than another species within the mycobacterium genus. Checking the coverages from each species on the TB genome, this proved to be the case. Looking more closely at the distribution of coverages of species in the oral metagenome, it was seen that genera containing many species from the oral metagenome were especially affected by the fairly relaxed policy on creating the uniqueness oral metagenome (**fig 4.4**). The genus most affected was *Streptococcus* which contained 36 species in the oral metagenome. Of these 36 species, 11 had coverages higher than 80% of their genome. The other genera with high number of species were not as heavily affected but did reveal some species with high coverages.

To reduce the number of species with a high percentage of their genomes covered by reads from other species within the oral metagenome and therefore with only a small portion of their genome available to align to using the uniqueness genome strategy, the alignment conditions in creating the uniqueness genome was tightened using the -m 5 option, and the uniqueness genomes were rebuilt. Upon completion of this second build of the uniqueness oral metagenome, it was determined that a much larger fraction of the genomes of the various bacteria were available for use in their respective uniqueness genomes (**fig 4.5**). No species displayed coverage over 90% of its genome and less than 5% of species displayed a coverage of 80% or more of their genome. Using the -m 5 conditions, over two-thirds of bacterial species had less than 10% of their genomes covered by reads from other species leaving the majority of their genomes available for use. Since this second creation of the uniqueness oral metagenome genome displayed a much higher proportion of available sequence for use in alignment to the uniqueness oral

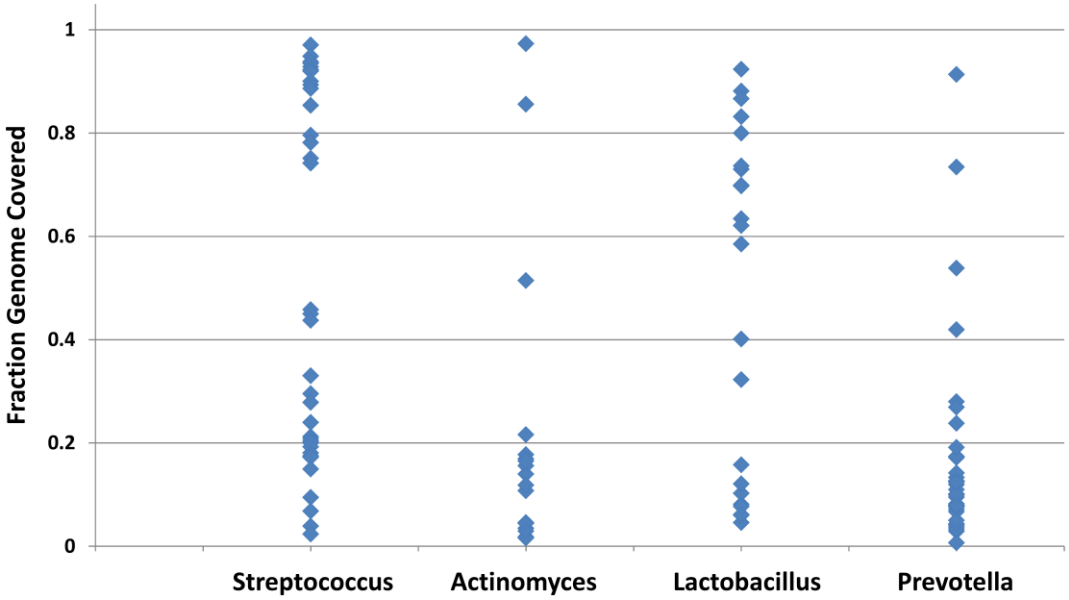


Figure 4.4: Distribution of Coverages for Genera with a High Number of Species in the Oral Metagenome. The distribution of the coverages for the four genera with the highest number of different species in the Oral Metagenome is displayed. The number of species in *Streptococcus*, *Actinomyces*, *Lactobacillus*, and *Prevotella* were 36, 17, 24, and 36 respectively.

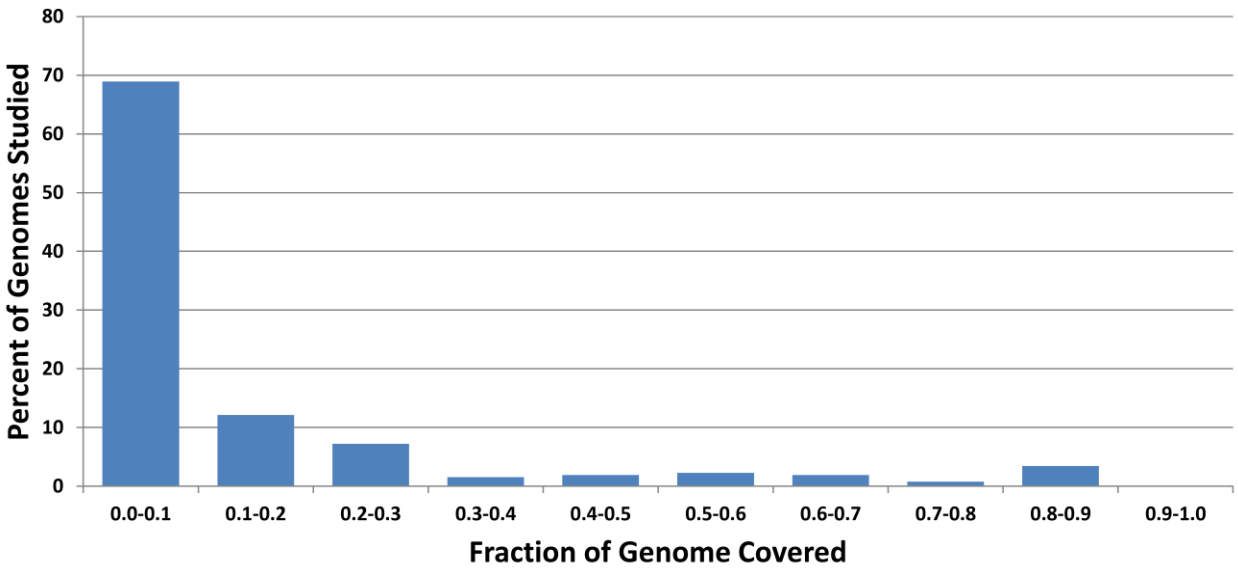


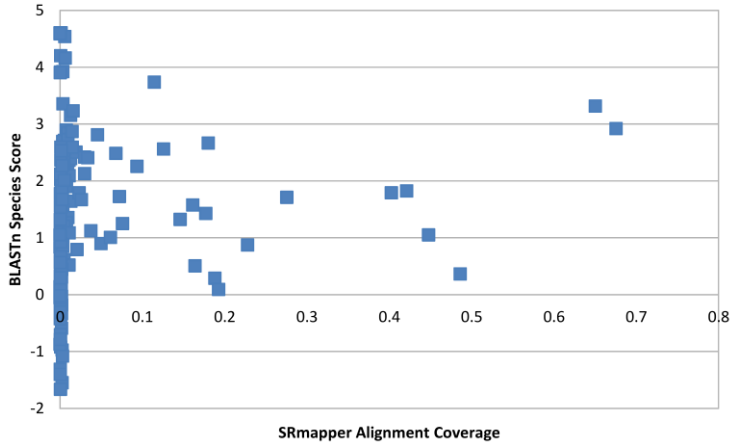
Figure 4.5: Rebuild of the Uniqueness Oral Metagenome. Uniqueness genomes for each of the species in the oral metagenome were generated using SRmapper with the -m 5 option to allow a maximum of 5 mismatches per alignment. The distribution of the fraction of each genome covered is displayed. Fraction covered denotes what portion of each genome is not unique to that particular species and is removed in the formation of the uniqueness genome for that species.

metagenome, it was used in all tests validating the possibility of using whole-genome NGS as a detection agent for any species within the oral metagenome by using SRmapper and the uniqueness oral metagenome.

