Mapping Bisulfite-Treated Short DNA Reads

Jacob S. Porter

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Liqing Zhang, Chair
Lenwood S. Heath
Layne T. Watson
Hehuang Xie
Xiaowei Wu
Siu-Ming Yiu

March 19, 2018
Blacksburg, Virginia

Keywords: DNA read alignment, hairpin whole genome bisulfite, indels, bisulfite Ion Torrent, BisPin, BFAST-Gap

Mapping Bisulfite-Treated Short DNA Reads

Jacob S. Porter

(ABSTRACT)

Epigenetics are stable heritable traits that are not a result of the DNA sequence. Epigenetic modification of DNA cytosine plays a role in development and disease. The covalent bonding of a methyl group or a hydroxymethyl group to the 5-carbon of cytosine epigenetically modifies cytosine to 5-methylcytosine or 5-hydroxymethylcytosine. Upon PCR amplification, the bisulfite treatment of DNA converts unmethylated cytosine to thymine, while 5-methylcytosine, 5-hydroxymethylcytosine, and other bases remain unchanged. The resulting sequences can be mapped to a reference genome; however, this can be challenging due to sequencing technology complexity, low sequence complexity, and biases and errors introduced with bisulfite treatment. Once the short read is mapped, the identity of 5-methylcytosine or 5-hydroxymethylcytosine can be determined by comparing the mapped read to the aligned reference genome. Bisulfite DNA read mapping is characterized by mapping performance as low as 40%. This research improves bisulfite short read mapping quality. First, reads generated from the bisulfite hairpin PCR protocol are used to study mapping failure and solutions. A read may not map to the genome; it may map uniquely, or it may map to multiple locations. Sequence complexity correlates with these mapping categories. The hairpin protocol allows for a recovery, in some cases, of the original untreated read, and mapping this read with the regular read mapper Bowtie2 improved mapper performance by 10%. New bisulfite read mapping software called BisPin was created that calls BFAST (BLAT-like Fast Accurate Search Tool) for mapping. BisPin resolves ambiguously mapped reads with a rescoring strategy, which yields a statistically significant improvement. BFAST-Gap for Ion Torrent reads was developed, since Ion Torrent machines are less expensive than Illumina machines and since Ion Torrent reads are longer. There are few mappers for Ion Torrent data. BFAST-Gap uses homopolymer run length for contextual gap penalty functions, since homopolymer runs cause errors in Ion Torrent reads. In conjunction with BisPin, this software performed well on real and simulated bisulfite Ion Torrent data and Illumina data. InfoTrim, a read trimmer with an entropy term, was developed with competitive results.

Mapping Bisulfite-Treated Short DNA Reads

Jacob S. Porter

(GENERAL AUDIENCE ABSTRACT)

DNA, deoxyribonucleic acid, is a large molecule comprised of four molecular bases: adenine, cytosine, thymine, and guanine, and it determines heritable traits in living organisms. Sequencing DNA determines the sequential arrangement of bases. A read is a small sequence of DNA bases. Epigenetics are stable heritable traits that are not a result of the DNA sequence. Chemical groups called methyl and hydroxymethyl can be attached to cytosine. These groups are an epigenetic modification of cytosine, and they play a role in disease and development. The chemical bisulfite is used to discover these chemical groups. The bisulfite sequencing of DNA is a process where bisulfite is introduced to DNA, and then the DNA is sequenced. Bisulfite treatment converts cytosines without the methyl and hydroxymethyl chemical groups into thymine. Software is then used to align and match the resulting DNA strands to a large reference DNA strand called a reference genome to distinguish between cytosines that have these chemical groups. This process is called mapping or alignment, and its performance can be as low as 40% for bisulfite data. This research improves this performance. The hairpin protocol is a known bisulfite sequencing method that sequences two opposing DNA strands, where the original untreated strand can sometimes be recovered. Mapping the recovered strands improved performance by 10%. Using hairpin data, sequence complexity, a measure of DNA sequence randomness, correlated with mapping performance. BisPin mapping software was created that implements the hairpin recovery approach. BisPin rescores DNA strands that map to multiple locations on the reference genome, and it supports multiple sequencing technologies. BFAST-Gap, a modified mapping program callable by BisPin, uses a context sensitive function to better align Ion Torrent reads, which tend to have errors in regions of repeated bases. BFAST-Gap was developed, since Ion Torrent sequencing machines are less expensive than Illumina machines and since Ion Torrent reads tend to be longer and have more information. The read trimmer InfoTrim was developed to trim the lengths of short DNA sequences to improve the quality of alignments. These programs were validated on real and simulated DNA data and performed well.

I would like to dedicate this dissertation to my father, mother, brothers, and their families for their moral support while I went to graduate school for so long. Thanks: Steven Porter, Marla Porter, Elijah Porter, Benjamin Porter, and Tobias Porter.

Acknowledgments

The hairpin work was supported by the VBI new faculty startup fund for Hehuang Xie and by an NSF Award No. OCI-1124123 to Liqing Zhang. The hairpin and InfoTrim work were supported by the IEEE International Conference on Computational Advances in Bio and Medical Sciences student travel fund, the Virginia Tech Department of Computer Science travel fund, and the Virginia Tech Graduate School travel fund.

I would like to thank my committee for all of their helpful feedback.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

Epigenetics are stable heritable traits that are not a result of the DNA sequence [125]. There is an epigenetic modification of DNA cytosine where a methyl group, a hydroxymethyl group [69, 144], a formyl group [59, 112], or a carboxyl group [59] is covalently bonded to the 5 carbon of cytosine. Cytosine methylation was first proposed as an epigenetic mark when it was used to explain X chromosome inactivation in 1975 [52, 124]. The other covalent modifications of cytosine were discovered more recently, but there is evidence for their epigenetic effects [12]. Other epigenetic phenomena include post-translational histone protein modification [61, 141]. DNA is stored and ordered by wrapping it around histone proteins [113].

My research concerns epigenetic methylation. Methylation affects evolution and inheritance [2], embryonic development [88, 159], and cancer [4, 21]. Discovering where DNA methylation occurs can be accomplished with the sequencing of bisulfite-treated DNA. Bisulfite transforms unmethylated cytosine into uracil. The methyl or the hydroxymethyl group attached to cytosine sterically hinders the sulfite ion, preventing transformation of methylated and hydroxymethylated cytosine into uracil. The DNA is amplified with the polymerase

chain reaction (PCR), and the uracil is converted to thymine [39]. Differences between a reference genome and the bisulfite-treated DNA can be used to search for DNA methylation [18, 85].

To identify such differences, the most important step is to map the bisulfite-treated short reads to the reference genome. Mapping software takes a DNA sequence and finds a matching position for it in a reference genome. Many mapping programs suitable for bisulfite-treated short reads exist such as Bismark [70], BSMap [163], BiSS from NGM [31], BatMeth [81], and BS-Seeker2 [43]. These programs work by first finding a potential location for the read with a hash table or a string index and then computing a score for the read.

The score can be computed by counting mismatches and indels for the alignment between the read and the mapped location. Most programs use the Smith-Waterman algorithm [139] or the Needleman-Wunsch algorithm [103] to align the read and the portion of the reference genome that gives the mapped location. The best scoring location is reported as the location for that read. Despite continued and intense effort in creating new or improving existing bisulfite read mappers, the proportion of mapped reads remains low, commonly around 40% [70, 147], but sometimes as high as 80% [81, 147], which is much lower than regular short read mapping (e.g., data for the 1000 Genomes project has mapping efficiency $> 90\%$ [137]). The purpose of this study is to determine some causes and solutions to poor bisulfite read mapping efficiency.

1.1 Notation and Terminology

Terminology from formal language theory is used [25, 44]. Given an *alphabet* Σ, a *string* s is a finite sequence over Σ. An element in the alphabet is called a *character*. The *length of a string* s is denoted $|s|$, and it is the number of characters in the string. For DNA, the

alphabet is $\Sigma_{DNA} = \{A, C, G, T, N\}$, where A represents adenine, C represents cytosine, G represents guanine, T represents thymine, and N represents an indeterminate nucleic acid. A *read* is a string over this DNA alphabet. For example, $s = GGACTCTAC$ is a read. The character at position k of string s is given by $s[k]$. In the example, $s[3] = A$. A *mismatch* between string s_1 at position k and string s_2 at position m occurs when $s_1[k]$ does not equal $s_2[m]$. If the characters $s_1[k]$ and $s_2[m]$ are identical, then there is a *match*. Two strings s_1 and s_2 are *identical* if $|s_1| = |s_2| = l$ and $s_1[i] = s_2[i]$ for each i from 1 to l.

The set of all strings formed from characters in Σ is denoted Σ^*, and the empty string is denoted ϵ, which is in Σ^*. A *substring* is a contiguous portion of another string and is denoted $s[m : n]$, which starts at position m and ends at position n. A substring of length k is called a k-mer. A *subsequence* of the sequence s is an ordered collection of characters that can be derived from s by deleting some of the characters from s. For example, ACA is a subsequence of $GGACTCTAC$ but not a substring.

The *concatenation* of two strings s_1 and s_2 is denoted $s_1 s_2$ and has length $|s_1| + |s_2|$ and consists of the characters of s_1 followed by the characters of s_2. A string w is a *prefix* of a string x if $x = wy$ for some string $y \in \Sigma^*$. This can be denoted as $w = x[0 : k]$ for some $0 \leq k \leq |x|$. Note that if w is a prefix of x, then $|w| \leq |x|$. Similarly, we say that a string w is a *suffix* of a string x if $x = yw$ for some $y \in \Sigma^*$. The suffix w can be denoted $x[k : |x|]$. As with a prefix, this implies that $|w| \leq |x|$. For example, GGA is a prefix of $GGACTCTAC$ and TAC is a suffix of $GGACTCTAC$. The empty string ϵ is a suffix and a prefix for all strings.

A suffix array S of the string s is a sorted array of all suffixes of a string. It can be represented as an integer array where the integers in the array give the starting locations of the suffixes. For example, the suffix array of $GGACTCTAC$ is $[8, 3, 9, 6, 4, 2, 1, 7, 5]$, since AC is the first suffix in sorted order and $TCTAC$ is the greatest suffix. A *proper* prefix, suffix, or substring

of the string s is, respectively, a prefix, suffix, or substring that is not the entire string s [44].

The exact matching problem for a pattern string x and a reference string y seeks to find a position k in string y so that $y[k : k + |x|]$ matches x. Well studied algorithms for solving this problem include the Knuth-Morris-Pratt algorithm [67] and the Rabin-Karp algorithm [64]. Exact string matching can be done in time $\mathcal{O}(|x| + |y|)$. The related problem of string matching with *wildcards* involves introducing a *wildcard character* ϕ to the alphabet. If the pattern string x contains wildcards, then the wildcard exact matching problem finds any substring in y that matches x where the wildcard character can match any character in the alphabet. When the number of wilcards is bounded, then this can be done in time $\mathcal{O}(|x| + |y|)$ according to an algorithm in [44].

An *alignment* between strings s_1 and s_2 finds strings s_1' and s_2' over the alphabet $\Sigma \cup \{-\}$, and the $-$ character is called the *gap character*. The alignment is such that $|s_1'| = |s_2'|$. A substring of s_1 must be a subsequence of s_1', and a substring of s_2 must be a subsequence of s_2'. Deleting the gap characters from s_1' gives a substring of s_1, and deleting the gap characters from s_2' gives a substring of s_2.

A *gap* between s_1' and s_2' is a substring in either string consisting of one or more gap characters. An optimal alignment is an alignment that minimizes the number of mismatches and gaps between s_1' and s_2' with respect to some scoring function. Dynamic programming algorithms exist to find optimal alignments with respect to a scoring function. Such algorithms are the Smith-Waterman algorithm [139] and the Needleman-Wunsch algorithm [103]. The Smith-Waterman algorithm finds the best *local alignment*, which is a best scoring alignment between substrings of s_1 and s_2. The Needleman-Wunsch algorithm finds the best scoring *global alignment*, which is a best scoring alignment between the whole strings s_1 and s_2. These algorithms have time complexity $\Theta(|s_1||s_2|)$. Section 1.3.4 gives the Smith-Waterman algorithm, and Figure 1.4 from that section gives examples of alignments.

1.2 Biological Background

1.2.1 DNA is the Code For Life

DNA is the code for life, and it exists in all cells. DNA stands for deoxyribonucleic acid and consists of four bases or nucleic acids: guanine (G), adenine (A), thymine (T), and cytosine (C). DNA bases are arranged along a deoxyribose polymeric sugar backbone. The polymeric deoxyribose molecules are connected by a phosphate group. A nucelotide is a base plus a sugar molecule connected to a phosphate group. Each base is hydrogen bonded to a unique base, its complement, and together they are called a base pair (bp). Therefore, DNA consists of two strands. A sequence of DNA bases are on one strand, and the complementary bases are on the other strand. Adenine bonds with thymine, and cytosine bonds with guanine. The deoxyribose sugar of DNA has 5 carbons numbered 1 through 5. One strand starts at the 5' end and the opposing strand starts at the 3' end. Figure 1.1 gives a drawing of DNA [123, 154].

DNA is transcribed into messenger ribonucleic acid (mRNA), another molecule consisting of four bases: A, G, C, and U (uracil). An RNA nucleotide has a ribose sugar instead of the deoxyribose sugar of DNA. The messenger RNA is translated into a protein. Proteins serve structural, messenger, catalytic, and other functions, and a protein consists of a sequence of amino acids. The process of converting DNA into RNA is called *transcription*, and the process of converting RNA to proteins is called *translation*. A gene is a sequence of nucleotides that determines the sequence of amino acids in a protein or the sequence of a functional nucleic acid molecule. Transcribed DNA is not necessarily converted into proteins since the RNA itself may serve a functional purpose in the cell. For example, RNA can serve a regulatory role controlling gene expression [17]. DNA regions that code for genes are called coding regions, and DNA regions that do not are called noncoding regions. A DNA strand

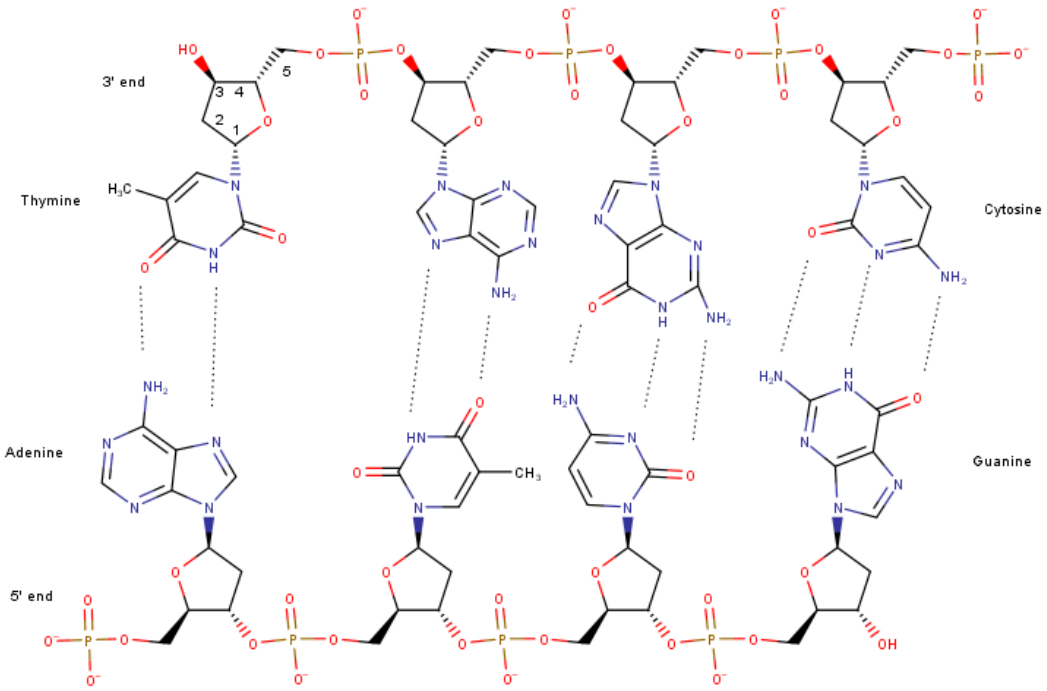


Figure 1.1: Chemical structure of DNA as a bond-line drawing. The carbon atoms are implicit in the lines and are saturated with hydrogen atoms. The numbers for the carbon atoms for the deoxyribose sugar is given in the upper left, and hydrogen bonds are indicated with a dashed line. The phosphate-deoxyribose backbone is on the exterior and the nucleotide bases are on the interior of the drawing. This was drawn with MarvinSketch 17.28.0 from ChemAxon (http://www.chemaxon.com).

can code for multiple proteins with alternative splicing, where different portions of the DNA are used for different proteins. DNA is replicated by a polymerase protein that synthesizes new DNA starting at the 5' carbon end and proceeding towards the 3' carbon end [26, 123].

One strand of a DNA molecule can be represented as a string over the alphabet Σ_{DNA}. One individual's DNA differs from another individual's DNA. A single mismatch between individuals' DNA strings is called a single nucleotide variant (SNV) [15, 113]. An indel is an insertion or a deletion of nucleotides in genomic DNA. Indels cause gaps in the alignment strings of strands of DNA from differing individuals. The study of SNVs and indels is useful in understanding disease, biological development, evolution, mortality, and the physical

structure of organisms [101, 160, 161]. Finding SNVs and indels relative to different DNA molecules reduces to problems in string matching.

Copying DNA can involve a protein enzyme called polymerase [146]. Polymerase proteins are often used in sequencing technology, discussed in Section 1.3.1. DNA of an offspring is inherited from the parent, but an offspring's DNA may have mismatches and indels relative to the parent's DNA. These can occur from copying errors from the polymerase protein and other phenomena. A genome is the complete set of DNA for an organism [113, 123].

1.2.2 Epigenetic Methylation

Cytosine may be methylated by a chemical reaction to produce 5-methylcytosine, and the methylation status of cytosine can be inherited by biological offspring. The average level of methylation varies between organisms — the worm *Caenorhabditis elegans* lacks cytosine methylation — while vertebrates have higher levels, with up to 1% of their DNA consisting of 5-methylcytosine [8]. Deanimation is a reaction involving water where an amine group (NH2) is removed from a molecule to produce ammonia (NH3). Despite the importance of 5-methylcytosine, it can deaminate to leave a thymine base, so methylated cytosines are prone to mutation [153]. This is a very common single nucleotide mutation [102].

Methylation of cytosine is an epigenetic mark relevant to both normal development and disease genesis. For example, it plays a role in X chromosome inactivation [66]. Finding where methylated and unmethylated cytosines exist in an organism's DNA is a challenging task that is often accomplished with bisulfite-treated short DNA reads that are mapped to a previously assembled reference genome. Bisulfite reacts with unmethylated cytosine to produce uracil while methylated cytosine remains the same. The reads are subjected to polymerase chain reaction amplification where uracil is transformed into thymine and other

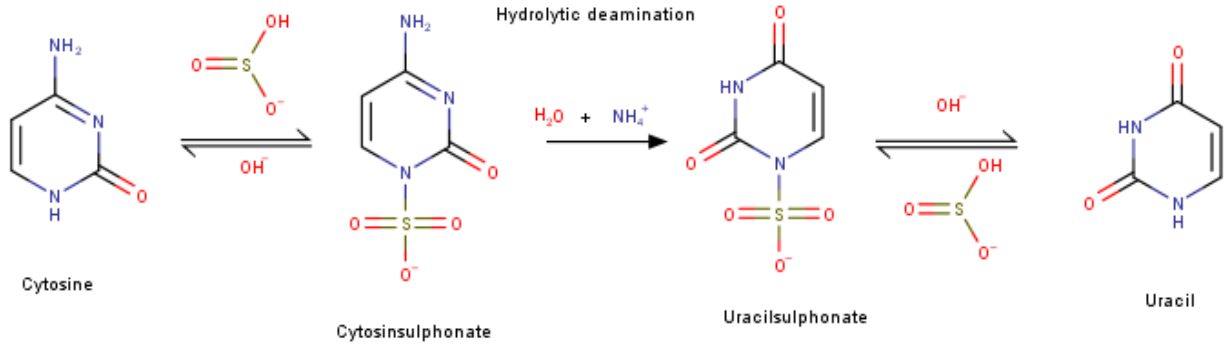


Figure 1.2: The chemical reaction involved in bisulfite treatment. Cytosine is converted into uracil through hydolytic deamination. Drawn with MarvinSketch 17.28.0 from ChemAxon.

bases remain the same. Analyzing where thymine on the read matches to cytosine on the reference genome gives evidence about the cytosine methylation marks of the organism being sequenced. See Figure 1.2 for a detailed diagram of the reaction that occurs with bisulfite treatment. Figure 1.3 gives a diagram of bisulfite sequencing.

1.3 Short Read Sequencing and Mapping

1.3.1 Sequencing Technology

Sequencing technology determines the order of nucleotides of DNA molecule fragments, and there are several generations of sequencing technology [50]. This involves *base calling*, which is the conversion of the sequencing signal to a base character. *Coverage* is the average number of times a particular base at a specific location has been sequenced. The base sequence of a single DNA fragment is called a *read*, and the number of bases in the read is called the *read length*. A read is a string over the nucleotide character alphabet Σ_{DNA}. Sanger sequencing (Applied Biosystems) was published in the mid-1970's, and the next generation is characterized by shorter reads from technologies by Solexa (bought by Illumina), 454 Life Sciences,

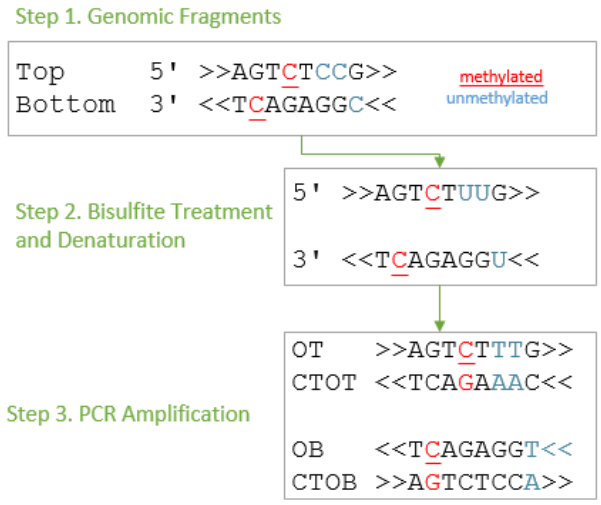


Figure 1.3: Bisulfite treatment is used to distinguish between methylated and unmethylated cytosine. Unmethylated cytosine is converted into uracil while methylated cytosines are not. Upon PCR amplification, uracil is transformed into thymine. The PCR amplification creates four distinct strands that show the effects of bisulfite treatment: OT (original top), CTOT (complement to the original top), OB (original bottom), and CTOB (complement to the original bottom). Sometimes the top strand is called the Watson strand, and the bottom strand is called the Crick strand. Different sequencing protocols (directional, nondirectional, and PBAT) will sequence a subset of these strands. Bioinformatics tools such as read mappers and methylation callers are used to identify methylated cytosine locations.

Technology	Average Length	Speed (Mb/h)	USD/Mb	Machine Cost	Error Type	Error Rate
Sanger	400-700bp	.66-.067	$500	$95k		0.001% - 1%
Illumina	100-200bp	55.5	$0.5	$100k-$600k	Substitution	0.1%
Ion Torrent	100-300bp	3.3-333.3	$0.63-$22.5	$80k	Indel	1-5 %
Pacific Biosciences	1500bp	50	$2	$695k	Insertions	13%
Oxford Nanopore	4000-7000bp	181.25	$0.06	$1000	Indel	2-13 %

Table 1.1: A summary of some commonly used DNA sequencing technologies [84, 87, 97, 99, 129, 132]. Sanger technology is slow with expensive base generation cost but more accurate. Illumina technology has intermediate speeds with low cost base generation but possibly high cost machines. Ion Torrent technology has low cost machines with longer reads and possibly fast speed. Pacific Biosciences technology has long reads but very high error rates. Ion Torrent technology has higher error rates for indels compared to Illumina technology.

and ABI SOLiD. This generation emerged in the mid-1990's and was commercialized around 2005 (454 Life Sciences) and 2006 (Solexa's Genome Analyzer). Ion Torrent sequencing was commercialized in 2010, and Pacific Biosciences first commercialized their technology in early 2011. The MinION machine of Oxford Nanopore Technologies was commercially released in 2015. The history and strategy of read sequencing technology is documented in the papers [50, 133]. Table 1.1 summarizes features of these technologies.

Most sequencing technologies consist of five steps [49]. First, the DNA molecule is randomly fragmented by enzymatic digestion, nebulisation, sonication, mechanical shearing, or hydrodynamic shearing into small molecules. These molecules range from tens of bases in early next generation sequencing to hundreds later on to thousands of bases long for Pacific Biosciences and Oxford Nanopore Technologies. Fragmentation can damage the ends of the DNA fragment by removing the 3' hydroxyl group or the 5' phosphate group. These groups are added back with a cocktail of enzymes, and this process is called end repair. Second, an adapter DNA sequence is ligated or attached to the fragment. The adapter may be a primer for amplification, sequencing, or immobilization. Third, the DNA fragment is immobilized by attachment to a solid surface such as a bead or a glass slide. Fourth, the DNA fragment is amplified, often with polymerase chain reaction (PCR), and many copies of the DNA

fragment are created and immobilized. Fifth, sequencing occurs where cycles of bases are incorporated by synthesis or ligation. After each cycle, bases are detected with base calling. Base identity inference software identifies each base at each cycle from colored light emission, pH change, or electrical induction, depending on the technology. This produces the DNA read [105]. Five sequencing technologies, Sanger, Illumina, Ion Torrent, Pacific Biosciences, and Oxford Nanopore are discussed in the following. Some sequencing technologies are uncompetitive, such as 454 Life Sciences technology, which has been discontinued [158].

Sanger

Sanger technology is the oldest of the sequencing technologies considered. It was developed by Frank Sanger in 1977 and is still used today [126]. It is slower and more expensive but more accurate than the other technologies considered, and it can sequence on average approximately 500 bases per read. It is used to sequence a locus of interest with high accuracy [99].

Illumina

Illumina Hi-Seq technology sequences small DNA reads, possibly around 100 bases, with modest error rates below one percent [128]. This technology is useful in whole-genome studies since it is faster and less expensive than whole-genome studies with Sanger technology [84].

Illumina technology works by binding the short DNA fragments to a flat glass substrate called a flow cell and then amplifying them with surface cluster PCR that creates clusters of the same DNA fragment. The polymerase protein is used to synthesize the complement of the DNA fragments, and the base is read by a camera based on the fluorescence of the cluster. There can be up to 10 million clusters per square centimeter on a flow cell [105].

Because Illumina technology uses the polymerase protein, it can have base incorporation errors [99]. There can be base calling errors from the fluorescence of the cluster, and these tend to occur more often in later base calling cycles [62]. Base calling error can be estimated and is documented in the PHRED score that is given along with the DNA string. The PHRED score is an alphanumeric string where each character encodes a probability that the base is called correctly [34].

Ion Torrent

Ion Torrent technology was commercialized in 2010, and sequencing machines cost approximately $80,000 in 2012, less than many other sequencing technologies [87]. The Ion Torrent Personal Genome Machine (PGM) is a lab machine for DNA sequencing that can be modularly upgraded with Ion Proton System semiconductor chips. Initially, sequencing runs produced reads with 100bp lengths. Recent chips produce reads with 400bp lengths [111]. Unlike Illumina and Pacific Biosciences, Ion Torrent technology uses hydrogen ion detection for base calling, while those other technologies use light detection. The chip contains the substrate and wells where the reaction chemistry occurs as well as the electronics for the hydrogen ion voltage sensor.

Ion Torrent technology works by immobilizing short DNA fragments on a bead. The bead is deposited into a well where the DNA fragment is amplified on the bead by emulsion bead PCR. There are millions of wells on a chip allowing for massively parallel sequencing. The DNA fragments are copied with polymerase, and every time a nucelotide is added into the copy, hydrogen ions are released. The well has a sensor on the bottom that detects the ion concentration through an electrical voltage. If a base is washed over the bead and is not incorporated into the fragments, then no ions will be released and the electrical voltage will be zero.

Nucleotides of the four bases are sequentially washed over the reaction well so that the sensor can detect which base causes a voltage change. The magnitude of the voltage gives the number of nucelotides of a particular base that have been added. For example, if two cytosines are added to the DNA fragment, then the voltage is doubled and two cytosines are called [134]. However, repeats of single bases (homopolymer runs such as TTTTT) cause high error rates of 1-5% [13]. The magnitude of the voltage is used to determine how many bases of the same kind are added in a single cycle; however, the voltage reading becomes less accurate with longer homopolymer runs. This introduces overcalling or undercalling of the length of the run causing error [134]. New bioinformatics tools, such as computational correction of indels from homopolymer runs, are needed for analyzing Ion Torrent reads.

Pacific Biosciences

Sequencing technology commercialized by Pacific Biosciences is called "single molecule real time sequencing." This technology can produce reads up to approximately 10,000 bases long [91]. However, error rates are high at approximately 15 percent, though errors are randomly distributed, allowing error correction with higher coverage [129]. This technology is used in rapid genome assembly of viruses and bacteria that have smaller genomes and is used to sequence full gene isoforms and splice variants [68]. This technology does not require an amplification step and takes advantage of the natural ability of DNA polymerase to incorporate ten or more nucleotides per second [134].

Oxford Nanopore

Oxford Nanopore [86, 94] sequencing technology has long average reads of 4-7 kilobases long, which have been used for previously difficult tasks such as sequencing a 50 kilobase

tandem repeat cluster assembly gap on human chromosome X at the location Xq24 [60]. This technology works by ratcheting bases from a DNA fragment through a bespoke protein nanopore. As bases pass through the pore, an ionic current change is induced, and this is used to identify DNA bases. This technology can sequence one strand (1D) as well as both sense (forward) and antisense (reverse complement) strands (2D) with a hairpin connector, which could be useful for error correction. The sequencing machines are very inexpensive compared to other machines, at this time, with the MinION having a price around one thousand US dollars. Unlike other sequencing technologies, Oxford Nanopore machines are portable, making them suitable for field work. The MinION is the size of a portable hard drive, and the SmidgION is not much bigger than a USB stick and works by attaching to a smart phone. The PromethION is a desktop device for high throughput, high sequence number projects [145].

1.3.2 DNA Sequencing for Methylation Calling

Sodium bisulfite sequencing introduces bisulfite into the sequencing process for distinguishing unmethylated cytosine from its methylated varieties, and it is presently the chief means for preparing DNA for methylation calling. The first recorded use of sodium bisulfite is found in 1992 [39], but this method was a labor intensive process unsuitable for whole genome processing; however, modern bisulfite sequencing derives from this method. Bisulfite sequencing has the same steps as regular sequencing, except that bisulfite is added prior to PCR amplification, and special adapters, primers, and enzymes may be used [150]. There are two types of PCR amplification: non-methylation specific PCR and methylation specific PCR [39, 51]. Methylation specific PCR uses pairs of primers that have both methylated and unmethylated cytosines [51].

There are several sequencing construction protocols for whole genome bisulfite-treated sequencing that have single-base methylation resolution. Figure 1.2 gives a summary of these technologies and compares them to the Oxford Nanopore sequencing technology. The newer Oxford Nanopore technology can directly distinguish between cytosine modifications since different functional groups attached to the cytosine will cause a different current signal as they are ratcheted through the nanopore [122, 136].

Construction strategies or protocols select DNA strands of different orientations from the PCR product of bisulfite-treated DNA. Exampes of strand orientations are given in Figure 1.3. There are four bisulfite-treated orientations: original top (OT), original bottom (OB), complement to the orignial top (CTOT), and complement to the orignial bottom (CTOT). One of the earliest sequencing protocols from 2007 is BS-Seq, which is sometimes called the Cokus protocol or the nondirectional protocol [23]. This protocol generates all four strand directions possible with bisulfite sequencing. The four strands of the BS-Seq protocol take more computational effort to align, and some alignments align a strand incorrectly since there is no way to distinguish the direction of a strand *a priori*. In my experience, BS-Seq is not used much anymore. A protocol published in 2008 called MethylC-Seq, also called the Lister protocol or the directional protocol, generates only the original top and the original bottom strands [82, 83]. In my observation, this is the most frequently used protocol.

In 2012, the post-bisulfite adapter tagging (PBAT) protocol was published [96]. Since the previous methods rely on global PCR amplification after bisulfite-treatment, they require micrograms of DNA to be effective, since bisulfite fragments DNA, but PBAT is an amplification free protocol that can be effective with subnanogram quantities of DNA. BS-Seq and MethylC-Seq tag DNA fragments with adapters, which are then bisulfite-treated, introducing fragmentation. PBAT fragments DNA with bisulfite treatment first, and then adapters are ligated to template DNA followed by sequencing. The PBAT method's global

Name	Year	Features	Strand Orientation
BS-Seq, nondirectional, Cokus [23]	2007	More computational effort in aligning	All directions
MethylC-Seq, directional, Lister [82, 83]	2008	Most popular today	Original top and original bottom
PBAT [96]	2012	amplification free, works with small DNA quantities	complement to the original top and complement to the original bottom
Oxford Nanopore [122, 136]	2017	distinguishes between cytosine modifications, does not use bisulfite	complementary strands are possible

Table 1.2: Features of three major protocols for bisulfite-treated whole genome sequencing for detecting methylcytosine. These protocols are compared with the sequencing technology Oxford Nanopore, which does not use bisulfite and can distinguish epigenetic cytosine modifications directly.

methylation rates correlate with MethylC-Seq [96]. PBAT produces the complement to the original top strand and the complement to the original bottom strand. My read mapper BisPin discussed in Chapter 4 works with the three construction strategies of directional, PBAT, and nondirectional.

There are several disadvantages to bisulfite treatment. Unmethylated cytosine will fail to convert into uracil at a frequency of approximately 1.1 percent [169]. Bisulfite treatment tends to reduce sequence complexity, which can make primer design more difficult and can be problematic for multiple PCR rounds [16]. BS-Seq, MethylC-Seq, and PBAT alone cannot discriminate between 5-methylcytosine and 5-hydroxymethylcytosine [58]. Bisulfite treatment causes DNA fragmentation, reducing sequencing yields, and multiple rounds of PCR amplification can introduce bias, since initial rounds may happen to copy certain fragments more frequently [96].

Recently, several strategies using bisulfite treatment have been published that can discriminate between different cytosine marks when used in conjunction with MethylC-Seq. These

strategies include Tet-assisted bisulfite sequencing (TAB-Seq [167]), oxidative bisulfite sequencing (oxBS-Seq [11]), and reduced bisulfite sequencing (redBS-Seq [12, 150]). These strategies work by oxidatively changing the mark on the cytosine. For example, TAB-Seq uses the ten-eleven translocation (TET) enzyme to convert 5-methylcytosine ultimately to 5-carboxylcytosine, while 5-hydroxymethylcytosine is unchanged. Bisulfite sequencing will convert the 5-carboxylcytosine into a T, while 5-hydroxymethylcytosine will remain a C [167].

