Identifying and Analyzing Indel Variants in the Human Genome Using Computational Approaches

Mohammad Shabbir Hasan

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

Doctor of Philosophy
in
Computer Science and Applications

Liqing Zhang, Chair
Lenwood S. Heath
Bert Huang
Xiaowei Wu
Xinghua (Mindy) Shi

May 14, 2019
Blacksburg, Virginia

Keywords: Genetic Variants, Indel, Somatic Mutation, Next Generation Sequencing

Identifying and Analyzing Indel Variants in the Human Genome Using Computational Approaches

Mohammad Shabbir Hasan

(ABSTRACT)

Insertion and deletion (indel), a common form of genetic variation, has been shown to cause or contribute to human genetic diseases and cancer. Despite this importance and being the second most abundant variant type in the human genome, indels have not been studied as much as the single nucleotide polymorphism (SNP). With the advance of next-generation sequencing technology, many indel calling tools have been developed. However, performance comparison of commonly used tools has shown that (1) the tools have limited power in identifying indels and there are significant number of indels undetected, and (2) there is significant disagreement among the indel sets produced by the tools. These findings indicate the necessity of improving the existing tools or developing new algorithms to achieve reliable and consistent indel calling results.

Two indels are biologically equivalent if the resulting sequences are the same. Storing biologically equivalent indels as distinct entries in databases causes data redundancy and misleads downstream analysis. It is thus desirable to have a unified system for identifying and representing equivalent indels. This dissertation describes UPS-indel, a utility tool that creates a universal positioning system for indels so that equivalent indels can be uniquely determined by their coordinates in the new system. Results show that UPS-indel identifies more redundant indels than existing algorithms.

While mapping short reads to the reference genome, a significant number of short reads are unmapped and excluded from downstream analyses, thereby causing information loss in the subsequent variant calling. This dissertation describes Genesis-indel, a computational pipeline that explores the unmapped reads to identify missing novel indels. Results analyzing sequence alignment of 30 breast cancer patients show that Genesis-indel identifies many novel indels that also show significant enrichment in oncogenes and tumor suppressor genes, demonstrating the importance of rescuing indels hidden in the unmapped reads in cancer and disease studies.

Somatic mutations play a vital role in transforming healthy cells into cancer cells. Therefore, accurate identification of somatic mutations is essential. Many somatic mutations callers are available with different strengths and weaknesses. An ensemble approach integrating the power of

the callers is warranted. This dissertation describes SomaticHunter, an ensemble of two callers, namely Platypus and VarDict. Results on synthetic tumor data show that for both SNPs and indels, SomaticHunter achieves recall comparable to the state-of-the-art somatic mutation callers and the highest precision, resulting in the highest F1 score.

Identifying and Analyzing Indel Variants in the Human Genome Using Computational Approaches

Mohammad Shabbir Hasan

(GENERAL AUDIENCE ABSTRACT)

Insertion and deletion (indel), a common form of genetic variation in the human genome, is associated with genetic diseases and cancer. However, indels are heavily understudied due to experimental and computational challenges. This dissertation addresses the computational challenges in three aspects. First, the current approach of representing indels is ambiguous and causes significant database redundancy. A universal positioning system, UPS-indel, is proposed to represent equivalent indels unambiguously and the UPS-indel algorithm is theoretically proven to find all equivalent indels and is thus exhaustive. Second, a significant number of indels are hidden in DNA reads not mapped to the reference genome. Genesis-indel, a computational pipeline that explores the unmapped reads to identify novel indels that are initially missed, is developed. Genesis-indel has been shown to uncover indels that can be important genetic markers for breast cancer. Finally, mutations occurring in somatic cells play a vital role in transforming healthy cells into cancer cells. Therefore, accurate identification of somatic mutation is essential for a better understanding of cancer genomes. SomaticHunter, an ensemble of two sensitive variant callers, is developed. Simulated studies using whole genome and whole exome sequences have shown that SomaticHunter achieves recall comparable to state-of-the-art somatic mutation callers while delivering the highest precision and therefore resulting in the highest F1 score among all the callers compared.

Dedication

This dissertation is dedicated to my beloved family members for their unconditional love, encouragement, and support at every stage of my life.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor, Dr. Liqing Zhang for her enormous support since my first day at Virginia Tech. Her knowledge and guidance helped me in quick onboarding of this research area. She was always enthusiastic and encouraging for me to pursue the research problem of my interest, which helped me to grow as an independent researcher. The lessons I learned from her will help me in my future career.

I want to extend my gratitude to my Ph.D. committee members Dr. Lenwood S. Heath, Dr. Bert Huang, Dr. Xiaowei Wu, and Dr. Xinghua (Mindy) Shi for their suggestions and guidance to improve this dissertation. I want to thank Dr. Layne T. Watson for his help in mathematical proof of the exhaustiveness of the UPS-indel algorithm and for his contribution in improving the quality of the manuscript. My special thanks to Dr. Xiaowei Wu for helping me with the statistical analysis done in most of my Ph.D. works.

I express my sincere gratitude to the Computer Science department for supporting me as a Graduate Teaching Assistant throughout my PhD. Thanks to Sharon Kinder-Potter, the graduate coordinator of the CS department for her administrative support. I want to thank the tech staff of CS department, especially Rob Hunter, for addressing IT-related issues since my first day here at Virginia Tech. Thanks to my awesome lab mates Mingming, Vinaya, Hong, Jacob, Shaohua, Gustavo, Tithi, Min, Dhoha, Suraj, and Qinglai. I spent quality time with them, learned from them that helped me solve research problems. Special thanks to Tithi and Gustavo for their critical review of my dissertation chapters and their comments helped to improve the quality of my dissertation. I want to thank the members of the Association for Bangladeshi Students at Virginia Tech for bringing the Bangladeshi atmosphere to Blacksburg.

I am ever grateful to my parents, my sisters Sharmeen and Shabrina, my beloved niece Roja, my brother-in-law Saimon bhai, my in-laws, my uncles, aunts, and cousins for encouraging me to pursue PhD. I have been away from home since 2004 and could not be there when my family needed me. Despite that, my parents and sisters always supported and encouraged me whenever I needed, even while staying more than 8,200 miles away.

Last but not the least, I am indebted to my wife, Tithi, and my precious son, Saadid. Tithi, also a Ph.D. student in the CS department at Virginia Tech, has always been the first person to discuss

and brainstorm my research ideas. Our lunch and dinner time research discussion have always been enjoying because of her never-ending enthusiasm. I am so blessed to have a smart person like her as my lab mate, as well as my life partner. She spent sleepless nights to take care of our son so that I could entirely focus on wrapping up my projects and dissertation. My three months old son, Saadid, is the most precious gift that I could ever ask. He is a bundle of joy for us after a day full of work and stress. I feel so lucky to have him in our life. Without the tremendous support from all the members of my family, I would never be able to finish my Ph.D. in the anticipated timeline. I love you all.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Background

1.1 Introduction

Genetic variants or genetic mutations refer to the differences in DNA among individuals' genome. Analyzing these differences carry great significance and can answer questions such as (1) how did the genome of ancient human evolve to the modern human? (2) How much possibility is that a child would inherit a disease from one of the affected parents? (3) Why there are differences in phenotypes such as skin color or hair color from one individual to another? The invention of Next Generation Sequencing (NGS) technology has facilitated the large-scale sequencing of the human genome. This advancement in sequencing has led to a rapid increase in identification of different classes of genetic variants in the human genome such as single nucleotide polymorphism (SNP), insertion and deletion (indel), copy number variation (CNV), and other Structural Variants (SVs). However, indels in the human genome are not identified at the expected frequency although it is the second most abundant type of variants in the human genome. Moreover, different studies analyzing the human genome usually agree on SNPs but disagree on indels not only in terms of the number but also position and length. Additionally, there are indels that are not identified by any of the variant callers. Finally, there is no systematic approach available to unambiguously represent biologically equivalent indels. Failing to do so causes data redundancy that is observed significantly in widely used mutation databases. These major concerns need immediate attention and this dissertation describes computational approaches to tackle these problems.

1.2 Preliminaries

This section explains terminology frequently used in this dissertation.

1.2.1 Genetic mutation

Genetic mutation or genetic variants refers to the difference in an individual's DNA with respect to other human. Genetic mutation can be classified in six different types, namely single nucleotide polymorphism (SNP), insertion, deletion, duplication, inversion, and translocation. From the hereditary point of view genetic mutations can be germline and somatic. This dissertation mostly

deals with SNP, insertion, and deletion for both germline and somatic. These classes of genetic variants are explained below.

1.2.2 Single Nucleotide Polymorphism

A single nucleotide polymorphism (SNP) refers to the genetic variant where a single nucleotide at a specific position in the genome is substituted by another nucleotide. This is also known as single nucleotide variants (SNV) or point mutation. Among all types of genetic variants in the human genome, SNP is the most frequent one. SNPs can occur as a block where adjacent bases are substituted by a block containing different bases. This event is known as Multi Nucleotide Polymorphism or MNP.

1.2.3 Indel

Indel collectively refers to the insertion or deletion of bases at a specific position in the genome. This is the second most frequent variation type that occurs in the human genome. Indels with fewer than ten base pairs are often regarded as short indels. For indels occurring in coding regions, they can be classified as frameshift and in-frame indels. A frameshift indel is caused by the insertion or deletion of a number of nucleotides that is not divisible by three, i.e., the length of a codon. A frameshift indel produces a resultant translation different from the original by changing the reading frame which often produces truncated protein sequences. In-frame indels have insertion or deletion length divisible by three. Unlike a frameshift indel, an in-frame indel does not cause a shift in the reading frame, however, it can still lead to an abnormal protein sequence.

1.2.4 Germline variant

Germline mutations are the ones that occur within the germ cells also known as sex cells (sperm and egg). This type of variant can be passed to the offspring when muted germ cells come together to form a zygote, an event known as fertilization. After fertilization, during cell division, germ cells are rapidly divided to produce cells in the body and hence a mutation in the germ cell is passed onto every somatic and germline cell of the progeny.

1.2.5 Somatic variant

Somatic variants are opposite to the germline variants which means it can occur in any cell of the body except the germ cells and hence cannot be passed onto the offspring. There is strong evidence showing that this class of variant is responsible for cancer or other diseases.

1.2.6 Depth of coverage

The term coverage or depth of coverage generally refers to the average number of sequencing reads that align to each base within the sample DNA. For example, a whole genome sequenced at 30X coverage means that, on average, each base in the genome was covered by 30 sequencing reads.

1.3 Research Objectives

1.3.1 Objective 1: A unified positioning system to represent indels unambiguously.

Storing biologically equivalent indels as distinct entries in databases causes data redundancy and misleads downstream analysis. It is thus desirable to have a unified system for identifying and representing equivalent indels. Moreover, a unified system is also desirable to compare the indel calling results produced by different tools. To address this concern, the first research objective is to develop a utility tool named UPS-indel (a Universal Positioning System for Indel) that creates a universal positioning system for indels so that equivalent indels can be uniquely determined by their coordinates in the new system. This universal positioning system can also be used to compare results produced by different indel calling tools. UPS-indel is applied to find the redundant indels in dbSNP and is compared with state-of-the-art variant normalization tools in finding the number of redundant indels. In addition, UPS-indel is also compared with current approaches to compare variant calling results from different tools. **(Chapter 3)**

1.3.2 Objective 2: Rescuing the missed indels by leveraging the unmapped reads.

In current practice, Next Generation Sequencing (NGS) applications start with mapping/aligning short reads to the reference genome, with the aim of identifying genetic variants. Although existing alignment tools have shown great accuracy in mapping short reads to the reference genome, a significant number of short reads are left unmapped due to (1) structural variants longer than the allowed number of gaps and mismatches by the mapper, (2) sequencing error, or (3) sample contamination. In current practice, these unmapped reads are not used for variant calling and downstream analyses and thus mutations harbored in these unmapped reads remain hidden from any inference on association with any disease and cancer. However, some of the “hidden” or “missing” mutations can contain the key for understanding the molecular mechanisms of genetic diseases or cancer and might be used as markers for disease/cancer diagnosis and prognosis. To address this concern, the second research objective is to develop a computational pipeline named

Genesis-indel that explores the unmapped reads to identify the missed indels and relate these mutations to the genes of interest with the aim to discover important biological insights. Genesis-indel is applied to explore the unmapped reads of 30 breast cancer patients from The Cancer Genome Atlas (TCGA) and targeted analysis is done for the cancer genes (oncogenes and tumor suppressor genes) and also non-cancer genes in the human genome. The newly identified indels are analyzed using the Variant Effect Predictor (VEP) to determine the effect on the human genome. Additionally, functional analysis is done for the genes overlapping with the newly identified indels to determine the pathways that are affected by these mutations. **(Chapter 4)**

1.3.3 Objective 3: Developing a somatic variant caller to call somatic variants.

Somatic mutations play a vital role in transforming healthy cells into cancer cells. Therefore, accurate identification of somatic mutation is essential for many reasons such as a better understanding of cancer genomes, facilitating cancer diagnosis, and clinical decision making in cancer treatment. Many somatic mutation calling tools are available, and more are likely to appear. Most of these tools achieve high recall. However, high recall comes at the cost of low precision. Some variant callers show the opposite. Both of these cases result in low F1 score, a widely used performance metric. Therefore, combining the strength of multiple callers followed by an intelligent mechanism of classifying somatic and non-somatic mutation is a powerful way to generate a comprehensive list of somatic mutations while achieving both high recall and precision. Therefore, the third research objective is to develop SomaticHunter, an ensemble of two sensitive variant callers namely Platypus and VarDict and mutations called by these two callers are filtered by a decision tree classifier built using adaptive boosting with features such as read depth, mapping quality, and base quality. SomaticHunter is applied to synthetic tumor data for Whole Genome Sequence (WGS) and Whole Exome Sequence (WXS), and its performance is compared to state-of-the art somatic mutation callers including Mutect2, VarScan2, SomaticSniper, and Strelka2 for both SNPs and indels. **(Chapter 5)**

The chapters are arranged as follows: Chapter 2 discusses the current status of indel calling by comparing seven widely used indel callers using real data from the 1000 Genomes Project; Chapter 3 addresses the first research objective; Chapter 4 addresses the second research objective; Chapter 5 addresses the third research objective; Chapter 6 concludes the dissertation.

Chapter 2

Performance evaluation of indel calling tools using real short-read data

Insertion and deletion (indel), a common form of genetic variation, has been shown to cause or contribute to human genetic diseases and cancer. With the advance of Next Generation Sequencing technology, many indel-calling tools have been developed; however, evaluation and comparison of these tools using large-scale real data are still scant. Here, we evaluated seven popular and publicly available indel calling tools, GATK Unified Genotyper, VarScan, Pindel, SAMtools, Dindel, GTAK HaplotypeCaller, and Platypus. These tools are evaluated using low coverage data for 78 human genomes from the 1000 Genomes project. Comparing indels called by these tools with a known set of indels, we found that Platypus outperforms other tools. In addition, a high percentage of known indels still remain undetected, and the number of common indels called by all seven tools is very low. All these findings indicate the necessity of improving the existing tools or developing new algorithms to achieve reliable and consistent indel calling results.

2.1 Introduction

Insertion and deletion (indel), is a common form of polymorphism corresponding to the addition or removal of base pairs in the DNA sequence of an organism. Indels have been recognized as the second most abundant source of genetic variation in human populations [1-3]. Studies have shown that in the human body, 16% to 25% of all sequence polymorphisms are indels [4]. Furthermore, indels have been identified to play a key role in causing diseases. For example, cystic fibrosis, a common genetic disease, is frequently caused by deletion of three nucleotides in the coding region of the CFTR gene [5]. Diseases such as Fragile X Syndrome [6], trinucleotide repeat disorders [7], Mendelian disorders [8], Bloom Syndrome [9], acute myeloid leukemia [10-12], and lung cancer [13] are often associated with short repeats/insertions in the DNA sequence. Moreover, insertion of transposable elements such as Alu, L1, and SVA can interrupt gene function and cause diseases like hemophilia, neurofibromatosis, muscular dystrophy, and cancer [14]. In addition, indels can also change gene expression by altering phasing and spacing of DNA sequences in the promoter regions [15]. For example, a small insertion of 5 bps can rotate the binding site to the opposite face of the DNA helix whereas a long insertion of 100 bps can increase the spacing between two binding sites [3]. Therefore, indels in the promoter regions might explain the certain difference in gene expression observed in humans [15] and can be used as genetic markers in natural populations

[16]. Since indels influence human traits and diseases, detection of indels in a reliable manner is a prerequisite to developing effective treatment and medicine [17, 18].

Recently, Next Generation Sequencing (NGS) has become more convenient because of its high efficiency, improved sensitivity of different sequencing platforms, and reduced cost as compared to Sanger sequencing [19, 20]. By applying NGS in large scale, whole genome sequencing (WGS) is now possible at an individual level [21-23], and it has revealed a significant number of structural variants that were not reported previously. Since indels can alter human traits and cause diseases, the result of indel calling from individual WGS can be used to predict the future health of sampled individuals and to develop customized medical treatments.

Numerous indel calling tools have been developed so far that can be divided into four major categories: alignment-based methods, split read mapping methods, paired-end read mapping methods, and haplotype-based methods.

Alignment-based methods first map the reads to the reference sequence using read mapping software such as BWA [24] and Novoalign [25] and then call indels using the alignment data by applying some filtering steps to separate true indels from common sequence alignment errors (Figure 2.1A). In Figure 2.1A, "True Call" refers to the indels that passed after the filters are applied to separate indels from sequence alignment errors. Therefore, "False Calls" are those variants which are probably not indels but are called due to the alignment errors. Many indel calling tools belong to this category including Dindel [26], Stampy [27], SAMtools [28], Genome Analysis Tool Kit (GATK Unified Genotyper) [29], and VarScan [30, 31]. The main difference among these tools is in the model they use to distinguish true indel calls from alignment errors. Some use Bayesian probabilistic model (GATK Unified Genotyper, SAMtools, and Dindel) whereas others (VarScan) use a heuristic approach.

Split read mapping methods, on the other hand, firstly identify discordant paired-end reads for which one end maps completely to the reference sequence and the other end does not. The unmapped ends of these reads are then clustered or aligned by the de novo assembly to determine indels (Figure 2.1B). Tools in this category include Pindel [32] that uses a pattern growth approach to detect breakpoints of indels and SV-M [33] that performs a discriminative classification based on features of the split read alignment profiles and then filters the result against empirically derived training set data to reduce the false-positive rate.

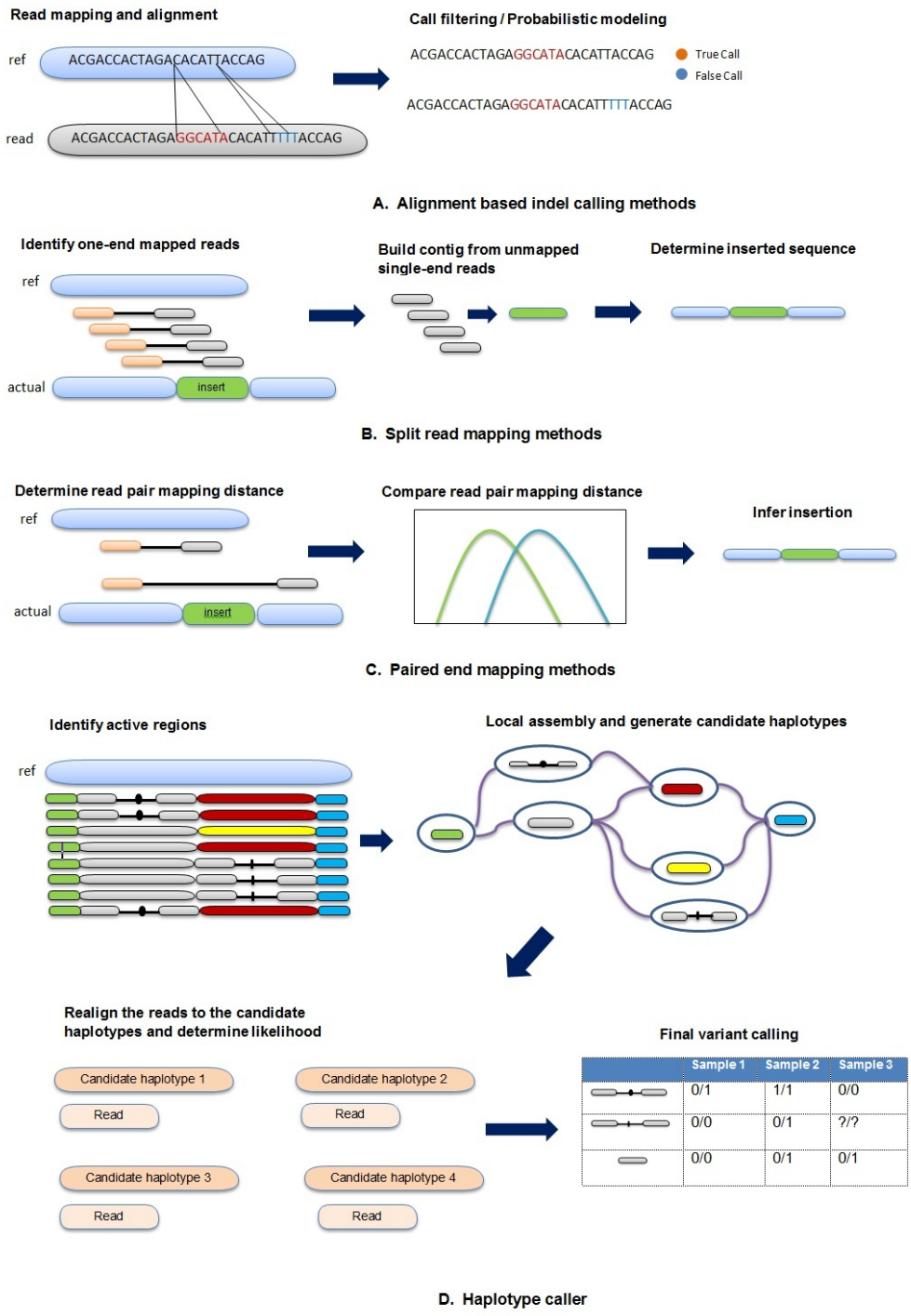


Figure 2.1:Categories of indel calling tools. Here, ref, is the reference sequence and actual is the observed sequence. (a) alignment based indel calling methods (b) split read mapping based indel calling method (c) paired end mapping based indel calling method (d) haplotype caller.

Paired-end read mapping methods compare the expected distance to the actual mapped distance to determine whether there is any indel in the sequence (Figure 2.1C). Tools belonging to this category include PEMer [34], Hydra [35], and BreakDancer [36].

Haplotype-based methods first identify the regions of interest where the reads show substantial evidence of having indels relative to the reference sequence. These regions are also known as active regions. For each active region, the callers build a De Bruijn graph to reassemble the active regions and yield the possible haplotypes present in the reads. After that, each read is realigned to the possible haplotypes and the likelihood of the haplotypes are calculated given the read data. Later Bayes' theorem or EM algorithms are applied to calculate the posterior probabilities and indels are called where the posterior probability exceeds a certain threshold value. In addition to that, some other filters are also applied to produce a fine-grained result. GATK HaplotypeCaller [37] and Platypus [38] belong to this category. Figure 2.1D shows the general overview of the haplotype-based indel callers.

Despite many indel calling tools, evaluation of the tools objectively and particularly using large-scale real data is sparse. There is an evaluation of four indel-calling tools (Dindel, VarScan, GATK Unified Genotyper, and SAMtools) done by Neuman et al. [39], however, it was based on simulated data. Instead of repeating the same experiment, here we performed the evaluation of the tools as well as three additional tools, and we use real data to gain insights. In this study, we investigated seven indel calling tools, GATK Unified Genotyper [29], VarScan [30], Pindel [32], SAMtools [28], Dindel [26], GATK HaplotypeCaller [37] and Platypus [38] using 78 human genomes from different populations in the 1000 Genomes project. All these tools are publicly available and are commonly used for benchmarking. Another reason for choosing these tools is that GATK Unified Genotyper, VarScan, SAMtools, Dindel, GATK Haplotype Caller, and Platypus can deal with short indels (< 50 bps) whereas Pindel can call medium to large indels ranging from 50bps to 10K bps. Therefore, altogether they cover indels of various lengths. Among these seven tools, four of them (GATK Unified Genotyper, VarScan, SAMtools, and Dindel) fall into the alignment based method category, one (Pindel) implements the split read mapping method and two (GATK HaplotypeCaller and Platypus) are haplotype-based methods. We did not consider tools that are based on paired-end read mapping because in most cases, they are insensitive to small indels, making it difficult to separate small perturbations in read pair distance from the

normal background variability [40]. Moreover, exact inserted or deleted sequence cannot be known from the results of tools that belong to this category [40]. We also note that only one of the two commonly used tools (Pindel and SV-M) from the split read mapping method category is included in this study. We did not consider SV-M mainly because this tool does not use BAM files as input. As described in the README file of SV-M, the input file requires start and end position of each chromosome along with several features corresponding to that chromosome such as number of uniquely mapped reads (UMRs) overlapping the deletion candidate, single position variation (SPV) from split read alignment and number of split reads supporting the same indel location. For consistency and to eliminate possible factors that could bias the comparison, we decide to exclude SV-M from this study.

2.2 Materials and Methods

2.2.1 Tools Investigated

We investigated seven indel calling tools, GATK Unified Genotyper, VarScan, Pindel, SAMtools, Dindel, GATK HaplotyCaller, and Platypus. A brief introduction of each tool and the commands for execution are provided below.

GATK Unified Genotyper (GATK_UG) [29] (version 2.7) is a tool developed by the Broad Institute of MIT and Harvard. For indel calling, it incorporates realistic read mapping error and base miscall models. Using a Bayesian genotype likelihood model, GATK_UG estimates the most likely genotypes and allele frequency in the sample while emitting an accurate posterior probability of having a segregating variant allele at each locus. We called indels by GATK_UG for each sample using the following command with default settings:

```
java -jar GenomeAnalysisTK.jar -R <reference.fasta> -T
UnifiedGenotyper -I <input.bam> -glm INDEL -o <output.vcf>
```

VarScan [30] (version 2.2.2) is a platform-independent software tool developed by the Genome Institute of Washington University. It uses the *mpileup* file generated by SAMtools [28] for scoring and sorting sequence alignments. The reads mapped uniquely to one location in the reference sequence are kept whereas the unmapped and ambiguous mapped reads are discarded. The uniquely mapped reads are further filtrated on read depth, base quality, variant allele frequency in

downstream analysis, and then used to call indels by a heuristic approach. Indels were called by VarScan with its default settings using the following commands:

To generate the mpileup file using SAMtools, following command is used:

```
./samtools mpileup -f <reference.fasta> <input.bam> >
out.mpileup
```

Following command is used to call indel from the mpileup file:

```
java -jar varscan.jar mpileup2indel out.mpileup --output-vcf 1 >
<output.vcf>
```

Pindel [32] (version 0.2.4) is a pattern growth approach based tool that detects breakpoints of large deletions, medium sized insertions, and other structural variants from NGS data at single-base resolution. In Pindel, all reads are initially mapped to the reference genome. The mapping results are then inspected to select paired reads that are mapped with indels or have only one end mapped. Based on the mapped reads, Pindel determines the anchor point on the reference genome as well as the direction of unmapped reads or the reads mapped with indels. Using this information and user-defined maximum deletion size, a sub-region in the reference genome is located where the unmapped reads are broken into fragments and then the fragments are mapped separately. Pindel was executed with its default settings using the following commands:

The contents of the Pindel configuration file is as follows:

```
<input.bam>    250  <sample_name>
```

Here 250 is the insert size, i.e., the length of the region between the paired-end adapters in paired-end sequence.

Following command is used to generate the output file:

```
./pindel -f <reference.fasta> -i <pindel_config.txt> -o
<pindel_output_file_name>
```

Following command is used to create the VCF file from the output file:

```
./pindel2vcf -r <reference.fasta> -R HUMAN_G1K_v37 -d <date> -p
<pindel_output_file_name_D> -e 5
```

SAMtools [28] (version 0.1.19) is a software package used for parsing and manipulating alignments in SAM/BAM format. For indel calling, it uses a Bayesian model for local realignment and base quality assessment. We called indels by SAMtools with its default settings using the following command:

Following command is used to generate the mpileup file, :

```
./samtools mpileup -uf <reference.fasta> <input.bam> |
./bcftools view -bvcg -> <output.bcf>
```

Following command is used to call indel from the mpileup file:

```
./bcftools view <output.bcf> | ./vcfutils.pl varFilter -D100 >
<output.vcf>
```

Dindel [26] (version 1.01), developed by the Wellcome Trust Sanger Institute in the UK, is a software tool that uses the Bayesian network for calling indels from NGS data. First, a number of candidate haplotypes, each containing at least 120 bps, are generated according to the hypothesis that indel events exist in pre-specified genomic segments. After realigning all reads to the candidate haplotypes using hidden Markov model, the posterior probability of a haplotype is calculated using the Bayesian approach and used to determine the presence of indels in the sample. Dindel assumes that all differences between the read and the candidate haplotype are caused by sequencing errors. By realigning reads to the candidate haplotype, it separates the indels from sequencing errors. Dindel uses mapping quality as the prior probability that a read should align to any of the candidate haplotypes and thus it effectively reduces the weight of reads that cannot be confidently mapped to that location in the genome. We used Dindel with default settings to call indels by the following commands:

Step 1: To extract candidate indels from the alignment file, following command is used:

```
./dindel --analysis getCIGARindels --bamFile <input.bam> --
outputFile <dindel_output> --ref <reference.fasta>
```

Step 2: Following command is used to create realignment windows

```
./makeWindows.py --inputVarFile <dindel_output.variants.txt> --
windowFilePrefix <dindel_output.realign_windows> --
numWindowsPerFile 1000
```

Step 3: For every window, generate candidate haplotypes from the candidate indels and realign the reads to these candidate haplotypes.

For each file created in Step 2

```
./dindel --analysis indels --doDiploid --bamFile <input.bam> --
ref <reference.fa> --varFile
<dindel_output.realign_windows.X.txt> --libFile
<dindel_output.libraries.txt> --outputFile
<dindel_output_windows.X> [Here X = window number currently
being analyzed]
```

Step 4: Create the final output

Merging results from all realignment windows:

```
ls | grep ".glf.txt" > <list.txt>
```

```
mergeOutputDiploid.py --inputFiles <list.txt> --outputFile
<output.vcf> --ref <reference.fasta>
```

GATK HaplotypeCaller (GATK_HC) (version 3.30) [37] is a tool developed by the Broad Institute of MIT and Harvard. For indel calling, at first, it determines the regions of the genome where there is significant evidence of variation. Therefore, regions that do not show any variation beyond the expected levels of background noise are skipped. After this step, the resulting regions having significant evidence of variations are passed to the next step. These regions are known as "Active Regions". For each active region, in the second step, GATK_HC builds a De Bruijn graph to reassemble the active regions and identifies the candidate haplotypes present in the reads of the given sample. Additionally, each haplotype is locally realigned to the reference haplotype to identify the potentially variant sites. In the next step, for each active region, each read is then pairwise aligned to each of the candidate haplotype using the PairHMM algorithm. This produces a matrix of likelihoods of haplotypes for the reads in the given sample. These likelihoods are then

marginalized to obtain the likelihoods of the alleles per read for each potentially variant site. For each potentially variant site, in the next step, Bayes' theorem is applied to determine the posterior likelihoods of each genotype per sample using the likelihoods of alleles obtained in the previous step. The most likely genotype is then assigned to the given sample. We called indels using GATK_HC with default settings using the following command:

```
java -jar GenomeAnalysisTK.jar –R <reference.fasta> –T
HaplotypeCaller –I <input.bam> –o <output.vcf>
```

Platypus [38] (version 0.7.9.1) is a haplotype-based variant calling tool developed by the Wellcome Trust Sanger Institute in the UK. In this tool, in the beginning, candidate variants are obtained from read alignments, local assembly, and external sources, and then candidate haplotypes are formed. After haplotypes are generated from candidate variants, their frequencies are estimated on the basis of their likelihood. These likelihoods are calculated by aligning a read to the haplotype sequence with an underlying Hidden Markov Model (HMM). The forward algorithm is used to calculate the likelihood of a read given a haplotype. After the likelihood is calculated for all combinations of reads and haplotypes, an EM algorithm is used to estimate the frequency of each haplotype under a diploid genotype model. In the next step, the posterior support for any variant is computed by comparing the likelihood of the data given all haplotypes and the likelihood given only those haplotypes that do not include a particular variant. Later, indels are called when their posterior support exceeds a threshold using these frequencies as a prior. The variants are also filtered based on allele bias, strand bias, mapping quality, quality over depth, posterior quality, and sequence context. We called indel with the default settings of Platypus using the following command:

```
python Platypus.py callVariants --bamFiles=<input.bam> --
refFile=<reference.fasta> --output=<output.vcf>
```

2.2.2 Data set

The data set consists of low coverage (~3X to ~12X) alignment profiles from 78 humans that belong to 26 populations and were collected for the 1000 Genomes project [41]. We used the alignment files of chromosome 11 as input for the tools we investigated. These short reads were sequenced on Illumina Genome Analyzer platform [42] and mapped using BWA [24]. We used

hs37d5 as the human reference genome, which is an extended version of the Build37 data set of the 1000 Genomes project with additional sequences. Note that, this reference genome was used by the 1000 Genomes project in the final phase. Table A.1 in Appendix A lists the samples we used with their corresponding ethnic background and coverage.

Ideally, a benchmarking data set for evaluating indel calling tools would consist of a list of known indels for the samples. However, such a benchmarking data set is not available in large quantity [43]. Hence, for evaluation purposes, we used the indels identified in Mills et al. [43] as the gold standard. To call indels, Mills et al [43] examined 98 million Applied Biosystems (Sanger) DNA resequencing traces from the trace archive of NCBI, which has been proved to be sufficient for accurate indel calling [4]. After some preprocessing of the traces based on the quality scores, they were compared to the human reference genome to call indels. Details about the indel calling procedure and some post-processing to generate the gold standard data set can be found at [4]. The called indels were validated using PCR-based methodologies and the validation rate was 97.2%. This data set reports almost two million small and large indels found in all 24 chromosomes of 79 diverse humans with lengths ranging from 1bp to 10,000 bps. Moreover, it has been confirmed that the sequence traces used in Mills et al. [43] provide excellent coverage of the human genome. Note that the samples we used here are sequenced on the Illumina Genome Analyzer platform and the indels listed in the "gold standard' data set are called using the Applied Biosystem (Sanger) DNA re-sequencing traces. In spite of these differences, the indels identified in the gold standard data set are considered being most likely reliable and they have been used as the gold standard in other studies [44, 45]. In Mills et al. [43], 58,811 indels were identified for chromosome 11, and, in the current study, we used this set as the gold standard. Note that we did not use simulated data for benchmarking because though simulated data are valuable, they do not always represent the actual phenomena. We could also use sample benchmark data set available in "Genome in a Bottle Consortium" [46] but that one relies on a single data set from one human only (NA12878).

2.2.3 Evaluation Criteria

We evaluated the tools according to running time, number of indels called, comparison with the set of gold standard indels, similarity among the tools, hierarchical clustering, and ranking of the tools.

For each sample, we executed the tools and recorded the number of indels called by each tool as well as the running time. To see the relation between running time and coverage of the read, besides the low coverage samples, we also included the sample NA12878 with ~64X coverage. All analyses were done on a Linux machine with Intel Core i7-2600 CPU @ 3.40 GHz with 8 processors, 16 GB RAM, and Ubuntu 12.04 LTS operating system.

Indels called by the seven tools were compared with those identified in Mills et al. [43]. From this comparison, we calculated the corresponding recall and precision for each of the tools using formula (1) and (2).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

For comparing the accuracy of the tools, we used the F-measure, the harmonic mean of the precision and recall, where an F-measure reaches its best value at 1 and worst score at 0. The F-measure was calculated using formula (3).

$$\text{F-measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{3}$$

Note that the position of an indel with respect to the reference sequence sometimes cannot be defined unambiguously by a single coordinate [20, 47]. As shown in Figure 2.2, the insertion of a guanine into the local sequence of $T_iG_{i+1}G_{i+2}C_{i+3}$ after position i produces the same mutated sequence as inserting guanine after position i+1 or i+2. Hence these insertions have identical biological meaning and therefore an unambiguous annotation for this insertion should list all equivalent indel positions, i.e., $+G$ $\{i, i+1, i+2\}$ [20]. For this reason, while comparing an indel called by each tool with the indel in position i in the gold standard data, we treated the indel called by the tool as true positive if it is within the range of $i \pm 5$ positions.

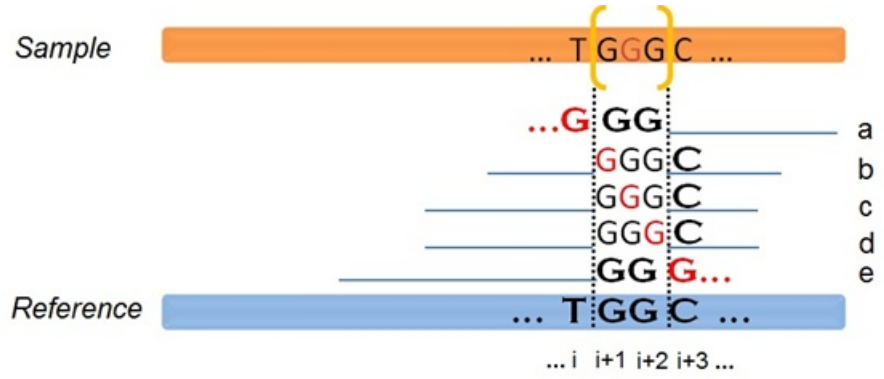


Figure 2.2: Example of identical indels taking place in relative positions. (Adapted from Krawitz et al., 2010 [20]).

