Fall 12-16-2016

ChimeRScope: a novel alignment-free algorithm for fusion gene prediction using paired-end short reads

You Li
University of Nebraska Medical Center

ChimeRScope: a novel alignment-free algorithm for fusion gene prediction using paired-end short reads

By

You Li

A DISSERTATION

Presented to the Faculty of

The Graduate College in the University of Nebraska

In Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Biomedical Informatics Graduate Program

Department of Genetics, Cell biology and Anatomy

Under the Supervision of Professor Chittibabu (Babu) Guda

University of Nebraska Medical Center

Omaha, Nebraska

September, 2016

ChimeRScope: a novel alignment-free algorithm

for fusion gene prediction using paired-end short reads

You Li, Ph.D.

University of Nebraska Medical Center, 2016

Advisor: Chittibabu (Babu) Guda, Ph.D.

Fusion genes are those that result from the fusion of two or more genes, and they are typically generated due to the perturbations in the genome structure in cancer cells. In turn, fusion genes can contribute to tumor formation and progression by promoting the expression of an oncogene, deregulation of a tumor-suppressor, or producing much more active abnormal proteins. More importantly, oncogenic fusion genes are specifically expressed in the tumor cells, which provide enormous diagnostic and therapeutic advantages for cancer treatment. With the development of next-generation sequencing (NGS) technology, RNA-Seq becomes increasingly popular for transcriptomic study because of its high sensitivity and the capability of detecting novel transcripts including fusion genes. To date, many fusion gene detection tools have been developed, most of which attempt to find reliable alignment evidence for chimeric transcripts from RNA-Seq data. It is well accepted that the alignment quality of sequencing reads against the reference genome is often limited when significant differences in the genomes exist, which is the case with cancer genomes that contain many genomic perturbations and structural variations. Hence, regions where fusion genes occur in the cancer genome tend to be largely different from those in the reference genome, which prevents the alignment-based fusion gene detection methods from achieving good accuracies.

We developed a tool called ChimeRScope. ChimeRScope, being an alignment-free method, bypasses the sequence alignment step by assessing the gene fingerprint profiles (in the form of *k*-

mers) from RNA-Seq paired-end reads for fusion gene prediction (Chapter Two). We also optimized the data structure and ChimeRScope algorithms, in order to overcome the common limitations (memory-utilization, low accuracies) that are commonly seen in alignment-free methods (Chapter Two). Results on simulated datasets, previously studied cancer RNA-Seq datasets, and experimental validations on in-house datasets have shown that ChimeRScope consistently performed better than other popular alignment-based methods irrespective of the read length and depth of sequencing coverage (Chapter Three). ChimeRScope also generates graphical outputs for illustrations of the fusion patterns. Lastly, we also developed downloadable software for ChimeRScope and implemented an online data analysis server using the Galaxy platform (Chapter Four). ChimeRScope is available at https://github.com/ChimeRScope/ChimeRScope/.

Acknowledgement

I would like to thank all the people who have supported me during my Ph.D. study and contributed to the work presented in this dissertation.

Special thanks to my enthusiastic supervisor and mentor, Dr. Babu Guda. This project will not be successful without his careful instructions, insightful thoughts, continuous support and encouragement.

Thanks to all my supervisory committee members, Dr. Hesham Ali, Dr. Michael A. Hollingsworth, Dr. James D. Eudy, and Dr. Fang Yu for all the valuable suggestions and guidance on this project.

I would also like to thank Dr. Javeed Iqbal, Dr. Neetha N Vellichirammal and Tayla B. Heavican for the contributions on the experimental validation part of this project.

Special mention goes to Qian Zhang, Adam Cornish, Sanjit Pandey who provided direct support to this project and the ChimeRScope software. Thanks to all other colleagues/friends from Guda Lab for the inspiring discussions. Thanks to other friends in Omaha for the good time we had together.

To my dear wife, Lu Yang (Lucy), who provided continuous support and made a better me because of her great personalities and positive attitudes to life, and to our beloved son, Timothy Jiaxi Li, I will love them both forever. To my parents, 2015 and 2016 were the toughest time for our family. I believe we can overcome those difficult times together and I will always by your side. To my parents-in-law, thank you all for sacrificing your own free time to take care of Timothy. My family members are the most important people in my life and I dedicated this dissertation to them.

Table of Contents

Chapter 1: INTRODUCTION TO CARCINOGENESIS AND FUSION GENES IN CANCER

1. Introduction to cancer and the genetic nature of cancer

Cancer is a group of genetic diseases that can cause severe health consequences. It is one of the leading causes (~13%) of all deaths worldwide. According to World Health Organization (WHO) GLOBOCAN 2012 estimation, there were 14.1 million new cancer cases and 8.2 million cancer deaths each year (Table 1-1). According to the WHO estimation, there will be an increase of 70% new cancer cases over the next 20 years.

Table 1-1. Cancer fact sheet in 2012 worldwide [1]. This table shows the statistics of new cancer cases, deaths, and patients that are alive with cancer within 5 years of diagnosis, for men, women and combined in 2012 worldwide.

Estimated numbers (thousands)	Men			Women			Both sexes		
	Cases	Deaths	5-year prev.*	Cases	Deaths	5-year prev.	Cases	Deaths	5-year prev.
World	7410	4653	15296	6658	3548	17159	14068	8202	32455
More developed regions	3227	1592	8550	2827	1287	8274	6054	2878	16823
Less developed regions	4184	3062	6747	3831	2261	8885	8014	5323	15632

* 5-year prevalence: The number of people who live with cancer within 5 years of diagnosis.

It is well accepted that mutations on cancer-susceptible genes are the most common causes of cancer [2]. Those genetic variations can be acquired from exposures to mutagens, or inherited from parents. The clinical manifestations of cancer result from the uncontrolled growth of cells, which can also be metastasized into other parts of the body. Based on the origins of the cells, cancer can be classified into different types such as carcinoma, sarcoma, lymphoma/leukemia, germ cell tumor, blastoma, etc. More than 100 types of cancers have been identified based on the

histological type, tissue type, cell type, or other biomarkers. The aggressive behaviors of cancer cells often occur as a consequence of the malfunctioning of cancer-susceptibility genes. These genes can be further divided into three types: (i) genes that control the cell growth and cell death; malfunctioning of these genes can lead to uncontrolled cell growth; (ii) genes that control the DNA repair mechanisms, which in normal state prevent the cells from accumulating deleterious mutations; (iii) genes that control the cell-cell interactions, which when impaired can lead to the neoplasm [3].

The biological system is a complex system. A single alteration on one cancer-susceptibility gene may not induce carcinogenesis in cells; however, cells with certain gene mutations in driver genes are more prone to chromosomal instability (one of the hallmarks of many cancers), thereby increasing the risk of acquiring more deleterious alterations. Once the cell accumulates a sufficient set of deleterious mutations, the tumorigenesis will be initiated. Therefore, it is crucial for researchers to identify those oncogenic driver events in order to prevent or treat cancer.