Oxford Nanopore sequencing can call methylation marks directly and avoids the problems of bisulfite treatment [122, 136]. This technology can discriminate between 5-methyl, 5-hydroxymethyl, and 5-formyl cytosines since these functional groups cause a different ionic current when passing through the protein nanopore [122, 136]. Hairpin sequencing can be used, sequencing both the sense and antisense strands, but Oxford Nanopore technology can have higher error rates but longer reads than other technologies [60, 86, 94].

The first whole genome processing using hairpin BS-PCR was done in 2014 [71, 169]. This approach is notable since it uses paired-end sequencing with a hairpin connector between sense and antisense strands in a way that the identity of the direction of each strand is maintained. This allows for a sort of recovery of the original untreated strand by comparing the two strands [118], and it can help discriminate between certain SNVs and the effects of bisulfite. This approach can use MethylC-Seq. This approach was instrumental in my research and discussion of it can be found in Chapter 3, and a read mapper with special processing for this protocol is discussed in Chapter 4.

Reduced representation bisulfite (RRBS) is a method for studying regions with CG-rich regions (CG islands) [42] rather than the whole genome. It uses enzymatic digestion for fragmentation followed by DNA template enrichment. This method can be more efficient than whole genome sequencing if only CG islands are studied.

The following methods determine the level of methylation for a DNA region rather than determining methylation at single-base resolution. Methylated DNA immunoprecipitation (MeDIP) works by isolating methylated DNA by attaching it to an antibody designed to attach to methylated DNA. The methylated DNA is immunoprecipitated and separated from unmethylated DNA by attaching the antibody to magnetic beads. There are varieties of this method that work with microarrays [155] and with DNA sequencing techologies such as Illumina [32]. Methylation-sensitive high resolution melting analysis (MS-HRM) [157] works by measuring the amount of an intercalated fluorescent dye released upon temperature ramping of PCR amplicons. Neither of these methods use bisulfite.

1.3.3 Read Trimming

Prior to mapping and alignment, the DNA reads from the sequencing technology are often trimmed and filtered to improve the quality of successive analysis. Trimming selects a substring from the read based on quality criteria such as substring length and sequencing error [30]. Reads with N's, indeterminate bases, or reads contaminated with adapter or primer sequences, may be filtered out or removed. Adapters or primers are DNA fragments that are ligated to the read to separate or sequence the read [143]. Software to perform trimming includes Trimmomatic [10] and my own InfoTrim [114]. Trimming software often works with a simple linear-time algorithm that gives a score to each location in a read and then removes the 3' portion of the read at the score maximizing position. Trimming software and my InfoTrim algorithm are discussed in Chapter 5.

1.3.4 Regular Mapping Algorithms and Software

Once millions of reads are generated by various sequencing platforms, they need to be mapped to a reference genome to determine their likely locations. Many read mappers have been developed. Read mappers use a seed-and-extend approach to map short reads to a reference genome. The short read is a pattern string that is matched to a much longer reference string. First, a substring or a subsequence from the pattern read string and the reference string are exactly matched with either a hash table or the Burrows-Wheeler Transform. These indexes will be described later. These indexing methods are advantageous since they solve the exact matching problem in time linear in the length of the seed. The exact match from the seed gives candidate locations on the reference string, and these candidate locations are then aligned to the read pattern with an alignment algorithm such as the Smith-Waterman algorithm [139] or the Needleman-Wunsch algorithm [103].

The location with the best score is reported as the mapped location of the read. Many read mapping programs exist for regular reads, such as BFAST [55], Bowtie2 [72], GSNAP [162], and SHRIMP [28], and several read mappers for bisulfite-treated short reads exist, such as Bismark [70], Batmeth [81], BS-Seeker [20], and BSMAP [163]. The output of these read mapping programs is a SAM format file [80] that gives the read pattern string and the location on the reference string as well as other pertinent information such as the alignment score, the name of the read pattern string, and an alignment string.

Sometimes a read has alignments at multiple locations on the reference genome with the same best score. Such a read is called *ambiguous*, or it is called a *multiread*. A read should have a unique location, since the sequencing technology samples the read from a single location, so multireads present a challenge in deciding which location to report. Some read mappers like BFAST have the ability to report all such locations, and others like Bowtie2 report

only a limited number of locations. Methods for resolving multireads and assigning them to a unique location have been developed [148]. Some areas of the genome are naturally repetitious, and this presents difficulty in assigning a read to a unique location, since a read will match well to multiple locations. Examples are transposable elements [92].

Hash Tables

The hash table indexing strategy used by BFAST will be described. Other read mappers that use hash tables in a different way include BSMAP [163] and SHRIMP [28]. A *table* is a finite ordered sequence of integers $T[1,\ldots,m]$ where an index t from the integer set $\{1,\ldots,m\}$ retrieves the integer $u = T[t]$ associated with t. The table T stores m integers, so the cardinality of the table is denoted $|T| = m$. The integer $u = T[t]$ may represent a memory location where an object such as a string or a list is stored, or it could represent a number. A *list* is an ordered sequence of elements. A *hash table* is a table where a string $k \in \Sigma^*$, called the *key*, is mapped to a string $v \in \Sigma^*$, called a *value*. This association is done with a *hash function* $h : \Sigma^* \rightarrow \{1,\ldots,m\}$. The value v is associated to the integer $T[h(k)]$. The hash mapping may not be unique. When $h(k_1) = h(k_2)$ and $k_1 \neq k_2$, this is called a *hash collision*. Thus, multiple keys may map to the same hash table index, and to store multiple values to the same location in a hash table, a list is often stored at the location to store the multiple values. There are other ways to cope with collisions such as open addressing and a probe sequence [25]. Hash tables typically allow for amortized $\mathcal{O}(1)$ retrieval and storage, and they take $\Theta(m)$ memory. There are several ways of implementing hash tables and hash functions, and details can be found in [25].

In the context of read mapping, hash tables are used to implement exact string matching with wildcards as described in Section 1.1. This matches subsequences from a read string to the reference string to create a candidate location where the read may be aligned and mapped.

The read mapping program BFAST uses what it calls a "mask" to select a subsequence of a string starting at some position. The mask is a fixed width binary bit string such as "1100011." This string is compared to either the read or the reference at some position i, and if there is a one at the position where the strings match, then the character at that position is concatenated to the end of the string $q_{i,j-1}$ to form the string $q_{i,j}$. If there is a zero from the mask at a position, then the character at that position is ignored. This selects a subsequence q_i of the string. The algorithm 1 gives a procedure for doing this.

Algorithm 1 This algorithm takes a bit string mask x, a string y, and a location i on that string and returns a string q representing a subsequence corresponding to the bit string mask x applied to the string y at i. This algorithm has time complexity $\Theta(|x|)$ and space complexity $\Theta(|x|)$ when string indexing and concatenation are constant time operations.

```
1: x ← a bit string mask
2: y ← a string
3: i ← an integer that represents a starting location in y
4: procedure SubSeq(x, y, i)
5:     q_i ← the empty string
6:     for j ← 0, |x| − 1 do
7:         if x[j] == 1 then
8:             q_i ← the concatenation of q_i and y[i + j]
9:     return q_i
```

As an example, the result of evaluating SubSeq(011, ACGT, 2) is "GT."

The reference genome hash table T_g with hash function h_g is constructed by the following. For each position i in the reference genome, get string q_i from algorithm SubSeq from the bit mask associated with T_g. The string q_i and the position i are stored in a list at location $T_g[h_g(q_i)]$ in the hash table T_g.

To match a read to a location in the genome, the Algorithm 2, IndexLookup, is used. For a read, at given locations, the string r_j is constructed from the procedure given in algorithm 1, and the string r_j is checked against the hash table T_g to see if r_j matches some q_i. The string r_j can be efficiently matched to strings in the list in T_g by *binary search*, which can

be read about in [25]. If there is a match, then the position i is a candidate location for the read, and the next phase in read mapping, alignment, is performed. BFAST can use multiple indexes constructed from different bit string masks.

Algorithm 2 IndexLookup checks if subsequences from a read string are present in the reference genome using a hash table.

```
1: x ← a bit string mask.
2: T_g ← the hash table for the reference genome.
3: r ← a read string.
4: l_r ← a list of positions in r.
5: procedure IndexLookup(x, T_g, r, l_r)
6:     l_g ← the empty set. Used to store locations in the reference genome.
7:     for j ∈ l_r do
8:         r_j ← SubSeq(x, r, i)
9:         if T_g[h_g[r_j]] is not empty then
10:            Add the locations from the hash table to l_g
11:    return l_g, candidate matching locations.
```

The Burrows-Wheeler Transform

The second type of string index for the reference genome used in read mapping software is the FM-Index [36] with the Burrows-Wheeler transform (BWT) [14]. Both Bowtie [72] and BWA [77] use the FM-Index. Algorithm 3 gives an inefficient algorithm for constructing the Burrows-Wheeler transform and the suffix array. First, a character \$ that is lexicographically smaller than the other characters Σ in the string's alphabet is concatenated to the end of a string y. In the **for** loop on line 5, the string y is replaced with the concatenation of the last character and the prefix consisting of everything but the last character. This string is added to a set of tuples where each tuple has a string and the position of the suffix. On line 8, this set is sorted lexicographically based on the string. The loop on line 11 concatenates the last character of the sorted strings together into the string b, and the loop creates the suffix array S. These are the outputs of the algorithm. The strings on line 6 and the set t

of those strings do not need to be created since pointers to the original string can be used, and the sorting can be done efficiently so that the Burrows-Wheeler transform can be done in $O(|y|)$ time [14]. Suffix arrays are used for read mapping in other ways [135].

Algorithm 3 A simple inefficient algorithm for calculating the Burrows-Wheeler transform. This algorithm has time complexity $\mathcal{O}(n^2 \log n)$ and space complexity $\Theta(n^2)$ where $n = |y|$, the size of the input string.

```
1: y ← a string
2: procedure BWT(y)
3:     y ← y$ the concatenation of y with $
4:     t ← {} an empty set
5:     for i ← 0, |y| do
6:         y ← y[|y| − 1] · y[0 : |y| − 2]
7:         t ← t ∪ {(y, i)}
8:     Sort t by the first element, a string, for each tuple
9:     b ← ϵ the empty string
10:    S ← []
11:    for x ∈ t do
12:        b ← b · x[0][|y| − 1]
13:        S.append(x[1])
14:    return b, S, the Burrows-Wheeler transform and the suffix array
```

For example, the BWT of $y = ACAG\$$ can be found by first creating the sorted array

$$[\$ACAG, ACAG\$, AG\$AC, CAG\$A, G\$ACA]$$

with suffix position array $[5, 1, 3, 2, 4]$, and the BWT$(y) = G\$CAA$. An inverse operation exists for the BWT that reconstructs the original string. Notice that for a given suffix w in y, the BWT sorts all occurrences of the substring w in y together, and this is why it is useful in substring matching. The BWT is useful in compression since it will create long runs of the same character [3].

The FM-Index includes an array $R(a)$ for $a \in \Sigma$ that gives the number of characters in y that are lexicographically smaller than a (not including \$). For example, $R(C) = 1$ for

the example above since only A is smaller than C. The FM-Index also includes a two-dimensional occurrence array $Occ(a, i)$ that counts the number of occurrences of $a \in \Sigma$ in $\text{BWT}(y)[0 : i]$. For example, $Occ(A, 5) = 1$ in the example above since only one occurrence of A comes before position 5 in the $\text{BWT}(y)$. A string x can be matched to y by using the R and Occ arrays to retrieve the suffix array interval of S that matches x. Formulas for this are included in the BWA paper [77] and the FM-Index paper [36], and there are algorithms for computing the FM-Index and the BWT in $\mathcal{O}(|y|)$ time. Furthermore, exact substring matching can be done in $\mathcal{O}(|x|)$ time once the BWT is constructed [77].

The FM-Index allows for a time-memory tradeoff, since only a portion of the arrays Occ and S need to be stored, and the absent values are calculated when needed. BWA stores $Occ(\cdot, k)$ for every k that is a multiple of 128, and calculates absent entries of Occ by scanning the $\text{BWT}(y)$. BWA stores every 32nd entry in the suffix array S and uses a formula for calculating the absent entries when needed [77]. This formula uses the inverse compressed suffix array defined as $\Psi^{-1}(i) = R(\text{BWT}(y)[i]) + Occ(BWT(y)[i], i)$. By a result from the original FM-Index paper by Ferragina and Manzini, $S(k) = S((\Psi^{-1})^j(k)) + j$ where $(\Psi^{-1})^j$ means applying the operation to its own output j times [36].

A hash table string index implements exact string matching with wildcards in a manner where matching pattern strings takes linear time in the length of the pattern. The Burrows-Wheeler transform does not necessarily implement exact string matching with wildcards as efficiently because algorithms for doing this can involve excessive backtracking [73] and must try all possible matches at wildcard locations. This gives an advantage to using a hash table; however, the Burrows-Wheeler transform can match pattern strings of any length while hash tables are more limited in matching patterns of variable length. Using exact string matching with wildcards, called spaced seeds in read mapping literature, can improve sensitivity [79].

The Smith-Waterman Algorithm

After finding candidate locations in the reference string, read mappers use an alignment algorithm to score the locations. One such algorithm is the Smith-Waterman local alignment algorithm, a dynamic programming algorithm that builds a matrix where each cell of the matrix represents a subproblem in the alignment problem [139]. This is the algorithm that Bowtie2 uses [72]. The following is the canonical form of the Smith-Waterman algorithm. Read mappers may use a modified version of the algorithm.

Suppose the following is given.

- x, y are strings over the alphabet Σ
- $m = |x|$
- $n = |y|$
- $w(x, y)$ is a similarity function on the alphabet
- $H(i, j)$ – is the maximum similarity-score between a subsequence of $x[0 : i]$ and a subsequence of $y[0 : j]$
- W_i is the gap-scoring function at location i

The Smith-Waterman matrix H is therefore recursively defined in the following way.

$$H(i, 0) = 0,\ 0 \leq i \leq m$$

$$H(0, j) = 0,\ 0 \leq j \leq n$$

$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + \ w(x_i, y_j) & \text{Match/Mismatch} \\ \max_{k \geq 1}\{H(i-k,j) + \ W_k\} & \text{Deletion in } x \\ \max_{l \geq 1}\{H(i,j-l) + \ W_l\} & \text{Insertion in } x \end{Bmatrix}$$
$$1 \leq i \leq m, 1 \leq j \leq n$$

A cell contains the optimal score, and an optimal alignment can be found by backtracing starting from a highest scoring cell and going in the direction that the matrix was created until an entry is reached with value zero. Figure 1.4 gives an example of a local and global alignment. More details can be found in textbooks [44]. Mapping software may use fast, highly optimized SIMD (single instruction multiple data) implementations of these alignment algorithms [170]. SIMD instructions in modern CPUs are a form of data processing parallelism. Bowtie2, for example, uses the striped Smith-Waterman algorithm [35]. The Smith-Waterman algorithm has time complexity $\Theta(|x||y|)$ since it must fill in all the values for the H matrix. Storing all of the values in H requires memory complexity $\Theta(|x||y|)$, but there are algorithms for finding an optimal local alignment that have linear space complexity [35, 57].

1.3.5 Bisulfite Short Read Mapping Software

Bisulfite read mapping software uses the same seed-and-extend strategy as regular read mapping software, but available reference genomes are not bisulfite converted; thus, to match a string from a bisulfite-treated DNA fragment to the reference genome, special processing is used. Some read mapping software use a three base approach by converting all C's to T's or all G's to A's in the reference and read strings and performing regular read alignment

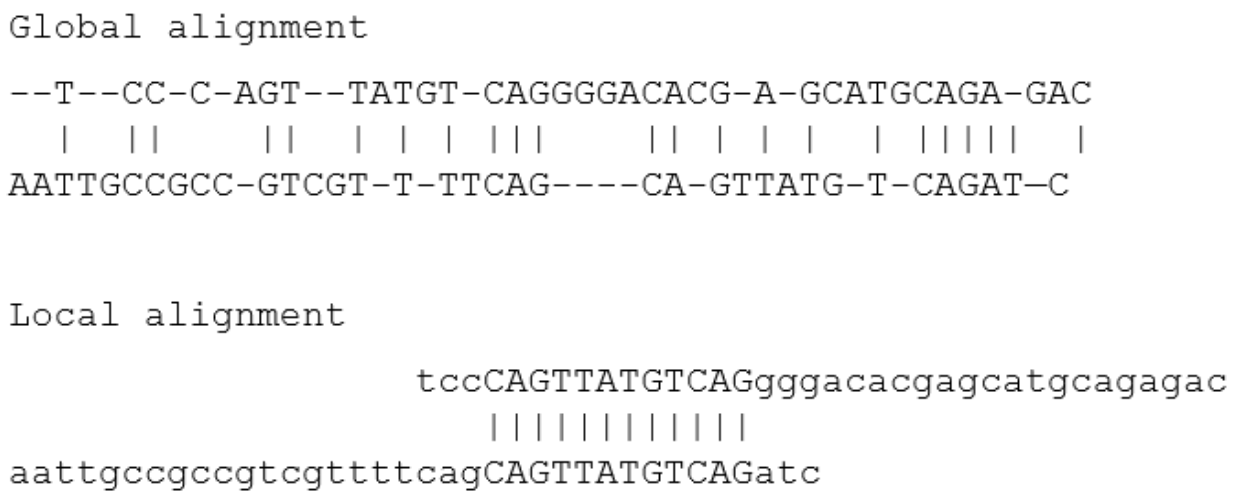


Figure 1.4: A global and local alignment. A global alignment aligns whole strings while a local alignment aligns substrings.

to the reference genome. Programs that do this include Bismark [70], BWA-Meth [110], and BRAT-BW [47]. This approach allows unchanged regular read mappers to be used with appropriate pre-processing and post-processing. Bismark and BS-Seeker use Bowtie2, and BWA-Meth uses BWA. The other approach is to use wild card matching. BSMAP, for example, enumerates all possible C-to-T (G-to-A) conversion combinations in the indexed seed [163]. Other software packages that use this approach include RMAPBS [138] and GSNAP [162]. CokusAlignment was an early mapper for the *A. thaliana* genome that used a suffix tree to search the genome [23], but suffix trees are unsuitable for large genomes since they have large memory requirements. Other bisulfite short read mappers include BatMeth [81], ERNE-BS5 [119], segemehl [106], BISS from NGM [31], Walt [19], and BSmooth [46]. The general probabilistic model presented in [65] could be used for bisulfite mapping with the appropriate arguments for modeling the probability of bisulfite conversion. RRBSMAP [164] and BS-Seeker directly support RRBS data. RRBS data can be aligned with whole genome bisulfite read mappers, but this can introduce error.

One challenge with bisulfite read mappers is computational complexity. Most read mappers

support multithreading or multiprocessing for speed. Bismark aligns the four strand orientations of BS-Seq data with four Bowtie2 processes with C-to-T and G-to-A converted reads each aligned against C-to-T and G-to-A converted genomes. This may require four copies of the genome index loaded into memory simultaneously. A BFAST index for the whole human genome can be 17 GB [55], and SHRIMP2 takes 48 GB [28]. In my experience, BWA and Bowtie2 indexes for the whole human genome take approximately 7-8 GB. Other challenges with bisulfite mapping involve reduced sequence complexity and sequencing bias and error due to bisulfite treatment [75, 100, 118].

A discussion and comparison of some bisulfite short read mapping software is found in Chapter 4 at Section 4.1 and in Table 4.1.

1.4 The Research Problem and Contributions

1.4.1 The Research Problem and Question

The READ MAPPING AND ALIGNMENT problem and the research question will now be formalized. Let the alphabet $\Sigma_{DNA} = \{A, C, G, T, N\}$. The characters A, C, G, T correspond to nucleic acids, while the character N is an indeterminate base used when the read sequencing technology is unable to determine the identity of the base. Let $x_i \in \Sigma^*_{DNA}$, then the set of m read strings is $X = \{x_1, \ldots, x_m\}$. Let $I = \{1, 2, 3, \cdots, |y|\}$, the set of all positions of the reference genome string y. Then $\mathcal{P}(I)$ is the power set of I, and this is the set of all sets of positions of the reference genome. The *mapping function* $f : \Sigma^*_{DNA} \to \mathcal{P}(I)$ maps a read string to a set of genome positions that the read maps to.

If L is the set of alignments between any two strings as defined in Section 1.1, then $\mathcal{P}(L)$ is the set of all sets of alignments. The *alignment function* $A_a : \Sigma^*_{DNA} \times \Sigma^*_{DNA} \times I \to L$

maps the read string, the reference string, and a location on the genome to an alignment. The *alignment scoring function* $A_s : \mathcal{P}(L) \rightarrow \Re$ maps a set of alignments to a real number, the alignment score. This function gives a way to choose which alignment is better. These functions, A_a and A_s, could be implemented in part by the Smith-Waterman algorithm. Let $a_{x_i} = \{A_a(x_i, y, l^j_{x_i}) \quad | \quad \text{for each} \quad l^j_{x_i} \in f(x_i)\}$ be the set of alignments of x_i and y from all of the locations in $f(x_i)$. Since A_s is defined on the set of all alignment sets, the function $A_s(a_{x_i})$ gives a score for the whole set of alignments of x_i. Note that $|f(x_i)| = |a_{x_i}|$. The Read Mapping and Alignment problem is the following.

> Read Mapping and Alignment
>
> Input: A set of read strings X, a reference genome string y, and the alignment scoring function A_s.
>
> Solution: A mapping f of reads to positions on the reference genome y, and an alignment function A_a such that for each $x_i \in X$ the alignments a_{x_i} are optimal with respect to the alignment scoring function A_s.

The problem of bisulfite-treated read mapping is a special case of the general Read Mapping and Alignment problem, since bisulfite-treated reads introduce variation relative to an untreated reference genome. Because of this variation, there are more challenges to this kind of read mapping problem.

To understand the quality of a solution, suppose there is a *quality function* $\mathcal{Q} : L \times \mathcal{P}(I) \times \mathcal{P}(I) \rightarrow \Re^{d_\mathcal{Q}}$ that maps the alignments, the alignment locations, and the correct alignment locations, if known, to a real number tuple, which implements some notion of alignment correctness. The *cumulative quality function* $\mathcal{C} : X \times y \rightarrow \Re^{d_\mathcal{C}}$ for mapped reads X and the reference genome y applies the quality function $\mathcal{Q}$ for each read $x_i \in X$ in some way and gives a real number tuple indicating the overall correctness of the mapping and alignments. The research question that my research addresses is the following.

> RESEARCH QUESTION: Which algorithmic and software improvements can be made to bisulfite-treated read mapping and other software such that the cumulative quality function $\mathcal{C}$ of the alignments is increased?

Pertinent to the research question are the *read mapping categories*: uniquely mapped, ambiguously mapped, filtered, and unmapped. Each read can be uniquely assigned to one of these categories. Let there be a threshold T for the alignment scoring function A_s. A read x_i is *uniquely mapped* if and only if $|a_{x_i}| = 1$ and $A_s(a_{x_i}) > T$. This means that the read mapping software returns a single unique location for the read and that the alignment quality is above some threshold. The read x_i is *ambiguously mapped* if and only if $|a_{x_i}| > 1$ and $A_s(a_{x_i}) > T$, and it is *filtered* if and only if $|a_{x_i}| \neq 0$ and $A_s(a_{x_i}) \leq T$. Read mapping software reports an ambiguously mapped read when it reports the read mapping to multiple locations and the alignment score at each location is above some threshold, and it is filtered when the alignment score falls below some threshold. The read x_i is *unmapped* if and only if $|a_{x_i}| = 0$, and this means that the read has no alignments. Not every read mapper will report all of these categories. Bismark, for example, does not distinguish between filtered and unmapped reads. My research mainly focuses on uniquely mapped reads since all reads have been drawn from a single location on the genome.

There are at least two cumulative quality functions used by my research. The first pertains to non-simulated real reads. Since the real location of the read is unknown, the cardinality of the locations given by the mapping function f together with the alignment scoring function A_s are used for the quality function. Let X_u be the set of uniquely mapped reads as determined by A_s and some threshold T, then the quality function for real reads is simply whether a read is uniquely mapped, and the cumulative quality function $\mathcal{C}_{real}$ for real reads is the proportion of reads that are uniquely mapped,

$$\mathcal{C}_{real}(X, y) = \frac{|X_u|}{|X|}.$$

A different cumulative quality function is used to assess the quality of simulated reads. For a simulated read x_i, the actual location is known since it was given by the simulation. Let X_q be the set of reads that are uniquely mapped to the correct location, then the quality function for simulated reads is whether the read is uniquely mapped to its correct location. To define the cumulative quality function for simulations $\mathcal{C}_{sim}$, precision q_p and recall q_r are defined. Precision is the proportion of correctly uniquely mapped reads of the uniquely mapped reads,

$$q_p = \frac{|X_q|}{|X_u|}.$$

Recall is the proportion of correctly uniquely mapped reads of all the reads,

$$q_r = \frac{|X_q|}{|X|}.$$

The cumulative quality function $\mathcal{C}_{sim}$ for simulated reads is the weighted harmonic average of precision q_p and recall q_r. This cumulative quality function is known as the F-measure, and it has a real number parameter β that gives a tradeoff between precision and recall. Typically, $\beta = 1$, and this is called the F1-score. Precision and recall equally contribute to the F1-score. The F-measure comes from information theory and document recall [37]. Formally, in this context, the β parameterized F-measure and cumulative quality function is given in the following:

$$\mathcal{C}_{sim}(X, y) = (1 + \beta^2) \cdot \frac{q_p \cdot q_r}{(\beta^2 \cdot q_p) + q_r}.$$

When the F1-score is maximized at 1, then all the reads are uniquely mapped to their correct locations, when it is 0, none of the reads are mapped to their correct locations. There can be other ways of defining the quality functions. Perhaps ambiguously mapped reads can be incorporated. Another cumulative quality function for simulated data is simply the tuple of precision and recall. The goal of my research is to find methods and algorithms to increase $\mathcal{C}_{sim}$ on simulations and $\mathcal{C}_{real}$ on real data with bisulfite-treated DNA read data.

1.4.2 Workflow

Figure 1.5 gives a flow chart for the data, processes, and analyses that have been conducted in my research and that are implicated in the READ MAPPING AND ALIGNMENT problem. The initial step is data gathering, and I got reference genomes and DNA sequence data from publicly available databases such as the NCBI trace archive (https://trace.ncbi.nlm.nih.gov/Traces/). Chapter 5 discusses Step 1, read trimming. The main focus of my research is Step 2, read mapping and alignment, and Chapter 3 and Chapter 4 discuss my contributions to Step 2. Assessing the accuracy of read mapping and alignment is done as discussed in Section 1.4.1. Step 3, variant calling, is discussed in Chapter 2 where indel calling pipeline accuracy is discussed.

1.4.3 Motivation

This research addresses several challenges and observations pertinent to the research question. There are varieties of bisulfite sequencing technologies such as BS-Seq, MethylC-Seq,

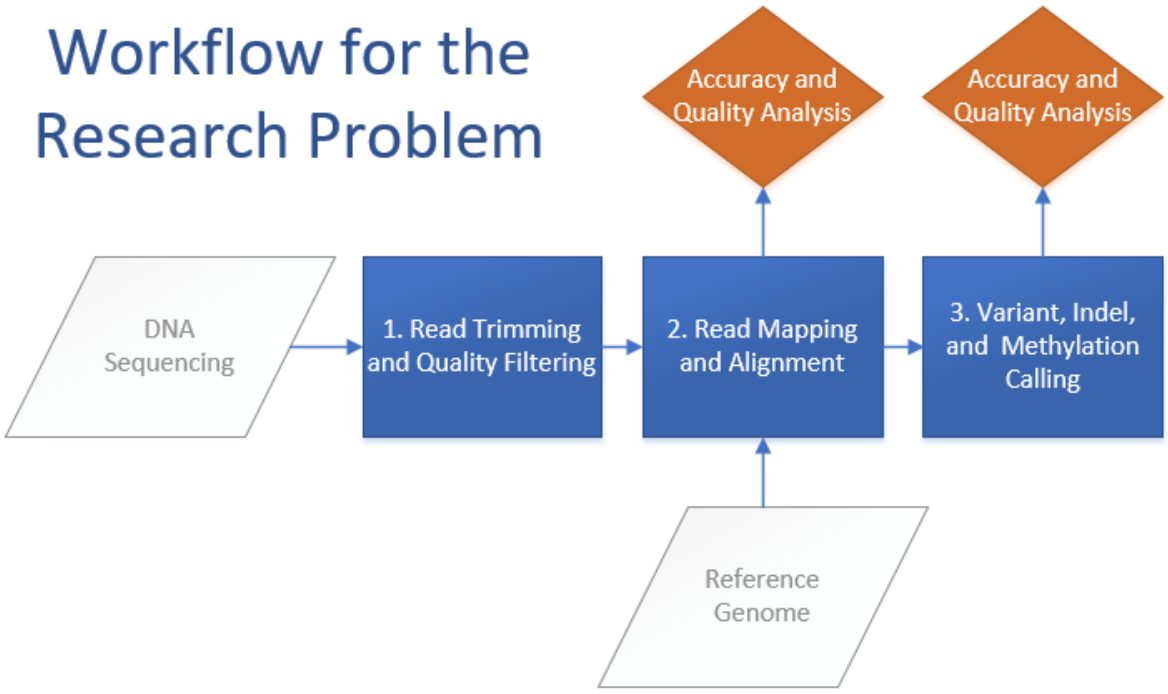


Figure 1.5: The workflow for the research problem.

and BS Hairpin PCR, and there are paired-end and single-end layouts. This creates complexity in mapping, since each technology may require special processing. Bisulfite-treatment tends to reduce sequence complexity, and less complex reads are more difficult to align [116]. Sequencing platforms differ in their error profiles, expense, and sequence length. Ion Torrent reads, for example, tend to introduce erroneous indels, but can be longer and less expensive to produce than Illumina reads [87]. There are few mappers specifically designed for Ion Torrent bisulfite data [108]. This provides an incentive to develop a new read mapper for bisulfite-treated Ion Torrent reads.

Raw reads generated by the sequencing machines can have varying quality, and quality trimming can improve mapping performance [147]. I developed InfoTrim, a quality trimmer, discussed in Chapter 5. My research implements strategies to cope with these challenges.

1.4.4 Contributions

This research has four main contributions. All of the work was done with Liqing Zhang as my advisor. First, Chapter 2 discusses the performance of indel calling pipelines on larger indels from regular reads. An indel calling pipeline is a read mapper with an indel caller. This study found that the read mapper BFAST performed well with a variety of indel calling software. This work was done with Jonathan Berkhahn, who performed some of the simulations and wrote part of the introduction of the paper [117].

Second, in Chapter 3, hairpin bisulfite PCR sequencing is used to identify problems and solutions with bisulfite read mapping. Sequence complexity was found to correlate with mapping category performance, and the hairpin recovery strategy, where the original untreated read was recovered, was found to improve mapper performance. This work was done in conjunction with biologists Ming-an Sun and Hehuang Xie [118], who provided data.

Third, motivated from this, the bisulfite short read mapping Python software BisPin was created by me. BisPin calls BFAST as a subprocess. The choice of using BFAST was motivated from the indel calling study from Chapter 2. It was thought that BFAST would handle Ion Torrent reads well since Ion Torrent reads tend to introduce gaps in homopolymer repeats. BisPin implements a rescoring method for ambiguously mapped reads based on differences in transition and transversion mutation frequencies. It supports BS-Seq, MethylC-Seq, PBAT, and the bisulfite hairpin PCR construction protocols, and it supports both single-end and paired-end layouts. It performs well on real and simulated Illumina and Ion Torrent reads.

For Ion Torrent reads, an algorithmic strategy that reduces the gap open and gap extension penalties in accordance with a logistic function in the length of the homopolymer run of a read was implemented in the form of BFAST-Gap, a forked version of the BFAST read mapping program. This strategy addresses the higher indel error rate of homopolymer runs in Ion Torrent sequencing technology. In conjunction with BisPin, this mapper can be used to align bisulfite-treated Ion Torrent reads. This mapper showed good performance with real and simulated bisulfite Ion Torrent reads. BisPin and BFAST-Gap are discussed in Chapter 4, which corresponds to the paper [115].

Fourth, Chapter 5 discusses InfoTrim, a read quality trimmer written in Python. Trimming is the process of shortening the raw DNA reads for quality. Trimming can contribute to mapper alignment performance. Motivated from the hairpin data work in Chapter 3, InfoTrim adds an entropy term to the Trimmomatic model. InfoTrim performed reasonably well in improving mapper performance on real and simulated bisulfite-treated short DNA reads. This work corresponds to the paper [114].

1.4.5 Academic Papers

The following is a list of my academic papers germane to this research.

1. Jacob Porter, Liqing Zhang. "BisPin and BFAST-Gap: Mapping Bisulfite-Treated Reads." biorxiv. 2018. doi: 10.1101/284596. 26 pages.

2. Jacob Porter, Liqing Zhang. "InfoTrim: A DNA Read Quality Trimmer Using Entropy." ICCABS 2017. 2 pages.

3. Jacob Porter, Ming-an Sun, Hehuang Xie, and Liqing Zhang. "Investigating Bisulfite Short-Read Mapping Failure with Hairpin Bisulfite Sequencing Data." BMC Genomics, 16(11):S2. Nov 2015.

   Jacob Porter, Ming-an Sun, Hehuang Xie, and Liqing Zhang. "Improving Bisulfite Short-Read Mapping Efficiency with Hairpin-Bisulfite Data." ICCABS 2014. 2 pages.

4. Jacob Porter, Jonathan Berkhahn, and Liqing Zhang. "A Comparative Analysis of Read Mapping and Indel Calling Pipelines for Next-Generation Sequencing Data." Emerging Trends in Computational Biology, Bioinformatics, and Systems Biology. Edited by Hamid Arabnia and Quoc Nam Tran. Elsevier. Chapter 29. 2015. pages 521 - 535.

   Jacob Porter, Jonathan Berkhahn, and Liqing Zhang. "A Comparative Analysis of Computational Indel Calling Pipelines for Next Generation Sequencing Data." BIOCOMP 2014. 6 pages.

5. Hong Tran, Jacob Porter, Ming-an Sun, Hehuang Xie, and Liqing Zhang. "Objective and Comprehensive Evaluation of Bisulfite Short Read Mapping Tools." Advances in Bioinformatics. 2014. 11 pages.

Chapter 2

Indel Calling Pipelines

2.1 Introduction

Indels are the second most common class of mutation in the human genome [101]. They consist of an insertion or deletion of one or more DNA bases into a genome. This can have far ranging effects concerning gene expression and genetic disease [101]. Detecting and identifying indels is a two step process that can introduce error at every step. Starting with a set of DNA sequence reads and a reference DNA genome, reads are first mapped to the reference genome with a mapping program, and then the mapped results are inputted into indel calling software to identify indels.