Based on the indel calling results, these tools were ranked in the ROC space, where the X and Y axes are denoted by False Positive Rate (FPR) and True Positive Rate (TPR), respectively. Here TPR is equivalent to recall and FPR is simply (1 – precision) as calculated using formulas (1) and (2). In the ROC space, each point represents the prediction result or instance of a confusion matrix. The diagonal (Y=X) that divides the ROC space represents the decision from a "Random Guess".

Points above the diagonal represent good classification results whereas points below the line represent poor results. For each sample, we first calculated the TPR and FPR for each tool and plotted it as a point in the ROC space, then ranked the tools based on the perpendicular distance of each point from the diagonal.

We also examined the similarity among the results produced by different tools. The Jaccard index, also known as the Jaccard similarity coefficient, is used to compare the similarity between indel predictions. For two finite sets A and B, the Jaccard index can be calculated using

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{4}$$

The maximum value of Jaccard index is 1 when two indel sets are the same whereas the minimum is 0 when two indel sets are completely different.

Another interesting question to ask is "how are the seven indel calling tools related to one another on the whole"? To answer this question, we clustered the tools using the following three steps: (1) Divide the reference sequence into windows of equal size. We tested with different window sizes (1,000 bps, 10,000 bps, 10,000 bps and 1,000,000 bps) and found that the window size does not affect the clustering result. For computational convenience, we set the window size to 1M bps. (2)

For each window, calculate the number of indels called by each tool. (3) Construct a vector of indel counts of all windows for each tool and apply the UPGMA hierarchical clustering algorithm to the seven vectors.

2.3 Results

2.3.1 Running time

We compared the tools on the average running time taken to call indels for a sample. Table 2.1 shows the average running time for samples with low coverage (average coverage ~6X) and high coverage (~64X). For both high and low coverage data, Platypus is the fastest and Dindel is the slowest of all the tools investigated. Clearly, indel calling is more time consuming for high coverage data than for low coverage data, which is especially evident for Dindel due to its complicated model for realignment. Since Dindel tests all indels identified by the read mapper many of which might be sequencing errors, with the increase of number of reads and increase in sequencing errors, the computation time increases quadratically [26].

Table 2.1: Average running time spent in calling indels for samples with low/high coverage.

Tool	Time	
	Low Coverage (~6X)	High Coverage (~64X)
GATK_UG	16 minutes 43 seconds	24 minutes 19 seconds
VarScan	16 minutes 1 second	84 minutes 02 seconds
Pindel	25 minutes 36 seconds	139 minutes 09 seconds
SAMtools	11 minutes 26 seconds	64 minutes 14 seconds
Dindel	165 minutes 22 seconds	1549 minutes 18 seconds
GATK_HC	58 minutes 27 seconds	91 minutes 13 seconds
Platypus	3 minutes 36 seconds	5 minutes 59 seconds

2.3.2 Number of indels called

Figure 2.3 shows the number of indels called by each tool for each sample. The seven tools under consideration call a different number of indels. The numbers of indels called across the 78 samples range from 1,431 to 15,585 for GATK_UG, from 114 to 10,619 for VarScan, from 1,845 to 11,455 for Pindel, from 9,351 to 20,245 for SAMtools, from 9,864 to 19,876 for Dindel, from 10,915 to 24,786 for GATK_HC, and from 15,062 to 34,600 for Platypus. On average, Platypus calls the maximum number of indels (average number = 23,321) whereas VarScan calls the minimum

(2,775). The average number of indels called by SAMtools (14,719) follows closely to that by Dindel across the samples. Similarly, the average numbers of indels called by GATK_UG (6,733) and Pindel (6,382) are very similar to each other across the samples. As we can see from these results, VarScan is evidently the most conservative one in calling indels. It calls many fewer indels than others. This might be due to its rather stringent filtering step during which all the unmapped and ambiguous reads are discarded. Although this step is helpful in keeping the false positives down, it also reduces the power of detecting true indels.

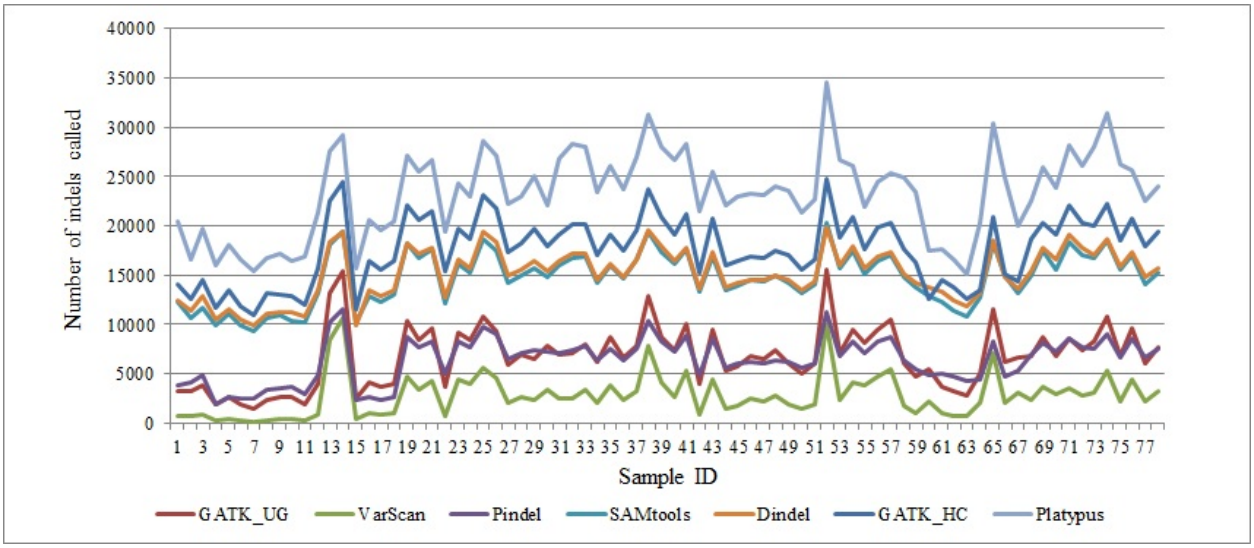


Figure 2.3: Number of indels called by the seven tools for the 78 human genomes.

2.3.3 The lengths of indels called

We examined the distributions of lengths of indels called by the seven tools and compared them to that of the gold standard data set. All the indel distributions based on lengths are shown in Figure A.1 in Appendix A which shows that 96.2% of the indels in the benchmark data set are 1-10 bps, 98.6% in GATK_UG, 99.0% in VarScan, 93.4% in Pindel, 92.2% in SAMtools, 95.1% in Dindel, 94.01% in GATK_HC, and 97.19% in Platypus. Therefore, most of the indels in the benchmark and the ones called by the tools are $\leq$10 bps. Chi-square statistical tests show that the distributions of indel sizes are not significantly different between the calling results of the tools and the gold standard (p-values for comparing the gold standard with GATK_UG, VarScan, Pindel, SAMtools, Dindel, GATK_HC, and Platypus are 0.89, 0.81, 0.96, 0.28, 0.94, 0.95 and 0.99 respectively). Note that Pindel is known for calling medium to large indels but here most of the indels called by Pindel are small indels.

Regardless of the gold standard indels, we are interested to see the similarity/dissimilarity of distribution of indels sizes among the tools themselves. From a Chi-square statistical test between tools, we see that the distributions of indel sizes are not significantly different among the tools. The p-values for intra-tool comparisons are shown in Table A.2 in Appendix A.

2.3.4 Effect of the depth of coverage on the number of indels called

To see how the number of indels called by these tools is affected by the depth of coverage, we estimated the depth of coverage for each human sample (shown in Appendix A, Table A.1). Figure 2.4 shows the relationship between the number of indels called by the seven tools and the coverage depth. Overall, the higher the coverage is, the more indels are called. Pearson correlation coefficients between the coverage and the number of indels called by GATK_UG, VarScan, Pindel, SAMtools, Dindel, GATK_HC, and Platypus are 0.97 (p-value = 8.02×10^{-48}), 0.97 (p-value = 5.25×10^{-48}), 0.91 (p-value = 3.10×10^{-30}), 0.89 (p-value = 4.64×10^{-27}), 0.88 (p-value = 2.76×10^{-26}), 0.86 (p-value = 2.24×10^{-23}), and 0.82 (p-value = 6.84×10^{-20}), respectively. Thus, consistent with previous findings [48], the number of indels called, regardless of the tools, is significantly positively correlated with the coverage depth.

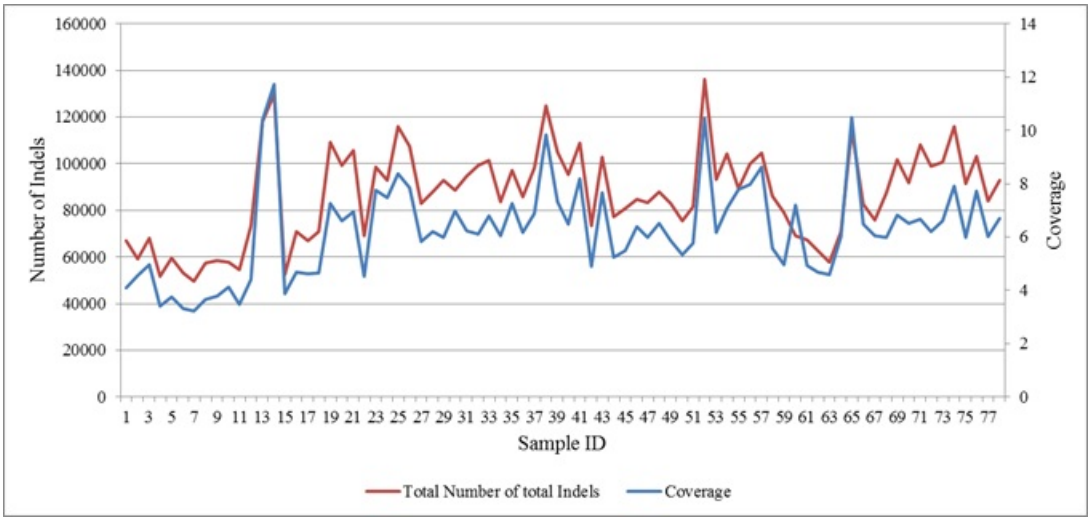


Figure 2.4: Relationship between coverage and the pooled number of indels by the seven tools.

2.3.5 Comparison with the set of "gold standard" indels and ranking of the tools

Figure 2.5 shows the percentage of gold standard indels called by the tools across the 78 samples. For chromosome 11, on average, only about 1.51 % of the gold standard indels are called by all seven tools, whereas about 76.91% are undetected by any of the tools. The remaining ~21.58% is

called by at least one tool. We also compared the tools for the percentage of their own indels called by others regardless of the gold standard indels. For this purpose, we picked up Dindel, SAMtools, GATK_HC, and Platypus as they call more indels than the other three tools. Venn diagram in Appendix A, Figure A.2 shows that only 15.64% of the indels were called by all of these four tools revealing that regardless of the gold standard indels, a major percentage of indels remain undetected.

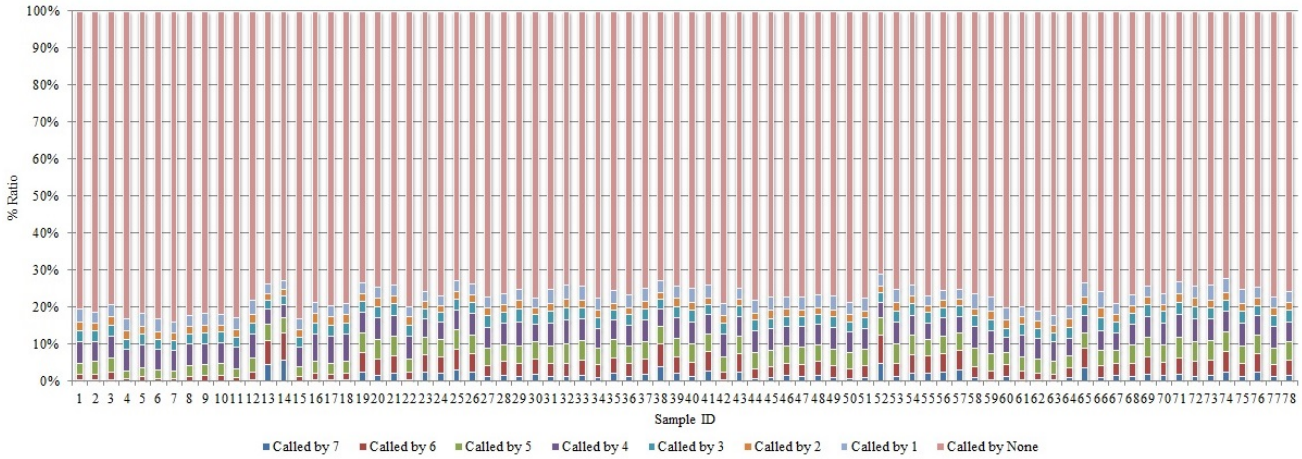


Figure 2.5: Percentage of the gold standard indels called by the tools.

We also examined the overall performance of each tool on the 78 samples. The average F-measure values for GATK_UG, VarScan, Pindel, SAMtools, Dindel, GATK_HC, and Platypus are 0.14, 0.06, 0.12, 0.26, 0.27, 0.28, and 0.31, respectively.

In the ROC analysis, we ranked the seven tools based on their distance from the "Random Guess" line in the ROC space. Table 2.2 shows the frequency of the ranks of the tools based on the 78 samples. Platypus ranked the best for all 78 samples and GATK_HC ranked the second best. VarScan performed poorly, ranking the worst for 76 samples. Pindel performed also poorly, ranking the worst for 2 samples, and second worst for 60 samples. In addition to the ranking, we also computed the average recall, precision, and F-measure for the tools in Table 2.2. For the average recall of the 78 samples, Platypus ranks the highest (0.22), followed closely by GATK_HC (0.18), and VarScan the lowest (0.03). For the average precision, GATK_UG ranks the highest (0.72), followed closely by VarScan (0.71). GATK_HC (0.61) and Platypus (0.56) have slightly lower average precision. For the average F-measure, Platypus (0.31) ranks the highest and VarScan (0.06) the lowest. To get a clear idea about how the performance of the tools depends on the indel

types, i.e., insertion and deletion, we split the benchmark data set based on the indel types and results are shown in Appendix A, Figure A.3. Results show that, except for Pindel, performance of the other tools remain consistent regardless of the indel type. Pindel shows better performance in calling deletion than insertion.

Table 2.2: Frequency of the ranks of the tools based on the ROC curve for the 78 samples. Average recall, precision, and F-measure across the samples are also provided.

Rank / Name	1	2	3	4	5	6	7	Average Recall	Average Precision	Average F-measure
GATK_UG	0	0	0	0	62	16	0	0.081884	0.72141	0.14435
VarScan	0	0	0	0	0	2	76	0.033987	0.717315	0.063333
Pindel	0	0	0	0	16	60	2	0.068635	0.636704	0.122591
SAMtools	0	0	1	77	0	0	0	0.160989	0.645108	0.256343
Dindel	0	5	73	0	0	0	0	0.170076	0.662287	0.269404
GATK_HC	0	73	4	1	0	0	0	0.181928	0.608907	0.278323
Platypus	78	0	0	0	0	0	0	0.220391	0.559842	0.314071

2.3.6 Performance of the tools on indels of different lengths

A natural question to ask is whether the seven tools' performance changes with different indel sizes. We computed the average F-measure (Figure 2.6), false negative rate (Figure 2.7), recall (Figure A.4 in Appendix A), precision (Figure A.5 in Appendix A) of the seven tools for indels of lengths 1-10 bps. Results show that for all the tools, the performance of calling indels correctly shows a slight decrease with the increase of indel lengths. Platypus, GATK_HC, Dindel, and SAMtools show highly similar patterns for four metrics (i.e., F-measure, false negative rate, recall, and precision) with respect to indel lengths. Altogether, this comparison based on indels of different lengths shows that these tools achieve similar performance for different subcategories of indels with a certain length. In other words, indel length is not a confounding factor that affects the performance of these calling tools.

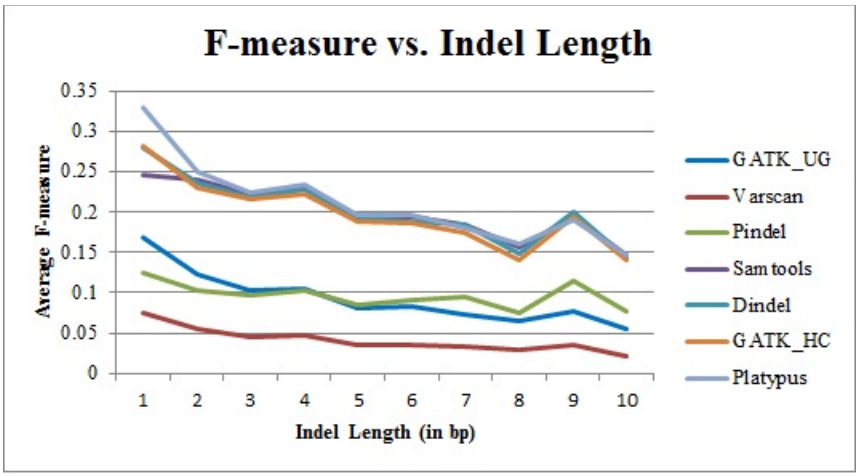


Figure 2.6: Average F-measure for each tool for different lengths of indels.

2.3.7 Similarity among the tools

We also compared the tools for their similarity regardless of the gold standard. For each sample, the Jaccard index of each pair of the tools is shown in Figure 2.8, and the average Jaccard index across all samples is listed in Table 2.3. From the Jaccard index, we found high similarity between SAMtools and Dindel. A possible reason is that both tools use a Bayesian approach for calling indels. SAMtools calculates a Bayesian prior probability and uses it to calculate the actual genotype for the variants detected. Dindel, on the other hand, calls indels by realigning the reads against candidate haplotypes for which prior probabilities calculated using a Bayesian approach are already known. Both SAMtools and Dindel perform local realignment and base quality assessment for calling indels and that is also another possible reason for their similarity. Similarly, Platypus and GATK_HC also have high Jaccard index value that represents their strong similarity. Being haplotype callers, they have underlying similarity such as generating candidate haplotypes and then realigning reads to each of these candidate haplotypes for variant calling, which explains the reason for their similarity.

Table 2.3: Average Jaccard index for each pair of the tools (Jaccard index is computed for each pair of the tools for each human sample and then averaged across all the 78 samples).

	GATK_UG	**VarScan**	**Pindel**	**SAMtools**	**Dindel**	**GATK_HC**	**Platypus**
GATK_UG	1	0.35	0.30	0.34	0.32	0.35	0.28
VarScan		1	0.15	0.15	0.14	0.14	0.11
Pindel			1	0.27	0.27	0.32	0.25
SAMtools				1	0.73	0.60	0.56
Dindel					1	0.65	0.57
GATK_HC						1	0.64
Platypus							1

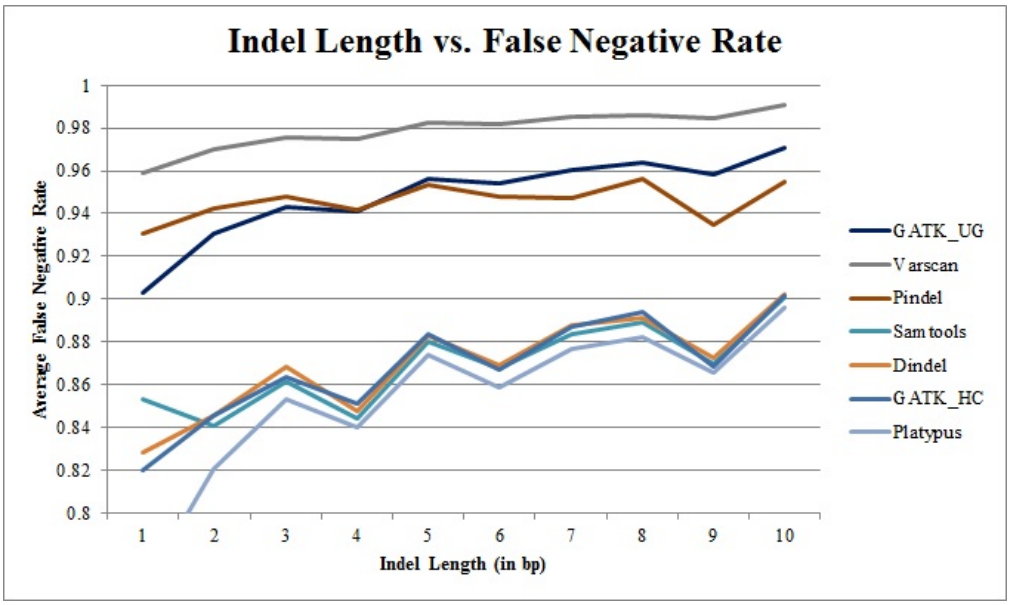


Figure 2.7: Average False Negative Rate for each tool for different lengths of indels.

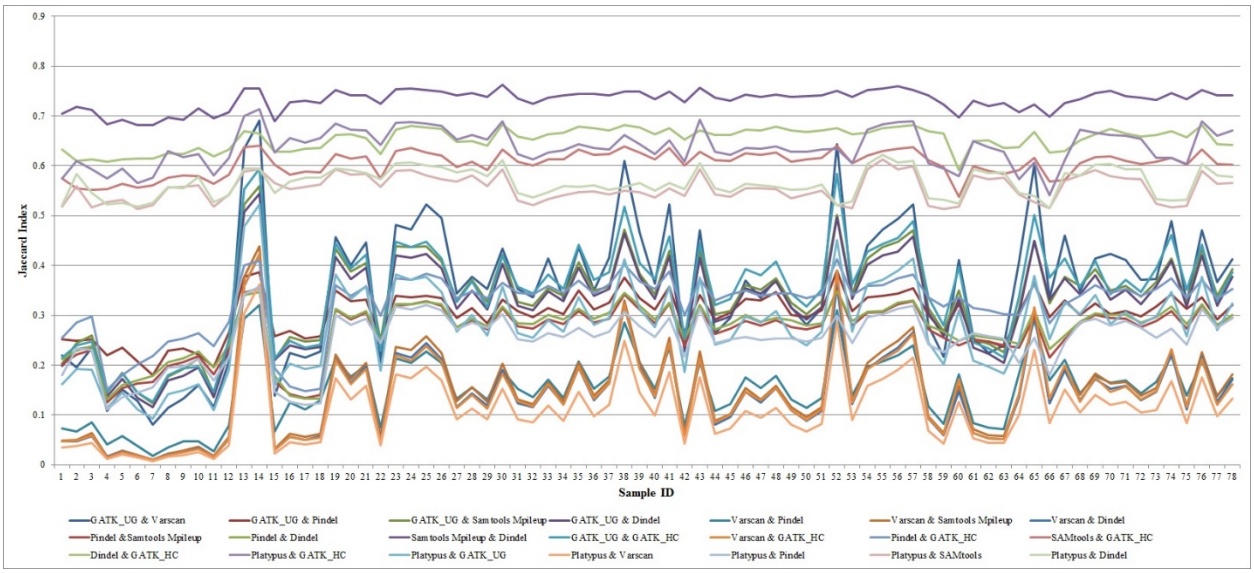


Figure 2.8: Jaccard Index for each pair of tools for each sample.

Figure 2.9 shows the dendrogram of UPGMA hierarchical clustering of the tools based on number of indels called by the tools. Again, we see that Dindel and SAMtools group together and Platypus and GATK_HC group together which is supporting our previous observation of similarity between these tools.

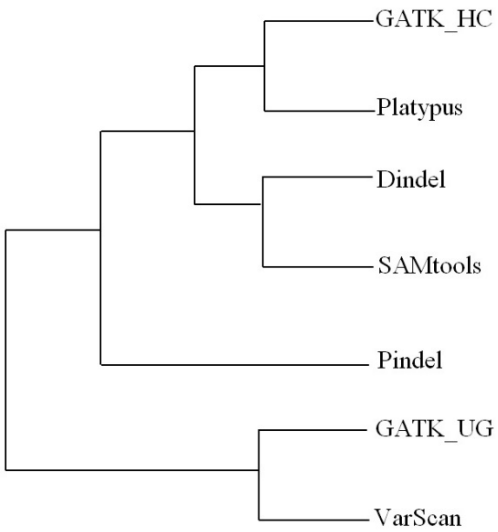


Figure 2.9: Hierarchical clustering of the tools.

2.4 Discussion

In this chapter, we investigated seven tools that are publicly available and well known for calling indels from short reads. Using 78 whole genome short reads data from the 1000 Genomes project,

we evaluated these tools based on several criteria, including running time, number of indels called, recall, precision, F-measure based on the "gold standard" data, and ranking and clustering of the tools. Results show that Platypus outperforms other tools in most of the aspects.

The low percentage of the called indels over the "gold standard" indels indicates that all these tools exhibit limited power in detecting indels. Several factors could contribute to the low true positive rate. Firstly, since existing read mappers map each read to the reference sequence independently of other reads, due to the alignment artifacts, insertions and deletions can be improperly placed relative to their true positions, and it affects the indel calling results greatly. Secondly, most of the indel calling tools do not have sophisticated methods for checking sequencing errors before calling indels. Though Platypus, GATK_HC, and Dindel realign the candidate indels to the known haplotypes, especially for Dindel and GATK_HC, due to their high computational time, it is not an efficient way when the depth of coverage of the reads is high. Therefore, indel calling results can be improved if these factors are considered. Thirdly, the indels we used as a gold standard were identified from the DNA traces obtained from the trace archive at NCBI (http://www.ncbi.nlm.nih.gov/Traces/trace.cgi), though it is more reliable than using short reads, indels identified in this way nevertheless can still be false positives, which could lead to an artificial decrease of the true positive rate. Fourthly, the set of gold standard indels is the pooled result of indels from 79 individuals, which naturally has more indels than individual humans. However, this might not be the dominating factor causing the low false positive rate as the number of pooled indels for 78 humans is still very low compared to the gold standard. Finally, the low true positive rate might also be due to the chromosome-specific behavior of the calling tools. Although we have no particular reason to suspect that the indel calling results for chromosome 11 should be different from those for other autosomes, we examined the performance of the seven tools on chromosome 20 to see whether the result is chromosome specific. Results show that all the metrics (i.e., recall, precision, and F-measures) follow closely to those of chromosome 11 (Figure A.6 in Appendix A), and, therefore, the poor performance of the tools evaluated by the gold standard indels is not chromosome specific.

Clearly, an important issue in evaluating various indel calling tools is the lack of gold standard data set or benchmark data set. In the current study, the performance comparison is done based on the gold standard data set that is the best possible resource available. Although it lists two million

short and long indels extracted from the genomes of 79 diverse humans, it does not list all the indels that take place in the genomes of the human samples we considered here. Though we can say that Platypus performs better than other tools based on the gold standard data set, however, in general, we cannot decide about which tool is the best unless we have the list of true indels for each sample. So, developing a list of indels for individual humans will be a good direction for future research and that list will be a useful resource for validating the existing as well as newly developed indel calling tools. Moreover, people from the same ethnic group tend to have common indels [49, 50]. Therefore, creating a list of known indels for the same ethnic group and comparing the tools based on the indels called for the samples from that ethnic group would be a better way to evaluate the performance of the tools.

Besides improving the indel calling tools, another strategy to improve the indel calling result is increasing the depth of coverage of the reads. For each of the tools, the performance shows a positive correlation with the coverage of the reads. Pearson correlation coefficients between coverage and F-measure for GATK_UG, VarScan, Pindel, SAMtools, Dindel, GATK_HC, and Platypus are 0.96 (p-value = 6.75×10^{-44}), 0.98 (p-value = 1.38×10^{-54}), 0.89 (p-value = 1.56×10^{-27}), 0.87 (p-value = 1.22×10^{-24}), 0.85 (p-value = 3.65×10^{-23}), 0.85 (p-value = 9.47×10^{-23}), and 0.81 (p-value = 1.15×10^{-19}) respectively. Moreover, we also performed down-sampling of the individual that has 64x coverage to create a 5x coverage sample and conducted indel calling using the seven tools. Results further confirm that higher coverage yield better results, reflected by higher F-measures for all seven tools in the 64x coverage. However, for all seven tools, precision is higher in the 5x coverage sample than in the 64x coverage sample. Detailed results are shown in Appendix A, Figure A.7. Hence the performance of the tools can be significantly improved by increasing the depth of coverage of the reads. Consistent with our finding, a previous evaluation of indel calling tools based on simulation data has shown that sensitivity of indel calling tools increases with coverage depth [39]. Joint sample calling is another strategy to call indels from low coverage data and greater sensitivity can be achieved through this. However, it has a few limitations such as (i) since it calls variants simultaneously across all samples, computational expense increases exponentially with the increase of number of samples and (ii) every time a new sample is added to the cohort, the process of variant calling needs to start again from the beginning; this is known as the (N+1) problem [51]. Haplotype callers like

GATK_HC and Platypus are free from these limitations; however, the other tools are yet to overcome these limitations.

Finally, although the indel calling results produced by the tools show a great discrepancy, these tools can show strengths in different aspects such as running time, the number of indels identified, and indels of different lengths. Thus integrating the strength of existing tools to call indels and then passing the results to an aggregating machine learning model to increase true positives and reduce false positives might be a good solution. Similar ideas were discussed in [52] for creating highly confident SNP, indel, and homozygous reference genotype calls.

2.5 Conclusion

The indel is one of the main types of disease-causing variation in humans. Detecting indels in an efficient manner is necessary for discovering proper medication. The advent of NGS technology has made it possible to sequence human genomes at an individual level. We have investigated seven well-known tools, GATK Unified Genotyper (GATK_UG), VarScan, Pindel, SAMtools, Dindel, GATK HaplotyCaller (GATK_HC), and Platypus that call indels using NGS data. Based on the benchmark data set we used, Platypus outperformed other tools. However, all of these tools have limitations as a large number of indels listed in the benchmark data set remain undetected. A sophisticated method to check sequencing errors before calling indels and an integrative approach to combine the strengths of existing indel calling tools might be a good solution to overcome this problem. Using reads with high coverage is another strategy to obtain better results. Although the benchmark data set we used for comparing the tools contains a large number of short and long indels that take place in diverse human genomes, it may not contain all the indels that occur in the genome of the samples we considered here. Hence, developing a list of known indels at an individual level will be helpful for validating the existing and newly developed tools.

Chapter 3

UPS-indel: A Universal Positioning System for Indels

Storing biologically equivalent indels as distinct entries in databases causes data redundancy, and misleads downstream analysis. It is thus desirable to have a unified system for identifying and representing equivalent indels. Moreover, a unified system is also desirable to compare the indel calling results produced by different tools. This chapter describes UPS-indel, a utility tool that creates a universal positioning system for indels so that equivalent indels can be uniquely determined by their coordinates in the new system, which also can be used to compare different indel calling results. UPS-indel identifies 15% redundant indels in dbSNP, 29% in COSMIC coding, and 13% in COSMIC noncoding data sets across all human chromosomes, higher than previously reported. Comparing the performance of UPS-indel with existing variant normalization tools vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants shows that UPS-indel is able to identify 456,352 more redundant indels in dbSNP; 2,118 more in COSMIC coding, and 553 more in the COSMIC noncoding indel data set in addition to the ones reported jointly by these tools. Moreover, comparing UPS-indel to state-of-the-art approaches for indel call set comparison demonstrates its clear superiority in finding common indels among call sets. UPS-indel is theoretically proven to find all equivalent indels and is thus exhaustive.

3.1 Introduction

Indel stands for insertion or deletion of bases in a DNA sequence. As the second most common form of genetic variation, indels play an important role in genome and protein evolution. Due to artificial factors such as sequencing errors, ambiguous alignment of the reads, inconsistent ways of representing the same variant by different tools, the same mutation may be recognized as distinct variations occurring at different location [53-55]. For example, consider a reference sequence AGGAAAGAAAGAAAGAAAGAG ranging from position 100285630 to 100285650 and two indels stored in dbSNP, rs147659011 (GAAA/+) and rs60376183 (AAGA/+), annotated to this region with positions 100285632 and 100285650, respectively. Although these indel mutations may indeed occur at different positions, they are biologically equivalent because they result in the same altered sequence AGGAAAGAAAGAAAGAAAGAAAGAG. Appendix B, Table B.1, shows another example of redundant insertion and deletion in dbSNP. Since many databases such as dbSNP, Database of Genomic Variants (DGV), and Ensembl combine indels resulting from

large-scale studies, similar cases often exist in those databases, leading to a nonnegligible problem of data redundancy. In fact, about 10% [47] of the human indels stored in dbSNP and 18% [53] in Ensembl are redundant. Resolving indel redundancy in major databases is important for subsequent genetics research. Nevertheless, this problem has not been given the attention it deserves.

Numerous approaches have been developed for systematic comparison of indels to determine equivalence and hence solve the redundancy problem. The strict matching approach matches two indels if they share the same position, reference, and alternate alleles in two different entries in the VCF file. However, as demonstrated in [55], this approach fails to find equivalent indels that are not identical. The distance based approach treats two indels as equivalent if both have the same length and occur within a certain distance such as ± 5 bp [56] or ± 25 bp [57]. However, this approach introduces false positives when neighboring indels are not equivalent [53] and misses equivalent indels that are farther apart than the distance cutoff. Clearly, selection of an optimal distance cutoff is a tradeoff between the two types of errors; smaller distance cutoffs result in a decreased false positive rate but an increased false negative rate.

To address the limitations of the two aforementioned approaches, the more widely used "normalization" approach attempts to solve the indel redundancy problem by left (or right) normalization, i.e., consistently shifting the start position of an indel to the left (or right) as long as the resulting sequence is the same as the one generated by the original mutation [58]. Tools using this type of variant normalization include vt normalize [54], BCFtools [28], and GATK LeftAlignAndTrimVariants [29]. These tools usually take a VCF file as input, output another VCF file with canonical VCF entries for the indels after normalization, and then perform "strict matching" to find equivalent indels with exactly the same canonical representation. The normalization approach generally performs well in identifying equivalent indels, but as shown in the result section, fails to normalize complex variants.

The positions of indels may be changed after left/right normalization, potentially misleading downstream analysis. For example, the deletion rs536379477 resides in the exon of the transcript ENST00000590192.1, but the equivalent deletion rs41436444 is in the intron of the same transcript. Therefore, reporting these two indels with the same normalized position might lead to missing significant insight into genetic diseases or phenotypes of interest. Since the exact positions

of most indel variations are not known, it is thus best to represent the indel of interest with a range of positions, within which equivalent indels can occur, rather than as a single normalized position. A similar idea was proposed by Krawitz et al. [20].

This chapter proposes UPS-indel, a universal positioning system for indels, whereby every indel variant is represented by a range of positions within which all equivalent indels can occur. This representation is added to the VCF file resulting in a UVCF file containing not only the original indel calling results but also the complete representation of all equivalent indels. The advantage of adding this column of information to the existing VCF file is (1) the original VCF file structure is unchanged so the UVCF file is still compatible with many downstream programs, (2) the UPS-indel notation facilitates the comparison of indels from different VCF files, and (3) for equivalent indels that overlap both coding and noncoding regions, having the range column in the indel calling output would allow a downstream indel annotation system to consider the range rather than a single position, possibly annotating both a coding and noncoding variant. In summary, this work extends the previous work of Krawitz et al. [20] and Assmus et al. [53] by a new coordinate system Universal Positioning System (UPS), a rigorous mathematical proof that all (deletion and insertion) equivalent indels are found, the handling of complex variants, and a simple modification of an input VCF file to produce an output UVCF file containing the indel equivalence information. Results show that UPS-indel identifies more redundant indels than the existing approaches, also enables a comparison between indel calling results produced by different indel callers, and performs better than other state-of-the-art approaches for finding indels in common among call sets.

3.2 Materials and Methods

This section defines some terms frequently used in this chapter.

Alternate Sequence: A sequence that is produced by introducing a specific indel to the reference sequence at a specific position. This is also known as the *mutant sequence*.

Let R be the reference sequence and p be either an insertion or a deletion of a given length that occurs at a given position in the reference sequence. The alternate sequence for insertion is denoted by $R'_I = R + p$ and for deletion by $R'_D = R - p$.

Equivalent Indels: Two indels are considered equivalent if and only if they produce the same alternate sequence. Note that equivalent indels must be of the same type (insertion and deletion) and same length.

Redundant Indels: Equivalent indels that are reported as distinct entries in a VCF file are defined as redundant indels.

Region of Equivalence: This is defined as the range of positions in the reference sequence where equivalent indels occur.

Cyclic Permutation: A permutation $(y_0, y_1, y_2, \dots, y_{n-1}) = f(x_0, x_1, x_2, \dots, x_{n-1})$ where $y_i = x_{(i+k) mod\, n}$ for $0 \leq i \leq n-1$, k can be positive (left cyclic) or negative (right cyclic). For example: for a string "ATCG", the left cyclic permutations are TCGA, CGAT, and GATC; the right cyclic permutations of this string are GATC, CGAT, and TCGA.

Table 3.1: An example of equivalent indels.

