IMPROVING QUALITY OF HIGH-THROUGHPUT SEQUENCING READS

BY

YUN HEO

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

ABSTRACT

Rapid advances in high-throughput sequencing (HTS) technologies have led to an exponential increase in the amount of sequencing data. HTS sequencing reads, however, contain far more errors than does data collected through traditional sequencing methods. Errors in HTS reads degrade the quality of downstream analyses. Correcting errors has been shown to improve the quality of these analyses.

Correcting errors in sequencing data is a time-consuming and memory-intensive process. Even though many methods for correcting errors in HTS data have been developed, no one could correct errors with high accuracy while using a small amount of memory and in a short time. Another problem in using error correction methods is that no standard or comprehensive method is yet available to evaluate the accuracy and effectiveness of these error correction methods.

To alleviate these limitations and analyze error correction outputs, this dissertation presents three novel methods. The first one, known as BLESS (Bloom-filter-based error correction solution for high-throughput sequencing reads), is a new error correction method that uses a Bloom filter as the main data structure. Compared to previous methods, it allows for the correction of errors with the highest accuracy at an average of 40 $\times$ memory usage reduction. BLESS is parallelized using hybrid OpenMP and MPI programming, which makes BLESS one of the fastest error correction tools. The second method, known as SPECTACLE (Software Package for Error Correction Tool Assessment on Nucleic Acid Sequences), supplies a standard way to evaluate error correction methods. SPECTACLE is the comprehensive method that can (1) do a quantitative analysis on both DNA and RNA corrected reads from any sequencing platforms and (2) handle diploid genomes and differentiate heterozygous alleles from sequencing errors.

Lastly, this research analyzes the effect of sequencing errors on variant calling, which is one of the most important clinical applications for HTS data. For this, the environments for tracing the effect of sequencing errors on germline and somatic variant calling was developed. Using the environment, this research studies how sequencing errors degrade the results of variant calling and how the results can be improved. Based on the new findings, ROOFTOP (RemOve nOrmal reads From TumOr samPles) that can improve the accuracy of somatic variant calling by removing normal cells in tumor samples.

A series of studies on sequencing errors in this dissertation would be helpful to understand how sequencing errors degrade downstream analysis outputs and how the quality of sequencing data could be improved by removing errors in the data.

To my parents, for their unwavering love and support

ACKNOWLEDGMENTS

While this dissertation bears the name of a single student, the process that led to its completion was carried out continually in combination with the dedicated work of other people. I would like to express my deepest gratitude to my advisor, Prof. Deming Chen, for the excellent guidance, care, and patience that he provided me over the past five years. I would also like to thank Prof. Wen-mei Hwu and Prof. Jian Ma for their advice and insight on high performance computing and computational biology. I would like to acknowledge Prof. Wong, of my dissertation committee, for generously giving his expertise and time. I would never have been able to finish my dissertation without their guidance.

I am grateful to Jaecheol Son and Youngmin Shin at Samsung Electronics for giving me the chance to pursue a PhD and for all the support over the years.

I am grateful, too, to all of the members of ESCAD group and to Anand Ramachandran in particular for his help as a cooperator, a consultant, and an auditor. I must acknowledge Xiao-Long Wu for his help when I started to study computational biology.

Finally, and most importantly, a special word of thanks also goes to my parents and sister for all their love and encouragement. I dedicate this dissertation to them.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

CCD	Charge-Coupled Device
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DBG	de Bruijn Graph
dbSNP	Single Nucleotide Polymorphism Database
DGV	Database of Genomic Variants
DNA	Deoxyribonucleic Acid
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
HMM	Hidden Markov Model
HTS	High-Throughput Sequencing
ICGC	International Cancer Genome Consortium
MPI	Message Passing Interface
MSA	Multiple Sequence Alignment
NGS	Next-Generation Sequencing
OpenMP	Open Multi-Processing
PTP	PicoTiter Plate
RNA	Ribonucleic Acid
SBS	Sequencing by Synthesis

SMRT	Single Molecule Real Time
SNP	Single Nucleotide Polymorphism
SNV	Single Nucleotide Variant
SV	Structural Variation
TCGA	The Cancer Genome Atlas
TEF	Target Error Format
TN	True Negative
TP	True Positive
TGS	Third-Generation Sequencing
ZMW	Zero-Mode Waveguides

CHAPTER 1

INTRODUCTION

All living cells keep their hereditary information in the form of double-stranded molecules of deoxyribonucleic acid (DNA). Genomic information encoded in the DNA sequences define the species and individuals, which makes the DNA sequence fundamental to research on the functions of cells.

DNA sequencing is the process that determines the order of nucleotides, an essential step in retrieving the information encoded. The recent advent of next-generation sequencing (NGS) has revolutionized the study of genetics and has also provided valuable resources for other scientific disciplines. As NGS has become more widely accessible, its use has extended beyond basic research into broader clinical contexts.

NGS sequencing technologies, however, have shortcomings. In addition to short read length, a main challenge in analyzing NGS data is its higher error rate than traditional sequencing technology [1, 2]. The most straightforward approach to prevent sequencing errors from degrading the output quality of downstream analyses is to increase sequencing read coverage [3]. However, increasing read coverage cannot be a universal solution because (1) high read coverage cannot solve the issues that arise from the batch effect that is the statistical bias observed in samples that go through the same sample preparation and sequencing processes [4] and (2) it costs more to generate high coverage reads.

Many error correction methods have been developed to alleviate the degradation of downstream analysis outputs. These methods can be divided into four major categories [5]: (1) k-mer spectrum based [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16], (2) suffix tree/array based [17, 18, 19, 20] (3) multiple sequence alignment (MSA) based [21, 22], and (4) hidden Markov model (HMM) based [23, 24]. None of them, however, has successfully corrected errors in HTS reads from large genomes without consuming large amounts of memory unavailable to most researchers. Previous evaluations have shown that some error correction tools require over 128 GB of memory to correct errors in genomes with 120 Mbp, and that others need tens of GBs of memory [5]. For a human genome, previous approaches would need hundreds of GBs of memory.

An additional problem in applying error correction methods to HTS data is that there is no standard way to compare error correction tools or to quantitatively analyze their outputs, which is needed to choose a method suitable for the user's purpose. Such scarcity is mainly due to the effort involved in discerning how many errors are corrected and how many are newly generated in the error correction process. Errors in HTS reads can be categorized into substitutions (bases in reads are different from the bases in original genome sequences), insertions (bases that do not exist in original genome sequences are added to reads), and deletions (bases in an original genome are not shown in corresponding reads). While checking whether substitution errors have been corrected is straightforward, it is not so simple to evaluate how exactly errors are corrected when insertions and deletions also exist. The evaluation becomes more complex when corrected reads have different lengths compared to precorrection reads. Many error correction tools produce reads of shorter length because they trim both ends of the reads to remove the errors that they cannot correct.

Motivated by the two aforementioned problems, this dissertation presents three methods. The first is a new Bloom filter-based error correction algorithm called BLESS. It belongs to the k-mer spectrum-based method but is designed to remove the limitations of the previous k-mer spectrum based solutions. Our approach has four important new features: (1) It is designed to target high memory efficiency in order for error correction to be run on a commodity computer. The k-mers that exist more than a certain number of times in reads are sorted out and programmed into a Bloom filter. (2) It can handle repeats in genomes better than previous k-mer spectrum-based methods, which leads to higher accuracy because BLESS is able to use longer k-mers compared to previous methods. Longer k-mers resolve repeats better. (3) It can extend reads to correct errors at the end as accurately as in other parts. Sometimes an erroneous k-mer may be identified as error-free because of an irregularly large multiplicity of k-mers. False positives from the Bloom filter can also cause the same problem. BLESS extends the reads to find multiple k-mers that cover the erroneous bases at the end of the reads to improve error correction. (4) It can be parallelized either on a server with a multi-core CPU using OpenMP or on multiple servers using MPI.

The second method presented in this dissertation is known as SPECTACLE, which stands for Software Package for Error Correction Tool Assessment on Nucleic Acid Sequences. SPECTACLE is a new error correction tool evaluation algorithm that can evaluate any error correction tool for NGS and TGS reads and work for both DNA and RNA sequencing data, and differentiate heterozygous alleles from sequencing errors.

The last one is ROOFTOP that is a tool that improves the accuracy of somatic variant calling. This research studies the effect of sequencing errors on variant calling—one of the most important applications from a clinical point of view—using in silico experiments. In this study, a new environment that generates ground truth variants and reads with sequencing errors was developed, and the effect of the errors on different variant calling algorithms have been analyzed using the outputs from the environment. The experiments suggest that, by removing sequencing errors from input reads, a meaningful number of false negatives can be removed from variant calling results. It is also found that correcting sequencing errors in reads from tumor samples and removing reads from normal cells in tumor samples are highly effective ways to increase sensitivity of somatic mutation calling especially when the ratio of tumor cells in the tumor samples is low. Based on the results, new software ROOFTOP is developed to remove reads from normal cells in tumor samples.

These comprehensive studies on sequencing errors could be of help in understanding how sequencing errors happen in each sequencing technology, how the errors could degrade downstream analysis results, and how sequencing data could be improved by removing sequencing errors to get better results with the data.

This dissertation is organized as follows. Chapter 2 compares the different HTS platforms and discusses what types of errors are common and why. Chapters 3, 4, and 5 present BLESS and SPECTACLE, examining their performance through a series of analyses. Finally, discussed in Chapter 6 are the effects of sequencing errors on variant calling and how to improve the analysis results.

CHAPTER 2

HIGH-THROUGHPUT DNA SEQUENCING TECHNOLOGIES

During the past ten years, many different HTS technologies have been introduced. They use different methods to determine the order of nucleotides in DNA sequences from samples, which gives each HTS technology a different error characteristic in the output of the sequencing process.

2.1 Illumina

2.1.1 Sequencing Process

The Illumina technology is currently considered to be the most popular sequencing technology. The sequencing process is shown in Figure 2.1. First, DNA samples are randomly fragmented and adapters are ligated to both ends of the fragments. The single stranded fragments are randomly bound to the surface of the flow cell–an eight-channel sealed glass device (Figure 2.1A). In order to later reach sufficient signal intensity during following sequencing steps, each of the fragments is then replicated using bridge amplification, and multiple identical copies of the original fragments are created in close proximity. A set of fragments made from the same original fragment is called a cluster (Figure 2.1B).

To determine each nucleotide in the fragments, the Illumina technology uses a method called sequencing by synthesis (SBS) in which all four modified nucleotides, sequencing primers, and DNA polymerases are added simultaneously to the flow cell channels, and the primers are hybridized to the fragments. Then, polymerases are used to extend the primers using the modified nucleotides. To each of the four types of nucleotides, fluorescent dye with four different colors is added in order for each type to be unique. The 3'-OH group of nucleotides is chemically blocked such that, after the

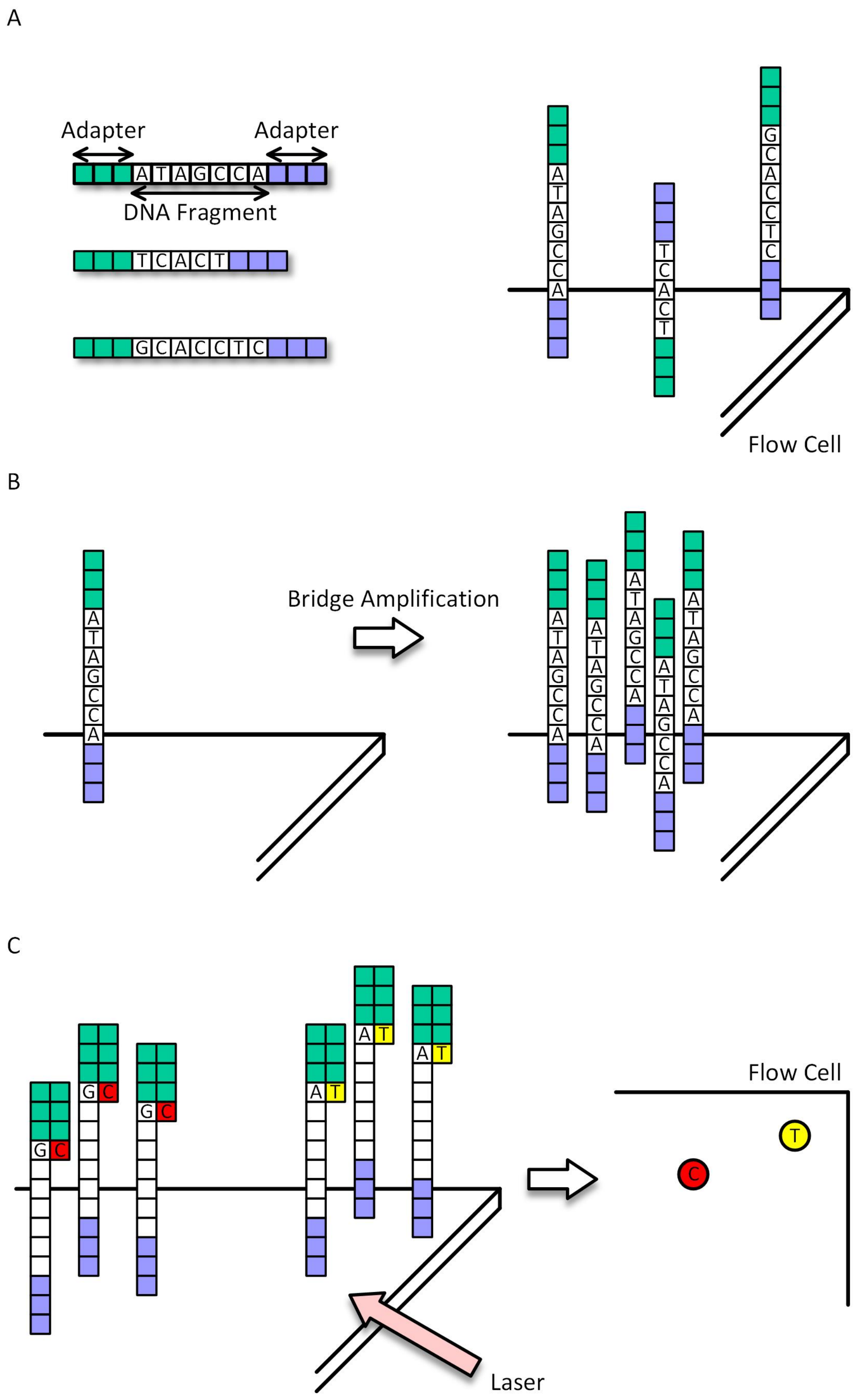


Figure 2.1: The Illumina sequencing technology.

nucleotides are incorporated, further incorporation cannot happen.

In the next step, a laser is used to excite each cluster, and the colors emitted are captured using a charge-coupled device (CCD). The images are then processed to be determined through the process called base calling. After base calling, the 3' blocking group is removed to prepare for the next incorporation. The incorporation and base calling are repeated for a specific number of cycles (Figure 2.1C).

2.1.2 Sequencing Errors

The most popular error type in the Illumina technology is substitutions [25], and there are several sources of sequencing errors. The first one is phasing. Phasing means nucleotides are incorporated at different positions in the fragments of a cluster during the same cycle. The effect of phasing on base calling accumulates in each cycle, making the bases determined in the later cycles have more errors. Second, clusters initiated from more than one DNA fragment can be made, resulting in mixed signals or crosstalk when the sequencer identifies the signals in the base calling step.

2.2 Roche 454

2.2.1 Sequencing Process

The 454 sequencing is a technology introduced in 2004 that uses pyrosequencing. DNA samples are randomly fragmented, and each fragment is attached to a bead whose surfaces carries primers (Figure 2.2A). Then, emulsion PCR is conducted to make each bead contain thousands of copies of the initial fragment (Figure 2.2B).

The beads are then arrayed into picotiter plate (PTP) wells that fix each. In pyrosequencing, each incorporation of a nucleotide releases pyrophosphate, which initiates a series of downstream reactions that ultimately produce light by luciferase. At each cycle during sequencing, a single type of nucleotide is added, and the incorporation of the nucleotide, complementing the next base in the fragments on beads, releases light that is detected by a CCD (Figure 2.2C).

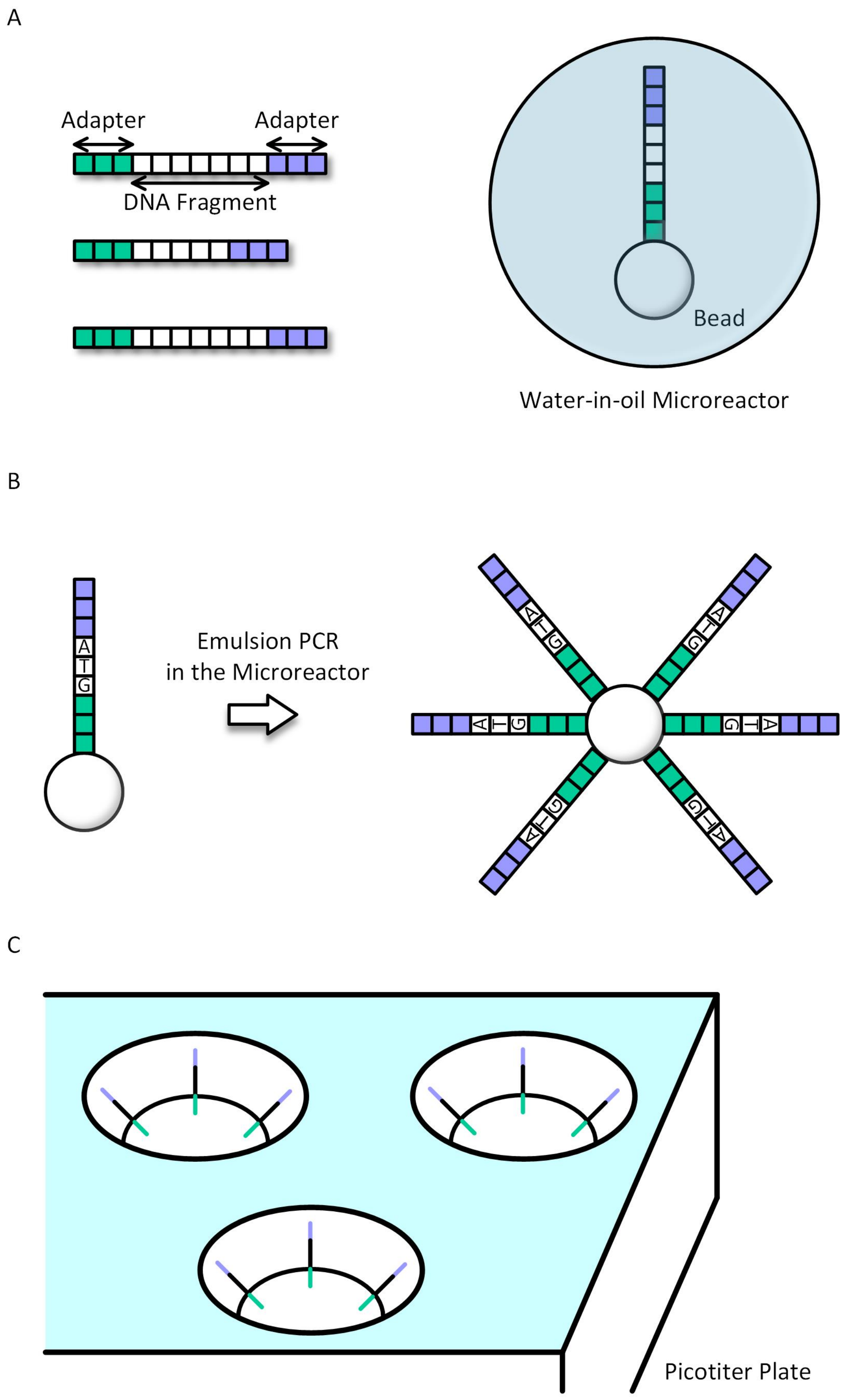


Figure 2.2: The Roche 454 sequencing technology.

2.2.2 Sequencing Errors

In pyrosequencing, multiple incorporation events occur in homopolymers; hence the length of a homopolymer should be determined by the intensity of the light, which is prone to errors [26]. Moreover, strong light signals in one well of the PTP may make insertions in neighboring ones, even though no incorporation happens here [27]. Phasing is also observed as in the Illumina technology, and it worsens the signal-to-noise ratio.

The error rate increases with the position in the sequence. This is because the efficiency DNA polymerases and luciferases drops over the sequencing cycles.

2.3 Life Technologies SOLiD

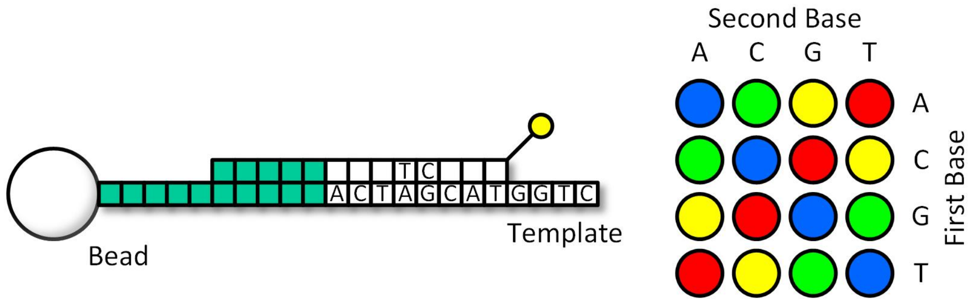
2.3.1 Sequencing Process

The SOLiD platform uses adapter-ligated fragments and an emulsion PCR approach with small beads to amplify the fragments, which is similar to the 454 sequencing technology. However, unlike the 454 technology, the SOLiD technology uses DNA ligase instead of DNA polymerase to identify nucleotides in the fragments.

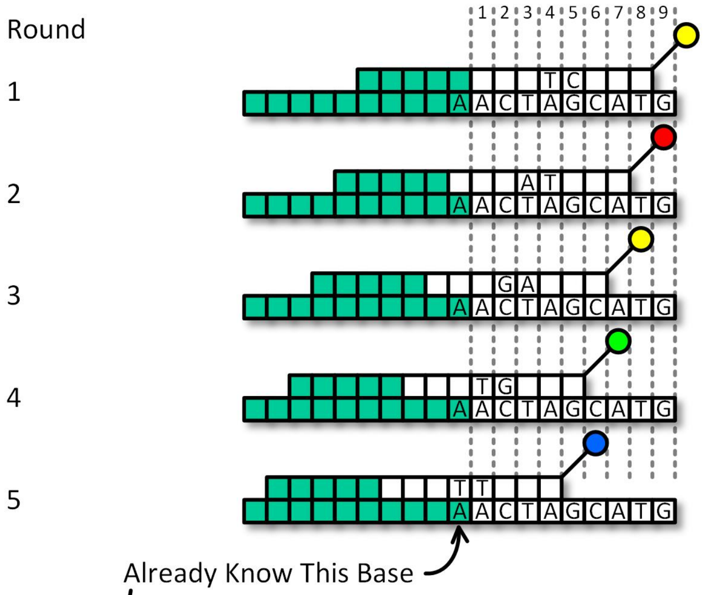
The SOLiD sequencing process consists of multiple sequencing rounds. After primes are annealed, 8-mers with a fluorescent label at the end are sequentially ligated to the DNA fragments, and the color emitted from the label is recorded (Figure 2.3A). Then, the end of the 8-mers is chemically cleaved to allow for the next ligation cycle.

The output of the SOLiD platforms is in a so-called color space which is the encoded form of the nucleotide sequence where four colors are used to represent 16 combinations of two bases. After a certain number of ligation cycles, the complementary strand is removed and a new sequencing round is started using a primer annealed one base further upstream. This means that the positions assessed by the 8-mers change in each round, and the sequencing stops once every base has been probed twice (Figure 2.3B). The color space data can then be decoded given prior knowledge of the leading base, usually the last base of the adapter (Figure 2.3C).

A



B



C

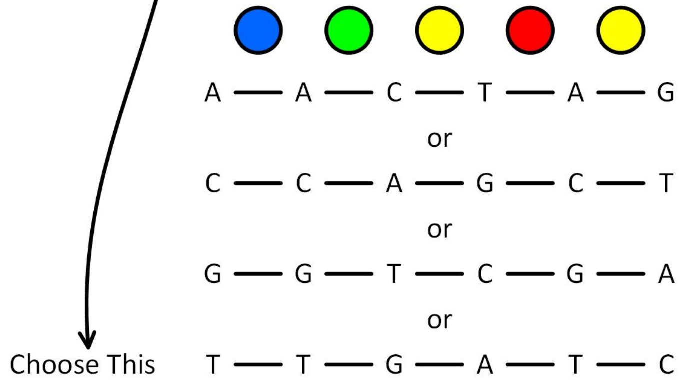


Figure 2.3: The Life Technologies SOLiD sequencing technology.

2.3.2 Sequencing Errors

Types of sequence errors in the SOLiD platform are diverse [27]. Hybridization is a stochastic process; this reduces the number of molecules participating in subsequent ligation reactions and, therefore, substantial signal decline. Consequently, the error rate increases as the ligation cycles progresses. Incomplete cleavage of the dyes is another source of phasing, and it can make noise in the identification process of the next ligation cycle.

2.4 Life Technologies Ion Torrent

2.4.1 Sequencing Process

The Ion Torrent sequencing is relatively new and the first sequencer to use technology introduced in 2010. An overview is shown in Figure 2.4. It uses a chip that has a a high-density array of wells on its surface, and each has a bead with multiple identical fragments. The chip is flooded with one type of nucleotide at each cycle. When a base is incorporated with a fragment in the bead, a hydrogen ion is released, which changes the pH of the solution. The chip has Ion sensors for each well and used to detect how many nucleotides are incorporated.

2.4.2 Sequencing Errors

The Ion Torrent should detect the number of nucleotides in a homopolymer by precisely detecting the pH change; it frequently causes indel errors like the 454 technology.