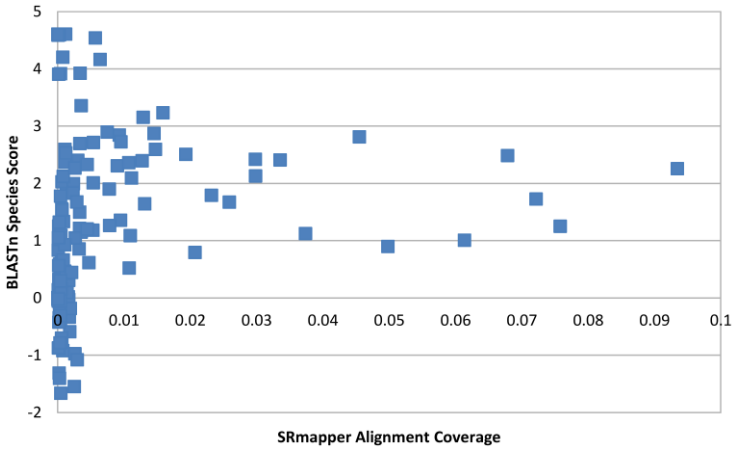
4.3.2 Detection Validation through BLASTn and Detection Limits for Species-Level and Genus-Level Detection

To determine the effectiveness and accuracy of the uniqueness oral metagenome in detecting any bacterial species in the oral metagenome, several real datasets were aligned to the uniqueness oral metagenome. Since the uniqueness oral metagenome was formed using the -m 5 option by SRmapper in alignment of simulated reads from the various species in the oral metagenome, the -m 5 option was also used in the alignment of real metagenomic samples. The samples used were SRR331033, SRR331034, SRR331035, SRR769511, SRR769512, SRR769517, SRR769521, SRR769522, SRR769535, SRR769536, SRR769539. After alignment of the reads to the uniqueness oral metagenome, the 50 best contigs from alignments to each species as determined by lowest Shannon Entropy were selected and formatted to be used in a BLASTn search. For every species in the oral metagenome, the coverage of the contigs by that species was determined as well as the highest coverage from other species in the same genus and the highest coverage from species outside the genus. From this information, the $BLASTn_{species}$ and $BLASTn_{genus}$ scores were calculated for each species in the oral metagenome (**fig 4.6** and **fig 4.7**). The results shown are for SRR331035 and are fairly representative of all tests performed. For all tests analyzed, an alignment coverage of 0.5% or higher on the uniqueness genome for any species in the oral metagenome always resulted in the presence of that species being verified by the BLASTn analysis regardless

A


BLASTn Species Score
SRmapper Alignment Coverage
5
4
3
2
1
0
-1
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

B


BLASTn Species Score
SRmapper Alignment Coverage
5
4
3
2
1
0
-1
-2
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1

C

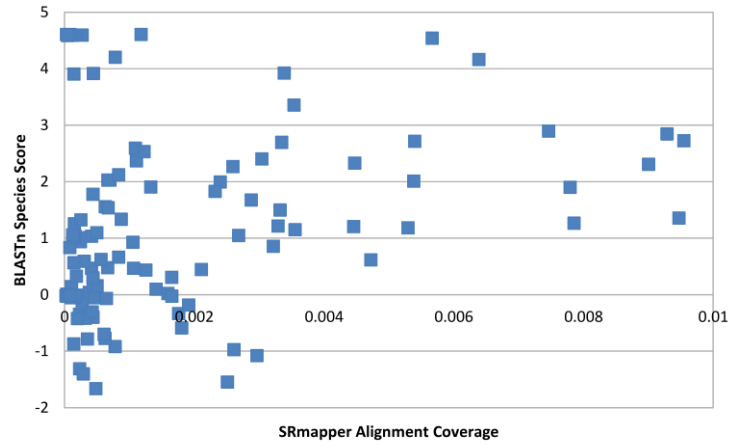
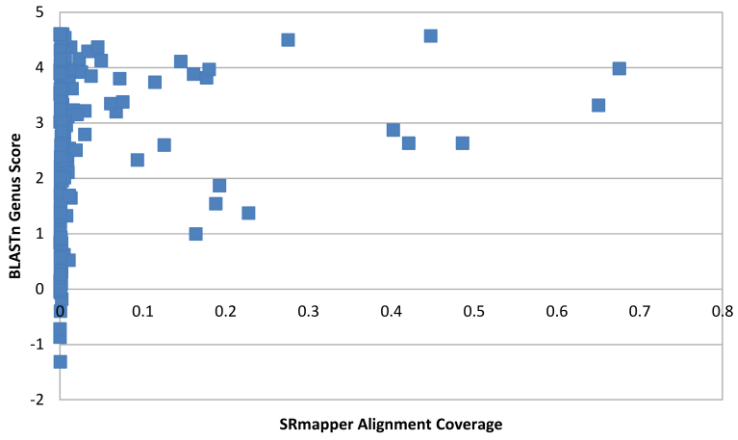
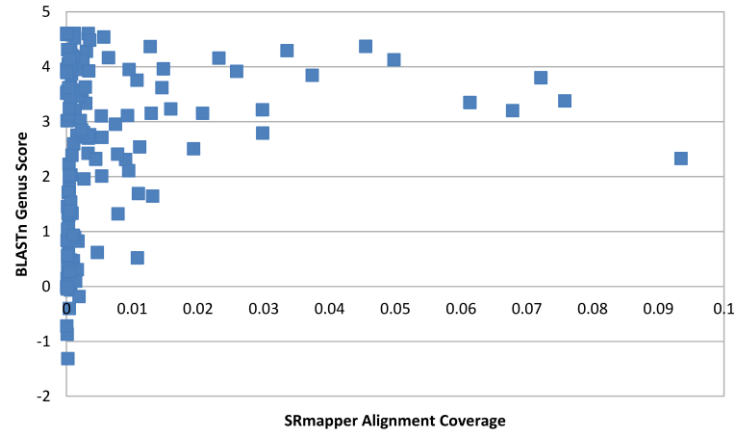

BLASTn Species Score
SRmapper Alignment Coverage
5
4
3
2
1
0
-1
-2
0
0.002
0.004
0.006
0.008
0.01

Figure 4.6: $BLASTn_{species}$ Scores Versus SRmapper Alignment Cover for Uniqueness Genomes in the Oral Metagenome. The reads from SRR331035 were aligned to the uniqueness oral metagenome using SRmapper with the -m 5 options and coverages on each of the uniqueness genomes were measured. Contigs from alignments were analyzed using BLASTn as described earlier. $BLASTn_{species}$ score for each species was determined by dividing the BLASTn query coverage for the species from which the contigs aligned by the highest BLASTn query coverage for any other species including those outside the oral metagenome. Panel A, B, and C focus on coverages ranging from 0% to 100%, 0% to 10%, and 0% to 1% respectively. The maximum $BLASTn_{species}$ score obtainable was ln(100/1) = 4.6. For SRR331035, no SRmapper alignment coverages higher than 0.3% resulted in negative $BLASTn_{species}$ scores although in certain other samples SRmapper alignment coverages as high as 0.5% resulted in negative $BLASTn_{species}$ score**.**