A growing variety of software is available for read mapping and indel calling. New tools are constantly being developed with an eye towards better performance and increased accuracy. As the variety of available tools and the complexity of the technologies involved in indel calling increases, it becomes increasingly important to understand the relationships between mapping software and indel calling software. While previous work [104, 107] assessed the

accuracy of indel calling by only varying mapping software or by only varying indel calling software, we studied the accuracy of mapper and indel calling software combinations. We evaluated the accuracy of pipelines consisting of four popular mapping programs and three indel calling programs on simulated data based on a portion of human chromosome one. For mapping, BFAST [55], Bowtie2 [72], BWA [77], and SHRIMP [28] were selected. For indel calling, we used Dindel [1], Freebayes [41], and SNVer [156]. We varied the coverage of the reads inputted to the mappers to study the effects of different levels of read coverage on the precision and recall of called indels. We evaluated the accuracy of these pipelines on indels from 1-30 bases long.

Furthermore, pipeline accuracy was assessed with real human data from chromosome 22. The indels were validated with an alternate method described in the paper [95].

The remainder of this Chapter is organized as follows. Section 2.2 discusses the read mapping software and the indel calling software that we selected. Section 2.3 discusses the methods that we used to generate our simulated and real data sets and the statistics that we used to evaluate the accuracy of our pipelines. Section 2.4 discusses the accuracy results of the pipelines and the runtime on real data. Section 2.5 concludes.

2.2 Mapping and Calling Software

Read mappers use a seed-and-extend approach to mapping short reads to a reference genome. A string index such as the Burrows-Wheeler Transform or a hash table is used as a string index. Typically, small portions of sequences from the read are matched to the reference genome string index. Next, different locations on the reference genome, for which the read is mapped from the seeding phase, are extended with an alignment algorithm such as the Smith-Waterman algorithm or the Needleman-Wunsch algorithm. These algorithms perform

local and global alignment respectively. Local alignment finds the most optimal similarity score between strings (DNA sequences) over all possible string lengths. Global alignment finds the optimal similarity over the entire strings. These algorithms include insertions, deletions, and point mutations, so they are ideal for DNA sequence alignment. The location with the best similarity is reported as the location to map the read. More detail on the software and algorithms used in read mapping can be found in Section 1.3.4.

2.2.1 Indel Calling Software and Models

Indel calling software uses probabilistic models based on the coverage of the reads at a locus to determine if an indel should be called. The SNVer indel caller uses a binomial distribution on the reads that cover a location to detect variation [156], and Freebayes uses Bayesian inference and a multinomial distribution. Freebayes includes a prior distribution on genotype frequencies, and it incorporates sequencing error [41]. Dindel uses Bayesian inference as well. The indel callers use similar probabilistic inference methods where coverage and sequencing error are incorporated into the model.

Dindel was one of the best performing indel callers in this study, and it has been used in the 1000 genomes project [1]. Dindel works in the following manner. First, candidate SNVs and indels are repositioned to canonical positions and grouped into realignment windows of 120bp, where a read overlaps with at least 20bp in the alignment window. In the Dindel model, a haplotype is a sequence of mismatches or gaps (indels) relative to the reference genome. One such sequence could be the true haplotype. An example haplotype is an AT mismatch at position 2 followed by a CAT deletion at position 10.

For each window, candidate haplotypes H_j are generated and each candidate haplotype for the window will be considered against the same set of reads R_i. Candidate haplotypes

are generated from the alignment file but can be created by the user. Candidate indels, which come from the user or from the alignment file are inserted into each of the candidate haplotypes. Haplotypes with the highest frequency of mismatches and gaps (indels) from the reads are chosen, and Dindel defaults to eight candidate haplotypes for computational efficiency. The haplotype posterior probability given the read data, $P(H_j|R)$, is estimated with Bayes' theorem, $P(H_j|R) \propto P(R|H_j)P(H_j)$ where $P(H_j)$ is the prior probability of the candidate haplotype, and $P(R|H_j)$ is the likelihood function [1].

The likelihood function $P(R|H_j)$ incorporates base-call reliability with base quality scores from the sequencing technology, read mapping quality, and indel sequencing error given specific sequence context. The likelihood model uses something similar to a profile Hidden Markov Model with "hidden parameters" that give the position and length of an insertion or a deletion [33, 166]. Probabilities with these hidden parameters are computed, and then the maximum likelihood is estimated with the Viterbi algorithm [121, 152]. The benefit of the hidden parameters is that they give an alignment for a read concerning deletions and insertions so that indel identity and position can be output [1].

2.3 Methods

2.3.1 Software Workflow

We selected mapping software that was both widely used and that covered a variety of different algorithms. Bowtie2 [72] and Burrows-Wheeler Aligner (BWA, [77]) are both popular tools that use the Burrows-Wheeler transform to map reads. SHRIMP [28] and BFAST [55] are both hash-based mapping tools. SHRIMP creates a hash table index of the read sequences, but BFAST creates a hash table index of the reference sequences. For indel calling,

we selected two programs that use Bayesian statistics, Dindel [1] and Freebayes [41]. SNVer [156] is based on a frequentist model developed by the SNVer authors and reports p-values for variant calls.

All of our experiments were run on SystemG nodes. SystemG is a research cluster at Virginia Tech with the Computer Science department. Each node had two quad-core 2.8 GHz Intel Xeon processors and 8 gigabytes of RAM. The mappers were run with four threads when possible, but the indel callers were single-threaded only.

Each tool was run with default settings since that is how the tools will most likely be used. The workflow consisted of creating SAM files from each read-mapper and then transforming the SAM files into BAM files with samtools [80]. Finally, each indel-caller produced a VCF file from the BAM files. The following are the version numbers for the software: bfast-0.7.0a, bowtie2-2.1.0, bwa-0.7.1, SHRiMP-2.2.3, dindel-1.01, freebayes-0.9.9, and SNVer-0.4.1. The real data workflow was similar to the simulated data workflow. The differences are that Shrimp was run with –no-qv-check and –qv-offset 33 because the real data were Sanger traces rather than Illumina reads, and Dindel was run with –numWindowsPerFile 1000000. Dindel was not used much on real data because after one day of running, it was still not finished.

2.3.2 Simulated Data

The simulated data was generated from 10 megabases of chromosome one from a publicly available human genome available from the National Center for Biotechnology Information (GRCh38). Artificial mutations were introduced using inGAP, a software tool for the manipulation of genetic data [120]. SNVs were inserted at a rate of 0.1% per base, and indels were inserted at a rate of 0.02% per base. These values were chosen since they were realistic for

the human genome [101]. Indel lengths were uniformly distributed from one to thirty bases. Ten replicates of simulated reads were produced to generate average and error statistics for the tests. Reads of uniform 50 base pair length were generated from the mutation sequences in the fastq file format using inGAP. Reads of length 50 were chosen because indel identification is more complex for shorter single-end reads since they "lack insert length variance" [104], so short single-end reads represented a good test of indel pipeline sensitivity. To study the effects of varying coverage on the accuracy of the pipelines, reads were generated for 10x, 50x, and 100x coverage for each of the mutation sequences. At a fixed coverage of 10x, reads of length 100bp and 150bp were sampled from the same ten replicates of mutated FASTA files to study read length effects on mapping.

2.3.3 Real Data

Applied Biosystems (Sanger) paired-end traces from the set 'center_name = "SC" and CENTER_PROJECT = "CHR_22_7340"' were identified and downloaded from the NCBI trace archive (https://www.ncbi.nlm.nih.gov/Traces/trace.cgi). The traces were used in a Devine lab study that searched for indels in human chromosome 22 [95]. The paper identified 6487 indels for the Chr_22_7340 traces in a supplemental spreadsheet.

We cleansed the traces of contamination using NCBI VecScreen, where traces with vector contamination in the middle were discarded and traces with vector contamination on the ends had the contamination trimmed off. After this, there were 217,924 traces with sizes as large as 2000bp. Since short read mappers perform poorly with very long sequences, three samples of 10, 20, and 30 million 100bp portions of the traces were sampled with replacement to simulate short reads. Short reads of 100bp were chosen since this is a realistic size to simulate Illumina short reads. The three sets of simulated single-end sequences were run

through the pipelines with timing tracked with the Linux `date` command.

A second real data set from the 1000 genomes project (www.1000genomes.org) [24] was evaluated with Sequence Read Archive (SRA) number SRR766008. This project called variants by aligning short reads to a reference genome and then computationally calling indels presumably with Dindel. The phase 3 variant set (VCF file) of 09/17/2014 for human chromosome 22 was downloaded and compared to the indels called by our pipelines for reads from a randomly chosen individual, NA18647. The fastq file used for this individual was SRR766008_2.filt.fastq, which comprised 28,936,312 101bp reads generated with Illumina short read technology. The chromosome 22 variants for individual NA 18647 were extracted from the phase 3 variant set with tabix [76] and vcf-subset from vcftools [27].

2.3.4 Indel Detection

A confusion matrix for each pipeline on each data set was created that recorded true positives, false positives and false negatives. Indels were recorded as true positives if the predicted indel's position was plus or minus 5 nucleotides of the actual indel's position and the predicted length was within 5 percent of the actual length (with all lengths set to be one if 5 percent of the actual length was less than 1). The indel had to be correctly classified as an insertion or a deletion to be marked a true positive; otherwise, it was classified as a false positive. The sequence identity of predicted and actual indels was not checked since differences in sequence identity were rare. A false positive was a predicted indel that did not meet the preceding criteria, and a false negative was an actual indel that was not identified by the indel classifier. Precision, recall, and F1-score were calculated for all pipelines to assess accuracy.

Precision is the ratio of true positives to the sum of true positives and false positives. Precision reveals the percentage of predicted indels that are genuine. Recall is the ratio of true

positives to the sum of true positives and false negatives. Recall reveals the percentage of genuine indels that are predicted. A good indel classifier will have both high precision and high recall. F1-score is the harmonic mean of precision and recall. F1-score is 2 *(precision * recall) / (precision + recall). A perfect classifier has precision and recall (and F1-score) of 1. Python 2.6 and Bash scripts were created to do the statistical analysis and workflow.

2.4 Results and Discussion

2.4.1 Analysis of F1-Score and Coverage on Simulated Data

In our results on simulated data with indels of size 1-30 bases, there were clearly pipelines that performed better than other pipelines as measured by the F1-score (Figure 2.1). For each pipeline, Figure 2.1 shows average F1-score and the minimum and maximum F1-score of the 10 replicates. Most indels called had fewer than 10 bases.

Figure 2.1 shows that pipelines with 10X coverage have the best F1-scores, and that 50X and 100X coverage perform less well. This was explained by a tradeoff between precision and recall caused by both increasing false positives and increasing true positives. As coverage increased, recall increased since indel callers return more predicted indels and thus more genuine indels. However, there were more false positives as coverage increased, so precision went down as coverage increased. The general downward trend of the F1-score was because precision decreased more than recall increased with increasing coverage. This suggests that there is some coverage amount that maximizes F1-score for the data, and increasing coverage is not always desirable. This result was consistent with other work that showed statistically significant precision and recall trends with increasing coverage [104].

The three top performing pipelines were BFAST-Dindel (avg F1-score 0.66), SHRIMP-SNVer

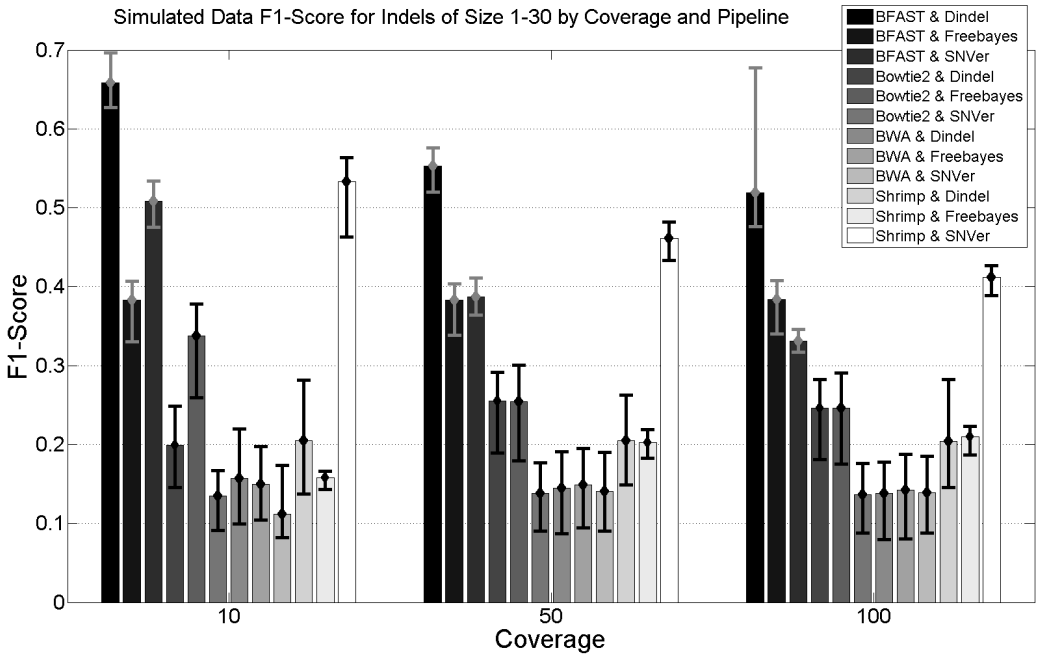


Figure 2.1: The F1-score of indel calling pipelines on simulated data with reads containing indels of 1-30 bases. The results are divided into sets of 10X, 50X, and 100X coverage. The F1-score is shown with average, low and high scores.

(0.53), and BFAST-SNVer (0.51) in the 10X coverage for 1-30 indels. The BFAST-Dindel pipeline had the best average F1-score for all coverage amounts (Figure 2.1). SHRIMP pipelines were interesting, since the F1-score varied considerably. The Shrimp-SNVer pipeline was among the top performing, but Shrimp-Freebayes and Shrimp-Dindel performed poorly. By default, Shrimp mapped some reads to multiple positions, while other read mappers only report uniquely aligned reads by default. This could be the cause of Shrimp's variable performance.

Read mappers use a seed and extend strategy, and BFAST's seeding strategy used a sliding window at every base. Bowtie2 used multiple 20bp seeds with an offset determined by the read length. Perhaps BFAST's seed strategy allowed it to be more accurate. All the mappers' extension phases are similar, since they use local or global alignment algorithms [28, 55, 72, 77].

The F1-score difference for 10X coverage between the best pipeline (BFAST-Dindel) and the worst pipeline (BWA-SNVer) was about 0.546. BWA generally performed poorly and this could be because it only supported gaps shorter than 10 bases in its alignments [77]. Even though BWA and Bowtie2 used a similar seeding strategy with the Burrows-Wheeler transform, Bowtie2 pipelines usually had better F1-scores. Bowtie2's split seed approach handles some variation in the seed [72].

Interestingly, there is not one clear indel caller that did the best overall. Bowtie2-SNVer was among the lowest performing, while SHRIMP-SNVer was among the best performing. Dindel did well with BFAST but not very well with SHRIMP.

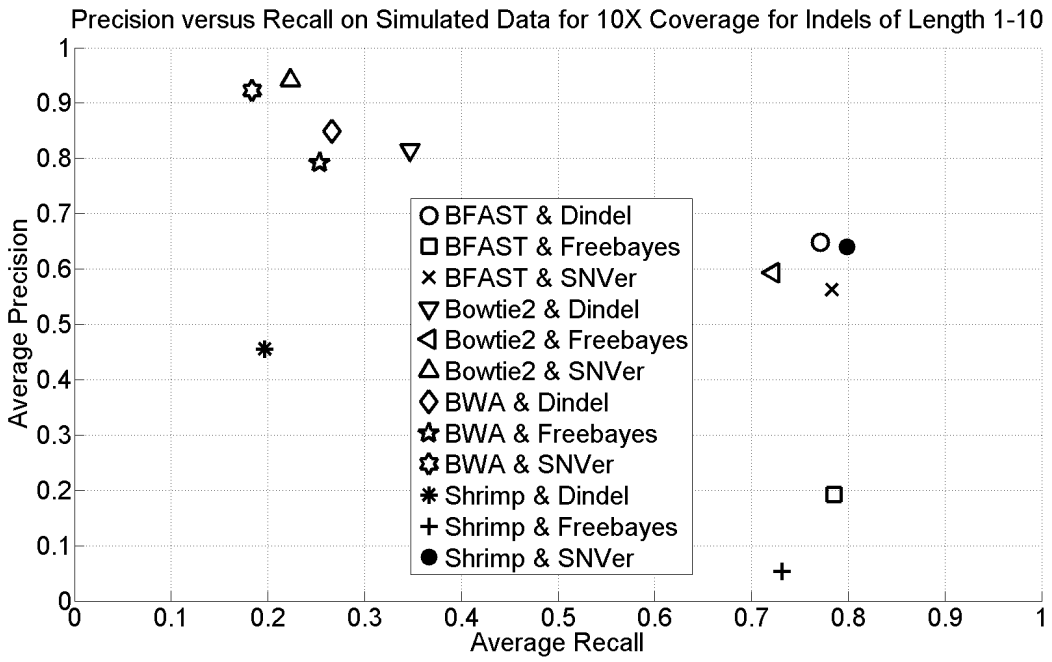


Figure 2.2: Average precision and average recall for 10 simulated data replicates for the indel calling pipelines. Precision and recall was calculated for indels with only 1-10 bases at 10X coverage. The reads contained indels as large as 30 bases.

2.4.2 Precision and Recall on Simulated Data With Smaller and Longer Indels

Figures 2.2 and 2.3 show a comparison of the effects of longer indels on precision and recall at 10X coverage. Pipelines performed worse for data with indels of 1-30 bases (Figure 3) than for indels with 1-10 bases (Figure 2.2). Figure 2.3's precision-recall tuples are generally shifted left when compared to Figure 2.2. The Bowtie2-Freebayes pipeline did noticeably better with 1-10 indel lengths.

The precision-recall plots show which pipelines are conservative, which are generous, and which are balanced. The pipelines involving BWA and Bowtie2 were the most conservative with high precision but low recall. Pipelines involving Freebayes were the most generous with low precision but higher recall. BFAST-Dindel and Shrimp-SNVer had the most bal-

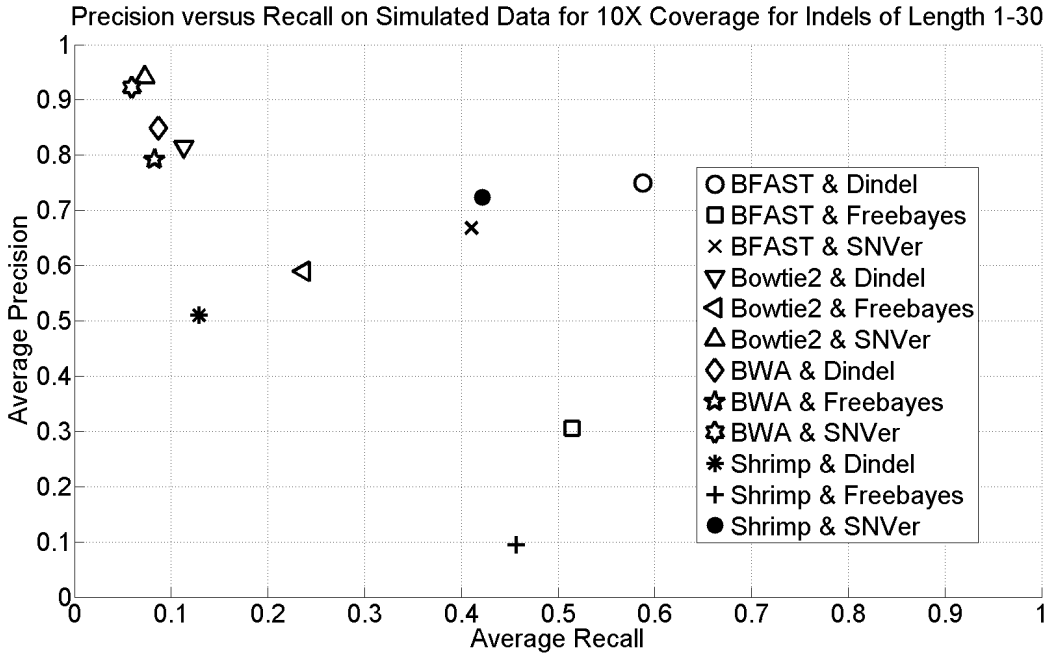


Figure 2.3: Average precision and average recall for 10 simulated data replicates for the indel calling pipelines. Precision and recall was calculated for all indels. Indels had 1-30 bases at 10X coverage.

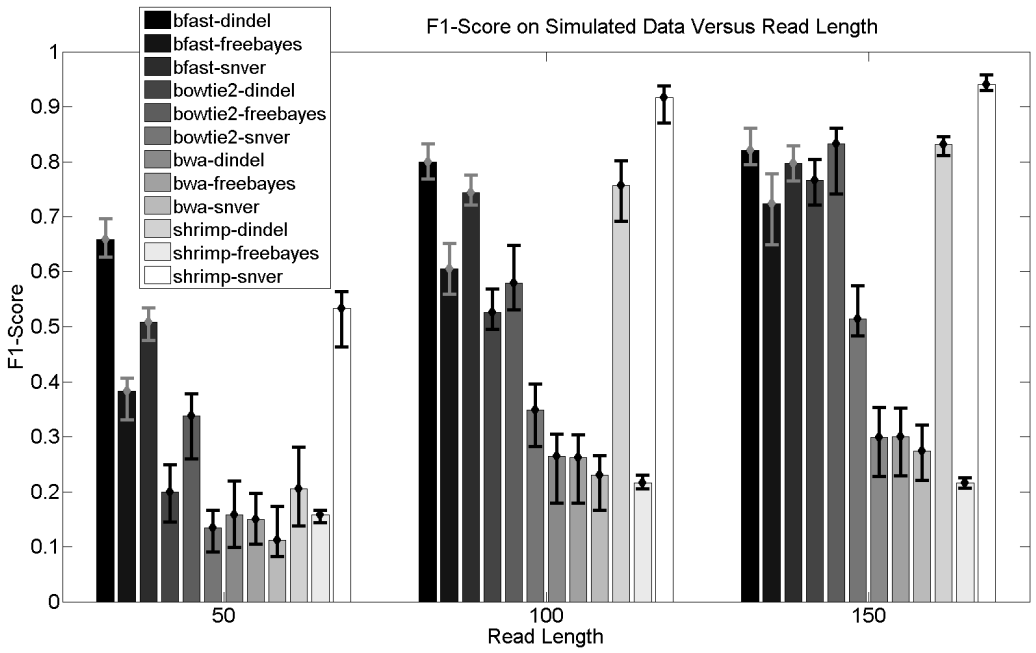


Figure 2.4: The F1-Score for the pipelines on simulated data versus reads of length 50, 100, 150 base pairs.

anced precision and recall results with (0.648,0.771) and (0.640,0.799) respectively for indels of length 1-10. The BFAST-Dindel pipeline performed better than Shrimp-SNVer for indels of length 1-30. Bowtie2 pipelines generally appeared mediocre in our tests. BFAST pipelines had generally good performance with different indel callers, while SHRIMP pipeline performance varied considerably with indel calling software. BWA pipelines performed poorly.

2.4.3 Effect of Read Length on Simulated Data

The Figure 2.4 shows the F1-score of the pipelines for read lengths 50bp, 100bp, and 150bp. There is a generally positive correlation with F1-score and read length for all pipelines. The results are similar to the coverage results with pipelines involving BFAST and Dindel performing well. Interestingly, the Shrimp-Dindel pipeline improved dramatically at 100bp, and Bowtie2 pipelines improved as well. BWA pipelines and Shrimp-Freebayes did poorly.

2.4.4 Accuracy of Sanger Real Data

The only accurately called indels on the real data were smaller than 5bp. Figure 2.5 shows the F1-scores of the real data pipelines. SNVer pipelines had the best F1-score. Similar to the simulated data, pipelines with Freebayes were too generous with high recall but low precision. Average precision and recall for Freebayes was 0.000440955 and 0.010906428, but, with SNVer, it was 0.00117591 and 0.001079081. Thus, SNVer was more conservative in indel calling. The Shrimp-SNVer and Bowtie2-SNVer pipelines did the best while BWA-Freebayes was the worst. The choice of read mapper made little difference, and this could be because only small indels were called. True positives were few relative to indels called (Table 2.1). For the BWA mappings, Dindel completed in 6.6 hours with similar precision (0.00062) and recall (0.0026) to the BFAST-SNVer pipeline (0.00079, 0.0012).

2.4.5 Run-Time Performance on Sanger Data

Table 2.1 summarizes run-time performance on the 10 million read set for the pipelines for the real human chr22 data. Bowtie2 and BWA, had the fastest runtimes at 27 and 23 minutes respectively. BFAST and Shrimp were the slowest mappers, and both used a sliding window hashing seed strategy. BFAST was about 10 minutes slower, but Shrimp was 6.2 times slower than BWA, making Shrimp pipelines the slowest. The Shrimp read mapping percent is more than 100 percent since it mapped some reads to multiple positions by default.

The indel callers did not have multithreading, so they were slow. Freebayes was always faster than SNVer, and SNVer took 145 minutes with Shrimp's input making the Shrimp-SNVer pipeline the slowest. Dindel took over a day (except with BWA input), making it less tolerable for big indel calling projects; however, Dindel has the ability to split its work into multiple files for manual multiprocessing.

Mapper	Caller	Minutes	Total Minutes	% Reads Mapped	Total Indels Called	True Indels
BFAST		38		0.5437498		
	Samtools	5				
	SNVer	18	61		20486	8
	Freebayes	37	80		358574	139
Bowtie2		8		0.4850195		
	Samtools	4				
	SNVer	19	31		9812	6
	Freebayes	16	28		114679	54
BWA		10		0.412648		
	Samtools	4				
	SNVer	18	32		6388	5
	Freebayes	9	23		30789	14
Shrimp		143		1.7618786		
	Samtools	4				
	SNVer	145	292		9145	9
	Freebayes	59	206		168684	76

Table 2.1: Mapper, caller, pipeline run-time, percent mapped, and indels called on 10 million real human 100bp reads.

2.4.6 Accuracy of 1000 Genomes Real Data

Similar to the Sanger data, F1-scores for the 1000 genomes data were very small. Figure 2.6 summarizes the results. Interestingly, SNVer consistently performed worse than Freebayes, which is the opposite of the Sanger data. The filtered indel variant file was used for SNVer to compute the F1-score to be consistent with the simulations and the Sanger data, but the unfiltered indel file for 1000 Genomes data had an F1-score on par with Freebayes. This was the opposite for the simulations and the Sanger data. This is the only data where Freebayes performed the best. Dindel was prohibitively slow.

2.5 Conclusions

To my knowledge, this work is the first to look at the accuracy of the combination of mapping software and indel calling software with larger (>10 nucleotides) indels. F1-score, a measure

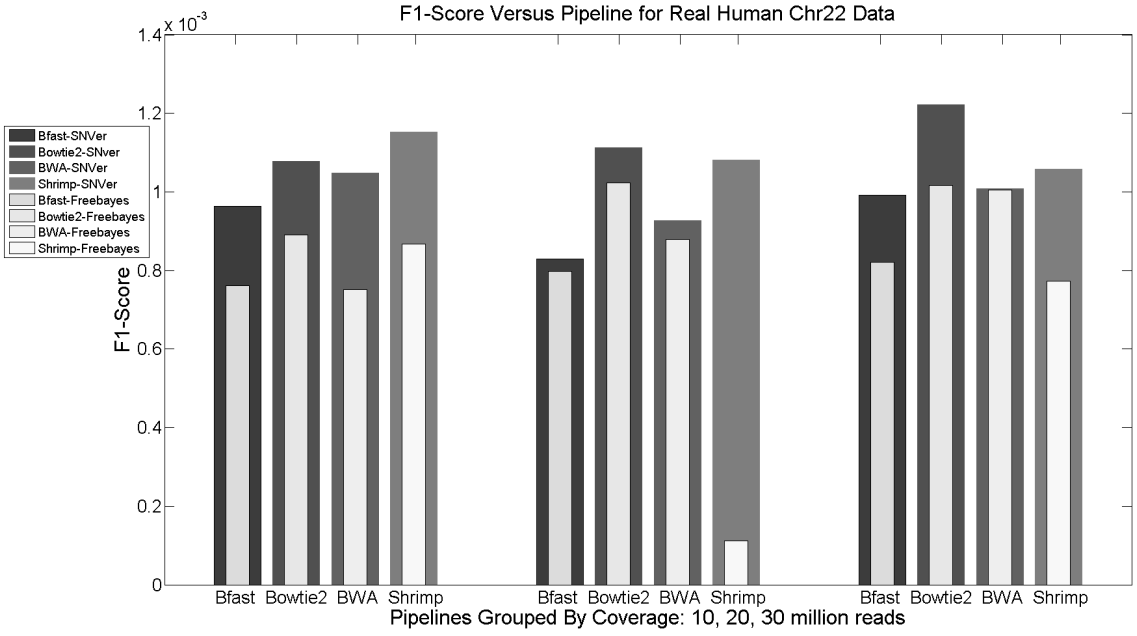


Figure 2.5: F1-Score for the indel calling pipelines on 10, 20, and 30 million reads on real human chromosome 22 Sanger traces.

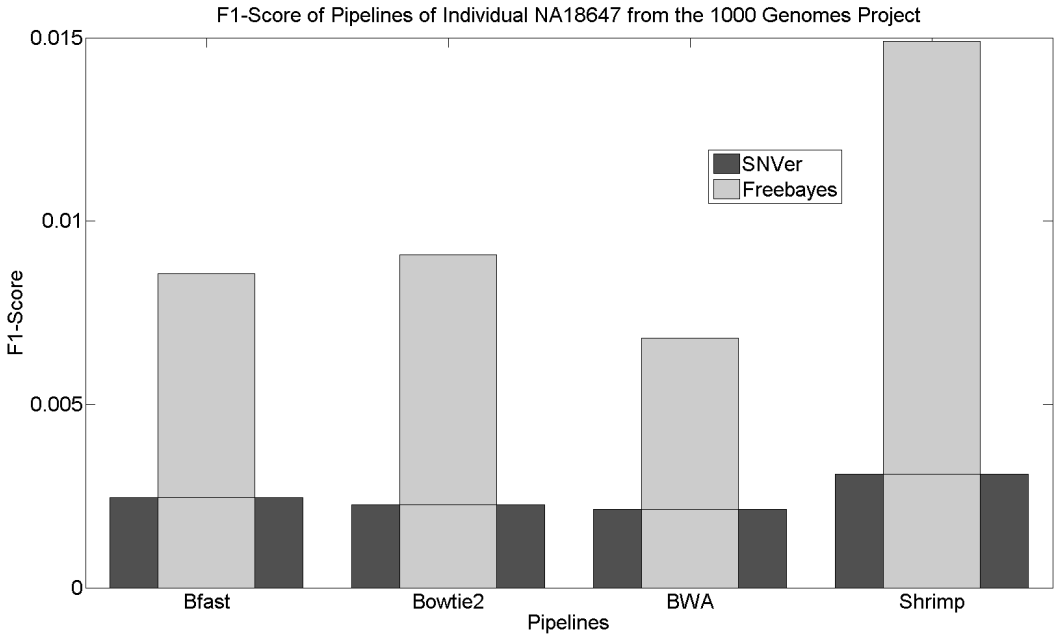


Figure 2.6: F1-Score for the indel calling pipelines for individual NA 18647 of the 1000 Genomes Project. Variants from the Phase 3 data were used in comparison.

of accuracy, fell with increased coverage, belying expectations. However, F1-score increased with read length on simulations. Indel calling accuracy depended on the combination of mapping software and indel calling software. Some of the top performing pipelines were BFAST-Dindel, SHRIMP-SNVer, and BFAST-SNVer on simulated data. The best pipeline had an F1-score 0.6 higher than the worst pipeline on simulated reads. All pipelines improved with longer read length, but Bowtie2 pipelines and the Shrimp-Dindel pipeline improved dramatically with longer read lengths. On real Sanger data, SNVer pipelines were more accurate than FreeBayes pipelines in all cases. The opposite was true for the 1000 Genomes data. SNVer and Shrimp can have slow runtimes, but Dindel was by far the slowest variant caller. Future work could include exploring the parameter space of the tools to observe the effects of argument selection on sensitivity.

Chapter 3

Insights into Read Mapping with the Hairpin Protocol

3.1 Background

In this work, we use a sequencing strategy called genome-wide hairpin sequencing of bisulfite-treated short reads to investigate factors that may adversely influence mapping efficiency of bisulfite reads [169]. Unlike previous bisulfite sequencing methods that destroy knowledge of the original (not bisulfite-treated) sequence, the hairpin technology (see Figure 3.1) allows for the recovery of the original sequences by putting a connector between the Watson (top) and Crick (bottom) strands and then using PCR and paired-end technology to sequence short reads [71]. The resulting sequences give paired strands that can be mapped to recover the original untreated read and to detect sequencing errors.

This study focuses on the read mapper Bismark, which is popular, easy to use, and reasonably accurate [147]. It converts all C's to T's in the read and the reference genome, and then calls

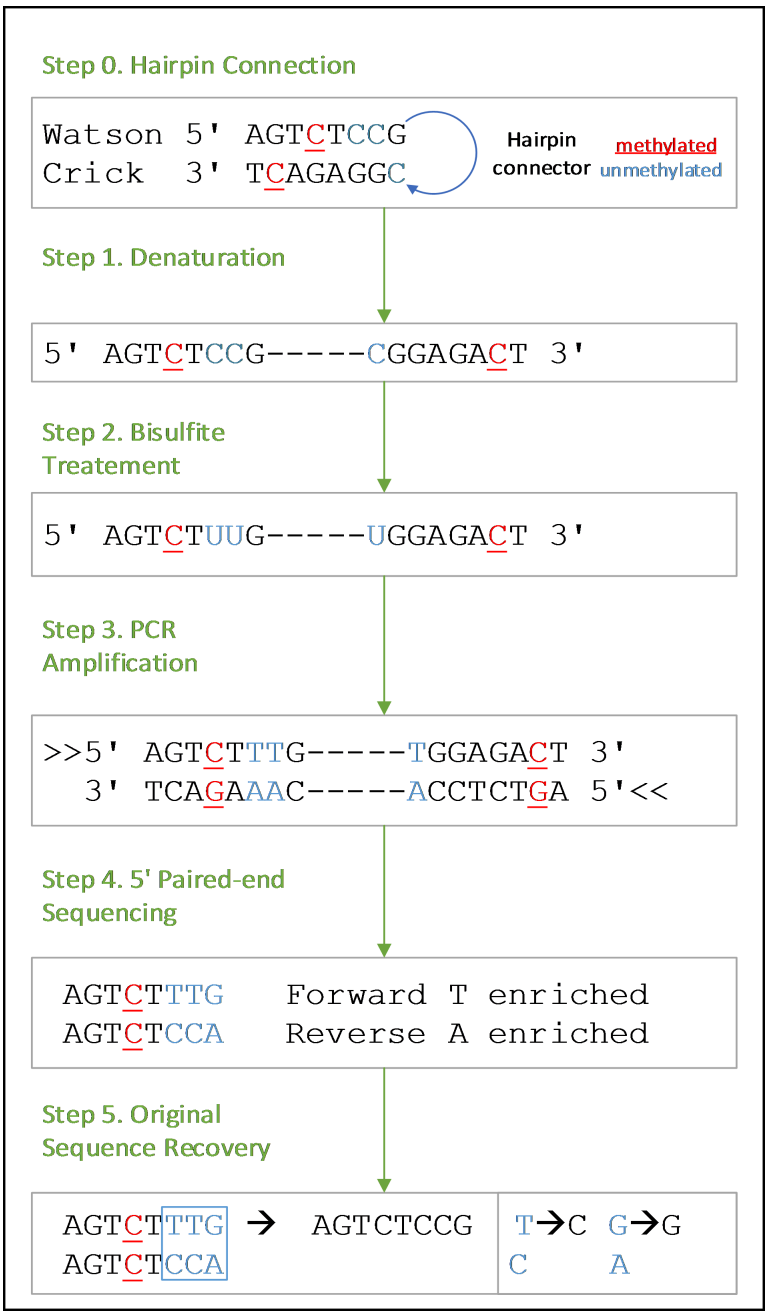


Figure 3.1: Example of bisulfite-treated hairpin PCR sequencing with recovery of the original sequence.