2. Oncogenic variations

One simple fact of biological systems is that genetic variation exists in all individuals. These variations can be classified into different types that include (i) **SNP**, or **S**ingle **N**ucleotide **P**olymorphism; (ii) **INDEL**, a genomic sequence (often less than 50 base pairs (bp)) that were **IN**serted into or **DEL**eted from the genome; (iii) **CNV** (**C**opy **N**umber **V**ariation) is the deletion or duplication of a large chunk (> 50 bp) of the genomic sequence; (iv) Other genomic rearrangements such as translocation, inversion or a combination of these two; A SNP variation is considered as synonymous if this variation does not impair the function of the protein due to the degeneracy of the genetic code, or non-deleterious if the SNP does not occur in the functional elements of the genome. Even if deleterious mutations occur, multiple DNA damage response processes will try to maintain genome stability via processes such as DNA repair, cell cycle checkpoints, and controlled cell death (apoptosis).

When oncogenic (meaning "causing the development of a tumor or tumors", often involving cancer-susceptibility genes) driver mutations are induced from the exposure to carcinogens, or are inherited from parents, it will make the genome less stable. In turn, an unstable genome is more likely to introduce more deleterious mutations with selective advantages in the microenvironment by either increasing the survival of the cell (e.g., bypassing the apoptosis) or reproduction rate of the cell (e.g., uncontrolled cell division). Therefore, it is important to identify those oncogenic driver mutations before the cancer progresses to advanced stages. Studies on different cancers over the last several decades revealed many oncogenic driver events with huge therapeutic potentials. For instance, a non-synonymous SNP on BRAF (B-Raf proto-oncogene, serine/threonine kinase) at position 600 that changes the amino acid from Valine to Glutamic Acid (V600E) has been reported in many different types of human cancers [4]. BRAF is a protein kinase that regulates the MAP kinase/ERKs signaling pathway (MAPK pathway), which affects cell division, differentiation and secretion [5]. BRAF V600E leads to hyper-activation of MAPK pathway, which will eventually lead to unregulated cell proliferation and survival [4]. Patients with melanoma treated with drugs that target BRAF with V600E mutation have shown improved survival rate [6].

3. Fusion genes in cancer

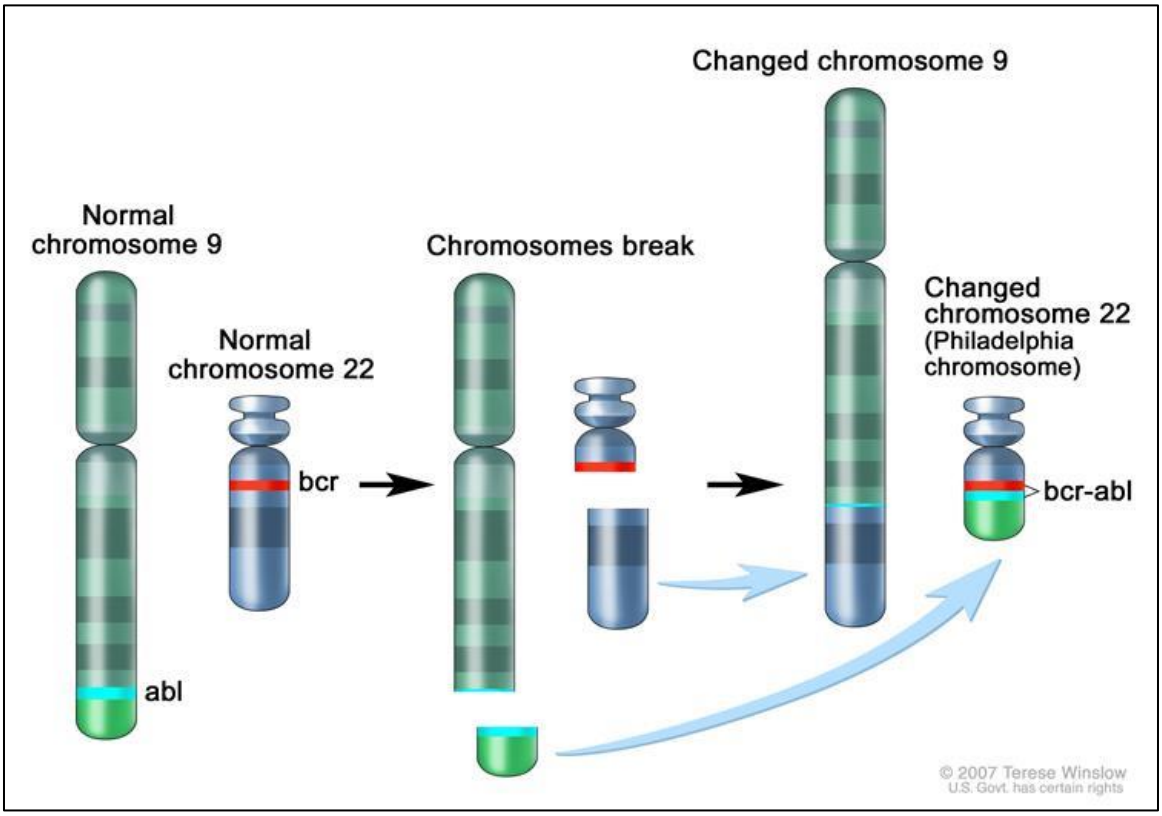
Fusion genes, also known as chimeric transcripts, are one class of abnormal transcripts that are formed by the fusion of two different genes. Fusion genes have gained increasing attention because of their significance in diagnosis and prognosis. However, identification of the fusion genes remains to be a big challenge due to the complexities of the cancer genomes. In this study, we focused on identifying fusion genes in cancer, particularly using an alignment-free approach for fusion gene prediction using NGS-based transcriptome datasets.

3.1. Formation of a fusion gene

Fusion genes results from the fusion of two or more genes, which typically happen during perturbations in the genomic level or during the gene transcription. At the genomic level, a fusion gene event can happen because of chromosomal rearrangements such as translocation, interstitial deletion, or inversion. These types of fusion genes can be generated between genes from the same chromosome or from different chromosomes. For instance, the Philadelphia chromosome [7] is a translocation event between chromosome 9 and chromosome 22 and it has been mainly discovered in Chronic Myelogenous Leukemia (CML) cells. This translocation event joins ABL1 gene on Chromosome 9 with BCR gene on Chromosome 22 and creates an abnormal fusion protein BCR-ABL (Figure 1-1). ABL1 encodes a protein kinase that regulates cell cycle and cellular differentiation. The activity of the wild type ABL1 is negatively regulated by its own SH3 domain. The BCR-ABL fusion often results in the deletion of SH3 region, thereby keeping the expression level of ABL1 in a constantly active state, which leads to uncontrolled proliferation of the cell.

Fusion genes can also be generated at transcription stage due to unterminated transcriptions (e.g., stop-loss mutations) or abnormal splicing events (e.g., mutations near splicing site). This type of fusion genes is more likely to happen between genes located on the same chromosome.

Figure 1-1. Philadelphia chromosome and BCR-ABL fusion [8]. A translocation event between Chr9 and Chr22 creates an abnormal fusion product BCR-ABL near the junction site.