A


BLASTn Genus Score
5
4
3
2
1
0
-1
-2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
SRmapper Alignment Coverage

B


BLASTn Genus Score
5
4
3
2
1
0
-1
-2
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
SRmapper Alignment Coverage

C

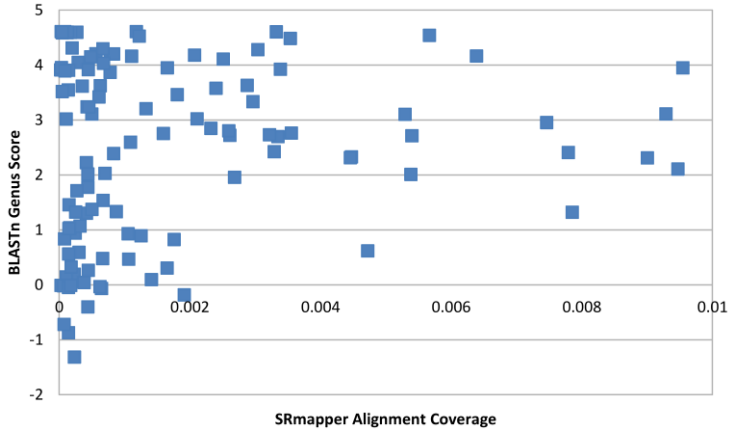

BLASTn Genus Score
5
4
3
2
1
0
-1
-2
0
0.002
0.004
0.006
0.008
0.01
SRmapper Alignment Coverage

Figure 4.7: BLASTn$_{genus}$ Scores Versus SRmapper Alignment Cover for Uniqueness Genomes in the Oral Metagenome. The reads from SRR331035 were aligned to the uniqueness oral metagenome using SRmapper with the -m 5 options and coverages on each of the uniqueness genomes were measured. BLASTn$_{genus}$ score for each species was determined by dividing the highest BLASTn query coverage for the genus from which the contigs aligned by the highest BLASTn query coverage for any other species outside that genus including those outside the oral metagenome. Panel A, B, and C focus on coverages ranging from 0% to 100%, 0% to 10%, and 0% to 1% respectively. For SRR331035, no SRmapper alignment coverages higher than 0.2% resulted in negative BLASTn$_{genus}$ scores, and this pattern held true for all samples analyzed.

of the sample being analyzed even though species from outside the oral metagenome were included in the BLASTn search. An alignment coverage of 0.2% or higher on the uniqueness genome for any species in the oral metagenome always resulted in the presence of the genus of that species being verified by the BLASTn analysis regardless of the sample being analyzed. Unfortunately, a correlation between the coverage of the uniqueness genomes and the corresponding numerical value of the BLASTn scores could not be identified. This is likely due to the variable amount of alignment overlap from other species mostly outside the oral metagenome. Thus, although the BLASTn query coverage was always high for species with coverages over a few tenths of a percent, the BLASTn scores varied due to varied scores in the denominator component of the score. Although a precise relationship between coverage and BLASTn score could not be obtained, using the BLASTn alignments was able to validate that the uniqueness genome demonstrated specificity for the correct species in identifying bacterial presence by alignment of reads in a sample using SRmapper. Since the coverages represented to correctly identify the species in question were also very low, the uniqueness genome method also demonstrated a high sensitivity.

The likelihood verifying the presence of a bacterial species or genus for SRmapper alignment coverages below 0.5% was also measured (**Table 4.1**). For species level detection, about 85% of coverages between 0.01% and 0.5% resulted in the verification of the species being present by using BLASTn and about an 80% verification rate for species level detection at coverages below 0.01%. Genus level detection fared slightly more favorably with over a 99% successful identification rate for SRmapper

SRmapper Coverage	Species Detection Rate	Genus Detection Rate
<0.01%	80.72%	85.88%
0.01%-0.1%	86.34%	87.04%
0.1%-0.5%	85.16%	99.22%
>0.5%	100.00%	100.00%

Table 4.1: Comparison between SRmapper Coverage Ranges and Bacterial Detection Rates. Datasets SRR331033, SRR331034, and SRR331035 were used to determine the correlation between SRmapper alignment coverages and how often the presence of the detected bacteria was verified by using BLASTn searches. Detection rate was determined by counting the number of bacterial species with SRmapper coverages in the selected ranges. Those with positive $BLASTn_{species}$ or $BLASTn_{genus}$ scores were considered to have been confirmed as present in the sample. All genus scores were higher than species scores since correctly identifying a species guaranteed the genus was correctly identified as well. However, correctly identifying the genus did not guarantee the species was correctly identified.

alignment coverages over 0.1% and higher than an 85% successful identification rate for SRmapper alignment coverages even lower than 0.01%.

Although these results are preliminary, they demonstrated two important points. First, they demonstrated that there were no species within the oral metagenome that were not viable for detection using the uniqueness genomes method developed. Although a small percentage of genomes had more than 80% of their bases similar to other genomes in terms of portions of the genome to which alignments could be generated to in an NGS experiment, none showed a prohibitive level of coverage where no portion or only an extremely small portion of the genome was available for use. A large majority of genomes demonstrated relatively minor coverage from all other species combined making them excellent candidates for detection using the uniqueness genomes method. Secondly although limited in number, the preliminary results from alignment of real datasets to the uniqueness oral metagenome demonstrated that hundreds of bacteria can be simultaneously detected using NGS technology, SRmapper, and the uniqueness genome methodology and that preliminary data suggested that thresholds can be set for coverage levels to warrant a confident detection or, for lower coverage levels, suggest the presence of bacterial species or genera. The work performed here was primarily limited by the lack of available whole-genome sequencing data for the oral metagenome. As the amount of available data continues to increase, better measurements of the ability of NGS to be used as a detection tool to test for the presence of bacterial species will become possible.

Chapter V

Conclusions and Future Directions

5.1 Overview

The research presented in this thesis has demonstrated several important advances in the processing of NGS data and the application of NGS as an agent for detecting the presence of bacteria in crude metagenomic samples. First, for the first time in several years, it was demonstrated that genome-hashing algorithms have the ability to outperform BWT-based algorithms in terms of alignment speed while retaining sensitivities similar to BWT methods. The source code for SRmapper was distributed freely as open source software to the academic community for use, analysis, or modification. Secondly, a method was developed which can be used to quickly identify the presence of TB in metagenome samples, and it was shown that use of this uniqueness genomes method both increases the sensitivity for confidently detecting TB at low sample loads while at the same time greatly reducing the rate of false positive alignments to the TB reference genome. Finally, it was demonstrated that this uniqueness genomes methodology has the ability to be applied to all genomes within the oral metagenome, and initial analysis demonstrated that combining the ability of SRmapper to simultaneously search through hundreds of references with the uniqueness genome methodology to reduce false positive alignments suggested that it was possible to simultaneously detect multiple bacteria within the oral metagenome. In this final chapter, a brief summary of the key findings from each portion of this thesis will be reviewed, and future directions for each area within this thesis will be discussed.