2.5 Pacific Biotechnology SMRT

2.5.1 Sequencing Process

The first single molecule real-time (SMRT) technology-based sequencer was commercially released in 2011. The name implies that (1) no amplification is

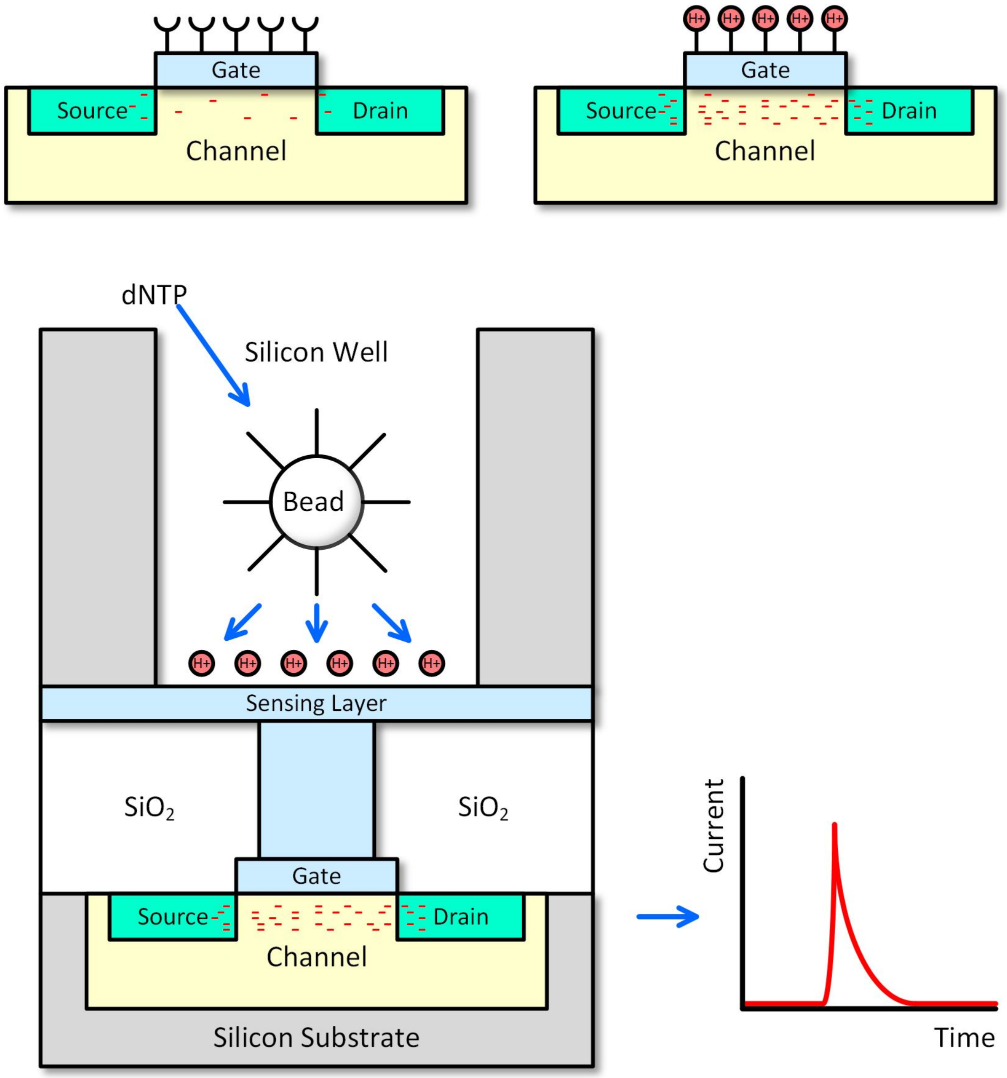
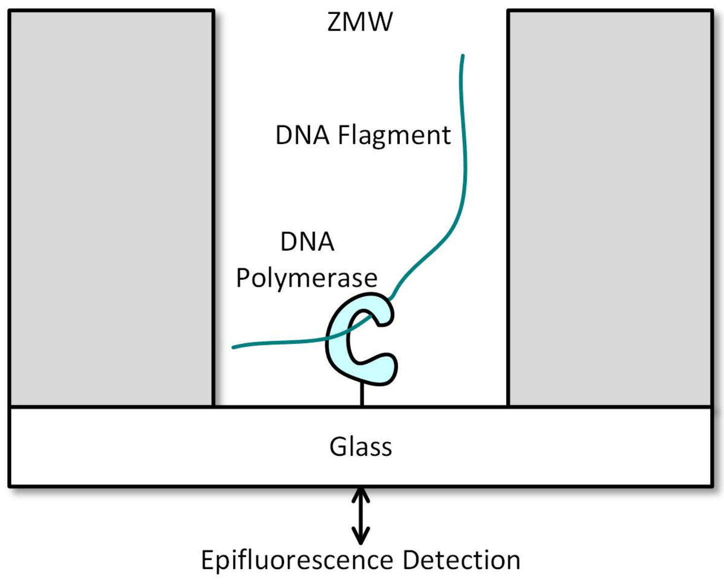


Figure 2.4: The Ion Torrent sequencing technology.

A



B

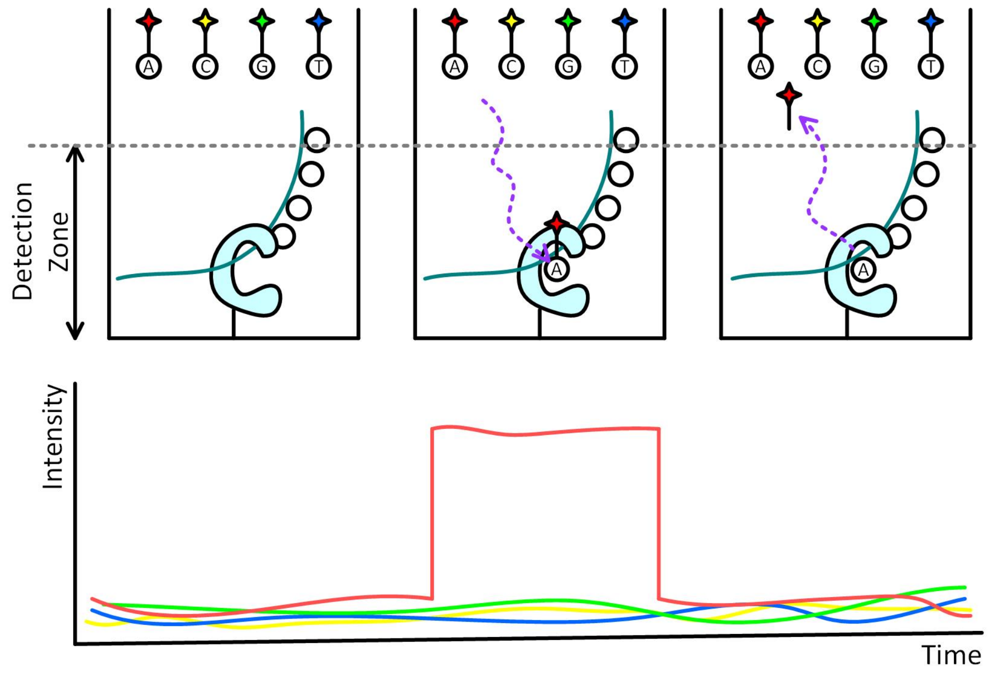


Figure 2.5: The Pacific Bioscience SMRT sequencing technology.

needed, and (2) it observes enzymatic reaction in real time. It is considered a third-generation sequencing (TGS) technology because it does not require any amplification before sequencing.

A SMRT cell has many zero-mode waveguides (ZMWs) as illustrated in Figure 2.5A. Since the ZMW is very small, the wavelength of the light excited from the bottom of the ZMW cannot pass through. Consequently, the light can just penetrate the lower 20-30 nm of the ZMW; the tiny space works as detection volume.

Each ZMW houses a molecule of a single-stranded DNA template and a DNA polymerase at the bottom. Nucleotides attached with four corresponding fluorescent dye molecules are then introduced, as the DNA polymerase performs DNA synthesis naturally. While a nucleotide is held in the detection volume by the incorporation process, light is produced and is recorded in a movie format (Figure 2.5B).

2.5.2 Sequencing Errors

The causes of sequencing errors in the SMRT technology are (1) two short intervals between two incorporation events, and (2) binding and release of nucleotides in the active site before incorporation [25]. The most common error types are insertions and deletions, and they are uniformly distributed along reads [28].

2.6 Oxford Nanopore

2.6.1 Sequencing Process

Oxford Nanopore is another TGS technology; the first commercial device was introduced in 2012. A nanopore is a nanoscale hole made up of protein or synthetic material such as silicon nitride or graphene. An ionic current passes through a nanopore by setting a voltage across this membrane as shown in Figure 2.6. If a single-stranded DNA sequence passes through the pore, a characteristic disruption appears in the ionic current. Measuring the current makes it possible to identify the nucleotide in question.

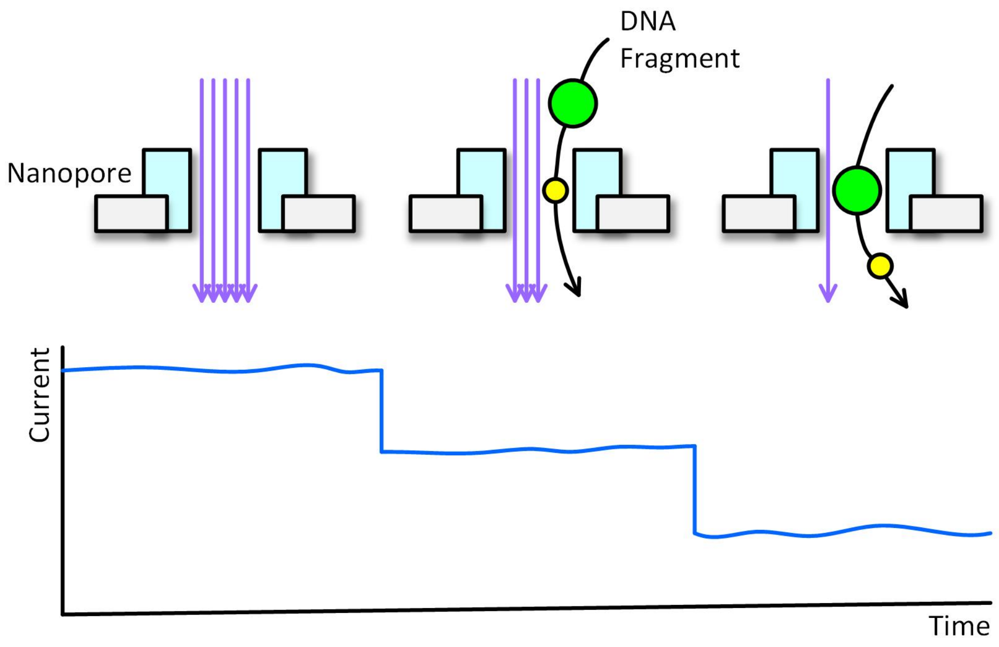


Figure 2.6: The Oxford Nanopore sequencing technology.

2.6.2 Sequencing Errors

This technology is very young and its characteristics have not been thoroughly studied. It is thought that the substitution error rate is similar to the insertion error rate, and the deletion error rate is two times higher than the error rate of the other two errors [29]. The errors are caused by the uneven movement of the DNA strand through the nanopore [30].

CHAPTER 3

BLESS: BLOOM-FILTER-BASED ERROR CORRECTION ALGORITHM

3.1 Introduction

Recent advances in HTS technologies have made it possible to rapidly generate high-throughput data at a much lower cost than traditional Sanger sequencing technology [25]. HTS technologies enable cost-efficient genomic applications, including de novo assembly of many non-model organisms [31], identifying functional elements in genomes [32], and finding variations within a population [33, 34, 35, 36]. In addition to short read length, a main challenge in analyzing HTS data is its higher error rate than traditional sequencing technology [1, 2], and it has been demonstrated that error correction can improve the quality of genome assembly [37] and population genomics analysis [38, 36].

Previous error correction methods can be divided into four major categories [5]: (1) k-mer spectrum based [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 39], (2) suffix tree/array based [17, 18, 19, 20], (3) MSA based [21, 22], and (4) HMM based [23, 24, 40]. However, none of these previous methods has successfully corrected errors in HTS reads from large genomes without consuming a large amount of memory that is not accessible to most researchers. Previous evaluations showed that some error correction tools require over 128 GB of memory to correct errors in genomes with 120 Mbp and the others need tens of GB of memory [5]. For a human genome, previous approaches would need hundreds of GB of memory. Even if a computer with hundreds of GB of memory is available, running such memory-hungry tools degrades the efficiency of the computer. While the error correction tool runs, we cannot do any other job using the computer if most of the memory is occupied by the error correction tool. This can be a critical problem for data centers, where a large amount of data should be processed in parallel.

In several works, Bloom filters [41] or counting Bloom filters [42] were used to save a k-mer spectrum, which includes all the strings of length k (i.e. k-mers) that exist more than a certain number of times in reads [43, 44, 45, 46]. Although the Bloom filter is a memory-efficient data structure, the memory reduction by previous Bloom filters-based methods did not reach their maximum potential because of the following four reasons: (1) The size of a Bloom filter should be proportional to the number of distinct k-mers in reads, and the number of distinct k-mers was conservatively estimated, thus could be much higher than the actual number. (2) They could not remove the effect of false positives from Bloom filters. In order to make the false positive rate of the Bloom filters small, the size of Bloom filters was made large. (3) Because they could not distinguish error-free k-mers from erroneous ones before a Bloom filter was constructed, both of the k-mers needed to be saved in Bloom filters. (4) Multiple Bloom filters (or counting Bloom filters) were needed to count the multiplicity of each k-mer.

Besides the large memory consumption of the existing methods, another problem encountered during the error correction process is that there exist many identical or very similar subsequences in a genome (i.e. repeats). Because of these repeats, an erroneous subsequence can sometimes be converted to multiple error-free subsequences, making it difficult to determine the right choice.

In this dissertation, we present a new Bloom-filter-based error correction algorithm, called BLESS. BLESS belongs to the k-mer spectrum based method but it is designed to remove the aforementioned limitations existed in the k-mer spectrum based solutions. Our new approach has three important new features: (1) BLESS is designed to target high memory efficiency in order for error correction to be run on a commodity computer. The k-mers that exist more than a certain number of times in reads are sorted out, and programmed into a Bloom filter. (2) BLESS can handle repeats in genomes better than previous k-mer spectrum based methods, which leads to higher accuracy. This is because BLESS is able to use longer k-mers compared to previous methods. Longer k-mers resolve repeats better. (3) BLESS can extend reads to correct errors at the end of reads as accurately as other parts of the reads. Sometimes an erroneous k-mer may be identified as an error-free one because of an irregularly large multiplicity of the k-mer. False positives from the Bloom filter can also cause the same problem. BLESS extends the reads to find multiple k-mers that cover the erroneous bases at the end of the reads to improve error correction at the end of the reads.

To identify erroneous k-mers in reads, we need to count the multiplicity of each k-mer. Counting k-mers without extensive memory is challenging [47, 48, 49, 50, 51]. BLESS uses the disk-based k-mer counting algorithm like Disk Streaming of k-mers (DSK) [50] and k-mer Counter (KMC) [47]. However, BLESS needs to save only half of the k-mers that DSK does in hash tables because it does not distinguish a k-mer and its reverse complement.

To evaluate the performance of BLESS, this dissertation used real HTS reads generated with the Illumina technology as well as simulated reads. These reads were corrected using BLESS as well as six previously published methods. Our results show that the accuracy of BLESS is the best while it only consumes 2.5 percent of the memory usage of all the compared methods on average. Our results further show that correcting errors using BLESS allowed us to align 69 percent of previously unaligned reads to the reference genome accurately. BLESS also increased NG50 of scaffolds by 50 percent and decreased assembly errors by 66 percent based on the results from Velvet [52].

3.2 Methods

3.2.1 Overview of the BLESS Algorithm

BLESS belongs to the k-mer spectrum based error correction category [12]. A k-mer is called solid if it exists more than M, the k-mers multiplicity threshold, times in the entire reads, and weak otherwise. If a k-mer extracted from a read is a weak k-mer, it can be considered as having sequencing errors.

Figure 3.1 depicts the high-level diagram of BLESS. In this figure, the cylinders and the rectangle with extra lines depict data written to disk and memory, respectively. To convert weak k-mers to solid k-mers, we need to save the list of the solid k-mers and to query a k-mer to the list efficiently. In Step 1, k-mers in reads are distributed into multiple files, and the multiplicity of k-mers in each file is counted. Then in Step 2, only solid k-mers are programmed into a Bloom filter, and errors in reads are corrected using the Bloom filter. The final step of BLESS is to restore the false corrections made

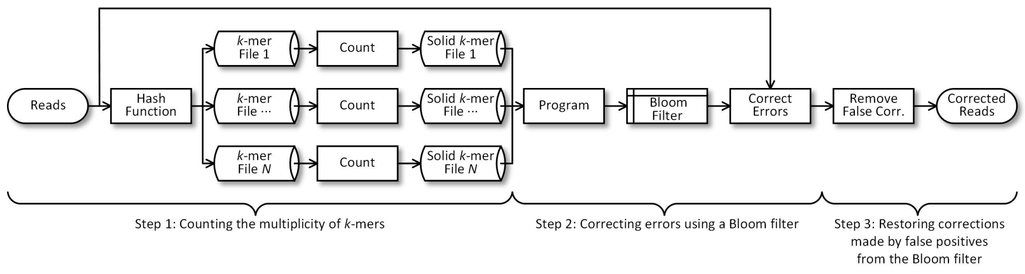


Figure 3.1: The high-level block diagram of BLESS.

by the false positives from the Bloom filter.

3.2.2 Counting the Multiplicity of k-mers

The first step in BLESS is to count the multiplicity of each k-mer, followed by finding the solid k-mers, and programming those solid k-mers into a Bloom filter. By counting the multiplicity of k-mers, we can sort out the solid k-mers that are needed for further analysis. We can also create a k-mer multiplicity histogram to be used to determine the multiplicity threshold M, if M is not predetermined by the user. The total number of solid k-mers is used to determine the size of the Bloom filter.

Figure 3.2 shows how to count the multiplicity of each k-mer. F_s contains unique solid k-mers in the reads and N_s is the number of unique solid k-mers. First, all the k-mers in the reads are distributed into N (default 100) files in order to reduce the required memory for this process. In BLESS, a k-mer and its reverse complement are treated as the same k-mer, which is called a canonical k-mer. If the middle base of a k-mer is A or C (k is always an odd number), the k-mer can be used as a canonical k-mer of itself. If the middle base is G or T, the reverse complement of the k-mer becomes the canonical k-mer of the original k-mer. A hash value is calculated for each canonical k-mer and the file that the k-mer will be written into is determined by using the hash value. Next, the k-mer is written to the file. After this process, all the identical k-mers and their reverse complements are written into the same file. The next step is to open each file that contains k-mers and count the number of k-mers using a hash table. After all the k-mers in the file are updated in the hash table, we check the multiplicity of each k-mer in the hash table. If the multiplicity of a particular k-mer is larger than M, it is a

```
OPEN $N$ files
FOR each read in the $N$ files
    FOR each $k$-mer $kmer$ in a read
        SET $kmer$ to canonical_form(kmer)
        SET $hash\_value$ to $hash\_function(kmer)$
        SET $file\_index$ to $hash\_value$ % $N$
        WRITE $kmer$ to $FILE_{file\_index}$ file
    END FOR
END FOR
CLOSE $N$ files
IF $M$ is not given from a user
    GENERATE the $k$-mer multiplicity histogram
    CALCULATE $M$ using the $k$-mer multiplicity histogram
ENDIF
$N_s$ = 0
FOR $file\_index$ = 0 to $N$ - 1
    OPEN $FILE_{file\_index}$
    FOR each $k$-mer $kmer$ in $FILE_{file\_index}$
        IF $kmer$ exists in the hash table
            INCREMENT the value of $kmer$ in the hash table
        ELSE
            SET the value of $kmer$ to 1
        END IF
    END FOR
    CLOSE $FILE_{file\_index}$
    FOR each key $key$ in the hash table
        IF the multiplicity of $key \geq M$
            INCREMENT $N_s$
            WRITE $key$ to the solid $k$-mer list file $F_s$
        END IF
    END FOR
    WRITE the hash table into a file for future use
END FOR
```

Figure 3.2: The procedure for counting the multiplicity of each k-mer.

solid k-mer and is subsequently written to the solid k-mer list file F_s.

If M is not given by the user, the k-mer multiplicity histogram is generated and M is determined using the histogram. The process of determining M using the k-mer multiplicity histogram is explained later (see Section 3.2.4). After completing this process for all the N files, we can create the solid k-mer list file F_s and determine the number of distinct solid k-mers N_s. The time complexity of counting the multiplicity of k-mers is $O(RL)$, where R is the number of reads and L is the read length.

3.2.3 Correcting Errors Using a Bloom Filter

In order to convert weak k-mers into solid k-mers, we must know the solid k-mer list. If this list was stored in file F_s, it would be impossible to rapidly check whether a k-mer is in the list or not. BLESS solves this problem by recording all the solid k-mers in a Bloom filter, which supports fast membership test while using little memory. An open source C++ Bloom filter library [53] is used in BLESS. When implemented, the size of the bit vector and number of hash functions in the Bloom filter are determined using N_s and a target false positive probability. After constructing the Bloom filter, all the solid k-mers in F_s are programmed into the Bloom filter. The weak k-mers are then converted into solid ones using this Bloom filter.

Let read r be a sequence of symbols A, C, G, T with length L. The i-th base of read r is denoted by $r[i]$, where $0 \leq i \leq L$ - 1. The form $r[i, j]$ is a substring from the i-th base to the j-th base of r. The pseudo code of the correction process for a read r is shown in Figure 3.3. SI_{all} is the set of solid k-mer islands in r, and Z is the number of solid k-mer islands in SI_{all}. This process is initiated from finding all the solid k-mer islands in r. A solid k-mer island consists of consecutive solid k-mers, which is in neighborhoods with weak k-mers or the end of the read. To find them, all the k-mers from $r[0, k$ - 1], to $r[L$ - k, L - 1] are converted to their canonical forms and the canonical forms are queried to the Bloom filter. If the Bloom filter output for a k-mer is true, then the k-mer is solid. If a solid k-mer island has a solid k-mer with quality scores below 10, the k-mer is removed from the solid k-mer island.

The relation between solid k-mer islands and weak k-mers is shown in Fig-

```
SET $SI_{all}$ to all solid $k$-mer islands in $r$
SET $Z$ to the number of solid $k$-mer islands in $SI_{all}$
FOR $si\_index$ = 0 to $Z$ - 1
    ADJUST the length of $SI_{si\_index}$
END FOR
FOR $si\_index$ = 1 to $Z$ - 1
    IF the number of weak $k$-mers between $SI_{si\_index-1}$ and $SI_{si\_index}$ < $k$
        SET $Z$ to 0
        BREAK
    END IF
END FOR
FOR $si\_index$ = 1 to $Z$ - 1
    CORRECT errors between $SI_{si\_index-1}$ and $SI_{si\_index}$
END FOR
IF $SI_{Z-1}$ does not end at the last $k$-mer of $r$
    CORRECT errors that are on the right side of $SI_{Z-1}$
    IF the index of the rightmost modified base > ($read\_length$ - $k$)
        EXTEND $r$ to the right
    END IF
END IF
IF $SI_0$ does not start from the first $k$-mer of $r$
    CORRECT errors that are on the left side of $SI_0$
    IF the index of the leftmost modified base < ($k$ - 1)
        EXTEND $r$ to the left
    END IF
END IF
IF $Z$ == 0
    CORRECT the first $k$-mer of $r$
    IF the first $k$-mer of $r$ is corrected
        CORRECT errors that are on the right side of the first $k$-mer of $r$
    END IF
END IF
```

Figure 3.3: The procedure for correcting errors in a read r.

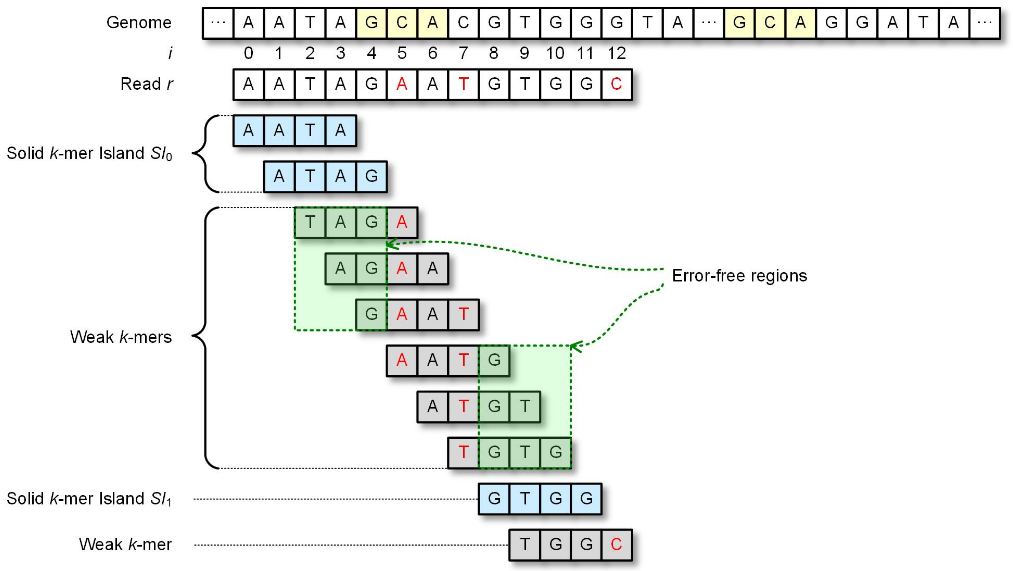


Figure 3.4: An example of solid k-mer islands and weak k-mers.

ure 3.4. Weak k-mers have errors but the errors cannot be in the bases that overlap solid k-mers. This is because the errors that are in the overlapped bases would make the solid k-mers erroneous, while we assume that solid k-mers do not have errors. Therefore a weak k-mer can be converted to a solid one by modifying bases that do not overlap with solid k-mers.

The weak k-mers that exist between two consecutive solid k-mer islands SI_i and SI_{i+1} can be corrected by using the rightmost k-mer of SI_i and the leftmost k-mer of SI_{i+1}. This makes all the corrected bases between SI_i and SI_{i+1} covered by k consecutive solid k-mers. If an erroneous base exists in the first or last k - 1 bases of a read, it is not possible to get consecutive k-mers covering the erroneous base. BLESS solves this problem by extending a read on both ends.

When there is no solid k-mer island in a read, BLESS tries to change the first k-mer to a solid one by substituting low-quality bases with different bases. If the first k-mer is successfully converted to solid k-mer(s), the solid k-mer(s) are traced to the right.

3.2.4 Determining Parameters

Output quality of BLESS is affected by the choice of the k-mer multiplicity threshold, M. The distribution of k-mer multiplicity in the original reads is

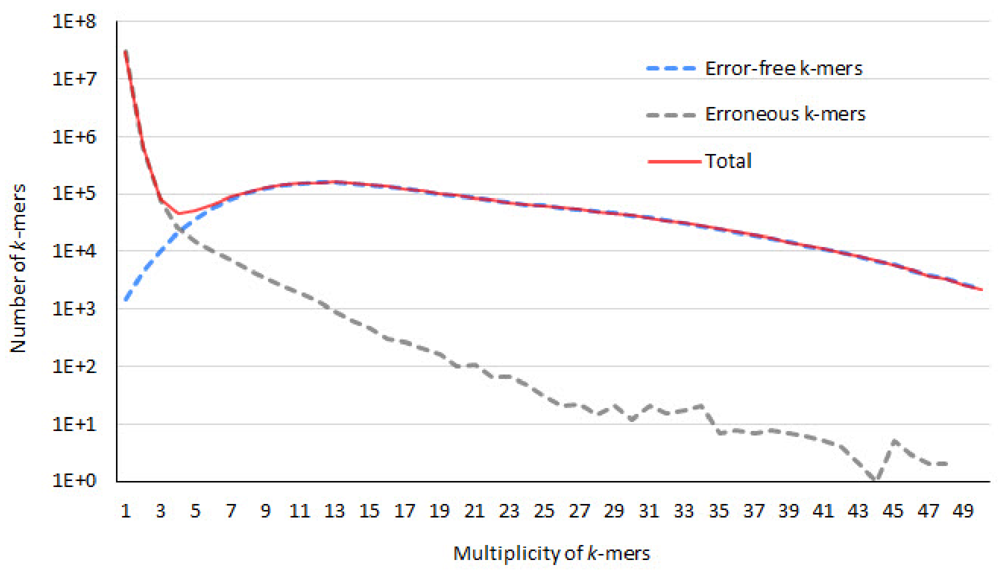


Figure 3.5: The histogram of the multiplicity of 21-mers in *S. aureus* reads.

the mixture of error-free k-mers and erroneous k-mers. The multiplicity of error-free k-mers is known to follow the Poisson distribution and the multiplicity of erroneous k-mers can be fit to the gamma distribution [8, 15]. The histogram of the multiplicity of k-mers usually has the curve like the red line in Figure 3.5 if k is in a reasonable range. Such a histogram can be decomposed into the histogram of error-free k-mers (blue line) and erroneous k-mers (gray line). If M is too small, many erroneous k-mers may be recognized as solid k-mers (i.e. larger false positives and smaller false negatives). On the other hand, if M is too large, many error-free k-mers become weak k-mers (i.e. larger false negatives and smaller false positives).