3.2. Role of Oncogenic Fusion Genes in Carcinogenesis

Cancer-susceptibility genes (Chapter 1, section 1) are responsible for carcinogenesis. A fusion gene involving one or more of these genes can become oncogenic when it functions as one of the following. (a) Promote the expression of a proto-oncogene; (b) Deregulate a tumor suppressor gene; (c) Modify the original structure/function of a protein or form a novel abnormal protein that stimulates tumorigenesis. Specifically, oncogenic transcription factors such as ERG, ETV1, and ETV4 are often fused with androgen-regulated promoters in 50-70% of prostate cancer patients [9]. This category of fusion genes (category a) up-regulate the expression of oncogenic transcription factors, thereby promoting tumorigenesis [9]. Another transcription factor

from the same ETS gene family, ETV6, functions as a tumor suppressor gene and is required for hematopoiesis and maintenance of vascular network development [10]. In pre-B acute lymphoblastic leukemia, fusions between ETV6 and BZA2A result in the loss of function of ETV6 [11] (category b fusion event). One example for category (c) fusion event is FGFR3-TACC3 fusion in glioblastoma. This fusion gene can promote cell proliferation and tumor progression and it can escape the miRNA regulation because of the deletion of FGFR3 3'-UTR in the fusion gene [12].

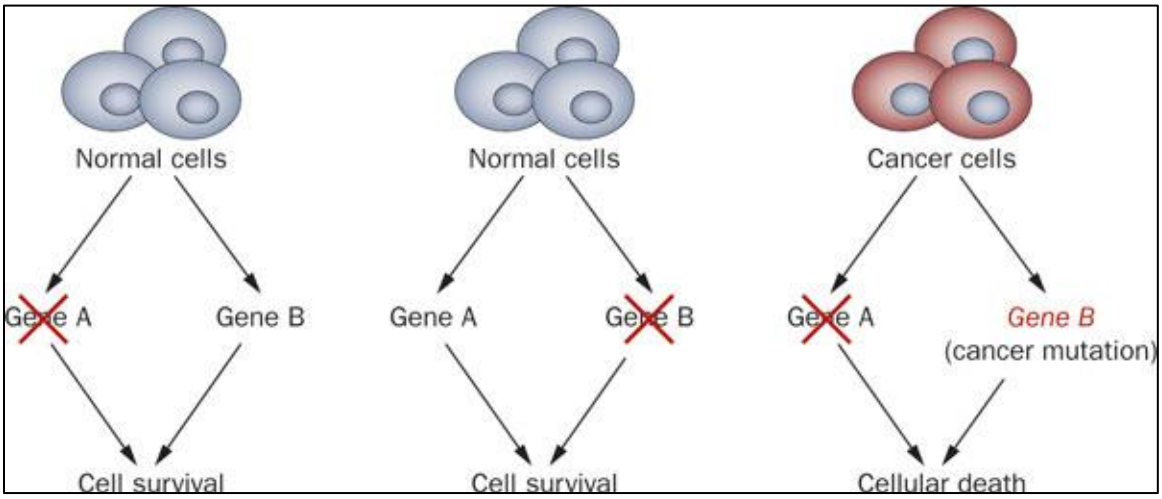
3.3. Clinical Significance of Oncogenic Fusion Genes

Fusion events at the genome level are naturally irreversible. Hence, the presence of an oncogenic fusion gene in tumor cells is traceable at the genomic or transcriptome level. More importantly, these evidences should not be observed in normal cells. Therefore, the cancer-specific fusion events can be used as special biomarkers for diagnostic and prognostic purposes and as ideal drug targets for targeted-therapy.

In the previous subsection, we described three different types of oncogenic fusion genes. Most of the FDA-approved drugs for cancer treatment today target category (a) fusion genes, a class of fusion genes that always involves a proto-oncogene (often a protein kinase). Protein kinase is a class of enzymes that play a major role in various processes such as regulating protein function/structure, enzyme activity, cell-cell signaling, and cell cycle. Category (a) fusion genes keep the expression level of a gene in a constant active state, resulting in aggressive cell proliferation. In such a scenario, we expect to see high expression level of the fusion gene, exclusively in cancer cells. Kinase inhibitors can be designed to target the active proto-oncogene product, in order to block the activity of the kinase protein without having too many side effects on normal cells. Imatinib, a drug that targets BCR-ABL fusion protein (category (a) fusion gene product) in CML, has recorded a global sale of $4.7b in 2012 [13].

Fusion genes involving tumor suppressors are category (b) fusion genes and their effect on tumorigenesis works differently from category (a) fusions. Tumor suppressors generally have one or more functions such as repairing DNA damage, regulating cell cycle, or promoting apoptosis. Hence, when tumor suppressors are not expressed or down-regulated, the cell can eventually progress to cause cancer. The category (b) oncogenic fusion genes will have similar regulatory effect on the tumor suppressors by either negatively regulating their expression level, or even create an abnormal fusion product with no or limited function, due to loss of certain functional domains. Therefore, category (b) oncogenic fusion genes generally exhibit low expression levels. In the drug discovery perspective, it is hard to design drugs to restore the expression levels of tumor suppressors to the normal state, in order to "cure" cancer cells. Instead, it is easier to kill cancer cells using synthetic lethality. To be more specific, synthetic lethality refers to a situation when a mutation of one gene does not lead to apoptosis, whereas combination of that mutated gene with other mutated genes does (Figure 1-2). If there exists such two or more genes that forms such synthetic lethality interaction, with one of these genes be the oncogenic category (b) fusion gene (e.g., mutated Gene B in figure 1-2), drugs that targets the other genes will trigger apoptosis in cancer cells because all these genes in the synthetic lethality schema are silenced, whereas for normal cells, the tumor suppressor (e.g., wild type Gene B in normal cells) is still functional and the normal cells will be viable.

Figure 1-2. Synthetic lethality in targeted cancer therapy [14]. Loss of either gene A or gene B does not lead to cell death. But loss of both genes will trigger apoptosis. Targeting Gene A will lead to cell death for cancer cells, whereas normal cells will still be viable.



Similarly, same strategy can be applied to category (c) fusion genes, depending on the functional impact of the oncogenic fusion gene.

4. Summary

Cancer is one of the leading causes of death worldwide. Fusion genes, one among several important classes of mutations observed in various cancers, has received increased attention because oncogenic fusion genes are cancer-specific, which makes them ideal drug targets for cancer therapy. The rapid advancement of NGS technology has resulted in affordable transcriptome sequencing (RNA-Seq) for cancer patients. Hence, computational tools to predict the fusion genes from transcriptome data are in need more than ever for cancer treatment. In this study, we address this very need by developing ChimeRScope, a tool for accurate prediction of fusion genes given the transcriptome data of a cancer patient.

Chapter 2: THE DISCOVERY OF FUSION GENES

1. Introduction

1.1. Use of sequencing technologies for the discovery of fusion genes

Before the advent of Next Generation Sequencing (NGS), the most common methods for identifying fusion genes were Fluorescence In Situ Hybridization (FISH), and Reverse Transcription Polymerase Chain Reaction (RT-PCR). Both the methods require probes (short DNA/RNA sequences that are complementary to the target sequences) that are designed to specifically bind to the targeted DNA/RNA sequences. Therefore, prior knowledge of the candidate fusion genes is required for performing these experiments, which restricts their use on large-scale fusion gene analysis. Moverover, they are also not applicable for identifying novel fusion genes.

Sanger sequencing, or Sanger Dideoxy Sequencing invented by Frederick Sanger and colleagues in 1977 represents the first generation sequencing technology. Sanger sequencing generates long sequences with high accuracy, which can make the downstream analysis (mapping/assembly) much easier when compared to NGS platforms. Sanger sequencing is often used in fusion gene analysis as a confirmation step because it can determine the contiguous nucleotide sequence of the fusion junctions. However, the bottleneck for Sanger sequencing is the low sequencing speed and high costs. In 2011, the average sequencing cost for Sanger sequencing is 6,000 dollars per trillion bases (or gigabyte) with a speed of only 1.5 million bases (or megabyte) per hour [15]. Comparatively, the sequencing cost for NGS platforms like Illumina is 100 dollars per trillion bases with a sequencing speed of 20 million bases per hour. Hence, Sanger sequencing is also not applicable to large scale fusion gene analysis.