5.2 SRmapper

5.2.1 Implementation and Results

SRmapper was demonstrated to be between 2X-8X faster than the leading BWT-based alignment software, BWA, depending on alignment conditions selected. This significant speed increase was obtained while retaining similar sensitivity to BWA. SRmapper was shown to allow increased mismatches in a less time dependent manner than BWA thereby allowing for detection of reads with higher mismatch rates. Additionally, it was shown that SRmapper overcame the traditionally higher memory usage requirements of genome-hashing algorithms. At the time of its implementation, SRmapper was the only fast algorithm capable of performing pair-end alignment on a system with 4GB of total memory whereas BWA requires a higher amount of memory.

SRmapper has several features not available in any other alignment algorithms. Most importantly, it utilizes a probabilistic methodology to dynamically and automatically determine the number of mismatches allowed between the read and reference depending on read length, reference length, and desired alignment quality. Other alignment algorithms set a cap for mismatches depending solely on read length in the case of BWA or an attempt to retain a fast alignment speed in the case of bowtie. The probabilistic methodology in SRmapper removes any guesswork from the user on the number of mismatches that can be permitted to confidently generate alignments. Additionally, SRmapper allows for the outputting of unaligned reads to a fastq file to be analyzed by other programs. Although SRmapper is exceptionally fast and moderately sensitive, there is no single, perfect alignment algorithm. Unlike other algorithms, SRmapper readily accepts this fact and includes a built in method to use SRmapper in conjunction with other algorithms in the hopes that a slower but more sensitive algorithm

may be able to extract a small amount of additional information while not requiring all the extra time to align reads that can quickly be aligned by SRmapper.

5.2.2 Future Directions for SRmapper

Although SRmapper was carefully designed in terms of its implementation to make its methods as fast as possible, there are always multiple ways to implement the same algorithm, and it is not outside the realm of possibility that one or more of the routines in SRmapper was not implemented in the most efficient manner. Indeed, SRmapper has been through dozens of revisions designed solely with the intentions of increasing speed. These revisions have had varying degrees of success with many of them occurring before the release of SRmapper to the academic community. However, even after its initial release, revisions and optimizations to SRmapper have resulted in an in-house version of the algorithm that is approximately 20% faster than the publically available versions although constantly releasing update versions of SRmapper to the public was not deemed necessary. Although the more successful and important optimizations were described in detail, it was not feasible to describe in detail every attempt made to increase the performance of SRmapper.

Currently, the slowest portion of the algorithm is the extension of alignments by performing a base-by-base comparison. This comparison is made slower by the fact that the compressed reference has to be decompressed to compare the bases in the read to those in the reference. Several attempted optimizations have been performed on this portion of the algorithm with varying degrees of success. Currently the method used involves calculating the uncompressed sequence for the two bit bases to compare them to the bases in the read since a bitwise comparison method proved to be less efficient than

decompression. A plausible optimization likely to increase alignment speed would be to instead store all the possible decompressions for the 2-bit bases in a table similarly to the method employed to store probabilities for mismatches and hashes. This would exchange calculations for lookups, and since it is almost always faster to perform a lookup than a calculation, it would be expected that implementing this optimization would result in increased alignment speed. Alternatively, since memory usage is becoming less of an issue as computers continue to improve, SRmapper, or a large memory version of SRmapper, could be designed to not use a compressed reference genome. This would allow for a much faster extension portion of alignment at the expense of using more memory.

Additionally, SRmapper lacks a few of the features of the more popular alignment algorithms. The most important among these are gapped alignment with insertion/deletion detection and Smith-Waterman pair-end alignment. Since SRmapper does not require alignment of the entire read at the same time, implementing gapped alignment would theoretically not be very difficult with two obvious methods available. The first would be to use the index twice to search for seeds for alignment. In this method, if seeds could be found on each side of a gap, extension on each of these seeds could be performed. This method has the advantage of allowing for a gap of any size since the index can seed to any portion of the genome. One potential drawback of this method is that it requires two seeds, so alignments that have a high number of mismatches as well as a gap would likely be missed. The second method for gapped alignment is more traditional and likely would be easier to implement. Alignment would proceed as currently described for SRmapper until a read failed to be aligned with no

gaps. For those reads, the bases used in extension of the alignment would be shifted left or right on the reference to attempt to find a gapped alignment. Methods similar to this are in use and limit gap sizes to a few nucleotides. Larger gaps are not allowed using this method since shifting the bases in the read more times and comparing each of these shifts to the reference requires more time for alignment.

Smith-Waterman alignment is a method used to produce pair-end end alignments when only one of the pairs in an alignment can be aligned by the primary alignment algorithm. Smith-Waterman alignment uses local alignment to attempt to align the mate from a pair-end read that could not be aligned. This method is usually time-consuming but allows for the alignment of a small additional portion of the reads, thereby increasing the sensitivity of the alignment algorithm. Since SRmapper already stores the entire reference sequence in a compressed format, it is theoretically possible to perform a local alignment on mates in a pair-end read that cannot be aligned without Smith-Waterman alignment. Practically, this would involve either performing local alignment without the index to seed alignments or would allow alignments with a higher number of mismatches than normally allowed by the probability function of SRmapper. Allowing a higher number of mismatches has been deemed acceptable since it produces valid pair-end alignments. As read lengths in read qualities increase, the number of unaligned reads has decreased reducing the need for Smith-Waterman alignment. However, since a large portion of sequencing currently performed is pair-end sequencing, an additional method to increase the sensitivity of pair-end alignment is a useful feature to have.

Finally, several other algorithms take advantages of modern computing technologies. Many algorithms now allow parallel processing, and a few take advantage

of graphical processing units (GPUs) to increase alignment speed. As SRmapper stores a quarter of its index in memory at any one time and writes alignments from each quarter of the reference to a temporary file, it would be easily imaginable for a version of SRmapper to be created where each processor handles alignments to one quarter of the index. These alignments could be written to file and final alignments could be chosen from them in the same manner that is currently employed. This would result in nearly a four-fold increase in SRmapper alignment speed depending on how efficiently the parallelization process could be implemented.

5.3 Detection of TB in Oral Metagenomic Samples

5.3.1 Summary of Results

By comparing every possible 100 bp read that could be generated from bacterial species in the oral metagenome to the TB reference genome H37Rv, it was possible to create a version of the TB genome, the uniqueness TB genome, which contained only portions of the TB genome that were not similar to any other species in the oral metagenome. In the process of ensuring that variety in the human genome from individual to individual did not result in significant overlap between the human genomes and TB genome by analyzing 46Tb of DNA from the 1000 genomes project, it was discovered that several samples from the Finnish HapMap project were possibly contaminated with TB DNA. By developing software to simulate the results of a metagenomic sequencing project, it was determined that using the uniqueness oral metagenome reduced the rate of false positive alignments to the TB genome by an order of magnitude and using the uniqueness genome for alignment of metagenomic samples increased the ability of SRmapper to confidently detect TB at low loads for low

sequencing depths. Revisiting the samples from Finnish HapMap project from the 1000 genomes project, extensive evidence was provided that these samples were indeed contaminated with TB. High coverages of the uniqueness genome even under alignment conditions allowing no discrepancies between the reads and reference, proper correlation between the fraction of reads aligned to the full TB reference genome and uniqueness TB reference genome, and exceptionally high coverage rate of the uniqueness TB reference genome all strongly suggested that these samples were indeed contaminated with TB DNA. In terms of TB-negative samples, the usage of the uniqueness genome completely eliminated false-positive alignments in several samples known to not contain TB DNA. Although the complete elimination of false-positive alignments was not expected for all samples, the results again demonstrated the ability of the uniqueness TB genome to greatly reduce background noise due to similarities between the TB genome and other genomes within the oral metagenome. Finally, a few experiments were performed to provide preliminary evidence that NGS and SRmapper have the ability to detect TB at a subspecies level.