Equivalent insertions		Equivalent deletions	
Reference, *R*	GTCTA	Reference, *R*	ACTGTTGTG
Case 1, R'_I	G[TC/+]TCTA	Case 1, R'_D	AC[TGT/-]TGTG
Case 2, R'_I	GT[CT/+]CTA	Case 2, R'_D	ACT[GTT/-]GTG
Case 3, R'_I	GTC[TC/+]TA	Case 3, R'_D	ACTG[TTG/-]TG
Case 4, R'_I	GTCT[CT/+]A	Case 4, R'_D	ACTGT[TGT/-]G

Table 3.1 shows an example of equivalent indels. Observe that all equivalent indels are cyclic permutations of each other (e.g., a cyclic permutation of CT is TC and cyclic permutations of TGT are GTT and TTG) and equivalence continues until there is a mismatch (see Appendix B, Table B.2). This observation leads to the following theorem.

Theorem 1: All equivalent indels in the region of equivalence are cyclic permutations of each other.

Proof: Consider two equivalent indels d_1 and d_2 and the equivalence region R they define.

For insertion within R, the alternate sequences are

$d_1S = Sd_2$

for some nonempty S. For deletion within R, the alternate (possibly empty) sequence is S starting with $d_1S = Sd_2$.

Case 1. For $|S| < |d_1|$, $d_1 = SX$ for nonempty X and $d_1S = SXS = Sd_2$ implies $d_2 = XS$ is a cyclic permutation of $d_1 = SX$.

Case 2. For $|S| = |d_1|$, $d_1 = d_2 = S$.

Case 3. For $|S| > |d_1|$, $S = d_1X$ for nonempty X with $|X| < |S|$, and $d_1d_1X = d_1S = Sd_2 = d_1Xd_2$ implies $d_1X = Xd_2$. Repeating this argument for $d_1X = Xd_2$ eventually reduces X to one of the previous two cases.

Another case for deletion is when R is periodic with period $|d_1|$, having the form

$R = d_1\ d_1 \ldots\ldots\ d_1\ (d_1)_1$ where $(d_1)_1$ is the first symbol of d_1. Then every consecutive subsequence d_2 of R with $|d_1| = |d_2|$ is an equivalent deletion, and d_2 is a cyclic permutation of d_1. **(Q.E.D)**

Corollary. For $|S| > |d_1|$, S must have the form $d_1\ d_1 \ldots\ldots * \ldots\ldots\ d_2\ d_2$ with an equal number of d_1s and d_2s.

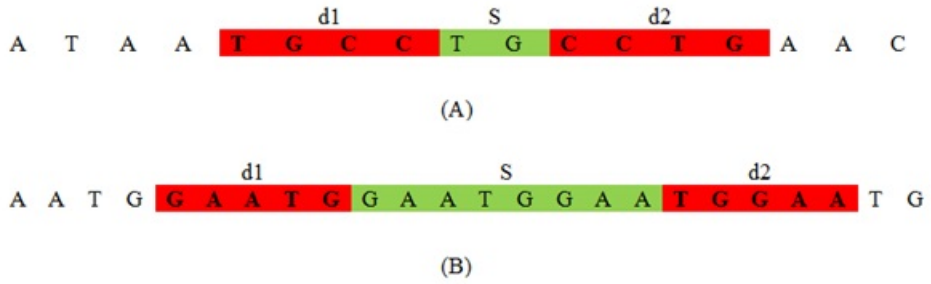


Figure 3.1: Illustration of two cases of Theorem 1. (A): |d1| > |S|, (B): |d1| < |S|.

Figure 3.1 illustrates Theorem 1 with two examples. Based on the theorem, an algorithm called UPS-indel (see Table 3.2) exhaustively increases the range of equivalence as far as possible in both left and right directions from a given indel position. Finally, for each indel in the VCF file, the algorithm reports its range of equivalence, which is called the Universal Positioning System coordinate (UPS-coordinate). Once indels are represented by their UPS-coordinates, identifying redundant indels becomes a trivial task of string comparison (e.g., Fig. 2(A), comparison across the 8th column). Note that since UPS-indel implements Theorem 1, which characterizes indels within an equivalence region, UPS-indel is exhaustive, finding all equivalent indels.

Table 3.2: UPS-indel algorithm.

```
UPS-indel (list_of_indels_in_VCF_file, reference_sequence)
{
For each indel in the list
1.      Extract REF allele and ALT allele from VCF file
2.      pattern ← diff (REF, ALT)
3.      indel ← pattern
4.      eq_indel ← getCyclicPermutationFromLeft(indel)
5.      pos ← position of indel according to the VCF file
6.      position ← pos
7.      upperBound ← position
8.      str ← reference_sequence (position + 1)
9.      while ((indel + str) == (str + eq_indel))
10.         indel ← eq_indel
11.         upperBound++
12.         str ← reference_sequence (position + 1)
13.         eq_indel ← getCyclicPermutationFromLeft(indel)
14.     End while
15.     indel ← pattern
16.     eq_indel ← getCyclicPermutationFromRight(indel)
17.     position ← pos
18.     lowerBound ← position
19.     str ← reference_sequence (position - 1)
20.     while ((str + indel) == (eq_indel + str))
21.         indel ← eq_indel
22.         lowerBound--
23.         str ← reference_sequence (position - 1)
24.         eq_indel ← getCyclicPermutationFromRight(indel)
25.     End while
26.     if (pattern is an insertion)
27.         UPS-coordinate ← +pattern [lowerBound, upperBound]
28.     else //pattern is a deletion
29.         UPS-coordinate ← –pattern [lowerBound, upperBound]
30.     End for
}
```

Note that left and right cyclic permutations are equivalent – there is no difference. In line 2 of the UPS-indel algorithm (Table 3.2), while extracting the pattern from the entries of the RFE and ALT columns of the input VCF file, UPS-indel performs horizontal decompositions of the complex variants and assigns the indel part as the value of the pattern. For example, suppose in the REF column of a VCF entry there is an allele ATAA, and, in the ALT column, there is an allele "AG". In this case, UPS-indel performs horizontal decompositions of the complex variants to produce two separate entries (AT → AG and AA → <empty> meaning that there is a deletion of AA).

UPS-indel is written in C++ and can run on Linux, Windows, or Mac operating systems that have a C++ compiler. The command line version of UPS-indel is available at https://github.com/shabbir005/ups-indel. UPS-indel uses SeqAn, an open source C++ library containing efficient algorithms and data structures to analyze large genome sequences [59]. The input to UPS-indel is a reference chromosome sequence, a VCF file containing a list of indels, an output file name, and a flag to enable/disable horizontal decomposition, for example,

```
./ups_indel example/chr1.fa example/chr1.vcf example/chr1.uvcf -
hd=true
```

This command line produces an output file named chr1.uvcf, containing the UPS-coordinates of all the indels in chr1.vcf when horizontal decomposition is enabled. Figure 3.2(A) shows an example UVCF file.

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	UPS-COORDINATE	INFO
1	61350	rs200672483	TA	T	.	.	-A[61351 - 61361]	RS=200672483;RSPOS=61351;
1	61871	rs368183979	C	CT	.	.	+T[61872 - 61881]	RS=368183979;RSPOS=61871;
1	62094	rs565195228	CT	C	.	.	-T[62095 - 62095]	RS=565195228;RSPOS=62095;
1	62239	rs375143083	TAC	T	.	.	-AC[62240 - 62255]	RS=375143083;RSPOS=62240;
1	62297	rs544370662	T	TCTTC	.	.	+CTTC[62298 - 62302]	RS=544370662;RSPOS=62297;
1	62298	rs372783161	C	CCTTC	.	.	+TTCC[62296 - 62299]	RS=372783161;RSPOS=62298;

(A)

```
[rs36021631, rs397814777, rs571399445]
[rs200975227, rs201682951]
[rs10680837, rs35730918, rs397972024]
[rs34417147, rs370372623, rs375940931, rs386367800, rs386367801]
[rs397710365, rs398074198, rs557264932, rs57380526]
```

(B)

```
Number of Common Indels in example/sample_1.uvcf and example/sample_2.uvcf : 592979
Number of Indels in example/sample_1.uvcf but not in example/sample_2.uvcf : 16864
Number of Indels in example/sample_2.uvcf but not in example/sample_1.uvcf : 2499
```

(C)

Figure 3.2: Different utilities of UPS-indel. (A)UVCF format, (B) redundant indel list, and (C) comparing two uvcf files.

The UVCF file keeps the same content/format as the VCF file, with an additional column that contains the indel's UPS-coordinate information. The interpretation of the UPS-coordinate follows:

Symbols + and – denote insertion and deletion, respectively, followed by the base pairs inserted/deleted from the reference, and the UPS-coordinate (in square brackets).

The UPS-coordinate contains a range of positions in the square brackets representing the region of equivalence for the indel. For example, the UPS-coordinate +CTTC [62298 - 62302] means there is an insertion of CTTC at position 62298, and the same alternate sequence can be produced by inserting TTCC at position 62299, or TCCT at position 62300, and so on.

Once indels are represented by the coordinates produced by UPS-indel, one can easily identify redundant indels within one indel call set or multiple indel call sets. For example, the following command line

```
./ups_generate_redundant_indel_list example/chr1.uvcf
example/redundant_indel_list.txt
```

produces a list of indel groups containing dbSNP IDs of redundant indels (Figure 3.2(B)).

UPS-indel groups all redundant indels together. For example, consider a group [rs34748242, rs59148039] with the UVCF entry shown in Table 3.3. These two indels belong to the same indel type (insertion), have same base pairs inserted (TG), and share the same UPS-coordinate and hence they are considered as equivalent.

Table 3.3: UVCF file for redundant indels.

#CHRM	POS	ID	REF	ALT	QUAL	FILTER	UPS-COORDINATE
1	10009638	rs34748242	T	TTG	.	.	+TG[10009639 - 10009648]
1	10009639	rs59148039	T	TGT	.	.	+TG[10009639 - 10009648]

UPS-indel can compare multiple indel call sets. This utility is particularly useful for generating a high-confidence indel call set by taking the intersection of the results of different indel callers [60], or merging the indel calling results from different tools for a consensus variant caller [61], or

comparing indel call sets generated by different indel callers to determine their relative recall, precision, and accuracy, and to understand the source of their dissimilarities. To use this utility of UPS-indel, after converting two VCF files to UVCF files, one can use the following command to get the comparison result (Figure 3.2(C)), which contains useful statistics for downstream analysis:

```
./ups_compare_uvcf_files example/sample1.uvcf
example/sample2.uvcf example/comparison_result.txt
```

A utility for UPS-indel can produce a filtered UVCF file after removing redundant indels. The following command, for example, can get the filtered UVCF file named out_filtered.uvcf containing nonredundant indels:

```
java GenerateFilteredUVCFFileAfterRemovingRedundantIndel
example/out.uvcf example/redundant_indel_list.txt
```

All of the above-mentioned utilities of UPS-indel are also included in the web version available at http://bench.cs.vt.edu/ups-indel/ (Figure 3.3).

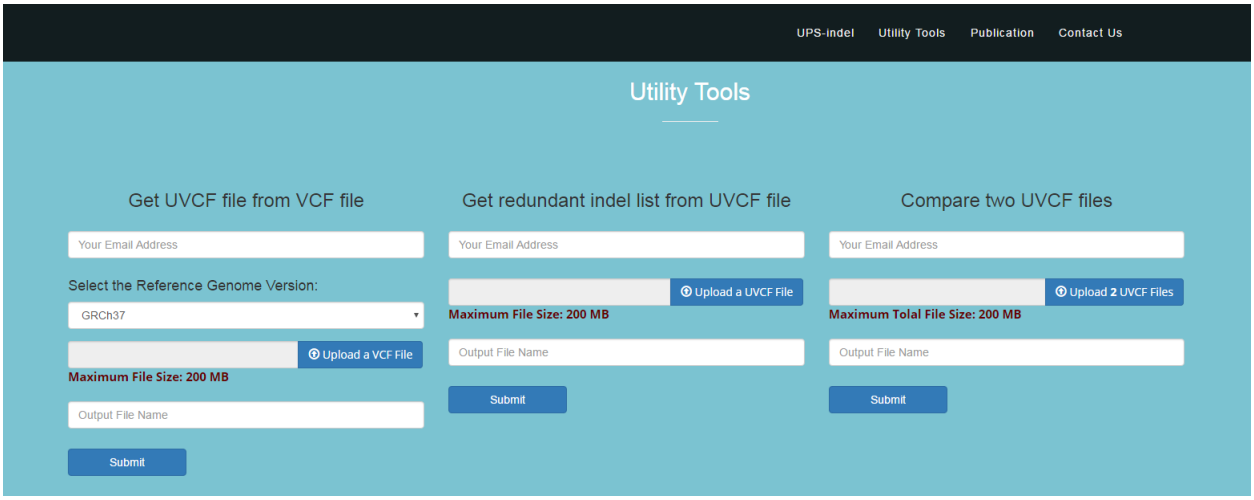


Figure 3.3: The main user interface of UPS-indel.

UPS-indel is compared with other existing tools that also find equivalent indels through variant normalization. These tools include vt normalize (version 0.5) [54], BCFtools (version 1.3) [28], and GATK LeftAlignAndTrimVariants (version 3.5) [29]. Like UPS-indel, all of these tools take a VCF file and the reference genome as input and produce the normalized position of the indels in the VCF file. Another tool, Vindel [47], also finds equivalent indels using a heuristic approach but was not included in the comparison as it uses a flat file as input instead of a VCF file.

A VCF file of dbSNP (version 142, GRCh37p13) and the GRCh37 reference genome were used as the inputs to these tools. The VCF file contains both SNPs and indels, and VCFtools [62] (Version 0.1.14) is used to extract indels from the VCF file. The comparison was extended to the COSMIC data set as well.

There are other tools that could also be considered for comparison. Both VarMatch [55] and RTGTools [63] use a branch and bound algorithm to search for equivalent indels. They are not suitable for processing population-scale indel call sets such as dbSNP and COSMIC because densely packed indels in such data sets make the search space too large to be processed by a branch and bound algorithm [55]. READDI [64] considers repeat-induced ambiguities as well as tool-induced inaccuracies while searching for equivalent deletions using the longest common extension algorithm. This tool is limited to finding deletions only and hence is not included in the comparison for the dbSNP and COSMIC data sets. Nevertheless, in this study, a smaller data set is used to compare UPS-indel with VarMatch (Version available on April 5, 2017), RTGTools (Version 3.7.1), and READDI (Version available on April 5, 2017).

3.3 Results and Discussion

3.3.1 Finding equivalent indels in the dbSNP data set

The input VCF file contains about 8.9 million indels from the human genome. For this input, UPS-indel produces the UVCF file and the other three tools, vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants, generate a normalized VCF file. These three tools perform left normalization of indels and output a left normalized representation. Therefore, for these three tools, two indels are equivalent if and only if they satisfy the following conditions:

(1) Both indels are of the same type (insertion or deletion).

(2) Both indels share the same pattern after normalization: [value of the REF column in the normalized VCF file – value of the ALT column in the normalized VCF file – value of the POS column in the normalized VCF file]. Note that one might think that considering the position should suffice because, after normalization, equivalent indels should have the same position in the VCF file. However, the example in Table 3.4 shows that indels rs371246544 and rs71724031 have the same normalized position but are not equivalent.

Table 3.4: An example explaining why considering only normalized position does not suffice for identifying redundant indels for vt normalize and BCFtools.

VCF Entry for input				
#CHRM	POS	ID	REF	ALT
1	39549110	rs371246544	AT	ACATAC
1	39549111	rs71724031	T	TAC
Entry in the normalized VCF				
#CHRM	POS	ID	REF	ALT
1	39549111	rs371246544	T	CATAC
1	39549111	rs71724031	T	TAC

The comparison is based on the criterion: the redundant indel ratio = $\frac{total\ number\ of\ redundant\ indels - total\ number\ of\ redundant\ indel\ groups}{total\ number\ of\ indels}$,

where the numerator is the total number of redundant indels reported since only one indel from each redundant indel group should be reported in the output and the remaining should be considered as redundant.

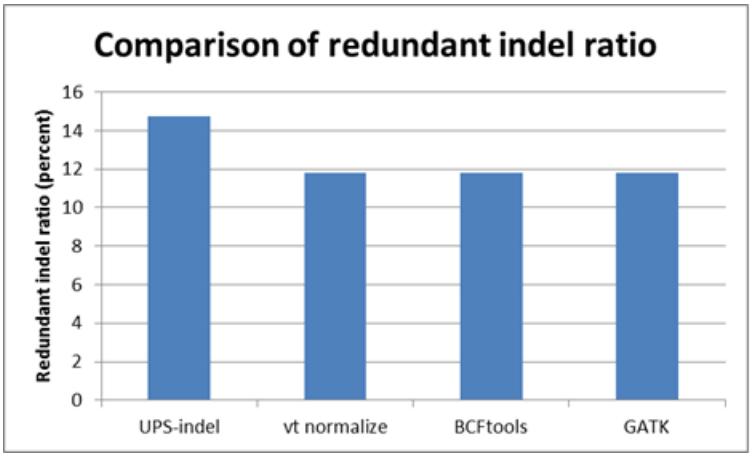


Figure 3.4: Comparison of the tools based on redundant indel ratio for the dbSNP data set.

Figure 3.4 shows the comparison of the redundant indel ratios reported by UPS-indel, vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants for indels in the dbSNP data set. For the entire human genome, UPS-indel identified approximately 15% redundant indels (see Appendix B, Table B.3 and Appendix B, Figure B.1 for chromosome-wise comparison), as compared to 11.82% by

vt normalize, 11.82% by BCFtools, and 11.81% by GATK LeftAlignAndTrimVariants. At the chromosome level, UPS-indel identified about 3% more redundant indels than the other three tools.

Examining the sets of redundant indels detected by UPS-indel and the other tools shows that vt normalize and BCFtools produce exactly the same results for all chromosomes. Moreover, all the redundant indels detected by vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants are also detected by UPS-indel, as shown in Figure 3.5. Further, for all chromosomes, UPS-indel identified a total of 456,352 more redundant indels than the other tools. As proved in the methods, UPS-indel identifies all the redundant indels, the comparison result shows that the other three tools are not exhaustive in finding all the redundant indels.

Why are several indels found as redundant by UPS-indel but not by other tools? An investigation shows that these equivalent indels are missed by the other tools because, due to the computation time limit, they cannot exhaustively search every cyclic permutation at every feasible position as is done by UPS-indel. For example, long multiallelic indels are not considered by default for normalization. Had the tools considered these indels separately, they would have been able to find an equivalent indel located at a different position. For this situation, UPS-indel splits the VCF entry into multiple entries by default and considers each of the indels separately while finding redundant indels. Table 3.5 provides such an example.

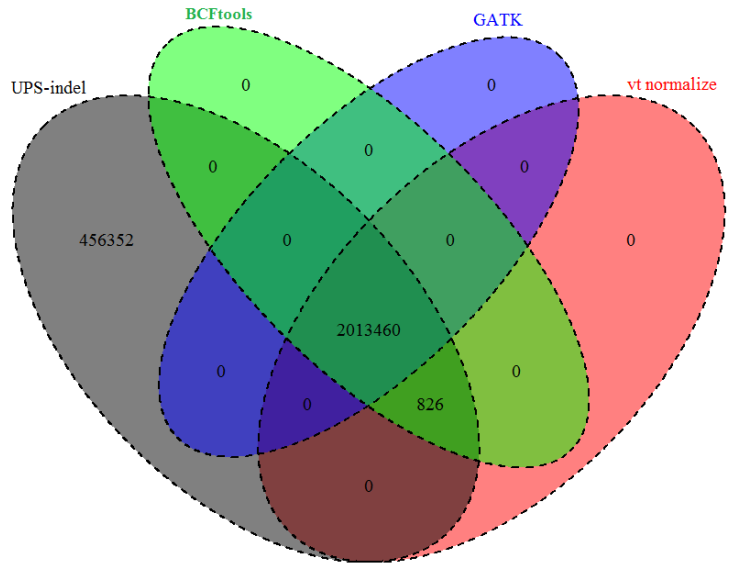


Figure 3.5: Venn diagram to compare the number of redundant indels detected by UPS-indel and other tools. (Venn Diagrams are generated using the R package VennDiagram [67].)

For the indel shown in Table 3.5 (panel A), no normalization was done by vt normalize, BCFtools, or GATK LeftAlignAndTrimVariants. UPS-indel splits the entry into three indels and finds the UPS-coordinate for each of them separately (Table 3.5, panel B). Splitting the VCF entry and considering the indels separately, UPS-indel managed to find another indel equivalent to one of the indels (Table 3.5, panel C). Therefore, UPS-indel reports indels with id rs374587598 and rs60022176 as redundant.

Table 3.5: Example of multiallelic insertion type indels missed by other tools but detected as redundant by UPS-indel.

VCF Entry					
#CHRM	POS	ID	REF	ALT	
1	724188	rs60022176	A	AATGGA, AATGGAATGGAATGGA, AATGGAATGGG	
UVCF Entry					
#CHRM	POS	ID	REF	ALT	UPS-COORDINATE
1	724188	rs60022176	A	AATGGA	+AATGG[724138 - 724189]
1	724188	rs60022176	A	AATGGAATGGAATGGA	+AATGGAATGGAATGG[724138 - 724189]
1	724188	rs60022176	A	AATGGAATGGG	+ATGGAATGGG[724189 - 724189]
Redundant indels					
#CHRM	POS	ID	REF	ALT	UPS-COORDINATE
1	724137	rs374587598	T	TAATGG	+AATGG[724138 - 724189]
1	724188	rs60022176	A	AATGGA	+AATGG[724138 - 724189]

The example in Table 3.5 is for insertion; an example for deletion is illustrated in Table 3.6.

Table 3.6: Example of multiallelic deletion type indels missed by other tools but detected as redundant by UPS-indel.

VCF Entry					
#CHRM	POS	ID	REF	ALT	
1	7552657	rs376707888	GTG	G, GTGCA	
UVCF Entry					
#CHRM	POS	ID	REF	ALT	UPS-COORDINATE
1	7552657	rs376707888	GTG	G	-GT[7552657 - 7552658]
1	7552657	rs376707888	GTG	GTGCA	+CA[7552658 - 7552658]

Redundant indels					
#CHRM	POS	ID	REF	ALT	UPS-COORDINATE
1	7552656	rs139294420	CGT	C	-GT[7552657 - 7552658]
1	7552657	rs376707888	GTG	G	-GT[7552657 - 7552658]

In addition to the scenario mentioned above, GATK LeftAlignAndTrimVariants does not normalize any of the multiallelic indels regardless of the size which is also mentioned in [54]. Table 3.7 shows an example of this occurrence explaining why GATKLeftAlignAndTrimVariants finds a fewer number of redundant indels than vt normalize and BCFtools.

Table 3.7: Example of a multiallelic indel that is normalized by vt normalize and BCFtools but not by GATKLeftAlignAndTrim.

VCF Entry for dbSNP				
#CHRM	POS	ID	REF	ALT
1	823905	rs397728418	AA	A, AAA
VCF Entry for GATK LeftAlignAndTrimVariants				
#CHRM	POS	ID	REF	ALT
1	823905	rs397728418	AA	A, AAA
VCF Entry for vt normalize and BCFtools				
#CHRM	POS	ID	REF	ALT
1	823903	rs397728418	GA	G, GAA

One might think that decomposing multiallelic indels into several biallelic indels produces the same results as UPS-indel for the normalization tools. To check this, the decompose utility of vt was used to perform a vertical decomposition of multiallelic indels into biallelic indels. Applying vt normalize to the decomposed indels could not find equivalent indels for complex variants, whereas UPS-indel is able to find the equivalent indels. Table 3.8 shows an example of this occurrence. Since vt normalize and BCFtools produce exactly the same results, these complex variants are missed by BCFtools as well.

Table 3.8: Example of a complex variant that is missed by vt normalize but detected as redundant by UPS-indel.

VCF Entry (A)					
#CHRM	POS	ID	REF	ALT	
1	2273131	rs369694942	GAAA	G	
1	2273140	rs373243812	AAAAA	AG	
UVCF Entry (B)					
#CHRM	POS	ID	REF	ALT	UPS-COORDINATE
1	2273131	rs369694942	GAAA	G	-AAA[2273132 - 2273147]
1	2273140	rs373243812	AAAAA	AG	-AAA[2273132 - 2273147]

In the example shown in Table 3.8, VCF entries for the indels with ids rs369694942 and rs373243812 remain the same in the input and the output for vt normalize (Panel A), i.e., no normalization is done. Here the second indel (rs373243812) is a complex variant containing both an SNP (A → G) and a deletion of length three (AAA) and is ignored by vt normalize. However, UPS-indel performs a horizontal decomposition of the complex variant to produce two separate entries (AA → AG and AAA → <empty>) and finds the equivalent indel with id rs369694942 having a deletion of length three (AAA) in the UPS-coordinate 2273132 to 2273147 (Panel B).

As defined in [24], complex variants come in two forms: (1) MNP and (2) a clumped indel. In MNP, (1) the lengths of the reference and alternate sequences are greater than one and (2) the nucleotides involved in the two sequences differ. In a clumped indel, on the other hand, there is a clumping of nearby variants and the sequences need not involve different base pairs. Table 3.9 shows an example of MNP and of a clumped indel found in dbSNP (version 142).

Table 3.9: An example of the two types of complex variant.

#CHRM	POS	ID	REF	ALT	Type
1	565003	rs386627398	ATAT	AAAC	MNP
1	884423	rs386627415	AGCA	AACAACAGCAAAG	Clumped Indel

UPS-indel decomposes complex variants based on the best-predicted outcome using the Needleman-Wunsch algorithm. This approach was used by Li et al. [65] to decompose complex variants into individual events.

3.3.2 Finding equivalent indels in the COSMIC data set

UPS-indel was used to find redundant indels in the COSMIC (Catalogue Of Somatic Mutations In Cancer) data set, the world's most comprehensive resource for exploring the impact of somatic mutations in human cancer [66]. With data collected for more than 2,500 human cancers, this archive describes millions of coding mutations, noncoding mutations, and other gene expression variants across the human genome.

For all chromosomes in the COSMIC data set, UPS-indel identified 28.17% and 13.11% redundant indels in the COSMIC coding and noncoding indel data sets, respectively, which are higher than the redundant indel ratios reported by the other tools. Figure 3.6 shows the comparison of the redundant indel ratios reported by UPS-indel, vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants for both the COSMIC coding and noncoding data sets. Comparisons for chromosome-wise redundant indel ratios among the tools are given in Appendix B (See Table B.4 and Figure B.2 for COSMIC coding and Table B.5 and Figure B.3 for noncoding indels).

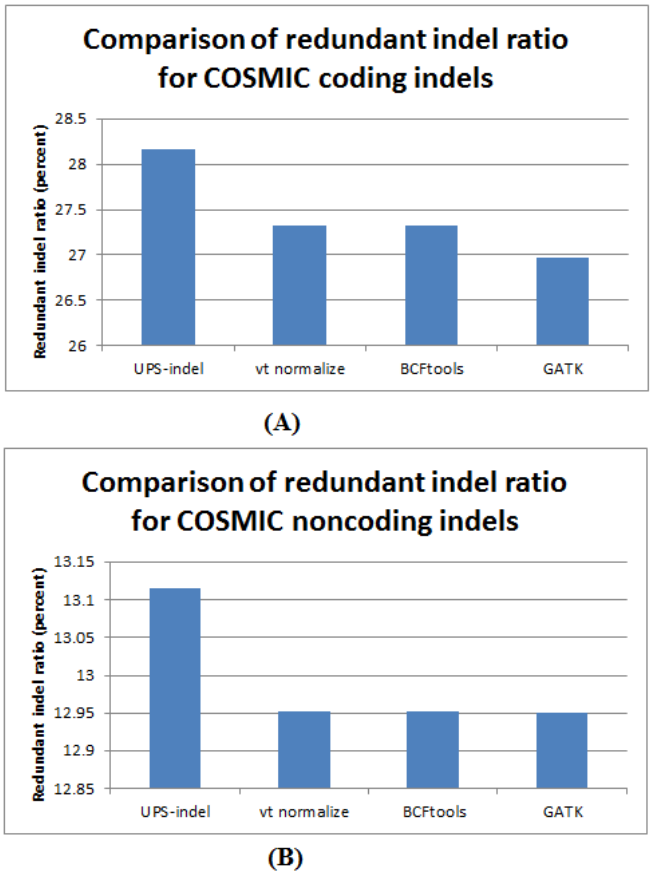


Figure 3.6: Comparison of redundant indel ratio for (A) COSMIC coding and (B) COSMIC noncoding indels.

Similarly, examining the sets of redundant indels identified by the tools, Figure 3.7 shows that for both the COSMIC coding and noncoding indels, UPS-indel identified all the redundant indels detected by the other tools. In addition to that, for the whole genome, 2,118 (Figure 3.7A) and 553 (Figure 3.7B) unique redundant indels for COSMIC coding and noncoding indels, respectively, are detected by UPS-indel but missed by other tools.

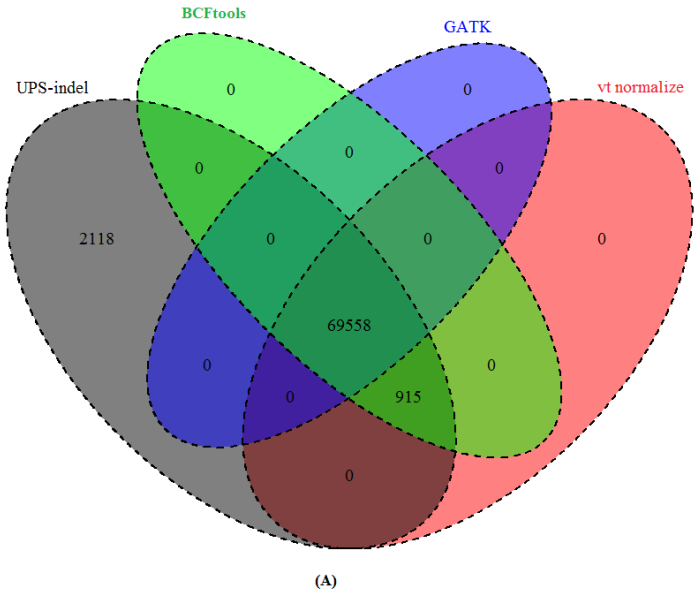


Figure 3.7: Venn diagram to compare the number of redundant indels detected by UPS-indel and other tools in (A) COSMIC coding and (B) COSMIC noncoding indel data sets.

As for dbSNP, the reason why some COSMIC coding and noncoding indels were considered as redundant by UPS-indel but missed by other tools is that, in the normalized VCF for these other tools, redundant indels must contain the same pattern: [value of the REF column in the normalized VCF file – value of the ALT column in the normalized VCF file – value of the POS column in the normalized VCF file]. The reason for this pattern match restriction was given earlier. In Table 3.10, all tools except UPS-indel missed the indel with id COSM5068028 in the redundant indel group consisting of indels with id COSM3732389 and id COSM5348791, because of not having the same pattern. Therefore, it might be assumed that only normalized position should be considered to group them together. However, then the indel with id COSM3685916 would be placed in the same group, although it is a deletion type indel whereas the others are insertion type indels, and also the resultant sequences are different. UPS-indel groups the indels correctly by placing indels with ids COSM5068028, COSM3732389, and COSM5348791 in the same redundant indel group as they have the same base pair inserted, have the same region of equivalence, and also are of the same indel type.

Table 3.10: Example of COSMIC indel that is missed by other tools but detected as redundant by UPS-indel.

VCF Entry for COSMIC					
#CHRM	POS	ID	REF	ALT	
1	150917623	COSM5068028	TG	TGG	
1	150917623	COSM3732389	T	TG	
1	150917623	COSM3685916	TG	T	
1	150917624	COSM5348791	G	GG	
VCF Entry for other tools after normalization					
#CHRM	POS	ID	REF	ALT	
1	150917623	COSM5068028	TG	TGG	
1	150917623	COSM3732389	T	TG	
1	150917623	COSM3685916	TG	T	
1	150917623	COSM5348791	T	TG	
UVCF Entry for UPS-indel					
#CHRM	POS	ID	REF	ALT	UPS-COORDINATE
1	150917623	COSM5068028	TG	TGG	+G[150917624 - 150917632]
1	150917623	COSM3732389	T	TG	+G[150917624 - 150917632]
1	150917623	COSM3685916	TG	T	-G[150917624 - 150917631]
1	150917624	COSM5348791	G	GG	+G[150917624 - 150917632]

GATKLeftAlignAndTrimVariants found fewer redundant indels than other tools because GATKLeftAlignAndTrimVariants does not consider very large indels for normalization. For example, the indels with ids COSM5196837 and COSM5066846, which are deletions of length 371bps and 222 bps, respectively, are not considered by GATKLeftAlignAndTrimVariants for normalization. The reason is that GATK LeftAlignAndTrimVariants uses 200 bps as the default size of the sliding window on the reference (the parameter `--reference_window_stop`) while left aligning the alleles which is smaller than the length of the missed deletions.

These tools are also compared in terms of average running time taken to process the whole genome VCF file of dbSNP (version 142, size 16.7GB) for normalization (by vt normalize, BCFtools, and GATKLeftAlignAndTrimVariants) or for generating the UPS-coordinate (by UPS-indel). All tools were run on a desktop computer having an Intel Core i7-2600 CPU with eight cores (at 3.40 GHz) and 16GB of RAM. Among these tools, vt normalize is the fastest, taking 12 minutes 27 seconds, followed by BCFtools (12 minutes 35 seconds), UPS-indel (17 minutes 10 seconds), and GATK LeftAlignAndTrimVariants (21 minutes 55 seconds). Since UPS-indel searches for equivalent indels exhaustively and is theoretically rigorous, the computation time is a little higher than that for other heuristic normalization tools such as vt normalize and BCFtools.

3.3.3 Evaluating UPS-indel's performance in comparing different indel call sets

In genomic research related to indel calling, an important step in downstream analysis is to compare multiple indel call sets for (1) generating a highly accurate benchmark indel call set by taking the intersection of multiple call sets as done by Zook et al. [60] for the sample NA12878, (2) merging the call sets of different indel callers in a consensus caller as done by Trubetskoy et al. [61] for exome data, and (3) evaluating the accuracy of a newly proposed indel calling tool by comparing its indel call set with the benchmark call set. Comparing different indel call sets is also a common step in studies comparing the performance of different indel callers as done in [56],[39], and [67]. Different indel callers having different representations of the same indel complicates the comparison of different indel call sets. In addition to the strict matching of indels, as mentioned earlier, a naïve but previously commonly used approach to compare multiple indel-calling results is based on a simple distance criterion, that is, indels are considered to be equivalent if they are within a distance threshold (e.g., ±5bp or ±25bp). For example, the original 1000 Genomes project used ±25bp to compare multiple indel-calling results [57]. To illustrate the advantage of using a

UVCF file instead of a distance criterion or normalized VCF for comparing multiple VCF files, the alignment file for chromosome 11 of a single sample (HG00851) was picked up from the 1000 Genomes project and five indel callers, Dindel [26] (Version 1.0.1), GATK Unified Genotyper [29] (Version 3.4), GATK Haplotype Caller (Version 3.4) [37], Platypus [38] (Version 0.7.9.1), and Pindel [32] (Version 0.2.5), were used to produce VCF files for indels. The resultant VCF files were compared to determine the number of common indels from these five tools using three different approaches, namely a distance-based approach, comparing the VCF files normalized by vt normalize and GATK LeftAlignAndTrimVariants, and comparing the UVCF files produced by UPS-indel. For the distance-based approach, two indels are considered equivalent if (1) they belong to the same indel type (either both are insertion type or both are deletion type), (2) have the same base pairs inserted/deleted, and (3) are in close proximity (within ±5 bps from each other). For the normalized VCF files and UVCF files, the same approach was used as discussed earlier for finding redundant indels.

First, the VCF files produced by the five indel calling tools were compared to find overlap among them to determine the number of common indels using the distance-based approach. In the second step, the VCF files of the five indel calling tools were normalized using vt normalize and GATK LeftAlignAndTrimVariants separately. For this sample, both normalization tools produced the same normalized VCF files. The normalized VCF files of five indel calling tools were compared to determine the common number of indels. Finally, UPS-indel was used to produce the UVCF files for the five indel calling tools and these UVCF files were compared to determine the common number of indels.

The result shows that the distance-based approach found 584 indels in common from the five indel calling tools while 5,514 and 5,575 common indels were found by the normalized VCF and UPS-indel UVCF approaches, respectively. This demonstrates the better suitability of UPS-indel, compared to distance based or existing normalization-based approaches, for comparing multiple VCF files. Further investigation revealed that indels that are missed by the normalization tools are complex variants that are skipped by normalization tools but processed by UPS-indel. Note that this small number (61) of common indels identified by UPS-indel, but missed by the normalization tools, is based on a single chromosome of a single sample only, and much better performance of

UPS-indel would be expected for the whole genome, as observed for the dbSNP and COSMIC data sets.

As mentioned earlier, the tools VarMatch [55], RTG Tools [63], and READDI [64] are also used for comparing indel call sets. However, VarMatch and RTG Tools, which use a branch and bound algorithm, are not suitable for population-scale indel call sets like dbSNP and COSMIC due to densely packed indels in those call sets [55]. READDI processes deletions only. These tools are compared with UPS-indel (using the deletion call set of Platypus containing 14,438 deletions for chromosome 11 of the above mentioned single sample from the 1000 Genomes project as the baseline) on the deletion call sets of Dindel, GATK Unified Genotyper, GATK Haplotype Caller, and Pindel as the query call set to check overlap with the baseline. Table 3.11 shows the comparison for finding the number of true positives.

Table 3.11: Comparison between VarMatch, RTG Tools, READDI, and UPS-indel based on the number of true positives found between the baseline and query call sets from chromosome 11 of an individual.