We define the optimal value of M, $M_{optimal}$, as the M value that minimizes the sum of false positives and false negatives. In BLESS, the histogram like the red line in Figure 3.5 can be easily generated because BLESS already calculated the multiplicity of each k-mer. In the histogram, the sum of false positives and false negatives becomes the minimum when M is the valley point of the U-shape curve with the following two assumptions: (1) As M increases from the value point, the corresponding value of the gray line becomes smaller and the corresponding value of the blue line becomes larger. (2) As M decreases from the valley point, the corresponding value of the gray line becomes larger and the corresponding value of the blue line becomes smaller. This is a reasonable assumption if error-free k-mers and erroneous

k-mers can be fit into the Poisson and gamma distribution respectively, and two distributions are away from each other.

If we assume that the current M value is the valley point and M moves to the right, the sum of false positives and false negatives increases even though the number of false positive decreases. Similarly, if M moves to the left, the sum of false positives and false negatives also increases even though the number of false negatives decreases. Therefore, the sum of false positives and false negatives becomes its minimum when M is the valley point of the histogram of the multiplicity of k-mers.

Choosing the appropriate k is also needed to get more accurate results from BLESS. If k is too long, the average multiplicity of solid k-mers becomes smaller. On the other hand, if k is too short, there may be too many unnecessary paths in the error correction process. This will increase not only the probability that wrong corrections are made but also BLESS's runtime. Unfortunately, BLESS currently cannot automatically determine the optimal k value. However, our empirical analysis shows that the k value that satisfies the following two conditions usually generates the results close to the best one: (1) N_s / $4^k \leq 0.0001$ where N_s represents the number of unique solid k-mers (BLESS reports N_s) and (2) the number of corrected bases becomes the maximum at the chosen k value.

3.3 Results

To assess the performance of BLESS, we corrected errors in five different read sets from various genomes using BLESS and six other error correction methods. All the evaluations were done on a server with two Intel Xeon X5650 2.67 GHz processors, 24 GB of memory, and Scientific Linux.

3.3.1 Data Sets Used in the Evaluation

We used three data sets generated by the Illumina sequencing technology and two simulated read sets. The characteristics of each read set are summarized in Table 3.1. The first read set, labeled D1, is the fragment library of *S. aureus* used in the GAGE competition [37]. The second genome (D2) is high coverage (160 $\times$) low error rate (0.5 percent) *E. coli* reads. The third read

Table 3.1: Details of the HTS read sets used to evaluate BLESS.

Genome	Accession number		Genome	Read	Number	Coverage	Per-base
	Reference	Read	length	length	of reads	(×)	error rate (%)
D1: *S. aureus*	NC010079 NC010063.1 NC012417.1	SRR022868	2,903,080	101	1,096,140	38.1	2.1
D2: *E. coli*	NC000913	SRR001665	4,639,675	36	20,693,240	160.6	0.5
D3: Human Chr14	NC000014.8	N/A	88,289,540	101	36,172,396	41.4	1.4
D4: Human Chr1	NC000001.10	N/A	225,280,621	101	89,220,048	40.0	0.6
D5 (10 ×): Human Chr1 10 Mbp	NC000001.10	N/A	10,000,000	101	990,100	10.0	0.6
D5 (20 ×): Human Chr1 10 Mbp	NC000001.10	N/A	10,000,000	101	1,980,198	20.0	0.6
D5 (30 ×): Human Chr1 10 Mbp	NC000001.10	N/A	10,000,000	101	2,970,298	30.0	0.6
D5 (40 ×): Human Chr1 10 Mbp^{p}	NC000001.10	N/A	10,000,000	101	3,960,396	40.0	0.6

Note: [Genome Length] Length of genomes without N_s, [Number of Reads] Number of reads after all paired reads that contain N_s are removed. [Coverage] Number of Reads × Read Length / Genome Length; [Per-base Error Rate] Mismatches / ((Number of Reads - Unaligned Reads) × Read Length).

set is the fragment library of human chromosome 14 reads (D3) that were also used in the GAGE competition. To check the scalability of BLESS, we also used simulated reads generated from GRCh37 human chromosome 1 (D4). The reads were generated using simLibrary and simNGS [54], after all Ns in the reference sequence were removed. The head of each read indicates the index of the reference sequence where the read is from. Using the information, we also generated an error-free version of D4 (D4$_{\text{Error-Free}}$ hereafter). The last data set D5 was generated to evaluate the improvement of de novo assembly results after error correction. Four read sets with 10-40 × of read coverage and their error-free versions were generated from the first 10 Mbp of the reference sequence for D4 using simNGS.

To provide a controlled assessment of the accuracy of corrections made by BLESS, errors in the input read sets are identified using the error correction evaluation toolkit (ECET hereafter) [5]. ECET first aligns reads to the reference sequence using BWA [55] and identifies a set of differences between the reads and the reference. ECET evaluates corrected reads by counting how many differences in the set are removed. In our evaluations, insertions and deletions were not included in the set because insertions and deletions can be corrected by substitutions and ECET regards these substitutions as wrong modifications. For example, if a genome sequence contains ACGT and a read from the genome has one insertion between C and G (i.e. ACAG), the insertion error can be corrected by substituting the third (fourth) base A (G) with G (T). ECET counts the third and fourth bases as wrong modifications.

3.3.2 Error Correction Accuracy

We compared BLESS with the following existing error correction tools: Quake [8], Reptile [16], HiTEC [17], ECHO [21], and Musket [10]. We chose these tools to compare mainly because they cover the three major categories of error correction methods, i.e., k-mer spectrum based, suffix tree based, and MSA based methods. To the best of our knowledge, PREMIER [24] is the only HMM based error correction tool for DNA reads, and it was not included in our comparison because its source code is not available. In addition, we also considered Bloom filter based methods that were previously published. We selected DecGPU [43] to compare with BLESS because it is the only Bloom filter based method that can run without a Graphics Processing Unit (GPU).

The comparison results of BLESS and the other six error correction tools are summarized in Table 3.2. The outputs of the error correction tools were converted to target error format (TEF) files using the software in ECET in order to measure the accuracy of the corrected reads. In each data set, we counted the following: erroneous bases successfully corrected (true positive, TP), correct or erroneous bases erroneously changed (false positives, FP), erroneous bases untouched (false negatives, FN), and the remaining bases (true negative, TN). Then, sensitivity, gain, and specificity were calculated using these four values. Sensitivity, defined as TP / (TP + FN), shows how many errors in the input reads are corrected. Gain, defined as (TP - FP) / (TP + FN), represents the ratio of the reduction of errors to the total number of errors in the original reads. Gain can be negative if the number of newly generated (FP) errors is greater than the number of corrected errors. Specificity, defined as TN / (TN + FP), shows the fraction of error-free bases left unmodified. DecGPU and Quake cut bases that they cannot correct, and these trimmed bases are considered as FPs in ECET. In our evaluation, trimmed bases were excluded from FPs and thus not used to calculate sensitivity, gain, and specificity because considering trimmed bases as FPs made gain of DecGPU and Quake worse than what they really were.

Table 3.2: Details of the HTS read sets used to evaluate BLESS.

Data	Software	Accuracy Sensitivity	Gain	Specificity	Memory (MB)	Wall-clock Time (min)	Number of Threads
D1	BLESS	0.895	0.894	1.000	11	6	1
	DecGPU	0.076	0.002	0.998	1,556	2	12
	ECHO	0.710	0.707	1.000	6,063	96	12
	HiTEC	0.859	0.838	0.999	2,127	12	1
	Musket	0.709	0.703	1.000	362	2	12
	Quake	0.145	0.144	1.000	644	8	12
	Reptile	0.564	0.518	0.999	1,232	7	1
D2	BLESS	0.968	0.967	1.000	14	23	1
	DecGPU	0.333	0.028	0.998	2,171	5	12
	HiTEC	0.920	0.880	1.000	14,096	83	1
	Musket	0.934	0.926	1.000	347	3	12
	Quake	0.838	0.837	1.000	8,339	74	12
	Reptile	0.957	0.951	1.000	1,008	52	1
D3	BLESS	10.674	0.644	1.000	150	180	1
	DecGPU	0.096	0.058	0.998	2223	28	12
	Musket	0.575	0.537	1.000	3763	31	12
	Quake	0.128	0.126	1.000	2126	62	12
	Reptile	0.577	0.529	0.999	11783	453	1
D4	BLESS	0.892	0.870	1.000	372	459	1
	DecGPU	0.358	0.017	0.998	2473	82	12
	Musket	0.888	0.866	1.000	7815	56	12
	Quake	0.583	0.539	1.000	8863	188	12
	Reptile	0.807	0.704	0.999	19007	1775	1

While some error correction tools such as HiTEC are able to independently choose appropriate parameters, the error correction quality of other tools depends on parameters that have to be set by the users. We generated the corrected read sets that provided the best gain using each error correction tool in order to compare the best results from the methods. To generate such read sets, the values of all the key parameters of each tool were scanned in a continuous fashion within their respective ranges until the gain of each tool reached the maximum. BLESS, Musket, Quake, Reptile, and DecGPU were able to generate results for all the four data sets. ECHO did not complete the error correction for D2 even after 60 hours of running, so we could not produce ECHO results for D2 and larger data sets (i.e. D3 and D4). HiTEC also failed to correct errors in D3 and D4 because it ran out of memory.

As shown in Table 3.2, BLESS consistently outperforms the other correction tools for all the input data sets. For D1-D4, the sensitivity of BLESS is higher than that of the other methods, while the difference between sensitivity and gain of BLESS is smaller than those of the other methods. This suggests that BLESS can correct more errors in the reads and that the results from BLESS always have fewer errors than those from other tools.

The higher accuracy of BLESS comes from its ability to use longer k-mers. If k is too short, an erroneous k-mer may be recognized as solid, because it is more probable that a short erroneous k-mer exists in other parts of the genome. Even though an erroneous k-mer is recognized as a weak k-mer, it may be possible to convert it to multiple solid k-mers if k is too short. Figure 3.6 shows how the number of distinct k-mers changes and approaches N_{ideal} in the reference sequence of D2 and D4 as k increases. N_{ideal} represents the number of distinct k-mers in the reference sequence when all the k-mers in it are distinct. Indeed, more repeats can be differentiated by using longer k, which is helpful in removing ambiguities in the error correction process. The number of distinct k-mers for *E. coli* becomes 96 percent of N_{ideal}, when k is 15. However, the same ratio for human chromosome 1 is only 50 percent for the same k value. When k becomes 31, this ratio for human chromosome 1 surpasses 90 percent. Note that a longer k value does not always guarantee better error correction results, as the average multiplicity of k-mers decreases as k increases. However, if k is too short, it would be more difficult to differentiate solid k-mers from weak ones and k should be increased until a sufficient average k-mer multiplicity is guaranteed.

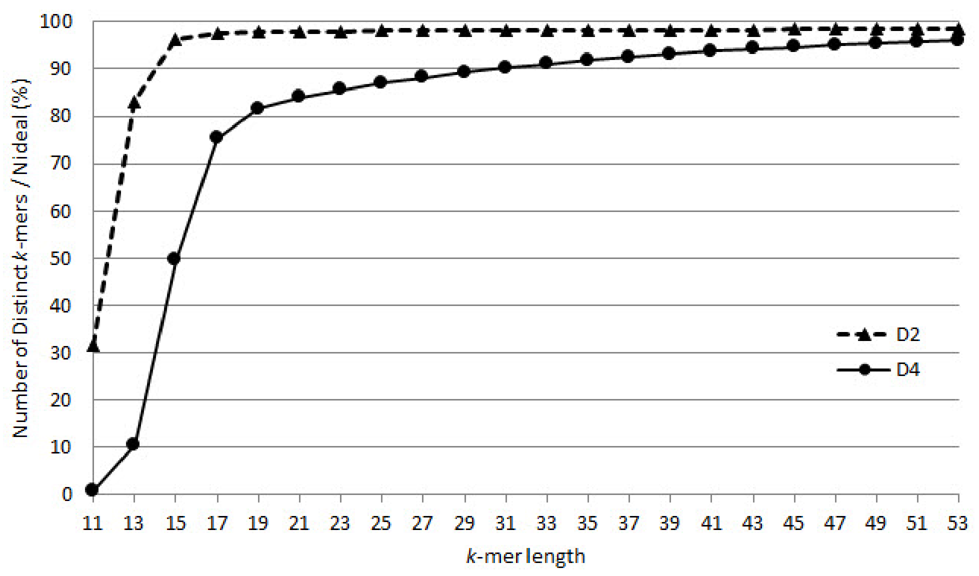


Figure 3.6: The ratio of the number of distinct k-mers in the D2 and D4 reference sequences to their N_{ideal} values.

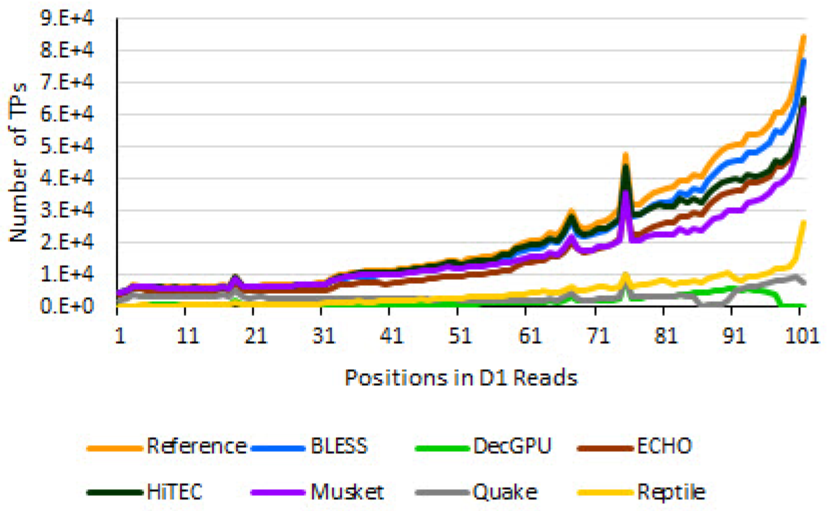
In the HTS reads that were generated using the Illumina technology, errors are usually clustered at the 3-end of the reads. Therefore correcting errors in that region is an important feature of error correction methods although correcting such errors is more difficult than correcting errors in the middle of the reads. BLESS can correct errors at the end of the reads as accurately as in other parts through a reads extension. To assess the number of corrected errors in each position of the reads, we calculated the number of TPs and sensitivity at each position. Figure 3.7A shows the number of TPs in each corrected read set for D1.

In this graph, Reference refers to the number of errors in each position of the original reads, which rapidly increases at the 3-end of the reads. Figure 3.7B shows the ratio of TPs to the number of errors (i.e. sensitivity) in each position of the reads in D1. We observed that BLESS maintains high sensitivity even in the regions where most of the errors are clustered, as indicated by the overall flat contour of the line shown in the figure.

3.3.3 Memory Usage

The peak memory usage and runtime of each method is also displayed in Table 3.2. BLESS' average memory usage is only 2.5 percent of the other

A



B

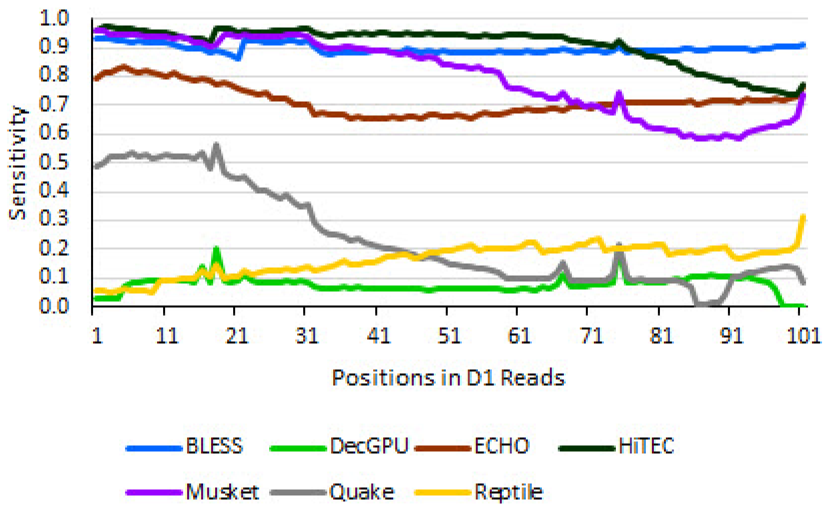


Figure 3.7: The number of TPs and per-base sensitivity calculated in each position of the D1 reads.

methods. On average, BLESS consumes 5.6 percent of the memory that DecGPU does, which is another Bloom filter based method. DecGPU programs k-mers into a counting Bloom filter, which helps the multiplicity of k-mers to be saved with small memory with a certain false positive probability.

BLESS requires less memory than previous Bloom filter based methods for the following reasons. First of all, BLESS can count the multiplicity of k-mers and find out the list of solid k-mers without constructing Bloom filters. Therefore, we eliminate the need to estimate the number of distinct k-mers. We also do not need to program weak k-mers into the Bloom filter. Second, BLESS uses a Bloom filter instead of a counting Bloom filter. Previous methods use counting Bloom filters to count the multiplicity of k-mers, and this information is then used to identify solid k-mers. In BLESS, however, we already know the list of solid k-mers. Therefore it is not necessary to know the multiplicity of k-mers to identify solid k-mers anymore, and solid k-mers can be programmed into a Bloom filter instead of a counting Bloom filter. Finally, BLESS is able to remove false corrections that are generated by false positives from the Bloom filter. Therefore the target false positive probability of the Bloom filter used in BLESS does not need to be very small, which helps to reduce the size of the Bloom filter.

3.3.4 Alignment

To evaluate the impact of error correction on read mapping, we compared the number of reads that could be aligned to the reference sequence with Bowtie [56] before and after error correction. In Table 3.3, each column denotes the percentage of exactly aligned reads out of all the reads. We used the paired-end alignment capability of Bowtie, and the reads that could not be aligned uniquely in the reference sequences were counted.

All error correction methods reduced the number of unaligned reads, but BLESS outperformed the others for all the four inputs. After errors were corrected using BLESS, 81 percent of the entire reads and 69 percent of the initially unaligned reads could be aligned to the reference on average without any mismatches. This ratio was higher than the ratio of the other methods.

D4 is a simulated read set, and we know where each read should be aligned.

Table 3.3: Ratio of the number of exactly aligned reads to the number of entire reads in percentage.

Software	D1	D2	D3	D4
Uncorrected	19.5	73.5	42.8	36.4
Error-Free	N/A	N/A	N/A	80.3
BLESS	75.1	96.5	74.1	77.2
DecGPU	36.6	90.7	55.7	63.8
ECHO	53.6	N/A	N/A	N/A
HiTEC	70.1	95.0	N/A	N/A
Musket	66.9	95.3	69.6	74.3
Quake	58.1	94.3	72.0	65.9
Reptile	48.5	96.1	66.4	67.5

Note: Alignment was performed using the paired-end alignment of Bowtie. [Error-Free] Result for $D4_{\text{Error-Free}}$. When we ran Bowtie, the maximum and minimum values of insert length were set, and this prevented 19.7 percent of the reads from being aligned to the reference sequence.

For each aligned read in the BLESS output, we compared the aligned position and the position where it originated. There were 99.94 percent of the aligned reads aligned to the correct positions. Even though this evaluation could not be done for D1-D3, the percentage of D1-D3 will not be very different from the D4 result because the same strict Bowtie options were used for all the data sets.

3.3.5 De Novo Assembly

Error correction can improve not only read alignment but also de novo assembly results. To compare the effect of error correction methods on de novo assembly, scaffolds were generated using two de Bruijn graph (DBG) based assemblers [52] and [9] with four D5 read sets (10 $\times$, 20 $\times$, 30 $\times$, and 40 $\times$ read coverage). A string graph based assembler SGA [57] was also used in order to show the effect on non-DBG based assemblers. Scaffolds were also made using the output reads of each error correction tool, and all the scaffold sets were compared with one another.

The output quality of Velvet and SOAPdenovo is sensitive to the choice of k. Therefore, all the odd numbers between 35 and 89 were applied to Velvet and SOAPdenovo as k for each input read set. The k value that gave the longest corrected scaffold NG50 was selected. NG50 is the length of the

Table 3.4: Summary of Velvet assembly results for D5 (40 $\times$).

Software	Scaffold corrected NG50 (kbp)	Number of errors in contigs	Number of errors in scaffolds	Genome coverage (%)
Uncorrected	671.0	1,321	0	99.5
Error-Free	1,239.1	543	7	99.8
BLESS	1,004.1	447	2	99.8
DecGPU	751.6	566	2	99.8
ECHO	665.4	827	8	99.8
HiTEC	805.2	813	0	99.7
Musket	1,004.1	476	3	99.8
Quake	850.4	553	2	99.8
Reptile	1,004.0	466	4	99.8

Note: [Error-Free] Assembly results for $\mathrm{D5}_{\text{Error-Free}}$ (40 $\times$). An inversion error means that part of a contig or scaffold comes from a different strand with respect to the true genome. A relocation means that part of a contig or scaffold is matched with a different part within a chromosome. [Number of errors in contigs] Single mismatches + Indels + Inversions + Relocations in contigs. [Number of errors in scaffolds] Inversions + Relocations + Indels in scaffolds.

longest scaffold, S, that the sum of the lengths of scaffolds whose lengths are greater than or equal to S is greater than or equal to half the length of the genome length [58]. For SGA, since the most important parameter is the minimum overlap, all the numbers from 50 to 90 were tested for each data set in order to find the value that generated the longest corrected scaffold NG50. Each scaffold set was evaluated using the GAGE assembly evaluation toolkit [37].

Table 3.4 shows the Velvet assembly results for D5 (40 $\times$). Corrected NG50 is equal to NG50 except that corrected NG50 is calculated after the scaffolds are broken at places where assembly errors occur [37]. The GAGE software generates contigs by splitting scaffolds whenever a run of Ns is found. Errors in contigs include single mismatches, indels, inversions, and relocations. Errors in scaffolds are the summation of indels, inversions, and relocations. Genome coverage shows how many bases in the reference sequence are covered by the scaffolds. Error-Free row shows the assembly results for $\mathrm{D5}_{\text{Error-Free}}$(40 $\times$).

The assembly results of BLESS were better than the others in terms of assembly length and accuracy. Corrected NG50 was improved from 670 kbp to 1,004 kbp after errors were corrected by BLESS. BLESS also reduced the

Table 3.5: Summary of Velvet assembly results for D5 (40 ×).

Data	Best value		BLESS' choice	
	M	*Gain*	*M*	*Gain*
D1	4	0.894	4	0.894
D2	26	0.968	24	0.967
D3	6	0.644	6	0.644
D4	4	0.870	5	0.870

number of errors in the contigs from 1,321 to 449, and improved genome coverage from 99.5 percent to 99.8 percent.

3.3.6 Choosing Parameters Automatically

In BLESS, M affects the output quality, and BLESS can automatically choose this value. Table 3.5 shows how close the values chosen by BLESS are to the best M that makes the gain of BLESS's output the highest. The second column represents the best M; the third column is the corresponding gain when M is the value in the second column. The fourth and fifth columns represent M chosen by BLESS and the corresponding gain. For D1 and D3, the values that BLESS chose were the same as the best M in the second column. For D2 and D4, there are small differences between M chosen by BLESS and the best M. However, the difference between the third and fifth column was 0.001 and 0, respectively. Therefore BLESS's auto M selection capability achieves the best gain or the nearly best gain in all the four input sets.

CHAPTER 4

ACCELERATING THE ERROR CORRECTION PROCESS

4.1 Introduction

Correcting errors in sequencing reads is a time-consuming and memory-intensive process. The occurrences of patterns (k-mers in many tools) in reads should be counted, and patterns with small occurrences should be substituted with ones that have high occurrences. Saving patterns requires a lot of memory for large genomes, and searching for alternative patterns usually takes a long time. Therefore, memory efficiency and fast runtime in error correction methods are as important as their accuracy.

To provide a memory-efficient error correction method, BLESS was developed. While BLESS could generate accurate results with a much smaller amount of memory than previous works, it was too slow to be applied to reads from large genomes.

Recently, some new error correction methods that can correct errors in a large data set in a short period of time have been developed [59, 60]. However, to the best of our knowledge, none of them satisfy all the three constraints (i.e., memory efficiency, runtime, and accuracy).

To address the three requirements, we developed a new version of BLESS, BLESS 2. In BLESS 2, the accuracy of the error correction algorithm has been further improved over that of BLESS by using the quality score distribution for finding solid k-mers with low-quality scores. Also, the algorithm was parallelized using the hybrid Message Passing Interface (MPI) and Open Multi-Processing (OpenMP) programming, which makes BLESS 2 the fastest tool without loss of memory efficiency.

We compared BLESS 2 with five top-performing error correction tools using reads from a human genome. BLESS 2 showed at least seven percent higher gain than its counterparts, and it could correct errors in reads from the entire human genome with only 5.6 GB of memory. In addition to these features, BLESS 2 became the fastest when it was parallelized on two computing nodes using MPI.

4.2 Methods

BLESS 2 is parallelized using the hybrid MPI and OpenMP programming. Therefore, the overall process can be parallelized on a server with multiple CPU cores and shared memory, and it can be accelerated further by running it on multiple servers.

The overall BLESS 2 architecture is shown in Figure 4.1. Rectangles and parallelograms mean procedures and data, respectively. Rectangles with diagonal lines are parallelized using OpenMP. The large gray boxes represent computing nodes in a cluster. First, Node 1 builds a quality score histogram that can be used for the error correction step. Then, all nodes start to fetch input reads to count the occurrence of k-mers. In order to accelerate the k-mer counting step, MPI was applied to KMC [61], which is one of the fastest and the most memory-efficient k-mer counters, and the modified KMC was integrated into BLESS 2. In KMC, k-mers are sent to one of 512 bins, and k-mers in each bin are counted separately. In BLESS 2, each of the N nodes invokes KMC, and counts k-mers in 512 / N bins.

Then Node 1 collects the outputs of N nodes and constructs a k-mer occurrence histogram. This histogram is used to determine the threshold for solid k-mers. Each node separates k-mers in its private bin that have occurrences larger than the threshold, and programs them into its own Bloom filter.

Each Bloom filter thus contains solid k-mers in private to the corresponding node. Bloom filter data in each node is broadcast to all the other nodes, and all the Bloom filter data from N nodes is reduced using a bit-wise OR operation. Now each Bloom filter stores all the solid k-mers in the entire read set. Each node then corrects R / N reads where R is the total number of input reads.

The accuracy of BLESS 2 has been significantly improved over its predecessor. The gain is due to (1) analyzing the distribution of quality scores to find solid k-mers with errors and (2) speculating the locations of errors in

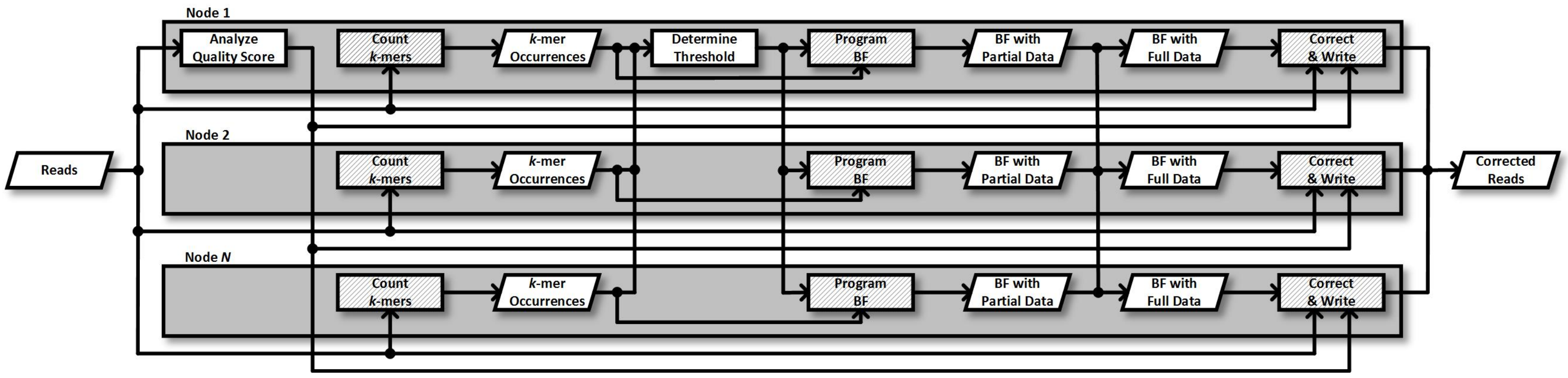


Figure 4.1: Overview of the BLESS 2 structure.

uncorrected regions and trimming these bases.