Bowtie2 as a sub-process to do the mapping. Bowtie2 uses an FM-Index, a compressed string index, of the reference genome to find read positions on the reference genome. Bowtie2's scoring function weights mismatches, indel starts, and indel extensions. It can report the score for the best mapping as well as the next best mapping [70, 72]. Because the hairpin method enables recovery of the original sequences from bisulfite converted reads, the effects of bisulfite treatment on a mapping program can be studied by mapping bisulfite converted reads with Bismark and recovered (original) reads with Bowtie2. This isolates bisulfite treatment as the only variable affecting mapping quality. We also used the bisulfite mapper BSMap, one of the first bisulfite mapper programs, for comparison. It creates a hash table of the reference genome of all possible C-to-T conversions in a sliding window of 16 base pairs in the reference genome [163].

3.2 Methods

Genome-wide mouse hairpin bisulfite-sequencing data creation. Genome-wide hairpin bisuflite-sequencing data for mouse embryonic stem (ES) cells (E14TG2a) was given to me by David Xie, and this data was generated by people in his lab (GEO accession number GSE48229) [169]. To induce differentiation, the mouse ES cells were cultured in ES cell culturing media for six days without LIF (Leukemia inhibitory factor). Genomic DNA isolated from the undifferentiated (E14-d0) and differentiating (E14-d6) states of mES cells were used for library construction. Briefly, after DNA sonication, end repair, and dA tailing, the DNA fragments were further ligated to Biotin-modified hairpin adapter and Illumina TruSeq adapters. Adapter-ligated DNA was digested with MseI and MluCI (NEB) to enrich regions with high CpG density, and then pulled down using Dynabeads® MyOne™ Streptavidin C1 beads (Invitrogen). After bisulfite conversion and PCR, size selection of

400-600 bp fragments was conducted to yield longer sequences that are more amenable for unambiguous mapping to the reference sequence.

Unconverted sequence recovery. Figure 3.1 shows how the bisulfite-treated hairpin PCR sequencing technology works and how the original non-bisulfite sequence can be recovered. In step 0, the hairpin connector is attached to opposing Watson and Crick strands. The opposing strands are denatured in step 1 and then treated with bisulfite in step 2. Bisulfite treatment converts un-methylated cytosine to uracil. Step 3 involves PCR amplification, which copies the forward and reverse sequences and converts uracil into thymine. In step 4, paired-end sequencing technology is used to sequence from the 5' ends. This gives a sequence pair. Notice that one strand is enriched for T's, since the PCR and bisulfite treatment converts un-methylated C's to T's. The other strand is enriched for A's, since the Crick strand's un-methylated C's become T's, which are complemented with A's. These A's are then paired with G's in the opposing strand. Finally, step 5 recovers the original sequence. When the T-enriched strand has a T and the A-enriched strand has a C, this maps to a C, and when the T-enriched strand has a G and the A-enriched strand has an A, this maps to a G. All other mismatches are due to errors in PCR or sequencing, since the two strands should match up perfectly as they come from opposing sequences in the DNA.

Software installation and use. All software development and analysis was performed on the bioinformatics clusters hosted by the Virginia Bioinformatics Institute and the Virginia Tech CS department. Two of the machines consist of two quad core processors (Intel(R) Xeon(R) CPU E5-2407 @ 2.20GHz) each with 128 gigabytes of memory. Two other machines consisted of three eight core processors (Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz) each, where one machine has 128 GB of memory and the other has 192 GB of memory. Bowtie2 (version 2.1.0) was compiled with the Gnu Compiler Collection. Bismark (version 0.7.12) was run with Perl (version 5.10.1) with the non-directional flag and default settings. BLASTN

(version 2.2.28+) was run with default settings on a C-to-T converted reference genome to map unmapped Bismark reads, where all of the reads had C's converted to T's. Bismark reports unique mappings, which are mappings that have a unique best score, ambiguous mappings, which are reads with locations that have multiple best scores, and unmapped reads, which had scores that were too low or reads where a location could not be assigned.

File processing and statistical calculation. Custom Python 2.7 [89] scripts using BioPython [22] were created to process files and calculate statistics such as sequence entropy and sequence length. File manipulation included randomly sampling from FASTA files. The scripts are available at

https://github.com/JacobPorter/PorterTools/tree/master/BioTools/BioScripts.

Sequence entropy was calculated for each sequence in each of the mapping categories (unique, ambiguous, and unmapped). Entropy for a sequence, s, is calculated by taking the negative sum of the frequency of each base, f_b, times the logarithm of the frequency of each base. More formally,

$$Ent(s) = -\sum_{b \in \{A,C,T,G\}} f_b \log_2(f_b).$$

Entropy gives a measure of sequence randomness [131]. This measure of sequence complexity was chosen, since it is simple and fast to compute. A sequence of all T's will have an entropy of 0 since $f_T = 1$ and $\log(1) = 0$, but a sequence with 25 percent of each base will have an entropy of 2. We hypothesized that entropy would affect mapping quality. Highly random-seeming sequences should uniquely map, while non-random sequences, for example, a sequence with mostly T's, will map ambiguously or be discarded, since the string of T's could map to many portions of the genome that have C's at indeterminate locations in the string of T's. Bisulfite treatment tends to reduces entropy, since most C's will be converted to T's, since cytosine methylation is usually rare. Any read with any N's in it was excluded

from the entropy analyses. Other analyses included reads with N's in them.

For comparing the unique mapping efficiency for converted and unconverted reads, 50 bootstrap replicates with replacement were created from all of the mouse data (both differentiating and undifferentiated). Each replicate had 6.46 million reads. Each replicate had the T-enriched pair and the A-enriched pair so that the untreated original sequence could be recovered. Reads that aligned exactly (that is, there was no sequencing error) were extracted and mapped with Bismark on default settings for both the T-enriched and the A-enriched reads. About 68% (4.4 million) of the reads in each replicate aligned exactly. The original read was recovered and mapped with Bowtie2 using the same settings as Bismark. Bismark reports a uniquely mapped percentage, and the recovered sequence mapping categories from Bowtie2 was used to calculate a uniquely mapped percent in the same way as Bismark. Read mappings were done for all 50 bootstrap replicates so that a p-value could be computed.

For a single bootstrap replicate for the day 0 data, reads were extracted that had 0, 1, 2, and 3 mismatches somewhere in the read. Mapping using Bismark was done for all of them. Bismark mappings for reads with mismatches only in the first 25 bases were compared with reads with mismatches only in the latter right most part. This comparison was done for 1, 2, and 3 mismatches.

With Illumina pair-end sequencing technology, a total of 8 lanes of data was produced. Lane 8 data was used throughout except in the two cases mentioned above that used bootstrap replicates drawn from all of the data. Both T-enriched and A-enriched strands were used for analysis, and the day 0 and day 6 data was used. The day 0 lane 8 data has 32,434,798 reads and the day 6 lane 8 data has 58,034,817 reads.

To understand what the best possible bisulfite mapping efficiency was, a simulation was performed where simulated bisulfite reads were sampled from the mouse reference genome

at random and read sequencing error was introduced. This simulation represented the best possible mapping scenario, since the reads did not include natural variation. The simulation sampled sequences of length 100, 75, and 50 bp in the same proportions that were in the lane 8 data and compared mapping efficiency to the real hairpin data with sequences of only 100, 75, and 50 bases. Simulations with sequencing error of 1% and 10% were created. (Sequencing error set to less than 1% had very little difference in the mapping efficiencies of 1% error.) The program Sherman from Babraham Bioinformatics was used to do the simulation with methylation rates set to CpG = 80% and CPH = 2% [7]. These rates were chosen since they were consistent with the literature [169]. Both BSMap and Bismark were used to do the mapping.

To investigate the benefit of using different algorithms, BLAST was run on default settings on 64 percent of the bisulfite-treated reads unmapped by Bismark for the T-enriched day 0 lane 8 data. The reads and the reference genome had all C's converted to T's. Only 64 percent of the reads were used since BLAST took a long time to complete.

3.3 Results and Discussion

Bisulfite-treated reads have low mapping efficiency. Bismark uniquely mapped 47.5-52 percent of the bisulfite converted reads in the mouse data, ambiguously mapped 27-34 percent, leaving 16-20 percent unmapped. Therefore, approximately half of the reads do not give a strong signal for their position.

Sequence entropy relates to mapping efficiency. We examined the average sequence entropy by mapping category for both day 0 and day 6 data when mapped with Bismark or BSMap (Figure 3.2). Both mappers show a trend of increasing average entropy when going from unmapped to ambiguous to unique mappings.

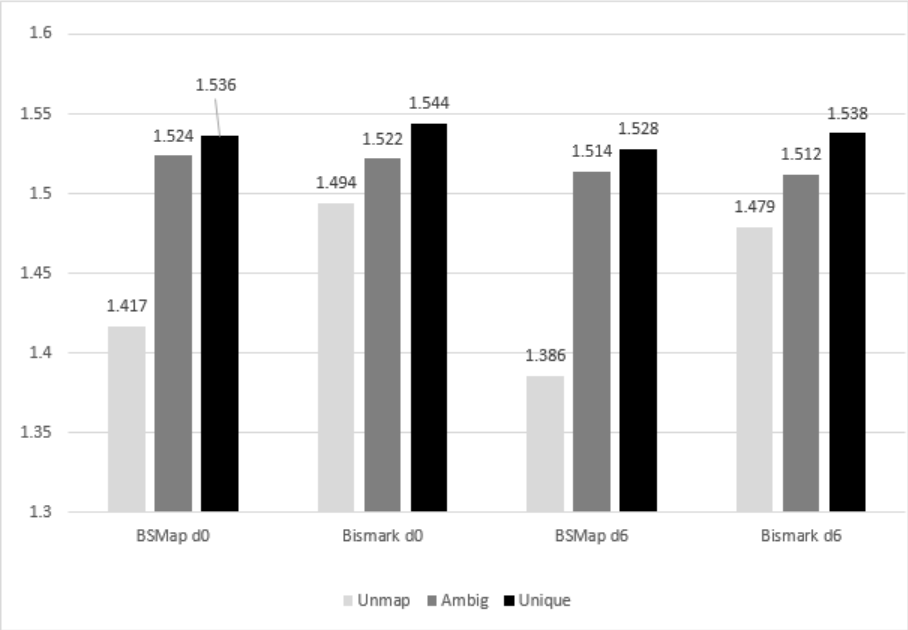


Figure 3.2: Bismark and BSMap average sequence entropy by mapping category for both day 0 and day 6 data.

Figure 3.3 shows a histogram of sequence entropy by mapping category for the hairpin data on day 0 mapped with Bismark. There is a spiking phenomenon around 1.5 that may correspond with reduced entropy due to bisulfite treatment. During bisulfite conversion, most of the C's were converted to T's and thus the frequency of the C's is greatly reduced. In this distribution, the maximum differences between the mapping categories were determined as the following: unmap-ambig: 0.05, unmap-unique: 0.12, unique-ambig: 0.17. The Kolmogorov-Smirnoff two-sample tests were performed and the differences between these distributions are with p-values less than 0.01. Because uniquely mapped reads tend to have higher entropy, entropy positively correlates with improved mapping efficiency. This suggests that reduced entropy from bisulfite treatment contributes to less mapping effectiveness.

Figure 3.3 shows that very few sequences have very high entropy, since all read mapping categories go to zero for high entropy buckets. This is because bases are not uniformly distributed in real genomes [9]. A completely equal frequency of bases in a read maximizes

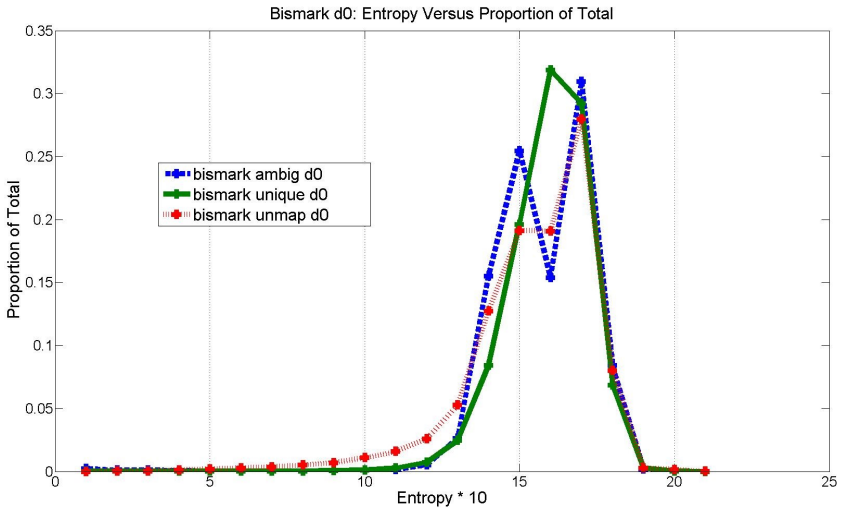


Figure 3.3: Bismark entropy distribution on day 0 data by mapping category: ambiguous, unique, unmapped. The x-axis is entropy times 10, and each integer point x is a bucket of sequences where x represents sequences with entropy values between $\frac{x}{10}$ and $\frac{x-1}{10}$. The y-value is the proportion of the total number of sequences in the mapping category.

entropy. The largest bucket at $x = 21$ is completely empty and has a y value of 0 since there are no sequences with entropy higher than 2.0, which are in the bucket at $x = 20$. All sequences that had any N's in them were excluded from the analysis.

One reason that lower entropy reads can be ambiguously mapped or filtered out is that homopolymer runs, such as 'TT' or 'AA', will naturally match to more locations in a reference string. To see this consider the strings 'ATAT' and 'TTTT,' which both repeat a two character string twice. The string 'AT' matches to two locations in 'ATAT,' but the string 'TT' matches to three locations in 'TTTT.' Homopolymer runs can reduce entropy by increasing the frequency of a base, so low entropy can be predictive of this phenomenon.

Recovering the original sequence improves mapping efficiency. We hypothesized that the hairpin sequencing technology can improve the mapping efficiency by recovering the original sequences, which have higher entropy. For the undifferentiated data, recovering the untreated sequence improved unique mapping efficiency by an average of 9.15% (p-value

$= 0$, variance $\approx 10^{-8}$) compared to the average unique mapping efficiency of the T-enriched and the A-enriched reads. For the differentiating data, the unique mapping efficiency was improved by 10.59% p-value $= 0$, variance $\approx 10^{-8}$). This shows the benefit of recovering the information lost from bisulfite treatment and is consistent with the notion that more entropic reads map better.

Different hairpin sequences may map differently. We next checked whether mapping efficiency can be improved without sequence recovery but with mapping information from additional sequence read, since one sequence from the hairpin sequences may uniquely map while the other does not. Bismark was run on the lane 8 data for both day 0 and day 6. Both the T-enriched and the A-enriched sequences were compared to see if one sequence uniquely mapped while the other did not. If one sequence uniquely maps, then the other sequence can be considered to uniquely map as well. This improves the overall mapping efficiency. For the day 0 data, about 19 percent of the T-enriched sequences that uniquely mapped did not uniquely map in the A-enriched sequences, and vice versa. For the day 6 data, about 22 percent of the T-enriched sequences that uniquely mapped did not uniquely map in the A-enriched sequences and vice versa. This suggests that the hairpin sequencing strategy can improve mapping efficiencies by uniquely mapping one sequence to find the position of the other sequence.

Sequencing error adversely affects mapping efficiency. The number of mismatches in reads could be determined by mapping two paired-end sequences together. We observed a downward trend in mapping efficiency with increasing mismatch count (Figure 3.4). Figure 3.4 shows the results of mapping reads with mismatches in the first 25 bases compared with mapping reads with mismatches only in the other bases. The unique mapping efficiency is consistently lower when mismatches are in the first 25 bases, and it is worsened with increased mismatch count. Thus, mismatches correlate with low mapping efficiency, especially in the

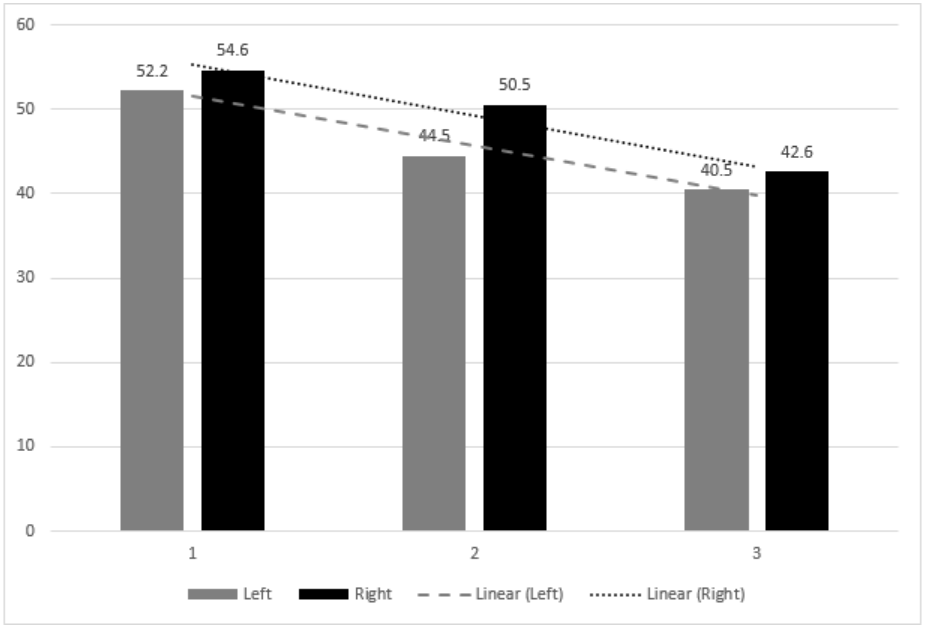


Figure 3.4: Number of mismatches on the left or the right side of the read versus alignment efficiency with day 0 data with Bismark unique alignment efficiency. The left side is the first 25 (5' end) bases of the read. The right side is the other bases.

5' end. The hairpin sequencing technology can be used to identify mismatch areas of a read for removal so that unique mapping efficiency can be increased.

Bismark and BSMap mappings are highly overlapping. The T-enriched sequences for the day 0 data were mapped with Bismark and BSMap. Combining the uniquely mapped sequences of both mappers (an OR operation) resulted in only about a 2 percent increase over BSMap alone. Other work has shown greater differences between these programs for other data [147]. Therefore using multiple mappers may improve mapping efficiencies. Using multiple mappers increases the confidence that a read is mapped correctly when the mappers map to the same location.

Longer sequences tend to map uniquely. Reads in the mouse hairpin data ranged from 40 to 101 bases. Day 0 lane 8 data had 63.4 percent reads with lengths either 100 or 101, and day 6 lane 8 data had 69.34 percent reads with 100-101 bases. We suspected that longer

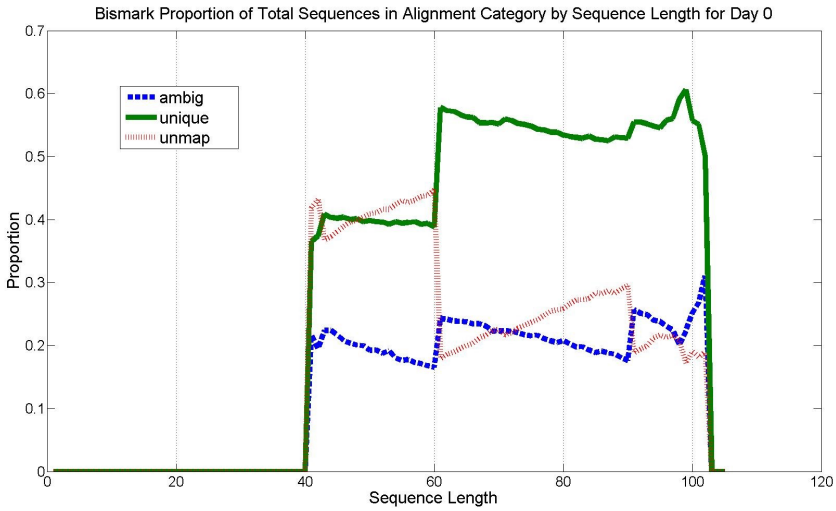


Figure 3.5: Bismark mapping categories by sequence length on day 0 data. This shows mapping efficiency by length for the day 0 lane 8 data for each mapping category: unique, ambiguous, and unmapped.

nucleotide sequences would have higher unique mapping efficiency since the probability that a longer sequence is repeated in the genome is low. For example, the sequence consisting of only A is often found, but the sequence ATCGGTGCCAT will be found less often.

The programs BSMap and Bismark were used to do mapping, and then the percentage of reads of a given length that belong to each mapping category was calculated as can be seen in Figure 3.5 for Bismark. For Bismark, there is a generally upward trend with longer sequences more often mapping uniquely and longer sequences less often unmapped. There are noticeable spikes at lengths 60 and 90. However, BSMap's mapping efficiency remains about flat (no spiking) until longer reads are mapped where there is a slight increase in mapping efficiency. These results confirm the hypothesis that longer reads tend to map uniquely.

Sequencing error simulation suggests an upper bound on mapping performance. Figure 3.6 shows that with one percent sequencing error and with reads drawn randomly

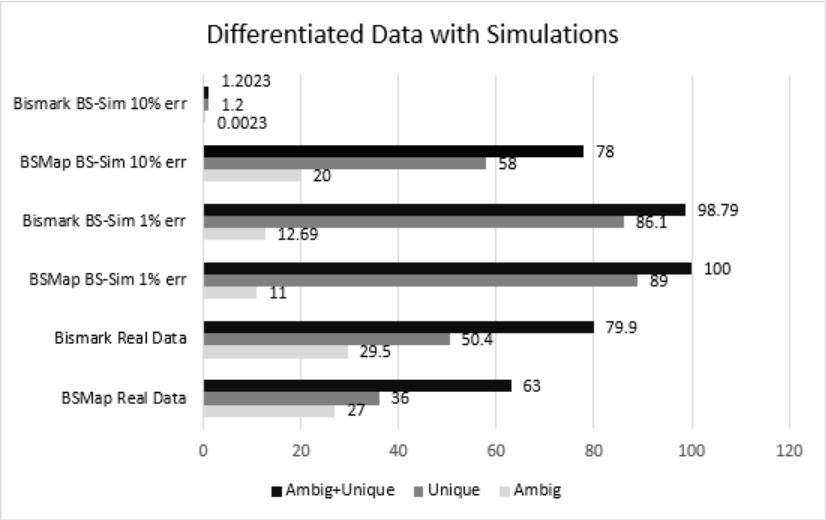


Figure 3.6: Mapping results for both simulated and real data on sequences of length 100, 75, and 50 distributed in the same way as the hairpin data set. Simulation on bisulfite-treated at 1% and 10% error rate. The mapping was done with BSMap and Bismark.

from the mouse reference genome, Bismark uniquely maps 86.1 percent of the reads and ambiguously maps 12.7 percent of the reads. The best possible mapper probably will not be able to do much better than this, since the simulation did not involve natural variation but only realistic sequencing error. This compares to 50.4 percent unique mapping efficiency by Bismark on bisulfite-treated data according to Figure 3.6. This means that read mappers could improve by over 30 percent. A high sequencing error of 10% causes an almost complete failure of Bismark, and BSMap worsens by 22%.

BLASTing unmapped reads produces hits. Last, we investigated the benefit of using different algorithms for mapping by using BLAST to map reads unmapped by Bismark. BLAST significantly mapped around 30 percent of these unmapped reads. BLAST is noticeably slower and its output is not in a convenient format for variant calling; nonetheless, this result shows that more computationally intense methods improve mapping efficiency.

3.4 Conclusions

In this study, we observed that low sequence entropy co-occurs with low unique mapping efficiency, and high entropy co-occurs with high unique mapping efficiency. This suggests that lost entropy from bisulfite treatment causes worse mapping efficiency. In addition, mismatches in the seed region for a mapper correlate with worsened performance. The hairpin sequencing strategy is useful in improving mapping performance of bisulfite-treated reads. With sequence information from two DNA strands, the original sequence can be recovered, and PCR or sequencing error mismatches can be identified. Hairpin data can be used to ameliorate lost entropy by recovering the original sequence and improving unique mapping efficiencies.

Additional coping strategies involve using multiple read mappers or more computationally intense software such as BLAST to improve mapping results, since they can be used to map reads that are unmapped by less computationally intense software such as Bismark.

Chapter 4

BisPin and BFAST-Gap: Mapping Bisulfite-Treated Reads

4.1 Background

Short DNA reads are treated with bisulfite to study epigenetic methylation, and these reads are mapped with software to a reference genome for epigenetic methylation discovery. Epigenetic methylation is a phenomenon where cytosine nucleic acids in DNA have a covalently bonded methyl group (CH_3) attached to the 5 carbon of the cytosine ring. Epigenetic methylation is inheritable, and it plays a role in disease and development [2, 169].

Bisulfite treatment converts unmethylated cytosines into thymines, while leaving methylcytosine unchanged. The DNA sequencing process may sequence the reverse complement so that adenine corresponds to unmethylated cytosine and guanine corresponds to methylcytosine. A read mapper compares these bisulfite converted reads to a normal reference genome to discover methylcytosine, since methylcytosine should align with a cytosine in the reference

genome, or in the case of the reverse complement, with a guanine. Read mapping of this sort is challenging, since the bisulfite treatment introduces differences in addition to natural variation and sequencing error between the reference genome and the reads. Bisulfite read mapping has been characterized by low mapping efficiency [147], which has been shown to be correlated with reduced sequence complexity [118]. Read mapping can be time consuming with millions of short reads.

Software packages that map bisulfite-treated DNA reads to a reference genome include Bismark [70], BWA-Meth [110], Walt [19], and others. All of these programs use three phases: (1) reference genome index creation, (2) seeding, and (3) extension with alignment. A string index is usually created with either the Burrows-Wheeler transform and FM-Index or a hash table [78]. Both Bismark, which calls Bowtie2 [72] for read mapping, and BWA-Meth, which calls BWA [78] for read mapping, use the former approach. Walt uses the latter hash table approach. Seeding is accomplished by taking short subsequences of the DNA read and matching them with the string index of the reference genome. This procedure gives candidate hits, which are possible locations where the read can be mapped. Finally, an extension step is performed where the entire read is aligned to the reference genome at the candidate locations. This is normally accomplished with the Smith-Waterman local alignment algorithm. The alignment algorithm scores each location so that the best location can be reported. Differences between the reference genome and the DNA read revealed by the alignment can be either the result of genetic variation, sequencing error, or unmethylated C's converted to T's (or G's to A's on the complementary strand), and therefore, methylation at single nucleotide resolution can be determined.

BatMeth, BRAT-nova, BSMAP, BS-Seeker2, and BSmooth are some other bisulfite read mappers. BatMeth filters out reads with low entropy but does not produce output in the standard SAM file format [81]. BRAT-nova is reported to be fast but could not be made to

work at the time of this writing [48]. BSMAP is outdated and cannot map reads longer than 144 base pairs, making it unsuitable for many modern sequencing projects [163]. BSMAP adds all possible combinations of C to T conversions to its index. BS-Seeker2 is similar to Bismark in that it uses Bowtie2 as a subprocess [43] for read mapping. BSmooth can call methylation marks [46]. These read mappers do nothing special to resolve ambiguously mapped reads, which are multiple high scoring alignments for a read. Their support for multiple protocols is limited to conventional sequencing protocols. They do nothing special for Ion Torrent reads.

To address these problems, this study presents BisPin, a bisulfite read mapper that deploys BFAST (BLAT-like Fast Accurate Search Tool) for read mapping. BFAST was chosen because its indexing strategy is very thorough and supports mapping with multiple indexes and spaced seeds. Spaced seeds are more sensitive [79], and indexes based on the Burrows-Wheeler transform implement a more limited spaced seed with less computational efficiency due to backtracking [73]. BFAST is feature-rich with informative output and multithreading. More importantly, BFAST has shown superior performance over tools such as Bowtie2 and BWA in the presence of indels over 10bp long [117].

This advantage becomes especially important when considering some sequencing platforms such as Ion Torrent where sequencing errors tend to introduce indels to reads [13]. Ion Torrent technology has two advantages over Illumina technology: longer reads and less expensive machines. Ion Torrent technology can produce 400bp reads [111] while Illumina MiSeq can produce 300bp reads [128]. Ion Torrent machines have a price around $80k, and Illumina machines have a price around $120k [87].

BisPin with BFAST has good performance on Ion Torrent data. Similar to Bismark and BWA-Meth, BisPin calls BFAST for read mapping, but BisPin adds special processing to accommodate short reads generated from the whole genome hairpin bisulfite construction

strategy [71, 169]. BisPin enables BFAST multithreading for alignment, multiprocessing for post processing, and read partitioning for deployment on compute clusters.

A forked version of BFAST, called BFAST-Gap, was developed with special processing for Ion Torrent reads. BFAST-Gap is backwards compatible with BFAST, and it has an implementation of the Smith-Waterman alignment algorithm that reduces the gap open and gap extension penalties depending on the length of the homopolymer run. This should improve performance on Ion Torrent reads where gaps occur more frequently in longer homopolymer runs. Tabsat is another read mapper that is specifically designed to map bisulfite-treated Ion Torrent reads [108]. Tabsat modifies the Bismark Perl code to call the Ion Torrent mapping program TMAP [53, 108].

The BisPin and BFAST-Gap software is available for download on GitHub at `https://github.com/JacobPorter/`.

4.2 Methods

4.2.1 BisPin Features

BisPin is a Python program that calls BFAST, a C++ program, to perform the three phases of read alignment and mapping. In this aspect, BisPin is similar to Bismark, which uses Bowtie2 to perform read alignment and mapping. BisPin supports directional (MethylC-Seq [82, 83]), nondirectional (BS-Seq [23]), post bisulfite adapter tagging (PBAT [96]), and hairpin [71, 169] read construction strategies. It maps both single-end and paired-end reads. For all reads, BisPin reports a methylation calling string in the style of Bismark. The output is given as a SAM file [80]. BisPin has an alignment result summary report, which includes mapping efficiency, methylation calling statistics, timing profile information, and

command line arguments. Table 4.1 summarizes BisPin's features compared to several programs including Bismark, Walt, and BWA-Meth. The BisPin software is freely available at https://github.com/JacobPorter/BisPin.

Feature	BisPin	Bismark	Walt	BWA-Meth
Paired-end and single-end support	✔	✔	✔	✔
Directional support	✔	✔	✔	✔
PBAT support	✔	✔	✔	✗
Nondirectional	✔	✔	✗	✗
Hairpin support	✔	✗	✗	✗
Methylation calling string	✔	✔	✗	✗
Ambiguously mapped resolution	✔	✗	✗	✗
Parallelization	✔	✔	✔	✔
Customizable alignment scoring	✔	✔	✗	✗
Read file partitioning	✔	✗	✗	✗

Table 4.1: Comparison of BisPin features

In bisulfite PCR amplification, four sequences are possible: the original forward strand, the original reverse strand, the reverse complement to the forward strand, and the reverse complement to the reverse strand. The directional construction strategy sequences the original forward and reverse strands [83]. Post-Bisulfite Adapter Tagging (PBAT) sequences the reverse complements to the original forward and reverse strands [96], and the nondirectional strategy sequences all strands [23]. Read mappers such as Bismark and Walt support mapping data with these construction strategies, and data generated with each of these methods is common. The hairpin construction strategy uses the Illumina paired-end layout to sequence the forward strand as well as the matching reverse strand with a hairpin adapter that connects the two strands [71]. The technique is especially powerful for bisulfite sequencing, as it allows for the recovery of the original untreated strand before bisulfite treatment.

This is called *hairpin recovery*. This strategy was shown in previous research to improve mapping efficiency by 10% [118]. No known read mappers support special processing for the hairpin construction strategy except BisPin. BisPin has special processing for methylation calling with hairpin data, as the hairpin recovery allows the distinction between single nucleotide variants and an unmethylated cytosine.

BisPin's postprocessing of the BFAST raw SAM files can be done in parallel with multiprocessing, which involves six processes. Postprocessing involves choosing the highest scoring alignment for each read, rescoring ambiguously mapped reads, calling the methylation string, updating the SAM record fields, and printing a summary report.

BisPin produces a methylation calling string by matching the aligned read base to the reference base. This is done similarly to Bismark [70]. More sophisticated methods exist for methylation calling that involve read coverage and probabilistic inference [85]. For hairpin recovered data, the read sequence is compared to the recovered untreated read if available instead of the reference.

Rescoring Ambiguously Mapped Reads

Compared to regular short read mapping, bisulfite short read mapping produces much more ambiguously mapped reads, which are reads mapped to multiple locations [147]. Thus, it is important to have a strategy to distinguish the ambiguously mapped reads.

BisPin employs a simple and fast rescoring technique to disambiguate reads mapped to multiple locations. The rescoring technique scores different mismatches between the read and the reference differently. The matrix that represents this function can be completely specified by the user. More complicated and slower methods for doing this exist, such as Bayesian inference [149], and few read mappers integrate any such disambiguation except

for random assignment.

The rescoring technique is a linear function in the mismatches, matches, and gaps. BisPin examines the alignment of a read to the reference genome as determined by BFAST. If the nucleotide bases match, a positive value is assigned based on the identity of the matching nucleotide base. If the bases do not match, a negative score is assigned based on the identity of the bases. Gaps are scored as they usually are. There is a score for the length of the gap and for the existence of the gap (a gap open). Different scores can be assigned for insertions and deletions. The matrix used to determine the score is the HOXD matrix in [165]. The gap open score is -400, and the gap extenstion score is -30. The gap function and the scoring matrix are the same as in Blastz [130]. The maximum scoring location, if one exists, is used to assign the read to a uniquely mapped location. This increases the uniquely mapped number of reads. The rescoring algorithm is given with Algorithm 4.