Variant Caller Name	**VarMatch (EVQ mode)**	**RTG Tools**	**READDI**	**UPS-indel**
Dindel	8,933	8,933	8,796	8,973
GATK Haplotype Caller	11,113	11,113	10,954	11,129
GATK Unified Genotyper	7,734	7,734	7,563	7,734
Pindel	6,507	9,893	9,524	9,836

Table 3.11 shows that in all cases except for Pindel, UPS-indel finds more common indels than the state-of-the-art tools when comparing multiple indel call sets. These tools are heuristic and therefore ignore indels that violate a particular heuristic criterion. For example, READDI searches for equivalent indels in an indel's neighboring region defined by the neighborhood size, and RTG Tools uses a cutoff strategy when the search space is too large. UPS-indel, on the other hand, exhaustively searches for and finds all equivalent indels, thus finds more common indels than the aforementioned tools. Why there are true positives identified by RTG Tools but missed by UPS-indel is because the current version of UPS-indel does not consider haplotypes formed by the combination of neighboring indels. Consider the following example in Table 3.12.

Table 3.12: A combination of variants identified by RTG Tools but missed by UPS-indel.

Reference	1	2	3	4	5	6	7	8
	T	G	C	C	G	C	T	A
Deletion D1: Delete CC from position 3 and 4	TGGCTA							
Deletion D2: Delete C from position 6	TGCCGTA							
Deletion D1D2 (Combination of D1 and D2): Delete CC from position 3 and 4, also delete C from position 6	TGGTA							
Deletion D3: GCCGC -> GG starting from position 2	TGGTA							

In the Table 3.12 example, D3 is equivalent to D1D2. UPS-indel does not consider the combination of D1 and D2 but rather considers them separately. Therefore, UPS-indel is not able to discover the aforementioned equivalence of the resultant haplotypes. This explains why UPS-indel missed 57 out of 9,893 deletions (where the deletions in a combination were separated by at most 45 bps) in the Pindel call set that are identified as equivalent by RTG Tools as shown in Table 3.11. Nevertheless, there are 103 indels that are found as true positive by UPS-indel but missed by RTG Tools.

4. Conclusion

This chapter describes UPS-indel, a tool that creates a universal positioning system called UPS-coordinates for all indels listed in a VCF file and exhaustively finds all equivalent indels. The UPS-coordinate is a range of positions where all indels equivalent to a specific indel can occur. Since equivalent indels produce the same mutant sequence and thus have the same biological effect, reporting them as separate indels causes data redundancy and may artificially inflate the statistics of indel variations. Under the proposed universal positioning system, all equivalent indels have the same UPS-coordinate which avoids possible annotation ambiguity. Therefore, by checking the UPS-coordinate, one can easily filter out redundant indels from variant databases. UPS-indel is robust enough to handle complex variants and is able to detect more redundant indels than the currently existing approaches. UPS-indel could be widely used for easy and accurate systematic comparison of indels generated by different indel calling programs or deposited in databases. By eliminating the indel redundancy issue, this work offers the community the proposed universal positioning system to represent indels (so as to avoid ambiguity), which can greatly improve various downstream genomic analyses related to indels.

Chapter 4

Uncovering missed indels by leveraging unmapped reads

In current practice, Next Generation Sequencing (NGS) applications start with mapping or aligning short reads to a reference genome, with the aim of identifying genetic variants. Although existing alignment tools have shown great accuracy in mapping short reads to a reference genome, a significant number of short reads still remain unmapped and are often excluded from downstream analyses thereby causing nonnegligible information loss in the subsequent variant calling procedure. This chapter describes Genesis-indel, a computational pipeline that explores the unmapped reads to identify novel indels that are initially missed in the original procedure. Genesis-indel is applied to the unmapped reads of 30 breast cancer patients from TCGA. Results show that the unmapped reads are conserved between the two subtypes of breast cancer investigated in this study and might contribute to the divergence between the subtypes. Genesis-indel identifies 72,997 novel high-quality indels previously not found, among which 16,141 have not been annotated in the widely used mutation databases. Statistical analysis of these indels shows significant enrichment of indels residing in oncogenes and tumor suppressor genes. Functional annotation further reveals that these indels are strongly correlated with pathways of cancer and can have high to moderate impact on protein functions. Additionally, some of the indels overlap with the genes that do not have any indel mutations called from the originally mapped reads but have been shown to contribute to the tumorigenesis in multiple carcinomas, further emphasizing the importance of rescuing indels hidden in the unmapped reads in cancer and disease studies.

4.1 Introduction

Next Generation Sequencing (NGS) facilitates generation of an enormous number of short reads and allows the identification of genomic mutations that cause phenotype changes and genetic diseases such as Mendelian disorders [8], acute myeloid leukemia (AML) [10], and lung cancer [13]. Applications analyzing the NGS reads typically start with mapping the short reads against a reference genome and then based on the mapped reads, determine the genetic mutations such as single nucleotide polymorphism (SNP) and sequence variants such as insertion and deletion (indel) of bases. Many alignment algorithms have been developed to map the short reads to the reference genome, including MAQ [68], SOAP [69], BWA [70], Bowtie [71], Bowtie2 [72], SNAP [73], and SOAP2 [74], to name a few. Although these alignment tools are very efficient in aligning short

reads, a nonnegligible fraction of reads are left unmapped due to (1) structural variants longer than the allowed number of gaps and mismatches by the mapper, (2) sequencing error, or (3) sample contamination [75]. In current practice, these unmapped reads are not used for variant calling and downstream analyses, and thus mutations harbored in these unmapped reads are not processed in identifying the associations with any disease such as cancer. However, as shown in Figure 4.1, some of the hidden or missing mutations can contain the key for understanding the molecular mechanisms of genetic diseases or cancer and might be used as markers for disease or cancer diagnosis and prognosis.

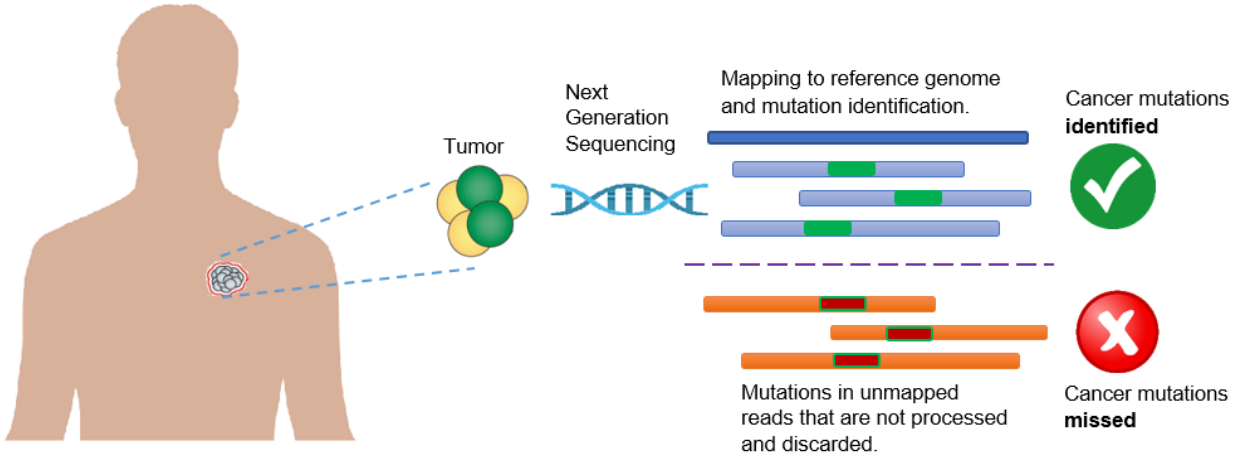


Figure 4.1: Limitation of current practice in cancer research which discards unmapped reads and therefore misses important mutations containing real biological signal.

The consequence of missing the mutations contained in the unmapped reads can lead to inaccurate downstream analyses such as characterizing the tumor evolution in a cancer patient. Some of these missed mutations can be the hallmark of tumors and can be useful for targeted therapy. Therefore, it is critical to identify the mutations in those regions for clinical decision-making as well as for guided personal treatment [76, 77]. With this objective in mind, it is essential to inspect the unmapped reads previously excluded from analyses to ensure that none of these essential mutations are missed in those regions of interest.

This chapter describes Genesis-indel, a computational pipeline to explore unmapped reads for the systematic identification of indels missed in the original alignments. Note that this pipeline focuses on indels only, the second most abundant form of genetic variation in human populations [1-3]. Despite being a common form of genetic variation in humans, indels have not been studied as thoroughly as SNPs, though they have been identified associated with diseases such as cystic

fibrosis [5], Fragile X Syndrome [6], acute myeloid leukemia [10-12], and lung cancer [13]. In addition, insertion of transposable elements such as Alu can affect gene function and change gene expression [15]. Genesis-indel is applied to explore unmapped reads of 30 breast cancer patients from The Cancer Genome Atlas (TCGA) [78] and identify indels hidden in the unmapped reads of these patient genomes. Results show that unmapped reads can be used to cluster samples to different cancer subtypes. In addition, Genesis-indel can successfully curate the unmapped reads and detect small to large novel high-quality indels that are missed previously and some of these indels are specific to a particular subtype of breast cancer. Functional annotation of the newly identified indels shows that the indels found from unmapped reads are strongly correlated with cancer pathways and may play an important role in cancer progression. Additionally, some of the indels overlap with the genes that do not have any indel mutations called from the originally mapped reads but have been shown to contribute to the tumorigenesis in multiple carcinomas, further emphasizing the importance of rescuing indels from the unmapped reads in cancer and disease studies. Therefore, this study complements the current procedure of read alignment and variant calling, shedding light on the hidden information in the unmapped reads that will be useful in downstream analysis.

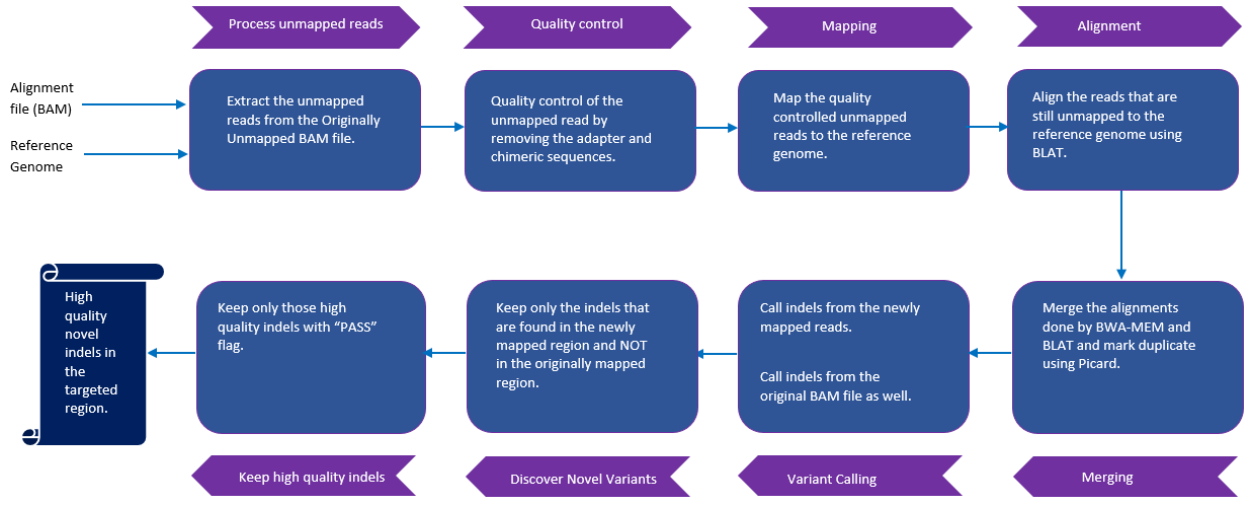


Figure 4.2: Genesis-indel workflow. The input to Genesis-indel is the alignment file (BAM file) and the reference genome (FASTA format). First, the unmapped reads are extracted from the input BAM and passed to the quality control module. After quality control, the reads are mapped to the reference genome using BWA-MEM. The reads that still remained unmapped are aligned using BLAT. In the merging step, the output of BWA-MEM and BLAT are merged and duplicates are marked using Picard. The merged alignment is then passed to the variant calling module followed by quality filtering of the indels. Finally, the output contains novel high-quality indels rescued from the originally unmapped reads.

4.2 Materials and Methods

4.2.1 The Genesis-indel pipeline

Genesis-indel is designed to leverage unmapped reads from an alignment with the goal to rescue indels that are hidden in the discarded unmapped reads. Figure 4.2 shows a schematic representation of the Genesis-indel workflow. The input to Genesis-indel is the alignment file (BAM file) of the patient genome and the reference genome. In the pre processing step, Genesis-indel extracts the unmapped alignment by checking the alignment flag using SAMtools (version 1.4) [28]. From this, it extracts the Originally Unmapped reads using SAMtools and stores the reads in a FASTQ file. This FASTQ file is then processed by Trimmomatic (version 0.36) [79] to do the quality control of the unmapped reads by removing adapter sequences. In this experiment, the Illumina adapter, TruSeq2 for single-end reads, is removed. Moreover, low quality or N bases where the base quality is below 3 are removed from both ends of the reads (LEADING:3, TRAILING:3). Reads are scanned with a 4-base wide sliding window and are cut when the average quality per base drops below 15 (SLIDINGWINDOW: 4:15). Reads with length below 36 bases are dropped (MINLEN:36). These quality controlled single-end reads are used as the input to the mapper in the next step.

The quality controlled unmapped reads are mapped to the reference genome using BWA-MEM (version 0.7.15-r1140) [80], a sensitive mapper to map reads with indels. After the reads are aligned by BWA-MEM, some reads still remain unmapped. These reads are aligned to the reference genome using BLAT (BLAST-Like Alignment Tool) [81], another sensitive local alignment tool. At the end of this step, the alignments from BWA-MEM and BLAT are merged. The resultant alignment is sorted and indexed using SAMtools and duplicates are marked in the newly mapped reads using MarkDuplicates tool from Picard (version 1.65) [36]. After read alignment and marking duplicates, indels are called using Platypus (version 0.7.9.1). Separately, indels are also called from the original (input) BAM file. Indels found only in the newly mapped reads and not in the original alignment are reported as novel indels. After identifying the novel indels, another step of filtering is done to keep only the high-quality indels, i.e., the indels that are called with high confidence by Platypus. Therefore, only the indels with the “PASS” flags are reported at the final step. These are the Novel High-Quality indel (NHQ indels) reported in the final output and selected for downstream analysis.

4.2.2 Preparing a list of oncogene and tumor suppressor genes

A list of oncogenes and tumor suppressor genes is obtained from an online resource [82], a list compiled from the CancerGenes [83]. While preparing the list, if a gene is marked as both an oncogene and a tumor suppressor gene in CancerGenes, a literature search is performed to determine the gene's role in tumor development. Any gene with an ambiguous role as an oncogene or tumor suppressor gene is excluded from the list. The final list contains 79 oncogenes and 63 tumor suppressor genes. The start and end positions of the genes are obtained from GENCODE (version 28 lift37). Appendix C Table C.1 and C.2 contains the list of the genes and their positions.

4.2.3 Software availability and system requirements

Genesis-indel is implemented in C++ and can run on any operating systems that have a C++ compiler. The source code and the command line version of Genesis-indel are freely available at https://github.com/mshabbirhasan/Genesis-indel. Users are welcome to report bugs and provide comments through the issue tracker on GitHub. The README describes the command line options available in Genesis-indel with examples. Although Genesis-indel uses BWA-MEM as the mapper and Platypus as the default variant caller, future version will allow the user flexibility to customize the program and use the mapper and caller of their choice by making small modifications to the tool settings.

4.2.4 Data availability

No new data sample is generated for this study. The alignment file (BAM) for the 30 breast cancer patients are obtained from The Cancer Genome Atlas (TCGA) project (https://portal.gdc.cancer.gov/). Appendix C Table C.3 lists the TCGA Sample Barcode and alignment filename for the patients. The reference genome used is Homo_sapiens_assembly19.fasta, the same reference used by TCGA to align the reads. The annotation of the genes is collected from GENCODE (version 28 lift37). All other data supporting the findings of this study are available in this chapter and in Appendix C. These data are also available from the author upon request.

4.3 Results

Genesis-indel is used to identify the novel high-quality indels from the alignment (BAM files) of 30 breast cancer patients deposited in TCGA. These BAM files were originally produced by

mapping the raw sequencing reads of these patients to the human reference genome using BWA [70].

4.3.1 Existence of a nonnegligible number of originally unmapped reads

The alignment file of each patient sample is processed by SAMtools [28] to extract the "Originally Unmapped" reads. For a given individual investigated here, the number of unmapped reads ranges from 6.6 to 74 million (average = 31.86 million). As shown in Figure 4.3, the unmapped reads constitute an average of 5% of the total reads (altogether there are more than 955 million reads unmapped for 30 patient samples) in the original alignment files provided by TCGA. Genesis-indel targets these discarded reads to rescue the indels missed in the original alignment.

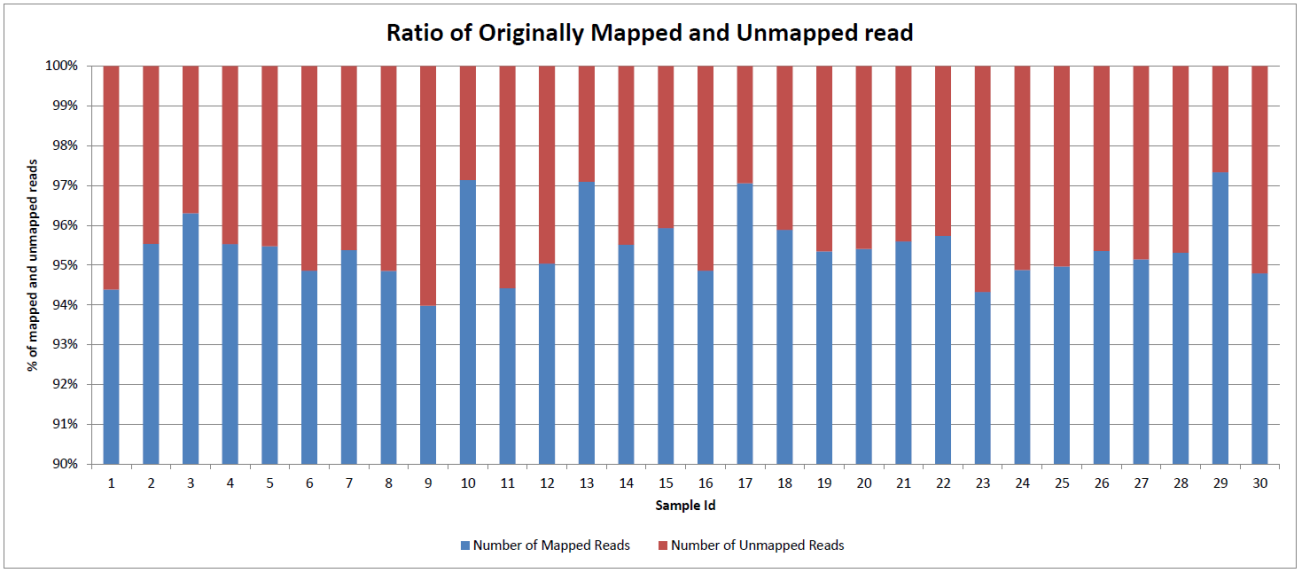


Figure 4.3: Percentage of mapped and unmapped reads in the original alignment files of the 30 breast cancer patients collected from TCGA.

4.3.2 Quality control of the unmapped reads

The extracted unmapped reads are processed for quality control. First, the unmapped reads from all samples are combined and passed to FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/) to get various statistics of the reads. According to the report produced by FastQC, the originally unmapped reads have some quality issues such as (1) overall poor per base sequence quality, (2) a poor score for per sequence quality, (3) overrepresentation of "N" contents, (4) overrepresentation of Illumina Paired-End PCR Primer 2 due to PCR over-amplification, and (5) other adapter contents (Appendix C Figures C.1(a),

C.2(a), and C.3(a)). In most cases, reads that are contaminated with adapter sequences are simply not mapped because of sequencing errors in the adapter sequences. Therefore, removing these contaminated sequences is expected to improve the quality of the unmapped reads. For this reason, Trimmomatic [79] is applied to the combined unmapped reads from all samples and then FastQC is used again to assess the quality of the reads. As shown in Appendix C Figures C.1(b), C.2(b), and C.3(b), after trimming adapter sequences, many issues were fixed and the quality of the unmapped reads improved significantly. Although there is a low-quality issue with some k-mer noise at the 3' end of the reads (Appendix C Figure C.4), the mapping is not affected by these k-mers as they are not mapped to the reference genome and hence are discarded during the alignment step. After the quality control by Trimmomatic, for the individuals investigated here, 29.29% to 89.5% of the originally unmapped reads are retained (average = 67.68%), constituting around 647 million reads.

4.3.3 Mapping the quality controlled unmapped reads

After quality control, the unmapped reads are mapped to the reference genome using a robust and variant sensitive mapper, BWA-MEM [80]. BWA-MEM can automatically choose between local and end-to-end alignments. It is applicable to map short as well as long reads and is sensitive in mapping reads with indels. Unlike other short-read mappers, it allows big gaps potentially caused by structural variants and shows better or comparable performance than several state-of-the-art read mappers to date in terms of speed and accuracy [80]. This mapper is robust to sequencing errors as well. After the reads are aligned by BWA-MEM, some reads still remain unmapped. At this step, another local alignment tool, BLAT (BLAST-Like Alignment Tool) [81] is used to align these reads using default settings. By merging the alignments from BWA-MEM and BLAT, 65.38% of the originally unmapped reads (624,892,089 out of 955,822,913) now are mapped to the reference genome. Out of these newly mapped reads, BWA-MEM mapped 479,064,451 reads and BLAT aligned 145,827,638 reads. As mentioned before, the mapper used by TCGA is BWA, which can map arbitrarily long reads theoretically, however, it has been observed in practice that, the performance in mapping long reads degraded with the increase of the sequencing error rate [84]. By removing bases with sequencing error and coupling BWA-MEM with BLAT for re-alignment, Genesis-indel manages to map many of the initially unmapped reads. Figure 4.4 shows the average mapping quality of the newly mapped reads for all samples. For most of the samples, the mapping quality is higher than that of the originally mapped reads (Appendix C Figure C.5).

Figure 4.4: Average mapping quality of the newly mapped reads of all samples.

4.3.4 Identifying the novel high-quality indels from the newly mapped reads

Genesis-indel uses Platypus [38] to call indels from the newly mapped reads. Separately, indels are also called from the reads that are originally mapped. Platypus is chosen as it performed the best among other existing indel callers based on real data as reported in a recent review [56]. After variant calling, indels from the newly mapped reads are inspected for any match with the indels found in the originally mapped reads. An indel already in the originally mapped reads can be called again in the newly mapped reads. These re-identified indels are discarded to avoid indel redundancy [85], and the remaining are considered novel indels.

Examination of the flags of the novel indels shows that for many of the indels, Platypus does not produce a high confidence value. Therefore, to consider only the high-quality indels for further analysis, novel indels are filtered again and only those with "PASS" flags are considered for the final result and are termed as "Novel High-Quality indel" (NHQ indel) in this chapter. In total, Genesis-indel reports 31,924 NHQ insertions (43.73% of the total NHQ indels) and 41,073 NHQ deletions (56.27% of the total NHQ indels) from the 30 samples investigated here. The deletion to insertion ratio for the NHQ indels is 1.29:1, similar to the deletion to insertion ratio 1.11: 1 for the originally mapped reads (7,313,641 insertions and 8,082,055 deletions).

Figure 4.5 shows IGV [86] snapshot of a novel 15-base deletion in Chromosome 1 that is identified in the newly mapped reads (lower panel) but missed in the original alignment (upper panel). Although this chapter focuses on indels only, as shown in Figure 4.5, new SNPs can also be identified by Genesis-indel in the originally unmapped reads.

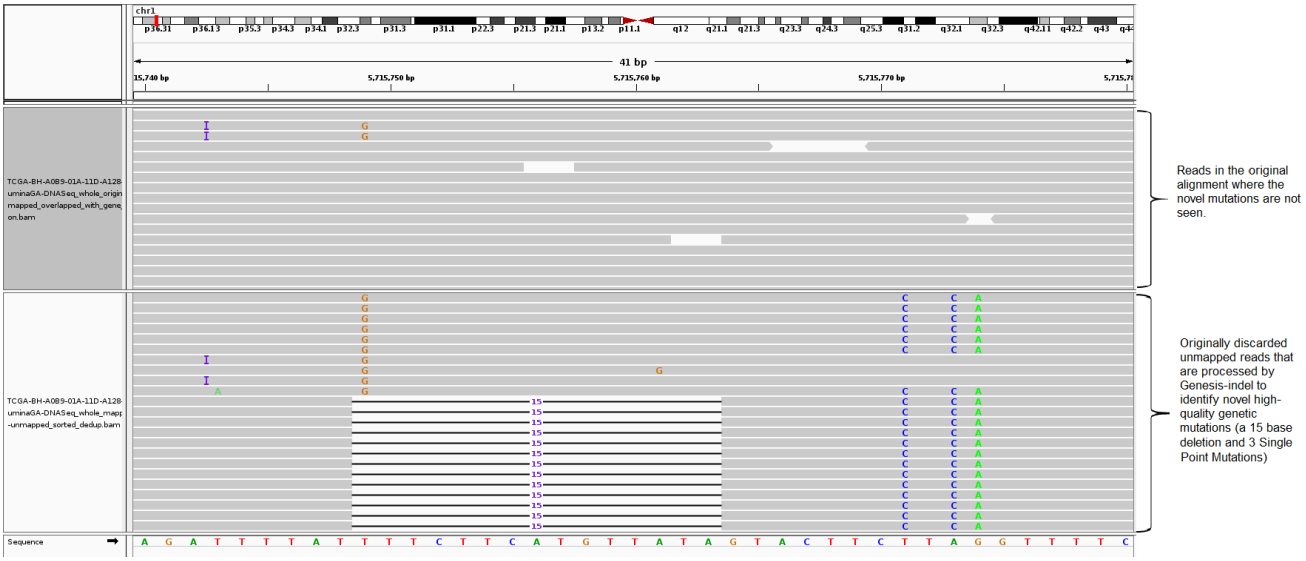


Figure 4.5: A NHQ indel identified in the newly mapped read but missed in the original alignment. The upper panel shows the originally mapped reads and the lower shows the newly mapped reads.

4.3.5 Frameshift indels are more frequent than in-frame types in the NHQ indels

NHQ indels identified here contain 53,623 frameshift and 19,374 in-frame indels, indicating a higher abundance of frameshift indels in the unmapped reads. Frameshift indels are also more abundant than in-frame indels in the originally mapped reads (13,911,266 versus 1,481,550). Particularly, frameshift indels of longer length (≥15 bases) are more frequent than in-frame indels of corresponding length (585 insertions, 1,479 deletions versus 404 insertions, 812 deletions). According to a study by Iengar et al. [87], 75.7% of the COSMIC indels are frameshift indels while only 24.3% are in-frame indels, suggesting that, unlike the distribution of coding indels in the genome of healthy people, frameshift indels dominate in cancer genomes. Because frameshift mutations are common in cancer patients and may increase the susceptibility to cancers and other diseases by causing loss of significant fractions of proteins [87-90], the NHQ frameshift indels newly uncovered by Genesis-indel may harbor important signals for linking indels to cancer or diseases and provide researchers new insights into the underlying mechanisms.

4.3.6 NHQ indels have significantly different length distribution than indels in the originally mapped reads

Figure 4.6 shows the distribution of the length of the NHQ indels analyzed here. It is observed that both insertion and deletion frequencies decrease with the increase of indel size. The longest NHQ insertions and deletions are 28 and 45 bases, respectively (34 and 55 bases for the indels from the originally mapped reads). It is expected that the novel indels would be long as they might have been missed because the lengths might exceed the number of gaps and mismatches allowed by the mapper. Surprisingly, as shown in Figure 4.6, most of the newly discovered indels (91% of insertion and 88.1% of deletion) are short (≤10 bases), indicating the limitation of the mapper used in the TCGA project. This figure also shows that NHQ indels have higher relative frequency than indels from the originally mapped reads for both insertion (3 to 28 base) and deletion (3 to 45 base). Nonetheless, a Pearson's chi-square test (i.e., testing for homogeneity in contingency table) shows that the length distribution of the NHQ indels is significantly different than that of the indels identified from the originally mapped reads (p-value < 2.2e-16).

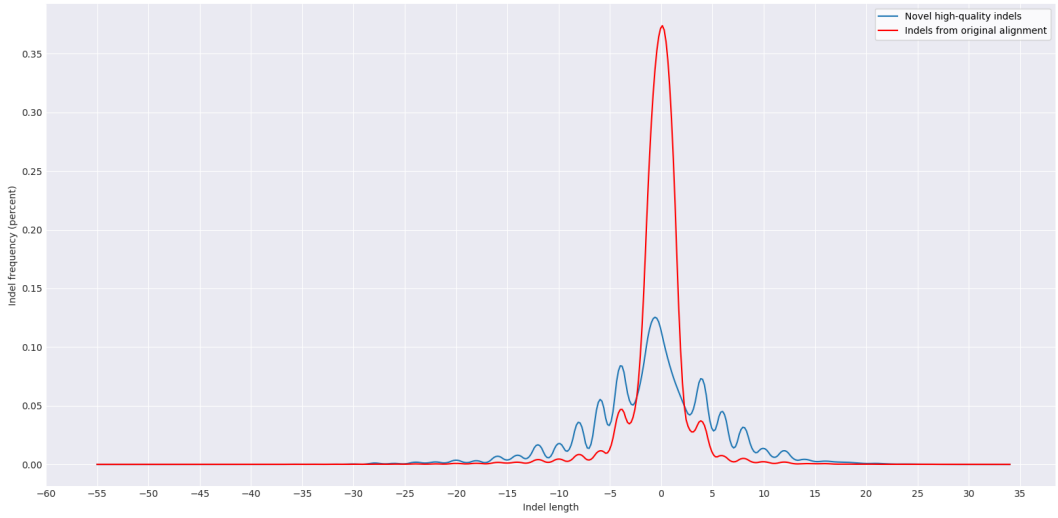


Figure 4.6: Length Distribution of the NHQ indels and indels from the originally mapped reads for all samples. Here a negative value indicates the deletion length.

4.3.7 Newly mapped reads can add more support to indels not recognized in the originally mapped reads

Most variant calling programs rely on hard evidence for indels marked in the alignment and therefore require a minimum number of reads to support an indel. This criterion is required to

distinguish real variants from the artefacts of sequencing errors. As shown in Figure 4.7 (upper panel), a 9-base deletion cannot be called from the original alignment due to lack of read support. After mapping the quality controlled originally unmapped reads through the Genesis-indel pipeline, such indels obtain sufficient read support and hence are called by the variant caller (lower panel). This provides an example scenario for how these indels are missed initially but are rescued by leveraging the unmapped reads.

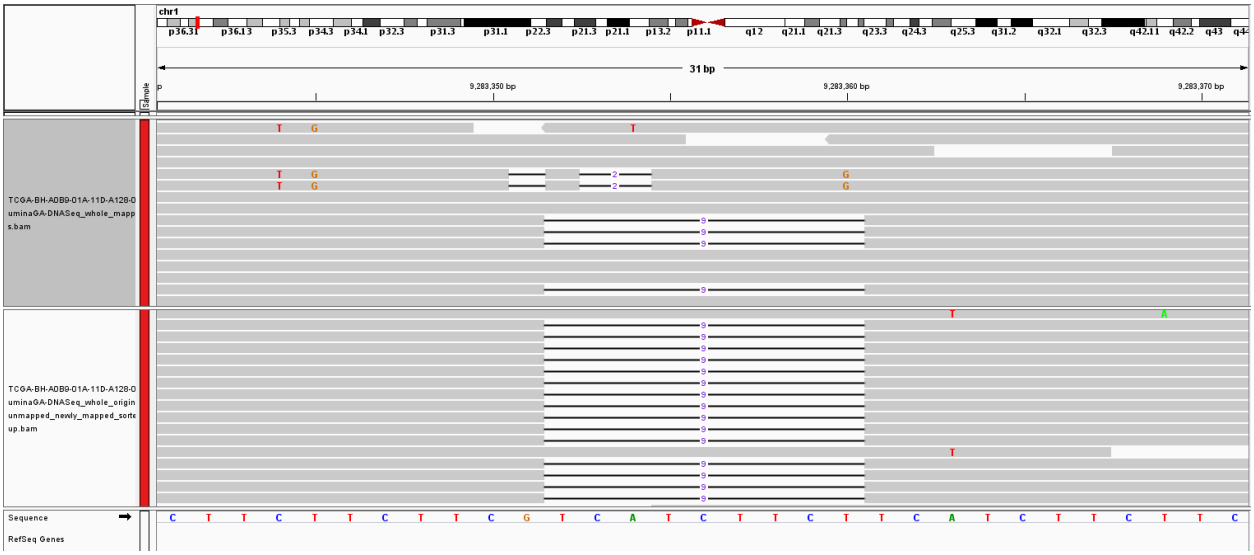


Figure 4.7: An example of a 9-base deletion which is not initially called from the original alignment due to lack of read-support but later called after the mapping of originally unmapped reads. Here the upper panel shows the original alignment and lower panel shows the original alignment and lower panel shows the alignment of the newly mapped reads.

4.3.8 Clustering of the samples based on quality-controlled unmapped reads

The samples are compared pairwise using the quality controlled unmapped reads to identify biologically relevant signals and to cluster the samples based on the number of similar reads. Pairwise distance is calculated for the unmapped reads from each sample using Mash, a distance estimator based on MinHash [91]. This pairwise distance is then used to cluster the samples. Figure 4.8 shows the hierarchical clustering of the samples based on Mash distance. The clustering results are then compared with the samples' PAM50 subtypes collected from TCGA [92]. Out of the total 30 samples, 16 belong to the Basal subtype and the remaining 14 belong to the LumA. As shown in Figure 4.8, all but three samples (samples 12, 13, and 14) cluster with the samples of their respective subtype. These three samples belong to Basal subtype but clustered with the samples from LumA subtype. The result reveals that the unmapped reads are most commonly shared among

the samples of the same subtype and suggests that these unmapped reads might contribute to the divergence between the two subtypes investigated here. This result also implies that perhaps there is a subtype-specific common cause of mapping failure. These results show that the sets of unmapped reads contain sequence information specific to sample subtype and hence leveraging such information may help understand or interpret the related biological questions.

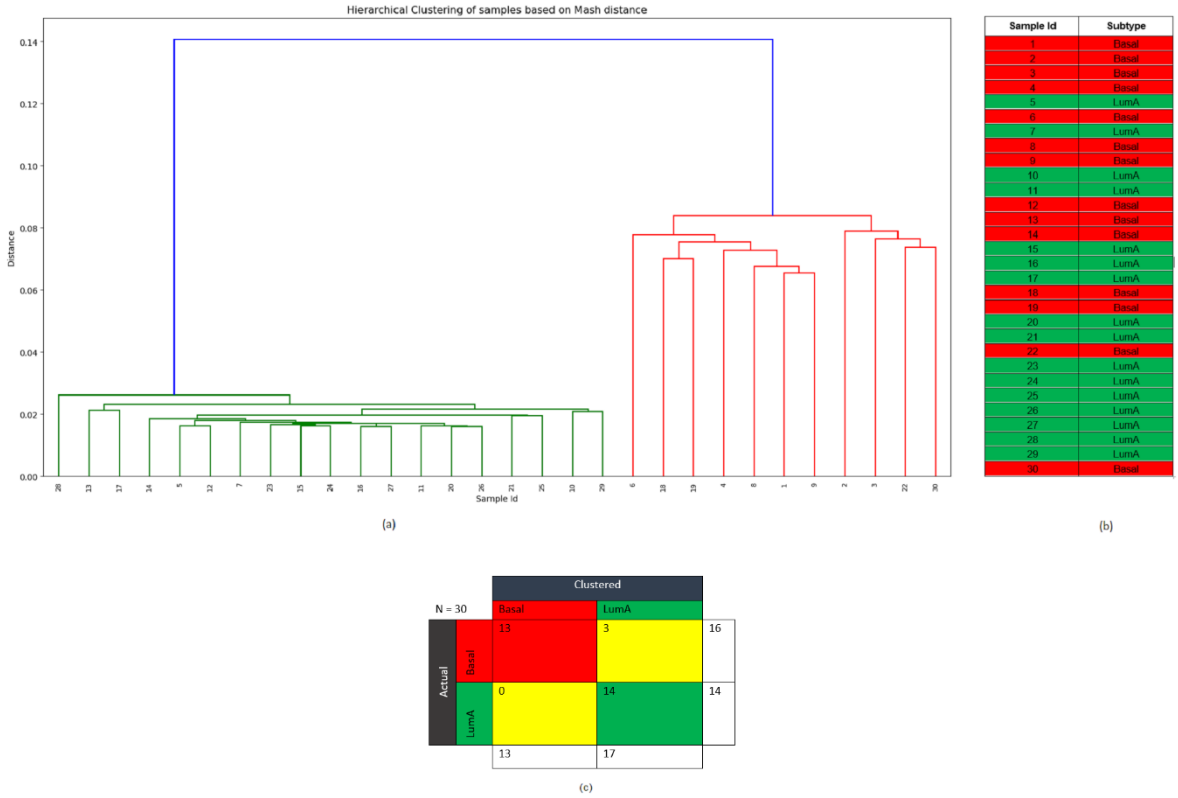


Figure 4.8: (a) Hierarchical clustering of the samples based on the pairwise Mash distance of the unmapped reads from each sample. (b) PAM50 subtype of the samples from TCGA. Here, the red color corresponds to Basal and green color corresponds to LumA subtype. (c) The confusion matrix.