Corrected reads from all nodes are concurrently written, using MPI, to a final output file.

4.3 Results

In order to evaluate the performance of BLESS 2, it was compared with five state-of-the-art error correction tools. All the experiments were done on servers with two hexa-core Xeon X5650 CPUs and 24 GB of memory.

Human reads from NA12878 were used as the evaluation input. The high-confidence variant set for NA12878 was downloaded from [62], and a new reference sequence R_{NEW} was made by merging the variants with the hg19 sequence. Then ERR194147, which is a read set from NA12878, was aligned to R_{NEW} using BWA [55]. Errors were extracted from the alignment result using SPECTACLE [63]. The input reads were corrected using BFC-KMC r157 [60], BLESS 1.01, Lighter 02/20/2015 [59], Musket 1.1, QuorUM 1.0.0 [64], SGA 01/29/2015 [57], and the accuracy of the outputs was evaluated using SPECTACLE. Each tool was executed multiple times with consecutive k-mer length values from 25 to 60, and the result with the best gain for substitutions, deletions, and insertions was chosen.

The evaluation results are summarized in Table 4.1. D2 is a set of reads that could be aligned to R_{NEW}. D1 is a set of reads from human chr1-3, which is a subset of D2. We prepared D1 because only BLESS 2, BFC-KMC, and Lighter could handle D2 with 24 GB of memory on our server. For D1, BLESS 2 generated corrected reads with the best accuracy; its gain was higher than those of the others by at least seven percent and ten percent on average. Also, there was little accuracy degradation in D2. While BLESS 2 consumed the smallest amount of memory for D2, Lighter used less memory than BLESS for D1. This is because KMC that BLESS invokes to count k-mers requires up to 4 GB of memory. For D2, the size of the Bloom filter in BLESS 2 was larger than 4 GB and KMC was no longer a memory bottleneck.

The runtime of BLESS 2 on one computing node was comparable to that of the other methods, and BLESS 2 became the fastest tool when more nodes were available. When four nodes were used, BLESS 2 became 2.3 times faster than when one node was used. The current version of BLESS 2 reads input

Table 4.1: Error correction results.

Data	Software	Gain	Memory (GB)	Runtime (min)
D1	BLESS			
	BLESS 2 (1 node)	0.565	3.7	65
	BLESS 2 (2 nodes)	0.565	3.7	41
	BLESS 2 (3 nodes)	0.565	3.7	33
	BLESS 2 (4 nodes)	0.565	3.7	30
	BFC-KMC	0.505	11.2	60
	Lighter	0.439	3.2	54
	Musket	0.454	13.2	133
	QuorUM	0.477	10.5	120
	SGA	0.448	12.0	396
D2	BLESS 2 (1 node)	0.563	5.6	320
	BLESS 2 (2 nodes)	0.563	5.6	203
	BLESS 2 (3 nodes)	0.563	5.6	160
	BLESS 2 (4 nodes)	0.563	5.6	139
	BFC-KMC	0.496	20.5	274
	Lighter	0.434	13.9	231

Note: [TP] erroneous bases that are correctly modified; [FP] all bases that are incorrectly modified; [FN] erroneous bases that are not modified; [Gain] (TP - FP) / (TP + FN); twelve threads were used in a node for all the tools.

read files three times (i.e., for analyzing quality scores, counting k-mers using KMC, and correcting errors), consuming a significant amount of time. Since there is no efficient way to read a compressed file in parallel, this part cannot be accelerated even though the number of involved nodes increases. In the next version, KMC will be merged with BLESS 2 and quality scores could be analyzed while k-mers are counted.

CHAPTER 5

SPECTACLE: SOFTWARE PACKAGE FOR ERROR CORRECTION TOOL ASSESSMENT

5.1 Introduction

Rapid improvements in next-generation sequencing (NGS) technologies have allowed us to generate a huge amount of sequencing data at a low cost. However, the quality of the data has not improved at the same pace as the throughput of the NGS technologies. For example, the latest Illumina sequencing machine HiSeq X Ten can produce 1.8 tera base pairs (bp) in each run, but only about 75 percent of the bases are guaranteed to have Phred scores of over 30 [65].

Errors in HTS reads degrade the quality of downstream analysis, which could be improved by correcting the errors [66, 67, 37]. Many stand-alone methods for correcting the errors in DNA sequencing data have been developed [68, 69, 17, 70, 21, 8, 10, 22, 59, 16]. Some DNA assemblers have their own error correction modules, which can be also used as error correction tools [71, 72, 57].

HTS has also been used for transcriptome analysis [73], and such RNA sequencing data have sequencing errors as well. However, not all the error correction methods for DNA sequencing data can correct errors in lowly expressed transcripts. To solve this problem, [23] developed an error correction tool dedicated for RNA sequencing data.

Recently, third-generation sequencing (TGS) technologies that do not require amplification have been developed [74], and single-molecule real-time (SMRT) sequencing technology from Pacific Biosciences is one of them. Even though the sequencing system that uses the SMRT sequencing technology can generate reads with up to tens of thousands of base pairs long, it has about 12 percent of error rate and errors are evenly distributed in reads [75]. Also, the dominant error types of the technology are insertions and deletions that are rare in Illumina reads. Due to these characteristics, dedicated error correction methods for PacBio reads have been developed [76, 77, 28, 78].

Despite such a large number of error correction methods, only a few studies have been carried out on the evaluation of the accuracy of these methods. Such scarcity is mainly due to the difficulty involved in discerning how many errors were corrected and how many were newly generated in the error correction process. While checking whether substitution errors are corrected can be easily done by measuring the Hamming distance between a reference sequence and a corrected read, it is not so simple to evaluate how accurately errors are corrected when insertions and deletions also exist as errors. The evaluation becomes more complex when reads are trimmed during error correction. Aligning a read to the source genome does not always solve this problem since multiple best alignments can exist [79]. Heterozygosity also makes the evaluation hard. In a diploid genome, the same locus in a pair of chromosomes could have different alleles. Therefore, one of them will be recognized as a sequencing error if reads from heterozygous genomes are compared with one reference sequence.

To the best of our knowledge, only three research works have been done to quantitatively evaluate how exactly errors in NGS reads have been corrected. The first, called Error Correction Evaluation Toolkit (ECET) [5] consists of two software packages, one of which evaluates Illumina reads and the other, 454 or Ion Torrent reads. The reason for having two separate algorithms for dealing with the different technologies is that the dominant error models of 454 and Ion Torrent reads are insertions or deletions in homopolymers while most errors in Illumina reads are substitutions [80, 81].

In the second research work, Molnar et al. [82] try to find out the correctness of reads or k-mers in the outputs from Illumina error correction tools instead of directly checking the correctness of bases. Their method calculates (1) how many error-free reads or k-mers cover each base in a genome and (2) how many bases in a reference sequence are covered by error-free reads or k-mers, then checks how the two numbers are changed by error correction.

The last one is compute_gain that is a part of an error correction tool package Fiona [79]. It aligns both a read and its corrected version to a reference sequence, and calculate the difference in edit distance between the two alignments. Ambiguities in alignments are resolved by placing gaps at the leftmost or rightmost possible position.

Even though the three methods opened up ways of evaluating the outputs from error correction methods, all of them have limitations. The software package for Illumina reads in ECET can only work with the tools that explicitly specify the number of bases trimmed from both ends of reads. Even when this information is available, separate programs for each error correction tool are needed to extract the number of trimmed bases, because the tools output the number in different ways.

The software package for 454 or Ion Torrent reads in ECET can evaluate reads with insertions and deletions but the evaluation results could be wrong for trimmed reads.

Even though the software developed by Molnar et al. [82]. can be applied to the outputs from any Illumina error correction method, it cannot be extended to other sequencing technologies. Since PacBio reads, for example, have a high error rate and errors are evenly distributed in the reads, it is hard to get error-free long k-mers. If short k-mers are used by this tool for the evaluation of PacBio reads, specificity would be low because it is likely that the same or similar k-mers exist in other parts of the genome sequence. Also, it would be hard to get sufficient number of error-free corrected reads due to high error rate and long length of PacBio reads.

The evaluation result of compute_gain, like that of ECET, could be wrong. Because the alignment scores used in compute_gain were designed to evaluate edit distance and users cannot change the values, a read could be aligned to a reference sequence in totally different ways before and after error correction, which makes a wrong evaluation result.

Addressing the limitations in these evaluation works, we have developed a Software Package for Error Correction Tool Assessment on nuCLEic acid sequences (SPECTACLE), and we used it to evaluate error correction methods for Illumina and SMRT that are the most popular NGS and TGS technologies. The key contributions of this work can be summarized as follows:

1. We have developed a new error correction tool evaluation algorithm that is independent of underlying error models, and have implemented it with 20,000 lines of Perl and C++ code. It can evaluate any error correction tool for NGS and TGS reads. It works for both DNA and RNA sequencing data, and differentiates heterozygous alleles from sequencing errors.

2. We have designed input read sets that stress the challenges in error correction such as heterozygosity, coverage variation, and repeats, and the

input sets are available in our website with the software parameters. These reads can be used as standard inputs for the evaluation of error correction tools.

3. We have compared 21 state-of-the-art error correction tools for NGS and TGS reads using our data set. This will give readers systematic evaluations of strengths and weaknesses of the tools and indicate potential ways for their further improvement.

In the sections that follow, we will explain how we prepared the inputs for our evaluation and how the evaluation algorithm works. We then present and discuss the evaluation results and what should be done in the future.

5.2 Evaluation Methods

Figure 5.1 shows the SPECTACLE flows for evaluating error correction tools with DNA simulated reads and DNA real reads. Each flow consists of two steps. In the first step, the locations of errors in input reads are determined, and in the next step this information is used to evaluate the output of an error correction tool. The two steps will be explained in detail in the following subsections. The basic flow for evaluating RNA error correction tools is similar and is explained in the supplementary document.

5.2.1 Preparing Input Data

SPECTACLE supports using both simulated reads and real reads to utilize their unique strengths. When simulated reads are used, we can determine the exact locations of errors in the reads. Moreover, reads can be generated from multiple reference sequences with some differences in order to check whether an error correction tool is able to differentiate heterozygosity from sequencing errors. However, if a read simulator cannot exactly model real reads, using such reads could produce misleading results.

The biggest advantage of using real reads is that no assumptions or modeling artifacts exist behind the sequencing data. Therefore, real reads can have some interesting properties that may not be accurately modeled in simulated reads. On the other hand, there can be ambiguities in finding error locations in real reads. In order to find the error locations in real reads, the

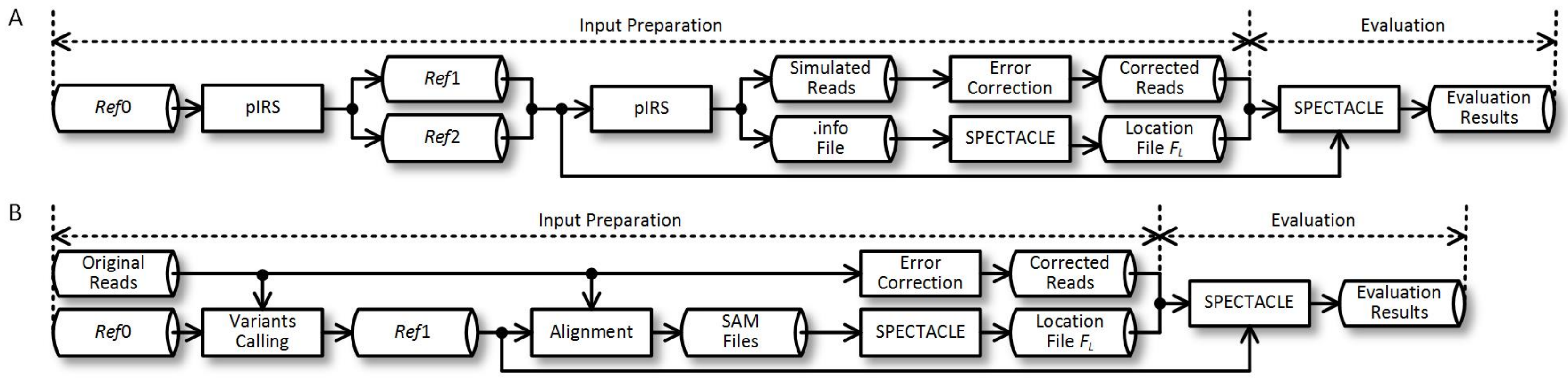


Figure 5.1: Flowchart of evaluating corrected DNA sequencing reads using SPECTACLE.

reads need to be aligned to a reference sequence, and this can cause some problems. First, it is possible that a read is aligned to multiple similar locations in a reference sequence (or to the same location in different ways), and determining the correct alignment is sometimes impossible. In the case of highly repetitive genomes, ambiguous alignments occur frequently, raising the chances of inaccurate evaluation results. Second, reads and a reference sequence might come from different samples, and the differences between them (i.e., variants) will also be recognized as errors. Third, the evaluation results depend on the accuracy of the alignment tool.

Even though SPECTACLE can work with the output reads from any read simulator that gives error location information in a Sequence Alignment/Map (SAM) format, we used pIRS [83] for generating simulated Illumina reads. Error correction becomes challenging when there are heterozygosity and read coverage variations [66, 83], and pIRS can produce reads that stress these characteristics. First, pIRS can generate reads using a diploid genome, and consequently the reads have both sequencing errors and heterozygosity. Second, pIRS can change read coverage depth of a specific genomic region according to the GC-content of the region. Figure 5.1A depicts the evaluation flow for simulated reads. First, two reference sequences $Ref1$ and $Ref2$ that represent a pair of chromosomes in a diploid genome are generated by adding different variant sets to the input reference sequence $Ref0$. Once the two sequences are created, reads are generated from $Ref1$ and $Ref2$. The maximum ploidy level that SPECTACLE supports is two.

After the reads are generated, the locations of errors in the reads should be written in an error location file F_L. F_L contains (1) the positions where reads originate in the genome, (2) the locations of substitutions, insertions, and deletions in each read, and (3) reference sequences from which each read was sampled (i.e. $Ref1$ or $Ref2$). When pIRS generates reads, it also produces a file containing the error locations (i.e. .info file) and .info file is converted into F_L. In order to simulate PacBio reads, we used PBSIM [84]. PBSIM generates a Mutation Annotation Format (MAF) file for indicating error locations, and the file is converted to FL. Because SMRT sequencing technology does not use amplification, coverage variation due to different GC-content values was not considered in simulated PacBio read generation. The PacBio reads are generated from one reference sequence because the error rate of PacBio reads is much higher than the frequency of heterozygous sites and we do not expect the evaluation results to be altered appreciably by adding heterozygous points.

Figure 5.1B shows the evaluation flow for real reads. If input reads and a reference sequence $Ref0$ do not come from the same sample, there can be variants between them; the variants would be recognized later in the flow as sequencing errors. To overcome this problem, a new reference sequence, $Ref1$, is generated by calling the variants and applying them to $Ref0$. In our evaluation, BWA [55] and SAMtools [85] were used for variant calling. The variants are added to $Ref0$ using VCFtools [86], the input reads are aligned to $Ref1$, and the alignment results in the SAM file are converted to F_L. Among the substitution errors in F_L, the errors generated by heterozygous alleles are removed by comparing F_L with the variant calling result.

5.2.2 Evaluating the Accuracy of Corrected Reads

Let R_C be the corrected version of a read R. In order to evaluate the accuracy of R_C, we should find corrected errors and newly added errors in R_C. SPECTACLE first takes the segment G_R from a reference sequence where read R was sampled. Then, R_C is aligned to G_R for finding the errors in R_C (errors missed by a tool, or introduced by a tool). We used a modified version of the Gotoh algorithm [87] for handling trimmed bases and extract all the alignment with the best alignment score.

There can be a set of alignments ALN_{BEST} having the same highest alignment score for a read R_C, but each alignment would imply different numbers of corrected and newly introduced errors. In this case, SPECTACLE calculates the penalty of the newly introduced errors in R_C of each alignment utilizing the scores used in the alignment step. Then, the alignment aln_{BEST} from ALN_{BEST} that has the least penalty is chosen. SPECTACLE makes the choice using the following equation, where $ERR(aln)$ and $ERR(R)$ are the sets of errors in an alignment aln and R:

$$aln_c = \underset{aln \in AL_{BEST}}{\arg\max} \sum_{err \in E((aln) \setminus E(R))} penalty(err) \tag{5.1}$$

After aln_{BEST} is chosen, we can discern from it which errors in $ERR(R)$

are corrected and how many errors are newly added during correction. In order to classify the bases in input reads, we introduce a triplet notation, each character of which should be either Y or N. The first character indicates whether the base in the original read is correct (Y) or not (N), the second character indicates whether the base has been modified by an error correction tool (Y) or not (N), and the third one indicates whether the base in the corrected read at that position is correct (Y) or not (N). For example, NYY describes a base that is erroneous in R, modified by an error correction tool, and error-free in R_C. All the bases should fall into one of the five categories: NNN, NYN, NYY, YNY, and YYN because YYY, YNN, and NNY are logically impossible. Using these triplets, the accuracy metrics that are summarized in Table 5.1 are calculated. Because substitutions, insertions, and deletions are counted separately, we can get three different sets of statistics for each error type, respectively.

Table 5.1: Accuracy metrics.

Metrics	Equations
Sensitivity	sum(NYY) / (sum(NYY) + sum(NYN) + sum(NNN))
Gain	(sum(NYY) - sum(YYN) - sum(NYN)) / (sum(NYY) + sum(NYN) + sum(NNN))
Specificity	sum(YNY) / (sum(YYN) + sum(YNY))
Precision	sum(NYY) / (sum(NYY) + sum(YYN) + sum(NYN))
F-score	2 sum (NYY) / (sum(NYY) + sum(YYN) + 2sum(NYN) + sum(NNN))

Though the above evaluation metric applies to PacBio reads as well, it may take a long time to apply the above algorithm to a large number of PacBio reads owing to their long length and high error rate. In order to evaluate long reads with high error rate in a reasonable amount of time, SPECTACLE supplies an alternative mode that calculates percentage similarity of reads. Percentage similarity of a read set S_R is defined using the follow equation, where N_{RM}, N_{RMM}, N_{RI}, and N_{RD} are the number of matched bases, the number of mismatched bases, the number of inserted bases, and the number of deleted bases in the alignment result of R, respectively:

$$\text{Percentage Similarity} = \sum_{R \in S_R} \frac{N_{RM}}{N_{RM} + N_{RMM} + N_{RI} + N_{RD}} \tag{5.2}$$

SPECTACLE calculates percentage similarity both for input reads and for their error correction results, and shows how this number is improved after error correction. Most PacBio error correction methods trim uncorrected regions in reads. After this process, R_C could be split into multiple pieces and become much shorter than R. Therefore, SPECTACLE reports read coverage that indicates how long total read length is and NG50 [37] that shows how long the average read length is. In addition to these metrics, SPECTACLE can report other detailed analyses such as related to supporting read coverage that help users understand the characteristics of an error correction tool in depth.

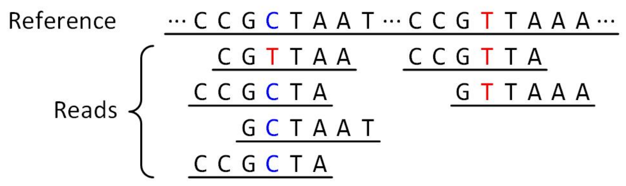


Figure 5.2: Supporting reads and supporting read coverage.

Figure 5.2 explains a supporting read, supporting read coverage, and differential supporting read coverage. Supporting reads are the reads that include a specific position of a reference sequence with a specific base at the position. In the left side, there is a read CGTTAA with an erroneous base T, and three more correct reads are also sampled there. In this example, the number of supporting reads (i.e. supporting read coverage) for T at that position of the reference sequence is 1, while supporting read coverage for C is 3. However, there is another similar sequence in the reference sequence (i.e. repeats) and the reads sampled at the right region could be supporting vector for T at the left side, which makes it hard to correct the error. Differential supporting read coverage of an erroneous base can be defined as (supporting read coverage of correct base) - (supporting read coverage for the erroneous base).

An error in a read becomes difficult to correct if the corresponding correct base has low supporting read coverage. This is because most error correction tools recognize bases with low supporting read coverage as errors. Low differential supporting read coverage also makes error correction harder, because then both a correct base and an erroneous base have a similar number of supporting reads. SPECTACLE gives the percentage of corrected bases against supporting read coverage for correct bases, and the percentage of corrected bases against differential supporting read coverage.

SPECTACLE also collects the percentage of corrected bases in each position of reads (i.e., point sensitivity). Based on this, users can judge whether an error correction tool can correct errors in a specific region of reads or not. This report can lead SPECTACLE users to discern how the output of an error correction tool can be polished further, how multiple error correction algorithms can be combined, and how an error correction algorithm can be improved further. There are some indirect measurements that provide an idea about how good the corrected reads are in the context of downstream analyses. One of the most intuitive ways to evaluate these is to count the number of corrected reads that can be aligned to a reference sequence without mismatches or indels. However, this result can be misleading when reads are aligned to wrong positions in a reference sequence. In order to avoid this, SPECTACLE has the capability to compare the aligned locations of reads in a SAM format with F_L. If insertions or deletions in a read are corrected, the aligned position of the read can be shifted. SPECTACLE determines the largest shift amount of aligned positions for each read using the number of insertions and deletions, and checks whether reads are aligned within this range. It then reports the number of reads aligned correctly within this predicted range. The average number of times each base in the reference sequence is covered by error-free reads (i.e. error-free read coverage) and the fraction of a reference sequence that is covered by error-free reads (i.e. chromosome coverage) are important metrics that indicate the quality of a read set [82]. SPECTALCE collects the two numbers using the exact alignment result above.

5.3 Results

We evaluated 17 Illumina read error correction tools and four PacBio read error correction methods using SPECTACLE. All the experiments were done on a cluster, each computing node of which has two six-core Intel Xeon X5650 processors and 24 GB of memory.

In the following sections, we have included only selected results that highlight the strengths and weaknesses of the tools. The remaining results, software versions, and software command line options are available in the supplementary document of [63].

5.3.1 Data Preparation

Preparing Illumina Read Sets

Table 5.2: Details of Illumina read sets.

ID	Reference					Read		
	Species	Accession Number	GL (Mbp)	GC (%) Avg.	GC (%) Std.	Length	Cov. (X)	Error Rate (%)
I1-10X	R. sphaeroids	NC_007488.1 NC_007489.1 NC_007490.1 NC_007493.1	4.6	68.8	6.3	100	10	0.4
I1-20X						100	20	0.4
I1-30X						100	30	0.4
I1-40X						100	40	0.4
I2-10X	B. cereus ATCC 10987	NC_003909.8 NC_005707.1	5.4	35.5	6.3	100	10	0.4
I2-20X						100	20	0.4
I2-30X						100	30	0.4
I2-40X						100	40	0.4
I3-10X	O. sativa Chr. 5	NC_008398.2	29.9	44.0	13.5	100	10	0.4
I3-10X						100	20	0.4
I3-10X						100	30	0.4
I3-10X						100	40	0.4
I4-10X	Mouse Chr. Y	NC_000087.7	88.1	38.9	8.0	100	10	0.4
I4-20X						100	20	0.4
I4-30X						100	30	0.4
I4-40X						100	40	0.4
I5-10X	Human Chr. 1	NC_00001.11	230.5	41.7	10.6	100	10	0.4
I5-20X						100	20	0.4
I5-30X						100	30	0.4
I5-40X						100	40	0.4
I6	B. cereus ATCC 10987	NC_003909.8 NC_005707.1	5.4	35.5	6.3	100	40	0.2

Note: [G_L] genome length without Ns; [GC] average GC contents; [Cov.] read coverage; [Error Rate] ((total number of substitutions) + (total number of inserted bases) + (total number of deleted bases)) / (total number of bases in reads).

As discussed above, coverage variation, heterozygosity, and repeats complicate error correction, and all the three factors were considered when we prepared input reads for our evaluation. The Illumina read sets we prepared are described in Table 5.2. Five different genomes I1-I5 were used to generate simulated read sets. Even though high coverage read sets are popular, correcting errors in low coverage reads is still important. For example, cancer genome samples could be the mixture of cancer genomes and normal genomes, and the portion of one of the genomes could be very low [88]. Error correction tools for such genomes should have the capability to correct errors in low coverage reads. Therefore, read sets having both high and low coverage values are considered, and the coverage of each set is indicated using the postfix -10X, -20X, -30X, and -40X.

I1, I2, and I3 are bacterium genomes that have different GC-content values. I4 is the mouse chromosome Y known as a highly repetitive genome [89]. I5 is human chromosome 1, the largest genome sequence used in our experiments.

To evaluate the results for real reads, we downloaded D6 from the Illumina website [90]. The reads were sequenced from the exact same strain as I2 using the Illumina MiSeq sequencer. Because the coverage of the reads is over 2,500 X, we down-sampled the reads to 40 X. Details regarding the down-sampling can be found in the supplementary document.

Even though SPECTACLE can assess the outputs from error correction tools for RNA sequencing reads, the evaluation results for such tools have been excluded from the main document. SEECER [23] is the only software available for correcting RNA sequencing reads; for some input parameters, however, SEECER would occasionally terminate abnormally. So we ran SPECTACLE with SEECER with the parameters for which the tool had merely completed execution.

Preparing PacBio Read Sets

The read sets used for evaluating PacBio error correction tools are shown in Table 5.3. The PacBio error correction tools evaluated in this study require, in addition to PacBio reads, Illumina reads that are much more accurate than the PacBio reads. These Illumina reads are described in the “Illumina” column of Table 5.3. In order to evaluate the effect of Illumina read coverage on the accuracy of error correction for PacBio reads, we prepared four different Illumina read sets with different read coverage values (suffixed -10X, -20X, -30X, and -40X). 40X-EF is an error-free version of 40X and the read set was used to evaluate the effects of sequencing errors in Illumina reads on error correction for PacBio reads. P1 is *E. coli* K12 M1665 strain, and both the PacBio reads and the Illumina reads are real reads. The PacBio reads were downloaded from Pacific Biosciences DevNet [91], and reads shorter than 500 bp were filtered out. Four Illumina read sets with different read coverage values were generated by taking different number of reads from SRR922409.

P2 is the first 10 Mbp region of human chromosome 19, which was used for evaluating the scalability of the PacBio error correction tools. We first tried using the entire human chromosome 19. However, only LoRDEC could

Table 5.3: Details of PacBio read sets.