The rescoring matrix and the default BisPin alignment function are appropriate for bisuflite data and mammalian genomes. The rescoring matrix represents ratios of aligned nucleotide frequencies from non-coding mouse and human genomic regions [165]. The average matching score over all bases is used for C to T matches (alternatively, G to A matches) for the bisulfite conversion case. Bases other than C and T (G and A) should be unaffected by bisulfite treatment, so the matching and mismatching scores for those regions are the same as for regular untreated reads; thus, the functions chosen for scoring are appropriate for bisulfite data.

If the default rescoring matrix and the alignment function are inappropriate for the data, then the user can set another function. Furthermore, the exact score for the C to T (or G to A) match for bisulfite converted reads can be set with an alternative rescoring function that uses two matrices in the format given in [38]; however, the matrices in that paper were based on a uniform distribution of bases, and this does little to disambiguate ambiguous

Algorithm 4 The BisPin rescoring algorithm.

```
1: A the alignment of a read to the reference; R the rescoring matrix
2: O the gap open penalty; E the gap extension penalty
3: procedure RESCORE(A, R, O, E)
4:     s ← 0
5:     for a_i ∈ A do
6:         if a_i is a methylation call (CT or GA mismatch) then
7:             s ← s + the average of all matching scores
8:         else if a_i is another mismatch then
9:             a_i^g ← the genome base of a_i
10:            a_i^r ← the read base of a_i
11:            s ← s + R[a_i^g][a_i^r]
12:        else if a_i is a match then
13:            a_i^r ← the read base of a_i
14:            s ← s + R[a_i^r][a_i^r]
15:        else if a_i is a gap open then
16:            a_i^d ← indicator for an insertion or deletion
17:            s ← s + O[a_i^d] + E[a_i^d]
18:        else if a_i is a gap extension then
19:            a_i^d ← indicator for an insertion or deletion
20:            s ← s + E[a_i^d]
21:    return s
```

reads. In real genomes, DNA bases are not uniformly distributed [9]. On one extreme there are some Actinobacteria with GC content as high as 70% [151], and on the other extreme there is *Plasmodium falciparum* with 80% AT content [45]. The alignment function can affect the quality of results [140], so a biologically motivated scoring function, as BisPin uses, makes sense. Bismark appears to use an arbitrarily chosen alignment scoring function. A representation of the rescoring function is given in Figure 4.1.

Hairpin Recovery

BisPin includes special processing for the hairpin construction strategy. This data uses a hairpin connector to connect the Watson and Crick strands, and then Illumina paired-end sequencing is performed to sequence the two strands. The strands can be matched together

$$\text{Rescore} = \begin{array}{c} \\ A \\ C \\ G \\ T \\ N \end{array} \begin{array}{c} \begin{array}{ccccc} A & C & G & T & N \end{array} \\ \begin{bmatrix} 91 & -114 & -31 & -123 & -100 \\ -114 & 100 & -125 & -31 & -100 \\ -31 & -125 & 100 & -114 & -100 \\ -123 & -31 & -114 & 91 & -100 \\ -100 & -100 & -100 & -100 & 100 \end{bmatrix} \end{array}$$

gap open [I, D] = $[-400, -400]$
gap extension [I, D] = $[-30, -30]$

Figure 4.1: This describes the default rescoring function used to disambiguate ambiguously mapped reads. It is taken from Blastz [130, 165]. The rows of the matrix refer to the reference genome, and the columns refer to the read string. For gap opening and extension, different values can be used for insertions (I) and deletions (D). BisPin uses the average matching and mismatching scores from this matrix as default values for doing the initial alignments with BFAST.

to allow a recovery of the original strand untreated by bisulfite. This strategy is called *hairpin recovery*. A thorough description can be found in the paper [118]. For this data, BisPin recovers the original strand and uses BFAST to align it. However, not all strands will perfectly match due to sequencing error, so BisPin aligns these as regular bisulfite-treated reads either in a paired-end layout or in a single-end layout.

4.2.2 BFAST-Gap Implementation

BFAST-Gap implements an adaptive weight to the gap open and gap extension penalties of the Smith-Waterman algorithm based on the length of the homopolymer run in the read. There are four function options for determining the weight: constant, logistic, exponential, and piecewise constant. The constant function is the regular method of alignment scoring that does nothing special for homopolymer runs. The exponential model opened gaps too frequently in early tests, and the piecewise constant function was thought to be too simple. For these reasons, these functions were not thoroughly examined but are provided as is. The logistic function is the most versatile, since it has portions that resemble exponential

growth, exponential decay, and linear growth. Since read length is effectively bounded, the logistic function can be used to approximate exponential growth, exponential decay, and linear growth with appropriate arguments. The exponential portions of the function can approximate low order polynomials. The BFAST-Gap software is freely available at https://github.com/JacobPorter/BFAST-Gap/.

To score alignments between the read r_1 and the reference r_2, the following constants must be defined. Suppose that the initial gap open score is given as g_o, and the initial gap extension score is given as g_e. The score for two matching characters is m_s, and the score for two mismatched characters is m_n. The length of the homopolymer run at position i in read r_1 is given as $D[i]$. The constants g_o, g_e, and m_n are negative integers, and the constant m_s is a positive integer.

The logistic gap open function G_o is a function of the homopolymer run length $D[i]$ with slope s_o and center c_o. It is given by the following:

$$G_o(D[i]) = \frac{g_o - m_n}{1 + e^{s_o \cdot (D[i] - c_o)}} + m_n.$$

The constant g_o, the constant gap open penalty, gives the maximum value that the function can give, and the constant m_n, the mismatch penalty, gives the minimum value.

The logistic gap extension function G_e, a function of the homopolymer run length $D[i]$, has slope s_e, center c_e, and minimum value $z = -1.0$, and it is given by the following:

$$G_e(D[i]) = \frac{g_e - z}{1 + e^{s_e \cdot (D[i] - c_e)}} + z.$$

The maximum value that the function can give is g_e, the constant gap extension penalty.

These functions are precomputed once for every execution of BFAST-Gap by storing the function values in a lookup table, since the run length parameter, $D[i]$, is discrete and effectively bounded by the maximum read length.

The run-length array is calculated once for every read by Algorithm 5. This algorithm scans through a read r and initially records in array D, at line 9, the length of a homopolymer run length up to position i. When a new homopolymer run is detected by detecting a change in the DNA base stored in b, the algorithm updates all the values in D associated with the homopolymer run with the total length of the homopolymer run with the **for** loop at line 11. The algorithm returns the array D when the outer **for** loop terminates. This algorithm has time in $\Theta(|r|)$, since the outer **for** loop at line 6 iterates over every base in r and the inner loop at line 11 iterates over every base in every homopolymer run, which comprises every base of the read.

Algorithm 5 An algorithm to calculate homopolymer run lengths for each position of a read.

```
1:  r ← a string over Σ_N representing a read
2:  procedure HomoRuns(r)
3:      D ← an array of length |r| with d[0] = 1. Used to store the lengths of the runs.
4:      s ← 0, a variable to keep track of the start of a run
5:      b ← r[0], a variable to keep track of the DNA base of a run
6:      for i ← 1, |r| do
7:          t ← r[i], a base for testing if a run continues or stops
8:          if b = t then                                   ▷ The run continues.
9:              D[i] ← D[i − 1] + 1
10:         else                                            ▷ The run has ended.
11:             for j ← s, i do          ▷ Update all values in the run to the same thing
12:                 D[j] ← D[i − 1]
13:             b ← t                ▷ Update the values to indicate the start of a new run
14:             D[i] ← 1
15:             s ← i
16:     return D after updating the final values by repeating the inner loop of the preceding.
```

Once the run-length array D is computed, the alignment score and backtrace and score

matrices are computed with Algorithm 6, the Run Length Smith-Waterman algorithm. This algorithm is a similar dynamic programming algorithm to the canonical Smith-Waterman algorithm except that the gap penalties are determined by the gap functions G_o and G_e. The Run Length Smith-Waterman algorithm allows for negative alignment scores and an affine gap penalty. This algorithm maintains four matrices. Matrix H is for deletions, and matrix V is for insertions. Matrix S keeps track of the alignment score, and B stores the backtrace as in the canonical Smith-Waterman algorithm. The backtrace is used to compute the alignment of the read to the reference genome.

The first column and row of each matrix is initialized so that an insertion may start an alignment but not a deletion by initializing the H matrix row and column to negative infinity and initializing the column for the score matrix S and the insertion matrix H to the value of the gap penalty for the appropriate length of the gap. The loops at lines 21 and 22 update the interior of these matrices with the given recurrence. Finally, the position with maximum value from the bottom row is returned along with the matrices S and B. From these elements, the alignment of the entire read locally aligned to the reference genome can be computed as with the canonical Smith-Waterman algorithm using the B matrix. The algorithm has time complexity in $\Theta(|r_1||r_2|)$, since the nested loops at lines 21 and 22 iterate over both the read r_1 and the reference r_2. The algorithm has space complexity in $\Theta(|r_1||r_2|)$, because there are a constant number of arrays of size $|r_1|$ by $|r_2|$.

4.2.3 Data Analysis Methods

To assess BisPin (with BFAST-v0.7.0a), BisPin was compared to Bismark (v0.16.3 with bowtie2-2.2.9), BWA-Meth (downloaded Jan 2017 with BWA-0.7.12-r1039), and Walt (1.0) using the Genome Research Consortium's primary assemblies of the mouse genome (GRCm38

Algorithm 6 The Smith-Waterman algorithm with gap open and gap extension weights determined by the homopolymer run length.

$r_1 \leftarrow$ a string over $\Sigma_{\mathcal{N}}$ representing a read
$r_2 \leftarrow$ a string over $\Sigma_{\mathcal{N}}$ representing the reference genome
$D \leftarrow$ an array representing run lengths for each position in the read
$A_f \leftarrow \{g_o, g_e, m_s, m_n, s_o, c_o, s_e, c_e, z\}$, parameters to determine the alignment function.
$G_o \leftarrow$ the gap open penalty function
$G_e \leftarrow$ the gap extension penatly function
procedure RLSW(r_1, r_2, D, A_f, G_o, G_e)
    $H \leftarrow$ a $|r_1|$ by $|r_2|$ numeric matrix for deletions
    $V \leftarrow$ a $|r_1|$ by $|r_2|$ numeric matrix for insertions
    $S \leftarrow$ a $|r_1|$ by $|r_2|$ numeric matrix for the alignment score
    $B \leftarrow$ a $|r_1|$ by $|r_2|$ matrix for the backtrace
    for $i \leftarrow 0, |r_2|$ **do**
        $H[0, i] = -\infty$
        $V[0, i] = -\infty$
        $S[0, i] = 0$
    for $j \leftarrow 1, |r_1|$ **do** ▷ Initialize the column so that an insertion is possible
        $H[j, 0] = -\infty$
        $V[j, 0] = G_o(D[j-1]) + (j-1)G_e(D[j-1])$
        $S[j, 0] = G_o(D[j-1]) + (j-1)G_e(D[j-1])$
    Update the backtrace matrix B
    for $i \leftarrow 1, |r_1|$ **do**
        for $j \leftarrow 1, |r_2|$ **do**

$$H[i,j] = \max \begin{Bmatrix} H[i, j-1] + G_e(D[i-1]) \\ S[i, j-1] + G_o(D[i-1]) \end{Bmatrix}$$

$$V[i,j] = \max \begin{Bmatrix} V[i-1, j] + G_e(D[i-1]) \\ S[i-1, j] + G_o(D[i-1]) \end{Bmatrix}$$

$$S[i,j] = \max \left\{ \begin{array}{c} S[i-1, j-1] = + \begin{Bmatrix} m_s \text{ if } r_1[i] = r_2[j] \\ m_n \text{ if } r_1[i] \neq r_2[j] \end{Bmatrix} \\ H[i,j] \\ V[i,j] \end{array} \right\}$$

            Update the backtrace matrix B
    $l \leftarrow$ the location of the position in S along the bottom row that has maximum value.
    return S, l, B

p5) and the Arabidopsis Information Resource (TAIR) version 10 of the *A. thaliana* genome. All tests were run on the Computer Science Department of Virginia Tech's bioinformatics machine, mnemosyne2, running Red Hat Linux 4.8.5-4 with 132 GB of RAM and 16 cores

comprising the Intel(R) Xeon(R) CPU E5620 @ 2.40GHz. Timing results were calculated with the Linux `time` command except for BisPin's postprocessing, which used Python's `datetime` module. For Ion Torrent reads only, Tabsat version 1.0.1 with TMAP version 3.4.1 was used. BFAST-Gap on regular Ion Torrent reads was compared with TMAP, Soap2 (2.21), BWA, and Bowtie2 on default settings. TMAP used the map4 algorithm.

BFAST can be run with multiple indexes that are divided into two categories: primary and secondary indexes. Whenever BFAST or BFAST-Gap were run with multiple indexes, only a single index was used as a primary index to increase run-time. Secondary indexes are used only if a match for a read cannot be found in the primary index. Unless indicated otherwise, the primary index had the mask "1111111111111111111", and a secondary index had the mask "11111111100111111111."

Simulated Data

Two data sets were generated to simulate Illumina reads. Simulations were performed with Sherman v0.1.7 from Babraham Bioinformatics [7] on the mouse genome with realistic settings of error 2, CG context 20, and CHH context 98 [169]. Different read lengths were used to simulate variety in read mapping tasks. The first data set consisted of ten sets of one million paired-end 75bp reads. The second data set consisted of ten sets of one million single-end 150bp reads.

All simulated data was aligned in the same fashion as the real data. A Python script was developed to check the accuracy of read mappers. These and other Python scripts are available with the BisPin code. Only uniquely mapped reads were checked, and they were considered accurately mapped if the starting location was within three bases of the real location. Average precision and recall were computed. Precision is the proportion of the

uniquely mapped reads that are correctly mapped of the total uniquely mapped reads, and recall is the proportion of the uniquely mapped reads that are correctly mapped of the total number of reads.

A test to compare the rescoring function against a random choice was performed to see if the rescored read was more likely to align correctly than random chance. This test used 100,000 reads from the 150bp single-end simulated data. The set of rescored ambiguously mapped reads was extracted, and for each read, a random alignment location was chosen. The percentage of these that were correctly mapped was compared to the percentage of the rescored reads that were correctly mapped. This process was repeated 32 times to calculate a p-value.

The preceding Sherman simulated data was used to test BisPin on simulated bisulfite-treated Illumina reads. BisPin with BFAST and BFAST-Gap were tested on simulated Ion Torrent reads generated by DWGSIM since it simulates the undercalling and overcalling of homopolymer runs found in Ion Torrent reads [54]. For DWGSIM, the bisulfite treatment was simulated on the reference genome with the CpG rate as 0.215, the CH rate 0.995, and the over conversion and under conversion rates of 0.0025 [147, 169]. This was done with custom Python scripts provided by Hong Tran. Custom Python scripts were used to select single-end data from DWGSIM. DWGSIM was used with the following realistic settings: "dwgsim -e 0.012 -E 0.012 -d 250 -s 30 -S 0 -N 1000000 -c 2 -1 200 -2 200 -f TACGTACGTCTGAG-CATCGATCGATGTACAGC" [5].

These settings were used to produce two data sets, one data set for simulated bisulfite-treated reads, and one data set for regular Ion Torrent reads. The simulated bisulfite-treated Ion Torrent reads were divided into a 10k read training set and a one million read test set to tune the logistic gap penalty functions.

The logistic gap open function for BFAST-Gap was trained on the 10k read training set by looking at slopes from 1.2 to 0.5 and centers from -20 to 150. A single index was used with mask "11111111100111111111." The alignment function had values of 96 for a match, -90 for a mismatch, -600 for a gap open penalty, and -50 for a gap extension penalty. The optimal slope ($s_o = 1.0$) and center ($c_o = -15$) were chosen based on which values maximized the F1-score and the area under the curve for all BisPin filter values from 0 to 96.

Once the logistic gap open function was tuned, the settings for the logistic gap open function were set, and the logistic gap extension function was tuned in a similar manner. This resulted in a slope of $s_e = 0.1$ and a center of $c_e = 90$. The tuned logistic gap open and extension functions are considered the default settings for BFAST-Gap. Figure 4.2 gives the tuned logistic gap extension penalty function. This function is approximately constant until a run length of approximately 50 base pairs.

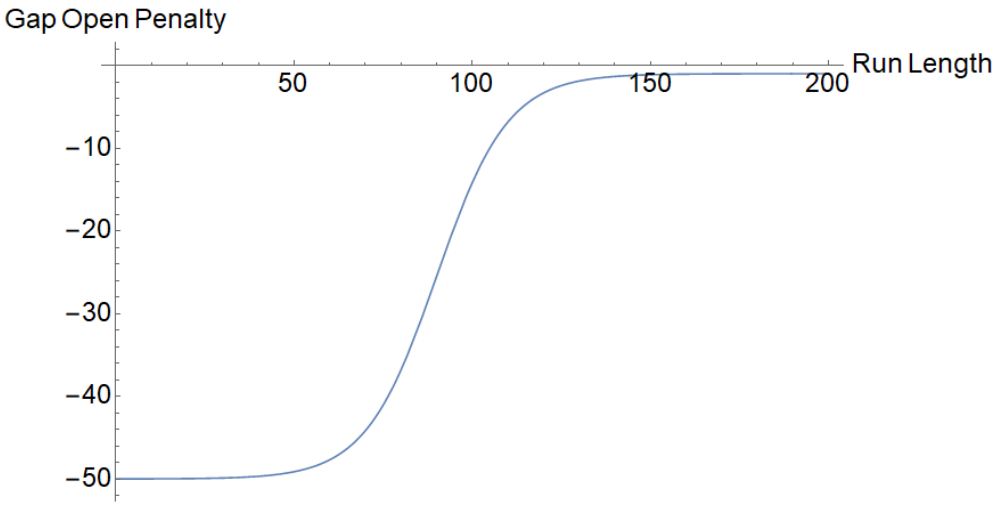


Figure 4.2: An example of a logistic function for the gap extension penalty for BFAST-Gap. This function has $g_e - z = -49$, $m_n = -50$, $s_e = 0.1$, and $c_e = 90$.

The entire one million test set was mapped with BisPin calling BFAST-Gap with these settings. One execution was run with constant open and extension penalties, another execution was run with the logistic open penalty function and a constant extension penalty function,

and a third execution was run with logistic open and extension penalty functions with the tuned parameters. The F1-score was calculated such that an alignment was considered correct if it was within three bases of the correct location on the reference genome. BisPin used two indexes, and rescoring for Ion Torrent reads was turned off since the floating point logistic function tends to resolve ambiguously mapped reads.

For one million DWGSIM simulated regular Ion Torrent 200bp reads, BFAST-Gap was run with the tuned logistic gap open and logistic gap extension penalty functions without a filter and two indexes. TMAP did not have a filter and was run with default settings for the map4 algorithm. Bowtie2, BWA, and Soap2 were run with default settings. Soap2 could not be accuracy tested since it does not produce a SAM file. For the simulated regular Ion Torrent data, the precision and recall were calculated with the same Python script as other simulations.

Real Data

Variegated real data was downloaded from the SRA (Sequence Read Archive) [74]. Each Illumina data set had reads of approximately 100bp length. The mouse data used 100k reads, and the plant data used 500k reads. Only SRR4295457 data was paired-end. The rest were single-end. Hairpin 101bp mouse data was constructed with the procedure given in [169]. Table 4.2 gives a summary of the real Illumina read data used in this study. The mapping efficiency was calculated with each mapper's self report except for BWA-Meth since it did not give such a report. For BWA-Meth, a Python script was created to determine the mapping efficiency. If a read had only one location, it was uniquely mapped. If it had several locations including locations given in the `XA:Z` tag used by BWA, the read was classified as ambiguously mapped. If the alignment had the unmapped or filtered flag set, it was classified as unmapped. The default settings of all mappers were used.

Organism	SRA #	Read Len.	Read Amt.	Layout	Construction	Sequencer
Mus *musculus*	SRR921759	101	100k	Single	Directional	HiSeq 2000
Mus *musculus*	DRR053271	96	100k	Single	PBAT	-
A. *thaliana*	SRR5014638	100	500k	Single	Nondirectional	HiSeq 4000
A. *thaliana*	SRR4295457	101	500k	Paired	Directional	HiSeq 2500

Table 4.2: A summary of the real Illumina read data features. All data was obtained from the Sequence Read Archive (SRA) at https://www.ncbi.nlm.nih.gov/sra.

Different data sets came from different sequencing construction strategies, which included directional (BS-Seq), nondirectional (MethylC-Seq), and PBAT constructions as indicated in Table 4.2. If a read mapper did not support the sequencing construction strategy, the read mapper was not run on that data set. Support for these construction strategies is indicated in the read mapper's interface. A discussion of construction strategies is found in Section 1.3.2.

BisPin used one primary index and two secondary indexes for mouse data but only one secondary index for A. *thaliana* data. These indexes work such that if a candidate location cannot be found in the primary index, then the secondary indexes are tried. The primary indexes searched for a seed with a length 20 exact match, and the secondary indexes used spaced seeds. Rescoring was on.

To test the logistic gap open function, the optimal slope and center from the simulated data was used, and BisPin with BFAST-Gap was run on three real data sets without an alignment quality filter using both a constant and a logistic gap open penalty function. The quality filter for BisPin was turned off since the logistic gap open function changes the alignment score making the alignment scores between the two alignments incomparable. The amount of uniquely mapped reads between these data sets indicates which alignment scoring function performs better. The data was downloaded from the SRA trace archive. The real Ion Torrent data used one million reads of mouse data from SRA numbers SRR1534391 and

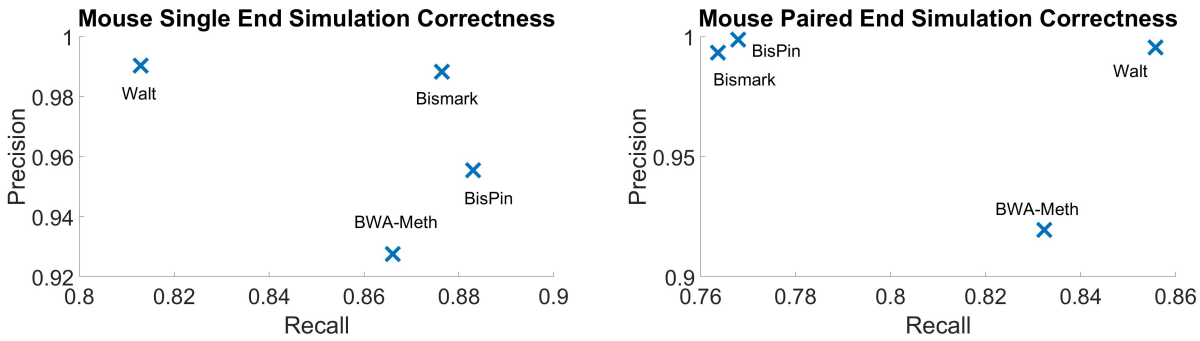
SRR1534392, and it used one million reads of human data from SRA numbers SRR2842546 and SRR2842547. A small human data set was used, consisting of 251,374 reads from SRA number SRR3305017. The data was aligned with one index when testing BisPin with BFAST-Gap and was aligned with two indexes when testing BisPin with BFAST. Rescoring was turned off.

Real mouse and human data was used to test BFAST-Gap with tuned logistic gap open and extension penalty functions on regular Ion Torrent reads. The SRA numbers were mouse ERR699568, human SRR2734774, and human SRR611141. Each data set had one million reads. The regular mapper programs TMAP, Bowtie2, BWA, and Soap2 were used for comparison. The uniquely mapped, ambiguously mapped, and filtered/unmapped percentages were calculated based on the programs self-report for Bowtie2 and Soap2. Custom Python scripts available with the BisPin source code were used to calculate these statistics for BFAST-Gap, TMAP, and BWA. BFAST-Gap was run with a stricter filter of 75 on its default settings with two indexes, and TMAP was run with a filter of 0.625, which approximately corresponds with a BFAST-Gap filter of 60. Other read mappers were run on their default settings.

4.3 Results

4.3.1 Simulated Data

The results for BisPin on simulated Illumina data were reasonable. BisPin had the highest recall for the single-end data and the highest precision for the paired-end data as shown in Figure 4.3. Both BisPin and Bismark correctly called the approximate amount of methylation in each context, 80% for CpG and 2% otherwise.



(a) Single-end directional 150bp data (b) Paired-end directional 75bp data

Figure 4.3: The average of precision and recall over ten replicates of one million reads of uniquely mapped reads on Sherman mouse read simulations with error 2, CG conversion 20, and CH conversion 98. The standard deviation for all statistics for all data was below 0.0005.

Interestingly, the recall generally decreased by approximately 0.10 for the one million paired-end 75bp reads for all the read mappers except for BWA-Meth. The F1-score, a balanced average of precision and recall, for this data was the following, 0.79 (BisPin), 0.80 (Bismark), 0.91 (BWA-Meth), and 0.75 (Walt).

Data consisting of 32 replicates of 100k 150bp single-end reads was used to assess the rescoring functionality. Choosing a random alignment for a read from the set of rescored reads mapped 19.1% correctly on average while the rescoring functionality gave an average of 23.5% correct alignments. The rescoring approach always returned more correctly aligned reads than the random choice with an average difference of 4.4%. This suggests a p-value close to zero and that there is a statistically significant difference in these distributions such that the rescoring functionality is better than random assignment for uniquely aligning some reads.

Using BisPin with both BFAST on default settings and with BFAST-Gap on tuned gap penalty functions, the one million DWGSIM reads were mapped, and precision, recall, and F1-score were calculated as shown in Figure 4.4. BWA-Meth and Bismark had moderate

performance, while Walt performed poorly. These read mappers appear ill-suited for mapping Ion Torrent reads. BisPin and Tabsat performed the best. BisPin with BFAST-Gap had slightly higher precision and slightly lower recall but with a higher F1-score than BisPin with BFAST.

The proportion of reads in each mapping category, uniquely mapped, ambiguously mapped, and unmapped or filtered, on the same simulated data for each read mapper is shown in Figure 4.5. BisPin and Tabsat performed the best with BisPin having about five percent more reads uniquely mapped with few ambiguously mapped. Bismark and BWA-Meth were less good, and Walt performed poorly.

Tuning the logistic gap open function on BFAST-Gap on 10k simulated Ion Torrent reads revealed interesting results, as shown in Figure 4.6. Settings with low slopes tended to perform the worst with low centers performing the best among these. Settings with slopes closer to 1.0 performed the best with centers around 0 to -30; however, with centers much larger than this, performance was worse. From this analysis, a slope of 1.0 and a center of -15.0 was chosen.

After tuning, BisPin with BFAST-Gap was run on the one million simulated reads test set, and the alignment with the tuned logistic gap open function achieved superior area under the curve and F1-score, as can be seen in Figure 4.7. Executions with two indexes and one index were performed, and using two indexes noticeably improved the performance. The maximizing filter value for the constant penalty function was 65, and for the executions involving the logistic function, it was 85. The logistic function tends to reduce alignment scores, so a higher filter value is needed to compensate. The AUC was improved by approximately 7, for the two index case, and the F1-score was improved by 0.003. This indicates that using the logistic gap open function and a constant gap extension penalty function improved mapper performance on this simulation. The executions where both the gap open and extension

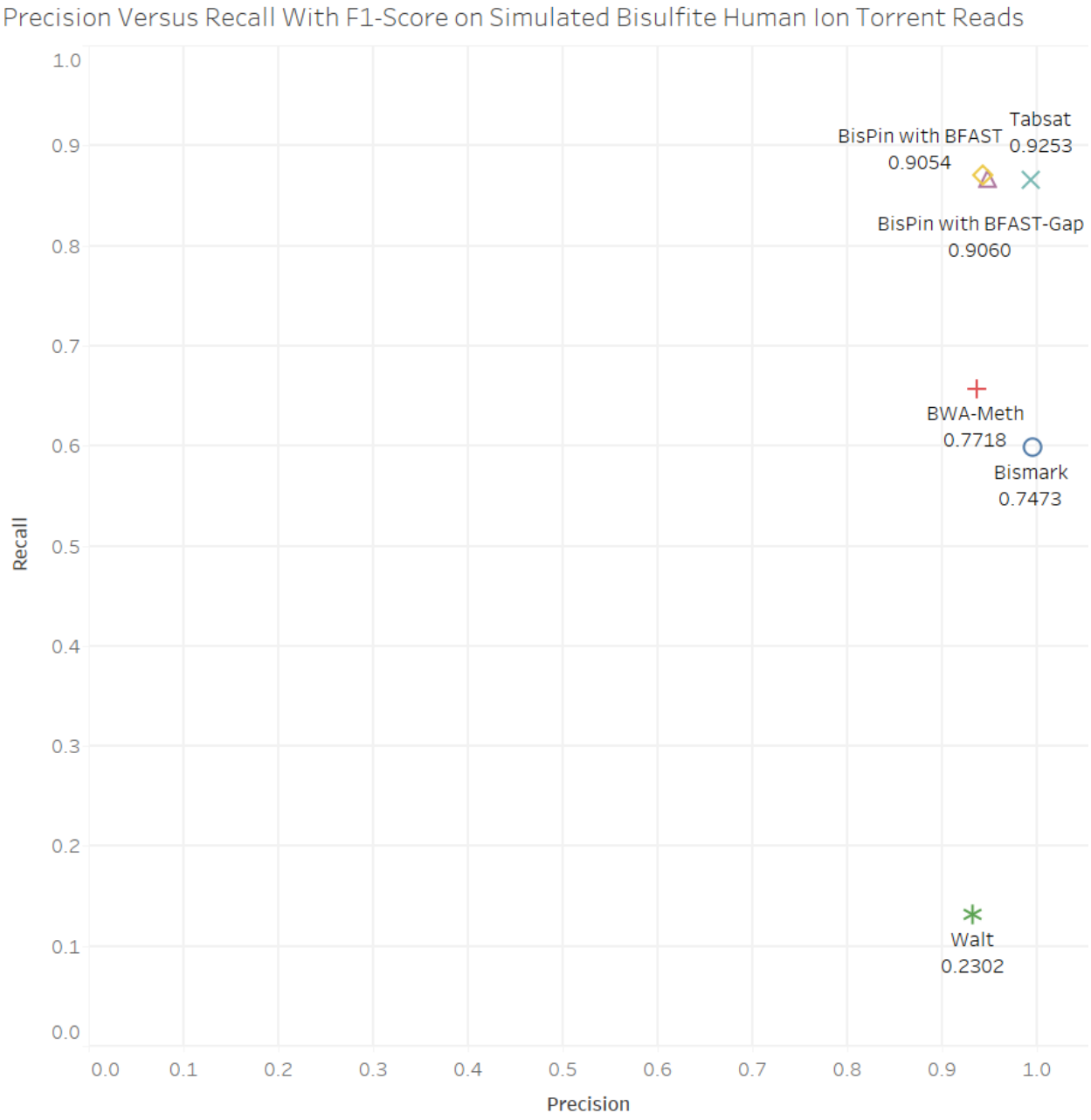


Figure 4.4: Precision versus recall on simulated human bisulfite-treated Ion Torrent reads. The numeric label is the F1-score.

functions were tuned logistic functions performed nearly identically to the executions with the logistic gap open penalty function and a constant gap extension function but with slightly worse performance.

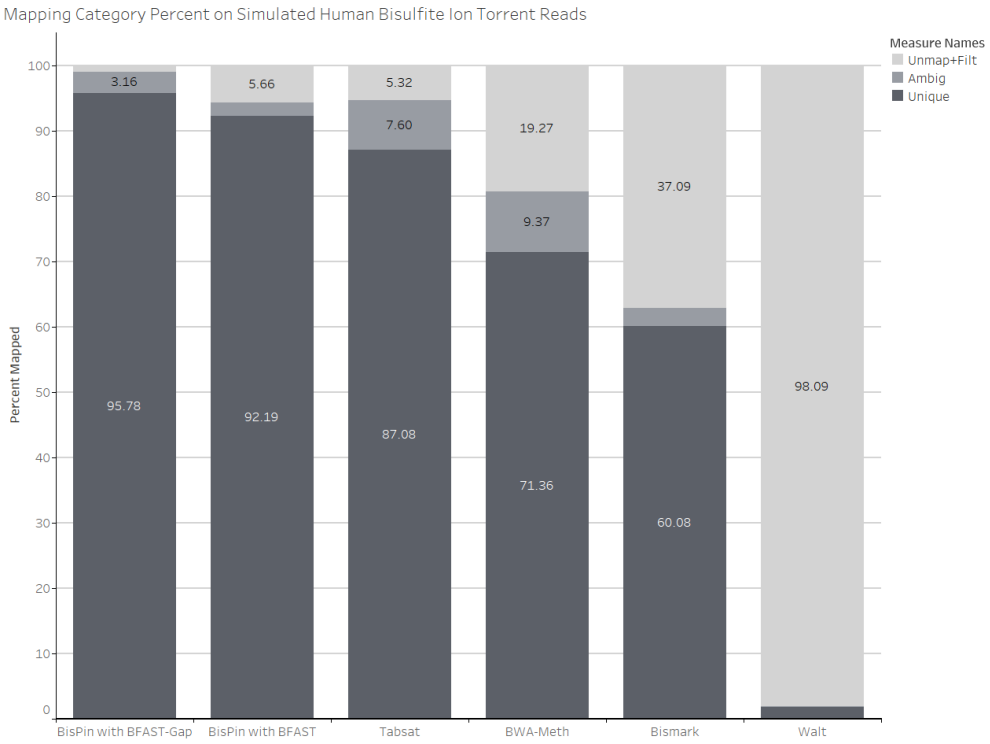


Figure 4.5: Mapping performance by category on one million DWGSIM simulated bisulfite Ion Torrent reads.

Figure 4.8 shows the precision, recall, and $F_{0.5}$-score for mappers on the simulated DWGSIM regular Ion Torrent data. BFAST-Gap had the highest overall score. TMAP performed the worst, which is surprising since it is designed for Ion Torrent reads.

4.3.2 Real Data

The percentage of uniquely mapped reads gives a reasonable test of read mapper performance, and recent papers use this method [19, 48]. The simulated data shows that the percentage of uniquely mapped reads that are correctly mapped is very high. Although the Sherman Illumina simulated data does not include indels, indels are an order of magnitude more rare than SNVs, so they should not affect mapper performance as much [101]. This suggests that many of the uniquely mapped reads are probably aligned correctly.