4.3.9 Subtype-specific indels from the NHQ indels

All except three samples (samples 12, 13, and 14) of the Basal subtype contain 5,818 indels on average and LumA samples contain 473 indels on average. Samples 12, 13, and 14 contain 484, 415, and 535 indels, respectively, i.e., similar to the number of indels found in the LumA samples. Similar phenomena are also observed for these three samples in the indels from the originally mapped reads, giving more evidence that these samples actually belong to the LumA subtype but were mislabeled as Basal, consistent with the result of clustering based on unmapped reads. In addition, the number of newly mapped reads in Samples 12, 13, and 14 are 3.6, 2.5, and 3.5 million,

respectively, which is closer to the number of newly mapped reads in the LumA samples (average number of newly mapped reads = 3.3 million) than to the Basal samples (average number of newly mapped reads = 37.89 million). This suggests possible subtype mislabeling of these samples.

NHQ indels are checked to see if they are specific to any of the two subtypes (Basal and LumA) investigated here. An indel is defined as specific to a subtype when it is found in the samples of one subtype and not in the samples of the other subtype. Among the 72,997 NHQ indels, 89 are found to be Basal specific indels and none is found to be LumA specific.

4.3.10 NHQ indels overlapped with the oncogenes and tumor suppressor genes

To see which oncogenes and tumor suppressor genes are frequently affected by the newly discovered indels, a list consisting of 142 protein-coding genes (79 oncogenes and 63 tumor suppressor genes, see Methods) is overlapped with the NHQ indels using BEDtools [93]. In total, 62 out of these 142 genes overlapped with these indels. Among these 62 genes, 32 are oncogenes and the remaining ones are tumor suppressor genes. Table 4.1 lists the top ten genes with the highest number of indels identified in them. RUNX1 (Runt Related Transcription Factor 1), a protein-coding tumor suppressor gene, has the highest number of indels (54) and thus likely contains important signatures of breast cancer. RUNX1 has received attention as a gene fusion in acute myeloid leukemia (AML) [94, 95]. Although a putative link to breast cancer has recently emerged [96], RUNX1 has not gained sufficient attention, and its role in breast cancer still remains elusive [97]. One reason for the understudy of the RUNX1 gene is the underpowered expression profile studies as identified by Janes et al. [98]. Another reason, as the result shows here, could be because of not discovering the indels hidden in the unmapped reads. This study provides new evidence to re-examine the role of RUNX1 in breast cancer, as a complement to the study performed by Janes et al. [98].

Table 4.1: Top ten oncogene and tumor suppressor genes and the number of indels identified in these genes.

Gene	Number of Indel
RUNX1	54
SYK	13
CBLB	11
ETV4	11
CCND3	10
ETV6	9
MAML2	7
PIK3CA	7
BMPR1A	6
EGFR	5

Frameshift indels are more abundant than in-frame indels in both oncogenes and tumor suppressor genes. Some genes contain only in-frame indel and some contain only frameshift indels. As shown in Table 4.2, out of the 62 genes overlapped with the NHQ indels, 46 contain either in-frame or frameshift indels. The remaining 16 genes contain both in-frame and frameshift indels. As shown in the previous section, frameshift indels are the dominant type of indels and RUNX1 contains the maximum number of indels for both in-frame (12) and frameshift (42) among all genes investigated here, making it an important candidate for a breast cancer marker.

Table 4.2: List of oncogenes and tumor suppressor genes containing either in-frame or frameshift NHQ indels.

Indel Type	Gene
In-frame	Oncogenes (5): HMGA2, TPR, RAF1, ROS1, FGFR2
	Tumor suppressor genes (3): EXT1, CREB1, GPC3
Frameshift	Oncogenes (19): ATF1, CBLB, AKT2, LMO2, TET2, ETV4, BCR, MAF, SMO, PPARG, CARD11, DDX6, PLAG1, EGFR, ABL1, NTRK1, BCL11A, BCL2, FGFR1
	Tumor suppressor genes (19): RB1, SMARCB1, FLT3, BRCA1, CDH1, SUFU, CHEK2, ARHGEF12, FBXW7, MSH2, NUP98, SUZ12, NPM1, BCL11B, IDH1, EXT2, NR4A3, ATM, BMPR1A

4.3.11 NHQ indels mostly alter cancer-related genes than noncancer-related genes

A list of whole genome protein-coding genes not containing the oncogene and tumor suppressor gene is generated from the whole genome gene list produced by GENCODE (version 28 lift37). This list contains 20,172 genes, and, among these, 6,829 genes are found overlapped with the NHQ indels. A hypothesis testing is done to compare the proportion of NHQ indels appearing in cancer-related genes (oncogene and tumor suppressor gene) to that in noncancer genes. Statistically, the hypothesis being tested is as follows,

$$H_0: p_1 \leq p_2$$

$$H_1: p_1 > p_2$$

where p_1 denotes the proportion of oncogenes and tumor suppressor genes that overlapped with the NHQ indels and p_2 denotes the proportion of noncancer genes that overlapped with the NHQ indels.

Given the data observed (62 out of the total 142 oncogenes and tumor suppressor genes are overlapped with the NHQ indels, and 6,829 out of the total 20,172 noncancer genes are overlapped with the NHQ indels), a z-test for testing the difference in two proportions reports an observed z value of 2.35, yielding a p-value of 0.0094. Equivalently, a chi-square test of homogeneity based on the 2×2 contingency table gives a p-value of 0.0177 with Yates' continuity correction. Both results show that the null hypothesis, H_0: $p_1 \leq p_2$ can be rejected at nominal level $\alpha = 0.05$. Therefore, the proportion of cancer genes overlapping with NHQ indels is significantly higher than that for noncancer genes.

An alternative approach by permutation test is also done to see if the NHQ indels have a higher enrichment in cancer genes (oncogenes and tumor suppressor genes) than in noncancer genes. For this test, from the list containing 20,172 genes (the whole genome gene list not containing the oncogene and tumor suppressor genes), 142 genes (number of total oncogene and tumor suppressor genes) are sampled randomly and checked for the number of genes that overlap with the NHQ indels. Repeating the sampling 1,000 times yields a distribution for the number of genes overlapping with the NHQ indels. Out of the 1,000 sets, the number of overlaps higher than 62 (the observed number of overlaps) only occurs 9 times, and, therefore, the permutation test p-value

is 9/1000 = 0.009 (which is also consistent with the p-value from the z-test above). Again, the null hypothesis can be rejected, and therefore, it can be concluded that there is a significant enrichment of the NHQ indels in cancer genes, further suggesting that the NHQ indels may harbor important genetic mechanisms for breast cancer.

4.3.12 Annotating the NHQ indels using Variant Effect Predictor (VEP)

For this analysis, the NHQ indels are annotated using Variant Effect Predictor (VEP) [99]. 16,141 of the indels are identified as novel, i.e., not annotated in the Ensembl variation database consisting of dbSNP, Cancer Gene Census, ClinVar, COSMIC, dbGap, and DGVa. This indicates the significance of this study in rescuing these indels from the discarded reads that can potentially be annotated.

The NHQ indels overlapped with 15,229 genes, 32,335 transcripts, and 2,136 regulatory features. As shown in Figure 4.9(a), approximately 75% of the NHQ indels are in the non coding regions located in the intron or intergenic region. Figure 4.9(b) shows that 72% of the indels in the coding regions are frameshift indels that cause a disruption of the translational reading frame and can have a disruptive impact in the protein by causing protein truncation and/or loss of function. In addition, a small amount of the indels are "Splice donor variants" changing the 2-base region at the 5' end of an intron and can have a similar impact as frameshift indels. 25% of the indels are in-frame indels having a "moderate" impact in the protein by not disrupting the protein but changing the effectiveness of that protein. 70% of the NHQ indels having disruptive and moderate impact overlap with the protein-coding transcripts, the leading biotype of all features (Transcripts, Regulatory Features, and Motif Features) as shown in Figure 4.9(c). Out of the remaining indels, 37.44% (12.48% of the total NHQ indels) have a "modifier" impact that overlap with long intergenic RNA transcripts (lincRNA). LincRNAs are noncoding transcripts with a length longer than 200 nucleotides and are the largest class of noncoding RNA molecules in the human genome. There is emerging evidence that noncoding RNAs regulate gene expression by influencing chromatin modification, mRNA splicing, and protein translation [100, 101] as well as contribute to mammary tumor development [102, 103] and progression. Therefore, the NHQ indels overlapped with these transcripts deserve more attention and studying these indels has biological significance.

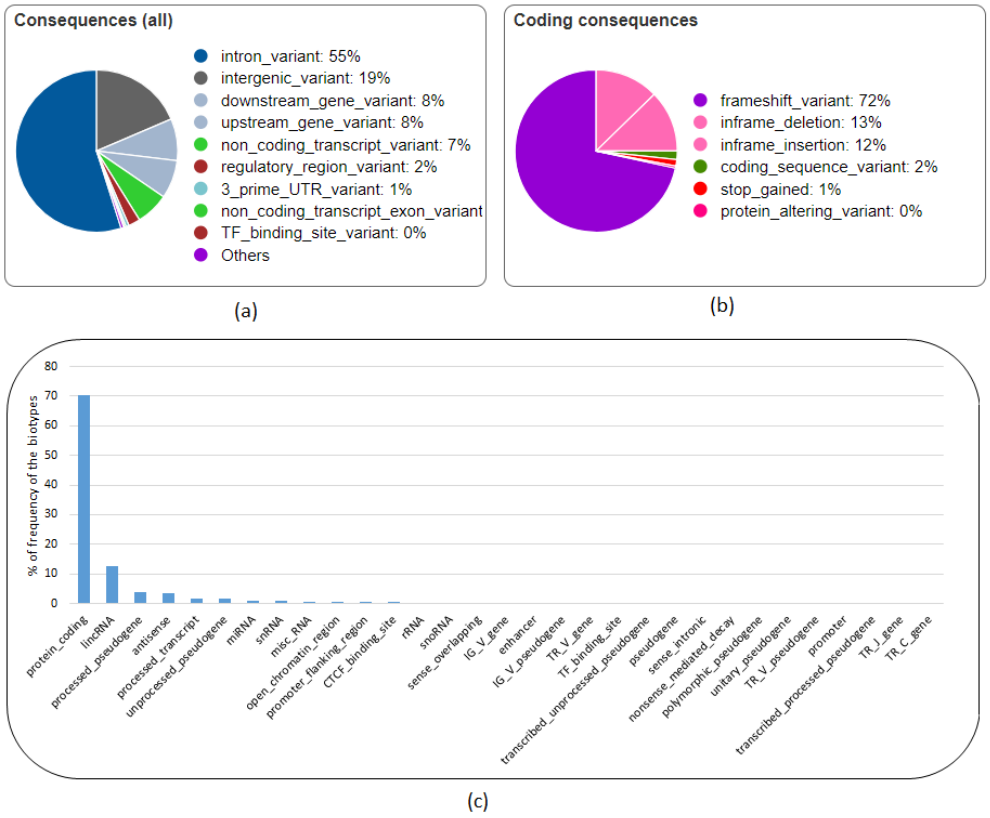


Figure 4.9: Analysis of the NHQ indels using Variant Effect Predictor (VEP). (a) All consequences, (b) coding consequences, (c) distribution of biotype of the features overlapped with the NHQ indels.

4.3.13 Functional annotation of the genes overlapping with the NHQ indels

Functional annotation of the genes overlapping with the NHQ indels using David [104] (version 6.7) shows strong correlation with "Pathways in Cancer" (Fisher Exact p-value: 6.2×10^{-5}), "PI3K-Akt signalling pathway" (Fisher Exact p-value: 1.5×10^{-4}), RAP1 signalling pathway (Fisher Exact p-value: 10^{-4}), and RAS signalling pathway (Fisher Exact p-value: 4.7×10^{-3}). A previous study shows that components of the PI3K-Akt signalling pathway are recurrently altered in cancers and the survival signals induced by several receptors are mediated mainly by this pathway [105]. Ras-associated protein-1 (RAP1) is an important regulator of cell functions and has been found playing a vital role in cell invasion and metastasis in cancers [106]. The signaling pathways involving RAS protein can contribute to tumor growth, survival, and spread and play a crucial role in the pathogenesis of other hematologic malignancies as well [107, 108]. Therefore, these results suggest that the newly found indels interacting with these genes may participate in cancer-related biological processes and play an important role in cancer progression.

4.3.14 Genes missed in the original mapping but found in NHQ indels show association with cancer and other diseases

There are 42 genes overlapping with the NHQ indels but not with the indels from the originally mapped reads. Table 4.3 lists the genes with their types. Functional annotation of the protein-coding genes shows that these genes are related to biological process such as immune response, protein localization, protein transport, regulation of transcription, and regulation of RNA metabolic process which can control molecular functions such as antigen binding, peptide binding, MHC protein binding, and peptide-antigen binding. In addition, these genes are associated with protein domain such as Immunoglobulin subtype and Krueppel-Associated Box (KRAB)–Zinc Finger Protein (ZFP). Immunoglobulin subtype which are involved in cell-cell recognition, cell-surface receptors, muscle structure, and the immune system [109] and therapy targeting this protein domain has been used for liver cancer [110], breast cancer [111], and Follicular Lymphoma [112, 113]. Krueppel-Associated Box (KRAB)–Zinc Finger Protein (ZFP) is the largest class of transcription factors in the human genome [114] and is largely involved in tumorigenesis [115].

Table 4.3: Name and type of the genes that overlap with the NHQ indels but not with the indels from the originally mapped reads.

Gene Type	**Gene Name**
Antisense	RP11-534L20.5, JMJD1C-AS1, CTB-22K21.2, RP11-1079K10.4, RP11-16N11.2, RP11-26P13.2, RP11-1299A16.3, RP1-16A9.1.
LincRNA	RP5-1065P14.2, RP11-309G3.3, RP11-382D12.1, RP11-14C22.3, RP11-386I8.6, LINC00379, RP3-503A6.2, RP3-416J7.4, RP11-100L22.4.
miRNA	MIR4477A.
Pseudogene	RP5-857K21.7, RP11-428G5.7, BNIP3P1, RP11-713H12.2, VN1R90P, MLLT10P1, UNC93B3, TBCAP3, CTD-2158P22.1, RP11-823P9.3, RP3-416J7.1, CYP4F44P, OR5BH1P, PABPC1P3, CASKP1, TRBV12-1, TRBV12-2.
Protein coding	OR10G8, ZNF26, ZNF84, KIF20A.
snRNA	RNU1-59P, RNU6-377P, RNU1-36P.

A PubMed search returned results for three genes, namely, KIF20A, BNIP3P1, and ZNF84.

Kinesin family member 20A (KIF20A), also known as RAB6KIFL, is a member of the kinesin superfamily of motor proteins, a conserved motor domain that binds to microtubules to generate

the energy required for trafficking of proteins and organelles during the growth of numerous cancers [116, 117]. KIF20A is found overexpressed at both the mRNA and protein levels than the normal counterparts in breast cancer [118-120] and also in several other cancers including gastric cancer [121], bladder cancer [122, 123], pancreatic cancer [124-126], hepatocellular cancer [127], lung cancer [128], glioma [129], and melanoma [130]. The overexpression of KIF20A is significantly associated with poor survival of a breast cancer patient [118, 119] and with drug resistance [119, 131]. Similar phenomena are observed with other cancer patients as well [121, 123, 124, 126, 128, 132]. Silencing or knockdown of KIF20A can significantly inhibit cell proliferation and cancer progression [125, 133]. Therefore, KIF20A has been suggested as a direct therapeutic target [125, 134], and a KIF20A-derived peptide has been used in immunotherapy in clinical trials to improve the prognosis of cancer patients [116, 129, 135-138]. Although KIF20A has a strong association with breast cancer, no mutation is found in this gene from the originally mapped reads which shows the limitation of the current approach. This limitation, however, can be alleviated by exploring the unmapped reads. Besides cancer, KIF20A is found associated with heart disease in infants. A recent study by Louw et al. [139] identified an undescribed type of lethal congenital restrictive cardiomyopathy, a disease affecting the right ventricle of two siblings. Exome sequencing analysis of these affected siblings and their unaffected sibling revealed two compound heterozygous variants in KIF20A: a maternal missense variant (c.544C>T: p. R182W) changing an arginine to a tryptophan and a paternal frameshift deletion (c.1905delT: p. S635Tfs15, in exon 15) that introduces a premature stop codon 15 amino acids downstream. Louw et al. [139] validated the variants by Sanger sequencing, found the presence of both variants in the affected siblings, and confirmed a heterozygous carrier status in both parents. In addition, both variants were absent in the unaffected sibling. The C>T missense SNP does not let KIF20A support efficient transport of Aurora B as part of the chromosomal passenger complex causing Aurora B trapped on chromatin during the cell division, and, hence it fails to translocate to the spindle midzone during cytokinesis. This claim is verified by Louw et al. [139] in the zebrafish model, where translational blocking of KIF20A resulted in a cardiomyopathy phenotype. A similar congenital restrictive cardiomyopathy is also identified to be caused by the deletion resulting in loss-of-function of KIF20A [139]. Despite such significance, these two variants that affect protein function were absent in the population control exome such as the ExAC Browser database, a catalogue of genetic data of 60,706 humans of various ethnicities [140]. The missense variant was

found in two individuals from South Asia and Europe, and the frameshift deletion was present in 32 individuals of African descent [139, 141]. This observation supports the claim that clinically important mutations can be missed, and one of the reasons might be because of overlooking the unmapped reads. By exploring the unmapped read of 30 breast cancer patients, Genesis-indel finds a frameshift deletion of T (chr5:137520225 CT -> C) that overlaps with the exon of KIF20A gene (Figure 4.10). Note that, in Figure 4.10, only two out of four reads support the deletion, whereas, in Figure 4.7, a 9-base deletion is not called from the originally mapped reads although 4 reads support that deletion. The reason is the single base deletion is supported by 50% of the reads aligned in that region, whereas the 9-base deletion in the originally mapped reads (Figure 4.7 upper panel) is supported by 25% of the aligned reads, which is possibly lower than the default threshold set by the variant caller.

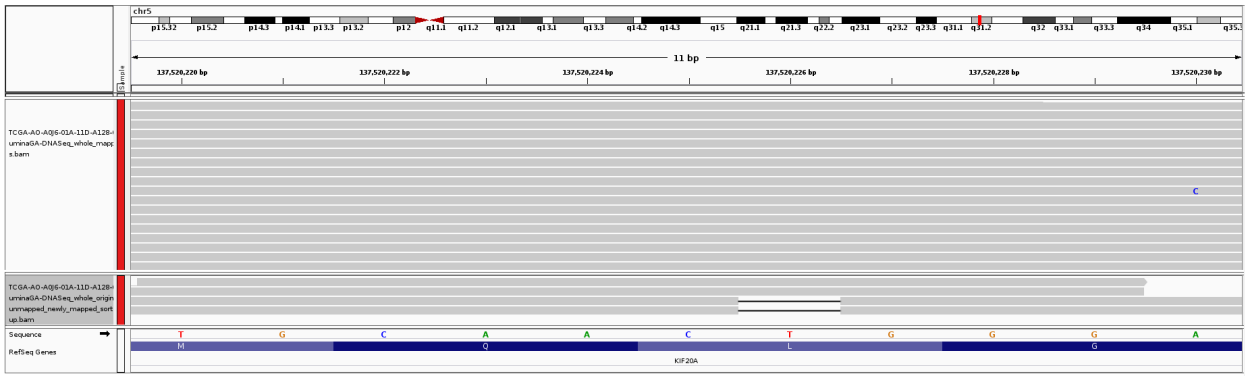


Figure 4.10: A 1-base deletion in the exon of KIF20A gene which is not initially called from the original alignment but called after the mapping of originally unmapped reads. The upper panel shows the original alignment and lower panel shows the alignment of the newly mapped reads.

Among the remaining genes, BCL2 interacting protein 3 pseudogene 1 (BNIP3P1) is found to be upregulated in patients with breast cancer Brain Metastases when compared to breast cancer (76% vs. 24%) or compared to Primary Brain Tumors (74% vs. 26%) [142] and is suggested to be used as a molecular biomarker for breast cancer Brain Metastases. Zinc Finger Protein 84 (ZNF84) is found significantly associated with tumor size and TNM (Tumor, Node, Metastases) staging for cervical cancer and squamous cell carcinoma and in vitro validation shows that it promotes cell proliferation via AKT signaling pathway [143]. Although the literature does not show any association between these genes and breast cancer, it is worth exploring due to their association with other cancers.

Out of the 42 genes, two LincRNAs namely RP3-416J7.4 and RP11-386I8.6 contain the same number of indels as the protein-coding genes. Although little is known about their association with breast cancer, analysis using TANRIC [144] on TCGA-BRCA data reveals that these two LincRNAs are differentially expressed (t-test p-value = 0.000023337 and 0.003812, respectively) between the carriers and non-carriers of somatic mutations in the TP53 gene, a tumor suppressor gene spontaneously found altered in breast carcinomas [145].

While this chapter shows the significance of uncovering NHQ indels from the originally unmapped reads in patients with breast cancer, there are two limitations. Firstly, this study is conducted by using a computational pipeline. Though the pipeline is computationally viable and results are convincing as well as supported by experimentally validated literature, it lacks some validation experiments in vivo to govern the clinical importance of the newly identified indels. Secondly, filtering indels solely based on the "PASS" flag may cause missing rare variants. Therefore, an algorithm such as ForestQC [146] that a combines traditional variant filtering approach with a machine learning algorithm to determine the quality of the variant can be incorporated in the present pipeline to improve the quality control procedure and achieve better results.

4.4 Conclusion

This chapter emphasizes the interest of studying unmapped reads to cope with potential loss of important information and describes Genesis-indel, a computational pipeline to rescue novel high-quality indels by exploring unmapped reads that are normally discarded from the downstream analysis.

Analyzing the whole genome DNA alignment of 30 breast cancer patients from TCGA reveals a nonnegligible number of unmapped reads that are overlooked earlier. After mapping the unmapped reads to the reference genome, Genesis-indel finds 72,997 novel high-quality indels of diverse lengths, and 16,141 have not been annotated in any of the genetic variation databases used by Ensembl. These novel high-quality indels are mainly enriched in frameshift indels and have high to moderate impact in the protein. These indels mostly alter the oncogenes and tumor suppressor genes and overlap with genes significantly related to different cancer pathways. Moreover, these indels overlap with genes not found in the indels from the originally mapped reads and functional annotation shows that these genes contribute to the development and growth of tumor in multiple carcinomas. Therefore, these findings collectively suggest that complete characterization of these

indels is essential for downstream cancer research. Genesis-indel is expected to be highly useful for uncovering the missed indels that can be further explored for clinical decision making.

Chapter 5

Identifying somatic mutations using SomaticHunter

Somatic mutations play a vital role in transforming healthy cells into cancer cells. Therefore, accurate identification of somatic mutations is essential for many reasons such as a better understanding of cancer genomes, facilitating cancer diagnosis, and clinical decision making in cancer treatment. Many somatic mutation calling tools are available, and more are likely to appear. Most of these tools achieve high recall. However, this high recall comes at the cost of low precision. Some variant callers show the opposite. Both of these cases result in low F1 score, a widely used performance metric. Additionally, somatic mutation calling is challenging, and it is unlikely to have a single caller that systematically outperforms all others. Therefore, combining the strength of multiple callers followed by an intelligent mechanism of classifying somatic and non-somatic mutation is a powerful way to generate a comprehensive list of somatic mutations while achieving both high recall and precision. This chapter describes SomaticHunter, an ensemble of two sensitive variant callers, namely, Platypus and VarDict. Mutations called by these two callers are filtered by a decision tree classifier built using adaptive boosting with features such as read depth, mapping quality, and base quality. After applying SomaticHunter to synthetic tumor data for Whole Genome Sequence (WGS) and Whole Exome Sequence (WXS), results show that for both SNPs and short indels, SomaticHunter achieves recall comparable (82% versus 20 to 93% for WGS SNP, 68% versus 45 to 81% for WXS SNP, 77% versus 62 to 90% for WGS indel, 71% versus 30 to 77% for WXS indel) to the state-of-the art somatic mutation callers including Mutect2, VarScan2, SomaticSniper, and Strelka2 while achieving the highest precision (76% versus 13 to 75% for WGS SNP, 99.74% versus 19 to 99.61% for WXS SNP, 76% versus 18% to 66% for WGS indel, 76% versus 38% to 72% for WXS indel) and therefore resulting in the highest F1 score (79% versus 23 to 35% for WGS SNP, 82% versus 31 to 62% for WXS SNP, 76% versus 29 to 63% for WGS indel, 74% versus 43 to 49% for WXS indel).

5.1 Introduction

Somatic mutations are genetic variations taking place in the somatic cells. Somatic mutations drive cancer by promoting normal cells to transform into cancer cells [31, 147-149]. There is strong evidence that somatic mutations such as SNPs and indels are common drivers in cancer progression and cancer genomes are known to harbor a wide range of these mutations [150]. For

example, oncogenes such as KRAS, NRAS, BRAF, and EGFR often contain hotspot missense mutations [31, 151] and tumor suppressors genes such as TP53, PTEN, BRCA1, BRCA2, RB1, STK11 and NF1 often contain large frameshift indels [66]. Therefore, identifying and analyzing somatic mutations is essential for better understanding of the cancer genomes [150, 152].

With the advancement of the next generation sequencing (NGS) technology, many tools have been developed to detect somatic SNPs and indels. Comparative studies on somatic variant calling tools using different data sets such as amplicon and whole exome data [153], exome and targeted deep sequencing data [154], whole genome sequencing data prepared and called by different sequencing centers [155], real data from patients with myelodysplastic syndrome, and two simulated data sets with 50 samples with varying coverage and error profiles [156] revealed that there is little agreement on the called results by these tools. The disagreement is more pronounced for indels [157], a phenomenon similarly observed in germline indel callers [56]. Every caller has its strengths and limitations. Moreover, some of the state-of-the-art variant callers achieve high sensitivity, but low specificity, while others show the opposite. Consequently, finding the single best somatic variant caller is challenging [152, 158, 159].

An ensemble of variant callers that integrates the strength of the callers can be a powerful way to address the problem mentioned above. Combining the variant callers can produce a comprehensive list of variants. However, combining too many variant callers increases the computation time. Combining many variant callers also brings the risk of combining the false positive calls, which can be overwhelming. A machine learning algorithm that can accurately classify variants as somatic or not somatic can be a solution to this problem. This chapter describes SomaticHunter, an ensemble of two variant callers, Platypus [38] and VarDict [150], to call somatic variant from paired tumor-normal sequence alignment.

Due to better sensitivity than alignment based variant calling methods in challenging regions, local-assembly based variant callers such as Platypus are now becoming a popular solution [160]. Tools solely depending on the alignment can miss variants, especially indels, if a mapper cannot map a read containing a long indel. Unlike the alignment-based caller, local-assembly based tools produce contigs longer than reads by doing localized assembly of the reads mapped around the location of the candidate variant. These longer contigs are then aligned to the reference genome to determine the variants. A recent review on germline indel callers [56] using real data of the 1000

Genomes Project [161-163] reveals the better performance of local-assembly based callers such as Platypus [38] and GATK HaplotypeCaller [37], as compared to other state-of-the-art alignment based callers. Also, assembly-based methods demonstrate substantial power to detect long indels compared to alignment-based methods [164]. Such advantages offered by Platypus make it a suitable candidate for being integrated into SomaticHunter. VarDict [150] uses a Fisher's exact test to determine whether a variant has a significant difference in allele frequency between the tumor and the normal sample. This test is an important feature to differentiate between a somatic and non-somatic variant. Moreover, by applying a series of filters, VarDict removes false positives and increases sensitivity. Also, the local realignment feature of VarDict allows it to call long indels and variants in challenging regions that are usually missed by other callers. These features of VarDict make it a suitable caller to be included in SomaticHunter. After combining the variants called by Platypus and VarDict, the combined call set is passed to a classifier built using adaptive boosting of decision trees which classifies a given variant as somatic or non-somatic.

SomaticHunter is trained using the synthetic truth data produced by the ICGC-TCGA DREAM Challenge [165] stage 3. This tool is validated with a synthetic normal-tumor pair for Whole Genome Sequence (WGS) data produced by Narzisi et al. [157] and Whole Exome Sequence (WXS) data produced by Meng et al. [166] and compared with Mutect2 (GATK v4.0.5.2) [19], VarScan2 (version 2.4.0) [31], Strelka2 (version 2.9.10) [167], and SomaticSniper (version 1.0.5.0) [168]. Results show that, for both WGS and WXS, SomaticHunter achieves the highest F1 score by achieving the highest precision and a comparable recall for both SNPs and indels. This performance achieved by SomaticHunter greatly facilitates the analysis of NGS data in cancer research by enabling researchers to use it to identify somatic variants accurately.

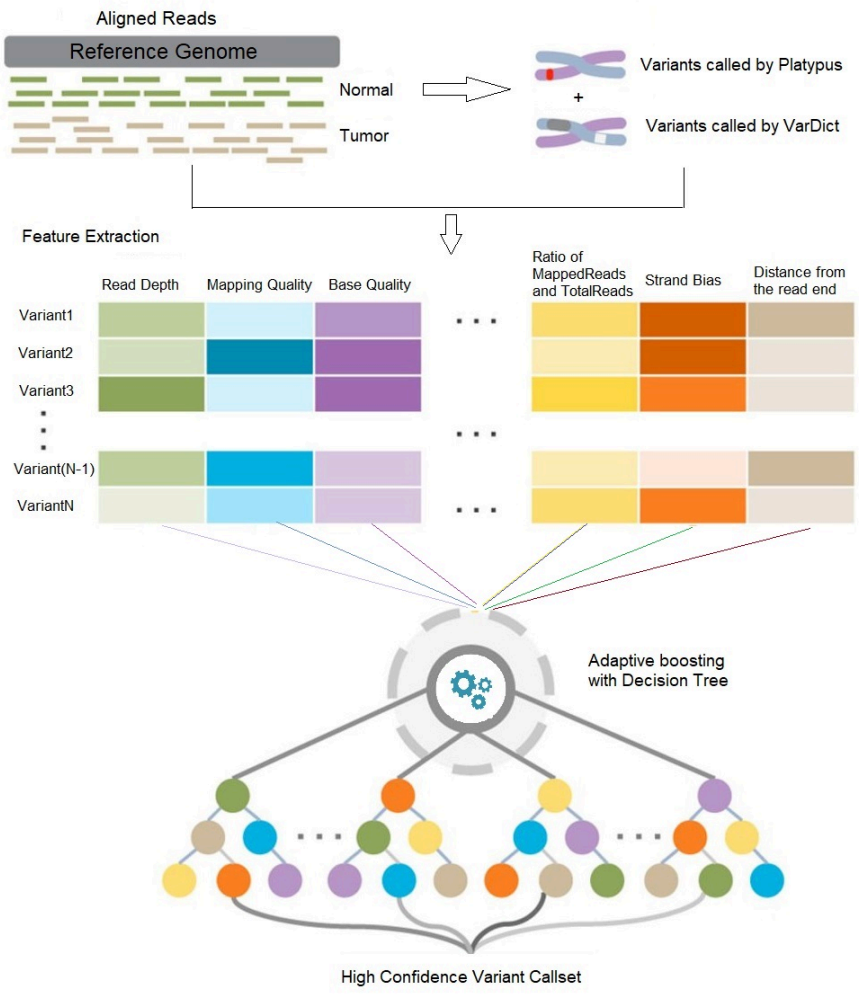


Figure 5.1: Overview of SomaticHunter. The input to SomaticHunter is sequence alignment file (BAM) from paired tumor and normal tissue and a reference genome in FASTA format. First, Platypus simultaneously calls SNPs and indels from the normal and tumor tissue. VariDict also calls variants in the paired-analysis mode. In the next step, variants that are identified by both Platypus and VarDict as somatic are reported to the final output without further checking. The remaining variants, i.e., called by one tool but not by the other one is passed to the machine learning module. The machine learning module contains an ensemble of decision tree classifiers built using adaptive boosting ensemble algorithm. It is trained using DREAM challenge stage3 data with features including read depth, mapping quality, base quality, and strand bias. Using this classifier, candidate variants are classified as somatic or nonsomatic. Finally, the SNPs and indels that are classified as somatic are reported in the final output in VCF format.

5.2 Materials and Methods

Figure 5.1 shows an overview of SomaticHunter. Like all existing somatic variant callers, SomaticHunter takes an alignment file (BAM file) for tumor tissue, one for the normal tissue, and a reference genome. In the variant calling step, Platypus simultaneously calls SNPs and indels from the normal tissue sequence alignment (germline) and also the tumor tissue alignment. If a variant is observed in both germline and tumor tissue, that variant is discarded from the

downstream analysis because it cannot be a somatic variant. This simple filter is useful to reduce false positive germline variants. VarDict calls somatic variants in the paired analysis mode to analyze the normal and tumor BAM files simultaneously to report only somatic variants. Variants called by Platypus and VarDict are combined. If a variant is identified as somatic by both callers, it is reported in the final output without further checking. The remaining variants, i.e., the ones called by one tool but not by the other one is the candidate variants and passed to the machine learning module to filter out false positives.

5.2.1 Training data set

The machine learning module of SomaticHunter uses an adaptive boosting ensemble algorithm containing ensembles of decision tree classifier. The classifier is trained based on the ICGC-TCGA DREAM Somatic Mutation Calling Challenge Stage 3 data. To generate this data, a deeply sequenced (60X coverage) sequence alignment file (BAM file) corresponding to the ATCC sample HCC1143 BL was obtained and randomly sampled into two non-overlapping subsets of equal size having 30X coverage each. After that, BAMSurgeon (https://github.com/adamewing/bamsurgeon) is used to add synthetic mutations to one of the sub-sampled BAM files. Some of these non-overlapping synthetic mutations are randomly selected, and the remaining ones are known variants in cancer-associated genes. The sub-BAM with added variations is treated as the tumor BAM, and the other one is treated as the normal. The variants added by BAMSurgeon are stored in a VCF file containing 7,903 SNPs and 8,292 indels. These sub-BAM files and the VCF files are used for training the machine learning model.

Features of the machine learning model include Read Depth, Mapping Quality, Base Quality, Strand bias, etc. and extracted from the input BAM files using SAMtools [28]. Appendix D Table D.1 lists all the features used for SomaticHunter.

5.2.2 Testing data set

SomaticHunter is tested and compared with Mutect2, VarScan2, Strelka2, and SomaticSniper based on two data sets, dataset1 is for Whole Genome Sequencing (WGS), and the other one is for Whole Exome Sequencing (WXS) referred to as dataset2 in this manuscript.

Virtual tumor for dataset1 is created by Narzisi et al. [157] by using alignment from HapMap sample NA12892. In the first step, it is partitioned into 80X and 40X coverage to be used as tumor

and normal, respectively. After that, HapMap sample NA12891 is used to introduce SNPs and indels in the tumor BAM by swapping reads between these two samples at loci where NA12892 is a homozygous reference, but NA12891 is homozygous variant. The list of loci where the mutations are introduced are based on the call sets from the 1000 Genomes Project phase 3. Separate tumor BAM files are created for SNPs containing 31,592 somatic SNPs and indels containing 4,945 somatic indels.

Dataset2 for WXS is created using BAMSurgeon by Meng et al. [166] by introducing SNPs and indels into homozygous reference sites of well-characterized HapMap/1000 Genomes CEU sample NA12878. Twenty-seven data sets are generated for WXS, containing 807 SNPs and 80 indels on average.