ID	Reference			PacBio			Illumina			
	Species	Accession Number	G_L (Mbp)	Length (bp)	Cov. (X)	Error Rate (%)	Accession Number	Length (bp)	Cov. (X)	Error Rate (%)
P1-10X	E. coli	NC_000913.3	4.6	500-14,494	21	23.4	SRR922409	97	10	0.7
P1-20X							SRR922409	97	20	0.6
P1-30X							SRR922409	97	30	0.6
P1-40X							SRR922409	97	40	0.6
P2-10X	Human Chr19 10 Mbp	NC_000019.10	10	500-15,000	20	20.3	N/A	100	10	0.4
P2-20X							N/A	100	20	0.4
P2-30X							N/A	100	30	0.4
P2-40X							N/A	100	40	0.4
P2-40X-EF							N/A	100	40	0.0

Note: [G_L] genome length without Ns; [Cov.] read coverage; [Error Rate] (⟨total number of substitutions ⟩+ ⟨total number of inserted bases⟩+ ⟨total number of deleted bases⟩) / ⟨total number of bases in reads⟩.

be finished within 70 hours, which is the maximum allocated runtime in our cluster; as a result, we had to use a portion of the chromosome. The PacBio reads and the Illumina reads for P2 were simulated using PBSIM and pIRS, respectively.

5.3.2 Running Error Correction Tools

Running Illumina Read Error Correction Tools

The input read sets were corrected using the 17 error correction tools that had shown good accuracy in the previous evaluations or had been newly published at the time of running the evaluations. Among these, the stand-alone error correction tools are BFC [60], BLESS, Blue [68], Coral [22], ECHO [21], HiTEC, Fiona, Lighter [59], Musket, Quake, QuorUM [64], RACER [70], Reptile [16], and Trowel [92]. The remaining three tools are parts of DNA assemblers, ALLPATHS-LG [71], SGA [57], and SOAPdenovo [9].

For each error correction method, we applied successive numbers to the key parameters of the tools, and generated multiple corrected output read sets corresponding to each parameter. The output read sets were assessed using SPECTACLE and the read set that had the highest gain for substitutions, insertions, and deletions was chosen. The maximum k-mer length for Quake was limited to 18, beyond which the memory capacity of our server was exhausted.

ALLPATHS-LG, BFC, BLESS, Blue, Musket, Quake, QuorUM, RACER, Reptile, SGA, and SOAPec succeeded in generating outputs for all the input read sets. Coral, HiTEC, Fiona, and Trowel failed to correct errors in large genomes because of insufficient memory. ECHO had not finished after 70 hours for the I4 and I5 read sets. Lighter finished correcting all the read sets but it made no correction for the read sets with 10 X coverage.

Running PacBio Read Error Correction Tools

Widely used PacBio read error correction tools LoRDEC [76], LSC [77], PBcR [28], and Proovread [78] were evaluated using P1 and P2. No parameter tuning was needed for LSC, PBcR, and Proovread. For LoRDEC, we generated multiple output sets by applying successive values for k-mer length and solid k-mer occurrence threshold, and chose the result that gave the highest percentage similarity explained in Section 5.2.2. We could not assess LSC using P2 because it had not finished after 70 hours.

5.3.3 Evaluation Results for Illumina Error Correction Tools

Accuracy of Illumina Error Correction Tools

Sensitivity and gain for substitution errors for the 40 X input read sets are summarized in Table 5.4. For all the bacterium genomes I1, I2, and I3, ALLPATHS-LG, BLESS, Lighter, Musket, Quake, QuorUM, and SGA generated outputs with gain above 0.95. For the highly repetitive genome I4, BLESS and Quake outperformed the others, and only these two tools obtained gain above 0.8. For I5, the largest input genome, ALLPATHS-LG, BFC, BLESS, Lighter, Musket, Quake, QuorUM, and SGA showed gain above 0.9. Other than BFC, these are the same tools that worked well for I1-I3. In the evaluation using I6, most tools showed similar performance as they did for I2 since both I2 and I6 were generated from B. cereus. However, Coral, Quake, Reptile, SOAPec, and Trowel showed a degradation of above 0.1 for the gain value in I6 when compared with I2.

The difference between sensitivity and gain shows how many false corrections were made by each tool. In general, BFC, BLESS, Quake, SGA, and SOAPec generated fewer false corrections than the others.

Table 5.5 shows variation in gain with different read coverage values for I5. Only BLESS, Musket, and Quake generated gain above 0.85 for all the read sets. Lighter showed good results for 20-40 X reads, but it could not correct the errors in I5-10X. BFC, BLESS, Musket, Quake, SGA, and SOAPec made a small number of false corrections for low coverage read sets. Gain was saturated in most tools when read coverage became 30 X.

The percentage of corrected bases as a function of supporting read coverage for I5-40X is shown in Figure 5.3. ALLPATHS-LG, Quake, and QuorUM corrected more errors than the others when supporting read coverage of correct bases was close to 1. Even though ALLPATHS-LG and QuorUM have the capability to correct errors with low supporting read coverage, gain for

Table 5.4: Sensitivity and gain of substitution errors for the 40 X Illumina read sets.

Software	I1-40X		I2-40X		I3-40X		I4-40X		I5-40X		I6	
	Sens.	Gain	Sens.	Gain	Sens.	Gain	Sens.	Gain	Sens.	Gain	Sens.	Gain
ALLPATHS-LG	0.998	0.983	0.998	0.984	0.990	0.966	0.851	0.690	0.969	0.904	0.960	0.958
BFC	0.964	0.964	0.960	0.959	0.948	0.940	0.777	0.711	0.934	0.920	0.981	0.979
BLESS	0.998	0.997	0.998	0.998	0.990	0.983	0.905	0.855	0.975	0.964	0.979	0.977
Blue	0.998	0.961	0.998	0.970	0.981	0.883	0.850	0.520	0.896	0.819	0.982	0.903
Coral	0.979	0.913	0.987	0.934	N/A	N/A	N/A	N/A	N/A	N/A	0.817	0.806
ECHO	0.831	0.784	0.949	0.900	0.856	0.803	N/A	N/A	N/A	N/A	0.831	0.822
Fiona	0.998	0.973	0.998	0.980	0.984	0.902	0.677	0.237	N/A	N/A	0.970	0.967
HiTEC	0.997	0.982	0.997	0.993	N/A	N/A	N/A	N/A	N/A	N/A	0.965	0.959
Lighter	0.995	0.992	0.996	0.995	0.974	0.966	0.656	0.586	0.939	0.913	0.973	0.971
Musket	0.996	0.995	0.996	0.995	0.973	0.964	0.773	0.698	0.909	0.886	0.958	0.955
Quake	0.988	0.988	0.990	0.990	0.973	0.970	0.856	0.830	0.920	0.913	0.738	0.736
QuorUM	0.999	0.997	0.999	0.998	0.981	0.969	0.779	0.709	0.951	0.925	0.982	0.977
RACER	0.996	0.913	0.997	0.968	0.961	0.708	0.587	-0.097	0.902	0.114	0.967	0.946
Reptile	0.958	0.933	0.968	0.960	0.926	0.824	0.672	0.562	0.878	0.760	0.852	0.831
SGA	0.996	0.996	0.996	0.996	0.975	0.968	0.738	0.673	0.959	0.939	0.947	0.944
SOAPec	0.671	0.670	0.664	0.664	0.650	0.648	0.478	0.446	0.624	0.614	0.539	0.538
Trowel	0.817	0.814	0.836	0.833	0.835	0.818	0.599	0.469	N/A	N/A	0.677	0.675

Note: [Sens.] Sensitivity.

Table 5.5: Sensitivity and gain of substitution errors for the I5 read sets with different coverage values.

Software	I5-10X		I5-20X		I5-30X		I5-40X	
	Sensitivity	Gain	Sensitivity	Gain	Sensitivity	Gain	Sensitivity	Gain
ALLPATHS-LG	0.911	0.811	0.964	0.886	0.968	0.897	0.969	0.904
BFC	0.810	0.749	0.919	0.891	0.929	0.912	0.934	0.920
BLESS	0.931	0.898	0.961	0.946	0.975	0.960	0.975	0.964
Blue	0.848	0.690	0.894	0.809	0.896	0.818	0.896	0.819
Fiona	0.942	0.837	N/A	N/A	N/A	N/A	N/A	N/A
Lighter	N/A	N/A	0.918	0.867	0.938	0.907	0.939	0.913
Musket	0.889	0.860	0.905	0.882	0.907	0.885	0.909	0.886
Quake	0.908	0.896	0.917	0.910	0.920	0.912	0.920	0.913
QuorUM	0.894	0.810	0.952	0.907	0.952	0.922	0.951	0.925
RACER	0.819	-2.287	0.898	-0.164	0.902	0.052	0.902	0.114
Reptile	0.805	0.612	0.869	0.728	0.876	0.754	0.878	0.760
SGA	0.852	0.803	0.941	0.917	0.955	0.936	0.959	0.939
SOAPec	0.585	0.545	0.622	0.609	0.624	0.613	0.624	0.614

Note: [Sens.] Sensitivity.

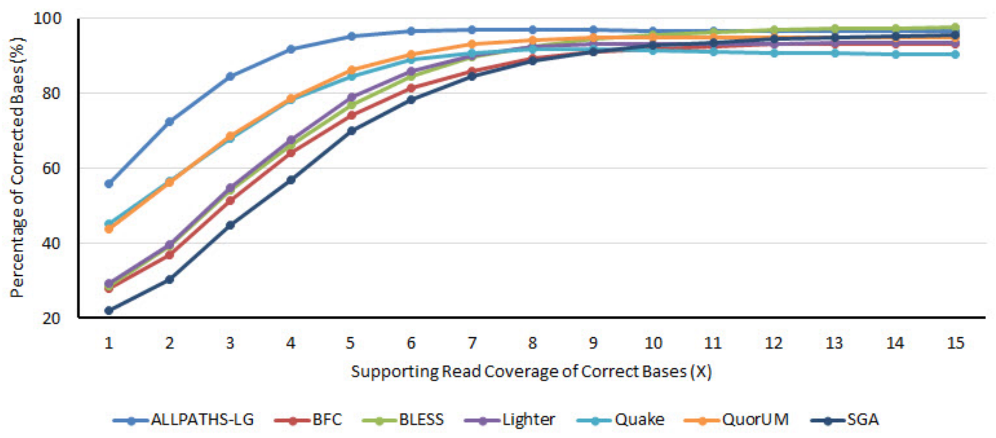


Figure 5.3: The percentage of corrected errors in I5-40X for various supporting read coverage of correct bases.

I5-10X of the tools in Table 5.5 was not as impressive as this result. This is because they also generated many false positives for this input set. The effect of differential supporting read coverage on error correction was significant only when read coverage was low.

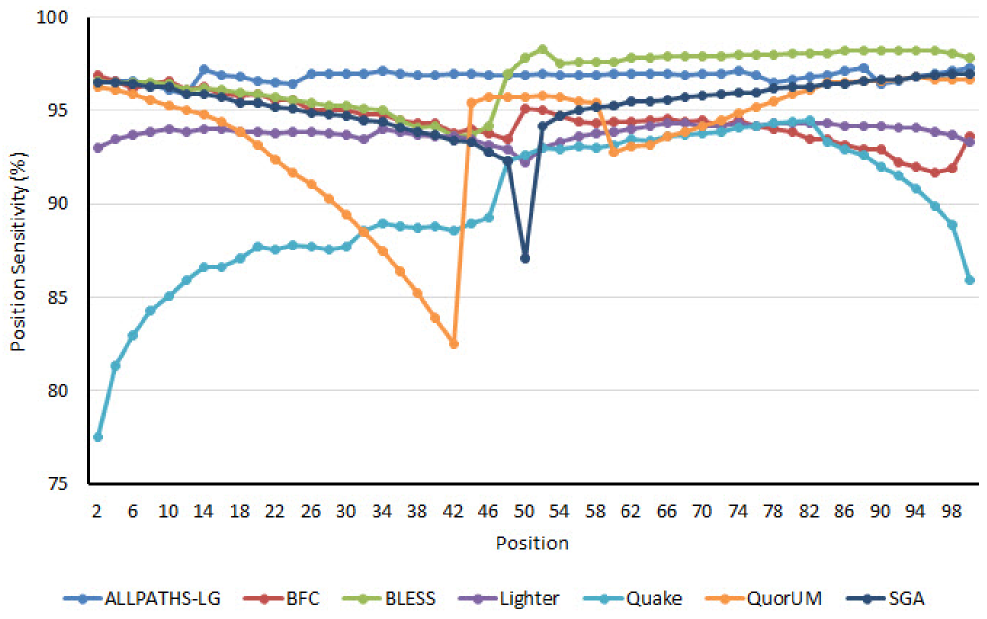


Figure 5.4: Point sensitivity of the I5-40X reads.

As shown in Figure 5.4, tools can correct different percentages of errors in different locations in reads. The plots for ALLPATHS-LG, BFC, BLESS, and Lighter show relatively flat lines, which means that they corrected almost the same proportion of errors in all the positions in reads. On the other hand, plots for QuorUM and SGA have deep valley points, and the positions of these regions with little correction match with the k-mer length used with these tools for generating the respective outputs. In addition, Quake could only correct a relatively small number of errors at both ends of reads compared to the others.

Alignment Results for Illumina Error Correction Tools

Table 5.6 shows how many corrected reads can be exactly aligned to reference sequences. Reads were aligned using the paired-end alignment feature of Bowtie [56] without allowing any mismatches or indels. The genomes I1-I5 have two reference sequences, and corrected read sets were aligned to the

Table 5.6: Alignment results of 40 X Illumina read sets.

Software	I1-40X		I1-40X		I3-40X		I4-40X		I5-40X		I6	
	Aligned	Correct	Aligned	Correct	Aligned	Correct	Aligned	Correct	Aligned	Correct	Aligned	Correct
Original	52.52	100.00	50.86	100.00	51.16	99.99	51.26	99.54	51.12	99.98	81.07	100.00
ALLPATHS-LG	99.07	99.98	99.07	99.97	98.51	99.93	88.76	97.52	96.88	99.91	98.68	99.99
BFC	98.40	100.00	98.23	100.00	97.41	99.98	89.33	98.14	96.65	99.98	98.39	100.00
BLESS	99.83	100.00	99.85	99.99	99.23	99.98	92.80	99.08	98.59	99.98	98.65	100.00
Blue	99.64	99.90	99.68	99.92	96.07	99.66	84.35	92.67	93.08	99.73	98.78	99.94
Coral	92.13	98.72	92.52	98.52	79.26	97.84	51.26	99.54	N/A	N/A	95.96	99.57
ECHO	87.46	99.99	93.33	99.99	88.52	99.94	N/A	N/A	N/A	N/A	94.98	100.00
Fiona	98.28	99.96	98.65	99.94	95.28	99.76	70.46	94.31	N/A	N/A	98.17	99.99
HiTEC	98.78	99.99	99.30	99.99	N/A	N/A	N/A	N/A	N/A	N/A	97.83	100.00
Lighter	99.30	100.00	99.47	100.00	98.13	99.99	79.71	99.33	96.13	99.98	98.22	100.00
Musket	99.49	100.00	99.50	100.00	97.87	99.98	84.32	98.33	93.86	99.98	97.79	100.00
Quake	99.57	100.00	99.58	100.00	98.41	99.99	88.17	98.76	94.71	99.98	95.82	100.00
QuorUM	99.88	100.00	99.90	100.00	98.78	99.98	86.54	98.74	97.29	99.98	98.64	99.99
RACER	98.51	99.96	99.29	99.96	96.40	99.94	74.16	99.24	92.95	99.95	98.36	99.99
Reptile	97.77	99.99	98.25	99.97	92.00	99.86	79.47	97.22	89.65	99.92	96.69	99.99
SGA	99.57	100.00	99.60	100.00	98.53	99.99	86.72	98.87	97.61	99.98	97.95	100.00

Note: [Aligned] the percentage of aligned reads to the total number of reads; Correct: the percentage of reads that were aligned to correct positions to the number of aligned reads; Original: pre-correction results.

reference sequence from which they originated among the two sequences. The alignment results are well matched with the results in Table 5.4, and the tools that showed high sensitivity also had more reads aligned correctly to the reference sequences.

In almost all the cases, the ratio of correctly aligned reads to the total number of aligned reads was over 99 percent with the exception of I4. For I4, only the corrected reads from BLESS, Lighter, and Racer showed the accuracy of over 99 percent.

Runtime and Memory Usage of Illumina Error Correction Tools

Table 5.7: Memory usage and runtime of Illumina error correction tools for I5-20X and I5-40X.

Software	Memory Usage (MB)		Runtime (min)	
	I5-20X	I5-40X	I5-20X	I5-40X
ALLPATHS-LG	12,287	18,424	122	435
BFC	10,753	10,889	12	21
BLESS (1 node)	3,813	3,825	9	15
BLESS (2 nodes)	3,809	3,799	5	9
Blue	20,286	20,398	29	46
Lighter	1,107	1,109	9	13
Musket	4,215	6,647	19	36
Quake	13,760	21,643	74	143
QuorUM	8,163	8,686	10	22
RACER	12,623	14,490	17	35
Reptile	13,016	17,422	815	1,711
SGA	1,874	3,508	61	125
SOAPec	4,985	9,708	42	71

To compare how the runtime and memory usage of various tools scale with size of the input, we compared each Illumina error correction method for two cases, I5-20X and I5-40X which has twice the number of reads as I5-20X. These results are summarized in Table 5.7. Except Reptile, all the evaluated Illumina error correction tools support parallelization, and 12 threads were used for the tools. In addition to running parallel threads on a single node, BLESS can also be parallelized across multiple nodes using MPI. BLESS results on two computing nodes are reported separately. For I5-40X, BLESS, Lighter, and SGA could correct the read set using under 4 GB of memory. BFC, BLESS, Blue, Lighter, QuorUM, and RACER used almost the same memory for both 20 X and 40 X coverage reads. The fastest tools were BLESS and Lighter and they were over 13 times faster than ALLPATHS-LG. ALLPATHS-LG required 3.6 times longer time for correcting I5-20X than I5-40X.

Effect of Using Different Alignment Tools on the Evaluation of Real Reads

For real reads, we compare the errors corrected by an error correction tool against mismatches and indels obtained in aligning the reads to a reference sequence. Therefore, the number and the locations of errors could vary according to alignment tools. We generated two FL files from I6 using BWA [55] and Bowtie 2 [93] with default options, and the two files were compared. While BWA found 473,090 substitution errors in D6, Bowtie 2 found 632,705. About 97 percent of substitutions in the BWA set were also found in the Bowtie 2 set, which means Bowtie 2 is more aggressive than BWA and it could indicate more errors in reads. When the error correction results were evaluated using the F_L file from Bowtie 2, sensitivity and gain dropped by up to 8 percent compared to the results with the F_L file from BWA because some of the new errors found by Bowtie 2 were not corrected in the error correction tools.

5.3.4 Evaluation Results for PacBio Error Correction Tools

Due to the high error rate of PacBio reads, error correction outputs could have many uncorrected bases. Therefore, most PacBio error correction tools generate two types of reads: (1) trimmed reads that only contain corrected regions in input reads and (2) untrimmed reads that include both corrected and uncorrected regions in input reads. While PBcR only produces trimmed reads, LSC and Proovread generate both trimmed reads and untrimmed reads, and they were assessed separately. For LoRDEC, trimmed reads were generated from the untrimmed reads using lordec-trim-split that is included in the LoRDEC package.

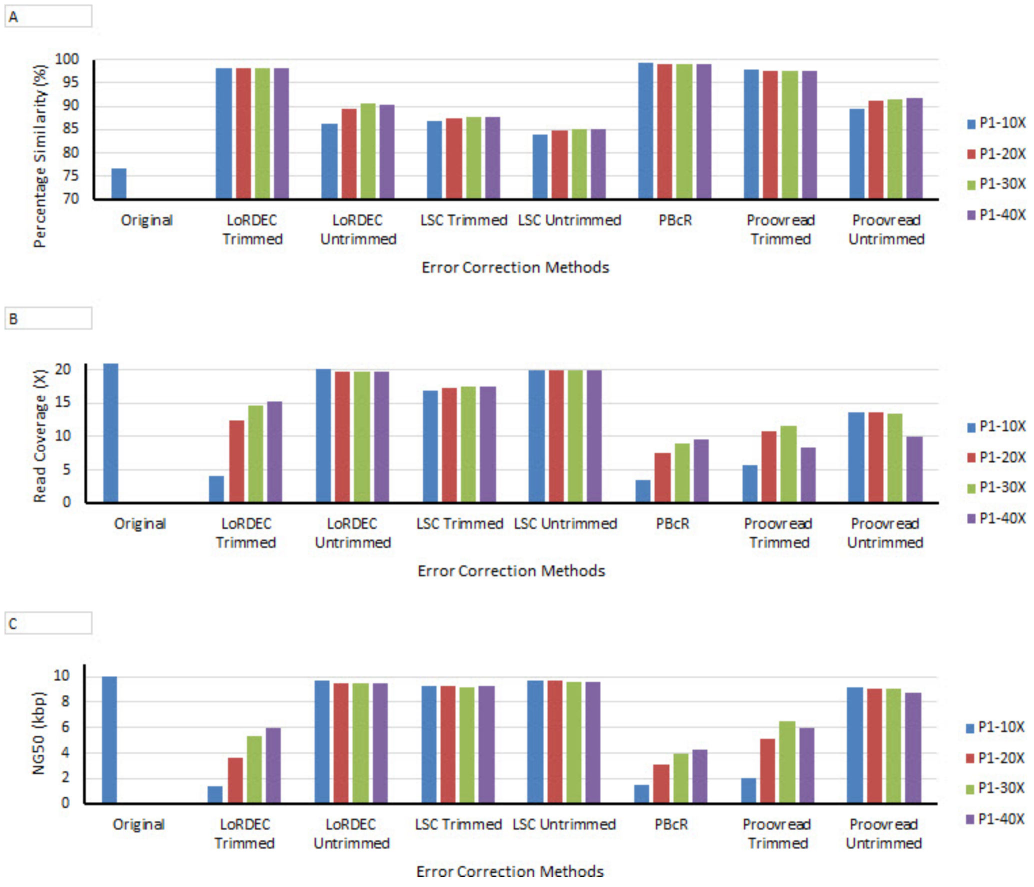


Figure 5.5: Point sensitivity of the I5-40X reads.

In Figure 5.5A, percentage similarity of the outputs from PacBio read error correction methods for P1 are compared. Percent similarity of the input reads was 76.6 percent before error correction, and all the output results were better than this number. Among the four tools, three tools except LSC showed percent similarity over 95 percent for the trimmed reads. For the untrimmed reads, LoRDEC and Proovread generated more accurate reads than LSC. Except the untrimmed LoRDEC reads, read coverage of Illumina reads gave almost no impact on percentage similarity.

Figure 5.5B and Figure 5.5C show read coverage and NG50 of the outputs of the compared tools. The two charts have similar shapes and the values became high when percentage similarity in Figure 5.5A was low. The trimmed LoRDEC reads and the PBcR outputs were improved a lot by increasing Illumina read coverage. The trimmed reads from Proovread were also improved

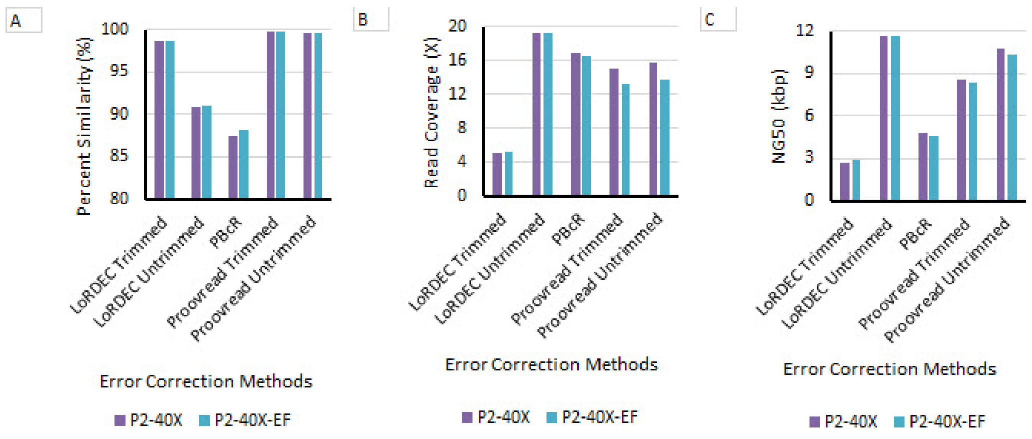


Figure 5.6: Percentage similarity, read coverage, and NG50 of PacBio read error correction methods for P2.

but the values were saturated for 30 X coverage.

Percentage similarity, read coverage, and NG50 are compared for P2-40X and P2-40X-EF that is the error-free version of P2-40X in Figure 5.6. Both the trimmed Proovread reads and the trimmed LoRDEC reads showed high percentage similarity. Percentage similarity and read coverage of the untrimmed Proovread reads were almost the same compared to those of the trimmed Proovread reads. However, NG50 of the trimmed Proovread reads was shorter than that of the untrimmed Proovread reads. LoRDEC generated the trimmed reads with high percent similarity but it removed too many bases and read coverage and NG50 of the read set became much lower than those of the original input reads.

For all the three metric, P2-40-EF did not make a meaningful difference when it was compared with P2-40. This means sequencing errors in Illumina reads are not important when Illumina read coverage is about 40 X.

Alignment Results for PacBio Error Correction Tools

We aligned input PacBio reads and their error correction results using BWA with "-x pacbio" option, and evaluated the alignment results. Before error correction, over 95 percent of P1 PacBio reads and over 98 percent of P2 PacBio reads could be aligned to the reference sequences, hence the number was not improved much after error correction.

The ratio of the number of reads that were aligned without any mismatches or indels to the total number of corrected reads is shown in Figure 5.7. The

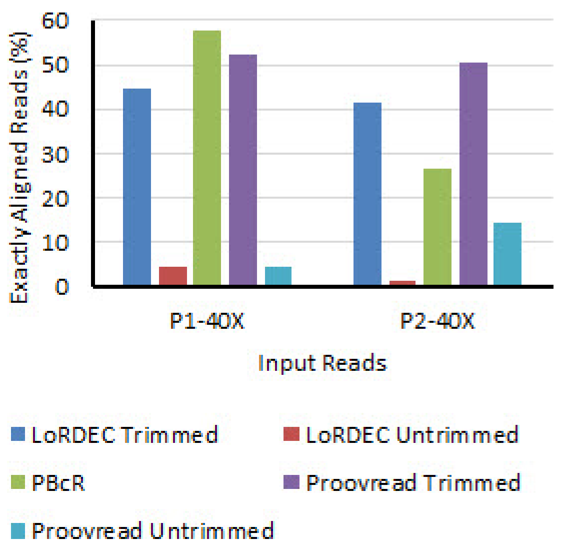


Figure 5.7: Ratio of the number of reads aligned without any mismatches or indels to the number of corrected reads for P1-40X and P2-40X.

ratio was 0 both for P1 and for P2 before error correction, and some error correction methods improved the number a lot. For P1, over 50 percent of trimmed reads from PBcR and Proovread could be aligned to the reference sequence without any differences. Proovread also showed a good result for P2. However, PBcR generated much worse results for P2 than for P1. The ratio of the LSC trimmed reads for P1 was 0.3 percent and no untrimmed LSC read could be aligned to the reference sequence with no difference. Among untrimmed corrected reads, the quality of the reads from Proovread was the best, and 4.3 percent and 14.5 percent of the reads could be aligned without mismatches or indels for P1 and P2, respectively.