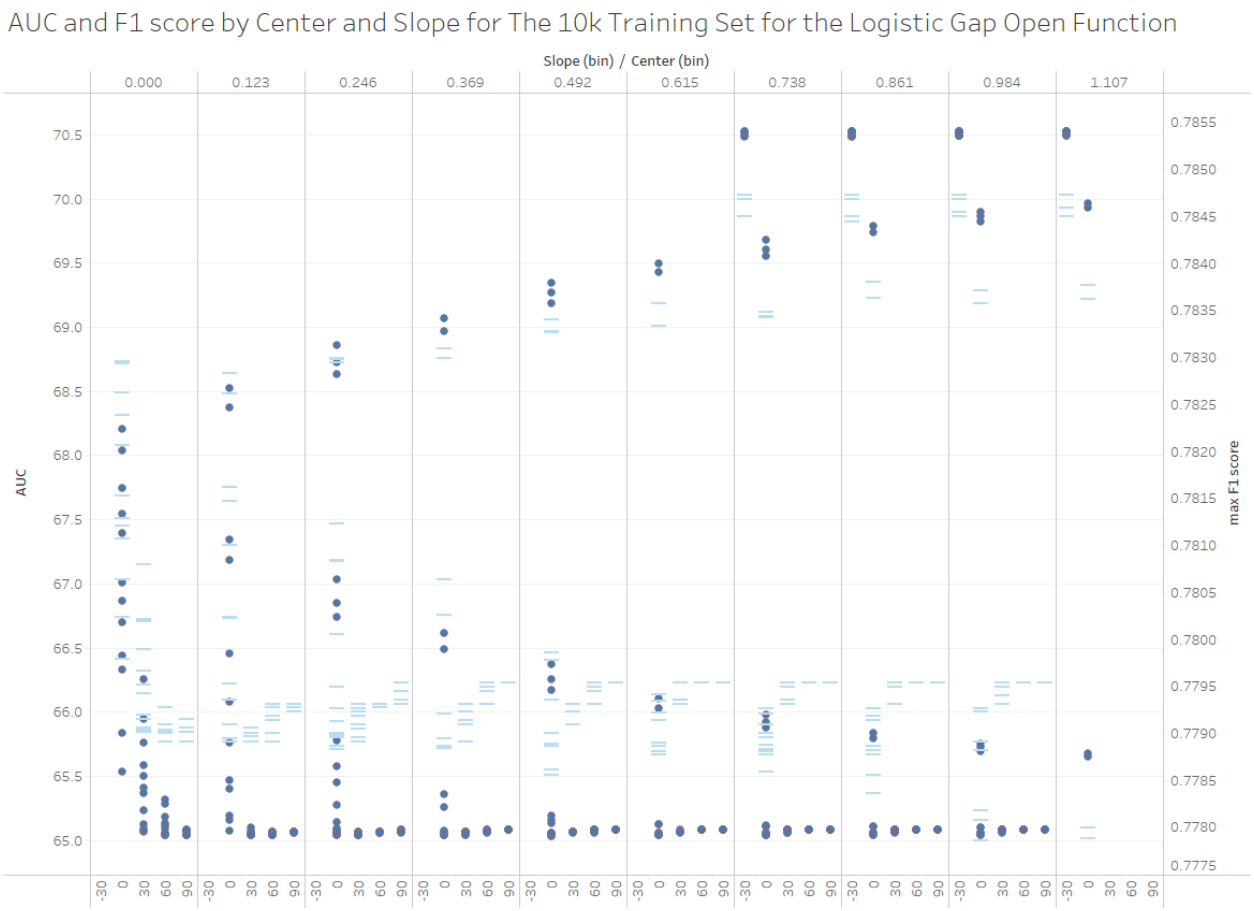


Figure 4.6: AUC and F1-Score by Slope and Center for the Logistic Gap Open Penalty Function on 10k Simulated BS Ion Torrent Reads. The circles represent AUC scores and the bars represent maximum F1-scores for alignment quality filter values ranging from 0 to 96.

Figure 4.9 summarizes the mapping efficiency of the read mappers on the four real data sets. In all cases, BisPin had the most uniquely mapped reads. Without rescoring, BisPin would have reported approximately 10% ambiguously mapped reads, but the rescoring moved from $\frac{1}{3}$ to $\frac{2}{3}$ of these to uniquely mapped. BisPin and Bismark comported on the percentage of methylation in all contexts.

On 100k reads of hairpin mouse embryonic data, BisPin did hairpin recovery on 62% of the reads. BisPin mapped this data as regular paired-end data with 81.7% uniquely mapped, but other read mappers mapped few reads uniquely in paired-end mode. This is probably

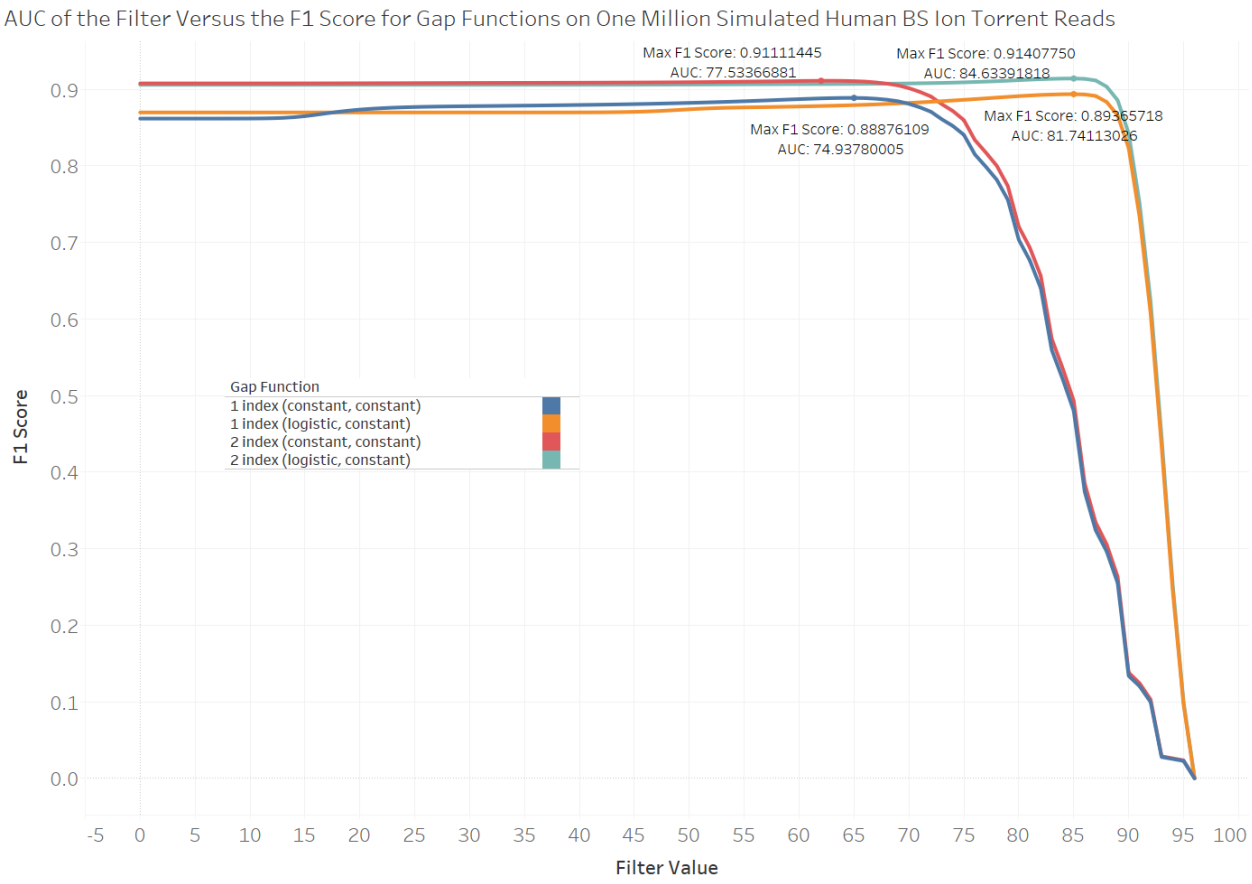


Figure 4.7: Performance of BisPin with the tuned BFAST-Gap logistic gap open function on one million simulated test reads as measured by area under the curve and F1-score using one index and two indexes.

because paired-end mapping usually assumes that one end is upstream of another rather than overlapping as with hairpin data.

A method of validating the correctness of the read mappers was performed with the hairpin data. One million reads were sampled, and 604431 reads were hairpin recovered. These reads were mapped with BFAST, with BisPin's defaults, and with Bowtie2 and BWA on their default settings. BisPin, Bismark, and BWA-Meth were run on the one million bisulfite reads, and the uniquely mapped reads were compared with the uniquely mapped (hairpin recovered) original reads from each regular read mapper. If the starting location of the

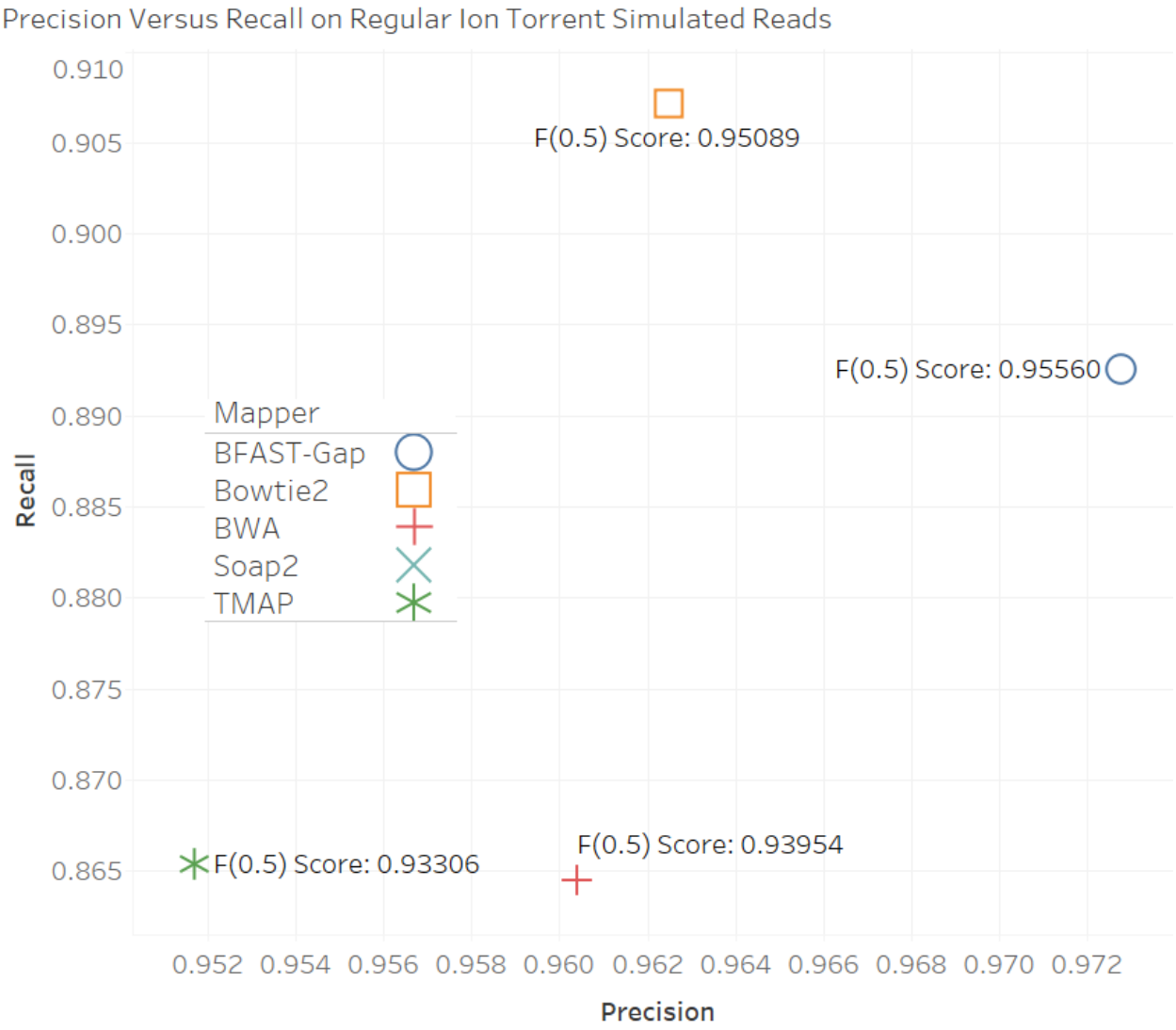
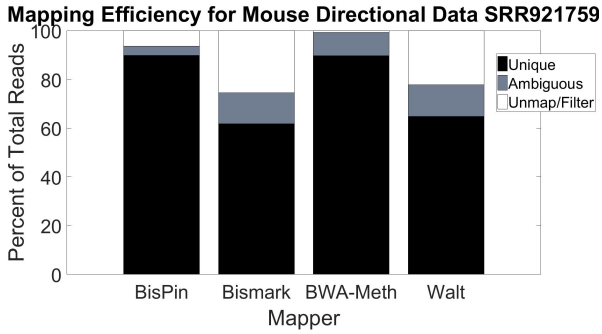
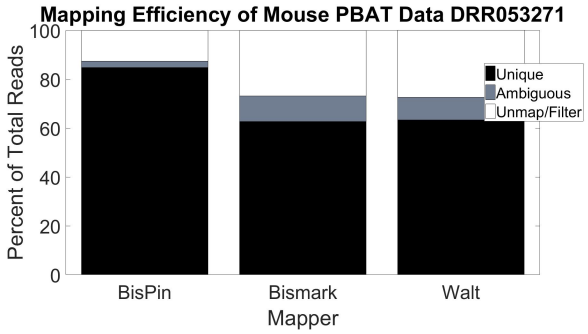


Figure 4.8: The precision, recall, and $F_{0.5}$-score for regular Ion Torrent DWGSIM 1 million simulated 200bp single-end reads with settings e=1.2 and E=1.2. BFAST-Gap has the best $F_{0.5}$ score and the highest precision. Default settings are used throughout. Soap2 is not shown since it did not produce SAM file output.

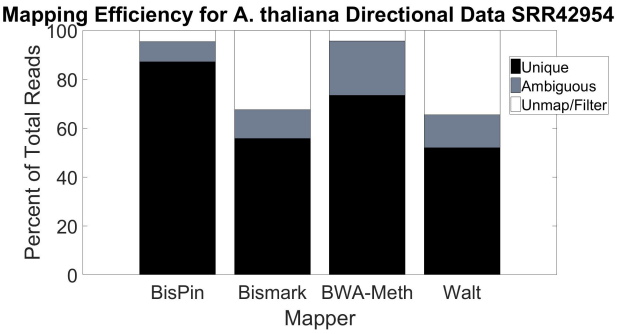
bisulfite read was within three bases of the mapped original read, then it was considered correctly mapped. The motivation for this is that the original read should more often map correctly because bisulfite treatment has a tendency to reduce sequence complexity, which can adversely affect mapping quality [118]. These results are summarized in Table 4.3. BisPin did pretty well with the highest correct using the maximum score.



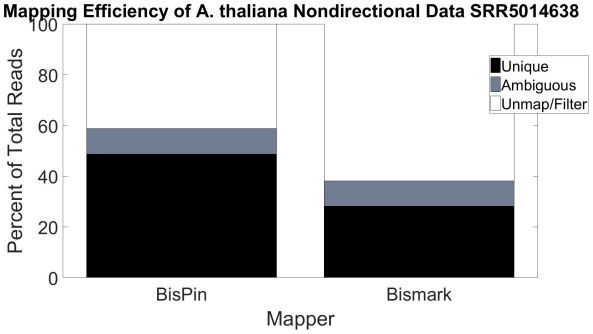
(a) Mouse directional data efficiency



(b) Mouse PBAT data efficiency



(c) A. *thaliana* directional data efficiency



(d) A. *thaliana* nondirectional data efficiency

Figure 4.9: BisPin mapping efficiency on real data compared to other mappers. BisPin always has the highest number of reads uniquely mapped. Figure 4.9c is paired-end data, but all other data is single-end.

Bisulfite Mapper	BFAST	Bowtie2	BWA	Average
BisPin	**73.8161%**	60.5174%	60.1448%	64.8261%
Bismark	**63.3881%**	53.0212%	53.8516%	56.75363%
BWA-Meth	**66.3428%**	56.926%	61.8104%	61.693067%
Walt	**72.2992%**	62.6082%	62.0344%	65.647267%

Table 4.3: Hairpin validation. BFAST, Bowtie2, and BWA were used to map the 604,431 original recovered reads from one million bisulfite reads, and the percent of these recovered reads that were uniquely mapped to the same location with each bisulfite read mapper on the one million bisulfite reads was reported. The maximum value is in bold, and BisPin had the highest value. Walt had the highest average, but BisPin was second with a difference of 0.8%.

To test the performance of BisPin with BFAST on bisulfite Ion Torrent data, it was run on default settings along with other mappers with performance results indicated in Table 4.4.

BisPin had the most uniquely mapped for all five data sets, and Tabsat was second best. For the mouse data, BisPin mapped more than ten percent of the reads uniquely compared to Tabsat. Walt was the worst with very low performance.

Uniquely Mapped % on Mouse BS Ion Torrent Reads

SRA Mouse #	BisPin	Bismark	BWA-Meth	Tabsat	Walt
SRR1534391	91.79	21.20	45.17	80.65	3.79
SRR1534392	91.38	16.60	38.79	80.05	3.77

Uniquely Mapped % on Human BS Ion Torrent Reads

SRA Human #	BisPin	Bismark	BWA-Meth	Tabsat	Walt
SRR2842546	90.00	60.60	70.42	84.43	19.19
SRR2842547	89.33	60.70	68.28	83.93	18.34
SRR3305017	92.63	35.70	65.32	86.67	23.26

Table 4.4: The percent of real bisulfite-treated Ion Torrent reads uniquely mapped on five data sets. BisPin performs the best.

The read mappers were run with their default settings because that is what users are likely to do. Bismark, BWA-Meth, and Walt perform poorly on the bisulfite-treated Ion Torrent reads in Table 4.4 because they may not be tuned to the read lengths and error profile of Ion Torrent data. To test if improvements could be made, tuning parameters were adjusted and the read mappers were run again on the mouse SRR1534391 data. BWA-Meth offers no tuning parameters, so there was no attempt at improving its performance. Bismark has a minimum score function that filters out low quality alignments. This was changed to 'L,0,-0.8' and this produced 76.6% uniquely mapped reads, over 50% of an improvement. Walt has a maximum number of mismatches parameter with a default of 6, and increasing this to 175 increased the uniquely mapped percent to 17.31%, only approximately 13% of an improvement. This was the only parameter that could be tuned to make an improvement. Thus, Bismark's performance can be greatly improved, but Walt's performance improvement

seems limited for Ion Torrent data. Walt filters out reads less than 38 base pairs in length, and this was 8.56% of the SRR1534391 data. This makes BWA-Meth and Walt less suitable for Ion Torrent data.

Since Ion Torrent reads can vary in length, a test of mapper performance in relation to read length from the SRR2842547 and SRR2842546 data was performed. Figure 4.10 gives a histogram of read lengths from the SRR1534392 data. This distribution has a long tail with a significant mode at the high end of the distribution indicating that many reads have a large length, but a substantial amount have short lengths.

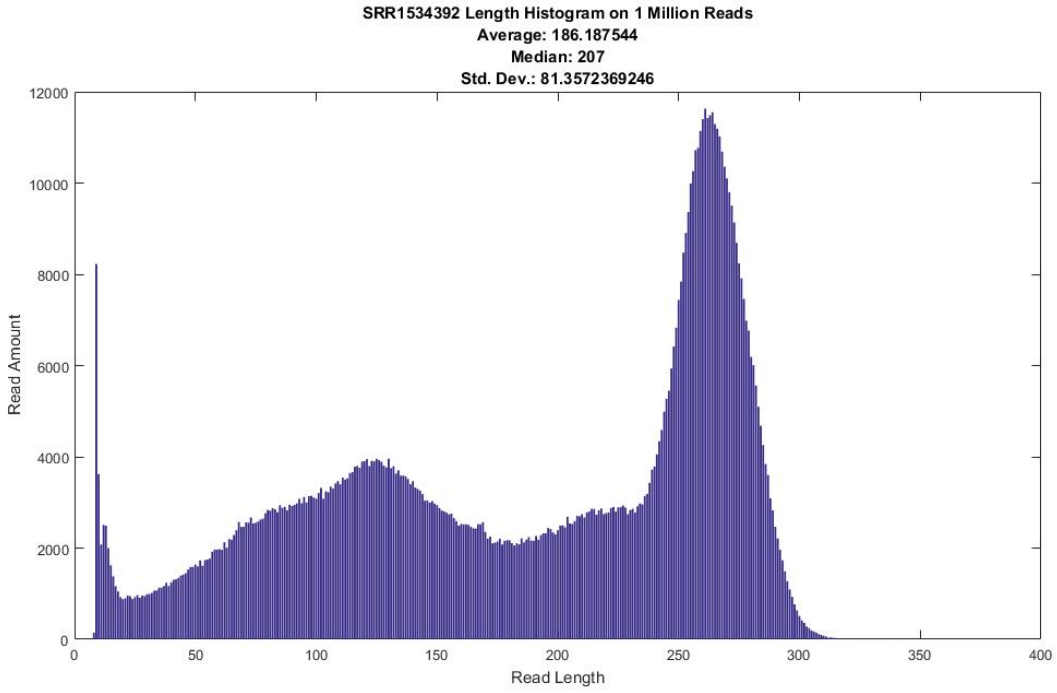


Figure 4.10: A read length histogram for Ion Torrent reads from data set SRR1534392.

Buckets, each consisting of 500k reads, were created at assorted read lengths from the SRR2842547 and SRR2842546 data, and the uniquely mapped percent was calculated for each mapper with the results visualized in Figure 4.11. Except for Walt, mapper performance tended to increase with higher read length; however, for BisPin, BWA-Meth, and Tabsat,

mapper performance suffered slightly with the longest reads. Bismark had the most dramatic improvement with increasing read length. Both Tabsat and BisPin performed similarily, but BisPin performed the best.

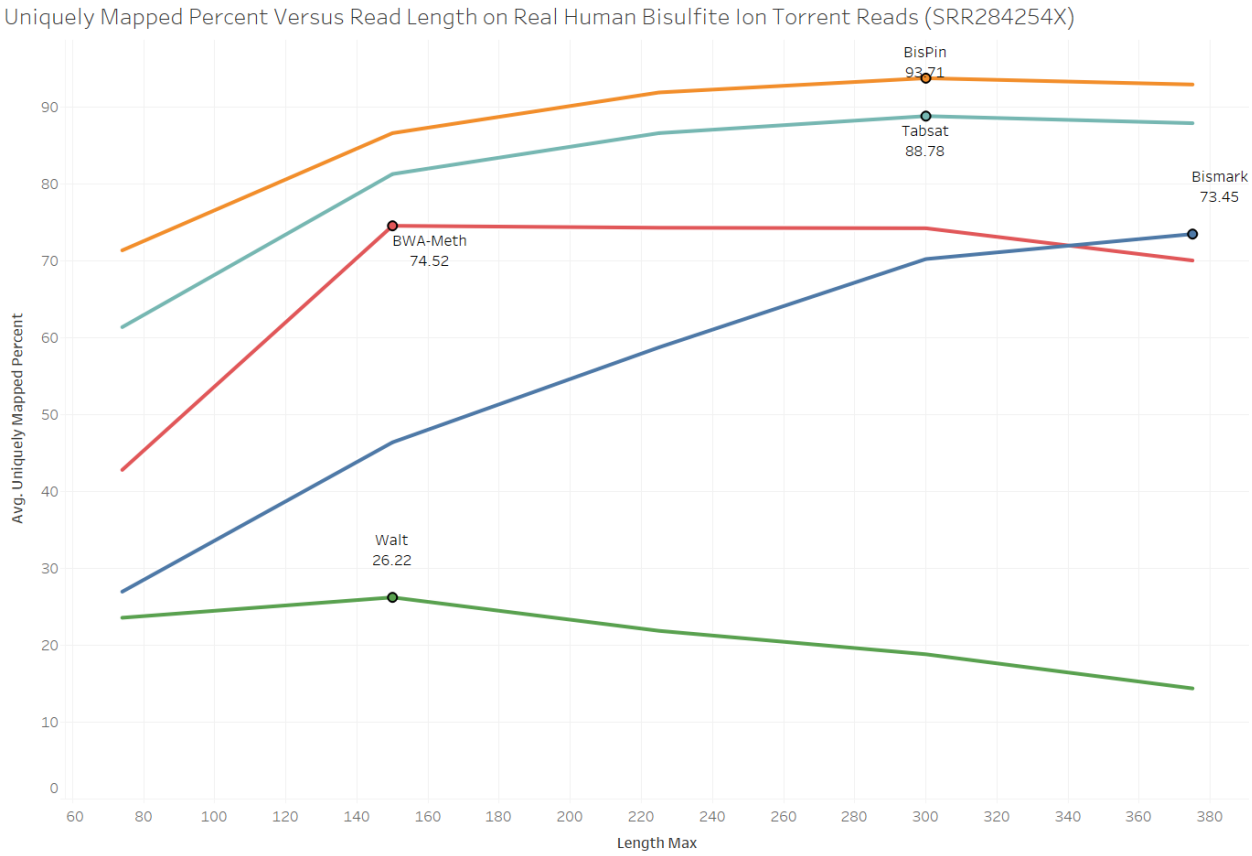


Figure 4.11: Mapper performance from uniquely mapped percent versus read length on real data. The percent is an average of results from data sets SRR2842547 and SRR2842546.

To see if using extra indexes could improve mapper performance for the smallest reads ranging from 9 to 75 base pairs, six indexes of assorted mask lengths with spaced seeds were created and used to align 500k of these reads, but the results were identical to using two indexes. The ineluctable conclusion is that this strategy will have no effect, so improving the indexing strategy was not pursued further.

For the one million bisulfite-treated Ion Torrent mouse SRR1534391 reads, a test of how

similar BisPin with BFAST-Gap results were to Tabsat and BWA-Meth was performed by calculating how many reads there were that mapped to the same location on the genome. A read was considered mapped to the same location for two or more read mappers if it was uniquely aligned and if the genome location differed by no more than three bases. The genome location is given in the SAM alignment file in the RNAME and POS fields. A Venn diagram shows the results in Figure 4.12.

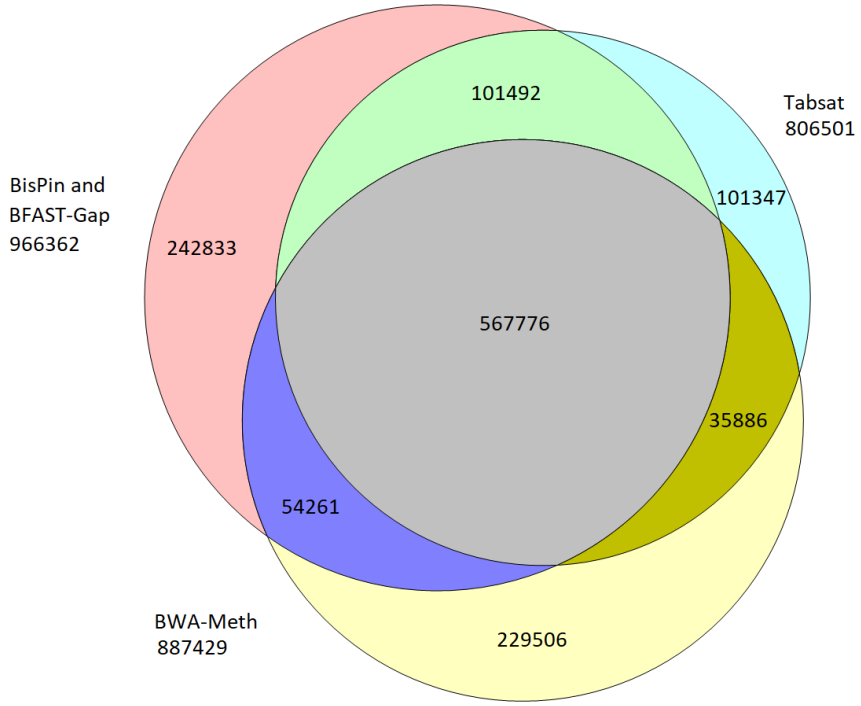


Figure 4.12: A Venn diagram showing how the uniquely mapped reads from SRR1534391 overlap using mappers BisPin with BFAST-Gap, Tabsat, and BWA-Meth. A read was considered to be in an overlapping set if it was mapped to the same location. The diagram was created with Venn Diagram Plotter available at https://omics.pnl.gov/software/venn-diagram-plotter.

The diagram shows that the different read mappers have fairly overlapping results with more than half of the alignments for each read mapper mapping to the same location. This set was 58.758% of BisPin's uniquely mapped reads, and about one quarter of BisPin's uniquely mapped reads had a location unique to BisPin. This shows that BisPin's results

are reasonable and similar to other mappers. The set where all read mappers agreed on the location likely has reads correctly mapped. There were 242,833 reads that were uniquely mapped by BFAST-Gap, where other read mappers did not agree on the mapped position. Of these reads, 90% were uniquely mapped to other locations by other read mappers. Because of this and since the locations reported by the read mappers disagreed on approximately $25\% - 40\%$ of the reads, this suggests that some kind of ensemble method combining the results of the read mappers could be useful. Perhaps a probabilistic or machine learning model could be employed to combine the results of read mappers.

With the tuned logistic gap open function, BisPin with BFAST-Gap was run on one million real and simulated Ion Torrent bisulfite-treated data with no filter. The execution with a logistic gap open penalty and a constant gap extension penalty had slightly more uniquely mapped reads indicating better performance as can be seen in Table 4.5. Using the logistic function for both gap open and gap extension penalty degraded performance a bit for the simulated data and the SRR1534391 data, but it improved performance by six percent on the SRR2842546 data. Two indexes were used for these executions. This table can be compared directly with Table 4.4 to compare BisPin using BFAST-Gap with other mappers. BisPin performed the best with this data.

Gap Function Type	DWGSIM	SRR1534391	SRR2842546
constant	96.31859%	92.25470%	82.59100%
logistic gap open, constant gap extension	96.46848%	92.87040%	83.24190%
logistic gap open, logistic gap extension	95.80231%	92.4691%	89.6092%

Table 4.5: Logistic gap open and extension penalty function performance on real and simulated data as measured by uniquely mapped percent without a filter. Compare with Table 4.4.

Table 4.6 shows mapper performance on three regular Ion Torrent read data sets. BFAST-Gap did the best with high amounts of uniquely mapped reads on data sets ERR699568

and SRR611141, and it was second best behind BWA on data SRR2734774. BFAST-Gap has zero reads ambiguously mapped since the homopolymer run based gap penalty function tends to make reads uniquely mapped since it uses precise decimal number penalties rather than integer penalties. TMAP did well. Soap2 generally performed the most poorly.

Mapper	Unique	Ambiguous	Unmapped or Filtered
Mouse ERR699568			
BFAST-Gap	91.9665%	0.0%	8.0335%
TMAP	86.8102%	7.95%	5.2398%
Bowtie2	64.15%	33.03%	2.82%
BWA	86.9847%	11.1932%	1.8221%
Soap2	25.08%	-	74.92%
Human SRR2734774			
BFAST-Gap	88.6083%	0.0%	11.3917%
TMAP	88.2288%	8.8664%	2.9048%
Bowtie2	77.55%	20.21%	2.23%
BWA	92.4334%	5.8566%	1.71%
Soap2	39.88%	-	60.12%
Human SRR611141			
BFAST-Gap	87.3747%	0.0%	12.6253%
TMAP	85.5568%	9.2704%	5.1728%
Bowtie2	75.06%	21.18%	3.76%
BWA	86.1396%	5.9415%	7.9189%
Soap2	19.7%	-	80.3%

Table 4.6: Regular Ion Torrent mapper performance on three real data sets of one million reads each. The percentage of reads for each category is shown. BFAST-Gap used a tuned logistic function for the gap open and extension penalty functions, and it generally performed well. Soap2 does not distinguish ambiguously mapped reads in its reported statistics.

As with bisulfite-treated Ion Torrent reads from Table 4.4, these mappers are not necessarily tuned for Ion Torrent data, and their performance could be improved by adjusting their parameters. Soap2 can increase the maximum number of mismatches from 5, and it can allow a continuous gap sized larger than 0. BWA and Bowtie2 have a minimum score parameter.

4.3.3 Timing

For alignment time only, on simulated paired-end 250k 100bp reads data, BisPin took 103.5 minutes, Bismark took 6.5 minutes, BWA-Meth 3.5 minutes, and Walt 5.5 minutes. BisPin is approximately 20 times slower, because BFAST is slow. For one million reads, multiprocessing improved BisPin postprocessing speed, excluding the reference genome load time, by approximately 1.29 times.

On one million reads from the mouse regular Ion Torrent data (SRA#: ERR699568) using 30 threads on a machine with 32 processing cores (E5-2660 @ 2.20GHz with 198 GB of memory), BFAST-Gap took about one hour. Bowtie2 took 5 minutes, and TMAP took 2 minutes. BWA and Soap2 each took approximately half a minute. BFAST-Gap is substantially slower.

4.4 Conclusion

Mapping bisulfite reads is challenging. BisPin employs several strategies to address these difficulties. It uses rescoring to disambiguate some ambiguously mapped reads, and it can perform extra processing with additional indexes to attempt to align unmapped reads. It uses multiprocessing and multithreading for improved run-time efficiency, and it can align a selected partition of reads from a file for deployment on a compute cluster. BisPin supports a variety of popular read constructions and layouts including the hairpin construction strategy, which is eminently useful for bisulfite-treated reads. BisPin performed well with a variety of real and simulated data compared to other read mappers.

BisPin with BFAST-Gap improves upon read mapping with bisulfite-treated Ion Torrent reads, and BFAST-Gap improves upon read mapping for regular Ion Torrent reads compared to other read mappers including TMAP, which was designed for Ion Torrent reads, in some

ways.

For improving run-time, BFAST could be modified to include a highly optimized SIMD implementation of the Smith-Waterman algorithm as in Bowtie2, and it could dispense with intermediate files, which are slow to create. BFAST has parameters that could be adjusted that might improve run-time at the expense of accuracy. An example of this is filtering out reads that match to too many locations. Perhaps BisPin's hairpin sequencing approaches could be applied to Oxford Nanopore data because it uses a hairpin connector. Ranking the alignments with a probability measure could be more precise and informative as is done in [65], but it could be less time efficient as computing a probability may require more calculations.

Chapter 5

InfoTrim: A Read Trimmer Using Entropy

5.1 Introduction

Raw DNA reads generated by next generation sequencing machines can have diminishing quality on the 5' ends [13, 127] and be contaminated by adapters [30]. These errors can affect the quality of alignment and downstream SNV, indel, and methylation calling [30]. Read trimming is a process where the raw DNA reads are cut to address issues of low quality or adapter contamination.

This study introduces a new read quality trimmer called InfoTrim derived from the maximum information approach of the popular trimmer Trimmomatic [10]. InfoTrim introduces an entropy term, a measure of sequence complexity, to the maximum information model of Trimmomatic. This term was added since sequence complexity can correlate with alignment quality, as low complexity reads are more frequently unmapped or mapped to multiple loca-

tions while high complexity reads are more frequently uniquely mapped [118]. InfoTrim uses Shannon entropy, which measures sequence complexity by counting the frequency of bases in a read. Shannon entropy plays a fundamental role in information theory [131].