5.2.3 Comparison with the existing somatic variant callers

Using the test data set mentioned above, SomaticHunter is compared with the other state-of-the-art somatic indel callers including MuTect2 [19], VarScan2 [31], Strelka2 [167], and SomaticSniper[168]. Among these tools, SomaticSniper calls only SNPs. These tools are compared based on recall and precision as calculated using formula (5.1) and (5.2):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5.2}$$

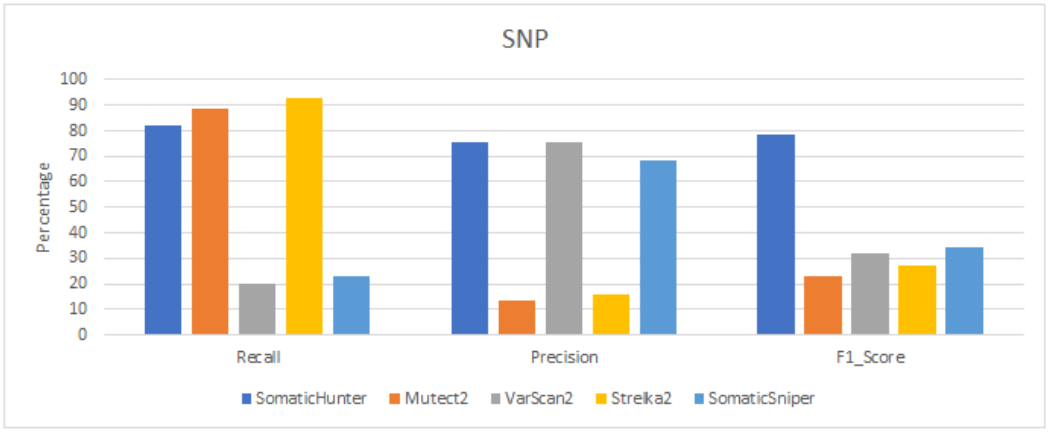
For comparing the accuracy of the tools, F1 score, the harmonic mean of the precision and recall, is used. A high F1 score indicates better performance. The F-measure is calculated using formula (5.3):

$$\text{F1\_Score} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{5.3}$$

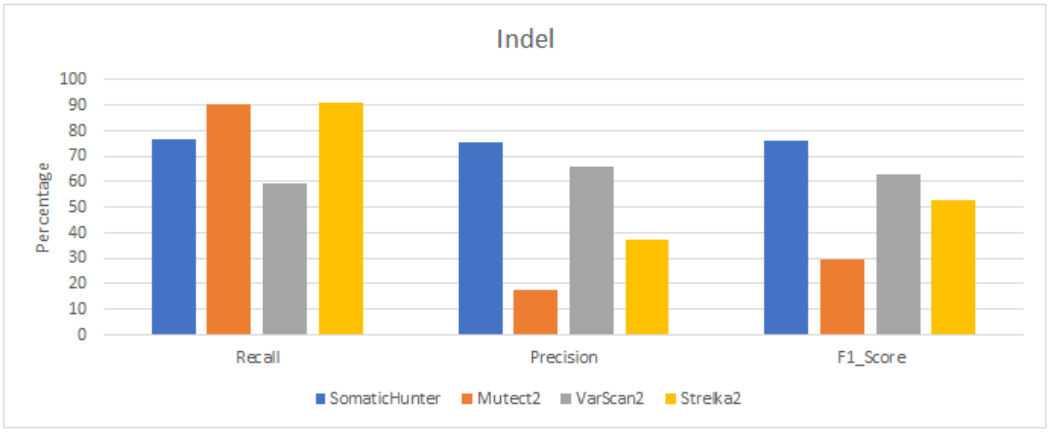
5.3 Results and Discussion

SomaticHuntuer is compared with widely used somatic variant calling tools based on dataset1 for Whole Genome Sequencing (WGS) and dataset2 for Whole Exome Sequencing (WXS).

5.3.1 Comparison of tools based on dataset1



(a)

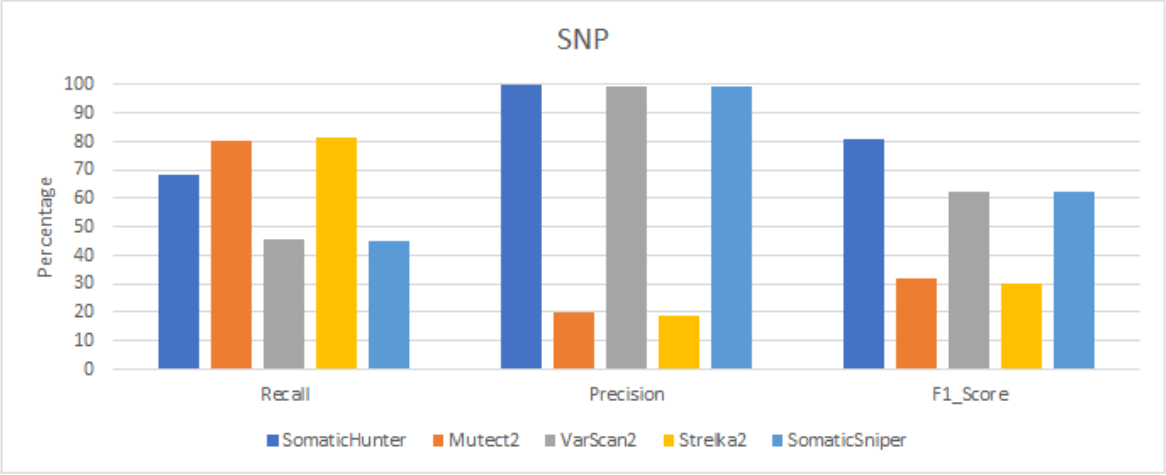


(b)

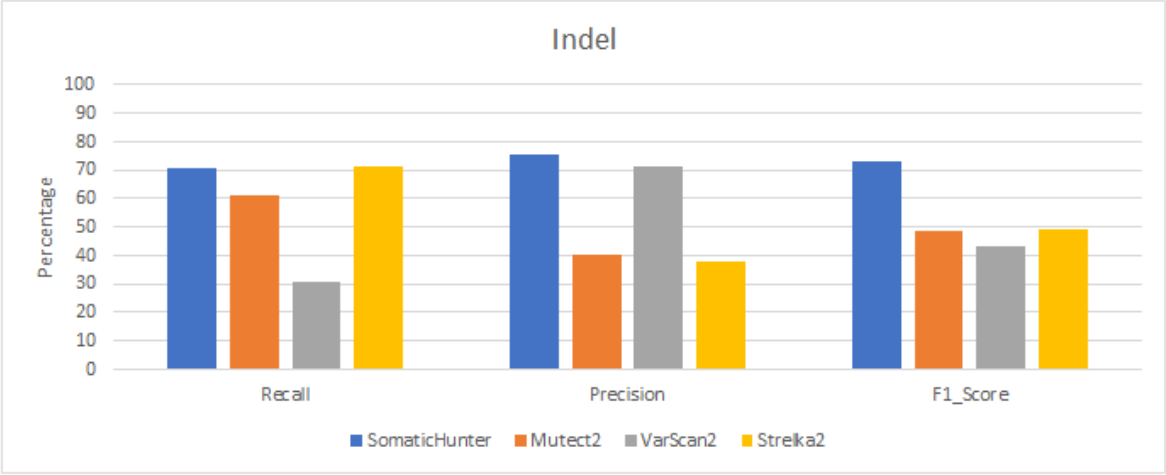
Figure 5.2: Comparison of somatic variant callers based on dataset1 for (a) SNPs and (b) indels.

Figure 5.2 shows the comparison result for dataset1. For SNPs (Figure 5.2a), Strelka2 and Mutect2 achieve high recall but low precision. On the other hand, VarScan2 and SomaticSniper achieve high precision but low recall. In both of these scenarios, the resultant F1 score is low for these tools. SomaticHunter, on the other hand, achieves recall comparable to the tools achieving high recall, and also produces the highest precision among all the tools. Therefore, the F1 score for SomaticHunter is the highest among the tools compared here. A similar observation is also made for somatic indels (Figure 5.2b).

5.3.2 Comparison of tools based on dataset2



(a)



(b)

Figure 5.3: Comparison of somatic variant callers based on dataset2 for (a) SNPs and (b) indels.

Figure 5.3 shows the comparison of the tools for dataset2. For SNPs (Figure 5.3a), Strelka2 and Mutect2 achieve high sensitivity, above 80%, but low precision, around 20% resulting in an F1 score of 30%. VarScan2 and SomaticSniper show very high precision, close to 99% but moderate precision (approximately 45%) which results in a fair F1 score of 62%. SomaticHunter achieves slightly higher precision than VarScan2 and SomaticSniper while maintaining a recall comparable to that of Strelka2 and Mutect2. Therefore, SomaticHunter achieves the highest F1 score among all the callers. A similar observation is made for indels (Figure 5.3b). However, the F1 scores for the other three tools are relatively close on this occasion.

5.4 Conclusion

Identifying somatic mutations is crucial to understand cancer better and also essential for making clinical decisions. Enormous data generated by next generation sequencing facilitate the analysis of cancer genomes. Many tools exist to identify somatic variants from this large amount of data. Some of these tools achieve high recall, which is desired, by not missing any important variant. However, this high sensitivity of the variant callers comes at the cost of low precision meaning that some of the germline variants can be mistakenly identified as somatic variants which are not desired. On the other hand, some somatic variant callers show the opposite characteristics. In both of these scenarios, the resultant F1 scores achieved by these tools are low. The F1 score is widely considered as a performance metric. To improve the performance of the existing somatic variant callers, ensembles of the tools can be a powerful solution. However, adding too many tools in the ensemble can increase the running time and may end up with a diminishing return in performance by incorporating too many false positive calls. A machine learning algorithm that can classify somatic and nonsomatic variants can be suitable to reduce false positives. This chapter describes SomaticHunter that combines the result from two sensitive variant callers Platypus and VarDict. The combined call set is passed to a classifier built using adaptive boosting with decision trees to classify each variant in the combined call set as somatic or nonsomatic. SomaticHunter is trained using DREAM Challenge stage 3 data and tested using virtual tumors, one Whole Genome Sequence (WGS) and one Whole Exome Sequence (WXS). After comparing with widely used somatic variant callers such as VarScan2, Mutect2, Strelka2, and SomaticSniper, results show that, for both data sets, SomaticHunter achieves highest precision (76% versus 13 to 75% for WGS SNP, 99.74% versus 19 to 99.61% for WXS SNP, 76% versus 18% to 66% for WGS indel, 76% versus 38% to 72% for WXS indel) while achieving a comparable recall (82% versus 20 to 93% for WGS SNP, 68% versus 45 to 81% for WXS SNP, 77% versus 62 to 90% for WGS indel, 71% versus 30 to 77% for WXS indel) for both SNPs and indels. Therefore, SomaticHunter achieves the highest F1 score (79% versus 23 to 35% for WGS SNP, 82% versus 31 to 62% for WXS SNP, 76% versus 29 to 63% for WGS indel, 74% versus 43 to 49% for WXS indel) making it suitable for downstream analysis of cancer genome that will help perform the appropriate clinical diagnosis.

Chapter 6

Conclusion and future prospects

Indels are one of the disease-associated variation types in humans. Hence, efficiently identifying indels is essential. This dissertation investigates seven well-known indel calling tools, namely, GATK Unified Genotyper, VarScan, Pindel, SAMtools, Dindel, GATK HaplotypeCaller, and Platypus using real sequence alignment of 78 individuals from the 1000 genomes project. Results show that there is substantial disagreement among the called results produced by these tools. Moreover, these tools show limited power in calling indels as a large number of indels listed in the benchmark data set are undetected. Essentially, these findings indicate that, despite importance in the human genome, indel calling is still challenging. This dissertation, therefore, aims to tackle these challenges using computation approaches.

Indels producing the same mutant sequence have the same biological effect and hence, are equivalent. Widely used public databases such as dbSNP and COSMIC store equivalent indels as distinct entries, which causes data redundancy and also artificially inflates the statistics of indel variations. This dissertation describes UPS-indel, a utility tool to create a universal positioning system called UPS-coordinates. The proposed UPS-coordinate is a range of positions where all indels equivalent to a specific indel can occur. Consequently, all equivalent indels have the same UPS-coordinate, which avoids possible annotation ambiguity. By checking the UPS-coordinate, one can easily filter out redundant indels from variant databases mentioned above. Moreover, UPS-indel searches for all the equivalent indels exhaustively, and therefore identifies more redundant indels than existing variant normalization algorithms. Additionally, UPS-indel is sufficiently robust to handle complex variants that are usually not processed by existing variant normalization approaches. For future work, two aspects can be improved for UPS-indel. First, the current version does not handle haplotypes formed by combination of neighboring indels. Updated version of UPS-indel can add this feature. Second, it will be interesting to see how annotations change due to the shift in indel positions. For example, indels in exons can end up in a position in introns or vice versa due to normalization, which can alter the functional annotation of the indels. Future version of UPS-indel should consider this scenario while checking redundancy to make sure that equivalent indels reside in the same genomic region as well.

This dissertation describes Genesis-indel, a computational pipeline to rescue novel high-quality indels by exploring unmapped reads that are usually discarded from downstream analysis. Application to breast cancer data proves that Genesis-indel is capable to effectively rescue missed indels that could be promising genetic markers for cancer and diseases. Therefore, Genesis-indel is expected to be highly useful to the research community. Future work can include the deployment of Genesis-indel in a secured cloud infrastructure that allows users to process a large cohort of patient data within a reasonable time frame.

Finally, identifying somatic mutations is crucial for a better understanding of cancer. Currently, many tools exist to find these variants from the next generation sequencing data. Some identify a large number of variants, resulting in high sensitivity but low precision. Others show the opposite characteristics. In both scenarios, the tools achieve low F1 scores, a widely considered performance metric. To improve the performance of existing somatic variant callers, this dissertation describes SomaticHunter, an ensemble of two variant callers Platypus and VarDict. The combined call set produced by these callers is passed to a classifier built using adaptive boosting with decision trees to classify each variant as somatic or nonsomatic. Comparison with widely used somatic variant callers shows that SomaticHunter has better overall performance for both SNP and indel calling. For future and further improvement, other variant callers can be incorporated in addition to Platypus and VarDict. Incorporating additional variant callers likely improves sensitivity, but the trade-off is the increase in computation time. Therefore, the number of callers incorporated and computation time needs to be examined closely to find a reasonable balance between prediction and time performance. Currently, SomaticHunter uses normal-tumor pair for identifying somatic variants. Although matched normal-tumor pair for an induvial is preferred, obtaining such a pair is not always feasible for liquid cancers such as leukemia. Moreover, requiring such pair doubles the number of samples to be sequenced, limiting the number of patients to be investigated. Therefore, SomaticHunter can be extended to integrate a tumor only mode that will allow studying somatic variants in a large cohort. Applying SomaticHunter to real cancer patient data will also be exciting and will let us get real insight into somatic variants.

Next-generation sequencing has facilitated the generation of an enormous amount of sequencing data at an individual level to aid in identifying genetic variants. However, identifying and

analyzing indels, the second most abundant variants in the human genome, remains challenging. This dissertation describes computational approaches to deal with these challenges. Techniques explained in this dissertation are necessary steps toward further improvement in identifying and analyzing indels that will substantially benefit the research community.

Bibliography

[1] T. R. Bhangale, M. J. Rieder, R. J. Livingston, and D. A. Nickerson, "Comprehensive identification and characterization of diallelic insertion-deletion polymorphisms in 330 human candidate genes," *Human Molecular Genetics,* vol. 14, no. 1, pp. 59-69, Jan 1 2005.

[2] E. Dawson *et al.*, "A SNP resource for human chromosome 22: extracting dense clusters of SNPs from the genomic sequence," *Genome Research,* vol. 11, no. 1, pp. 170-8, Jan 2001.

[3] J. M. Mullaney, R. E. Mills, W. S. Pittard, and S. E. Devine, "Small insertions and deletions (INDELs) in human genomes," *Human Molecular Genetics,* vol. 19, no. R2, pp. R131-6, Oct 15 2010.

[4] R. E. Mills *et al.*, "An initial map of insertion and deletion (INDEL) variation in the human genome," *Genome Research,* vol. 16, no. 9, pp. 1182-90, Sep 2006.

[5] F. S. Collins, M. L. Drumm, J. L. Cole, W. K. Lockwood, G. F. Vande Woude, and M. C. Iannuzzi, "Construction of a general human chromosome jumping library, with application to cystic fibrosis," *Science,* vol. 235, no. 4792, pp. 1046-9, Feb 27 1987.

[6] S. T. Warren, F. Zhang, G. R. Licameli, and J. F. Peters, "The fragile X site in somatic cell hybrids: an approach for molecular cloning of fragile sites," *Science,* vol. 237, no. 4813, pp. 420-3, Jul 24 1987.

[7] K. Usdin, "The biological effects of simple tandem repeats: lessons from the repeat expansion diseases," *Genome Research,* vol. 18, no. 7, pp. 1011-9, Jul 2008.

[8] D. G. MacArthur and C. Tyler-Smith, "Loss-of-function variants in the genomes of healthy humans," *Human Molecular Genetics,* vol. 19, no. R2, pp. R125-30, Oct 15 2010.

[9] T. Kaneo, S. Tahara, and M. Matsuo, "Non-linear accumulation of 8-hydroxy-2′-deoxyguanosine, a marker of oxidized DNA damage, during aging," *Mutation Research/DNAging,* vol. 316, no. 5, pp. 277-285, 1996.

[10] P. Paschka *et al.*, "Adverse prognostic significance of KIT mutations in adult acute myeloid leukemia with inv(16) and t(8;21): a Cancer and Leukemia Group B Study," *Journal of Clinical Oncology,* vol. 24, no. 24, pp. 3904-11, Aug 20 2006.

[11] B. Falini *et al.*, "Cytoplasmic nucleophosmin in acute myelogenous leukemia with a normal karyotype," *New England Journal of Medicine,* vol. 352, no. 3, pp. 254-66, Jan 20 2005.

[12] M. Nakao *et al.*, "Internal tandem duplication of the flt3 gene found in acute myeloid leukemia," (in English), *Leukemia,* vol. 10, no. 12, pp. 1911-1918, Dec 1996.

[13] L. V. Sequist *et al.*, "First-line gefitinib in patients with advanced non-small-cell lung cancer harboring somatic EGFR mutations," *Journal of Clinical Oncology,* vol. 26, no. 15, pp. 2442-9, May 20 2008.

[14] E. M. Ostertag and H. H. Kazazian, Jr., "Biology of mammalian L1 retrotransposons," *Annual Review of Genetics,* vol. 35, no. 1, pp. 501-38, 2001.

[15] V. G. Cheung and R. S. Spielman, "Genetics of human gene expression: mapping DNA variants that influence gene expression," (in English), *Nature Reviews Genetics,* vol. 10, no. 9, pp. 595-604, Sep 2009.

[16] S. A. Lee *et al.*, "Naturally Occurring Hepatitis B Virus X Deletions and Insertions Among Korean Chronic Patients," (in English), *Journal of Medical Virology,* vol. 83, no. 1, pp. 65-70, Jan 2011.

[17] M. S. Hasan and L. Zhang, "P-Dindel: A multi-thread based tool for calling indels from short reads," in *11th International Symposium on Bioinformatics Research and Applications*, 2015, pp. 71-74.
[18] M. S. Hasan and L. Zhang, "SPAI: an interactive platform for indel analysis," *BMC Genomics,* vol. 17, no. 5, p. 496, Aug 31 2016.
[19] L. Ding *et al.*, "Genome remodelling in a basal-like breast cancer metastasis and xenograft," *Nature,* vol. 464, no. 7291, pp. 999-1005, Apr 15 2010.
[20] P. Krawitz, C. Rodelsperger, M. Jager, L. Jostins, S. Bauer, and P. N. Robinson, "Microindel detection in short-read sequence data," *Bioinformatics,* vol. 26, no. 6, pp. 722-9, Mar 15 2010.
[21] G. CT, "Primer: Sequencing—the next generation," *Nature Methods,* vol. 5, no. 1, p. 15, 2008.
[22] M. L. Metzker, "Sequencing technologies—the next generation," (in English), *Nature Reviews Genetics,* vol. 11, no. 1, pp. 31-46, Jan 2010.
[23] E. R. Mardis, "Next-generation DNA sequencing methods," *Annual Review of Genomics and Human Genetics,* vol. 9, pp. 387-402, 2008.
[24] H. Li and R. Durbin, "Fast and accurate long-read alignment with Burrows-Wheeler transform," *Bioinformatics,* vol. 26, no. 5, pp. 589-95, Mar 1 2010.
[25] H. Matsumura *et al.*, "High-Throughput SuperSAGE for Digital Gene Expression Analysis of Multiple Samples Using Next Generation Sequencing," (in English), *Plos One,* vol. 5, no. 8, p. e12010, Aug 6 2010.
[26] C. A. Albers, G. Lunter, D. G. MacArthur, G. McVean, W. H. Ouwehand, and R. Durbin, "Dindel: accurate indel calls from short-read data," *Genome Research,* vol. 21, no. 6, pp. 961-73, Jun 2011.
[27] G. Lunter and M. Goodson, "Stampy: a statistical algorithm for sensitive and fast mapping of Illumina sequence reads," *Genome Research,* vol. 21, no. 6, pp. 936-9, Jun 2011.
[28] H. Li *et al.*, "The Sequence Alignment/Map format and SAMtools," *Bioinformatics,* vol. 25, no. 16, pp. 2078-9, Aug 15 2009.
[29] M. A. DePristo *et al.*, "A framework for variation discovery and genotyping using next-generation DNA sequencing data," (in English), *Nature Genetics,* vol. 43, no. 5, pp. 491-+, May 2011.
[30] D. C. Koboldt *et al.*, "VarScan: variant detection in massively parallel sequencing of individual and pooled samples," *Bioinformatics,* vol. 25, no. 17, pp. 2283-5, Sep 1 2009.
[31] D. C. Koboldt *et al.*, "VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing," *Genome Research,* vol. 22, no. 3, pp. 568-76, Mar 2012.
[32] K. Ye, M. H. Schulz, Q. Long, R. Apweiler, and Z. Ning, "Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads," *Bioinformatics,* vol. 25, no. 21, pp. 2865-71, Nov 1 2009.
[33] D. Grimm, J. Hagmann, D. Koenig, D. Weigel, and K. Borgwardt, "Accurate indel prediction using paired-end short reads," (in English), *BMC Genomics,* vol. 14, no. 1, p. 132, Feb 27 2013.
[34] J. O. Korbel *et al.*, "PEMer: a computational framework with simulation-based error models for inferring genomic structural variants from massive paired-end sequencing data," *Genome Biology,* vol. 10, no. 2, p. R23, Feb 23 2009.

[35] A. R. Quinlan *et al.*, "Genome-wide mapping and assembly of structural variant breakpoints in the mouse genome," *Genome Research,* vol. 20, no. 5, pp. 623-35, May 2010.
[36] K. Chen *et al.*, "BreakDancer: an algorithm for high-resolution mapping of genomic structural variation," *Nature Methods,* vol. 6, no. 9, pp. 677-81, Sep 2009.
[37] G. A. Van der Auwera *et al.*, "From FastQ data to high-confidence variant calls: the genome analysis toolkit best practices pipeline," *Current protocols in bioinformatics,* vol. 43, no. 1, pp. 11-10, 2013.
[38] A. Rimmer *et al.*, "Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications," *Nature Genetics,* vol. 46, no. 8, pp. 912-918, Aug 2014.
[39] J. A. Neuman, O. Isakov, and N. Shomron, "Analysis of insertion-deletion from deep-sequencing data: software evaluation for optimal detection," *Briefings in Bioinformatics,* vol. 14, no. 1, pp. 46-55, Jan 2013.
[40] H. J. Abel and E. J. Duncavage, "Detection of structural DNA variation from next generation sequencing data: a review of informatic approaches," *Cancer Genetics,* vol. 206, no. 12, pp. 432-40, Dec 2013.
[41] M. Via García and G. P. Consortium, "An integrated map of genetic variation from 1,092 human genomes," *Nature,* 2012.
[42] D. R. Bentley *et al.*, "Accurate whole human genome sequencing using reversible terminator chemistry," (in English), *Nature,* vol. 456, no. 7218, pp. 53-59, Nov 6 2008.
[43] R. E. Mills *et al.*, "Natural genetic variation caused by small insertions and deletions in the human genome," *Genome research,* vol. 21, no. 6, pp. 830-839, 2011.
[44] C. Whelan, "Detecting and Analyzing Genomic Structural Variation Using Distributed Computing," Doctor of Philosophy in Computer Science & Engineering, School of Medicine, Oregon Health and Science University, 2014.
[45] C. W. Whelan, J. Tyner, A. L'Abbate, C. T. Storlazzi, L. Carbone, and K. Sönmez, "Cloudbreak: accurate and scalable genomic structural variation detection in the cloud with MapReduce," *arXiv preprint arXiv:1307.2331,* 2013.
[46] J. M. Zook and M. Salit, "Genomes in a bottle: creating standard reference materials for genomic variation-why, what and how?," *Genome Biology,* vol. 12, pp. 1-27, 2011.
[47] Z. Li, X. Wu, B. He, and L. Zhang, "Vindel: a simple pipeline for checking indel redundancy," *BMC Bioinformatics,* vol. 15, no. 1, p. 359, 2014.
[48] H. Fang *et al.*, "Reducing INDEL calling errors in whole genome and exome sequencing data," *Genome Medicine,* vol. 6, no. 10, p. 89, 2014.
[49] H. T. Meng *et al.*, "Genetic polymorphism analyses of 30 InDels in Chinese Xibe ethnic group and its population genetic differentiations with other groups," *Scientific Reports,* vol. 5, p. 8260, Feb 5 2015.
[50] S. M. Ahn *et al.*, "The first Korean genome sequence and analysis: full genome sequencing for a socio-ethnic group," *Genome Research,* vol. 19, no. 9, pp. 1622-9, Sep 2009.
[51] (2014, November 29). *Should I analyze my samples alone or together?* Available: https://www.broadinstitute.org/gatk/guide/article?id=4150
[52] J. M. Zook *et al.*, "Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls," *Nature Biotechnology,* vol. 32, no. 3, pp. 246-51, Mar 2014.

[53] J. Assmus, J. Kleffe, A. O. Schmitt, and G. A. Brockmann, "Equivalent Indels - Ambiguous Functional Classes and Redundancy in Databases," (in English), *Plos One,* vol. 8, no. 5, p. e62803, May 2 2013.
[54] A. Tan, G. R. Abecasis, and H. M. Kang, "Unified representation of genetic variants," *Bioinformatics,* vol. 31, no. 13, pp. 2202-4, Jul 1 2015.
[55] C. Sun and P. Medvedev, "VarMatch: robust matching of small variant datasets using flexible scoring schemes," *Bioinformatics,* p. btw797, 2016.
[56] M. S. Hasan, X. Wu, and L. Zhang, "Performance evaluation of indel calling tools using real short-read data," *Human genomics,* vol. 9, no. 1, p. 20, 2015.
[57] T. G. P. Consortium, "A map of human genome variation from population-scale sequencing," *Nature,* vol. 467, no. 7319, pp. 1061-1073, 2010.
[58] H. Fang *et al.*, "Indel variant analysis of short-read sequencing data with Scalpel," *Nature Protocols,* vol. 11, pp. 2529–2548, 2016.
[59] A. Doring, D. Weese, T. Rausch, and K. Reinert, "SeqAn an efficient, generic C++ library for sequence analysis," *BMC Bioinformatics,* vol. 9, no. 1, p. 11, Jan 9 2008.
[60] J. M. Zook *et al.*, "Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls," *Nature Biotechnology,* vol. 32, no. 3, pp. 246-251, 2014.
[61] V. Trubetskoy *et al.*, "Consensus Genotyper for Exome Sequencing (CGES): improving the quality of exome variant genotypes," *Bioinformatics,* vol. 31, no. 2, pp. 187-93, Jan 15 2015.
[62] P. Danecek *et al.*, "The variant call format and VCFtools," *Bioinformatics,* vol. 27, no. 15, pp. 2156-8, Aug 1 2011.
[63] J. G. Cleary *et al.*, "Comparing Variant Call Files for Performance Benchmarking of Next-Generation Sequencing Variant Calling Pipelines," *bioRxiv,* p. 023754, 2015.
[64] R. Wittler, T. Marschall, A. Schonhuth, and V. Makinen, "Repeat- and error-aware comparison of deletions," (in English), *Bioinformatics,* vol. 31, no. 18, pp. 2947-54, Sep 15 2015.
[65] H. Li, "Towards better understanding of artifacts in variant calling from high-coverage samples," *Bioinformatics,* p. btu356, 2014.
[66] S. A. Forbes *et al.*, "COSMIC: exploring the world's knowledge of somatic mutations in human cancer," *Nucleic Acids Research,* vol. 43, no. Database issue, pp. D805-11, Jan 2015.
[67] G. Highnam *et al.*, "An analytical framework for optimizing variant discovery from personal genomes," *Nature Communications,* vol. 6, p. 6275, Feb 25 2015.
[68] H. Li, J. Ruan, and R. Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," (in English), *Genome Research,* vol. 18, no. 11, pp. 1851-1858, Nov 2008.
[69] R. Li, Y. Li, K. Kristiansen, and J. Wang, "SOAP: short oligonucleotide alignment program," *Bioinformatics,* vol. 24, no. 5, pp. 713-4, Mar 1 2008.
[70] H. Li and R. Durbin, "Fast and accurate short read alignment with Burrows-Wheeler transform," *Bioinformatics,* vol. 25, no. 14, pp. 1754-60, Jul 15 2009.
[71] B. Langmead, C. Trapnell, M. Pop, and S. L. Salzberg, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biology,* vol. 10, no. 3, p. R25, 2009.
[72] B. Langmead and S. L. Salzberg, "Fast gapped-read alignment with Bowtie 2," *Nature Methods,* vol. 9, no. 4, pp. 357-9, Mar 4 2012.

[73] M. Zaharia *et al.*, "Faster and more accurate sequence alignment with SNAP," *arXiv preprint arXiv:1111.5572,* 2011.
[74] R. Li *et al.*, "SOAP2: an improved ultrafast tool for short read alignment," *Bioinformatics,* vol. 25, no. 15, pp. 1966-7, Aug 1 2009.
[75] D. C. Koboldt, L. Ding, E. R. Mardis, and R. K. Wilson, "Challenges of sequencing human genomes," *Briefings in Bioinformatics,* vol. 11, no. 5, pp. 484-98, Sep 2010.
[76] T. Mitsudomi and Y. Yatabe, "Epidermal growth factor receptor in relation to tumor development: EGFR gene and cancer," *The FEBS Journal,* vol. 277, no. 2, pp. 301-8, Jan 2010.
[77] H. Yasuda, S. Kobayashi, and D. B. Costa, "EGFR exon 20 insertion mutations in non-small-cell lung cancer: preclinical data and clinical implications," *Lancet Oncology,* vol. 13, no. 1, pp. e23-31, Jan 2012.
[78] N. Cancer Genome Atlas Research *et al.*, "The Cancer Genome Atlas Pan-Cancer analysis project," *Nature Genetics,* vol. 45, no. 10, pp. 1113-20, Oct 2013.
[79] A. M. Bolger, M. Lohse, and B. Usadel, "Trimmomatic: a flexible trimmer for Illumina sequence data," *Bioinformatics,* vol. 30, no. 15, pp. 2114-20, Aug 1 2014.
[80] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM," *arXiv preprint arXiv:1303.3997,* 2013.
[81] W. J. Kent, "BLAT--the BLAST-like alignment tool," (in English), *Genome Research,* vol. 12, no. 4, pp. 656-64, Apr 2002.
[82] (2012, 05/13/2018). *A list of oncogenes and tumor suppressors used in the comparison of gene functional groups*. Available: http://cancerres.aacrjournals.org/content/canres/suppl/2012/01/23/0008-5472.CAN-11-2266.DC1/T3_74K.pdf
[83] M. E. Higgins, M. Claremont, J. E. Major, C. Sander, and A. E. Lash, "CancerGenes: a gene selection resource for cancer genome projects," *Nucleic Acids Research,* vol. 35, no. Database issue, pp. D721-6, Jan 2007.
[84] X. Peng, J. Wang, Z. Zhang, Q. Xiao, M. Li, and Y. Pan, "Re-alignment of the unmapped reads with base quality score," in *BMC Bioinformatics*, 2015, vol. 16, no. 5, p. S8: BioMed Central.
[85] M. S. Hasan, X. Wu, L. T. Watson, and L. Zhang, "UPS-indel: a Universal Positioning System for Indels," *Scientific Reports,* vol. 7, no. 1, p. 14106, Oct 26 2017.
[86] H. Thorvaldsdottir, J. T. Robinson, and J. P. Mesirov, "Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration," *Briefings in Bioinformatics,* vol. 14, no. 2, pp. 178-92, Mar 2013.
[87] P. Iengar, "An analysis of substitution, deletion and insertion mutations in cancer genes," *Nucleic Acids Research,* vol. 40, no. 14, pp. 6401-13, Aug 2012.
[88] N. Rampino *et al.*, "Somatic frameshift mutations in the BAX gene in colon cancers of the microsatellite mutator phenotype," *Science,* vol. 275, no. 5302, pp. 967-9, Feb 14 1997.
[89] J. Li *et al.*, "PTEN, a putative protein tyrosine phosphatase gene mutated in human brain, breast, and prostate cancer," *Science,* vol. 275, no. 5308, pp. 1943-7, Mar 28 1997.
[90] Y. Ogura *et al.*, "A frameshift mutation in NOD2 associated with susceptibility to Crohn's disease," *Nature,* vol. 411, no. 6837, pp. 603-6, May 31 2001.
[91] B. D. Ondov *et al.*, "Mash: fast genome and metagenome distance estimation using MinHash," (in English), *Genome Biology,* vol. 17, no. 1, p. 132, Jun 20 2016.

[92] C. G. A. Network, "Comprehensive molecular portraits of human breast tumours," *Nature,* vol. 490, no. 7418, p. 61, 2012.
[93] A. R. Quinlan and I. M. Hall, "BEDTools: a flexible suite of utilities for comparing genomic features," *Bioinformatics,* vol. 26, no. 6, pp. 841-2, Mar 15 2010.
[94] F. P. Silva *et al.*, "Identification of RUNX1/AML1 as a classical tumor suppressor gene," (in English), *Oncogene,* vol. 22, no. 4, pp. 538-47, Jan 30 2003.
[95] H. Miyoshi, K. Shimizu, T. Kozu, N. Maseki, Y. Kaneko, and M. Ohki, "t (8; 21) breakpoints on chromosome 21 in acute myeloid leukemia are clustered within a limited region of a single gene, AML1," *Proceedings of the National Academy of Sciences,* vol. 88, no. 23, pp. 10431-10434, 1991.
[96] N. Ferrari *et al.*, "Expression of RUNX1 correlates with poor patient prognosis in triple negative breast cancer," *PLoS One,* vol. 9, no. 6, p. e100759, 2014.
[97] G. Browne *et al.*, "Runx1 is associated with breast cancer progression in MMTV-PyMT transgenic mice and its depletion in vitro inhibits migration and invasion," *Journal of Cellular Physiology,* vol. 230, no. 10, pp. 2522-2532, 2015.
[98] K. A. Janes, "RUNX1 and its understudied role in breast cancer," *Cell Cycle,* vol. 10, no. 20, pp. 3461-5, Oct 15 2011.
[99] W. McLaren *et al.*, "The Ensembl Variant Effect Predictor," *Genome Biology,* vol. 17, no. 1, p. 122, Jun 6 2016.
[100] J. L. Rinn and H. Y. Chang, "Genome regulation by long noncoding RNAs," *Annual Review of Biochemistry,* vol. 81, pp. 145-66, 2012.
[101] T. C. Roberts, K. V. Morris, and M. S. Weinberg, "Perspectives on the mechanism of transcriptional regulation by long non-coding RNAs," *Epigenetics,* vol. 9, no. 1, pp. 13-20, Jan 2014.
[102] J. M. Silva, N. J. Boczek, M. W. Berres, X. Ma, and D. I. Smith, "LSINCT5 is over expressed in breast and ovarian cancer and affects cellular proliferation," *RNA Biology,* vol. 8, no. 3, pp. 496-505, 2011.
[103] R. A. Gupta *et al.*, "Long non-coding RNA HOTAIR reprograms chromatin state to promote cancer metastasis," *Nature,* vol. 464, no. 7291, pp. 1071-6, Apr 15 2010.
[104] G. Dennis, Jr. *et al.*, "DAVID: Database for Annotation, Visualization, and Integrated Discovery," *Genome Biology,* vol. 4, no. 5, p. P3, 2003.
[105] J. V. Fresno, E. Casado, P. Cejas, C. Belda-Iniesta, and M. González-Barón, "PI3K/Akt signalling pathway and cancer," *Cancer Treatment Reviews,* vol. 30, no. 2, pp. 193-204, 2004.
[106] Y. L. Zhang, R. C. Wang, K. Cheng, B. Z. Ring, and L. Su, "Roles of Rap1 signaling in tumor cell migration and invasion," *Cancer Biology and Medicine,* vol. 14, no. 1, pp. 90-99, Feb 2017.
[107] J. Downward, "Targeting RAS signalling pathways in cancer therapy," *Nature Reviews Cancer,* vol. 3, no. 1, pp. 11-22, Jan 2003.
[108] C. W. Reuter, M. A. Morgan, and L. Bergmann, "Targeting the Ras signaling pathway: a rational, mechanism-based treatment for hematologic malignancies?," *Blood,* vol. 96, no. 5, pp. 1655-69, Sep 1 2000.
[109] S. A. Teichmann and C. Chothia, "Immunoglobulin superfamily proteins in Caenorhabditis elegans," *Journal of Molecular Biology,* vol. 296, no. 5, pp. 1367-83, Mar 10 2000.