5.3.2 Future Directions in Detecting TB Using NGS and SRmapper

Although all the presented results are very encouraging and demonstrate the effectiveness of NGS, SRmapper, and uniqueness oral metagenome in detecting the presence or absence of TB in a crude sample containing DNA from many species within the oral metagenome, additional work would be necessary to transform this detection method into an effective and publically available detection tool or diagnostic method for TB infection. The largest current limitation to this method is a lack of available oral metagenomic sequences with known bacterial content to use. Although simulations

demonstrate that the detection limit for this method is quite low, this does not provide conclusive evidence that this method would work in the real world. To firmly establish the practicality of this method, collaboration with experimental groups would need to be established with the purpose of obtaining real samples with known patterns of bacterial presence and absence especially for samples positive for TB. Among these, it would be especially beneficial to be able to validate that the individuals from whom samples were generated for the Finnish HapMap project were indeed TB positive.

Although it was demonstrated that it was possible to detect TB in crude samples containing multiple bacteria from the oral metagenome, the abilities of NGS as a detection tool are expected to far exceed the ability to detect the presence or absence of a species. Since NGS results in a detection test at the nucleotide level, the possibility for using NGS to perform subspecies detection exists. This allows for the possibility of using NGS to not only provide a conclusive diagnostic for TB but to also identify either the strain of TB infection or the drug-resistance pattern in a TB infection. Again, the largest obstacle in the way of performing this type of analysis is the lack of available information on mutations resulting in drug resistance. Even for strains of TB with known resistance patterns, the genomic basis for these resistance patterns has not always been established. Additionally, the complete set of mutations resulting in drug resistance is unknown in part due to the fact not all mutations resulting in drug resistance are likely to have ever existed in nature and been observed through sequencing. Continued monitoring of the literature for drug resistance patterns in TB as well as establishing collaborations to sequence drug resistant strains of TB to identify additional mutations that result in drug

resistance will bolster the ability of NGS to be used for bacterial detection and could eventually shift NGS into a diagnostic tool for use in hospitals or clinics.

5.4 NGS as a Metagenomic Detection Tool for the Oral Metagenome

5.4.1 Summary of Results

Although certain genera in the oral metagenome contain many species, the method utilized to construct the uniqueness genome for TB was also successfully deployed to create uniqueness genomes for every species in the oral metagenome. Although the initial alignment settings, -q 6, proved to be too loose in terms of required similarity to leave significant portions of all genomes for alignment, reducing the number of mismatches allowed for a 100 bp read to be considered similar to a reference genome using the -m 5 option with SRmapper resulted in the successful formation of uniqueness genomes for all species within the oral metagenome. Using the -m 5 alignment conditions, nearly 70% of all genomes demonstrated greater than 90% uniqueness, and less than 5% demonstrated between 10%-20% uniqueness suggesting that all genomes within the oral metagenome can be detected using the uniqueness genomes methodology. Alignment to several real metagenomic samples suggested that regardless of sample or species within the oral metagenome, coverage of 0.5% or higher always resulted in the presence of the species being verified by BLASTn and a coverage of 0.2% or higher always resulted in the presence of the genus the species belongs to being verified. Lower coverages demonstrated that the uniqueness genomes methodology could still sometimes be used to verify the presence of a bacterial species or genus.

5.4.2 Future Direction in Detection of All Species from the Oral Metagenome

Preliminary results have demonstrated that the uniqueness genomes for the oral metagenome allow for the possibility that all species within the oral metagenome can be detected using uniqueness genomes methodology. Preliminary analysis of available data has provided theoretical evidence that this methodology does in fact correctly identify any species present in an oral metagenomic sample. However, and as with identifying TB via NGS, there is a lack of available oral metagenomic data available - especially data that has had its bacterial content evaluated by other means. These two factors have thus far precluded the establishment of a quantitative measure of confidence in detection of bacterial species within a metagenomic sample and have limited or detection methodology to a qualitative call. Thus, as in the case of using NGS to identify TB, furthering this technique would require experimental collaboration to acquire samples of known bacterial content. Assuming collaboration could be established, the first goal would be to attempt to establish a quantitative method by which the confidence in detection could be determined. This would be determined by collecting coverages on multiple samples positive and negative for each species within the oral metagenome. Ideally, samples containing different loads of various bacteria would be analyzed to monitor the effect of loads on coverage.

The long term goal of this project is to eventually establish methodology by which NGS can be used as a diagnostic tool for bacterial infection. Although the methodology developed and discussed in this thesis deals exclusively with the oral metagenome, there is potentially applicability for any human metagenome including the lung, skin, and gut metagenomes. For NGS to successfully be applied as a diagnostic tool, a fast, reliable, and relatively inexpensive method would need to be established.

Although not explicitly focused on in the proof of concept testing in this thesis, the NGS analysis methodology developed here lends itself well to be fast enough for use in a clinical or hospital setting, and the amount of DNA required to be sequenced for this method suggests that testing would be inexpensive. Given an alignment rate of 150Gb per day for SRmapper, alignment of a sample containing 100M nucleotides would take approximately one minute. Downstream of the SAM alignment file to determine loads and coverages on each species requires less than half a minute of computer time. Thus, the analysis performed would require approximately 90s for a 100Mb sequencing and approximately 15 minutes for a 1Gb sequencing. Using a cost of $0.10 per million nucleotides sequenced, performing the sequencing would cost $100 for 1Gb sequencing depth making it extremely feasible from a cost perspective. Thus, from a theoretical standpoint, using SRmapper and the uniqueness genome methodology makes NGS as a diagnostic tool feasible in terms of cost and the time necessary to perform analysis of a sequencing sample. Preliminary results suggest that this method also has the sensitivity required for diagnostics. Thus, future work should focus on more strongly establishing quantitative correlations between measurable parameters from alignment - load, coverage, and Shannon Entropy of alignments - and presence or absence of a bacterial sample. As mentioned earlier, this will require collaboration for the acquisition of large amounts of sequencing data from samples positive and negative for the range of species in the oral metagenome. **Figure 5.1** provides a hypothetical scheme of how NGS and SRmapper could one day be used as a diagnostic tool for bacterial infection.