The rapid advances in the NGS technologies in the past decade provided an excellent opportunity to explore the genetic architecture of personal genomes. The low costs and high-throughput capacities make it possible for sequencing complete genomes/transcriptomes in an

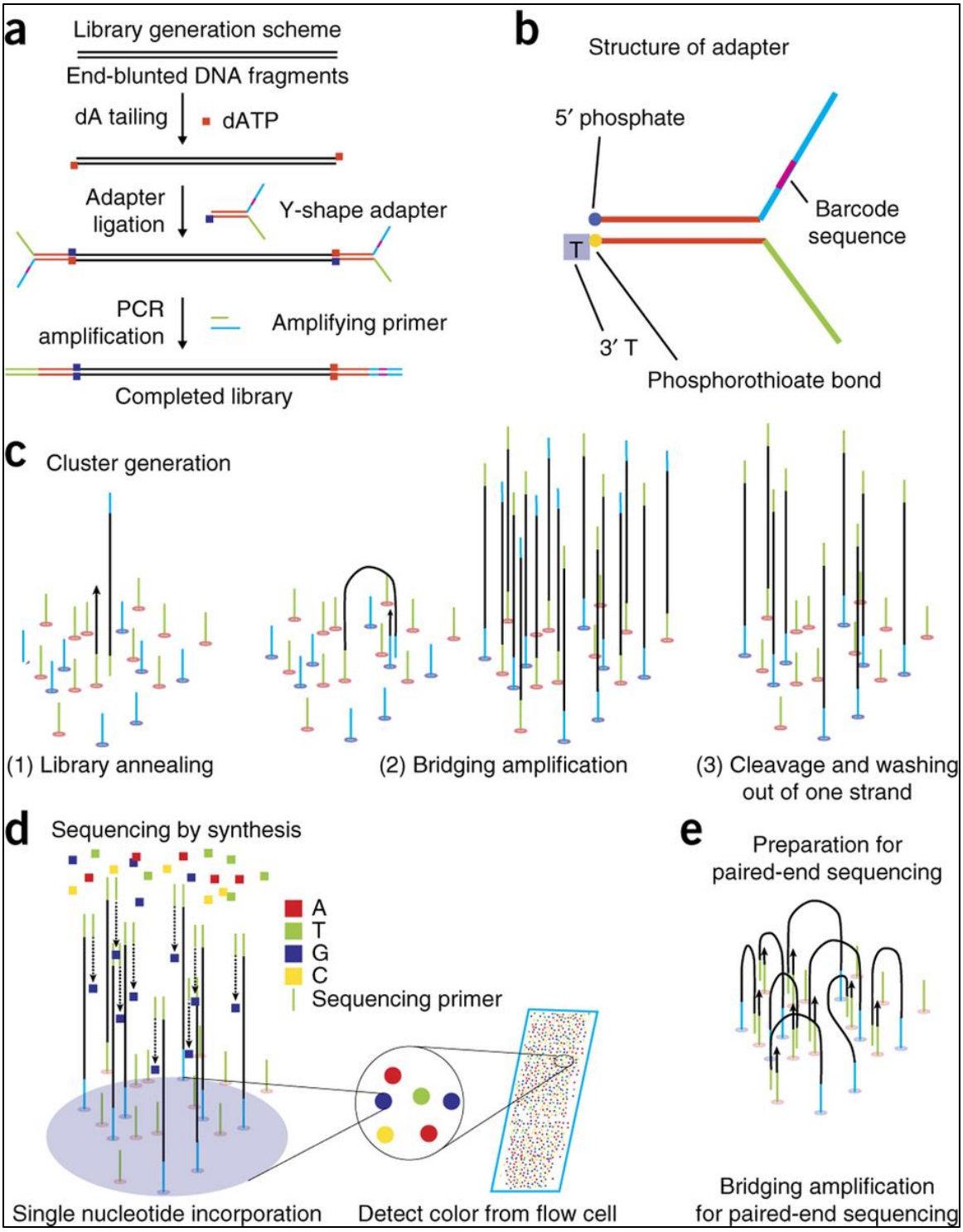
efficient manner at an affordable cost. Among all the NGS platforms, Illumina(Solexa) sequencing technology has been the most widely used NGS method across the world, dominating the field by taking 70 percent of the market share in the genome-sequencing platforms [16]. Illumina platform is a massively parallel sequencing (sequencing-by-synthesis) system that uses "bridge amplification" to produce millions of short reads at once. To be more specific, the first step of Illumina sequencing is to randomly fragment the genomic DNA (or cDNA for transcriptome sequencing). Normally, only fragments with length in a certain range (300-500bp) will be used for sequencing, in order to improve the overall sequencing quality. If the fragment is too short (or shorter than the designed read length), it will introduce adapter contaminations to the reported reads because the sequencing range will extend to the adapters. This will lead to several problems in the data analysis step. On the other hand, if the fragment is too long, the sequencing quality diminishes toward the tail end of the sequence read (Figure 2-1) because longer sequences tend to bend, resulting in low sequencing quality for that cluster or even infiltration of adjacent clusters during bridge amplification step. After washing out the unqualified DNA fragments, adaptors will be added to both sides of the remaining DNA fragments for amplification. After the initial fragment amplification step, all these DNA fragments will be amplified and the library is ready for sequencing (Figure 2-1a). The bridge amplification step begins by washing DNA library across the flow cell. All the DNA fragments in the library will then bind to the flow cell (Figure 2-1c, part 1). The flow cell is a plate with short oligonucleotides complementary to the adaptor sequences. It can have multiple lanes and each lane can form millions of clusters. During each cycle of the bridge amplification, a DNA fragment will form a "bridge" with its neighboring primer. By using this DNA fragment as the template, it will synthesize a double stranded DNA, where the synthesized strand is the reverse (because of the adaptor position) complementary strand of the template (Figure 2-1c part 2). Next, the hydrogen bonds between these double stranded DNA break and each of DNA strand (including newly synthesized ones) will then form a new "bridge" with its neighboring primer. After a certain number of cycles, this "cluster

formation" stage will be completed. Ideally, each DNA fragment in the library will be amplified in an exponential rate ($2^{\text{number of cycles}}$) and the cluster originated from that DNA fragment consists of two mixed sub-clusters (original strand cluster and synthesized strand cluster). In single-end sequencing, the synthesized strand cluster will be cleaved and washed out (Figure 2-1c part 3). Comparatively, in paired-end sequencing, another cycle of bridge amplification will be conducted after the sequencing for the original strand is done, in order to generate the synthesized strand cluster within the same cluster. Then, the original strand cluster will be cleaved and washed out, leaving the synthesized strand cluster for sequencing (Figure 2-1e).