Memory Usage and Runtime of PacBio Error Correction Tools

Memory usage of the PacBio error correction methods is summarized in Figure 5.8A. LoRDEC was the most memory efficient method and it could correct all the reads with under 1 GB of memory. Memory usage of LSC was sensitive to Illumina read coverage, and correcting P1-40X required two times larger memory than that for correcting P1-20X. PBcR corrected errors with relatively small memory for P1, but memory usage increased by four times from P1 to P2. Memory usage of Proovread was constant for all the inputs. This was because Proovread splits PacBio reads into chunks with the small

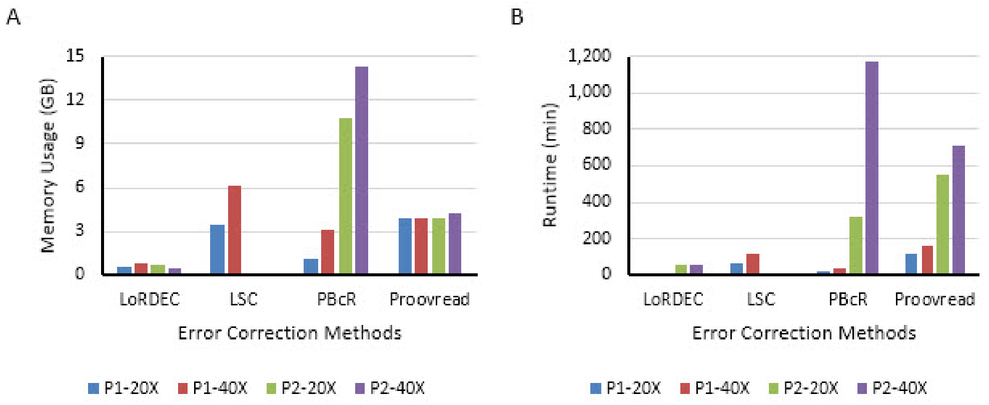


Figure 5.8: Memory usage and runtime of PacBio error correction tools for P1-20X, P1-40X, P2-20X, and P2-40X.

size (20 MB in the experiments).

Runtime of the tools are shown in Figure 5.8B. LoRDEC was much faster than the others and the difference became larger as the size of genome and Illumina read coverage increased. Runtime of LSC was not that long for P1 but it could not finish error correction for P2 even after 40 times longer duration was allowed compared to the runtime for P1. Runtime of PBcR was sensitive both to genome length and Illumina read coverage. Proovread was the slowest among the assessed tools for P1 but it was less sensitive to genome size than PBcR and it became the second fastest for P2.

CHAPTER 6

ENHANCING VARIANT CALLING ACCURACY BY IMPROVING THE QUALITY OF SEQUENCING DATA

6.1 Introduction

Earlier chapters have gone over what sequencing errors are, how they can be corrected, and how the accuracy of corrected outputs can be evaluated. This chapter discusses how sequencing errors affect downstream analyses that use sequencing data as inputs.

One of the most important aspect of research in genetics is to associate genetic variations with heritable phenotypes. To find germline genetic variations, reads should be aligned to a reference sequence. If many reads that span a specific position of the reference sequence have the same base that is different from the reference base, we can suppose that the sample from which the reads were generated have a variant at the position.

In some cases, new genetic variations that were not inherited from the parents could happen in a cell in the course of cell division. These variants are called somatic variants as distinguished from germline variants. From a clinical point of view, finding somatic variants is a very important process, as they are related to many diseases like cancer. Variants found in tumor samples should be partitioned into germline variants and somatic variants. This is usually done by comparing variants in tumor samples with those in normal samples that are taken from the same patient [94].

When sequencing errors exist in reads, the errors could dilute the signal from variants, and consequently the variant might not be detected (i.e., false negatives). It is also possible that multiple sequencing errors that exist at the same position could cause a variant calling tool to report a wrong variant (i.e., false positives).

In this research, the effect of sequencing errors on germline and somatic variant calling has been analyzed. In order to do it, an environment that can generate reads and ground truth variants with different properties was created, and variant calling results for the generated reads were compared with the corresponding ground truth variants. Based on the results, it has been studied how to improve the accuracy of variant calling by manipulating input reads, and new software called ROOFTOP (RemOve nOrmal reads From TumOr samPles) has been implemented using the new knowledge. The performance of ROOFTOP was evaluated using reads from the environment and another read set that was used for a somatic variant calling challenge. The results showed that ROOFTOP improved the accuracy of somatic variant calling by up to 23 percent.

The sections that follow explain the process to prepare reads and the experimental results using the read sets.

6.2 Method

6.2.1 Germline Variant Calling

Preparing Input Reads

Figure 6.1 shows how to generate data sets for germline variant calling. This environment is built using VarSim [95] as a baseline framework. First, single nucleotide variants (SNVs) and indels that are shorter than 50 bp are randomly sampled from the Single Nucleotide Polymorphism Database (dbSNP) [96]. Structural variations (SVs) that are longer than or equal to 50 bp are also randomly chosen from Database of Genomic Variants (DGV) [97]. These two variant sets are used as the inputs for VarSim. These variants V_N can be used as a ground truth variant set with which a germline variant calling result is compared.

VarSim generates a new diploid reference genome REF_N by adding V_N to an input reference sequence REF_{IN}. Then, reads are simulated from REF_N using ART [98]. In order to generate multiple read sets with different read coverage values, C read sets R_{NPi} $(1 \leq i \leq C)$ are generated. Each of R_{NPi} has reads with 1 $\times$ read coverage, and reads sets with different coverage can be made by merging different number of R_{NPi}. R_{NEFPi} that are error-free versions of R_{NPi} are also generated. When ART simulates reads, it can

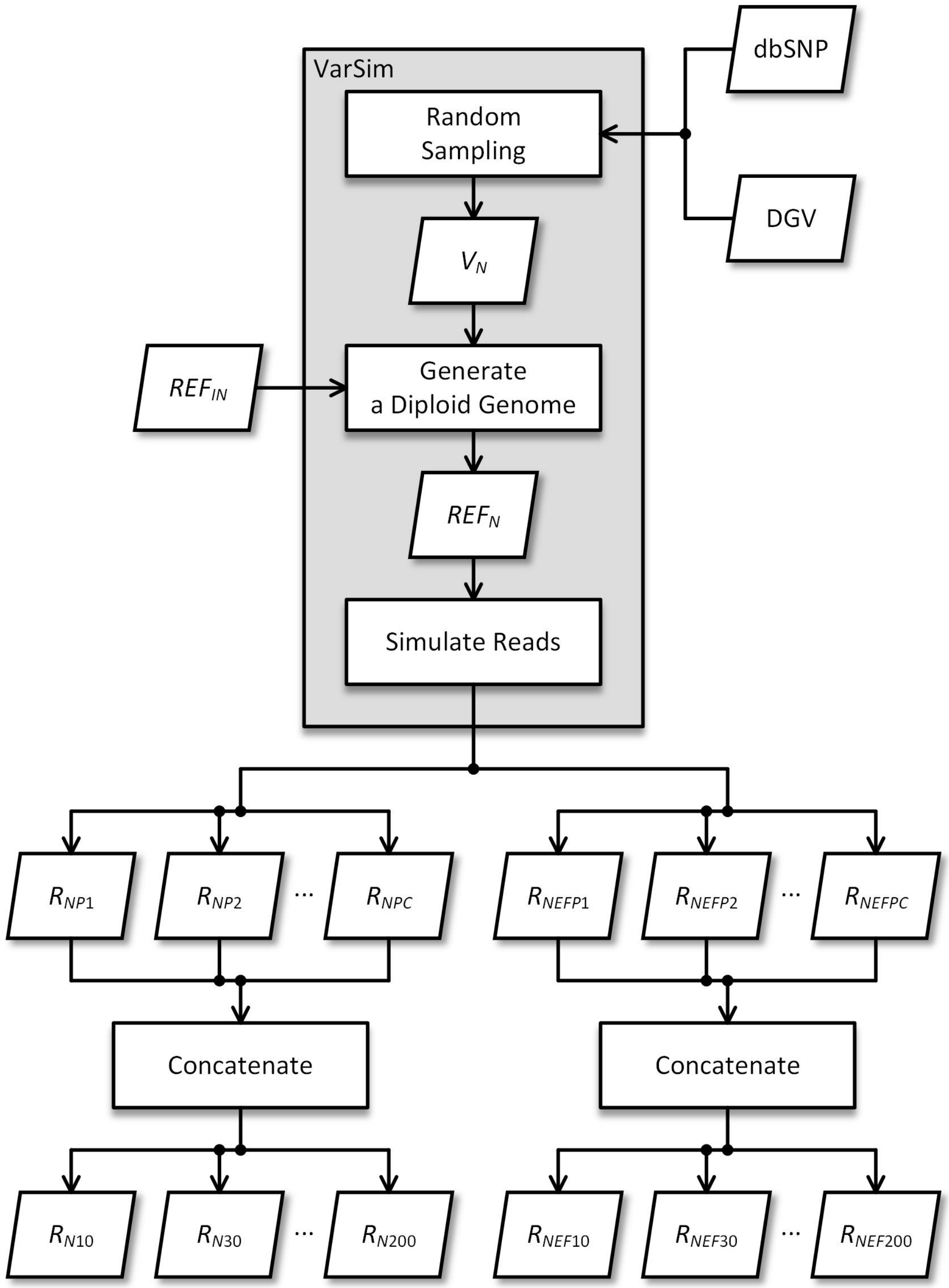


Figure 6.1: Overall flow of generating data sets for germline variant calling.

generate SAM files that have no sequencing errors, and R_{NEFPi} are generated by converting the SAM files to FASTQ files using Picard [99].

R_{N10}, R_{N30}, R_{N50}, R_{N100}, and R_{N200} are the final output read sets with 10, 30, 50, 100, and 200 $\times$ read coverage values, and they are made by concatenating a different number of R_{NPi}. R_{NEF10}, R_{NEF30}, R_{NEF50}, R_{NEF100}, and R_{NEF200} are the error-free versions of R_{N10}, R_{N30}, R_{N50}, R_{N100}, and R_{N200}; they are generated by concatenating R_{NEFPi}.

Running Germline Variant Calling Tools

Germline mutation calling is done using GATK [100]. GATK is executed multiple times using R_{N10}, R_{N30}, and R_{N50}, and these results are compared with the results for R_{NEF10}, R_{NEF30}, and R_{NEF50} to see the effect of sequencing errors.

Before GATK is executed, the reads are aligned to the reference REF_{IN} using BWA. The alignment output BAM file is then polished using the indel realignment and the base quality recalibration capability of GATK. GATK calls germline variants using the polished BAM file, and the variant calling result is compared with V_N using bcftools [101].

It is also necessary to check whether sequencing errors show the same effect on other variant calling tools. Therefore, the experiments that are done for GATK are repeated using samtools.

6.2.2 Somatic Variant Calling

Preparing Input Reads

The process to generate reads for somatic variant calling is shown in Figure 6.2. A new set of somatic variants V_T are sampled from the COSMIC [102] database and used as inputs to VarSim. Then a new diploid genome sequence REF_T is generated by adding both V_T and V_N to REF_{IN}. REF_T represents the DNA sequence of the tumor samples in the specimen that was used in 6.2.1.

Multiple read sets R_{TPi}, each of which has reads with 1 $\times$ read coverage, are simulated from REF_T using ART. R_{TEFPi}, which are error-free versions

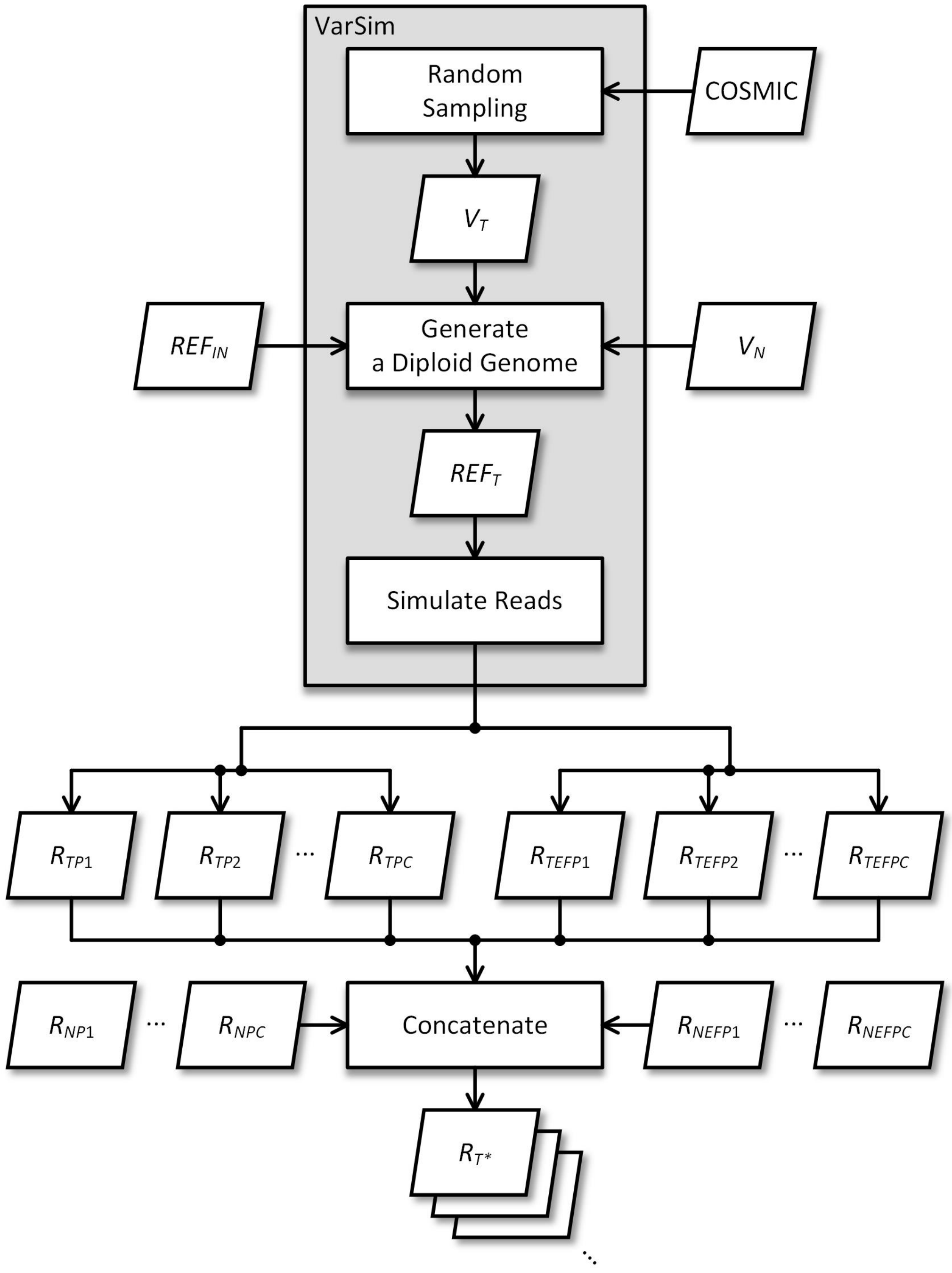


Figure 6.2: Overall flow of generating data sets for somatic variant calling.

of R_{TPi}, are also generated as in the germline variant calling data generation flow.

The final read sets for somatic variant calling are generated by mixing the four read sets R_{NPi}, R_{NEFPi}, R_{TPi}, and R_{TEFPi}. To represent normal tissues and tumor tissues in tumor samples, R_{NPi} and R_{TPi} are mixed at a different ratio to generate the final read sets for tumor samples. In a similar way, the error-free version of the read sets for tumor samples can be generated by mixing R_{NEFPi} and R_{TEFPi}.

Running Somatic Variant Calling Tools

MuTect [103] is used to find somatic variants. MuTect is executed multiple times using read sets with different coverage values and different fractions of tumor reads, and these results are compared with the results for their error-free versions to see the effect of sequencing errors.

Before MuTect is executed, as in the germline variant calling flow, the reads are aligned to the reference REF_{IN} using BWA. The alignment output BAM file is then polished using the indel realignment and the base quality recalibration capability of GATK. MuTect calls somatic variants using the polished BAM file, and the variant calling result is compared with V_T using bcftools.

It is also necessary to check whether sequencing errors show the same effect on other variant calling tools. Therefore, the experiments that are done for MuTect are repeated using Strelka [104].

6.3 Results

6.3.1 Data Preparation

It takes a great deal of time to prepare multiple read sets for the entire human genome and to run variant calling tools for them. Hence, all the experiments were done using one chromosome of the human genome instead of using the entire human genome. Chromosome 22 in GRCh37 was used as REF_{IN}. For germline variants, dbSNP 138 and DGV 2013-07-23 were used as the input variant database. Somatic variations were randomly sampled from COSMIC

V72. In these experiments, only SNVs were investigated despite small indels and SVs also being inserted when a new diploid genome was made.

All the preliminary read sets R_{NPi}, R_{NEFPi}, R_{TPi}, and R_{TEFPi} were generated using VarSim 0.5.1 and ART 03.19.15. Input read sets for the experiments were made by mixing the above four read sets at a different ratio. A recent study showed that the ratio of tumor cells in biopsy samples could be 7-87 percent [105]. Based on the results, four reads sets, the tumor cell ratios of which were 20, 50, 80, and 100 percent were generated. The final read sets for germline variant calling and somatic variant calling are summarized in Table 6.1 and Table 6.2.

Table 6.1: Input read sets for germline variant calling.

ID	Type	Coverage (X)
R_{N10}	10	Original
R_{N30}	30	Original
R_{N50}	50	Original
R_{N100}	100	Original
R_{N200}	200	Original
R_{NEF10}	10	Error-free
R_{NEF30}	30	Error-free
R_{NEF50}	50	Error-free
R_{NEF100}	100	Error-free
R_{NEF200}	200	Error-free

6.3.2 Effect of Sequencing Errors on Germline Variant Calling

GATK results for the original reads R_{N10}, R_{N30}, and R_{N50} were compared with those for the matched error-free reads R_{NEF10}, R_{NEF30}, and R_{NEF50}, and they are summarized in Figure 6.3. TP, FP, and FN mean true positives, false positives, and false negatives. When read coverage is 10 $\times$, true positives increased by 11.3 percent, from R_{N10} to R_{NEF10}. When read coverage was 30 $\times$ and 50 $\times$, however, the difference in true positives was just 1.6 and 0.9 percent. The ratios for R_{N30} and R_{N50} do not look significant but the numbers of the true positives that were newly detected by using error-free reads were 6,692; 952; and 562 for R_{N10}, R_{N30}, and R_{N50}, respectively. The input reads were generated only using chromosome 22, and had the en-

Table 6.2: Input read sets for somatic variant calling.

ID	Normal Tissue		Tumor Tissue	
	Type	Coverage (X)	Type	Coverage (X)
$R_{NNI000} - R_{TO010}$	Not Included	0	Original	10
$R_{NO002} - R_{TO008}$	Original	2	Original	8
$R_{NO005} - R_{TO005}$	Original	5	Original	5
$R_{NO008} - R_{TO002}$	Original	8	Original	2
$R_{NNI000} - R_{TEF010}$	Not Included	0	Error-free	10
$R_{NEF002} - R_{TEF008}$	Error-free	2	Error-free	8
$R_{NEF005} - R_{TEF005}$	Error-free	5	Error-free	5
$R_{NEF008} - R_{TEF002}$	Error-free	8	Error-free	2
$R_{NEF008} - R_{TO002}$	Error-free	8	Original	2
$R_{NO008} - R_{TEF002}$	Original	8	Error-free	2
$R_{NNI000} - R_{TO002}$	Not Included	0	Original	2
$R_{NNI000} - R_{TEF002}$	Not Included	0	Error-free	2
$R_{NNI000} - R_{TO030}$	Not Included	0	Original	30
$R_{NO006} - R_{TO024}$	Original	6	Original	24
$R_{NO015} - R_{TO015}$	Original	15	Original	15
$R_{NO024} - R_{TO006}$	Original	24	Original	6
$R_{NNI000} - R_{TEF030}$	Not Included	0	Error-free	30
$R_{NEF006} - R_{TEF024}$	Error-free	6	Error-free	24
$R_{NEF015} - R_{TEF015}$	Error-free	15	Error-free	15
$R_{NEF024} - R_{TEF006}$	Error-free	24	Error-free	6
$R_{NEF024} - R_{TO006}$	Error-free	24	Original	6
$R_{NO024} - R_{TEF006}$	Original	24	Error-free	6
$R_{NNI000} - R_{TO006}$	Not Included	0	Original	6
$R_{NNI000} - R_{TEF006}$	Not Included	0	Error-free	6
$R_{NNI000} - R_{TO050}$	Not Included	0	Original	50
$R_{NO010} - R_{TO040}$	Original	10	Original	40
$R_{NO025} - R_{TO025}$	Original	25	Original	25
$R_{NO040} - R_{TO010}$	Original	40	Original	10
$R_{NNI000} - R_{TEF050}$	Not Included	0	Error-free	50
$R_{NEF010} - R_{TEF040}$	Error-free	10	Error-free	40
$R_{NEF025} - R_{TEF025}$	Error-free	25	Error-free	25
$R_{NEF040} - R_{TEF010}$	Error-free	40	Error-free	10
$R_{NEF040} - R_{TO010}$	Error-free	40	Original	10
$R_{NO040} - R_{TEF010}$	Original	40	Error-free	10
$R_{NNI000} - R_{TO010}$	Not Included	0	Original	10
$R_{NNI000} - R_{TEF010}$	Not Included	0	Error-free	10
$R_{N080} - R_{TO020}$	Original	80	Original	20
$R_{NNI000} - R_{TO020}$	Not Included	0	Original	20
$R_{NNI000} - R_{TEF020}$	Not Included	0	Error-free	20
$R_{N160} - R_{TO040}$	Original	160	Original	40
$R_{NNI000} - R_{TO040}$	Not Included	0	Original	40
$R_{NNI000} - R_{TEF040}$	Not Included	0	Error-free	40

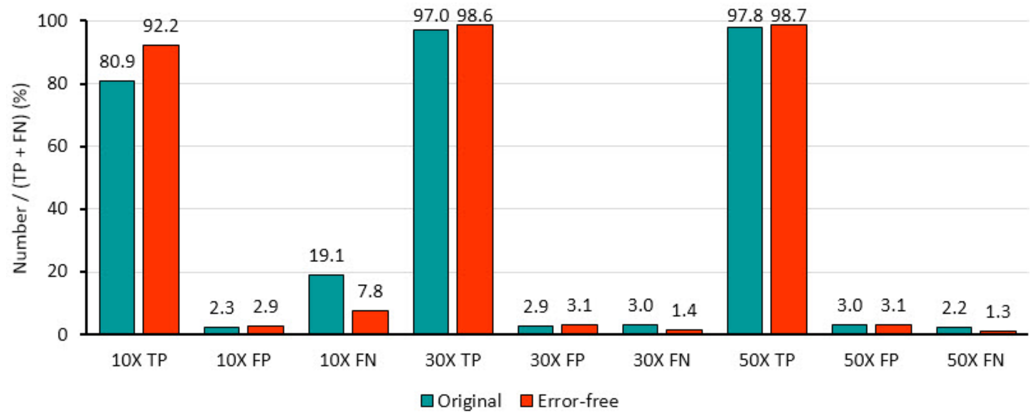


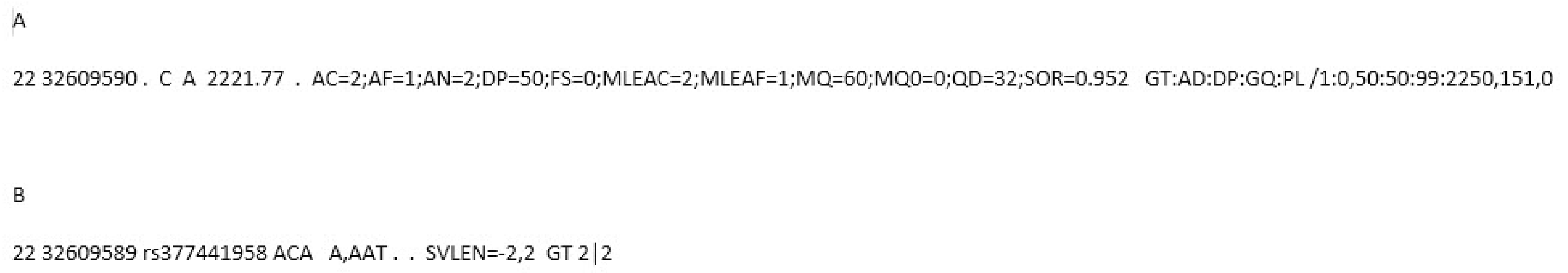
Figure 6.3: Germline variant calling results from GATK.

tire human genome been used the number of newly detected true positives would have been much larger. Figure 6.3 shows that the number of false positives slightly increased when sequencing errors were removed from the reads, regardless of read coverage. Many of the newly generated false positives were falsely reported because of the limitation of bcftools that was used to compare two variant calling results.

One example of false positives that GATK reports is shown in Figure 6.4. GATK found a false positive variant at the 32,609,590th base of chromosome 22 (Figure 6.4A). This variant was reported as a false positive because the same variant does not exist in V_N. However, V_N contains the variant in a different form, as shown in Figure 6.4B. The specimen has a heterozygous variant from the 32,609,590th base to the 32,609,591st base. One chromosome of the diploid genome has a deletion CA from the 32,609,590th base to the 32,609,591st base, and the other chromosome has SNVs at those bases. Because one of the chromosomes has an SNV at the 32,609,590th base, the variant in Figure 6.4A is not a false positive but a true positive. It was, nonetheless recognized as a false positive because the same variant was described as a variant at the 32,609,589th base in V_N.

The same experiment was repeated using samtools, and the results are summarized in Figure 6.5. Samtools was much less sensitive to sequencing errors than GATK. The numbers of true positives increased by 672, 8, and 22, for 10 X, 30 ×, and 50 × read coverage, when R_{NEF10}, R_{NEF30}, and R_{NEF50} were used instead of R_{N10}, R_{N30}, and R_{N50}. There were almost no changes in the number of false positives.

The effect of sequencing errors on GATK results for extremely high cover-


A

```
22 32609590 .  C  A  2221.77  .  AC=2;AF=1;AN=2;DP=50;FS=0;MLEAC=2;MLEAF=1;MQ=60;MQ0=0;QD=32;SOR=0.952   GT:AD:DP:GQ:PL /1:0,50:50:99:2250,151,0
```

B

```
22 32609589 rs377441958 ACA   A,AAT .  .  SVLEN=-2,2  GT 2|2
```

Figure 6.4: Example of wrongly reported false positives.

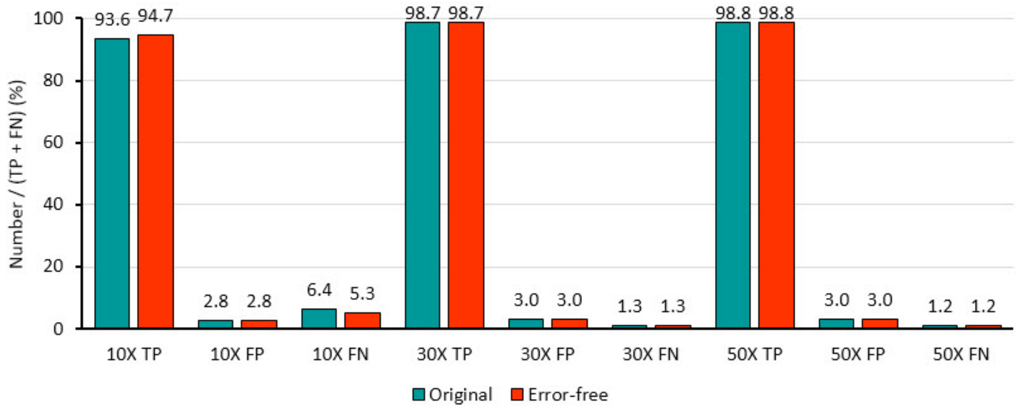


Figure 6.5: Germline variant calling results from samtools.