Some other read processing tools incorporate sequence complexity, but they do not use Shannon entropy in the same way as InfoTrim does. The trimmer Reaper uses a measure of trinucleotide complexity that is the same as is used in the base masking program DUST [29, 98]. The read mapper Novoalign trims low complexity 5' tails up to 5-9 bp [6]. The bisulfite DNA read mapper BatMeth filters out reads with low Shannon entropy [81]. The UEA Toolkit has a low sequence complexity filter that eliminates reads with at most two bases, but it is for RNA reads [142].

The other terms in the InfoTrim model include the sequence length and the phred score. There is a term for sequence length threshold and for the total sequence length. Everything else being equal, preserving the length of a read is generally preferred, as this uses more of the raw DNA read data. Sequence length can affect indel calling [117]. The phred score of a DNA read gives a per-base measure of the probability that the base was called correctly [34]; thus, bases near the 5' end of the read with low phred score can be trimmed. These terms are the same as the Trimmomatic model. InfoTrim does not support adapter trimming at this time, but there are many trimmers for this including Trimmomatic.

To compare the performance of InfoTrim, data involving bisulfite-treated DNA reads (BS-Seq) was used. Bisulfite treatment of DNA converts unmethylated cytosine to thymine upon PCR amplification while leaving methylated cytosine unchanged. It is a way to study epigenetic methylation, which relates to disease and development [169]. BS-Seq data was used since aligning bisulfite-treated DNA is a challenging task since bisulfite treatment tends to reduce sequence complexity [118] and can introduce major biases [93].

Trimmer	Version	Website
InfoTrim	0.1.0	github.com/JacobPorter/InfoTrim
Trimmomatic	0.36.0	www.usadellab.org/cms/?page=trimmomatic
Cutadapt	1.14	cutadapt.readthedocs.io/en/stable/guide.html
Erne-filter	2.1.1	erne.sourceforge.net/
Reaper	13.100	www.ebi.ac.uk/ stijn/reaper/reaper.html
Sickle	1.33	github.com/najoshi/sickle

Table 5.1: Trimmer versions used in this study and websites.

This study compares InfoTrim with five other trimmers involving four read mappers. The read trimmers were Cutadapt [90], Erne-filter [30], Reaper [29], Sickle [63], and Trimmomatic [10]. The read mappers were BisPin [115], Bismark [70], BWA-Meth [110], and Walt [19]. Tables 5.1 and 5.2 give information on the read trimmers used in this study.

5.2 Implementation

InfoTrim is a multiprocess Python 2.7 program featuring a maximum information score used to cut DNA reads at the 5' end. For a read d, the substring d_l is the read starting at base position 0 and ending at base position l. Starting at the first base, the score $S_{total}(d_l)$ is computed for all l starting at 0 and ending at the total length of the read d. The position l that maximizes the score is chosen and every base after l is discarded from the read. The InfoTrim maximum information model involves four terms.

The first term, $S_{len}(d_l)$, is a logistic function that provides a length threshold m. This causes the trimmer to strongly prefer reads larger than m. The reason for this is that reads that are too small will have too little information to be useful. The default is $m = 25$, since that

Trimmer	Algorithm Type	Language	Features	Settings
InfoTrim	Running Score	Python 2.7	has an entropy term, multi-processing	s:0.1, r:0.4, m:25, t:0
Trimmomatic	Running Score, Windowed	Java	Adapter stripping, multi-threaded	MAXINFO: 25: 0.5
Cutadapt	Running Score	C and Python	Adapter stripping	-q 10
Erne-filter	Running Score	C++	contaminant removal, multi-threaded	Default
Reaper	Quality Score	C	demuxes barcoded sequences, adapter stripping, tri-nucleotide complexity score	-geom no-bc -3pa "" -tabu "" -dust-suffix 90 -qqq-check 50/15 –nozip
Sickle	Windowed	C	Supports gzip files	-t sanger

Table 5.2: Trimmer features, information, and settings used in this study

is the default for Trimmomatic. The equation is the following:

$$S_{len}(d_l) = \frac{1}{1 + e^{m-l}}.$$

The second term, $S_{cov}(d_l)$, causes the trimmer to prefer longer reads to shorter reads to increase the coverage that the read represents. It is simply the length of the read, giving a linear relationship between the length and the score. It is described in the following:

$$S_{cov}(d_l) = l.$$

The phred score, $S_{phred}(d_l)$, is the third term, and it measures base calling correctness. The phred score at base position i in read d is the probability that the base was called correctly as determined by the sequencing platform. The phred number, a positive integer, at base position i is given by Q_i. The probability derived from Q_i is given by $P_{correct}(i) = 10^{\frac{-Q_i}{10}}$. The phred term is given by the following formula:

$$S_{phred}(d_l) = \prod_{i=1}^{l} P_{correct}(i).$$

The rationale for using the product is given in the Trimmomatic paper [10].

InfoTrim adds a fourth term for entropy. The probability $P_l(b)$ is the frequency that base b is found in read d_l. The Shannon entropy is multiplied by one half to give it a scaling between 0 and 1. The formula that InfoTrim uses is the following:

$$S_{ent}(d_l) = -\frac{1}{2} \sum_{b \in \{A,C,T,G,N\}} P_l(b) \log(P_l(b)).$$

The last three terms are combined using the geometric mean so that proportional changes in each term will contribute equally to the final score, and this allows for an intuitive interpretation of the effects of each term. The geometric mean is a balanced mean that does not depend on the scale of the terms. Parameters r (for entropy) and s (for phred) are used to weight the relative contributions of the last three terms. The values r, s, and $1-r-s$ must all add up to one. For example, if $r = 1$, then only the entropy term will be used while the coverage and phred scores will not be used. Thus, the final score is the following:

$$S_{total}(d_l) = S_{len}(d_l) \cdot S_{cov}(d_l)^{\frac{1-r-s}{3}} \cdot S_{phred}(d_l)^{\frac{s}{3}} \cdot S_{ent}(d_l)^{\frac{r}{3}}.$$

The algorithm 7 gives a linear-time algorithm for computing the maximum score and the cut location for a read, and the algorithm 8 gives a quadratic-time algorithm that calls the linear-time algorithm. This algorithm trims both the 5' and 3' ends. The algorithm 7 computes the previously described terms for each position of a read. The position in the read with the maximum score is returned along with the maximum score. To trim the read, all bases after this position are removed. The quadratic-time algorithm calls the linear-time algorithm at every starting position.

5.3 Data Analysis Methods

Real and simulated data was used to assess the accuracy of InfoTrim. The arguments used for each trimmer are given in Table 5.2. Default settings were used when available. Otherwise, reasonable settings were used. The DUST score was used with Reaper to compare its sequence complexity term with InfoTrim. For the read mappers, BisPin 0.1.1, Walt 1.0, Bismark 0.16.3, and BWA-Meth 0.2.0, default settings were used. A single BisPin index with

Algorithm 7

$read \leftarrow$ A short DNA read
$phred \leftarrow$ The phred quality string for the DNA $read$
$m \leftarrow$ The cutoff for the minimum length
$r, s, 1 - r - s \leftarrow$ Parameters to control the tradeoff between length, correctness, and probability. Must add up to one.
procedure InfoTrimLinear($read, phred, m, r, s, 1 - r - s$)
    $maxScore \leftarrow 0$
    $maxPosition \leftarrow -1$
    for $i \leftarrow 0, |read|$ **do**
        $prob \leftarrow \prod_{j=0}^{i} phred[i]$
        $ent \leftarrow$ entropy of $read[0 : i + 1]$
        $score \leftarrow \frac{1}{1+e^{m-(i+1)}}(i+1)^{\frac{r}{3}} prob^{\frac{s}{3}} \frac{ent}{2}^{\frac{1-r-s}{3}}$
        if $score > maxScore$ **then**
            $maxScore \leftarrow score$
            $maxPosition \leftarrow i$
    return $maxScore$, $maxPosition$

Algorithm 8 The InfoTrim Quadratic-Time Algorithm for Finding a Maximum Scoring Substring

$read \leftarrow$ A short DNA read
$phred \leftarrow$ The phred quality string for the DNA $read$
$m \leftarrow$ The cutoff for the minimum length
$t \leftarrow$ This parameter controls how biased the trimmed read should be towards the front.
$r, s, 1 - r - s \leftarrow$ Parameters to control the tradeoff between length, correctness, and probability. Must add up to one.
procedure InfoTrimNonLinear($read, phred, m, t, r, s, 1 - r - s$)
    $maxScore \leftarrow 0$, $begin \leftarrow 0$, $end \leftarrow 0$
    for $i \leftarrow 0, |read|$ **do**
        $s, p \leftarrow$ InfoTrimLinear($read[i :], phred[i :], m, r, s, 1 - r - s$)
        if $s > maxScore$ **then**
            $maxScore \leftarrow s$, $begin \leftarrow i$, $end \leftarrow p$
    return $maxScore, begin, end$

mask "11111111100111111111" was generated. The FASTQ reads files were first trimmed with the read trimmers and then mapped with the read mappers.

5.3.1 Simulated Data

Five hundred thousand simulated 150bp Illumina bisulfite reads were generated with Sherman using the mouse reference genome (GRCm38.p5) [7] and a two percent error rate, a twenty percent CG context conversion rate, and a two percent CH conversion rate. These settings were chosen, since they are realistic. The error rate simulates both sequencing error and natural SNV variation. Sherman does not simulate indels and has a flat phred score. Five hundred thousand simulated Ion Torrent 200bp reads were generated with DWGSIM using the human reference genome (GRCh38.p9) [54]. For DWGSIM, the bisulfite treatment was simulated on the reference genome with the CpG rate as 0.215, the CH rate 0.995, and the over-conversion and under-conversion rates of 0.0025. This was done with custom Python scripts provided by Hong Tran. DWGSIM generates paired-end data, but single-end data was desired, so appropriate ends were chosen to simulate single-end data. DWGSIM was used with the following realistic settings: "dwgsim -e 0.012 -E 0.012 -d 250 -s 30 -S 0 -N 1000000 -c 2 -1 200 -2 200 -f TACGTACGTCTGAGCATCGATCGATGTACAGC." This data was used to compare InfoTrim with other trimmers. DWGSIM simulates the overcalling and undercalling of homopolymer runs that are characteristic of Ion Torrent reads [13]. This higher error rate makes Ion Torrent reads more challenging to align.

Another data set of 500k 100bp simulated Illumina human DWGSIM reads were used to explore the effect of varying the r and s parameters for InfoTrim. Data was generated with settings that were the same as for the Ion Torrent reads except that the "-f" parameter was not used, and 100bp reads were generated. Varying the r and s parameters was done with

a step size of 0.1. All the read mappers were used to align and map the data.

Recall, the percentage of the total reads aligned correctly, and precision, the percentage of uniquely mapped reads aligned correctly, was calculated for each alignment SAM file generated by the read mappers. These statistics control for false positives and false negatives. A read was correctly aligned if it was mapped to within 3bp of the true location's starting position. The F1-score, the harmonic mean of precision and recall, was calculated to give a single numeric score for each trimmer and mapper pipeline. A higher F1-score indicates better accuracy performance.

The mapper Walt could not always map the output from Reaper, and some pipelines and data had zero for Reaper results. For this reason, Walt or Reaper were sometimes excluded from the results.

5.3.2 Real Data

Five hundred thousand real whole mouse genome bisulfite 101 bp Illumina HiSeq 2000 reads were downloaded from the SRA trace archive with accession number SRR921759. The data was trimmed with the read quality trimmers and then mapped with BisPin. A strict alignment quality filter of 85 was used since reads passing this filter should have a precision approaching one hundred percent, so the unique alignments are presumptively correct. A filter value of 96 indicates that the read perfectly matches the genome, so a filter value of 85 indicates very high quality alignments.

A second real data set consisting of one million hairpin bisulfite-treated Illumina mouse reads was taken from the data described in [169], and this hairpin data was used to compare pipelines using presumptively correct alignments in the following manner. Hairpin bisulfite data uses paired-end sequencing such that the original DNA strand, untreated by bisulfite,

can be recovered [115]. The original strand generally has more entropy and can align better than bisulfite-treated strands [118]. This recovery procedure was performed using BisPin, which recovered approximately 600k reads, and the resulting original reads were aligned with BFAST [55], Bowtie2 [72], and BWA [78] to create three files of presumptively correct unique alignments. The corresponding one million C-to-T converted bisulfite forward direction reads were then trimmed and aligned with all of the tools used in this study. The uniquely mapped reads from each trimmer and bisulfite mapper pipeline were then compared to the presumptively correct reads of each regular read mapper and averaged.

5.4 Results and Discussion

5.4.1 Simulated Data

Figure 5.1 shows the results of varying the r and s parameters on F1-score for each read mapper. The left-most pane of Figure 5.1 shows the F1-score when no trimming was performed. The highest F1-score and (r, s) arguments for each mapper are the following: Bismark 0.910 (0.3, 0.1), BisPin 0.881 (0.4, 0.1), BWA-Meth 0.880 (0.1, 0.0), and Walt 0.914 (0.1, 0.2). This setting allowed BWA-Meth to beat the untrimmed F1-score by 7.40E-5, a small amount. The other mappers had similar performance. According to Figure 5.1, low to moderate settings for r and low settings for s substantially outperformed other settings. This suggests that the coverage term, $S_{cov}(d_l)$, should have a high weight and is more important than entropy and the phred score; however, trimming with entropy can still usefully improve the accuracy. For this reason, $r = 0.4$ and $s = 0.1$ were chosen as the default settings for InfoTrim. Bismark and BisPin were generally the most consistent with the F1-score falling the least in the right-most panes. The quality of Walt and BWA-Meth fell off more quickly.

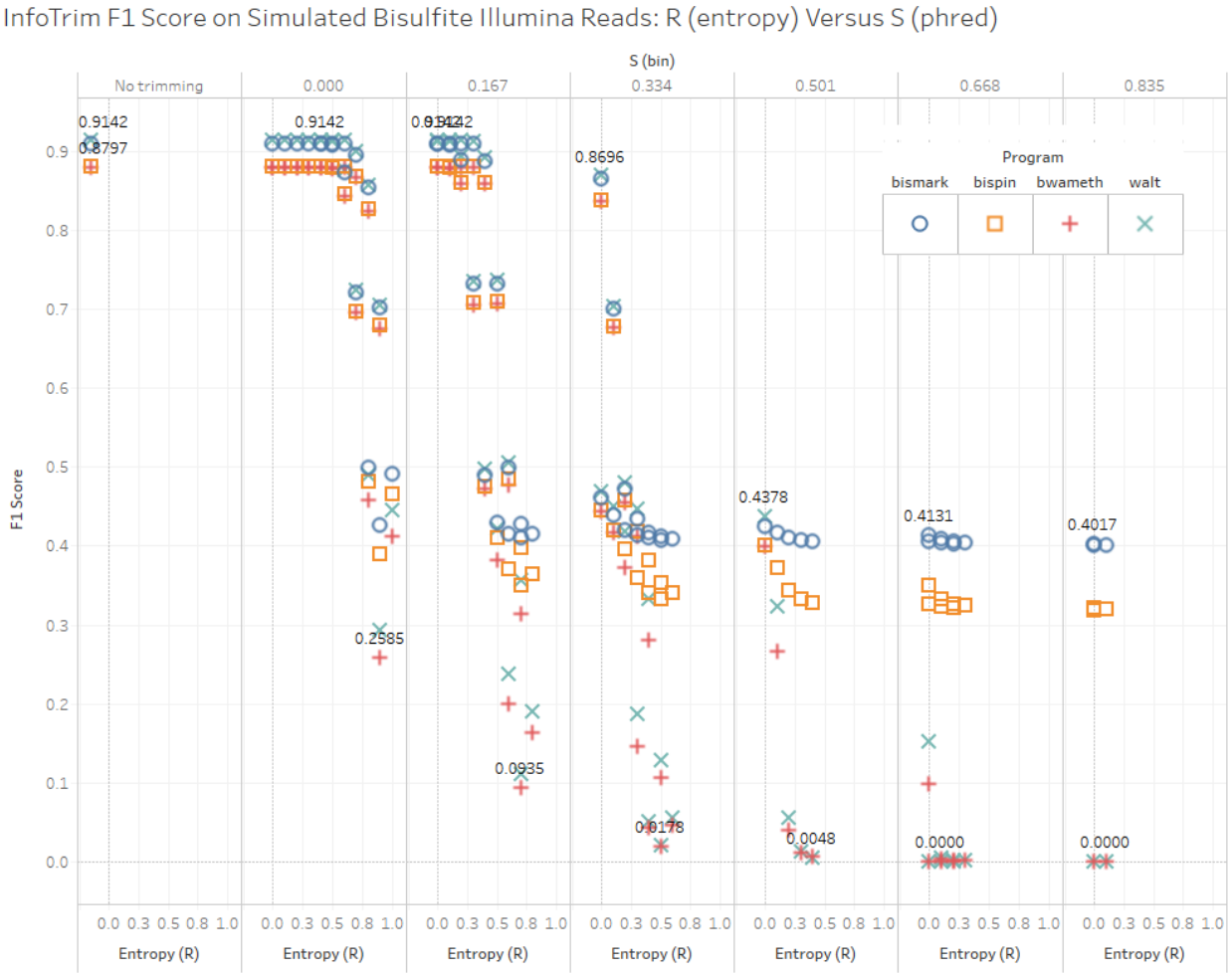


Figure 5.1: InfoTrim simulation results vis-a-vis systematically varying the r and s parameters.

On the Sherman simulated Illumina mouse reads, InfoTrim improves the F1-score better than all the other trimmers except for pipelines involving BWA-Meth as shown in Figure 5.2. With the BWA-Meth pipeline, InfoTrim was second best, beating Reaper, the other trimmer that uses sequence complexity, and beating Trimmomatic, the trimmer that InfoTrim derives from. Pipelines involving InfoTrim do not always have the highest percentage of uniquely mapped reads, but more uniquely mapped reads do not always equate to a higher accuracy as the F1-score shows.

Figure 5.3 shows a plot of precision versus recall, while the numeric label is the F1-score on the DWGSIM simulated human Ion Torrent bisulfite-treated reads. InfoTrim, Cutadapt, and Reaper performed similarily while Trimmomatic was the next best. Erne-filter and Sickle were the worst performing. The read mapper BisPin generally had the highest F1-score for all pipelines with the exception of Erne-filter and Sickle. However, trimming did little to improve the F1-score compared to no trimming.

5.4.2 Real Data

On the strict filter test of real mouse reads, all trimmers improved the alignment score as indicated in Figure 5.4. InfoTrim performed the best on the real data and aligned 1300 more reads than Cutadapt, the closest competitor. InfoTrim aligned 15,025 more reads than without trimming.

The proportion of presumptively correct alignments for the hairpin data by trimmer and mapper is shown in Table 5.3. According to the average proportion of reads aligned correctly across all mappers, InfoTrim was third best overall, behind Cutadapt and Erne-filter; however, there is very little difference in this score except for Reaper, which did poorly since BWA-Meth and Walt did not map this data well. Pipelines invovling BisPin were

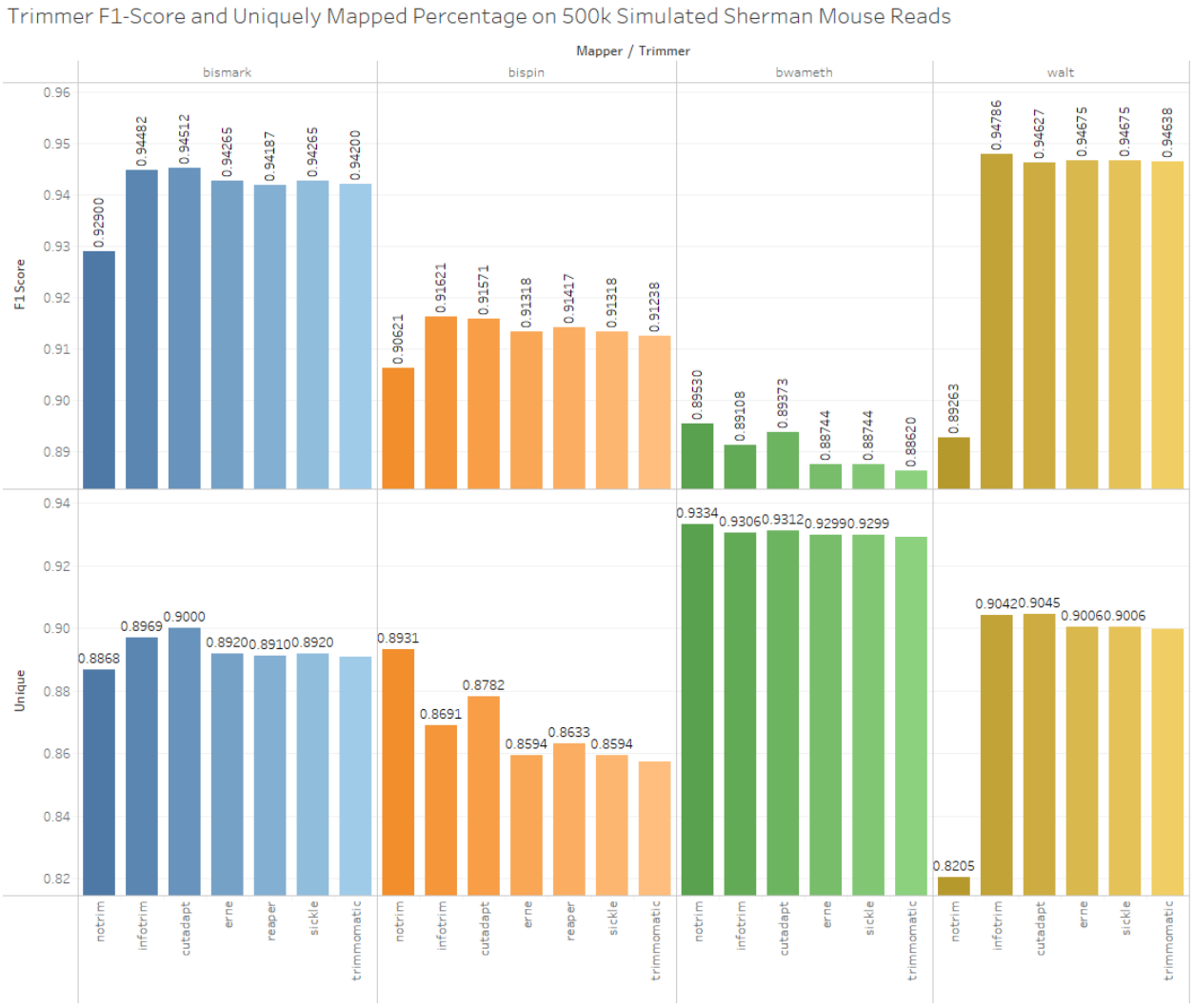


Figure 5.2: Sherman mouse Illumina simulation on 500k trimmed reads with F1-score and uniquely mapped percentage

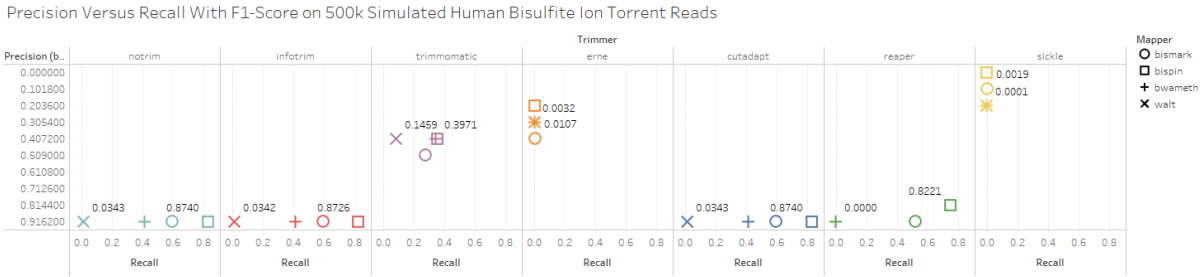


Figure 5.3: Precision versus recall and numeric F1-score on DWGSIM human Ion Torrent simulated bisulfite trimmed reads

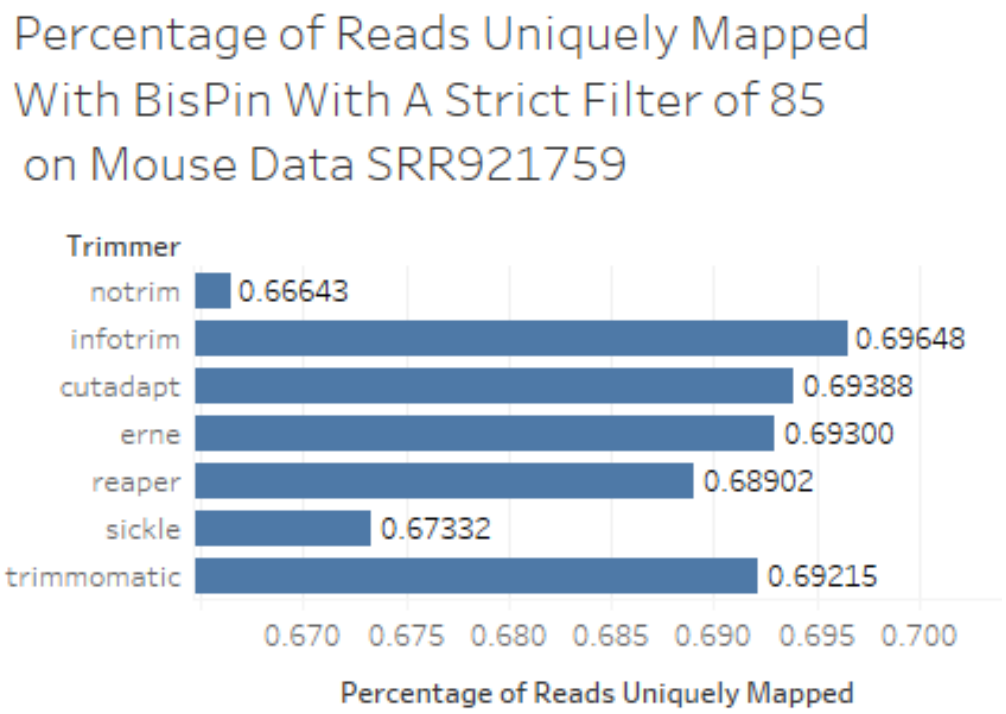


Figure 5.4: Percentage of 500k mouse real trimmed reads aligned presumptively correctly with a strict BisPin filter of 85.

Proportion of Correct Hairpin Hits Across Mappers and Trimmers

	Trimmer						
Mapper	notrim	cutadapt	erne	infotrim	sickle	trimmomatic	reaper
bismark	0.5707	0.5706	0.5678	0.5672	0.5668	0.5568	0.5668
bispin	0.6161	0.6160	0.6127	0.6124	0.6116	0.6009	0.6135
bwameth	0.6206	0.6203	0.6165	0.6167	0.6157	0.6039	0.0001
walt	0.6013	0.6014	0.5990	0.5980	0.5973	0.5872	0.3008
Average	0.6022	0.6021	0.5990	0.5986	0.5979	0.5872	0.3703

Table 5.3: Hairpin data accuracy analysis. The cell for a trimmer and mapper represent an average of proportions of correct alignments using presumptively correct alignments from BFAST, BWA, and Bowtie2 on recovered reads. The average performance across all bisulfite mappers for a given trimmer is shown in the row "Average."

Data set	Multireads + Uniquely Mapped %	Uniquely Mapped %
no trimming	61.7	39.6
linear trimmer with $r = 0.5$ and $s = 0.25$	67.1	42.2

Table 5.4: Read mapping performance with Bismark and InfoTrim trimming on one million 101bp bisulfite-treated hairpin reads.

generally good, but BWA-Meth often did better except for the pipeline with the trimmer Reaper. Pipelines with Bismark were the worst. This shows that the results for InfoTrim are reasonable and consistent with other trimmers.

A second simple test on the mouse hairpin data was conducted consisting of single-end C-to-T converted reads in the forward direction. The reads were trimmed with InfoTrim with settings $r = 0.5$ and $s = 0.25$ using the linear time algorithm. These settings increase the importance of entropy compared to the default InfoTrim settings. Results are summarized in Table 5.4. Trimming improved the uniquely mapped percent by 2.6 percent.

Number of processes	1	2	3
linear algorithm	5 min	3.25 min	2.17 min
quadratic algorithm	245.33 min	148.68 min	88.12 min

Table 5.5: InfoTrim timing in minutes on one million hairpin 101bp reads.

5.4.3 Timing

A run time analysis was performed by trimming 5 million real human bisulfite reads consisting of 2 million reads from SRR1104850, 2 million reads from SRR1799718, and one million reads from SRR3106764. All the read trimmers except for InfoTrim took approximately one minute to complete using a single thread. Trimmomatic was the fastest with 30 seconds and Reaper was the slowest with 81 seconds. InfoTrim was considerably slower on this data, and it took 4,369 seconds (approximately 73 minutes) with a single process and 926 seconds (approximately 15 minutes) with six processes, an improvement of 4.7 times. InfoTrim is probably slow, since it is implemented in Python. Timing was calculated with the Linux `time` command, and the read trimmers were run on the Virginia Tech CS Department's bioinformatics machine mnemosyne2 consisting of 16 processing cores of Intel(R) Xeon(R) CPU E5620 @ 2.40GHz with 132 GB of memory.

Table 5.5 gives a timing performance summary on one million hairpin 101bp reads for InfoTrim. It shows that the linear-time algorithm is substantially faster than the quadratic-time algorithm, and using multiple processes improves run-time. This timing test was run with an Intel 5820K six core CPU at 3.3 Ghz per core with 1, 2, and 3 processes. InfoTrim was run with $r = 0.5$ and $s = 0.25$. Times are in minutes and were measured with the Linux `time` command.

5.5 Conclusion

Using sequence complexity to improve read trimming is a promising strategy, as sequence complexity can relate to alignment accuracy performance. InfoTrim is a new Python read trimmer that performs better than five other trimmers on simulated Sherman bisulfite data and real mouse data involving alignment pipelines of four read mappers. InfoTrim did not substantially reduce accuracy on simulated human Ion Torrent bisulfite reads unlike some read trimmers. Unfortunately, InfoTrim is slow, but the techniques could be incorporated into the Java based Trimmomatic, the fastest read trimmer. The only other read trimmer studied that used sequence complexity was Reaper, and InfoTrim outperformed this trimmer. Given the presented results, perhaps sequence complexity should be incorporated into more read trimmers.

Chapter 6

Concluding Remarks

6.1 Conclusions

I examined the Read Mapping and Alignment problem by studying indel calling and by studying read mapping failure with hairpin data. Entropy correlates with read alignment categories. I created the programs BisPin, BFAST-Gap, and InfoTrim to address the research question and improve the cumulative quality function. BisPin with BFAST-Gap is a versatile read mapper program that accurately aligns bisulfite-treated reads from several construction protocols and from both Illumina and Ion Torrent reads on the same default settings as shown from real and simulated data. BFAST-Gap is an enhanced version of BFAST that is backwards compatible with BFAST indexes. BFAST-Gap uses an alignment function customized to address Ion Torrent's error mode where there is overcalling or undercalling of homopolymer runs. InfoTrim is a new read trimmer that uses Shannon entropy to trim reads for improved read quality. Read trimming often improves mapping outcomes.

6.2 Future Directions

The run-time performance of BFAST-Gap could be substantially improved. For improving run-time, BFAST-Gap could be modified to include a highly optimized SIMD implementation of the Smith-Waterman algorithm as in Bowtie2, and it could dispense with intermediate files, which are slow to create. Rather than using the slower Smith-Waterman algorithm, a fast and reasonably accurate heuristic could be developed and used instead. Some of the BFAST command-line parameters could be adjusted to discard reads with too many matches and to look for matching locations at fewer points on the read. This could improve run-time but may sacrifice some accuracy. The source code could be modified to selectively choose which alignments to perform in an intelligent way. Perhaps there are code optimizations such as function inlining that could be done on the BFAST-Gap code. The use of distributed computing could be enhanced and automated with a distributed computing framework. Alternatively, a fast read mapping program could replace BFAST for doing read mapping. A good candidate could be SNAP, which is 3-20 times faster than BWA and Bowtie2 [168].

A more thorough formalization and theorem proving of the Read Mapping and Alignment problem and related problems could be undertaken. For example, problems in probabilistic inference over discrete random variables can be related to algebraic geometry, and dynamic programming algorithms, such as the Smith-Waterman algorithm, is computation over a semiring [56, 109]. Bayesian inference, which is used in methylation and variant calling, is related to secant varieties and toric varieties [40, 109].

Machine learning could be used to predict which features of the read or the genome correlate with read alignment quality. Perhaps this will lead to insights into improving read mapping software and read quality trimmers. However, improving the quality of read quality trimming using similar techniques, algorithms, and features as existing software is unlikely to improve

read quality trimming as examined trimmers, including InfoTrim, performed very similarly.

Combining read mapper alignment results in some way as in an ensemble method could improve confidence in the read alignment quality. Perhaps machine learning or a probabilistic inference model could be used. Although using multiple read mappers to map reads will be substantially more time consuming, generating more data may be more expensive than the costs associated with the ensemble method. Using machine learning to predict mapping categories from read features could lead to insights into how to improve read mapping, read trimming, or variant calling since some predictive features could be incorporated into such software somehow.

Investigating how to map other protocols and sequencing technologies could be done. My research did not study Oxford Nanopore reads, for example, and these reads can be long and may require different approaches such as combining read mapping and genome assembly. Oxford Nanopore is potentially very useful for studying cytosine epigenetics since it can distinguish between covalent modifications of cytosine directly. Downstream analysis such as variant calling and variant function detection in relation to epigenetic cytosine modifications could be researched.

Bibliography

[1] Cornelis A. Albers, Gerton Lunter, Daniel G. MacArthur, Gilean McVean, Willem H. Ouwehand, and Richard Durbin. Dindel: Accurate indel calls from short-read data. *Genome Research*, 21(6):961–973, 2011.

[2] C. David Allis, Thomas Jenuwein, Danny Reinberg, and Marie-Laure Caparros. *Epigenetics*. Cold Spring Harbor Laboratory Press Cold Spring Harbor, NY, 2007.

[3] Bernhard Balkenhol and Stefan Kurtz. Universal data compression based on the Burrows-Wheeler transformation: Theory and practice. *IEEE Transactions on Computers*, 49(10):1043–1053, 2000.

[4] Stephen B. Baylln, James G. Herman, Jeremy R. Graff, Paula M. Vertino, and Jean-Pierre Issa. Alterations in DNA methylation: A fundamental aspect of neoplasia. *Advances in Cancer Research*, 72:141–196, 1997.

[5] Novocraft Technologies Sdn Bhd. Benchmarking ION Torrent PGM Aligners, 2017. URL http://www.novocraft.com/documentation/other-sequencing-platforms/benchmarking-ion-torrent-pgm-aligners/.

[6] Novocraft Technologies Sdn Bhd. Novoalign and NovoalignCS Reference Manual, 2017. URL http://www.novocraft.com/userfiles/file/Novocraft.pdf.