Figure 2-1d illustrates how Illumina sequencers capture the chemical signals and translate them into human-readable sequence information. Four types of fluorescently labeled nucleotides (A, T, C, and G) are added to the flow cell. During each sequencing cycle, only one nucleotide will be added to the synthesizing DNA in that cluster and the related fluorescent signal is released. Because all the fragments within the same cluster are identical, they will release the same fluorescent signal during that sequencing cycle, and the signal will be strong enough to be captured by the built-in light sensor (Figure 2-1 d). At the end of each sequencing cycle, an image will be taken from the flow cell, capturing all the nucleotides that were added to all the clusters during that sequencing cycle. After all the sequencing cycles are done, all the images will be stacked together. The nucleotide sequence for a certain cluster (DNA fragment) can be obtained by checking all the colors at the corresponding location from the first image to the last image. Because the flow cell contains millions of sites in each lane Illumina sequencers can generate millions of short reads in one run in the massive parallel manner; thus, drastically improving the sequencing throughput. The paired-end reads reported from Illumina sequencers are in the ***forward-reverse*** direction because of the bridge amplification. For example, a paired-end read of 200bp (forward read 100bp, reverse read 100bp) sequenced from a DNA fragment of 300bp is reported as follows. The forward read reports nucleotide sequence that is the same as the fragment from position 1 to 100, and the reverse read reports nucleotide sequence that is the

complementary form of the fragment from position 300 to 201. This leaves the 101 to 200 bases unsequenced (defined as the ***insert-size*** for paired-end reads). This gives information about how close the forward read and the reverse read should be, and hence paired-end sequencing is extremely useful for accurate alignment of NGS reads to the reference genomes and especially aid in resolving alignments near the repetitive regions or multiple alignments.

Figure 2-1. Bridge amplification mechanism from Illumina Sequencers [17]. a. Library preparation. Random fragmented DNA with certain size (usually 300 to 500bp) were selected and ligated with adaptors on both sides. The next PCR amplification step will amplify these DNA fragments in order to construct the complete library for sequencing. b. The structure of the Y-shaped adaptor. The Phosphorothioate bond on the 3' end provides resistance to nuclease. c. Bridge amplification. The main purpose of this step is to, for each DNA fragment, generate a cluster of the same DNA fragment, thereby creating sufficient fluorescent signals when Sequencing-by-synthesis step actually happens. d. Sequencing by synthesis step. Primers and fluorescently labeled nucleotides are added to the sequencing chip for sequencing step to begin. When a new nucleotide is added in each sequencing cycle/reaction, the fluorescent signal of that type of nucleotide is emitted. Because the sequence cluster formed in the bridge amplification step are identical and the reaction time for each sequencing cycle is also constant. Each sequence in the same cluster will emit the same fluorescent signal during each sequencing cycle, representing an addition of that nucleotide during that sequencing cycle. Once all the sequencing cycles are completed, all images will be stacked together and translated to text-based sequencing result. e. During bridge amplification step, each sequence cluster actually contains both the original (original strand) form and its reverse-complementary (synthesizing strand) form of the same DNA fragment, due to the specific feature of the bridge PCR amplification. In single-end reads experiment, the synthesized strand cluster will be washed out and only the original strand will be sequenced. The strategy for generating paired-end reads involves an extra step by performing another cycle of bridge PCR. The synthesized strand will be generated in the same cluster. This time, the original strand cluster will be washed out and the synthesized strand will be sequenced (resulting in forward-reverse direction for Illumina paired-end reads).


a
Library generation scheme
End-blunted DNA fragments
dA tailing
dATP
Adapter ligation
Y-shape adapter
PCR amplification
Amplifying primer
Completed library
b
Structure of adapter
5′ phosphate
Barcode sequence
T
3′ T
Phosphorothioate bond
c
Cluster generation
(1) Library annealing
(2) Bridging amplification
(3) Cleavage and washing out of one strand
d
Sequencing by synthesis
A
T
G
C
Sequencing primer
Single nucleotide incorporation
Detect color from flow cell
e
Preparation for paired-end sequencing
Bridging amplification for paired-end sequencing

The first and foremost advantage of using NGS (particularly transcriptome sequencing, or RNA-Seq) for fusion gene prediction is that RNA-Seq offers novel and dynamic range of transcripts than probe-based methods. RNA-Seq does not require any prior information about the fusion gene and no transcript-specific probes are required in the sequencing step. In this case, there will not be any preferences on which transcripts (normal transcripts, alternative splicing isoforms, and fusion genes) will be captured for sequencing. This is extremely useful for data-driven research projects, or hypothesis-driven projects with limited information of the target fusion gene(s). Other advantages of using RNA-Seq over other probe-based methods include: 1) RNA-Seq is a method for comprehensive transcriptome analysis and it is cost-effective; 2) Probes designed for the target fusion genes can be less specific if mutations are manifested near the probe binding site, which makes the probe-based methods less sensitive to detect such fusion genes. Therefore, RNA-Seq has become one of the most widely used methods for fusion gene prediction in cancer research [18, 19].

Although NGS has been widely used in biomedical research, accurate methods that can be applied for genome-scale data analysis are still evolving. Each NGS run generates millions of reads and it requires enormous computational resources to process and analyze these datasets Additionally, NGS technologies like Illumina sequencing have higher error rate (~0.1%) when compared to Sanger sequencing (~0.001%). Also, the sequencing quality of the reads from Illumina sequencer tends to deteriorate towards the end of the reads due to the technical aspects of the Illumina sequencers (e.g., lower fluorescent signals caused by the decreased concentration of the fluorescent labeled nucleotides, or sequencing error caused by decreased DNA polymerase activity). Accordingly, attention should be paid at the data pre-processing step for NGS datasets. Moreover, the read length from Illumina sequencers (100-300bp) is much shorter than that from Sanger sequencing (500-1000bp), making the NGS data analysis harder to resolve highly repetitive regions than Sanger sequencing reads. Shorter reads also have higher chances of

aligning at multiple locations and it will require more sophisticate statistic models to determine the original genomic locations for these reads.

1.2. Current methods for fusion gene prediction using RNA-Seq

Over the last several years, many methods have been developed for detecting fusion genes from RNA-Seq datasets, most of which are alignment-based methods (methods that perform sequence alignment). Sequence alignment is a way of comparing two or more sequences, in order to identify similar (homologous) regions. Generally, there are two different approaches for sequence alignment: ***global alignment*** and ***local alignment***. Global alignment is used to compare two sequences that are similar with approximately the same size, whereas local alignment often aligns a shorter sequence (such as NGS reads) to a substring of a longer sequence (such as the reference genome) and it is more suitable for this problem. Therefore, in NGS data analysis, pairwise local alignment is used by these alignment-based methods for determining the most possible origins of the reads by checking which part of the genome/transcriptome contains the most similar sequences to those reads.