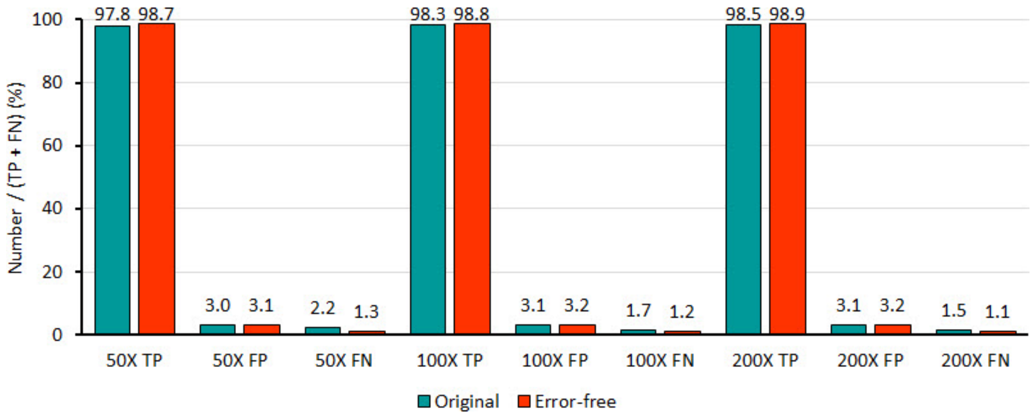


Figure 6.6: Germline variant calling results from GATK for 100 $\times$ and 200 $\times$ coverage read sets.

age reads is summarized in Figure 6.6. When read coverage was 100 $\times$, 0.5 percent of false negatives could be removed by removing sequencing errors in input reads. When read coverage increased from 100 $\times$ to 200 $\times$, the fraction of false negatives was improved by 0.4 percent. For 200 $\times$ coverage, 0.4 percent of false negatives could be still improved by removing sequencing errors.

6.3.3 Effect of Sequencing Errors on Somatic Variant Calling

Figure 6.7 shows how MuTect results were changed by removing sequencing errors when read coverage and the ratio between normal cells and tumor cells in the tumor samples varied. The input reads used in Figure 6.7 consist of the three read sets: reads from the normal sample, reads from normal cells in the tumor sample, and reads from tumor cells in the tumor samples. Figure 6.7A depicts the results for 10 $\times$ read coverage. When the read coverage was just 10 $\times$, the percentage of true positives increased regardless of the fraction of tumor cells in the tumor sample.

On the other hand, for 30 $\times$ of read coverage (Figure 6.7B), the percentage of true positives increased only when the fraction of tumor cells in the tumor samples was low. When the fraction of tumor cells was 100 percent, there was no improvement in the percentage of true positives. The improvement was just 0.5 percent when the fraction was 80 percent. If the fraction decreased to 20 percent, the improvement of true positives became 6.2 percent.

A similar tendency was shown in 50 $\times$ coverage as shown in Figure 6.7C. The number of true positives changed little even when the fraction of tumor cells was 50 percent or higher. However, when the fraction was 20 percent, the percentage of true positives improved from 70.3 percent to 79.2 percent.

As explained above, the input reads used in Figure 6.7 consist of the three read sets (i.e., reads from the normal sample, reads from normal cells in the tumor sample, and reads from tumor cells in the tumor samples). Additional experiments were done to find out which read set was sensitive to sequencing errors when the read coverage was 50 $\times$. These results are summarized in Figure 6.8. The left chart depicts the MuTect result when a normal sample is R_{NEF50} and a tumor sample is R_{NO40}-R_{TO10}. Similarly, the middle chart lays out the results for the combination of R_{N50} and R_{NEF40}-R_{TO10}. These results

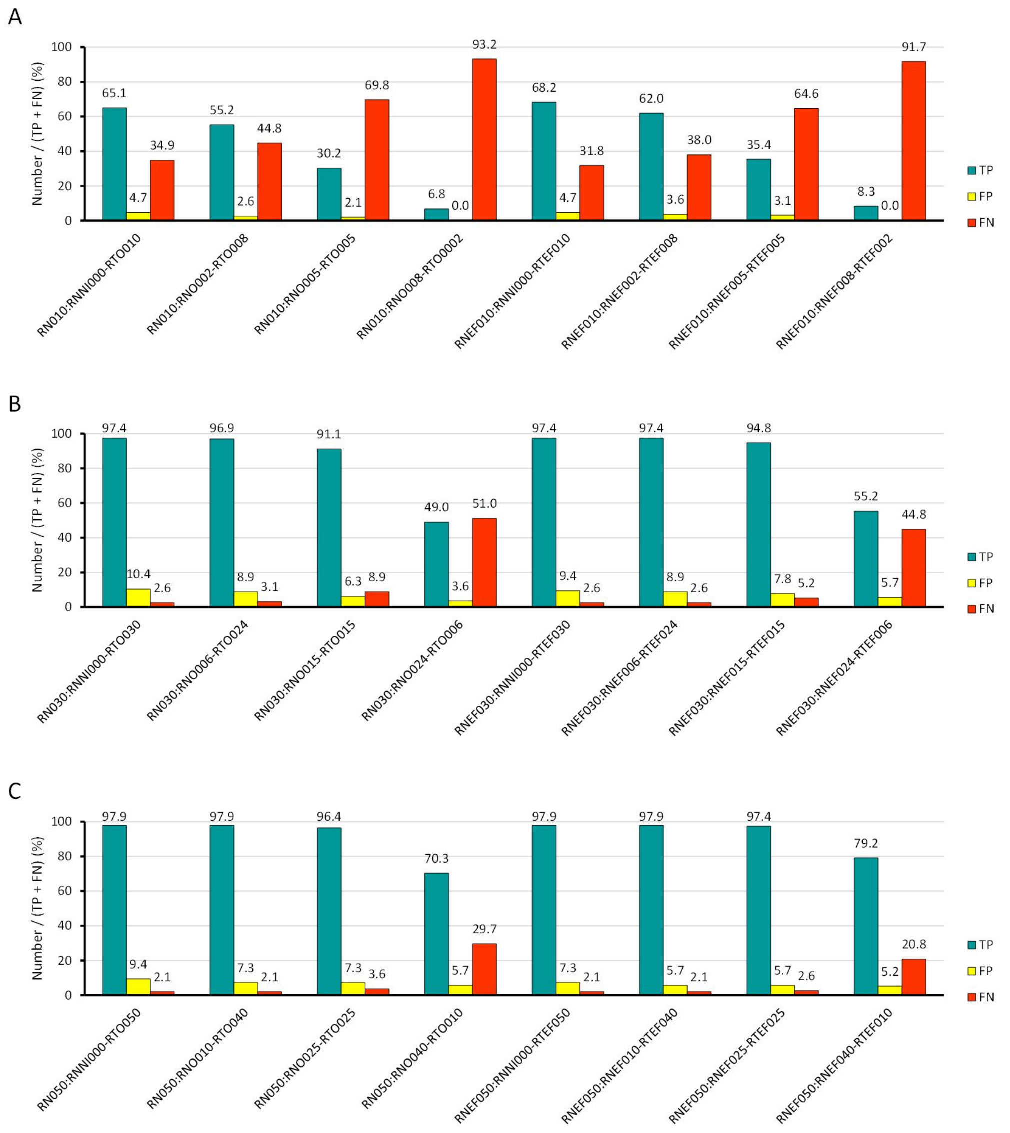


Figure 6.7: Somatic variant calling results from MuTect.

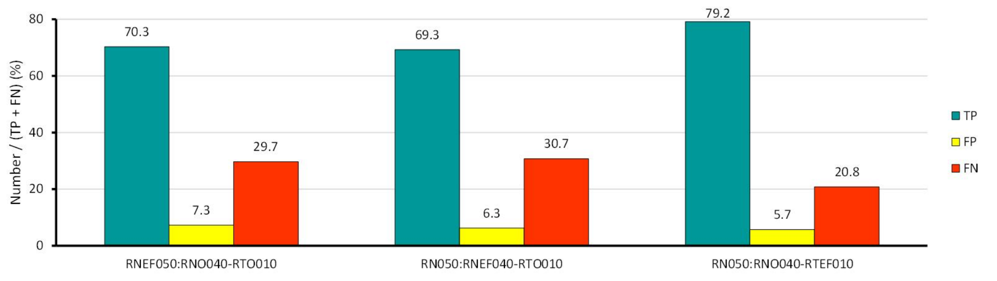


Figure 6.8: MuTect results for 50 × normal sample reads and 50 × tumor sample (40 × normal cells in the tumor sample + 10 × tumor cells in the tumor sample) reads.

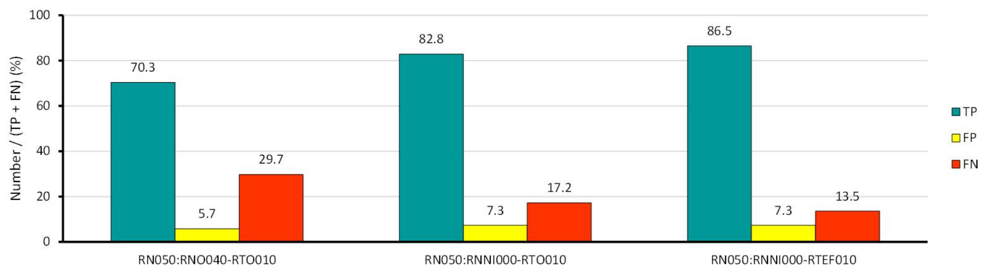


Figure 6.9: Effect of removing normal reads from the tumor sample on MuTect results.

look similar to the result for the combination of R_{N50} and R_{NO40}-R_{TO10} in Figure 6.7C. This means that sequencing errors in the normal sample and normal cells in the tumor sample do not much affect the quality of somatic mutation calling for the read coverage and the tumor cell fraction.

However, when the combination of R_{N50} and R_{NO40}-R_{TEF10} was used, the MuTect result was improved, and it was similar to the result for the combination of R_{NEF50} and R_{NEF40}-R_{TEF10}. Therefore, we can conclude that the improvement that was made in R_{NEF50} and R_{NEF40}-R_{TEF10} was caused by removing errors in reads from tumor cells in the tumor sample.

However, it would be hard to correct sequencing errors in reads from tumor cells in tumor samples when the fraction of the tumor cells is low. While correcting errors in reads from low coverage tumor cells is difficult, it could be relatively easier to identify reads from normal cells in the tumor samples because high coverage reads from normal samples are available. The left chart in Figure 6.9 shows the MuTect result when reads from normal cells were removed from tumor samples, and it shows that removing reads from normal cells can give a better result than correcting errors in reads in tumor cells. The right chart in Figure 6.9 says that the quality of the MuTect results could be improved further by the combination of removing normal cells in tumor samples and correcting sequencing errors in tumor cells in the tumor samples.

The same experiments were repeated using Strelka and the results are shown in Figure 6.10, Figure 6.11, and Figure 6.12. In all the three charts, Strelka showed the same tendency as MuTect. Removing sequencing errors in the reads from tumor cells in tumor samples managed to improve the accuracy of somatic mutation calling when the ratio of tumor cells was low.

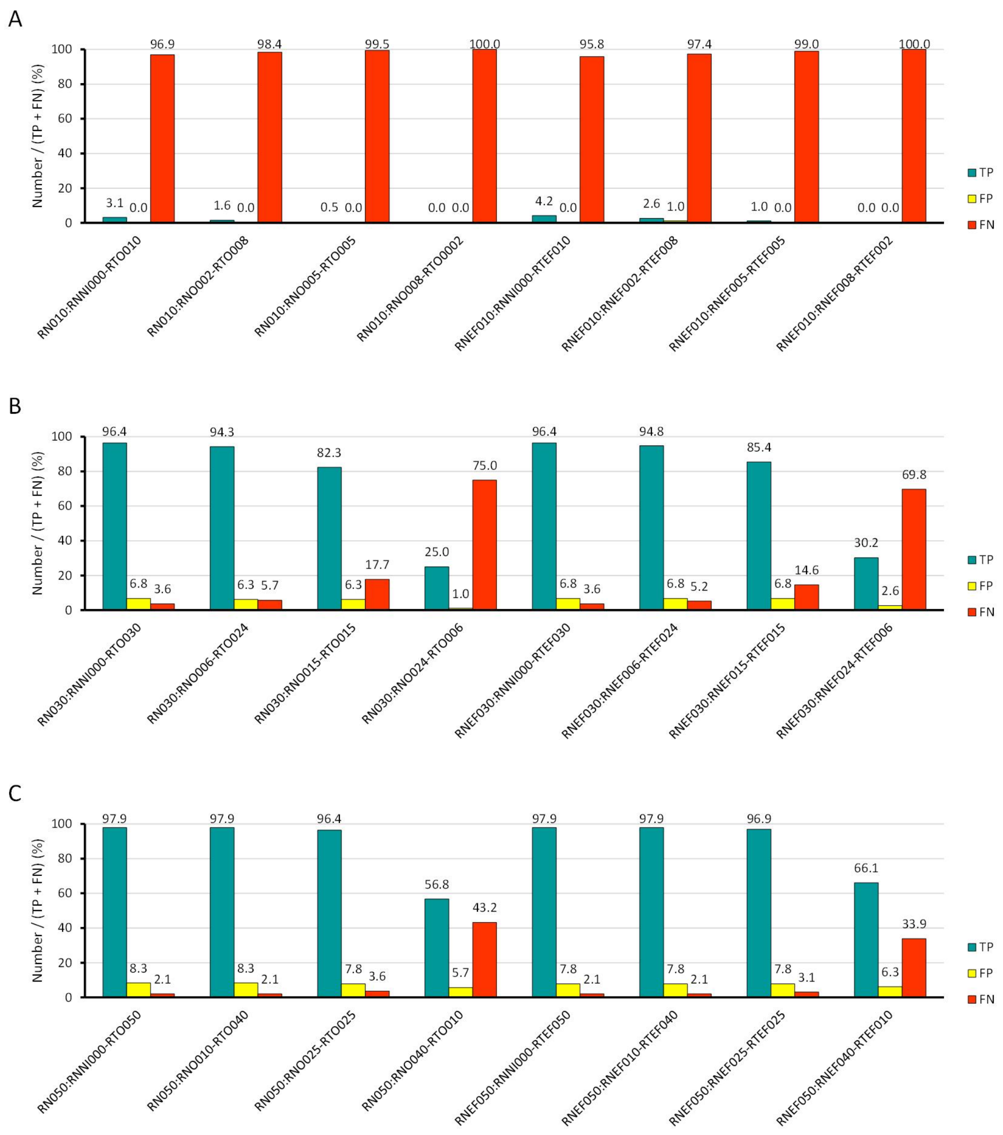


Figure 6.10: Strelka results for different ratio between normal tissues and tumor tissues in tumor samples.

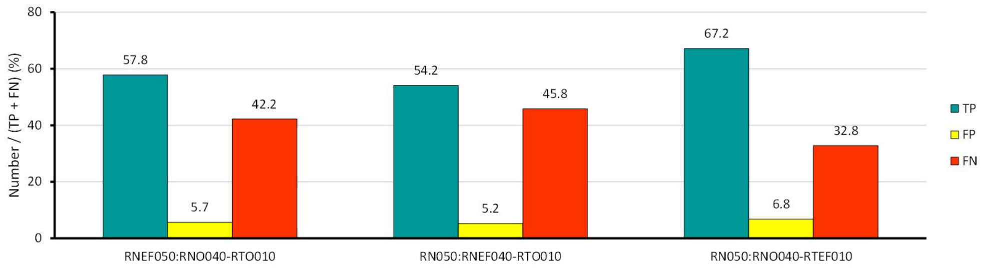


Figure 6.11: Strelka results for different ratio between normal tissues and tumor tissues in tumor samples.

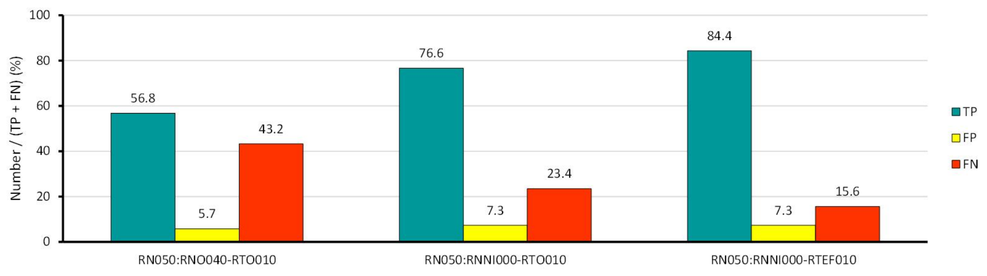


Figure 6.12: Effect of removing normal reads from the tumor sample on Strelka results.

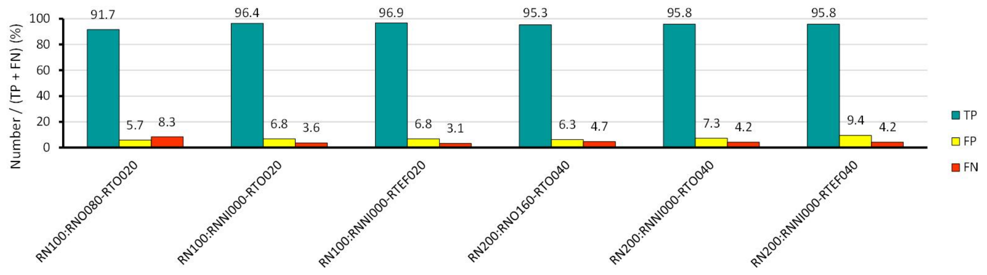


Figure 6.13: Somatic variant calling results from GATK for 100 × and 200 × coverage read sets.

Also, removing reads from normal cells in tumor samples gave a better result than removing the sequencing errors in tumor cells, and the best result could be made by combining the two methods. The only difference is the result for 10 × read coverage. Strelka could call almost no variant regardless of the existence of sequencing errors and the fraction of tumor cells in the tumor samples.

The effect of sequencing errors MuTect results for extremely high read coverage is summarized in Figure 6.13. The left three charts are for 100 × coverage reads and the remaining three are for 200 × reads. Even for 100 × of read coverage, 4.7 percent of false negatives could be removed by removing reads from normal tissues in tumor samples. False negatives could be improved further by removing sequencing errors in reads from tumor tissues in tumor samples.

For 200 × reads, even though the amount of improvement was not as significant as for 100 × reads, false negatives could be reduced using the same way. False negatives dropped by 0.5 percent by removing reads from normal tissues. For 200 × read coverage, removing sequencing errors in reads from tumor tissues could not improve the accuracy of somatic variant calling further.

6.3.4 Effect of Sequencing Errors on Alignment Runtime

A significant amount of time is spent for aligning reads in the variant calling process and the runtime can be reduced by removing sequencing errors. Table 6.3 compares the runtime of BWA for R_{N50} and R_{NEF50}.

Table 6.3: BWA runtime for R_{N50} and R_{NEF50}.

Reads	BWA Runtime (sec)	# of Calls of the Extension Procedure	Average CPU Time of the Extension Procedure (sec)
Original	612	6,000,962	2.8×10^{-4}
Error-free	382	2,735,732	2.4×10^{-4}

The runtime of BWA for R_{NEF50} was 38 percent shorter than that for R_{N50}. BWA first finds super-maximal exact matches between reads and a reference sequence and they are extended using a dynamic programming algorithm that consumes a significant amount of time [106]. When R_{NEF50} were aligned, many reads could be exactly matched with a reference sequence, which reduced the number of calls of the dynamic programming procedure.

In addition, as shown in the fourth column, the average runtime of the dynamic programming procedure became shorter when R_{NEF50} was aligned. This is because longer super-maximal exact matches could be made in R_{NEF50} compared to R_{N50}, and the extension had to be done for shorter sequences.

6.3.5 Improving Germline Variant Calling Accuracy Using BLESS

As shown in Section 6.3.2, germline variant calling results can be improved by removing sequencing errors. Sequencing errors in R_{N50} were corrected using BLESS and the variant calling results for the BLESS output are summarized in Figure 6.14.

R_{N50} is the input of BLESS and the result for R_{NEF50} indicates the best result that can be made by removing sequencing errors in R_{N50}. The difference of the number of false negatives between R_{N50} and R_{NEF50} was 562, and the difference between R_{N50} and BLESS was 490. This result shows that

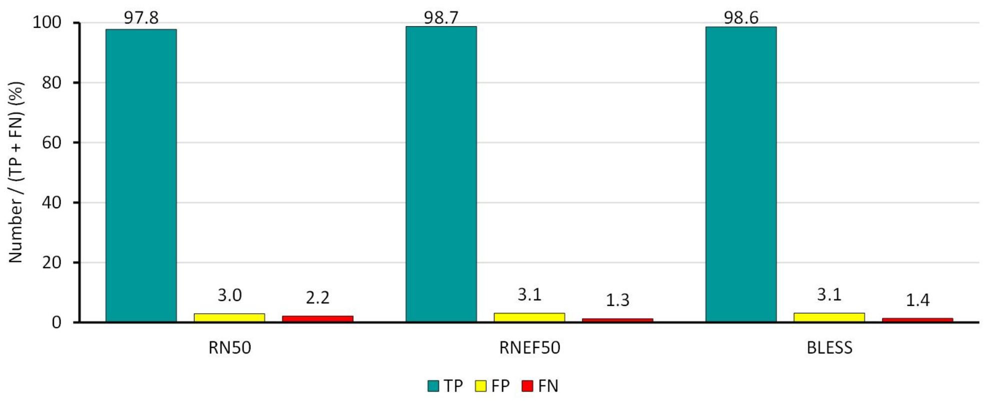


Figure 6.14: Germline variant calling results for BLESS outputs.

BLESS can improve the accuracy of germline variant calling results as high as possible.

6.3.6 Improving Somatic Variant Calling Accuracy Using ROOFTOP

When the fraction of tumor tissues in tumor samples is low, correcting sequencing errors in reads from the tumor tissues can improve the accuracy of somatic variant calling. However, it is hard to correct such errors because reads from normal samples have higher read coverage than reads from tumor tissues. Error correction tools would modify reads from tumor tissues based on the information on reads from normal tissues. It was shown in Section 6.3.3 that removing reads from normal tissues is a good alternative solution for removing errors in reads from tumor tissues. ROOFTOP has been developed to remove normal reads in tumor samples.

Figure 6.15 shows how ROOFTOP works. ROOFTOP tries to filter out reads from normal samples and reads that might have sequencing errors. In order to remove reads from normal samples, ROOFTOP should know which bases normal samples have at a specific location of the genome. To identify this, germline variants V_G in reads from normal samples R_N are found by running a variant calling tool with R_N (Line 1). For each location of the genome, ROOFTOP finds out which bases normal samples have at the position, and the bases are saved in A_{REF}. If a location ref_{index} has germline variants, the variants become A_{REF}. If there is no variant at the location,

```
FIND $V_G$ in $R_N$
FOR $ref\_index$ = 1 to $L_{REF}$
    $COUNT$["A"] = 0
    $COUNT$["C"] = 0
    $COUNT$["G"] = 0
    $COUNT$["T"] = 0
    IF $\{x | x \in V_G, x$ exists at $ref\_index\} \neq 0$
        $A_{REF} = \{x | x \in V_G, x$ exists at $ref\_index\}$
    ELSE
        $A_{REF} = \{REF[ref\_index]\}$
    END IF
    FOREACH $p$ in $P_T[ref\_index]$
        IF $(p \notin A_{REF})$ AND $(Q_p \geq Q_T)$
            $COUNT[p]$++
        END IF
    END FOREACH
    FOREACH $p$ in $P_T[ref\_index]$
        IF $COUNT[p] \geq C_T$
            IF $READ_p$ was not written to $FILE_{OUT}$
                WRITE $READ_p$ to $FILE_{OUT}$
            END IF
        END IF
    END FOREACH
END FOR
```

Figure 6.15: The overall procedure of ROOFTOP.

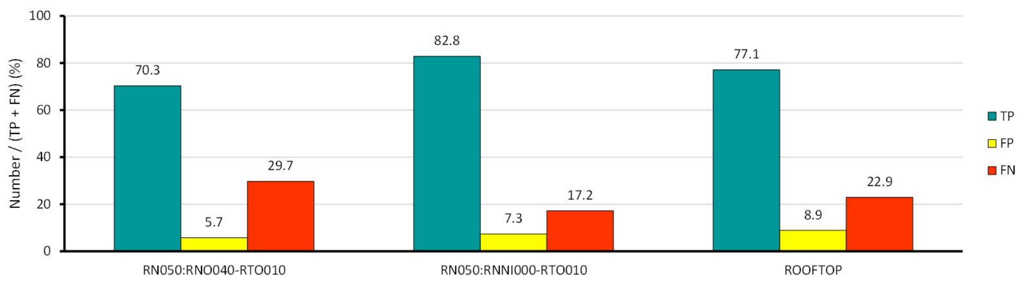


Figure 6.16: Accuracy of ROOFTOP outputs for the inputs from VarSim.

the base in REF at the location $REF[ref_{index}]$ becomes A_{REF} (Line 7-11).

After A_{REF} is found, each base p in $P_T[ref_{index}]$ that is the tumor sample pileup data at ref_{index} is compared with bases in A_{REF}. If p is not included in A_{REF} and its quality score is higher than Q_T, the multiplicity of p is incremented by one. Checking quality scores is needed because erroneous bases in normal samples could also be different from the bases in A_{REF}. After this process is finished, we can know the multiplicity of high-quality bases in the pileup data at the position (Line 12-16).

Then, the multiplicity of each p is checked. If the multiplicity of p is very low, they are thought of as erroneous bases. If the multiplicity of p is higher than a threshold C_T, $READ_p$, a read including p is regarded as a tumor read and it is written to an output file (Line 17-23). In all the following experiments, Q_T and C_T were set to 20 and 3.

The performance of ROOFTOP was compared with the results in Figure 6.8, and the comparison results are summarized in Figure 6.16. Read coverage of the input reads is 50 $\times$, and only 10 $\times$ among them are reads from tumor tissues. When the 40 $\times$ of reads from normal tissues were removed the fraction of false negatives was reduced to 17.2 percent, which is the best result that we can get by removing reads from normal tissues. When ROOFTOP was applied to the input, the fraction of false negatives was reduced from 29.7 percent to 22.9 percent. On the other hand, ROOFTOP only increased the number of false positives by three when it was compared with the best result, which corresponds to 1.6 percent of increase in false positives from the second chart to the third one.

In order to evaluate ROOFTOP with more realistic inputs, the same experiments were repeated using IS1, one of the read sets used in the International Cancer Genome Consortium (ICGC)-The Cancer Genome Atlas (TCGA) Di-

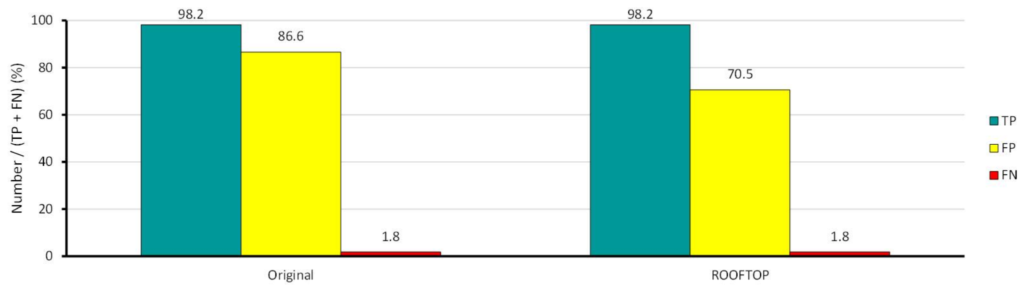


Figure 6.17: Accuracy of ROOFTOP outputs for the ICGC-TCGA DREAM Somatic Mutation Calling Challenge input.

alogue for Reverse Engineering Assessments and Methods (DREAM) Somatic Mutation Calling Challenge [107]. Among the input BAM files, the reads that were aligned to chromosome 22 were used to the experiments. These experimental results are summarized in Figure 6.17. Unlike the results for the inputs from VarSim, MuTect generated many false positives and the fraction of false negatives was negligible (left chart in Figure 6.17). ROOFTOP could improve the accuracy of somatic variant calling results and the fraction of false positives was reduced from 86.6 percent to 70.5 percent (right chart in Figure 6.17). The original MuTect results had only two false negatives and they were not reduced further even after ROOFTOP was applied.