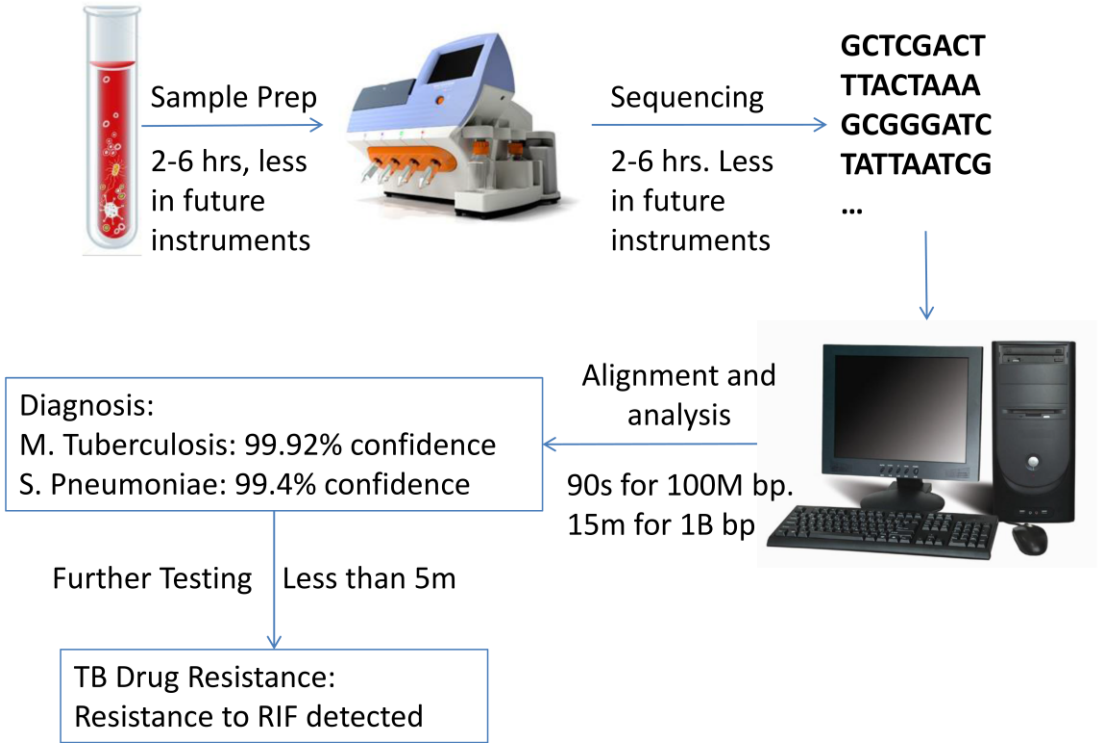


Figure 5.1: Theoretical Diagnostic Scheme for Using NGS to Diagnose Bacterial Infection. Current sequencing technologies can generate sequencing data in as little as two hours with another two to six hours of time needed for sample preparation. Both the time required for sequencing as well as the amount of sample preparation are expected to be reduced in the future. A hypothetical diagnostic scheme would start with a sample - saliva, sputum, blood, skin - being taken from a patient. Minimal preparation would be utilized with the purposes of generating sequencing data as quickly as possible. Analysis of sequencing data using SRmapper and downstream analysis software developed in-house have demonstrated that the analysis of this NGS data would require approximately 15 minutes of time at a sequencing depth of 1Gb and only 90s at a sequencing depth of 100Mb. Future analysis of metagenomic samples would eventually lead to being able to assigning confidence levels to any diagnosis. Further analysis of alignment data to certain species could be used to quickly determine known drug resistance patterns.

References

Agarwala, R., T. Barrett, J. Beck, D. A. Benson, C. Bollin, E. Bolton, D. Bourexis, et al. 2015. "Database Resources of the National Center for Biotechnology Information." *Nucleic Acids Research* 43 (D1): D6-D17.

Alkan, C., J. M. Kidd, T. Marques-Bonet, G. Aksay, F. Antonacci, F. Hormozdiari, J. O. Kitzman, et al. 2009. "Personalized Copy Number and Segmental Duplication Maps using Next-Generation Sequencing." *Nature Genetics* 41 (10): 1061-1067.

Altshuler, D. M., R. M. Durbin, G. R. Abecasis, D. R. Bentley, A. Chakravarti, A. G. Clark, P. Donnelly, et al. 2012. "An Integrated Map of Genetic Variation from 1,092 Human Genomes." *Nature* 491 (7422): 56-65.

Bankevich, A., S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, et al. 2012. "SPAdes: A New Genome Assembly Algorithm and its Applications to Single-Cell Sequencing." *Journal of Computational Biology*19 (5): 455-477.

Biesecker, L. G., W. Burke, I. Kohane, S. E. Plon, and R. Zimmern. 2012. "Next-Generation Sequencing in the Clinic: Are we Ready?" *Nature Reviews Genetics* 13 (11): 818-824.

Burrows, M., and D.J. Wheeler. 1994. "A Block-sorting Lossless Data Compression Algorithm." SRC Research Report 124. Pala Alto, CA, Digital Corporation.

Campagna, D., A. Albiero, A. Bilardi, E. Caniato, C. Forcato, S. Manavski, N. Vitulo, and G. Valle. 2009. "PASS: A Program to Align Short Sequences." *Bioinformatics* 25 (7): 967-968.

Chargaff, E., R. Lipshitz, and C. Green. 1952. "Composition of the Desoxypentose Nucleic Acids of Four Genera of Sea-Urchin." *The Journal of Biological Chemistry* 195 (1): 155-160.

Crick, F. 1970. "Central Dogma of Molecular Biology." *Nature* 227 (5258): 561-563.

David, M., M. Dzamba, D. Lister, L. Ilie, and M. Brudno. 2011. "SHRiMP2: Sensitive Yet Practical Short Read Mapping." *Bioinformatics* 27 (7): 1011-1012.

Desai, A. N. and A. Jere. 2012. "Next-Generation Sequencing: Ready for the Clinics?" *Clinical Genetics* 81 (6): 503-510.

Dewhirst, F. E., T. Chen, J. Izard, B. J. Paster, A. C. R. Tanner, W. -H Yu, A. Lakshmanan, and W. G. Wade. 2010. "The Human Oral Microbiome." *Journal of Bacteriology* 192 (19): 5002-5017.

Drobniewski, F. A., V. Nikolayevskyy, Y. Balabanova, D. Bang, and D. Papaventsis. 2012. "Diagnosis of Tuberculosis and Drug Resistance: What can New Tools Bring Us?" *International Journal of Tuberculosis and Lung Disease* 16 (7): 860-870.

Dunne, W. M., L. F. Westblade, and B. Ford. 2012. "Next-Generation and Whole-Genome Sequencing in the Diagnostic Clinical Microbiology Laboratory." *European Journal of Clinical Microbiology and Infectious Diseases* 31 (8): 1719-1726.

Eaves, H. L. and Y. Gao. 2009. "MOM: Maximum Oligonucleotide Mapping." *Bioinformatics* 25 (7): 969-970.

Eid, J., A. Fehr, J. Gray, K. Luong, J. Lyle, G. Otto, P. Peluso, et al. 2009. "Real-Time DNA Sequencing from Single Polymerase Molecules." *Science* 323 (5910): 133-138.

Ferragina, P., & Manzini, G. 2000. Opportunistic data structures with applications. Paper presented at the *Annual Symposium on Foundations of Computer Science - Proceedings*, 390-398.