Here, we take the most representative local alignment algorithm used in NGS data analysis, the Smith-Waterman algorithm, as an example to briefly explain the details of how sequence alignment works. The Smith-Waterman algorithm, first introduced by Temple F. Smith and Michael S. Waterman in 1981 [20], is a dynamic programming algorithm that aligns all the possible combinations of the segments from two sequences and reports the best similarity measure between two subsequences of the query sequences. Figure 2-2 illustrates the detailed calculation step using SW algorithm. Here, *SeqA* of size 8bp (*CTTAAGCG*) and *SeqB* of size 7bp (*GGAGCGT*) were aligned against each other using a parameter set, where a match scores +2 and a mismatch scores -1. The first step is to construct a 2D matrix *H* of size $a+1$ by $b+1$, where a is the size of the first sequence (i.e. SeqA) and b is the size of the second sequence (i.e. SeqB). The H matrix is build using following rules.

$$H(i,0)=0, i\in[0,a]$$
$$H(0,j)=0, j\in[0,b]$$
$$H(i,j)=\max\begin{Bmatrix}0\\ H(i-1,j-1)+m(i,j), match/mismatch\\ \max_{k\geq 1}\{H(i-k,j)-W_k\}, deletion\\ \max_{l\geq 1}\{H(i,j-l)-W_l\}, insertion\end{Bmatrix}, i\in[0,a], j\in[0,b]$$

The basic idea of this algorithm is, for a given cell H(i, j), the score of that cell is calculated based on the scores of the top, left and top-left cells, represented by *H(i, j-1)*, *H(i-1, j)*, and *H(i-1, j-1)* respectively. Three new scores will be generated from each of these three directions and the maximum score will be taken for *H(i, j)*. The new score from top-left cell is updated with m(i,j). In the example in Figure 2-2, *m(i,j)* is defined as +2 if i^{th} character from SeqA matches j^{th} character from SeqB, or -1 if otherwise. The new scores calculated from the left cell or the top cell are updated based on *W* (gap-penalty score). If the gap-penalty score is a constant value *w*, meaning the gap-opening penalty and the gap-extension penalty are the same (e.g., gap penalty is -1 in Figure 2-2), the new score is calculated directly by using *H(i-1, j)-w* and *H(i, j-1)-w* for cells from left and top, respectively. If the gap-penalty score is different (e.g., most frequently used parameter set is, match = 2; mismatch = -3, gapOpen = 5, gapExtension = 2), then, the gap-penalty score *W* for a gap of size *g* should be calculated as *W= gapOpen + (g-1)* gapExtension*. If all of these three new scores are negative, it will reset the score to zero for the current cell. Therefore, bad alignment results from previous regions will not affect the score for other unchecked combinations. Once the whole scoring matrix done updating, we trace back from the cell with the highest score to each of the previous cell where the maximum score of that cell is generated from. The corresponding tracing path is the final alignment result. The example in

Figure 2-2 shows the best trace path *H(8, 6) -> H(7,5) -> H(6,4) -> H(5, 3)*, which gives the best local alignment sequence "AGCG".

Figure 2-2 SW-algorithm example. A sequence match scores +2, a mismatch scores -1, and gap penalty is 1. Alignment result for SeqA and SeqB found a high similar subsequence "AGCG".

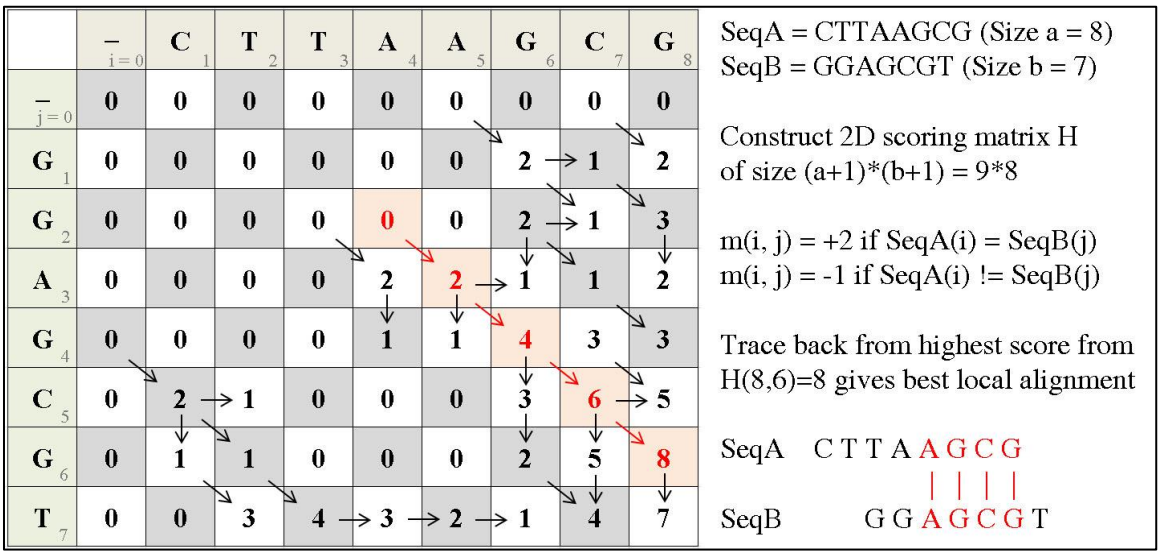


Table 2-1 lists 10 of the most widely used fusion gene detection methods using RNA-Seq data. These methods are all alignment-based methods. Based on different alignment strategies they use, these methods can be further categorized into five different groups. 1) FusionSeq [21], deFuse [22], FusionHunter [23], and SOAPfuse [24] identify discordant paired-end reads with the forward read completely aligning to one gene, and the reverse read completely aligning to another gene (referred to as ***spanning reads***). The fusion junction falls in between the ***insert-size*** region, or inside the gap between the paired-end reads. Then, they will look for paired-end reads that align across the gene fusion boundary (referred to as ***split reads***. Fusion junction is inside the forward read or the reverse read. Or the forward/reverse read is split by the fusion junction). 2) Methods such as FusionFinder [25], create smaller pseudo paired-end reads (pseudo-PE reads)

from each read by discarding the middle part of each single read. The smaller fragments from the pseudo-PE reads are then aligned to different genes separately. If there are enough pseudo-PE reads that have similar alignment pattern as the ***spanning reads*** for a gene pair, this gene pair will be reported as a fusion gene. 3) Tophat-Fusion [26] splits each single read into three equivalent fragments and tries to align the first fragment and the last fragment to different genes, just like the strategy used by FusionFinder for pseudo-PE reads. Next, if a fusion pair is obtained, the middle part of the read is then used to confirm the fusion boundary of that gene pair. For MapSplice [27], it splits the reads into more than three parts. Then the strategies of identifying fusion pairs and confirming fusion boundaries for MapSplice are quite similar to Tophat-Fusion. 4) FusionMap [28] and FusionCatcher [29] look for splitting reads from unmapped reads directly using a more exhaustive search approach. 5) JAFFA [30] performs the assembly from the short reads and constructs the data-specific contigs. These contigs are then aligned against the reference transcripts. If the contig sequence can be mapped to two genes exclusively without the mapped regions from each gene overlapping with each other, this gene pair will be considered as a fusion gene candidate for downstream analysis.

Table 2-1. 10 most widely used fusion gene prediction methods published since 2010. These methods are all alignment-based methods (sorted by published date).

Name	Author Affiliation	Publish date	Journal
FusionSeq	Weill Cornell Medical College	2010/10	Genome Biology
MapSplice	University of Kentucky	2010/10	Nucleic Acids Research
deFuse	University of British Columbia	2011/05	Plos Computational Biology
FusionHunter	University of Illinois at Urbana-Champaign	2011/06	Bioinformatics
FusionMap	Amgen Inc	2011/07	Bioinformatics
TopHat-Fusion	University of Maryland	2011/08	Genome Biology
FusionFinder	The University of Western Australia	2012/06	Plos One
SOAPfuse	Beijing Genome Institute	2013/02	Genome Biology
FusionCatcher	Orion Corporation	2014/11	bioRXiv
JAFFA	Murdoch Childrens Research Institute	2015/05	Genome Medicine

Sequence alignment, in contrast to alignment-free algorithm, has advantages in sequence analysis in certain conditions. It computes all possible pairwise comparisons between short reads and the reference genome sequences, and it will give excellent outcomes when the reads correctly map to the reference genome [31]. However, the reference genome used in most of the analysis is the one assembled from healthy individuals (e.g., human reference genome GRCh37/hg19 is derived from 13 healthy individuals). Aligning reads originated from perturbed cancer genome against the normal reference genome will prevent those reads from achieving good alignment, especially in the complex rearranged regions where fusion events occur, thereby resulting in low prediction accuracies [32].