[7] Babraham Bioinformatics and Felix Krueger. Sherman - bisulfite-treated read FastQ simulator, 2011. URL https://www.bioinformatics.babraham.ac.uk/projects/sherman/.

[8] Adrian Bird. DNA methylation patterns and epigenetic memory. *Genes and Development*, 16(1):6–21, 2002.

[9] John A. Birdsell. Integrating genomics, bioinformatics, and classical genetics to study the effects of recombination on genome evolution. *Molecular Biology and Evolution*, 19(7):1181–1197, 2002.

[10] Anthony M. Bolger, Marc Lohse, and Bjoern Usadel. Trimmomatic: A flexible trimmer for Illumina sequence data. *Bioinformatics*, 30(15):2114–2120, 2014.

[11] Michael J. Booth, Miguel R. Branco, Gabriella Ficz, David Oxley, Felix Krueger, Wolf Reik, and Shankar Balasubramanian. Quantitative sequencing of 5-methylcytosine and 5-hydroxymethylcytosine at single-base resolution. *Science*, 336(6083):934–937, 2012.

[12] Michael J. Booth, Giovanni Marsico, Martin Bachman, Dario Beraldi, and Shankar Balasubramanian. Quantitative sequencing of 5-formylcytosine in DNA at single-base resolution. *Nature Chemistry*, 6(5):435–440, 2014.

[13] Lauren M. Bragg, Glenn Stone, Margaret K. Butler, Philip Hugenholtz, and Gene W. Tyson. Shining a light on dark sequencing: Characterising errors in Ion Torrent PGM data. *PLOS Computational Biology*, 9(4):e1003031, 2013.

[14] Michael Burrows and David J. Wheeler. A block-sorting lossless data compression algorithm. Technical report, DEC Systems Research Center, 1994.

[15] Scitable by Nature Education. SNP, 2018. URL https://www.nature.com/scitable/definition/single-nucleotide-polymorphism-snp-295.

[16] Pauline A. Callinan and Andrew P. Feinberg. The emerging science of epigenomics. *Human Molecular Genetics*, 15(suppl_1):R95–R101, 2006.

[17] Richard W. Carthew and Erik J. Sontheimer. Origins and mechanisms of miRNAs and siRNAs. *Cell*, 136(4):642–655, 2009.

[18] Aniruddha Chatterjee, Peter A. Stockwell, Euan J. Rodger, and Ian M. Morison. Comparison of alignment software for genome-wide bisulphite sequence data. *Nucleic Acids Research*, 40(10):e79, 2012.

[19] Haifeng Chen, Andrew D. Smith, and Ting Chen. WALT: Fast and accurate read mapping for bisulfite sequencing. *Bioinformatics*, 32(22):3507–3509, 2016.

[20] Pao-Yang Chen, Shawn J. Cokus, and Matteo Pellegrini. BS-Seeker: Precise mapping for bisulfite sequencing. *BMC Bioinformatics*, 11(1):203, 2010.

[21] Richard Z. Chen, Ulf Pettersson, Caroline Beard, Laurie Jackson-Grusby, and Rudolf Jaenisch. DNA hypomethylation leads to elevated mutation rates. *Nature*, 395(6697): 89–93, 1998.

[22] Peter J.A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: Freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[23] Shawn J. Cokus, Suhua Feng, Xiaoyu Zhang, Zugen Chen, Barry Merriman, Christian D Haudenschild, Sriharsa Pradhan, Stanley F. Nelson, Matteo Pellegrini, and Steven E. Jacobsen. Shotgun bisulphite sequencing of the Arabidopsis genome reveals DNA methylation patterning. *Nature*, 452(7184):215–219, 2008.

[24] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012.

[25] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, third edition, 2009.

[26] Francis Crick. Central dogma of molecular biology. *Nature*, 227(5258):561, 1970.

[27] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A. Albers, Eric Banks, Mark A. DePristo, Robert E. Handsaker, Gerton Lunter, Gabor T. Marth, Stephen T. Sherry, et al. The variant call format and VCFtools. *Bioinformatics*, 27(15):2156–2158, 2011.

[28] Matei David, Misko Dzamba, Dan Lister, Lucian Ilie, and Michael Brudno. SHRiMP2: Sensitive yet practical short read mapping. *Bioinformatics*, 27(7):1011–1012, 2011.

[29] Matthew P.A. Davis, Stijn van Dongen, Cei Abreu-Goodger, Nenad Bartonicek, and Anton J. Enright. Kraken: A set of tools for quality control and analysis of high-throughput sequence data. *Methods*, 63(1):41–49, 2013.

[30] Cristian Del Fabbro, Simone Scalabrin, Michele Morgante, and Federico M. Giorgi. An extensive evaluation of read trimming effects on Illumina NGS data analysis. *PLOS One*, 8(12):e85024, 2013.

[31] Huy Q. Dinh, Manu Dubin, Fritz J. Sedlazeck, Nicole Lettner, Ortrun Mittelsten Scheid, and Arndt von Haeseler. Advanced methylome analysis after bisulfite deep sequencing: An example in Arabidopsis. *PLOS One*, 7(7):e41528, 2012.

[32] Thomas A. Down, Vardhman K. Rakyan, Daniel J. Turner, Paul Flicek, Heng Li, Eugene Kulesha, Stefan Graef, Nathan Johnson, Javier Herrero, Eleni M. Tomazou,

et al. A Bayesian deconvolution strategy for immunoprecipitation-based DNA methylome analysis. *Nature Biotechnology*, 26(7):779, 2008.

[33] Sean R. Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998.

[34] Brent Ewing and Phil Green. Base-calling of automated sequencer traces using PHRED. ii. Error Probabilities. *Genome Research*, 8(3):186–194, 1998.

[35] Michael Farrar. Striped Smith–Waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, 23(2):156–161, 2006.

[36] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *41st Annual Symposium on Foundations of Computer Science*, pages 390–398. IEEE, 2000.

[37] William Bruce Frakes and Ricardo Baeza-Yates. *Information retrieval: Data structures & algorithms*, volume 331. Prentice Hall Englewood Cliffs, New Jersey, 1992.

[38] Martin C. Frith, Ryota Mori, and Kiyoshi Asai. A mostly traditional approach improves alignment of bisulfite-converted DNA. *Nucleic Acids Research*, 40(13):e100, 2012.

[39] Marianne Frommer, Louise E. McDonald, Douglas S. Millar, Christina M. Collis, Fujiko Watt, Geoffrey W. Grigg, Peter L. Molloy, and Cheryl L. Paul. A genomic sequencing protocol that yields a positive display of 5-methylcytosine residues in individual DNA strands. *Proceedings of the National Academy of Sciences*, 89(5):1827–1831, 1992.

[40] Luis David Garcia, Michael Stillman, and Bernd Sturmfels. Algebraic geometry of Bayesian networks. *Journal of Symbolic Computation*, 39(3-4):331–355, 2005.

[41] Erik Garrison and Gabor Marth. Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907*, page 9, 2012.

[42] Hongcang Gu, Zachary D. Smith, Christoph Bock, Patrick Boyle, Andreas Gnirke, and Alexander Meissner. Preparation of reduced representation bisulfite sequencing libraries for genome-scale DNA methylation profiling. *Nature Protocols*, 6(4):468, 2011.

[43] Weilong Guo, Petko Fiziev, Weihong Yan, Shawn Cokus, Xueguang Sun, Michael Q. Zhang, Pao-Yang Chen, and Matteo Pellegrini. BS-Seeker2: A versatile aligning pipeline for bisulfite sequencing data. *BMC Genomics*, 14(1):774, 2013.

[44] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 1997.

[45] William L. Hamilton, Antoine Claessens, Thomas D. Otto, Mihir Kekre, Rick M. Fairhurst, Julian C. Rayner, and Dominic Kwiatkowski. Extreme mutation bias and high AT content in Plasmodium falciparum. *Nucleic Acids Research*, 45(4):1889–1901, 2017.

[46] Kasper D. Hansen, Benjamin Langmead, and Rafael A. Irizarry. BSmooth: From whole genome bisulfite sequencing reads to differentially methylated regions. *Genome Biology*, 13(10):R83, 2012.

[47] Elena Y. Harris, Nadia Ponts, Karine G. Le Roch, and Stefano Lonardi. BRAT-BW: Efficient and accurate mapping of bisulfite-treated reads. *Bioinformatics*, 28(13):1795–1796, 2012.

[48] Elena Y. Harris, Rachid Ounit, and Stefano Lonardi. BRAT-nova: Fast and accurate mapping of bisulfite-treated reads. *Bioinformatics*, 32(17):2696–2698, 2016.

[49] Steven R. Head, H. Kiyomi Komori, Sarah A. LaMere, Thomas Whisenant, Filip Van Nieuwerburgh, Daniel R. Salomon, and Phillip Ordoukhanian. Library construc-

tion for next-generation sequencing: Overviews and challenges. *Biotechniques*, 56(2): 61, 2014.

[50] James M. Heather and Benjamin Chain. The sequence of sequencers: the history of sequencing DNA. *Genomics*, 107(1):1–8, 2016.

[51] James G. Herman, Jeremy R. Graff, S.B.D.N. Myöhänen, Barry D. Nelkin, and Stephen B. Baylin. Methylation-specific PCR: A novel PCR assay for methylation status of CpG islands. *Proceedings of the National Academy of Sciences*, 93(18):9821–9826, 1996.

[52] Robin Holliday and John E. Pugh. DNA modification mechanisms and gene activity during development. *Science*, 187(4173):226–232, 1975.

[53] Nils Homer. TMAP: The torrent mapping program, 2011. URL https://github.com/iontorrent/TMAP/blob/master/doc/tmap-book.pdf.

[54] Nils Homer. DWGSIM, 2017. URL https://github.com/nh13/DWGSIM.

[55] Nils Homer, Barry Merriman, and Stanley F. Nelson. BFAST: An alignment tool for large scale genome resequencing. *PLOS One*, 4(11):e7767, 2009.

[56] Liang Huang. Advanced dynamic programming in semiring and hypergraph frameworks. *Coling 2008: Advanced Dynamic Programming in Computational Linguistics: Theory, Algorithms and Applications-Tutorial notes*, pages 1–18, 2008.

[57] Xiaoqiu Huang and Webb Miller. A time-efficient, linear-space local similarity algorithm. *Advances in Applied Mathematics*, 12(3):337–357, 1991.

[58] Yun Huang, William A. Pastor, Yinghua Shen, Mamta Tahiliani, David R. Liu, and Anjana Rao. The behaviour of 5-hydroxymethylcytosine in bisulfite sequencing. *PLOS One*, 5(1):e8888, 2010.

[59] Shinsuke Ito, Li Shen, Qing Dai, Susan C. Wu, Leonard B. Collins, James A. Swenberg, Chuan He, and Yi Zhang. TET proteins can convert 5-methylcytosine to 5-formylcytosine and 5-carboxylcytosine. *Science*, 333(6047):1300–1303, 2011.

[60] Miten Jain, Ian T. Fiddes, Karen H. Miga, Hugh E. Olsen, Benedict Paten, and Mark Akeson. Improved data analysis for the MinION nanopore sequencer. *Nature Methods*, 12(4):351–356, 2015.

[61] Thomas Jenuwein and C. David Allis. Translating the histone code. *Science*, 293 (5532):1074–1080, 2001.

[62] Yoon-Seong Jeon, Sang-Cheol Park, Jeongmin Lim, Jongsik Chun, and Bong-Soo Kim. Improved pipeline for reducing erroneous identification by 16S rRNA sequences using the Illumina MiSeq platform. *Journal of Microbiology*, 53(1):60–69, 2015.

[63] N.A. Joshi, J.N. Fass, et al. Sickle: A sliding-window, adaptive, quality-based trimming tool for FastQ files., 2011. URL https://github.com/najoshi/sickle.

[64] Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987.

[65] Peter Kerpedjiev, Jes Frellsen, Stinus Lindgreen, and Anders Krogh. Adaptable probabilistic mapping of short reads using position specific scoring matrices. *BMC Bioinformatics*, 15(1):100, 2014.

[66] Robert J. Klose and Adrian P. Bird. Genomic DNA methylation: The mark and its mediators. *Trends in Biochemical Sciences*, 31(2):89–97, 2006.

[67] Donald E. Knuth, James H. Morris, Jr., and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

[68] Jonas Korlach, Keith P. Bjornson, Bidhan P. Chaudhuri, Ronald L. Cicero, Benjamin A. Flusberg, Jeremy J. Gray, David Holden, Ravi Saxena, Jeffrey Wegener, and Stephen W. Turner. Real-time DNA sequencing from single polymerase molecules. *Methods in Enzymology*, 472:431–455, 2010.

[69] Skirmantas Kriaucionis and Nathaniel Heintz. The nuclear DNA base 5-hydroxymethylcytosine is present in Purkinje neurons and the brain. *Science*, 324 (5929):929–930, 2009.

[70] Felix Krueger and Simon R. Andrews. Bismark: A flexible aligner and methylation caller for Bisulfite-Seq applications. *Bioinformatics*, 27(11):1571–1572, 2011.

[71] Charles D. Laird, Nicole D. Pleasant, Aaron D. Clark, Jessica L. Sneeden, KM Anwarul Hassan, Nathan C. Manley, Jay C. Vary, Todd Morgan, R. Scott Hansen, and Reinhard Stöger. Hairpin-bisulfite PCR: Assessing epigenetic methylation patterns on complementary strands of individual DNA molecules. *Proceedings of the National Academy of Sciences*, 101(1):204–209, 2004.

[72] Ben Langmead and Steven L. Salzberg. Fast gapped-read alignment with Bowtie 2. *Nature Methods*, 9(4):357–359, 2012.

[73] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology*, 10(3):R25, 2009.

[74] Rasko Leinonen, Hideaki Sugawara, Martin Shumway, and International Nucleotide Sequence Database Collaboration. The sequence read archive. *Nucleic Acids Research*, 39(suppl_1):D19–D21, 2010.

[75] Chrysanthia A. Leontiou, Michael D. Hadjidaniel, Petros Mina, Pavlos Antoniou, Marios Ioannides, and Philippos C. Patsalis. Bisulfite conversion of DNA: Performance comparison of different kits and methylation quantitation of epigenetic biomarkers that have the potential to be used in non-invasive prenatal testing. *PLOS One*, 10(8): e0135058, 2015.

[76] Heng Li. Tabix: Fast retrieval of sequence features from generic TAB-delimited files. *Bioinformatics*, 27(5):718–719, 2011.

[77] Heng Li and Richard Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[78] Heng Li and Richard Durbin. Fast and accurate long-read alignment with Burrows–Wheeler transform. *Bioinformatics*, 26(5):589–595, 2010.

[79] Heng Li and Nils Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in Bioinformatics*, 11(5):473–483, 2010.

[80] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[81] Jing-Quan Lim, Chandana Tennakoon, Guoliang Li, Eleanor Wong, Yijun Ruan, Chia-Lin Wei, and Wing-Kin Sung. BatMeth: Improved mapper for bisulfite sequencing reads on DNA methylation. *Genome Biology*, 13(10):R82, 2012.

[82] Ryan Lister, Ronan C. O'Malley, Julian Tonti-Filippini, Brian D. Gregory, Charles C. Berry, A. Harvey Millar, and Joseph R. Ecker. Highly integrated single-base resolution maps of the epigenome in Arabidopsis. *Cell*, 133(3):523–536, 2008.

[83] Ryan Lister, Mattia Pelizzola, Robert H. Dowen, R. David Hawkins, Gary Hon, Julian Tonti-Filippini, Joseph R Nery, Leonard Lee, Zhen Ye, Que-Minh Ngo, et al. Human DNA methylomes at base resolution show widespread epigenomic differences. *Nature*, 462(7271):315–322, 2009.

[84] Lin Liu, Yinhu Li, Siliang Li, Ni Hu, Yimin He, Ray Pong, Danni Lin, Lihua Lu, and Maggie Law. Comparison of next-generation sequencing systems. *BioMed Research International*, 2012:11, 2012.

[85] Yaping Liu, Kimberly D. Siegmund, Peter W. Laird, and Benjamin P. Berman. Bis-SNP: Combined DNA methylation and SNP calling for bisulfite-seq data. *Genome Biology*, 13(7):R61, 2012.

[86] Nicholas J. Loman and Mick Watson. Successful test launch for nanopore sequencing. *Nature Methods*, 12(4):303, 2015.

[87] Nicholas J. Loman, Raju V. Misra, Timothy J. Dallman, Chrystala Constantinidou, Saheer E. Gharbia, John Wain, and Mark J. Pallen. Performance comparison of benchtop high-throughput sequencing platforms. *Nature Biotechnology*, 30(5):434–439, 2012.

[88] Mellissa R.W. Mann and Marisa S. Bartolomei. Epigenetic reprogramming in the mammalian embryo: Struggle of the clones. *Genome Biology*, 3(2):1003–1, 2002.

[89] Alex Martelli. *Python in a Nutshell*. O'Reilly Media, Inc., 2006.

[90] Marcel Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet Journal*, 17(1):10, 2011.

[91] Alice McCarthy. Third generation DNA sequencing: Pacific Biosciences' single molecule real time technology. *Chemistry and Biology*, 17(7):675–676, 2010.

[92] Barbara McClintock. The origin and behavior of mutable loci in maize. *Proceedings of the National Academy of Sciences*, 36(6):344–355, 1950.

[93] Gordon R. McInroy, Dario Beraldi, Eun-Ang Raiber, Katarzyna Modrzynska, Pieter van Delft, Oliver Billker, and Shankar Balasubramanian. Enhanced methylation analysis by recovery of unsequenceable fragments. *PLOS One*, 11(3):e0152322, 2016.

[94] Alexander S. Mikheyev and Mandy M.Y. Tin. A first look at the Oxford Nanopore MinION sequencer. *Molecular Ecology Resources*, 14(6):1097–1102, 2014.

[95] Ryan E. Mills, W. Stephen Pittard, Julienne M. Mullaney, Umar Farooq, Todd H. Creasy, Anup A. Mahurkar, David M. Kemeza, Daniel S. Strassler, Chris P. Ponting, Caleb Webber, et al. Natural genetic variation caused by small insertions and deletions in the human genome. *Genome Research*, 21(6):830–9, 2011.

[96] Fumihito Miura, Yusuke Enomoto, Ryo Dairiki, and Takashi Ito. Amplification-free whole-genome bisulfite sequencing by post-bisulfite adaptor tagging. *Nucleic Acids Research*, 40(17):e136–e136, 2012.

[97] Sowmiya Moorthie, Christopher J. Mattocks, and Caroline F. Wright. Review of massively parallel DNA sequencing technologies. *The HUGO Journal*, 5(1-4):1–12, 2011.

[98] Aleksandr Morgulis, E. Michael Gertz, Alejandro A. Schäffer, and Richa Agarwala. A fast and symmetric DUST implementation to mask low-complexity DNA sequences. *Journal of Computational Biology*, 13(5):1028–1040, 2006.

[99] Olena Morozova and Marco A. Marra. Applications of next-generation sequencing technologies in functional genomics. *Genomics*, 92(5):255–264, 2008.

[100] Evgeny A. Moskalev, Mikhail G. Zavgorodnij, Svetlana P. Majorova, Ivan A. Vorobjev, Pouria Jandaghi, Irina V. Bure, and Jörg D. Hoheisel. Correction of PCR-bias in quantitative DNA methylation studies by means of cubic polynomial regression. *Nucleic Acids Research*, 39(11):e77–e77, 2011.

[101] Julienne M. Mullaney, Ryan E. Mills, W. Stephen Pittard, and Scott E. Devine. Small insertions and deletions (INDELs) in human genomes. *Human Molecular Genetics*, 19 (R2):R131–R136, 2010.

[102] Petra Neddermann, Paola Gallinari, Teresa Lettieri, Daniel Schmid, Oanh Truong, J. Justin Hsuan, Karin Wiebauer, and Josef Jiricny. Cloning and expression of human G/T mismatch-specific thymine-DNA glycosylase. *Journal of Biological Chemistry*, 271(22):12767–12774, 1996.

[103] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[104] Joseph A. Neuman, Ofer Isakov, and Noam Shomron. Analysis of insertion–deletion from deep-sequencing data: Software evaluation for optimal detection. *Briefings in Bioinformatics*, 14(1):46–55, 2012.

[105] Lan Nguyen and Leslie Burnett. Automation of molecular-based analyses: A primer on massively parallel sequencing. *The Clinical Biochemist Reviews*, 35(3):169, 2014.

[106] Christian Otto, Peter F. Stadler, and Steve Hoffmann. Lacking alignments? The next-generation sequencing mapper segemehl revisited. *Bioinformatics*, 30(13):1837–1843, 2014.

[107] Stephan Pabinger, Andreas Dander, Maria Fischer, Rene Snajder, Michael Sperk,

Mirjana Efremova, Birgit Krabichler, Michael R. Speicher, Johannes Zschocke, and Zlatko Trajanoski. A survey of tools for variant analysis of next-generation genome sequencing data. *Briefings in Bioinformatics*, 15(2):256–278, 2014.

[108] Stephan Pabinger, Karina Ernst, Walter Pulverer, Rainer Kallmeyer, Ana M. Valdes, Sarah Metrustry, Denis Katic, Angelo Nuzzo, Albert Kriegner, Klemens Vierlinger, et al. Analysis and visualization tool for targeted amplicon bisulfite sequencing on Ion Torrent sequencers. *PLOS One*, 11(7):e0160227, 2016.

[109] Lior Pachter and Bernd Sturmfels. *Algebraic Statistics for Computational Biology*, volume 13. Cambridge University Press, 2005.

[110] Brent S. Pedersen, Kenneth Eyring, Subhajyoti De, Ivana V. Yang, and David A. Schwartz. Fast and accurate alignment of long bisulfite-seq reads. *arXiv preprint arXiv:1401.1129*, page 14, 2014.

[111] Jeffrey Perkel. Making contact with sequencing's fourth generation. *BioTechniques*, 50(2):93–95, 2011.

[112] Toni Pfaffeneder, Benjamin Hackner, Matthias Truß, Martin Münzel, Markus Müller, Christian A. Deiml, Christian Hagemeier, and Thomas Carell. The discovery of 5-formylcytosine in embryonic stem cell DNA. *Angewandte Chemie*, 123(31):7146–7150, 2011.

[113] Benjamin A. Pierce. *Genetics: A Conceptual Approach*. W. H. Freeman, fifth edition, 2013.

[114] Jacob Porter and Liqing Zhang. InfoTrim: A DNA read quality trimmer using entropy. In *Computational Advances in Bio and Medical Sciences (ICCABS), 2017 IEEE 7th International Conference on*, pages 1–2. IEEE, 2017.

[115] Jacob Porter and Liqing Zhang. BisPin and BFAST-Gap: Mapping bisulfite-treated reads. *bioRxiv*, page 26, 2018. doi: 10.1101/284596. URL https://www.biorxiv.org/content/early/2018/03/19/284596.

[116] Jacob Porter, Ming-an Sun, Hehuang Xie, and Liqing Zhang. Improving bisulfite short-read mapping efficiency with hairpin-bisulfite data. In *2014 IEEE 4th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)*, pages 1–2. IEEE, 2014.

[117] Jacob Porter, Jonathan Berkhahn, and Liqing Zhang. A comparative analysis of read mapping and indel calling pipelines for next-generation sequencing data. In Quoc Nam Tran and Hamid Arabnia, editors, *Emerging Trends in Computational Biology, Bioinformatics, and Systems Biology*, chapter 29, pages 521–535. Elsevier, Massachusetts, 2015.

[118] Jacob Porter, Ming-an Sun, Hehuang Xie, and Liqing Zhang. Investigating bisulfite short-read mapping failure with hairpin bisulfite sequencing data. *BMC Genomics*, 16 (11):S2, 2015.

[119] Nicola Prezza, Cristian Del Fabbro, Francesco Vezzi, Emanuale De Paoli, and Alberto Policriti. ERNE-BS5: Aligning BS-treated sequences by multiple hits on a 5-letters alphabet. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, pages 12–19. ACM, 2012.

[120] Ji Qi, Fangqing Zhao, Anne Buboltz, and Stephan C. Schuster. inGAP: An integrated next-generation genome analysis pipeline. *Bioinformatics*, 26(1):127–129, 2009.

[121] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[122] Arthur C. Rand, Miten Jain, Jordan M. Eizenga, Audrey Musselman-Brown, Hugh E. Olsen, Mark Akeson, and Benedict Paten. Mapping DNA methylation with high-throughput nanopore sequencing. *Nature Methods*, 14(4):411–413, 2017.

[123] Jane Reece, Lisa A. Urry, Noel Meyers, Michael L. Cain, Steven A. Wasserman, Peter V. Minorsky, Robert B. Jackson, and Bernard N. Cooke. *Campbell Biology*. Pearson Higher Education AU, 2011.

[124] Arthur D. Riggs. X inactivation, differentiation, and DNA methylation. *Cytogenetic and Genome Research*, 14(1):9–25, 1975.

[125] Vincenzo E.A. Russo, Robert A. Martienssen, and Arthur D. Riggs. *Epigenetic mechanisms of gene regulation*. Cold Spring Harbor Laboratory Press, 1996.

[126] Frederick Sanger, Steven Nicklen, and Alan R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977.

[127] Melanie Schirmer, Umer Z. Ijaz, Rosalinda D'Amore, Neil Hall, William T. Sloan, and Christopher Quince. Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. *Nucleic Acids Research*, 43(6):e37–e37, 2015.

[128] Melanie Schirmer, Rosalinda D'Amore, Umer Z. Ijaz, Neil Hall, and Christopher Quince. Illumina error profiles: Resolving fine-scale variation in metagenomic sequencing data. *BMC Bioinformatics*, 17(1):125, 2016.

[129] Matthew B. Scholz, Chien-Chi Lo, and Patrick S.G. Chain. Next generation sequencing and bioinformatic bottlenecks: The current state of metagenomic data analysis. *Current Opinion in Biotechnology*, 23(1):9–15, 2012.

[130] Scott Schwartz, W. James Kent, Arian Smit, Zheng Zhang, Robert Baertsch, Ross C. Hardison, David Haussler, and Webb Miller. Human–mouse alignments with BLASTZ. *Genome Research*, 13(1):103–107, 2003.

[131] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication.* University of Illinois Press, 1949.

[132] Jay Shendure and Hanlee Ji. Next-generation DNA sequencing. *Nature Biotechnology*, 26(10):1135–1145, 2008.

[133] Jay A. Shendure, Gregory J. Porreca, George M. Church, Andrew F. Gardner, Cynthia L. Hendrickson, Jan Kieleczawa, and Barton E. Slatko. Overview of DNA sequencing strategies. *Current Protocols in Molecular Biology*, 96(1):1–23, 2011.

[134] Shadi Shokralla, Jennifer L. Spall, Joel F. Gibson, and Mehrdad Hajibabaei. Next-generation sequencing technologies for environmental DNA research. *Molecular Ecology*, 21(8):1794–1805, 2012.

[135] Anish Man Singh Shrestha, Martin C. Frith, and Paul Horton. A bioinformatician's guide to the forefront of suffix array construction algorithms. *Briefings in Bioinformatics*, 15(2):138–154, 2014.

[136] Jared T. Simpson, Rachael E. Workman, P.C. Zuzarte, Matei David, L.J. Dursi, and Winston Timp. Detecting DNA cytosine methylation using nanopore sequencing. *Nature Methods*, 14(4):407–410, 2017.

[137] Nayanah Siva. 1000 genomes project. *Nature Biotechnology*, 26(3):256–256, 2008.

[138] Andrew D. Smith, Wen-Yu Chung, Emily Hodges, Jude Kendall, Greg Hannon, James Hicks, Zhenyu Xuan, and Michael Q. Zhang. Updates to the RMAP short-read mapping software. *Bioinformatics*, 25(21):2841–2842, 2009.

[139] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[140] Moritz Smolka, Philipp Rescheneder, Michael C. Schatz, Arndt von Haeseler, and Fritz J. Sedlazeck. Teaser: Individualized benchmarking and optimization of read mapping results for NGS data. *Genome Biology*, 16(1):235, 2015.

[141] Ning Song, Jie Liu, Shucai An, Tomoya Nishino, Yoshitaka Hishikawa, and Takehiko Koji. Immunohistochemical analysis of histone H3 modifications in germ cells during mouse spermatogenesis. *Acta Histochemica et Cytochemica*, 44(4):183–190, 2011.

[142] Matthew B. Stocks, Simon Moxon, Daniel Mapleson, Hugh C. Woolfenden, Irina Mohorianu, Leighton Folkes, Frank Schwach, Tamas Dalmay, and Vincent Moulton. The UEA sRNA workbench: A suite of tools for analysing and visualizing next generation sequencing microRNA and small RNA datasets. *Bioinformatics*, 28(15):2059–2061, 2012.

[143] Marc Sturm, Christopher Schroeder, and Peter Bauer. SeqPurge: Highly-sensitive adapter trimming for paired-end NGS data. *BMC Bioinformatics*, 17(1):208, 2016.

[144] Mamta Tahiliani, Kian Peng Koh, Yinghua Shen, William A. Pastor, Hozefa Bandukwala, Yevgeny Brudno, Suneet Agarwal, Lakshminarayan M. Iyer, David R. Liu, L. Aravind, et al. Conversion of 5-methylcytosine to 5-hydroxymethylcytosine in mammalian DNA by MLL partner TET1. *Science*, 324(5929):930–935, 2009.

[145] Oxford Nanopore Technologies. Oxford Nanopore products, 2018. URL https://nanoporetech.com/products.

[146] Irwin Tessman and Matthew A. Kennedy. DNA polymerase II of Escherichia coli in the bypass of abasic sites in vivo. *Genetics*, 136(2):439–448, 1994.

[147] Hong Tran, Jacob Porter, Ming-an Sun, Hehuang Xie, and Liqing Zhang. Objective and comprehensive evaluation of bisulfite short read mapping tools. *Advances in Bioinformatics*, 2014:11, 2014.

[148] Hong Tran, Xiaowei Wu, and Liqing Zhang. A Bayesian method for assigning ambiguous bisulfite short reads. In *The 7th International Conference on Bioinformatics and Computational Biology*. ISCA, 2015.

[149] Hong Tran, Xiaowei Wu, Saima Tithi, Ming-an Sun, Hehuang Xie, and Liqing Zhang. A Bayesian assignment method for ambiguous bisulfite short reads. *PLOS One*, 11(3): e0151826, 2016.

[150] Mark A. Urich, Joseph R. Nery, Ryan Lister, Robert J. Schmitz, and Joseph R. Ecker. MethylC-seq library preparation for base-resolution whole-genome bisulfite sequencing. *Nature Protocols*, 10(3):475–483, 2015.

[151] Marco Ventura, Carlos Canchaya, Andreas Tauch, Govind Chandra, Gerald F. Fitzgerald, Keith F. Chater, and Douwe van Sinderen. Genomics of Actinobacteria: Tracing the evolutionary history of an ancient phylum. *Microbiology and Molecular Biology Reviews*, 71(3):495–548, 2007.

[152] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.

[153] C.P. Walsh and G.L. Xu. Cytosine methylation and DNA repair. In *DNA Methylation: Basic Mechanisms*, pages 283–315. Springer, 2006.

[154] James D Watson, Francis HC Crick, et al. A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953.

[155] Michael Weber, Jonathan J. Davies, David Wittig, Edward J. Oakeley, Michael Haase, Wan L. Lam, and Dirk Schuebeler. Chromosome-wide and promoter-specific analyses identify sites of differential DNA methylation in normal and transformed human cells. *Nature Genetics*, 37(8):853, 2005.

[156] Zhi Wei, Wei Wang, Pingzhao Hu, Gholson J. Lyon, and Hakon Hakonarson. SNVer: A statistical tool for variant calling in analysis of pooled or individual next-generation sequencing data. *Nucleic Acids Research*, 39(19):e132–e132, 2011.

[157] Tomasz K. Wojdacz and Alexander Dobrovic. Methylation-sensitive high resolution melting (MS-HRM): A new approach for sensitive and high-throughput assessment of methylation. *Nucleic Acids Research*, 35(6):e41, 2007.

[158] Bio-IT World. Six years after acquisition, Roche quietly shutters 454, 2013. URL http://www.bio-itworld.com/2013/10/16/six-years-after-acquisition-roche-quietly-shutters-454.html.

[159] Robert Woroniecki, Anil Bhanudas Gaikwad, and Katalin Susztak. Fetal environment, epigenetics, and pediatric renal disease. *Pediatric Nephrology*, 26(5):705–711, 2011.

[160] Jiaxin Wu, Yanda Li, and Rui Jiang. Integrating multiple genomic data to predict disease-causing nonsynonymous single nucleotide variants in exome sequencing studies. *PLoS genetics*, 10(3):e1004237, 2014.

[161] Mengmeng Wu, Ting Chen, and Rui Jiang. Leveraging multiple genomic data to prioritize disease-causing indels from exome sequencing data. *Scientific Reports*, 7(1): 1804, 2017.

[162] Thomas D. Wu and Serban Nacu. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26(7):873–881, 2010.

[163] Yuanxin Xi and Wei Li. BSMAP: Whole genome bisulfite sequence mapping program. *BMC Bioinformatics*, 10(1):232, 2009.

[164] Yuanxin Xi, Christoph Bock, Fabian Müller, Deqiang Sun, Alexander Meissner, and Wei Li. RRBSMAP: A fast, accurate and user-friendly alignment tool for reduced representation bisulfite sequencing. *Bioinformatics*, 28(3):430–432, 2011.

[165] V.B. Yap and W. Miller. Scoring pairwise genomic sequence alignments. In *Pacific Symposium on Biocomputing*, page 115. World Scientific, 2001.

[166] Byung-Jun Yoon. Hidden Markov models and their applications in biological sequence analysis. *Current Genomics*, 10(6):402–415, 2009.

[167] Miao Yu, Gary C. Hon, Keith E. Szulwach, Chun-Xiao Song, Liang Zhang, Audrey Kim, Xuekun Li, Qing Dai, Yin Shen, Beomseok Park, et al. Base-resolution analysis of 5-hydroxymethylcytosine in the mammalian genome. *Cell*, 149(6):1368–1380, 2012.

[168] M. Zaharia, W.J. Bolosky, K. Curtis, A. Fox, D. Patterson, S. Shenker, I. Stoica, R.M. Karp, and T. Sittler. Faster and more accurate sequence alignment with SNAP. *arXiv preprint arXiv:1111.5572*, pages 1–10, 2011.

[169] Lei Zhao, Ming-an Sun, Zejuan Li, Xue Bai, Miao Yu, Min Wang, Liji Liang, Xiaojian Shao, Stephen Arnovitz, Qianfei Wang, et al. The dynamics of DNA methylation fidelity during mouse embryonic stem cell self-renewal and differentiation. *Genome Research*, 24(8):1296–1307, 2014.

[170] Mengyao Zhao, Wan-Ping Lee, Erik P. Garrison, and Gabor T. Marth. SSW library: An SIMD Smith-Waterman C/C++ library for use in genomic applications. *PLOS One*, 8(12):e82138, 2013.