Fire, A., S. Xu, M. K. Montgomery, S. A. Kostas, S. E. Driver, and C. C. Mello. 1998. "Potent and Specific Genetic Interference by Double-Stranded RNA in Caenorhabditis Elegans." *Nature* 391 (6669): 806-811.

Ford, C., K. Yusim, T. Ioerger, S. Feng, M. Chase, M. Greene, B. Korber, and S. Fortune. 2012. "Mycobacterium Tuberculosis - Heterogeneity Revealed through Whole Genome Sequencing." *Tuberculosis* 92 (3): 194-201.

Fournier, P. -E, G. Dubourg, and D. Raoult. 2014. "Clinical Detection and Characterization of Bacterial Pathogens in the Genomics Era." *Genome Medicine* 6: 1-15.

Galagan, J. E., P. Sisk, C. Stolte, B. Weiner, M. Koehrsen, F. Wymore, T. B. K. Reddy, et al. 2010. "TB Database 2010: Overview and Update." *Tuberculosis* 90 (4): 225-235.

Gey Van Pittius, N. C., J. Gamieldien, W. Hide, G. D. Brown, R. J. Siezen, and A. D. Beyers. 2001. "The ESAT-6 Gene Cluster of Mycobacterium Tuberculosis and Other High G+C Gram-Positive Bacteria." *Genome Biology* 2 (10)

Gnerre, S., I. MacCallum, D. Przybylski, F. J. Ribeiro, J. N. Burton, B. J. Walker, T. Sharpe, et al. 2011. "High-Quality Draft Assemblies of Mammalian Genomes from Massively Parallel Sequence Data." *Proceedings of the National Academy of Sciences of the United States of America* 108 (4): 1513-1518.

Gontarz, P. M., J. Berger, and C. F. Wong. 2013. "SRmapper: A Fast and Sensitive Genome-Hashing Alignment Tool." *Bioinformatics* 29 (3): 316-321.

Guo, J., N. Xu, Z. Li, S. Zhang, J. Wu, H. K. Dae, S. M. Mong, et al. 2008. "Four-Color DNA Sequencing with 3′-O-Modified Nucleotide Reversible Terminators and Chemically Cleavable Fluorescent Dideoxynucleotides." *Proceedings of the National Academy of Sciences of the United States of America* 105 (27): 9145-9150.

Hach, F., F. Hormozdiari, C. Alkan, F. Hormozdiari, I. Birol, E. E. Eichler, and S. C. Sahinalp. 2010. "MrsFAST: A Cache-Oblivious Algorithm for Short-Read Mapping." *Nature Methods* 7 (8): 576-577.

Hernandez, D., Tewhey, R., Veyrieras, J. -., Farinelli, L., Østerås, M., François, P., & Schrenzel, J. (2014). De novo finished 2.8 mbp staphylococcus aureus genome assembly from 100 bp short and long range paired-end reads.*Bioinformatics, 30*(1), 40-49

Hershey, A. D. and M. Chase. 1952. "Independent Functions of Viral Protein and Nucleic Acid in Growth of Bacteriophage." *The Journal of General Physiology* 36 (1): 39-56.

Hoffmann, S., C. Otto, S. Kurtz, C. M. Sharma, P. Khaitovich, J. Vogel, P. F. Stadler, and J. Hackermüller. 2009. "Fast Mapping of Short Sequences with Mismatches, Insertions and Deletions using Index Structures." *PLoS Computational Biology* 5 (9).

Homer, N., B. Merriman, and S. F. Nelson. 2009. "BFAST: An Alignment Tool for Large Scale Genome Resequencing." *PLoS ONE* 4 (11).

Hung, G. -C, K. Nagamine, B. Li, and S. -C Lo. 2012. "Identification of DNA Signatures Suitable for use in Development of Real-Time PCR Assays by Whole-Genome Sequence Approaches: Use of Streptococcus Pyogenes in a Pilot Study." *Journal of Clinical Microbiology* 50 (8): 2770-2773.

Kim, Y. J., N. Teletia, V. Ruotti, C. A. Maher, A. M. Chinnaiyan, R. Stewart, J. A. Thomson, and J. M. Patel. 2009. "ProbeMatch: Rapid Alignment of Oligonucleotides to Genome Allowing both Gaps and Mismatches." *Bioinformatics*25 (11): 1424-1425.

Köser, C. U., M. J. Ellington, E. J. P. Cartwright, S. H. Gillespie, N. M. Brown, M. Farrington, M. T. G. Holden, et al. 2012. "Routine use of Microbial Whole Genome Sequencing in Diagnostic and Public Health Microbiology." *PLoS Pathogens* 8 (8).

Lander, E. S., L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin, K. Devon, et al. 2001. "Initial Sequencing and Analysis of the Human Genome." *Nature* 409 (6822): 860-921.

Langmead, B. and S. L. Salzberg. 2012. "Fast Gapped-Read Alignment with Bowtie 2." *Nature Methods* 9 (4): 357-359.

Langmead, B., C. Trapnell, M. Pop, and S. L. Salzberg. 2009. "Ultrafast and Memory-Efficient Alignment of Short DNA Sequences to the Human Genome." *Genome Biology* 10 (3).

Lee, W. -P, M. P. Stromberg, A. Ward, C. Stewart, E. P. Garrison, and G. T. Marth. 2014. "MOSAIK: A Hash-Based Algorithm for Accurate Next-Generation Sequencing Short-Read Mapping." *PLoS ONE* 9 (3).

Leung, K., H. Zahn, T. Leaver, K. M. Konwar, N. W. Hanson, A. P. Pagé, C. -C Lo, P. S. Chain, S. J. Hallam, and C. L. Hansen. 2012. "A Programmable Droplet-Based Microfluidic Device Applied to Multiparameter Analysis of Single Microbes and Microbial Communities." *Proceedings of the National Academy of Sciences of the United States of America* 109 (20): 7665-7670.

Levene, H. J., J. Korlach, S. W. Turner, M. Foquet, H. G. Craighead, and W. W. Webb. 2003. "Zero-Mode Waveguides for Single-Molecule Analysis at High Concentrations." *Science* 299 (5607): 682-686.

Li, H. and R. Durbin. 2009. "Fast and Accurate Short Read Alignment with Burrows-Wheeler Transform." *Bioinformatics* 25 (14): 1754-1760.

Li, H. and R. Durbin. 2010. "Fast and Accurate Long-Read Alignment with Burrows-Wheeler Transform." *Bioinformatics* 26 (5): 589-595.

Li, H., J. Ruan, and R. Durbin. 2008. "Mapping Short DNA Sequencing Reads and Calling Variants using Mapping Quality Scores." *Genome Research* 18 (11): 1851-1858.

Li, H. 2012. "Exploring Single-Sample Snp and Indel Calling with Whole-Genome De Novo Assembly." *Bioinformatics* 28 (14): 1838-1844.

Li, R., Y. Li, K. Kristiansen, and J. Wang. 2008. "SOAP: Short Oligonucleotide Alignment Program." *Bioinformatics* 24 (5): 713-714.

Li, R., C. Yu, Y. Li, T. -W Lam, S. -M Yiu, K. Kristiansen, and J. Wang. 2009. "SOAP2: An Improved Ultrafast Tool for Short Read Alignment." *Bioinformatics* 25 (15): 1966-1967.