2. Predicting fusion genes using alignment-free approach

2.1. ChimeRScope method Overview

Alignment-free methods are based on a broad collection of methods, including those based on *k*-mer frequency or substrings, on information theory, on graphical representation, or on sequence clustering. In this project, we designed a novel method named ChimeRScope (implemented in Java) for fusion gene prediction by assessing gene fingerprint (in the form of *k*-mers) composition from RNA-Seq short reads for fusion gene prediction. Unlike other methods that rank fusion candidates based on the number of supporting reads with reliable alignment, ChimeRScope generates gene fingerprint profile for each read and assigns different weights to the read based on the pattern of its gene fingerprint profile. This novel approach eliminates the need for alignment of reads to the reference genome; hence it is expected to work better with tumor transcriptomes that are encoded by severely perturbed cancer genomes. Specifically, a Gene Fingerprint Library (***GF-library***) will be constructed before the analysis (ChimeRScope *Builder*, Chapter 2 Subsection 2.2). This GF-library stores every possible gene sequence of size K, along with the

gene IDs that contains that *k*-mer sequence. Then, discordant reads that fail the initial alignment step in the standard RNA-Seq data analysis will be used as input for ChimeRScope. This helps to reduce the search space and improves the search speed. After that, a *k*-mer list will be generated using all possible substring of length k for each discordant read. A fingerprint profile will then be obtained by searching each *k*-mer of that read against the GF-library for gene IDs that contain that *k*-mer. Reads containing two sets of gene fingerprints from different genes will be scored based on the quality and the quantity of the fingerprint sequences. Such reads will then be marked as a Fusion Event Supporting Read (FESR) that supports the fusion event of corresponding genes (ChimeRScope *Scanner*, Chapter 2 Subsection 2.3).. After parsing all the discordant reads using this approach, all predicted fusion events will be ranked according to the overall scores of their FESRs (ChimeRScope *Sweeper*, Chapter 2 Subsection 2.3).. Our method is less vulnerable to the high rate of chromosomal abnormality (SNP, INDELs, translocation, and inversion, etc.) of cancer genome. For instance, if an inversion and an insertion both happened near a fusion junction and one read is generated from that fusion transcript, other tools will not be able to achieve reliable alignment because of this complicated chromosomal rearrangement. Consequently, this fusion read will be discarded. However, this read should still contain several gene fingerprints. ChimeRScope is capable of capturing those gene fingerprints (which indicates high sensitivity) and will predict this read as an FESR, although with relatively lower weightage score. We also implemented a targeted alignment module that maps the FESRs against the related fusion partners for detailed forms of the fusion genes. The results are reported in both text format and graphical outputs (ChimeRScope *Examiner*, Chapter 2 Subsection 2.5). Detailed methods and results will be discussed in the following chapters.

2.2. ChimeRScope Builder: constructing of species-specific Gene Fingerprint libraries

GF-library serves as a dictionary of *k*-mers so that for a given sequence of size *k*, the list of genes that contain that *k*-mer (considered as a gene fingerprint) can be retrieved in a constant time,

irrespective of the input data size (*O(1)* time complexity). The first step for the GF-library construction is to obtain the nucleotide sequences for every transcript. TopHat2 [33] aligner provides a script called *gtf_to_fasta* that takes the reference genome file (.fasta) and the gene model (in Gene Transfer Format, *.gtf) as inputs, outputting a single file that contains all the nucleotide sequences for all the transcripts in the gene model. Example command for *gtf_to_fasta* is shown as follows.

```
1 ## example for human reference genome hg38
2 gtf_to_fasta hg38.gtf hg38.fa hg38_allRNAs.fa
```

Once we obtained the file that contains all the nucleotide sequences for all transcripts, we can create the *k*-mer profiles for all transcripts by generating all the subsequences of size k from each transcript sequence. Each of these *k*-mer profiles contains all the unique *k*-mers found in that transcript. We then compare all the *k*-mer profiles, in order to get the dictionary file that tracks the origins for all *k*-mers (Figure 2-3).

Figure 2-3. Screenshot of the example *k*-mer library in the early development version (k=25). The first column shows the 25-mers and the second column lists all the related genes with 25-mer. Each element in the gene list is separated by a comma and each element contains information like built-in ID, chromosomal location of the gene, and the location of the *k*-mer in the sequence.

```
TGTTGCTGATTTTAATGTTATTTAA:      [>8049 chr10:102672326-102719347 TCONS_00008050:3354, >8050 chr10:102672326-102724891 TCONS_000080
51:3354, >8047 chr10:102672150-102713330 TCONS_00008048:3530]
AACTCTAATTTACAAAGGACAATCT:      [>7933 chr10:94050920-94113721 TCONS_00007934:1025]
CCAAATACAAAAATGAATAAAGCTT:      [>7396 chr10:22541001-22547477 TCONS_00007397:2450]
AAAAGAAAAACGAAAATATTGTGTT:      [>9741 chr10:76959177-76959880 TCONS_00009742:269]
ATTTAATTCTAGATCTTTCATTCTA:      [>7848 chr10:86039736-86054415 TCONS_00007849:1153]
TTCCAAAGCCTCATGTACGAGTGAG:      [>9298 chr10:115881220-115934364 TCONS_00009298:209, >9297 chr10:115881220-115934364 TCONS_0000929
9:209]
AAAGCAGAATCTTTGTAAAGTAATA:      [>7879 chr10:89267590-89313218 TCONS_00007880:32]
GGACGTGAAGCAGGTGTATTAAAAA:      [>9034 chr10:90973330-91011660 TCONS_00009035:225, >9033 chr10:90973326-91011660 TCONS_00009034:22
9]
GCAAATAAACGGAAACAGTTTTAAA:      [>9231 chr10:105062553-105109160 TCONS_00009232:1804]
TTATGACTTCATCTAGTTTTGGTAC:      [>8456 chr10:15147771-15210695 TCONS_00008457:4517]
TGGACAGGCTGCATGCCGTGTACTA:      [>8914 chr10:74766526-74790026 TCONS_00008915:1547, >8918 chr10:74766980-74856732 TCONS_00008916:1
093, >8916 chr10:74766980-74856732 TCONS_00008918:1093, >8915 chr10:74766980-74856732 TCONS_00008919:1093, >8917 chr10:74766980-74
856732 TCONS_00008917:1093]
ATAGTATCAAAAGTGGTACTTCAGT:      [>8213 chr10:116853124-117708496 TCONS_00008213:358, >8210 chr10:116853124-116931125 TCONS_0000821
2:358]
ATGCCAAAAGAGGAAGGTCTGCTAC:      [>8965 chr10:78629359-79397577 TCONS_00008966:3380, >8964 chr10:78629359-79397577 TCONS_00008968:3
380, >8967 chr10:78629359-79397577 TCONS_00008963:3380, >8966 chr10:78629359-79397577 TCONS_00008964:3380, >8962 chr10:78629359-79
397577 TCONS_00008965:3380, >8963 chr10:78629359-79397577 TCONS_00008967:3380]
AGCCCCTGAAAATACCTCCAGAGAG:      [>10009 chr10:116759790-116760333 TCONS_00010010:120]
GAGTGAAGGGCTGCTTTCTTCGAAA:      [>8900 chr10:73855790-73976199 TCONS_00008903:1973, >8902 chr10:73855790-73976199 TCONS_00008901:2
170, >8903 chr10:73855790-73976892 TCONS_00008906:1973, >8897 chr10:73855790-73975867 TCONS_00008900:1973, >8904 chr10:73855790-73
976892 TCONS_00008905:2170, >8898 chr10:73855790-73975867 TCONS_00008899:2170, >8899 chr10:73855790-73975867 TCONS_00008898:2170,
>8901 chr10:73855790-73976199 TCONS_00008902:2170, >8905 chr10:73855790-73976892 TCONS_00008904:2170]
GCAGATAAACCAAGGAAGGAGGAGG:      [>7530 chr10:44124265-44170147 TCONS_00007531:276]
CACTAGGTGTTGAGGACATAGAAAT:      [>8581 chr10:31605457-31608024 TCONS_00008582:445]
GGCGGCTTCCTCATCAGGCCTTTTC:      [>9419 chr10:127702902-128077127 TCONS_00009420:3337]
TACAAACTGATTTCTGAGAAACCAG:      [>9329 chr10:119764427-119806114 TCONS_00009330:3692, >9328 chr10:119764427-119793524 TCONS_000093
29:3692]
CACTCCACGACTTTAGACATCAAAT:      [>7794 chr10:81370695-81375199 TCONS_00007792:425, >7790 chr10:81370695-81375199 TCONS_00007793:31
1, >7795 chr10:81370695-81375199 TCONS_00007791:455, >7791 chr10:81370695-81375199 TCONS_00007795:395, >7793 chr10:81370695-813751
99 TCONS_00007794:319, >7792 chr10:81370695-81375199 TCONS_00007796:466]
```