CHAPTER 7

CONCLUSIONS

This dissertation has studied why sequencing errors occur in HTS data, how they can be corrected, and how corrected reads are evaluated. It has also studied how to analyze the effect of sequencing errors on clinical applications and how to improve them.

Sequencing errors in each HTS technology have different characteristics. The major error type in the most popular sequencing platform—Illumina—is substitution errors. Substitution errors are usually corrected using the k-mer spectrum based algorithm. This algorithm requires a large amount of memory to save solid k-mers, which was the obstacle to correcting errors in reads from large genomes using conventional servers. To address this limitation, BLESS was developed. BLESS could correct errors in Illumina reads with higher accuracy while using only 2.5 percent of the memory usage that existing tools used. BLESS was also accelerated using hybrid OpenMP and MPI programming, and it became the fastest error correction method by using more than one computing node.

To compare the accuracy of the different error correction algorithms, a novel software package called SPECTACLE has been developed. It can quantitatively assess the accuracy of corrected reads regardless of sequencing technologies. It can also differentiate heterozygous alleles from sequencing errors, which gives more accurate evaluation results than previous ones.

Finally, the effect of sequencing errors on variant calling has been studied. It was investigated how sequencing errors affect germline variant calling and somatic variant calling results and how sequencing data can be modified to improve the quality of variant calling results.

Taken together, the studies provide a source of understanding of the characteristics of sequencing errors and of correcting errors in sequencing data. The research in Chapter 6 in particular could be adapted to other applications to develop a new way for improving the quality of the application outputs.

Even though many different aspects of sequencing errors have been studied in Chapter 5, it will be extended in different ways. First, not discussed here were correcting errors in some sequencing technologies such as 454, Ion Torrent, and Oxford Nanopore. Since dominant error models in the sequencing data are indels, the k-mer spectrum based algorithm in BLESS is not the best method of correcting such errors. Also, the characteristics of errors in each platform are not equal. For example, indels mainly happen in homopolymers 454 and Ion Torrent, but the errors in PacBio reads are known to exist at random positions. Therefore, correcting the errors in each platform calls for totally different algorithms.

It would also be desirable to study how sequencing errors degrade the quality of detection for other types of variants such as small indels and SVs, because these variants also play an important role in many diseases. The size of some SVs could be longer than the Illumina read length. It would thus be necessary to combine this study with research on errors in the sequencing platforms that can generate much longer reads than Illumina.

REFERENCES

[1] N. Loman, R. Misra, T. Dallman, C. Constantinidou, S. Gharbia, J. Wain, and M. Pallen, "Performance comparison of benchtop high-throughput sequencing platforms," *Nature Biotechnology*, vol. 30, no. 5, pp. 434–439, 2012.

[2] X. Wang, N. Blades, J. Ding, R. Sultana, and G. Parmigiani, "Estimation of sequencing error rates in short reads," *BMC Bioinformatics*, vol. 13, no. 1, p. 185, 2012.

[3] S. S. Ajay, S. C. Parker, H. O. Abaan, K. V. F. Fajardo, and E. H. Margulies, "Accurate and comprehensive sequencing of personal genomes," *Genome Research*, vol. 21, no. 9, pp. 1498–1505, 2011.

[4] K. Robasky, N. Lewis, and G. Church, "The role of replicates for error mitigation in next-generation sequencing," *Nat Rev Genet*, vol. 15, no. 1, pp. 56–62, 2014.

[5] X. Yang, S. Chockalingam, and S. Aluru, "A survey of error-correction methods for next-generation sequencing," *Briefings in Bioinformatics*, vol. 14, no. 1, pp. 56–66, 2013.

[6] M. Chaisson, D. Brinza, and P. Pevzner, "De novo fragment assembly with short mate-paired reads: Does the read length matter?" *Genome Research*, vol. 19, no. 2, pp. 336–346, 2009.

[7] J. Dohm, C. Lottaz, T. Borodina, and H. Himmelbauer, "Substantial biases in ultra-short read data sets from high-throughput DNA sequencing," *Nucleic Acids Research*, vol. 36, no. 16, p. e105, 2008.

[8] D. Kelley, M. Schatz, and S. Salzberg, "Quake: Quality-aware detection and correction of sequencing errors," *Genome Biology*, vol. 11, no. 11, p. R116, 2010.

[9] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, and J. Wang, "De novo assembly of human genomes with massively parallel short read sequencing," *Genome Research*, vol. 20, no. 2, pp. 265–272, 2010.

[10] Y. Liu, J. Schröder, and B. Schmidt, "Musket: A multistage k-mer spectrum-based error corrector for Illumina sequence data," *Bioinformatics*, vol. 29, no. 3, pp. 308–315, 2013.

[11] P. Medvedev, E. Scott, B. Kakaradov, and P. Pevzner, "Error correction of high-throughput sequencing datasets with non-uniform coverage," *Bioinformatics*, vol. 27, no. 13, pp. i137–i141, 2011.

[12] P. A. Pevzner, H. Tang, and M. S. Waterman, "An Eulerian path approach to DNA fragment assembly," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 17, pp. 9748–9753, 2001.

[13] W. Qu, S.-i. Hashimoto, and S. Morishita, "Efficient frequency-based de novo short-read clustering for error trimming in next-generation sequencing," *Genome Research*, vol. 19, no. 7, pp. 1309–1315, 2009.

[14] E. Wijaya, M. C. Frith, Y. Suzuki, and P. Horton, "Recount: Expectation maximization based error correction tool for next generation sequencing data," *Genome Inform*, vol. 23, no. 1, pp. 189–201, 2009.

[15] X. Yang, S. Aluru, and K. Dorman, "Repeat-aware modeling and correction of short read errors," *BMC Bioinformatics*, vol. 12, no. Suppl 1, p. S52, 2011.

[16] X. Yang, K. Dorman, and S. Aluru, "Reptile: Representative tiling for short read error correction," *Bioinformatics*, vol. 26, no. 20, pp. 2526–2533, 2010.

[17] L. Ilie, F. Fazayeli, and S. Ilie, "HiTEC: Accurate error correction in high-throughput sequencing data," *Bioinformatics*, vol. 27, no. 3, pp. 295–302, 2011.

[18] L. Salmela, "Correction of sequencing errors in a mixed set of reads," *Bioinformatics*, vol. 26, no. 10, pp. 1284–1290, 2010.

[19] J. Schroder, H. Schroder, S. Puglisi, R. Sinha, and B. Schmidt, "SHREC: A short-read error correction method," *Bioinformatics*, vol. 25, no. 17, pp. 2157–2163, 2009.

[20] Z. Zhao, J. Yin, Y. Zhan, W. Xiong, Y. Li, and F. Liu, "PSAEC: An improved algorithm for short read error correction using partial suffix arrays," in *Proceedings of the 5th Joint International Frontiers in Algorithmics, and 7th International Conference on Algorithmic Aspects in Information and Management*, Conference Proceedings. Springer-Verlag, pp. 220–232.

[21] W.-C. Kao, A. Chan, and Y. Song, "ECHO: A reference-free short-read error correction algorithm," *Genome Research*, vol. 21, no. 7, pp. 1181–1192, 2011.

[22] L. Salmela and J. Schroder, "Correcting errors in short reads by multiple alignments," *Bioinformatics*, vol. 27, no. 11, pp. 1455–1461, 2011.

[23] H.-S. Le, M. Schulz, B. McCauley, V. Hinman, and Z. Bar-Joseph, "Probabilistic error correction for RNA sequencing," *Nucleic Acids Research*, vol. 41, no. 10, p. e109, 2013.

[24] X. Yin, Z. Song, K. Dorman, and A. Ramamoorthy, "PREMIER—Probabilistic error-correction using Markov inference in errored reads," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, Conference Proceedings, pp. 1626–1630.

[25] M. Metzker, "Sequencing technologies—The next generation," *Nature Reviews Genetics*, vol. 11, no. 1, pp. 31–46, 2009.

[26] M. D. Cao, S. Balasubramanian, and M. Boden, "Sequencing technologies and tools for short tandem repeat variation detection," *Briefings in Bioinformatics*, p. bbu001, 2014.

[27] M. Kircher and J. Kelso, "High-throughput DNA sequencing—Concepts and limitations," *Bioessays*, vol. 32, no. 6, pp. 524–536, 2010.

[28] S. Koren, M. Schatz, B. Walenz, J. Martin, J. Howard, G. Ganapathy, Z. Wang, D. Rasko, R. McCombie, E. Jarvis, and A. Phillippy, "Hybrid error correction and de novo assembly of single-molecule sequencing reads," *Nature Biotechnology*, vol. 30, no. 7, pp. 693–700, 2012.

[29] D. Laehnemann, A. Borkhardt, and A. C. McHardy, "Denoising DNA deep sequencing data—High-throughput sequencing errors and their correction," *Briefings in Bioinformatics*, p. bbv029, 2015.

[30] A. S. Mikheyev and M. M. Tin, "A first look at the Oxford Nanopore MinION sequencer," *Molecular ecology resources*, vol. 14, no. 6, pp. 1097–1102, 2014.

[31] D. Haussler, S. J. O'Brien, O. A. Ryder, F. K. Barker, M. Clamp, A. J. Crawford, R. Hanner, O. Hanotte, W. E. Johnson, and J. A. McGuire, "Genome 10K: A proposal to obtain whole-genome sequence for 10 000 vertebrate species," *Journal of Heredity*, vol. 100, no. 6, pp. 659–674, 2009.

[32] K. Frazer, "Decoding the human genome," *Genome Research*, vol. 22, no. 9, pp. 1599–1601, 2012.

[33] N. Beerenwinkel and O. Zagordi, “Ultra-deep sequencing for the analysis of viral populations,” *Current Opinion in Virology*, vol. 1, no. 5, pp. 413–418, 2011.

[34] R. Durbin, D. Altshuler, R. Durbin, G. Abecasis, D. Bentley, A. Chakravarti, A. Clark, F. Collins, F. De La Vega, P. Donnelly, M. Egholm, P. Flicek, S. Gabriel, R. Gibbs, B. Knoppers, E. Lander, H. Lehrach, E. Mardis, G. McVean, D. Nickerson, L. Peltonen, A. Schafer, S. Sherry, J. Wang, R. Wilson, R. Gibbs, D. Deiros, M. Metzker, D. Muzny, J. Reid, D. Wheeler, J. Wang, J. Li, M. Jian, G. Li, R. Li, H. Liang, G. Tian, B. Wang, J. Wang, W. Wang, H. Yang, X. Zhang, H. Zheng, E. Lander, D. Altshuler, L. Ambrogio, T. Bloom, K. Cibulskis, T. Fennell, S. Gabriel, D. Jaffe, E. Shefler, C. Sougnez, D. Bentley, N. Gormley, S. Humphray, Z. Kingsbury, P. Koko-Gonzales, J. Stone, K. McKernan, G. Costa, J. Ichikawa, C. Lee, R. Sudbrak, H. Lehrach, T. Borodina, A. Dahl, A. Davydov, P. Marquardt, F. Mertes, W. Nietfeld, P. Rosenstiel, S. Schreiber, A. Soldatov, B. Timmermann, M. Tolzmann, M. Egholm, J. Affourtit, D. Ashworth, S. Attiya, M. Bachorski, E. Buglione, A. Burke, A. Caprio, C. Celone, S. Clark, D. Conners, B. Desany, L. Gu, L. Guccione, K. Kao, A. Kebbel, J. Knowlton, M. Labrecque, L. McDade, C. Mealmaker, M. Minderman, A. Nawrocki, F. Niazi et al., “A map of human genome variation from population-scale sequencing,” *Nature*, vol. 467, no. 7319, pp. 1061–1073, 2010.

[35] M. Prosperi, L. Yin, D. Nolan, A. Lowe, M. Goodenow, and M. Salemi, “Empirical validation of viral quasispecies assembly algorithms: State-of-the-art and challenges,” *Scientific Reports*, vol. 3, 2013.

[36] M. Schirmer, W. Sloan, and C. Quince, “Benchmarking of viral haplotype reconstruction programmes: An overview of the capacities and limitations of currently available programmes,” *Briefings in Bioinformatics*, 2012.

[37] S. Salzberg, A. Phillippy, A. Zimin, D. Puiu, T. Magoc, S. Koren, T. Treangen, M. Schatz, A. Delcher, M. Roberts, G. Marais, M. Pop, and J. Yorke, “GAGE: A critical evaluation of genome assemblies and assembly algorithms,” *Genome Research*, vol. 22, no. 3, pp. 557–567, 2012.

[38] R. Jiang, S. Tavar, and P. Marjoram, “Population genetic inference from resequencing data,” *Genetics*, vol. 181, no. 1, pp. 187–197, 2009.

[39] S. Sheikhizadeh and D. de Ridder, “ACE: Accurate correction of errors using k-mer tries,” *Bioinformatics*, p. btv332, 2015.

[40] X. Yin, Z. Song, K. Dorman, and A. Ramamoorthy, "PREMIER Turbo: Probabilistic error-correction using Markov inference in errored reads using the turbo principle," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE.* IEEE, 2013, pp. 73–76.

[41] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[42] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, 2000.

[43] Y. Liu, B. Schmidt, and D. Maskell, "DecGPU: Distributed error correction on massively parallel graphics processing units using CUDA and MPI," *BMC Bioinformatics*, vol. 12, no. 1, p. 85, 2011.

[44] H. Shi, B. Schmidt, W. Liu, and W. Muller-Wittig, "A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware," *Journal of Computational Biology*, vol. 17, no. 4, pp. 603–615, 2010.

[45] H. Shi, B. Schmidt, W. Liu, and W. Muller-Wittig, "Quality-score guided error correction for short-read sequencing data using CUDA," *Procedia Computer Science*, vol. 1, no. 1, pp. 1129–1138, 2010.

[46] H. Shi, B. Schmidt, W. Liu, and W. Muller-Wittig, "Accelerating error correction in high-throughput short-read DNA sequencing data with CUDA," in *Parallel and Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, Conference Proceedings. IEEE, pp. 1–8.

[47] S. Deorowicz, A. Grabysz, and S. Grabowski, "Disk-based k-mer counting on a PC," *BMC Bioinformatics*, vol. 14, no. 1, p. 160, 2013.

[48] G. Marcais and C. Kingsford, "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers," *Bioinformatics*, vol. 27, no. 6, pp. 764–770, 2011.

[49] P. Melsted and J. Pritchard, "Efficient counting of k-mers in DNA sequences using a Bloom filter," *BMC Bioinformatics*, vol. 12, no. 1, p. 333, 2011.

[50] G. Rizk, D. Lavenier, and R. Chikhi, "DSK: k-mer counting with very low memory usage," *Bioinformatics*, vol. 29, no. 5, pp. 652–653, 2013.

[51] R. Roy, D. Bhattacharya, and A. Schliep, "Turtle: Identifying frequent k-mers with cache-efficient algorithms," *Bioinformatics*, 2014.

[52] D. Zerbino and E. Birney, “Velvet: Algorithms for de novo short read assembly using de Bruijn graphs,” *Genome Research*, vol. 18, no. 5, pp. 821–829, 2008.

[53] “Bloom Filter Library,” https://github.com/ArashPartow/bloom, Accessed: 2013-01-01.

[54] “simNGS and simLibrary — Software for Simulating Next-Gen Sequencing Data,” http://www.ebi.ac.uk/goldman-srv/simNGS, Accessed: 2013-01-01.

[55] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows-Wheeler transform,” *Bioinformatics*, vol. 25, no. 14, pp. 1754–1760, 2009.

[56] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg, “Ultrafast and memory-efficient alignment of short DNA sequences to the human genome,” *Genome Biology*, vol. 10, no. 3, pp. R25–10, 2009.

[57] J. Simpson and R. Durbin, “Efficient de novo assembly of large genomes using compressed data structures,” *Genome Research*, vol. 22, no. 3, pp. gr.126 953.111–556, 2011.

[58] D. Earl, K. Bradnam, J. St John, A. Darling, D. Lin, J. Fass, H. O. K. Yu, V. Buffalo, D. Zerbino, M. Diekhans, N. Nguyen, P. N. Ariyaratne, W.-K. Sung, Z. Ning, M. Haimel, J. Simpson, N. Fonseca, I. Birol, R. Docking, I. Ho, D. Rokhsar, R. Chikhi, D. Lavenier, G. Chapuis, D. Naquin, N. Maillet, M. Schatz, D. Kelley, A. Phillippy, S. Koren, S.-P. Yang, W. Wu, W.-C. Chou, A. Srivastava, T. Shaw, G. Ruby, P. Skewes-Cox, M. Betegon, M. Dimon, V. Solovyev, I. Seledtsov, P. Kosarev, D. Vorobyev, R. Ramirez-Gonzalez, R. Leggett, D. MacLean, F. Xia, R. Luo, Z. Li, Y. Xie, B. Liu, S. Gnerre, I. MacCallum, D. Przybylski, F. Ribeiro, S. Yin, T. Sharpe, G. Hall, P. Kersey, R. Durbin, S. Jackman, J. Chapman, X. Huang, J. DeRisi, M. Caccamo, Y. Li, D. Jaffe, R. Green, D. Haussler, I. Korf, and B. Paten, “Assemblathon 1: A competitive assessment of de novo short read assembly methods,” *Genome Research*, vol. 21, no. 12, pp. 2224–2241, 2011.

[59] L. Song, L. Florea, and B. Langmead, “Lighter: Fast and memory-efficient sequencing error correction without counting,” *Genome Biology*, vol. 15, no. 11, p. 509, 2014.

[60] H. Li, “BFC: Correcting Illumina sequencing errors,” *Bioinformatics*, 2015.

[61] S. Deorowicz, M. Kokot, S. Grabowski, and A. Debudaj-Grabysz, "KMC 2: Fast and resource-frugal k-mer counting," *Bioinformatics*, p. btv022, 2015.

[62] "Platinum Genomes," http://www.illumina.com/platinumgenomes, Accessed: 2015-05-10.

[63] "SPECTACLE," http://thirtyeggs.github.io/spectacle, Accessed: 2015-06-05.

[64] G. Marcais, J. A. Yorke, and A. Zimin, "QuorUM: An error corrector for Illumina reads," *arXiv preprint arXiv:1307.3515*, 2013.

[65] "HiSeq X System Specifications," http://www.illumina.com/systems/hiseq-x-sequencing-system/performance-specifications.html, Accessed: 2014-10-10.

[66] M. S. Fujimoto, P. M. Bodily, N. Okuda, M. J. Clement, and Q. Snell, "Effects of error-correction of heterozygous next-generation sequencing data," *BMC Bioinformatics*, vol. 15, no. Suppl 7, p. S3, 2014.

[67] M. D. MacManes and M. B. Eisen, "Improving transcriptome assembly through error correction of high-throughput sequence reads," *PeerJ*, vol. 1, p. e113, 2013.

[68] P. Greenfield, K. Duesing, A. Papanicolaou, and D. C. Bauer, "Blue: Correcting sequencing errors using consensus and context," *Bioinformatics*, p. btu368, 2014.

[69] Y. Heo, X.-L. Wu, D. Chen, J. Ma, and W.-M. Hwu, "BLESS: Bloom filter-based error correction solution for high-throughput sequencing reads," *Bioinformatics*, vol. 30, no. 10, pp. 1354–1362, 2014.

[70] L. Ilie and M. Molnar, "RACER: Rapid and accurate correction of errors in reads," *Bioinformatics*, vol. 29, no. 19, pp. 2490–2493, 2013.

[71] S. Gnerre, I. MacCallum, D. Przybylski, F. Ribeiro, J. Burton, B. Walker, T. Sharpe, G. Hall, T. Shea, S. Sykes, A. Berlin, D. Aird, M. Costello, R. Daza, L. Williams, R. Nicol, A. Gnirke, C. Nusbaum, E. Lander, and D. Jaffe, "High-quality draft assemblies of mammalian genomes from massively parallel sequence data," *Proceedings of the National Academy of Sciences*, vol. 108, no. 4, pp. 1513–1518, 2011.

[72] A. Zimin, G. Marcais, D. Puiu, M. Roberts, S. Salzberg, and J. Yorke, "The MaSuRCA genome assembler," *Bioinformatics*, vol. 29, no. 21, pp. 2669–2677, 2013.

[73] Z. Wang, M. Gerstein, and M. Snyder, "RNA-Seq: A revolutionary tool for transcriptomics," *Nature Reviews Genetics*, vol. 10, no. 1, pp. 57–63, 2009.

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

[75] M. Quail, M. Smith, P. Coupland, T. Otto, S. Harris, T. Connor, A. Bertoni, H. Swerdlow, and Y. Gu, "A tale of three next generation sequencing platforms: Comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers," *BMC Genomics*, vol. 13, no. 1, p. 341, 2012.

[76] L. Salmela and E. Rivals, "LoRDEC: Accurate and efficient long read error correction," *Bioinformatics*, p. btu538, 2014.

[77] K. F. Au, J. G. Underwood, L. Lee, and W. H. Wong, "Improving PacBio long read accuracy by short read alignment," *PLoS One*, vol. 7, no. 10, p. e46679, 2012.

[78] T. Hackl, R. Hedrich, J. Schultz, and F. Forster, "proovread: Large-scale high-accuracy PacBio correction through iterative short read consensus," *Bioinformatics*, vol. 30, no. 21, pp. 3004–3011, 2014.

[79] M. H. Schulz, D. Weese, M. Holtgrewe, V. Dimitrova, S. Niu, K. Reinert, and H. Richard, "Fiona: A parallel and automatic strategy for read error correction," *Bioinformatics*, vol. 30, no. 17, pp. i356–63, 2014.

[80] C. Luo, D. Tsementzi, N. Kyrpides, T. Read, and K. T. Konstantinidis, "Direct comparisons of Illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample," *PloS one*, vol. 7, no. 2, p. e30087, 2012.

[81] L. Bragg, G. Stone, M. Butler, P. Hugenholtz, and G. Tyson, "Shining a light on dark sequencing: Characterising errors in Ion Torrent PGM data," *PLoS Comput Biol*, vol. 9, no. 4, p. e1003031, 2013.

[82] M. Molnar and L. Ilie, "Correcting Illumina data," *Briefings in Bioinformatics*, p. bbu029, 2014.

[83] X. Hu, J. Yuan, Y. Shi, J. Lu, B. Liu, Z. Li, Y. Chen, D. Mu, H. Zhang, N. Li, Z. Yue, F. Bai, H. Li, and W. Fan, "pIRS: Profile-based Illumina pair-end reads simulator," *Bioinformatics*, vol. 28, no. 11, pp. 1533–1535, 2012.

[84] Y. Ono, K. Asai, and M. Hamada, "PBSIM: PacBio reads simulator-toward accurate genome assembly," *Bioinformatics*, vol. 29, no. 1, pp. 119–121, 2013.

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

[86] P. Danecek, A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. DePristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T. Sherry, G. McVean, R. Durbin, and 1000 Genomes Project Analysis Group, "The variant call format and VCFtools," *Bioinformatics*, vol. 27, no. 15, pp. 2156–8, 2011.

[87] O. Gotoh, "An improved algorithm for matching biological sequences," *J Mol Biol*, vol. 162, no. 3, pp. 705–8.

[88] C. Yau, D. Mouradov, R. N. Jorissen, S. Colella, G. Mirza, G. Steers, A. Harris, J. Ragoussis, O. Sieber, and C. C. Holmes, "A statistical approach for detecting genomic aberrations in heterogeneous tumor samples from single nucleotide polymorphism genotyping data," *Genome Biol*, vol. 11, no. 9, p. R92, 2010.

[89] B. Haubold and T. Wiehe, "How repetitive are genomes?" *BMC Bioinformatics*, vol. 7, no. 1, p. 541, 2006.

[90] "MiSeq Scientific Data," http://www.illumina.com/systems/miseq/scientific_data.html, Accessed: 2014-10-10.

[91] "*E coli* K12 MG1665 Hybrid Assembly," https://github.com/PacificBiosciences/DevNet/wiki/E%20coli%20K12%20MG1655%20Hybrid%20Assembly, Accessed: 2014-10-10.

[92] E.-C. Lim, J. Muller, J. Hagmann, S. R. Henz, S.-T. Kim, and D. Weigel, "Trowel: A fast and accurate error correction module for Illumina sequencing reads," *Bioinformatics*, vol. 30, no. 22, pp. 3264–3265, 2014.

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

[94] K. Robison, "Application of second-generation sequencing to cancer genomics," *Briefings in Bioinformatics*, p. bbq013, 2010.

[95] J. C. Mu, M. Mohiyuddin, J. Li, N. B. Asadi, M. B. Gerstein, A. Abyzov, W. H. Wong, and H. Y. Lam, "VarSim: A high-fidelity simulation and validation framework for high-throughput genome sequencing with cancer applications," *Bioinformatics*, p. btu828, 2014.

[96] S. T. Sherry, M.-H. Ward, M. Kholodov, J. Baker, L. Phan, E. M. Smigielski, and K. Sirotkin, "dbSNP: The NCBI database of genetic variation," *Nucleic acids research*, vol. 29, no. 1, pp. 308–311, 2001.

[97] J. R. MacDonald, R. Ziman, R. K. Yuen, L. Feuk, and S. W. Scherer, "The database of genomic variants: A curated collection of structural variation in the human genome," *Nucleic Acids Research*, vol. 42, no. D1, pp. D986–D992, 2014.

[98] W. Huang, L. Li, J. Myers, and G. Marth, "ART: A next-generation sequencing read simulator," *Bioinformatics*, vol. 28, no. 4, pp. 593–594, 2012.

[99] "Picard," http://broadinstitute.github.io/picard, Accessed: 2015-01-01.

[100] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M. A. DePristo, "The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data," *Genome Research*, vol. 20, no. 9, pp. 1297–303.

[101] H. Li, "A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data," *Bioinformatics*, vol. 27, no. 21, pp. 2987–2993, 2011.

[102] S. A. Forbes, D. Beare, P. Gunasekaran, K. Leung, N. Bindal, H. Boutselakis, M. Ding, S. Bamford, C. Cole, and S. Ward, "COSMIC: Exploring the world's knowledge of somatic mutations in human cancer," *Nucleic Acids Research*, vol. 43, no. D1, pp. 805–811, 2015.

[103] K. Cibulskis, M. S. Lawrence, S. L. Carter, A. Sivachenko, D. Jaffe, C. Sougnez, S. Gabriel, M. Meyerson, E. S. Lander, and G. Getz, "Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples," *Nature Biotechnology*, vol. 31, no. 3, pp. 213–219, 2013.

[104] C. T. Saunders, W. S. Wong, S. Swamy, J. Becq, L. J. Murray, and R. K. Cheetham, "Strelka: Accurate somatic small-variant calling from sequenced tumor-normal sample pairs," *Bioinformatics*, vol. 28, no. 14, pp. 1811–1817, 2012.

[105] T. Takahashi, Y. Matsuda, S. Yamashita, N. Hattori, R. Kushima, Y.-C. Lee, H. Igaki, Y. Tachimori, M. Nagino, and T. Ushijima, "Estimation of the fraction of cancer cells in a tumor DNA sample using DNA methylation," *PLoS ONE*, vol. 8, no. 12, p. e82302, 2013.

[106] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," *arXiv preprint arXiv:1303.3997*, 2013.

[107] A. D. Ewing, K. E. Houlahan, Y. Hu, K. Ellrott, C. Caloian, T. N. Yamaguchi, J. C. Bare, C. P'ng, D. Waggott, V. Y. Sabelnykova et al., "Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection," *Nature Methods*, 2015.