Li, Y., A. Terrell, and J. M. Patel. 2011. "WHAM: A High-Throughput Sequence Alignment Method.".

Lin, H., Z. Zhang, M. Q. Zhang, B. Ma, and M. Li. 2008. "ZOOM! Zillions of Oligos Mapped." *Bioinformatics* 24 (21): 2431-2437.

Liu, L., Y. Li, S. Li, N. Hu, Y. He, R. Pong, D. Lin, L. Lu, and M. Law. 2012. "Comparison of Next-Generation Sequencing Systems." *Journal of Biomedicine and Biotechnology* 2012.

Lunter, G. and M. Goodson. 2011. "Stampy: A Statistical Algorithm for Sensitive and Fast Mapping of Illumina Sequence Reads." *Genome Research* 21 (6): 936-939.

Margulies, M., M. Egholm, W. E. Altman, S. Attiya, J. S. Bader, L. A. Bemben, J. Berka, et al. 2005. "Genome Sequencing in Microfabricated High-Density Picolitre Reactors." *Nature* 437 (7057): 376-380.

Marshall, O. J. 2004. "PerlPrimer: Cross-Platform, Graphical Primer Design for Standard, Bisulphite and Real-Time PCR." *Bioinformatics* 20 (15): 2471-2472.

Maxam, A. M. and W. Gilbert. 1977. "A New Method for Sequencing DNA." *Proceedings of the National Academy of Sciences of the United States of America* 74 (2): 560-564.

McKernan, K. J., H. E. Peckham, G. L. Costa, S. F. McLaughlin, Y. Fu, E. F. Tsung, C. R. Clouser, et al. 2009. "Sequence and Structural Variation in a Human Genome Uncovered by Short-Read, Massively Parallel Ligation Sequencing using Two-Base Encoding." *Genome Research* 19 (9): 1527-1541.

Miller, J. R., S. Koren, and G. Sutton. 2010. "Assembly Algorithms for Next-Generation Sequencing Data." *Genomics* 95 (6): 315-327.

Ng, P. C. and E. F. Kirkness. 2010. *Whole Genome Sequencing*. Methods in Molecular Biology. Vol. 628.

Ning, Z., A. J. Cox, and J. C. Mullikin. 2001. "SSAHA: A Fast Search Method for Large DNA Databases." *Genome Research* 11 (10): 1725-1729.

Nirenberg, M., P. Leder, M. Bernfield, R. Brimacombe, J. Trupin, F. Rottman, and C. O'Neal. 1965. "RNA Codewords and Protein Synthesis, VII. on the General Nature of the RNA Code." *Proceedings of the National Academy of Sciences of the United States of America* 53 (5): 1161-1168.

Octavia, S., Q. Wang, M. M. Tanaka, S. Kaur, V. Sintchenko, and R. Lan. 2015. "Delineating Community Outbreaks of Salmonella Enterica Serovar Typhimurium by use of Whole-Genome Sequencing: Insights into Genomic Variability within an Outbreak." *Journal of Clinical Microbiology* 53 (4): 1063-1071.

Pareek, C. S., R. Smoczynski, and A. Tretyn. 2011. "Sequencing Technologies and Genome Sequencing." *Journal of Applied Genetics* 52 (4): 413-435.

Quail, M. A., M. Smith, P. Coupland, T. D. Otto, S. R. Harris, T. R. Connor, A. Bertoni, H. P. Swerdlow, and Y. Gu. 2012. "A Tale of Three Next Generation Sequencing Platforms: Comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq Sequencers." *BMC Genomics* 13 (1).

Robertson, K. D. and P. A. Jones. 2000. "DNA Methylation: Past, Present and Future Directions." *Carcinogenesis* 21 (3): 461-467.
Ronaghi, M., S. Karamohamed, B. Pettersson, M. Uhlén, and P. Nyrén. 1996. "Real-Time DNA Sequencing using Detection of Pyrophosphate Release." *Analytical Biochemistry* 242 (1): 84-89.
Rothberg, J. M., W. Hinz, T. M. Rearick, J. Schultz, W. Mileski, M. Davey, J. H. Leamon, et al. 2011. "An Integrated Semiconductor Device Enabling Non-Optical Genome Sequencing." *Nature* 475 (7356): 348-352.
Ruffalo, M., T. Laframboise, and M. Koyutürk. 2011. "Comparative Analysis of Algorithms for Next-Generation Sequencing Read Alignment." *Bioinformatics* 27 (20): 2790-2796.
Rumble, S. M., P. Lacroute, A. V. Dalca, M. Fiume, A. Sidow, and M. Brudno. 2009. "SHRiMP: Accurate Mapping of Short Color-Space Reads." *PLoS Computational Biology* 5 (5).
Sanger, F., S. Nicklen, and A. R. Coulson. 1977. "DNA Sequencing with Chain-Terminating Inhibitors." *Proceedings of the National Academy of Sciences of the United States of America* 74 (12): 5463-5467.
Sherry, N. L., J. L. Porter, T. Seemann, A. Watkins, T. P. Stinear, and B. P. Howden. 2013. "Outbreak Investigation using High-Throughput Genome Sequencing within a Diagnostic Microbiology Laboratory." *Journal of Clinical Microbiology* 51 (5): 1396-1401.
Simpson, J. T., K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, and I. Birol. 2009. "ABySS: A Parallel Assembler for Short Read Sequence Data." *Genome Research* 19 (6): 1117-1123.
Smith, T. F. and M. S. Waterman. 1981. "Identification of Common Molecular Subsequences." *Journal of Molecular Biology* 147 (1): 195-197.
Staden, R. 1979. "A Strategy of DNA Sequencing Employing Computer Programs." *Nucleic Acids Research* 6 (7): 2601-2610.
Voelkerding, K. V., S. A. Dames, and J. D. Durtschi. 2009. "Next-Generation Sequencing:From Basic Research to Diagnostics." *Clinical Chemistry* 55 (4): 641-658.
Watson, J. D. and F. H. C. Crick. 1953. "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid." *Nature* 171 (4356): 737-738.
Witney, A. A., K. A. Gould, C. F. Pope, F. Bolt, N. G. Stoker, M. D. Cubbon, C. R. Bradley, et al. 2014. "Genome Sequencing and Characterization of an Extensively Drug-Resistant Sequence type 111 serotype O12 Hospital Outbreak Strain of Pseudomonas Aeruginosa." *Clinical Microbiology and Infection* 20 (10): O609-O618.
Wilkins, M. H. F., A. R. Stokes, and H. R. Wilson. 1953. "Molecular Structure of Nucleic Acids: Molecular Structure of Deoxypentose Nucleic Acids." *Nature* 171 (4356): 738-740.
Woolley, A. T. and R. A. Mathies. 1995. "Ultra-High-Speed DNA Sequencing using Capillary Electrophoresis Chips." *Analytical Chemistry* 67 (20): 3676-3680.
Wu, R. 1970. "Nucleotide Sequence Analysis of DNA. I. Partial Sequence of the Cohesive Ends of Bacteriophage λ and 186 DNA." *Journal of Molecular Biology* 51 (3): 501-521.

Zerbino, D. R. and E. Birney. 2008. "Velvet: Algorithms for De Novo Short Read Assembly using De Bruijn Graphs." *Genome Research* 18 (5): 821-829.