[110] D. S. Ettinger, S. E. Order, M. D. Wharam, M. K. Parker, J. L. Klein, and P. K. Leichner, "Phase I-II study of isotopic immunoglobulin therapy for primary liver cancer," *Cancer Treatment Reports,* vol. 66, no. 2, pp. 289-97, Feb 1982.
[111] A. Musolino *et al.*, "Immunoglobulin G fragment C receptor polymorphisms and clinical efficacy of trastuzumab-based therapy in patients with HER-2/neu-positive metastatic breast cancer," *Journal of Clinical Oncology,* vol. 26, no. 11, pp. 1789-96, Apr 10 2008.
[112] W. K. Weng and R. Levy, "Two immunoglobulin G fragment C receptor polymorphisms independently predict response to rituximab in patients with follicular lymphoma," *Journal of Clinical Oncology,* vol. 21, no. 21, pp. 3940-7, Nov 1 2003.
[113] W. K. Weng, D. Czerwinski, J. Timmerman, F. J. Hsu, and R. Levy, "Clinical outcome of lymphoma patients after idiotype vaccination is correlated with humoral immune response and immunoglobulin G Fc receptor genotype," (in English), *Journal of Clinical Oncology,* vol. 22, no. 23, pp. 4717-24, Dec 1 2004.
[114] C. Mark, M. Abrink, and L. Hellman, "Comparative analysis of KRAB zinc finger proteins in rodents and man: Evidence for several evolutionarily distinct subfamilies of KRAB zinc finger genes," (in English), *DNA and Cell Biology,* vol. 18, no. 5, pp. 381-396, May 1999.
[115] Y. D. Cheng *et al.*, "KRAB Zinc Finger Protein ZNF382 Is a Proapoptotic Tumor Suppressor That Represses Multiple Oncogenes and Is Commonly Silenced in Multiple Carcinomas," (in English), *Cancer Research,* vol. 70, no. 16, pp. 6516-6526, Aug 15 2010.
[116] N. Suzuki *et al.*, "A phase I clinical trial of vaccination with KIF20A-derived peptide in combination with gemcitabine for patients with advanced pancreatic cancer," *Journal of Immunotherapy,* vol. 37, no. 1, pp. 36-42, Jan 2014.
[117] R. D. Vale, T. S. Reese, and M. P. Sheetz, "Identification of a novel force-generating protein, kinesin, involved in microtubule-based motility," *Cell,* vol. 42, no. 1, pp. 39-50, Aug 1985.
[118] J. X. Zou *et al.*, "Kinesin family deregulation coordinated by bromodomain protein ANCCA and histone methyltransferase MLL for breast cancer cell growth, survival, and tamoxifen resistance," *Molecular Cancer Research,* vol. 12, no. 4, pp. 539-49, Apr 2014.
[119] P. Khongkow *et al.*, "Paclitaxel targets FOXM1 to regulate KIF20A in mitotic catastrophe and breast cancer paclitaxel resistance," *Oncogene,* vol. 35, no. 8, pp. 990-1002, Feb 25 2016.
[120] L. Groth-Pedersen, S. Aits, E. Corcelle-Termeau, N. H. Petersen, J. Nylandsted, and M. Jaattela, "Identification of cytoskeleton-associated proteins essential for lysosomal stability and survival of human cancer cells," *PLoS One,* vol. 7, no. 10, p. e45381, 2012.
[121] S. Claerhout *et al.*, "Gene Expression Signature Analysis Identifies Vorinostat as a Candidate Therapy for Gastric Cancer," (in English), *Plos One,* vol. 6, no. 9, p. e24662, Sep 9 2011.
[122] R. Neef, U. Gruneberg, and F. A. Barr, "Assay and functional properties of Rabkinesin-6/Rab6-KIFL/MKlp2 in cytokinesis," *Methods in Enzymology,* vol. 403, pp. 618-28, 2005.
[123] Y. Lu *et al.*, "Cross-species comparison of orthologous gene expression in human bladder cancer and carcinogen-induced rodent models," (in English), *American Journal of Translational Research,* vol. 3, no. 1, pp. 8-27, 2011.
[124] K. Taniuchi, M. Furihata, and T. Saibara, "KIF20A-mediated RNA granule transport system promotes the invasiveness of pancreatic cancer cells," *Neoplasia,* vol. 16, no. 12, pp. 1082-93, Dec 2014.

[125] D. Stangel *et al.*, "Kif20a inhibition reduces migration and invasion of pancreatic cancer cells," *Journal of Surgical Research,* vol. 197, no. 1, pp. 91-100, Jul 2015.
[126] K. Imai *et al.*, "Identification of HLA-A2-restricted CTL epitopes of a novel tumour-associated antigen, KIF20A, overexpressed in pancreatic cancer," *Br J Cancer,* vol. 104, no. 2, pp. 300-7, Jan 18 2011.
[127] I. Gasnereau *et al.*, "KIF20A mRNA and its product MKlp2 are increased during hepatocyte proliferation and hepatocarcinogenesis," *The American Journal of Pathology,* vol. 180, no. 1, pp. 131-40, Jan 2012.
[128] H. Fang *et al.*, "Quantitative T cell repertoire analysis by deep cDNA sequencing of T cell receptor α and β chains using next-generation sequencing (NGS)," *Oncoimmunology,* vol. 3, no. 12, p. e968467, 2014.
[129] K. Saito, S. Ohta, Y. Kawakami, K. Yoshida, and M. Toda, "Functional analysis of KIF20A, a potential immunotherapeutic target for glioma," *Journal of Neuro-Oncology,* vol. 132, no. 1, pp. 63-74, Mar 2017.
[130] J. Yamashita *et al.*, "Kinesin family member 20A is a novel melanoma-associated antigen," (in English), *Acta Derm Venereol,* vol. 92, no. 6, pp. 593-7, Nov 2012.
[131] G. C. Bobustuc *et al.*, "MGMT inhibition in ER positive breast cancer leads to CDC2, TOP2A, AURKB, CDC20, KIF20A, Cyclin A2, Cyclin B2, Cyclin D1, ERalpha and Survivin inhibition and enhances response to temozolomide," *Oncotarget,* vol. 9, no. 51, pp. 29727-29742, Jul 3 2018.
[132] J. R. Ho *et al.*, "Deregulation of Rab and Rab effector genes in bladder cancer," *PLoS One,* vol. 7, no. 6, p. e39469, 2012.
[133] K. Taniuchi *et al.*, "Down-regulation of RAB6KIFL/KIF20A, a kinesin involved with membrane trafficking of discs large homologue 5, can attenuate growth of pancreatic cancer cell," (in English), *Cancer Research,* vol. 65, no. 1, pp. 105-112, Jan 1 2005.
[134] W. Zhang *et al.*, "High Expression of KIF20A Is Associated with Poor Overall Survival and Tumor Progression in Early-Stage Cervical Squamous Cell Carcinoma," (in English), *PLoS One,* vol. 11, no. 12, p. e0167449, Dec 12 2016.
[135] S. Asahara, K. Takeda, K. Yamao, H. Maguchi, and H. Yamaue, "Phase I/II clinical trial using HLA-A24-restricted peptide vaccine derived from KIF20A for patients with advanced pancreatic cancer," (in English), *Journal of Translational Medicine,* vol. 11, no. 1, p. 291, Nov 16 2013.
[136] A. Aruga *et al.*, "Phase I clinical trial of multiple-peptide vaccination for patients with advanced biliary tract cancer," *Journal of Translational Medicine,* vol. 12, no. 1, p. 61, Mar 7 2014.
[137] Y. Fujiwara *et al.*, "Multiple therapeutic peptide vaccines for patients with advanced gastric cancer," *International Journal of Oncology,* vol. 50, no. 5, pp. 1655-1662, May 2017.
[138] M. Miyazawa *et al.*, "Phase II clinical trial using novel peptide cocktail vaccine as a postoperative adjuvant treatment for surgically resected pancreatic cancer patients," (in English), *International Journal of Cancer,* vol. 140, no. 4, pp. 973-982, Feb 15 2017.
[139] J. J. Louw *et al.*, "Compound heterozygous loss-of-function mutations in KIF20A are associated with a novel lethal congenital cardiomyopathy in two siblings," *PLoS Genetics,* vol. 14, no. 1, p. e1007138, Jan 2018.
[140] K. J. Karczewski *et al.*, "The ExAC browser: displaying reference data information from over 60 000 exomes," *Nucleic Acids Research,* vol. 45, no. D1, pp. D840-D845, Jan 4 2017.

[141] M. Lek *et al.*, "Analysis of protein-coding genetic variation in 60,706 humans," *Nature,* vol. 536, no. 7616, pp. 285-91, Aug 18 2016.
[142] H. J. Schulten *et al.*, "Comprehensive molecular biomarker identification in breast cancer brain metastases," *Journal of Translational Medicine,* vol. 15, no. 1, p. 269, Dec 29 2017.
[143] P. Li *et al.*, "Increased ZNF84 expression in cervical cancer," *Archives of Gynecology and Obstetrics,* vol. 297, no. 6, pp. 1525-1532, Jun 2018.
[144] J. Li *et al.*, "TANRIC: An Interactive Open Platform to Explore the Function of lncRNAs in Cancer," *Cancer Research,* vol. 75, no. 18, pp. 3728-37, Sep 15 2015.
[145] A. L. Borresen-Dale, "TP53 and breast cancer," *Human Mutation,* vol. 21, no. 3, pp. 292-300, Mar 2003.
[146] J. Li *et al.*, "ForestQC: quality control on genetic variants from next-generation sequencing data using random forest," *bioRxiv,* p. 444828, 2018.
[147] D. Hanahan and R. A. Weinberg, "Hallmarks of cancer: the next generation," *Cell,* vol. 144, no. 5, pp. 646-74, Mar 4 2011.
[148] I. Martincorena and P. J. Campbell, "Somatic mutation in cancer and normal cells," *Science,* vol. 349, no. 6255, pp. 1483-9, Sep 25 2015.
[149] B. Vogelstein, N. Papadopoulos, V. E. Velculescu, S. Zhou, L. A. Diaz, Jr., and K. W. Kinzler, "Cancer genome landscapes," *Science,* vol. 339, no. 6127, pp. 1546-58, Mar 29 2013.
[150] Z. Lai *et al.*, "VarDict: a novel and versatile variant caller for next-generation sequencing in cancer research," *Nucleic Acids Research,* vol. 44, no. 11, p. e108, Jun 20 2016.
[151] A. McKenna *et al.*, "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data," *Genome Research,* vol. 20, no. 9, pp. 1297-303, Sep 2010.
[152] S. Y. Kim, L. Jacob, and T. P. Speed, "Combining calls from multiple somatic mutation-callers," *BMC Bioinformatics,* vol. 15, no. 1, p. 154, May 21 2014.
[153] H. Xu, J. DiCarlo, R. V. Satya, Q. Peng, and Y. Wang, "Comparison of somatic mutation calling methods in amplicon and whole exome sequence data," *BMC Genomics,* vol. 15, no. 1, p. 244, Mar 28 2014.
[154] A. B. Kroigard, M. Thomassen, A. V. Laenkholm, T. A. Kruse, and M. J. Larsen, "Evaluation of Nine Somatic Variant Callers for Detection of Somatic Mutations in Exome and Targeted Deep Sequencing Data," *PLoS One,* vol. 11, no. 3, p. e0151664, 2016.
[155] T. S. Alioto *et al.*, "A comprehensive assessment of somatic mutation detection in cancer using whole-genome sequencing," (in English), *Nature Communications,* vol. 6, p. 10001, Dec 9 2015.
[156] S. Sandmann *et al.*, "Evaluating Variant Calling Tools for Non-Matched Next-Generation Sequencing Data," *Scientific Reports,* vol. 7, p. 43169, Feb 24 2017.
[157] G. Narzisi *et al.*, "Genome-wide somatic variant calling using localized colored de Bruijn graphs," (in English), *Communications Biology,* vol. 1, no. 1, p. 20, 2018.
[158] X. Liu, J. Wang, and L. Chen, "Whole-exome sequencing reveals recurrent somatic mutation networks in cancer," *Cancer Letters,* vol. 340, no. 2, pp. 270-6, Nov 1 2013.
[159] S. L. Carter *et al.*, "Absolute quantification of somatic DNA alterations in human cancer," *Nature Biotechnology,* vol. 30, no. 5, pp. 413-21, May 2012.
[160] G. Narzisi and M. C. Schatz, "The challenge of small-scale repeats for indel discovery," *Frontiers in Bioengineering and Biotechnology,* vol. 3, p. 8, 2015.

[161] G. P. Consortium, "A map of human genome variation from population-scale sequencing," *Nature,* vol. 467, no. 7319, p. 1061, 2010.
[162] G. P. Consortium, "An integrated map of genetic variation from 1,092 human genomes," *Nature,* vol. 491, no. 7422, p. 56, 2012.
[163] G. P. Consortium, "A global reference for human genetic variation," *Nature,* vol. 526, no. 7571, p. 68, 2015.
[164] G. Narzisi *et al.*, "Accurate de novo and transmitted indel detection in exome-capture data using microassembly," (in English), *Nature Methods,* vol. 11, no. 10, pp. 1033-6, Oct 2014.
[165] A. D. Ewing *et al.*, "Combining tumor genome simulation with crowdsourcing to benchmark somatic single-nucleotide-variant detection," *Nature Methods,* vol. 12, no. 7, pp. 623-30, Jul 2015.
[166] J. Meng and Y. P. Chen, "A database of simulated tumor genomes towards accurate detection of somatic small variants in cancer," *PLoS One,* vol. 13, no. 8, p. e0202982, 2018.
[167] S. Kim *et al.*, "Strelka2: fast and accurate calling of germline and somatic variants," *Nature Methods,* vol. 15, no. 8, pp. 591-594, Aug 2018.
[168] D. E. Larson *et al.*, "SomaticSniper: identification of somatic point mutations in whole genome sequencing data," *Bioinformatics,* vol. 28, no. 3, pp. 311-7, Feb 1 2012.

Appendix A

Supplementary Material of Chapter 2

Table A.1: List of the input samples with the corresponding population and read coverage

Sample ID	Sample	Ethnic Background	Average Coverage
1	HG00151	GBR (British in England and Scotland)	4.07945
2	HG00154	GBR (British in England and Scotland)	4.55756
3	HG00157	GBR (British in England and Scotland)	4.97155
4	HG00171	FIN (Finnish in Finland)	3.42147
5	HG00174	FIN (Finnish in Finland)	3.75145
6	HG00190	FIN (Finnish in Finland)	3.3264
7	HG00553	PUR (Puerto Rican in Puerto Rico)	3.20752
8	HG00560	CHS (Han Chinese South)	3.6613
9	HG00565	CHS (Han Chinese South)	3.77876
10	HG00580	CHS (Han Chinese South)	4.11136
11	HG00637	PUR (Puerto Rican in Puerto Rico)	3.46052
12	HG00638	PUR (Puerto Rican in Puerto Rico)	4.41764
13	HG00844	CDX (Chinese Dai in Xishuangbanna, China)	10.4013
14	HG00851	CDX (Chinese Dai in Xishuangbanna, China)	11.7362
15	HG00864	CDX (Chinese Dai in Xishuangbanna, China)	3.86342
16	HG01356	CLM (Colombian in Medellin, Colombia)	4.68011
17	HG01357	CLM (Colombian in Medellin, Colombia)	4.63414
18	HG01377	CLM (Colombian in Medellin, Colombia)	4.65449
19	HG01524	IBS (Spanish Iberian populations in Spain)	7.26326
20	HG01525	IBS (Spanish Iberian populations in Spain)	6.60486
21	HG01531	IBS (Spanish Iberian populations in Spain)	6.93617
22	HG01599	KHV (Kinh in Ho Chi Minh City, Vietnam)	4.52677
23	HG01860	KHV (Kinh in Ho Chi Minh City, Vietnam)	7.76182
24	HG01861	KHV (Kinh in Ho Chi Minh City, Vietnam)	7.47835
25	HG01882	ACB (African Caribbean in Barbados)	8.37547
26	HG01883	ACB (African Caribbean in Barbados)	7.84466
27	HG01892	PEL (Peruvian in Lima, Peru)	5.82321
28	HG01893	PEL (Peruvian in Lima, Peru)	6.20132
29	HG01914	ACB (African Caribbean in Barbados)	5.9745
30	HG01923	PEL (Peruvian in Lima, Peru)	6.97538
31	HG02813	GWD (Gambian in Western Division, The Gambia)	6.23018

32	HG02814	GWD (Gambian in Western Division, The Gambia)	6.10095
33	HG02839	GWD (Gambian in Western Division, The Gambia)	6.80767
34	HG03009	BEB (Bengali in Bangladesh)	6.04669
35	HG03234	PJL (Punjabi in Lahore, Pakistan)	7.25371
36	HG03235	PJL (Punjabi in Lahore, Pakistan)	6.17131
37	HG03268	ESN (Esan in Nigeria)	6.89912
38	HG03470	MSL (Mende in Sierra Leon)	9.82357
39	HG03473	MSL (Mende in Sierra Leon)	7.319
40	HG03514	ESN (Esan in Nigeria)	6.48069
41	HG03515	ESN (Esan in Nigeria)	8.20044
42	HG03571	MSL (Mende in Sierra Leon)	4.89722
43	HG03619	PJL (Punjabi in Lahore, Pakistan)	7.67182
44	HG03645	STU (Sri Lankan Tamil in the UK)	5.22783
45	HG03646	STU (Sri Lankan Tamil in the UK)	5.4842
46	HG03777	ITU (Indian Telugu in the UK)	6.40374
47	HG03785	ITU (Indian Telugu in the UK)	5.9841
48	HG03790	ITU (Indian Telugu in the UK)	6.51514
49	HG03808	BEB (Bengali in Bangladesh)	5.84893
50	HG03809	BEB (Bengali in Bangladesh)	5.32334
51	HG03999	STU (Sri Lankan Tamil in the UK)	5.76619
52	NA07000	CEU (Utah residents with Northern and Western European ancestry)	10.4452
53	NA07357	CEU (Utah residents with Northern and Western European ancestry)	6.18814
54	NA12234	CEU (Utah residents with Northern and Western European ancestry)	7.08588
55	NA18550	CHB (Han Chinese in Beijing, China)	7.78798
56	NA18570	CHB (Han Chinese in Beijing, China)	7.96559
57	NA18582	CHB (Han Chinese in Beijing, China)	8.63521
58	NA18944	JPT (Japanese in Tokyo, Japan)	5.58686
59	NA18973	JPT (Japanese in Tokyo, Japan)	4.97637
60	NA19005	JPT (Japanese in Tokyo, Japan)	7.20848
61	NA19238	YRI (Yoruba in Ibadan, Nigeria)	4.92286
62	NA19239	YRI (Yoruba in Ibadan, Nigeria)	4.67774
63	NA19240	YRI (Yoruba in Ibadan, Nigeria)	4.59382
64	NA19429	LWK (Luhya in Webuye, Kenya)	6.02545

65	NA19443	LWK (Luhya in Webuye, Kenya)	10.5005
66	NA19462	LWK (Luhya in Webuye, Kenya)	6.48215
67	NA19725	MXL (Mexican Ancestry in Los Angeles, California)	6.03405
68	NA19755	MXL (Mexican Ancestry in Los Angeles, California)	5.9912
69	NA19764	MXL (Mexican Ancestry in Los Angeles, California)	6.82051
70	NA19913	ASW (African Ancestry in Southwest US)	6.52439
71	NA20357	ASW (African Ancestry in Southwest US)	6.66108
72	NA20359	ASW (African Ancestry in Southwest US)	6.20712
73	NA20510	TSI (Toscani in Italia)	6.61414
74	NA20518	TSI (Toscani in Italia)	7.91557
75	NA20529	TSI (Toscani in Italia)	5.98378
76	NA20863	GIH (Gujarati Indian in Houston, TX)	7.7302
77	NA20867	GIH (Gujarati Indian in Houston, TX)	6.00576
78	NA20868	GIH (Gujarati Indian in Houston, TX)	6.68754

Table A.2: P-value for Chi-square statistical test of indel size distribution between the tools.

	VarScan	**Pindel**	**SAMtools**	**Dindel**	**GATK_HC**	**Platypus**
GATK_UG	0.99	0.69	0.8	0.27	0.35	0.99
VarScan		0.58	0.57	0.21	0.27	0.98
Pindel			0.71	0.98	0.99	0.88
SAMtools				0.96	0.98	0.58
Dindel					0.99	0.83
GATK_HC						0.80

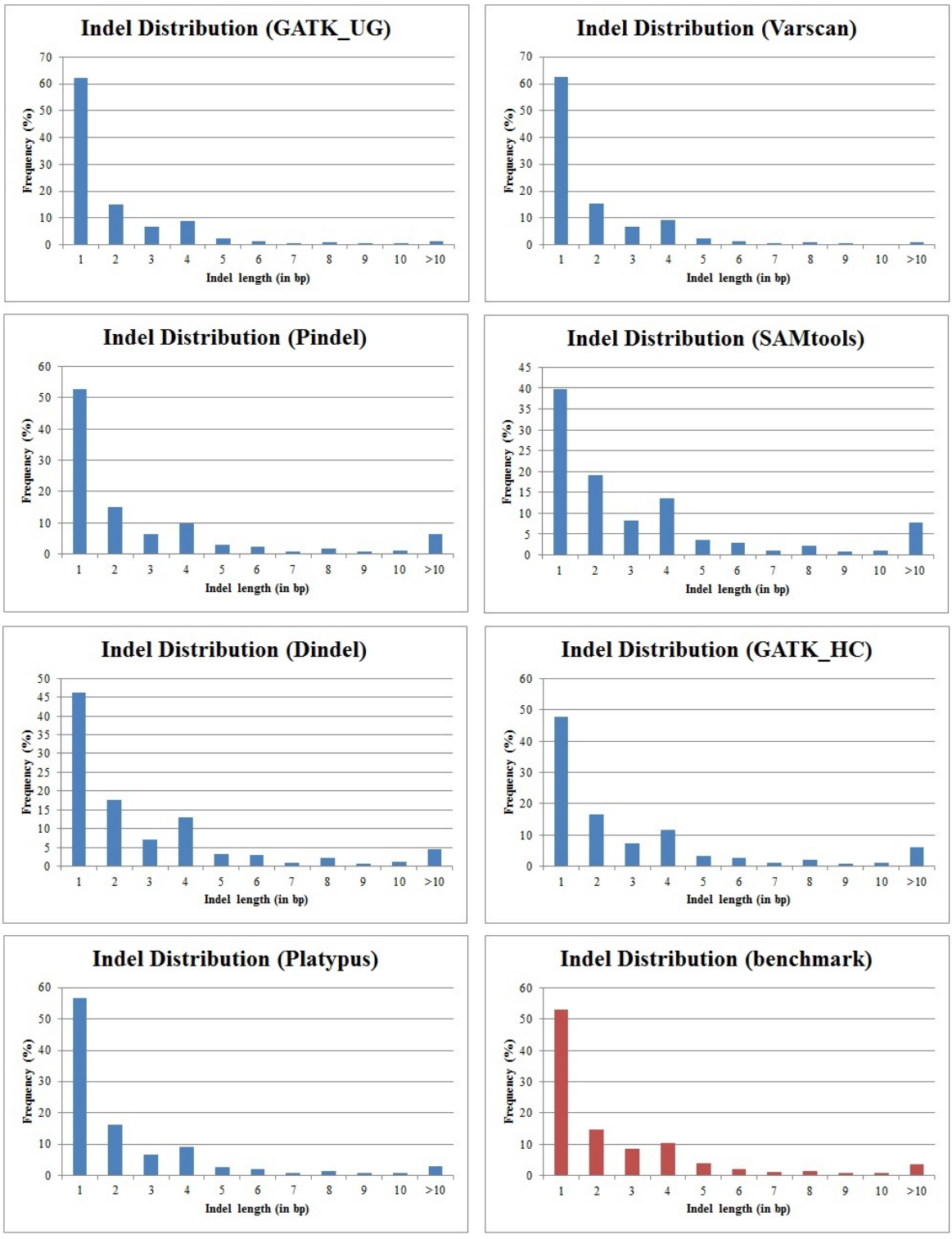


Figure A.1: Distribution of indels based on lengths (1 to 10 bp) for GATK_UG, VarScan, Pindel, SAMtools, Dindel, GATK_HC, Platypus and Benchmark data set.

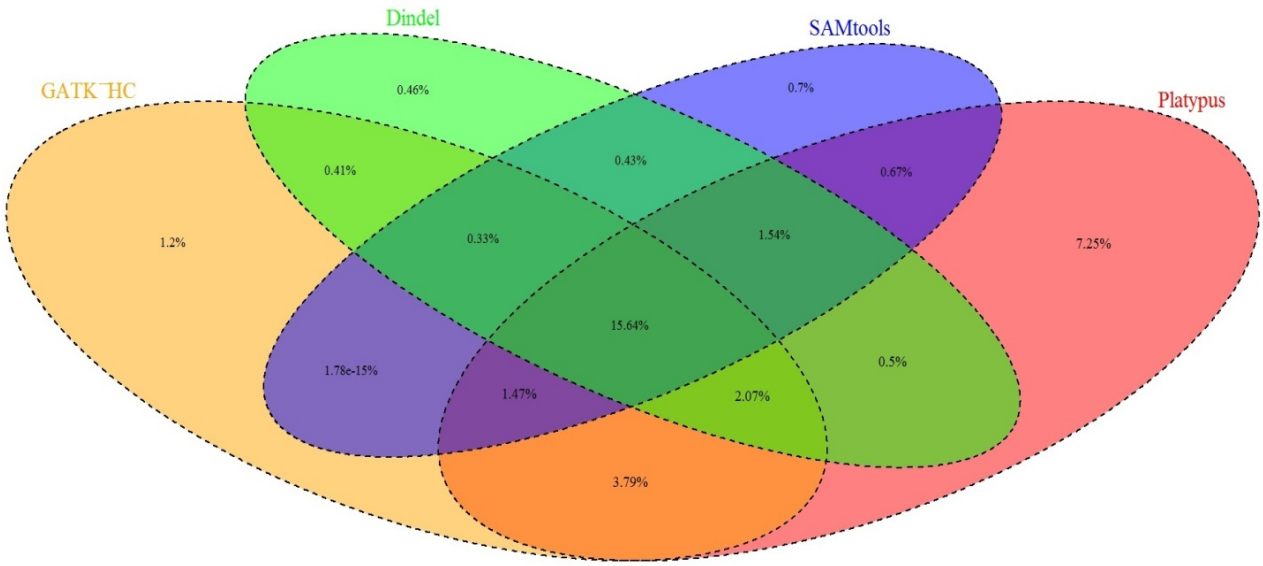


Figure A.2: Intra-tool comparison among GATK_HC, Dindel, SAMtools, and Platypus for a percentage of their own indels called by others.

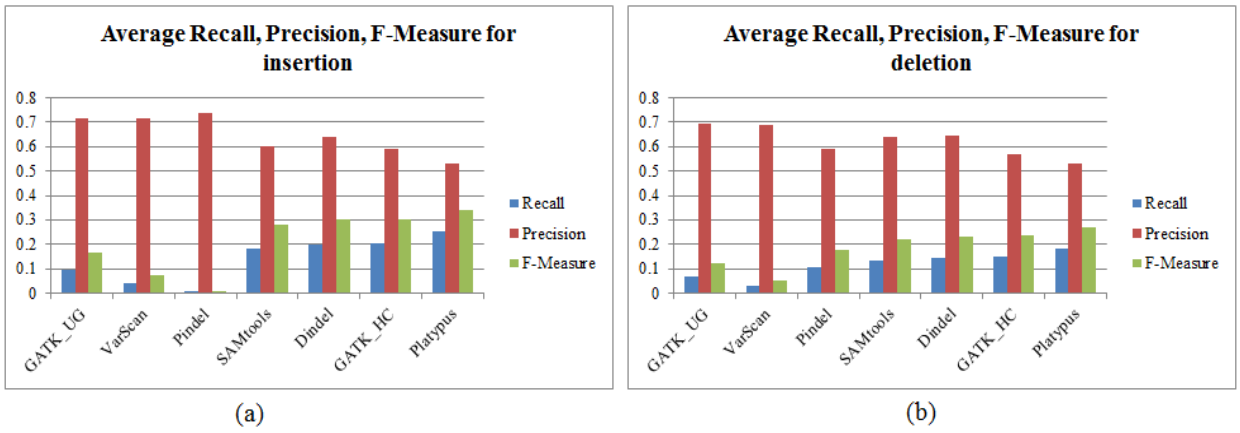


Figure A.3: Average Recall, Precision, and F-Measure of each tool for (a) insertion and (b) deletion.

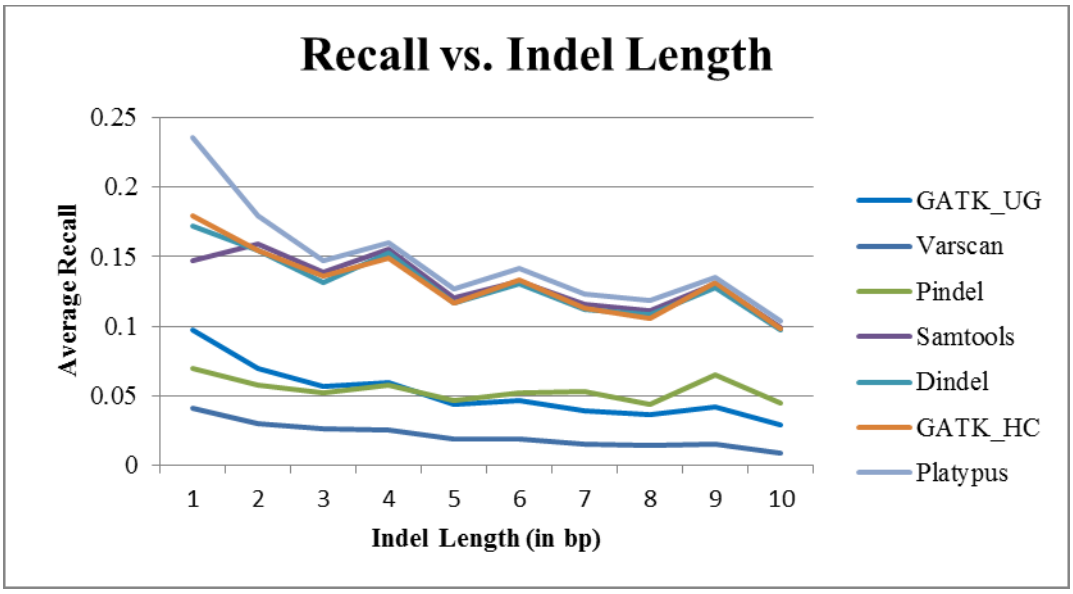


Figure A.4: Average recall for each tool for different lengths of indels.

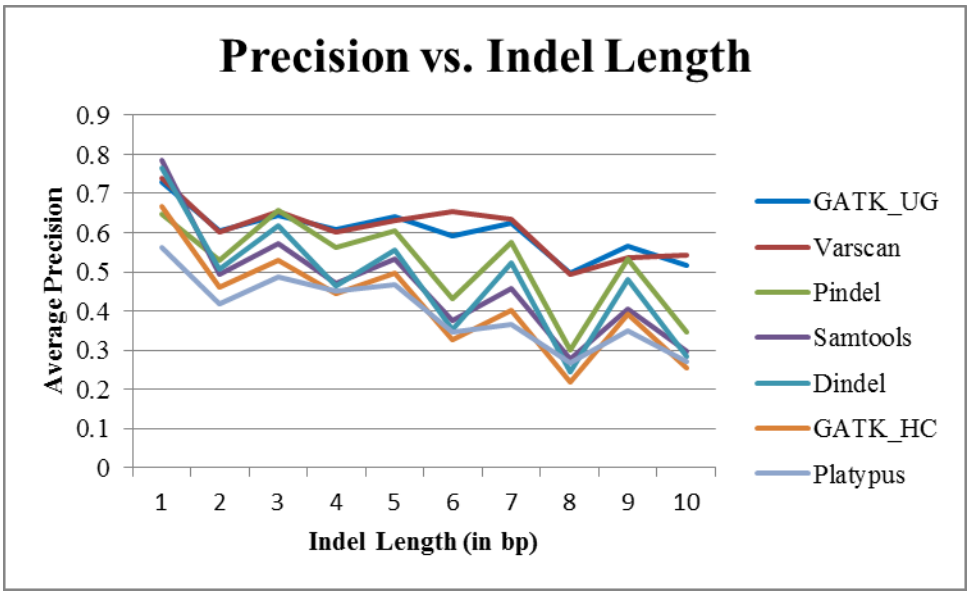


Figure A.5: Average precision for each tool for different lengths of indels.

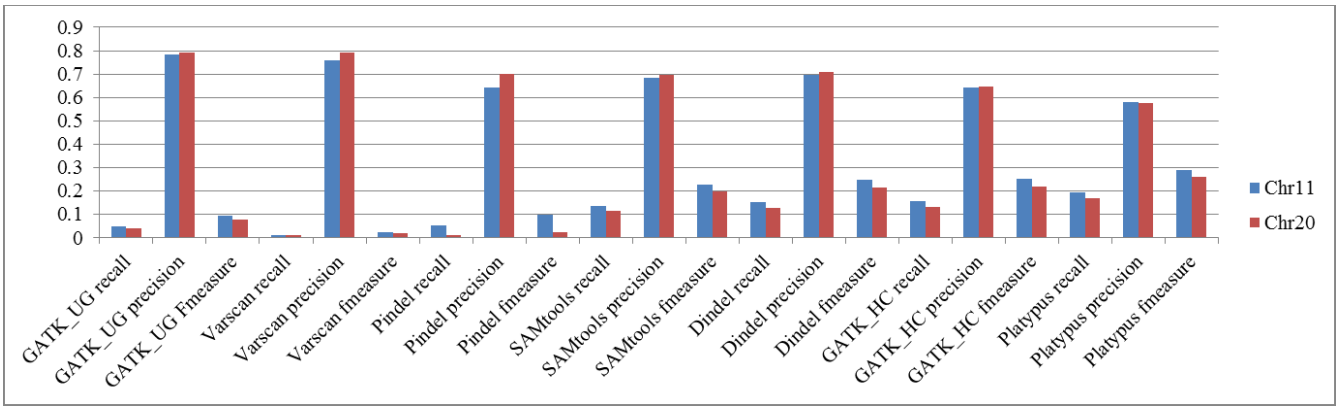


Figure A.6: Comparison between Chromosome 11 and Chromosome 20 for HG00157.

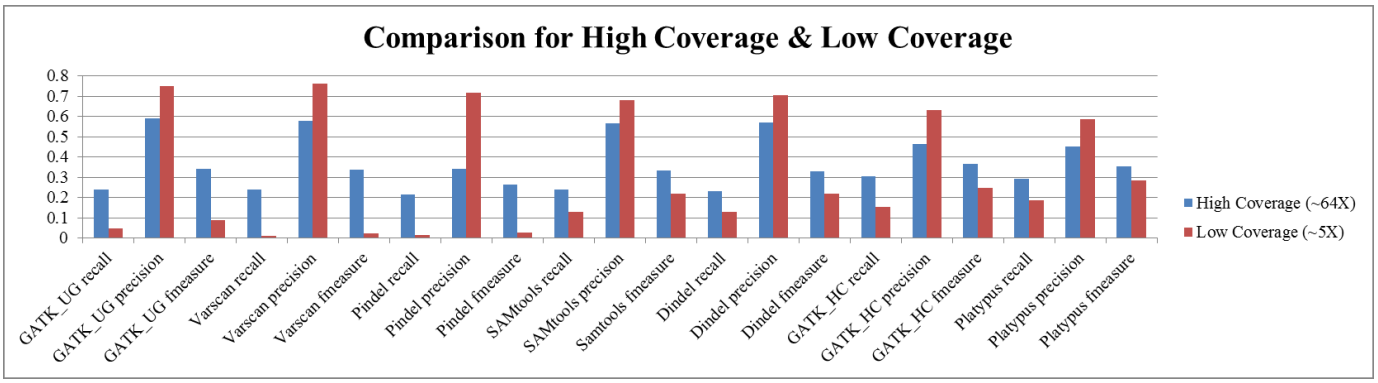


Figure A.7: Comparison between High (~64x) and Low (~5x) coverage samples for NA12878.

Appendix B

Supplementary Material of Chapter 3

Table B.1: Example of redundant indels in dbSNP.

Deletion	
Reference Sequence	TTTGAAAAAAAAAAA
rs34434241	TTTG[A/-]AAAAAAAAAA
rs34174354	TTTGA[A/-]AAAAAAAAA
Insertion	
Reference Sequence	CAACCTAATGACTCCTT
rs200449532	CAAC[CTTT/+]CTAATGACTCCTT
rs4010175	CAACCT[TTCT/+]AATGACTCCTT

Table B.2: Example of equivalent deletions.

Reference sequence	ATAATGCCTGCCTGAAC
Case 1	ATAA[TGCC/-]TGCCTGAAC
Case 2	ATAAT[GCCT/-]GCCTGAAC
Case 3	ATAATG[CCTG/-]CCTGAAC
Case 4	ATAATGC[CTGC/-]CTGAAC
Case 5	ATAATGCC[TGCC/-]TGAAC
Case 6	ATAATGCCT[GCCT/-]GAAC
Case 7	ATAATGCCTG[CCTG/-]AAC
Mismatch	ATAATGCCTGC***CTGA***AC

Table B.3: Redundant indel ratio of UPS-indel, vt normalize, BCFtools and GATK LeftAlignAndTrimVariants.

Chr Num	Total Indels	UPS-indel		vt normalize		BCFtools		GATK LeftAlignAndTrimVariants	
		Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100
1	700390	105969	15.13	85013	12.14	85013	12.14	84978	12.13
2	738109	109274	14.80	88014	11.92	88014	11.92	87981	11.92
3	619804	92047	14.85	73705	11.89	73705	11.89	73680	11.89

4	610352	90592	14.84	72321	11.85	72321	11.85	72293	11.84
5	553533	81877	14.79	65540	11.84	65540	11.84	65513	11.84
6	551502	83866	15.21	67234	12.19	67234	12.19	67209	12.19
7	502332	73054	14.54	58573	11.66	58573	11.66	58543	11.65
8	445604	64725	14.53	51822	11.63	51822	11.63	51800	11.62
9	367764	53863	14.65	43030	11.70	43030	11.70	43008	11.69
10	423800	65509	15.46	52382	12.36	52382	12.36	52356	12.35
11	415214	62539	15.06	50030	12.05	50030	12.05	50018	12.05
12	430538	64750	15.04	51511	11.96	51511	11.96	51491	11.96
13	316901	47490	14.99	37667	11.89	37667	11.89	37653	11.88
14	286524	43296	15.11	34345	11.99	34345	11.99	34327	11.98
15	263981	39720	15.05	31568	11.96	31568	11.96	31558	11.95
16	263863	38221	14.49	30190	11.44	30190	11.44	30177	11.44
17	265708	40554	15.26	32588	12.26	32588	12.26	32578	12.26
18	242978	36469	15.01	28971	11.92	28971	11.92	28964	11.92
19	216888	33062	15.24	26271	12.11	26271	12.11	26253	12.10
20	198594	30940	15.58	24884	12.53	24884	12.53	24879	12.53
21	131149	20757	15.83	16651	12.70	16651	12.70	16645	12.69
22	124352	18862	15.17	15164	12.19	15164	12.19	15159	12.19
X	213877	28103	13.14	22570	10.55	22570	10.55	22567	10.55
Y	24459	2586	10.57	2175	8.89	2175	8.89	2175	8.89

Table B.4: Redundant indel ratio of UPS-indel, vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants for COSMIC coding indel data set.

Chr Num	Total Indels	UPS-indel		vt normalize		BCFtools		GATK LeftAlignAndTrimVariants	
		Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100
1	14528	3779	26.01	3710	25.54	3710	25.54	3668	25.25
2	10621	3267	30.76	3228	30.39	3228	30.39	3180	29.94
3	9517	2657	27.92	2509	26.36	2509	26.36	2449	25.73
4	6572	1761	26.80	1642	24.98	1642	24.98	1628	24.77
5	8197	2420	29.52	2262	27.60	2262	27.60	2177	26.56
6	7493	2183	29.13	2152	28.72	2152	28.72	2134	28.48
7	7263	2114	29.11	2009	27.66	2009	27.66	1982	27.29

8	4992	1428	28.61	1410	28.25	1410	28.25	1397	27.98
9	6512	2104	32.31	2022	31.05	2022	31.05	1996	30.65
10	6284	1884	29.98	1821	28.98	1821	28.98	1790	28.49
11	8177	1986	24.29	1896	23.19	1896	23.19	1858	22.72
12	7793	2124	27.26	2096	26.90	2096	26.90	2077	26.65
13	3116	916	29.40	901	28.92	901	28.92	895	28.72
14	4340	1121	25.83	1101	25.37	1101	25.37	1095	25.23
15	4009	1077	26.86	1073	26.76	1073	26.76	1057	26.37
16	5487	1273	23.20	1250	22.78	1250	22.78	1235	22.51
17	11175	3888	34.79	3636	32.54	3636	32.54	3599	32.21
18	2132	554	25.98	548	25.70	548	25.70	541	25.38
19	9243	2259	24.44	2173	23.51	2173	23.51	2113	22.86
20	3312	869	26.24	852	25.72	852	25.72	841	25.39
21	1525	364	23.87	347	22.75	347	22.75	342	22.43
22	3221	750	23.28	733	22.76	733	22.76	728	22.60
X	6083	2020	33.21	1959	32.20	1959	32.20	1931	31.74
Y	51	19	37.25	19	37.25	19	37.25	19	37.25

Table B.5: Redundant indel ratio of UPS-indel, vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants for COSMIC noncoding indel data set.

Chr Num	Total Indels	UPS-indel		vt normalize		BCFtools		GATK LeftAlignAndTrimVariants	
		Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100	Total redundant Indels	Redundant Indel Ratio*100
1	16553	2123	12.83	2083	12.58	2083	12.58	2083	12.58
2	16617	1875	11.28	1850	11.13	1850	11.13	1850	11.13
3	12648	1444	11.42	1423	11.25	1423	11.25	1423	11.25
4	12146	1066	8.78	1052	8.66	1052	8.66	1051	8.65
5	11770	1130	9.60	1112	9.45	1112	9.45	1112	9.45
6	11169	1324	11.85	1309	11.72	1309	11.72	1309	11.72
7	11950	1602	13.41	1588	13.29	1588	13.29	1588	13.29
8	9890	1128	11.41	1115	11.27	1115	11.27	1115	11.27
9	8469	1288	15.21	1278	15.09	1278	15.09	1278	15.09
10	8872	943	10.63	922	10.39	922	10.39	922	10.39
11	9036	1312	14.52	1305	14.44	1305	14.44	1305	14.44

12	9646	1256	13.02	1239	12.84	1239	12.84	1239	12.84
13	5906	485	8.21	475	8.04	475	8.04	475	8.04
14	6244	701	11.23	693	11.10	693	11.10	693	11.10
15	5828	714	12.25	702	12.05	702	12.05	702	12.05
16	6234	870	13.96	860	13.80	860	13.80	860	13.80
17	6999	1073	15.33	1063	15.19	1063	15.19	1063	15.19
18	4904	482	9.83	475	9.69	475	9.69	475	9.69
19	6101	1322	21.67	1300	21.31	1300	21.31	1300	21.31
20	4349	481	11.06	475	10.92	475	10.92	475	10.92
21	3070	565	18.40	560	18.24	560	18.24	560	18.24
22	3109	558	17.95	546	17.56	546	17.56	546	17.56
X	6831	870	12.74	863	12.63	863	12.63	863	12.63
Y	561	102	18.18	102	18.18	102	18.18	102	18.18

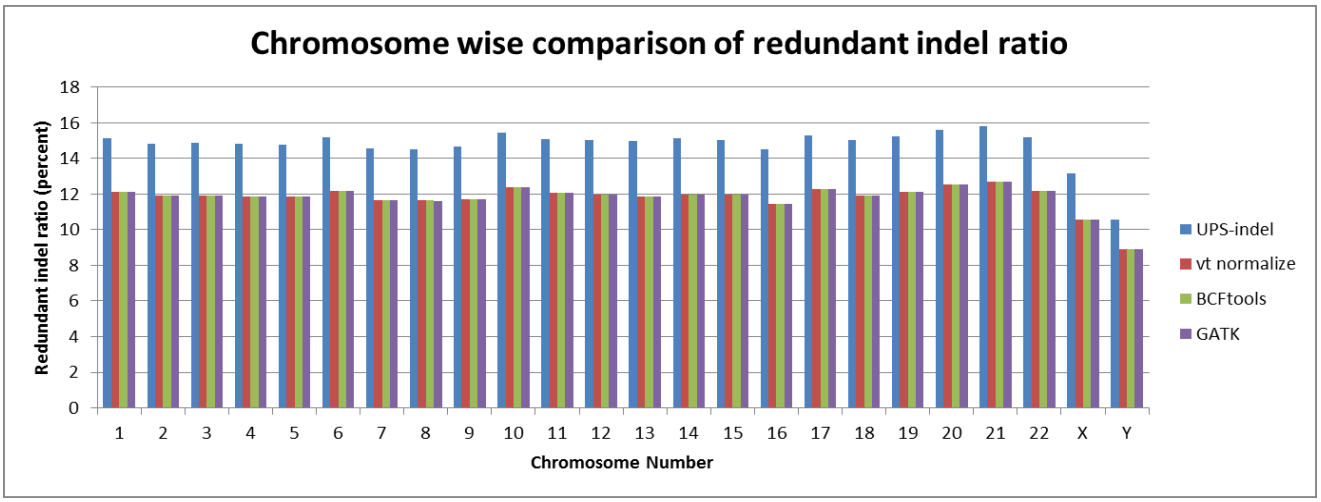


Figure B.1: Chromosome-wise comparison of redundant indel ratio among UPS-indel, vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants for the dbSNP data set.

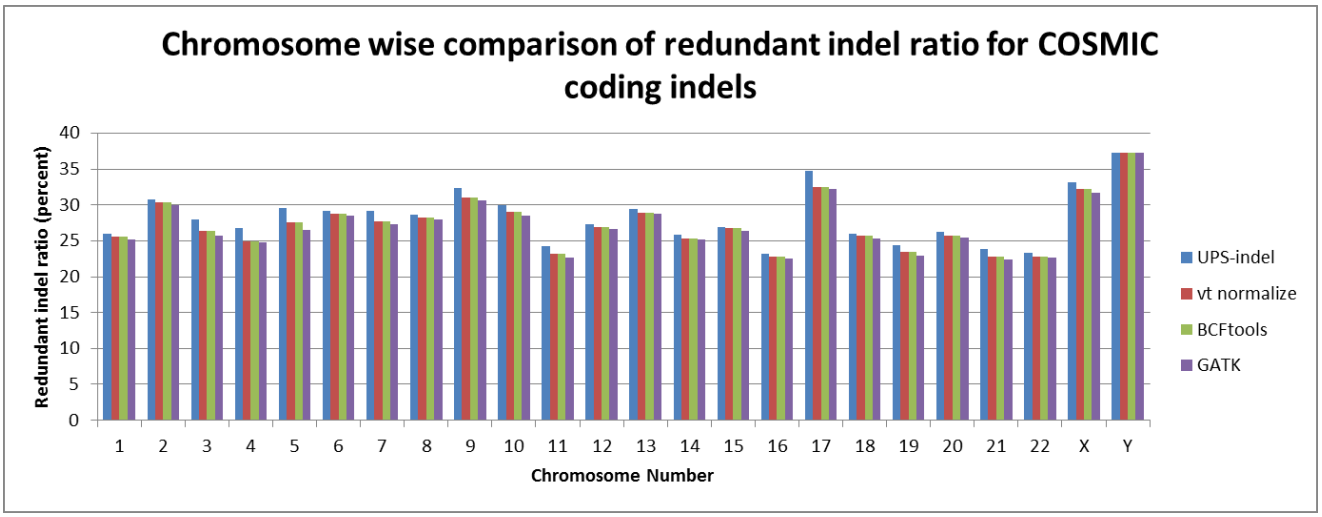


Figure B.2: Chromosome-wise comparison of redundant indel ratio among UPS-indel, vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants for COSMIC coding indels.

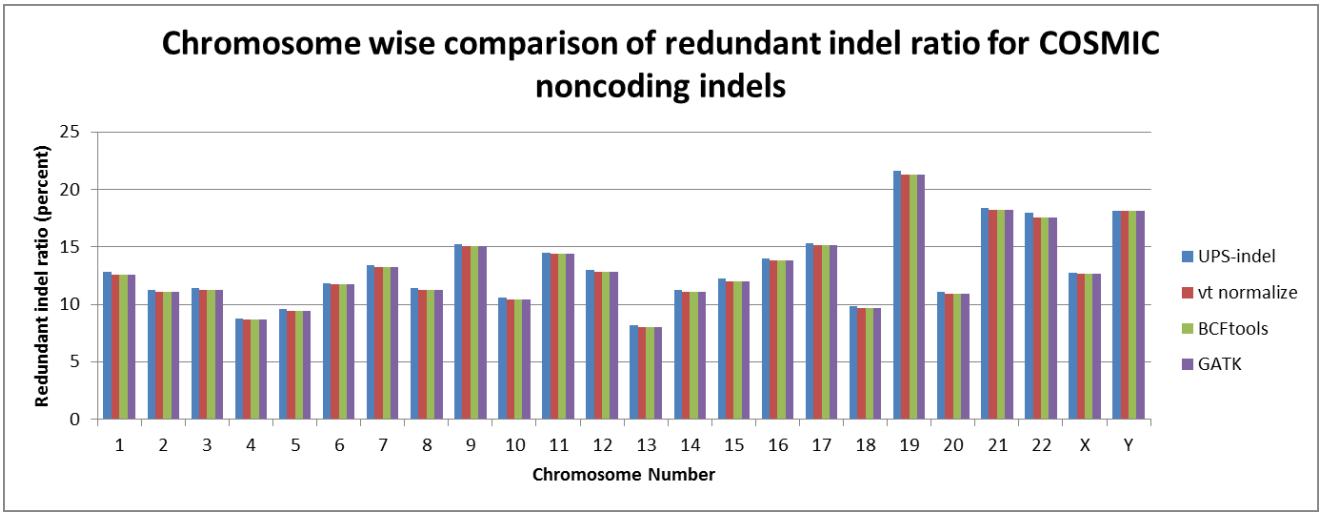


Figure B.3: Chromosome-wise comparison of redundant indel ratio among UPS-indel, vt normalize, BCFtools, and GATK LeftAlignAndTrimVariants for COSMIC noncoding indels.

Appendix C

Supplementary Material of Chapter 4

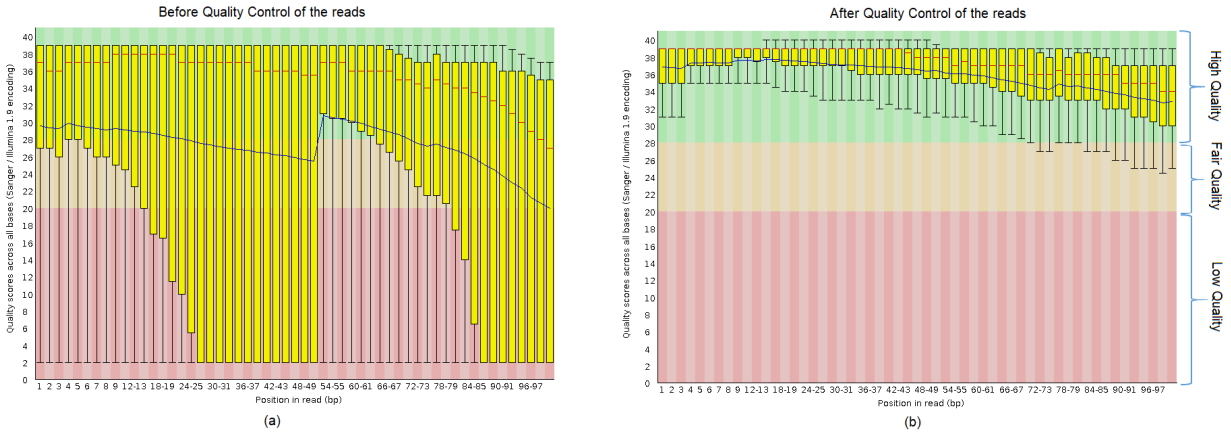


Figure C.1: Per base sequence quality (a) before and (b) after the quality control of the reads.

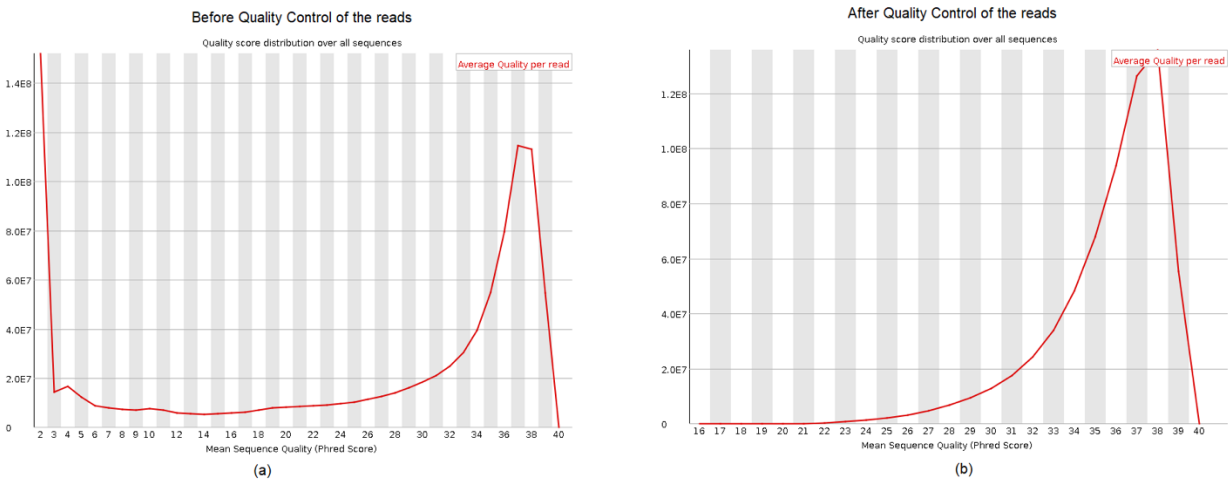


Figure C.2: Per sequence quality score (a) before and (b) after the quality control of the reads.

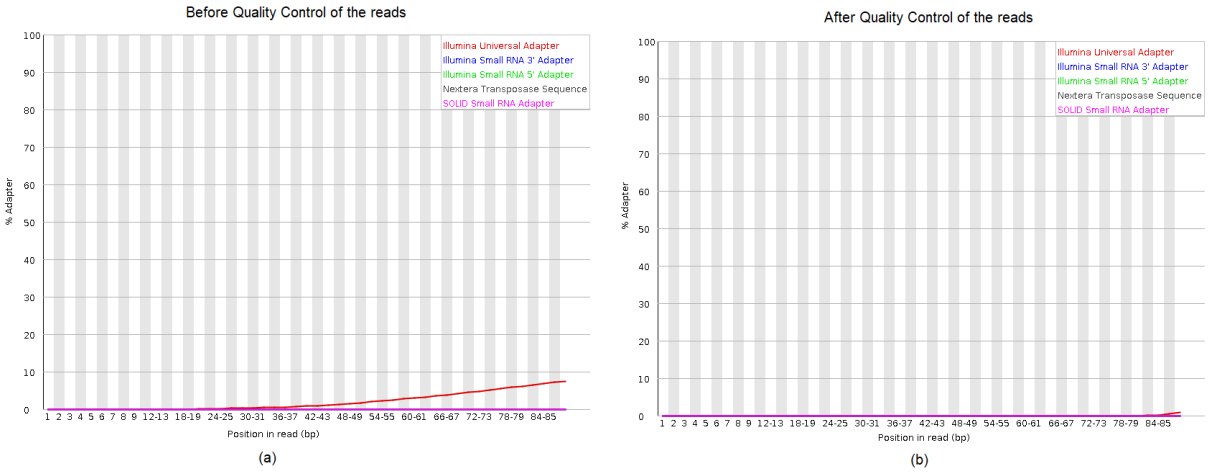


Figure C.3: Percentage of Adapter contents in the reads (a) before and (b) after the quality control.

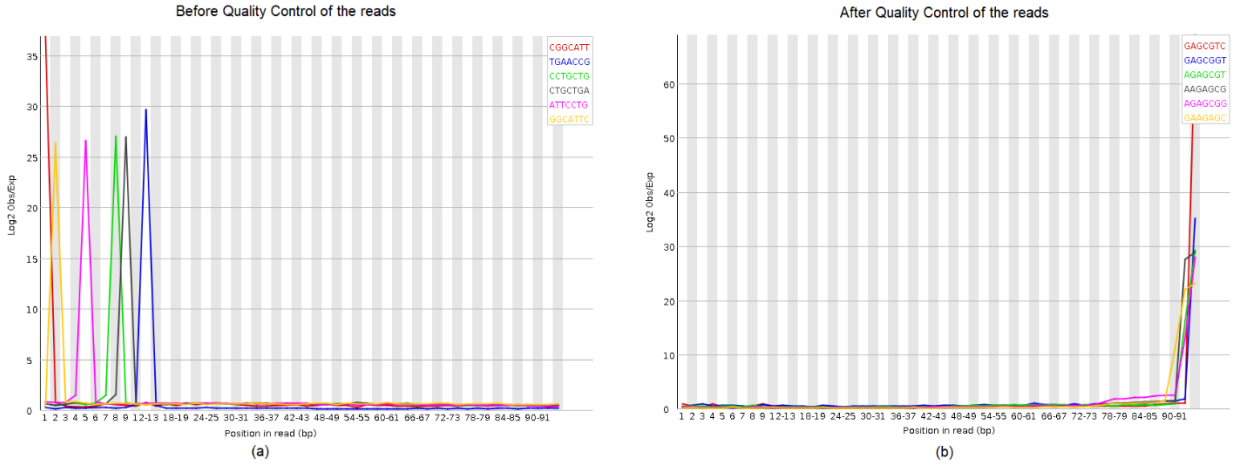


Figure C.4: Observance of different k-mers in the reads (a) before and (b) after the quality control.

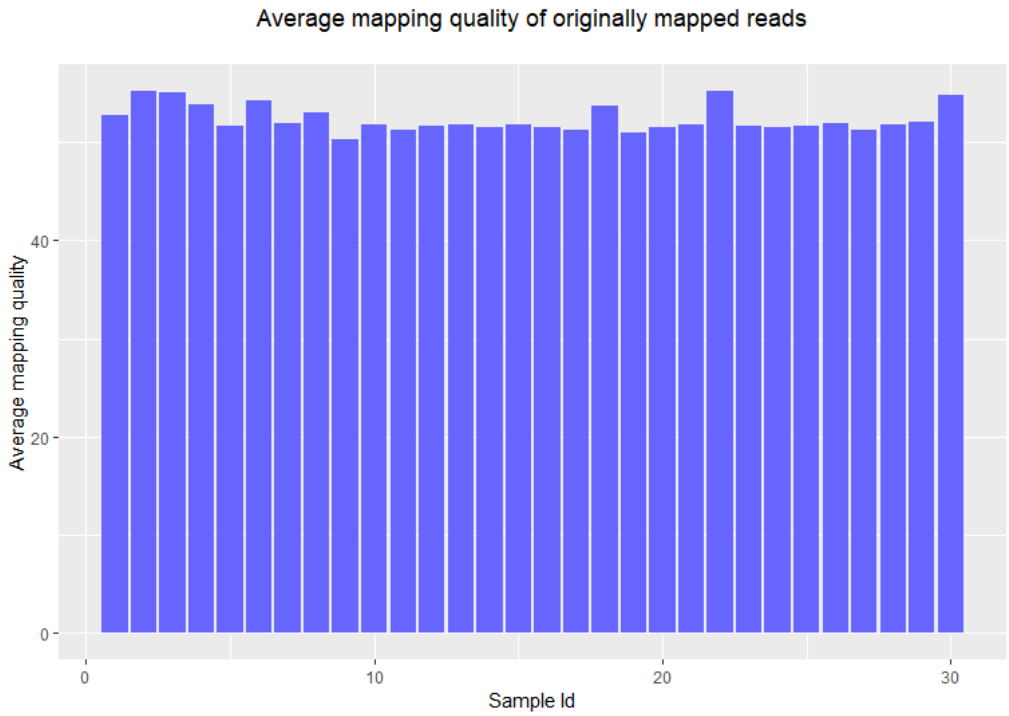


Figure C.5: Average mapping quality of the samples for the originally mapped reads.

Table C.1: List of oncogenes.

chr1	32716840	32751766	LCK	.	+
chr1	43803520	43818443	MPL	.	+
chr1	59246460	59249999	JUN	.	-
chr1	115247090	115259515	NRAS	.	-
chr1	156785432	156851642	NTRK1	.	+
chr1	179068462	179198819	ABL2	.	-
chr1	186280784	186344825	TPR	.	-
chr1	205566684	205601090	ELK4	.	-
chr2	16080686	16087129	MYCN	.	+
chr2	60677655	60782289	BCL11A	.	-
chr2	61108656	61158745	REL	.	+
chr2	113973574	114036527	PAX8	.	-
chr2	219845809	219850379	FEV	.	-
chr3	12328867	12475855	PPARG	.	+
chr3	12625100	12705725	RAF1	.	-
chr3	41236232	41301587	CTNNB1	.	+
chr3	69788586	70017488	MITF	.	+

chr3	100428175	100467810	TFG	.	+
chr3	105374305	105588396	CBLB	.	-
chr3	178865902	178957881	PIK3CA	.	+
chr3	187439165	187463515	BCL6	.	-
chr4	55524085	55606881	KIT	.	+
chr4	106067032	106200973	TET2	.	+
chr6	391739	411447	IRF4	.	+
chr6	18224099	18265054	DEK	.	-
chr6	34204650	34214008	HMGA1	.	+
chr6	37137979	37143202	PIM1	.	+
chr6	41902671	42018095	CCND3	.	-
chr6	117609463	117747018	ROS1	.	-
chr6	117881432	117923705	GOPC	.	-
chr6	135502453	135540311	MYB	.	+
chr6	167412670	167508277	FGFR1OP	.	+
chr7	2945775	3083579	CARD11	.	-
chr7	55086714	55279321	EGFR	.	+
chr7	116312444	116438440	MET	.	+
chr7	128828713	128853386	SMO	.	+
chr7	140419127	140624728	BRAF	.	-
chr8	38268656	38326352	FGFR1	.	-
chr8	57073463	57123883	PLAG1	.	-
chr8	128747680	128753680	MYC	.	+
chr9	133589333	133763062	ABL1	.	+
chr9	134000948	134110057	NUP214	.	+
chr10	43572475	43625799	RET	.	+
chr10	51565108	51590734	NCOA4	.	+
chr10	102890262	102897546	TLX1	.	+
chr10	104153867	104162286	NFKB2	.	+
chr10	123237848	123357972	FGFR2	.	-
chr11	532242	537287	HRAS	.	-
chr11	33880122	33913836	LMO2	.	-
chr11	47236016	47260791	DDB2	.	+
chr11	69455855	69469242	CCND1	.	+
chr11	95709762	96076344	MAML2	.	-
chr11	118618475	118661858	DDX6	.	-

chr12	4382938	4414516	CCND2	.	+
chr12	11802788	12048336	ETV6	.	+
chr12	25357723	25403870	KRAS	.	-
chr12	51157493	51214905	ATF1	.	+
chr12	57910371	57915520	DDIT3	.	-
chr12	66217911	66360075	HMGA2	.	+
chr12	69201956	69244466	MDM2	.	+
chr12	112856155	112947717	PTPN11	.	+
chr13	28536274	28545276	CDX2	.	-
chr14	93260576	93306308	GOLGA5	.	+
chr14	96176304	96180533	TCL1A	.	-
chr14	105235686	105262088	AKT1	.	-
chr16	31191431	31206192	FUS	.	+
chr16	79619740	79634611	MAF	.	-
chr17	5019733	5078329	USP6	.	+
chr17	37844167	37886679	ERBB2	.	+
chr17	41605212	41656988	ETV4	.	-
chr18	23596217	23671181	SS18	.	-
chr18	60790579	60987361	BCL2	.	-
chr19	40736224	40791443	AKT2	.	-
chr19	45250962	45263301	BCL3	.	+
chr19	45281126	45303891	CBLC	.	+
chr20	39314488	39317880	MAFB	.	-
chr22	23521891	23660224	BCR	.	+
chr22	29663998	29696515	EWSR1	.	+
chr22	39619364	39640756	PDGFB	.	-

Table C.2: List of tumor suppressor genes.

chr1	17345217	17380665	SDHB	.	-
chr1	51426417	51440305	CDKN2C	.	+
chr1	204485507	204527248	MDM4	.	+
chr1	241660903	241683061	FH	.	-
chr2	47630108	47890285	MSH2	.	+
chr2	208394461	208468155	CREB1	.	+
chr2	209100951	209130798	IDH1	.	-

chr3	10182692	10193904	VHL	.	+
chr3	37034823	37092409	MLH1	.	+
chr3	71001968	71633144	FOXP1	.	-
chr4	123372625	123377880	IL2	.	-
chr4	153241696	153457215	FBXW7	.	-
chr5	112043195	112181936	APC	.	+
chr5	170814120	170838141	NPM1	.	+
chr6	138188325	138204449	TNFAIP3	.	+
chr7	92234235	92465908	CDK6	.	-
chr8	30891317	31031285	WRN	.	+
chr8	118806729	119124092	EXT1	.	-
chr9	4984390	5128183	JAK2	.	+
chr9	93564069	93660831	SYK	.	+
chr9	102584137	102629173	NR4A3	.	+
chr9	135766735	135822261	TSC1	.	-
chr9	139388885	139440238	NOTCH1	.	-
chr10	88516358	88692595	BMPR1A	.	+
chr10	89622870	89731687	PTEN	.	+
chr10	104263744	104393292	SUFU	.	+
chr11	3022152	3078843	CARS	.	-
chr11	3692313	3819022	NUP98	.	-
chr11	32409321	32457176	WT1	.	-
chr11	44117099	44266979	EXT2	.	+
chr11	64570982	64578766	MEN1	.	-
chr11	108093211	108239829	ATM	.	+
chr11	111957497	111990736	SDHD	.	+
chr11	120207623	120360645	ARHGEF12	.	+
chr13	28577411	28674729	FLT3	.	-
chr13	32889611	32974403	BRCA2	.	+
chr13	48877862	49173572	RB1	.	+
chr14	99635624	99737861	BCL11B	.	-
chr15	74287014	74340153	PML	.	+
chr15	91260558	91359395	BLM	.	+
chr16	2097466	2139488	TSC2	.	+
chr16	3775055	3930727	CREBBP	.	-
chr16	11348262	11350057	SOCS1	.	-

chr16	23614481	23652631	PALB2	.	-
chr16	50775961	50835846	CYLD	.	+
chr16	64977656	65160015	CDH11	.	-
chr16	68771128	68869451	CDH1	.	+
chr16	88941266	89043612	CBFA2T3	.	-
chr17	7565097	7590868	TP53	.	-
chr17	11924141	12047147	MAP2K4	.	+
chr17	29421945	29709134	NF1	.	+
chr17	30264037	30328064	SUZ12	.	+
chr17	41196312	41322262	BRCA1	.	-
chr17	62495734	62504317	DDX5	.	-
chr19	1177557	1228434	STK11	.	+
chr19	1609289	1652604	TCF3	.	-
chr19	11071501	11190102	SMARCA4	.	+
chr19	33790840	33793470	CEBPA	.	-
chr21	36160098	37376965	RUNX1	.	-
chr22	24129118	24180195	SMARCB1	.	+
chr22	29083731	29138410	CHEK2	.	-
chr22	29999545	30094587	NF2	.	+
chrX	132669773	133119922	GPC3	.	-

Table C.3: TCGA Sample Barcode and alignment filename for the patients.

Sample Id	**TCGA_Sample_Barcode**	**Filename**
1	TCGA-A2-A04P-01A-31D-A128-09	TCGA-A2-A04P-01A-31D-A128-09_IlluminaGA-DNASeq_whole.bam
2	TCGA-A2-A04T-01A-21D-A128-09	TCGA-A2-A04T-01A-21D-A128-09_IlluminaGA-DNASeq_whole.bam
3	TCGA-A2-A0D0-01A-11D-A128-09	TCGA-A2-A0D0-01A-11D-A128-09_IlluminaGA-DNASeq_whole.bam
4	TCGA-A2-A0D2-01A-21D-A128-09	TCGA-A2-A0D2-01A-21D-A128-09_IlluminaGA-DNASeq_whole.bam
5	TCGA-A2-A0EU-01A-22D-A060_130807	TCGA-A2-A0EU-01A-22D-A060_130807_SN590_0235_AC29RAACXX_s_1_rg.sorted.bam
6	TCGA-A7-A0CE-01A-11D-A12L-09	TCGA-A7-A0CE-01A-11D-A12L-09_IlluminaGA-DNASeq_whole_1.bam

7	TCGA-A7-A0D9-01A-31D-A060_130807	TCGA-A7-A0D9-01A-31D-A060_130807_SN590_0235_AC29RAACXX_s_5_rg.sorted.bam
8	TCGA-AO-A0J4-01A-11D-A128-09	TCGA-AO-A0J4-01A-11D-A128-09_IlluminaGA-DNASeq_whole.bam
9	TCGA-AO-A0J6-01A-11D-A128-09	TCGA-AO-A0J6-01A-11D-A128-09_IlluminaGA-DNASeq_whole.bam
10	TCGA-AO-A0JF-01A-11D-A060_130719	TCGA-AO-A0JF-01A-11D-A060_130719_SN1120_0270_AC2CVRACXX_s_1_rg.sorted.bam
11	TCGA-AO-A0JJ-01A-11D-A060_130725	TCGA-AO-A0JJ-01A-11D-A060_130725_SN590_0233_AD2B3HACXX_s_3_rg.sorted.bam
12	TCGA-AO-A0JL-01A-11D-A060_130725	TCGA-AO-A0JL-01A-11D-A060_130725_SN590_0234_BC29HAACXX_s_1_rg.sorted.bam
13	TCGA-AR-A0TU-01A-31D-A106_130719	TCGA-AR-A0TU-01A-31D-A106_130719_SN1120_0270_AC2CVRACXX_s_5_rg.sorted.bam
14	TCGA-B6-A0RE-01A-11D-A060_130725	TCGA-B6-A0RE-01A-11D-A060_130725_SN590_0234_BC29HAACXX_s_7_rg.sorted.bam
15	TCGA-B6-A0RG-01A-11D-A060_130807	TCGA-B6-A0RG-01A-11D-A060_130807_SN590_0235_AC29RAACXX_s_3_rg.sorted.bam
16	TCGA-B6-A0RI-01A-11D-A060_130807	TCGA-B6-A0RI-01A-11D-A060_130807_SN590_0236_BC291KACXX_s_3_rg.sorted.bam
17	TCGA-B6-A0X4-01A-11D-A106_130719	TCGA-B6-A0X4-01A-11D-A106_130719_SN1120_0270_AC2CVRACXX_s_7_rg.sorted.bam
18	TCGA-BH-A0B3-01A-11D-A128-09	TCGA-BH-A0B3-01A-11D-A128-09_IlluminaGA-DNASeq_whole.bam
19	TCGA-BH-A0B9-01A-11D-A128-09	TCGA-BH-A0B9-01A-11D-A128-09_IlluminaGA-DNASeq_whole.bam
20	TCGA-BH-A0BM-01A-11D-A060_130725	TCGA-BH-A0BM-01A-11D-A060_130725_SN590_0234_BC29HAACXX_s_3_rg.sorted.bam
21	TCGA-BH-A0DK-01A-21D-A060_130807	TCGA-BH-A0DK-01A-21D-A060_130807_SN590_0236_BC291KACXX_s_5_rg.sorted.bam
22	TCGA-BH-A0E0-01A-11D-A128-09	TCGA-BH-A0E0-01A-11D-A128-09_IlluminaGA-DNASeq_whole.bam
23	TCGA-BH-A0GY-01A-11D-A060_130807	TCGA-BH-A0GY-01A-11D-A060_130807_SN590_0235_AC29RAACXX_s_7_rg.sorted.bam

24	TCGA-BH-A0H6-01A-21D-A060_130725	TCGA-BH-A0H6-01A-21D-A060_130725_SN590_0233_AD2B3HACXX_s_1_rg.sorted.bam
25	TCGA-BH-A0H7-01A-13D-A060_130725	TCGA-BH-A0H7-01A-13D-A060_130725_SN590_0233_AD2B3HACXX_s_7_rg.sorted.bam
26	TCGA-BH-A0HB-01A-11D-A060_130725	TCGA-BH-A0HB-01A-11D-A060_130725_SN590_0234_BC29HAACXX_s_5_rg.sorted.bam
27	TCGA-BH-A0HK-01A-11D-A060_130807	TCGA-BH-A0HK-01A-11D-A060_130807_SN590_0236_BC291KACXX_s_1_rg.sorted.bam
28	TCGA-BH-A0HX-01A-21D-A060_130807	TCGA-BH-A0HX-01A-21D-A060_130807_SN590_0236_BC291KACXX_s_7_rg.sorted.bam
29	TCGA-BH-A0W5-01A-11D-A106_130719	TCGA-BH-A0W5-01A-11D-A106_130719_SN1120_0270_AC2CVRACXX_s_3_rg.sorted.bam
30	TCGA-BH-A0WA-01A-11D-A128-09	TCGA-BH-A0WA-01A-11D-A128-09_IlluminaGA-DNASeq_whole.bam

Appendix D

Supplementary Material of Chapter 5

Table D.1: Features used by SomaticHunter.

NORMAL_BAM_DEPTH
NORMAL_BAM_REF_MQ
NORMAL_BAM_ALT_MQ
NORMAL_BAM_Z_Ranksums_MQ
NORMAL_BAM_REF_BQ
NORMAL_BAM_ALT_BQ
NORMAL_BAM_Z_Ranksums_BQ
NORMAL_BAM_REF_NM
NORMAL_BAM_ALT_NM
NORMAL_BAM_REF_Concordant
NORMAL_BAM_REF_Discordant
NORMAL_BAM_ALT_Concordant
NORMAL_BAM_ALT_Discordant
NORMAL_BAM_Concordance_FET
NORMAL_BAM_StrandBias_FET
NORMAL_BAM_Z_Ranksums_EndPos
NORMAL_BAM_REF_Clipped_Reads
NORMAL_BAM_ALT_Clipped_Reads
NORMAL_BAM_Clipping_FET
NORMAL_BAM_MQ0
NORMAL_BAM_Poor_Reads
NORMAL_BAM_REF_InDel_3bp
NORMAL_BAM_REF_InDel_2bp
NORMAL_BAM_REF_InDel_1bp
NORMAL_BAM_ALT_InDel_3bp
NORMAL_BAM_ALT_InDel_2bp
NORMAL_BAM_ALT_InDel_1bp
MaxHomopolymer_Length
SiteHomopolymer_Length
TUMOR_BAM_DEPTH
TUMOR_BAM_REF_MQ
TUMOR_BAM_ALT_MQ

TUMOR_BAM_Z_Ranksums_MQ
TUMOR_BAM_REF_BQ
TUMOR_BAM_ALT_BQ
TUMOR_BAM_Z_Ranksums_BQ
TUMOR_BAM_REF_NM
TUMOR_BAM_ALT_NM
TUMOR_BAM_REF_Concordant
TUMOR_BAM_REF_Discordant
TUMOR_BAM_ALT_Concordant
TUMOR_BAM_ALT_Discordant
TUMOR_BAM_Concordance_FET
TUMOR_BAM_StrandBias_FET
TUMOR_BAM_Z_Ranksums_EndPos
TUMOR_BAM_REF_Clipped_Reads
TUMOR_BAM_ALT_Clipped_Reads
TUMOR_BAM_Clipping_FET
TUMOR_BAM_MQ0
TUMOR_BAM_Poor_Reads
TUMOR_BAM_REF_InDel_3bp
TUMOR_BAM_REF_InDel_2bp
TUMOR_BAM_REF_InDel_1bp
TUMOR_BAM_ALT_InDel_3bp
TUMOR_BAM_ALT_InDel_2bp
TUMOR_BAM_ALT_InDel_1bp