According to NCBI (National Center for Biotechnology Information) Reference Sequence Database (RefSeq annotation), there are 38,834 annotated mRNAs (at transcript level) in the latest built of human reference (GRCh38/hg38). Depending on the size of the *k*-mer, there will be 60 to 75 million unique *k*-mers (when *k* ranges from 15 to 29) shown in all these transcripts. For each discordant paired-end read, we compare all the *k*-mer sequences against the GF-library. The idea is to check every *k*-mer for all discordant paired-end reads (millions in average sized datasets) against the GF-library. Therefore, it is crucial to choose the most optimal data structure in order to solve the searching problem in constant *O(1)* complexity time.

2.2.1. Data structure for the GF-library

Alignment-free methods are often memory-intensive. During the development stage of ChimeRScope, we have tested many different data structures. Here, we discuss each of the data structure we have tried and the performance of that data structure.

2.2.1.1. HashMap

HashMap is an implementation of Map class in Java. It uses hash functions to each hash key to track the hash value. The average search time complexity for HashMap is *O(1)*. During early development stage, we stored detailed information of the *k*-mers in the HashMap, including gene IDs, chromosomal location, and fingerprint position (Figure 2-3). The library file was also saved to the hard disk using the default UTF-8 encoding. The average running time for creating such GF-library took more than 8 hours to finish by consuming 25 GB of RAM. It is still feasible to use because GF-library only needs to be created once. However, the primitive structure of the hash key and hash value makes it very slow and memory-intensive for massive data analysis.

We introduced binary transformation for *k*-mers and the gene IDs using the following ideas. Before binary transformation, the hash key in GF-library is a *String* object of size *k*. A *String* object in java can be considered as an array of Char type (character type known as 'Char' in computer science to represent any of the 256 printable characters), and each *Char* takes 16 bits. Therefore, each *k*-mer in the HashMap takes 16**k* bits. Each hash value is an array of strings with each element in the array stores all the related information for each gene with that *k*-mer. The estimated size for each hash value is at least 40**n* (*n* denotes the total number of associated transcripts and 40 is the estimated length of each string element, see Figure 2-3). Here we convert all these Strings into the binary forms using mapping functions. Each character in *k*-mer represents a nucleotide, and it can only take 4 different values (A, T, C, G). Here, we encode each 'A' into '00', 'T' into '01', 'C' into '10', and 'G' into '11'. Every four nucleotides can now be

converted into a 16-bit binary *String*, which can then be converted into one *Char*. Therefore, there will be up to four times decrease in size for hash value (Table 2-2).

As we mentioned earlier, there are 38,834 transcripts in human reference model. In the improved version, we assign a built-in ID of two *Chars* (reserved for up to $2*2^{16}$=131,072 different values) for each transcript. The hash value is now a String of size 2**t*, where *t* is the number of associated genes for the hash key.

Table 2-2. Estimated improvements for the HashMap object using binary transformation. There is up to 4 times reduction for hash key and at least 20 times improvement in average for hash value. Here, *k* is the size of the *k*-mer, and *t* is the number of genes associated with the hash key.

Binary Transformation	Before	After	Improvement rate
Hash key (*k*-mer)	k	$\left\lceil \frac{k}{4} \right\rceil^{a}$	Up to 4 times
Hash value (gene lists)	$> 40 \times t$	$2 \times t$	At least 20 times

a The math symbol here is the ceil function. Ceil function round up the number to the smallest following integer.

The binary strings of zeroes and ones can be interpreted directly as a numeric value by computers. Each binary string for hash key is converted to the *Chars* as an extra computational step. To test if the conversion step is necessary or not, we compare the computational costs for *Long* (a numeric data type that takes 64-bits, equivalent to four *Chars* in size) keys against *Char* keys with equivalent size in bits. Results (Table 2-3) have shown that HashMap using *Char* as keys perform much better than *Long* keys when the number of entries increases (average GF-library size is 70 million).

Table 2-3. The writing (write the object to the local disk) and the reading speed of the HashMap object using Long type or Char type as hash key data type. Tested for HashMap with 1,000 to 1,000,000 entries. The values are in nanoseconds. The ones with better performance in each group are highlighted in red.

Number of entries	Writing speed (nanoseconds)		Reading Speed (nanoseconds)	
	Long	Char	Long	Char
1,000	21,042,746	43,611,302	18,627,304	42,011,000
10,000	107,513,624	57,007,196	101,237,318	57,720,639
100,000	971,794,028	364,712,442	822,193,258	393,103,809
1,000,000	9,389,967,972	7,762,445,515	18,838,800,741	8,892,180,472

In summary, we used the HashMap data structure for GF-library. Since the size of the GF-library is too big, several approaches were used to improve the performance. The final GF-library uses the binary transformed String (Chars) as hash keys, and the shortened String as the hash values (every two Chars represent a gene ID). The detailed improvement for GF-libraries used in analysis listed in Table 2-4.

Table 2-4. Average run time, memory consumption, and library size on local disk for GF-libraries with different k value on average sized datasets. The code improvement shows that, for the same 25-mer library, the time costs decreased from 25 hours to only 25 mins (60 times improvement). It also reduces the memory usage from 100 GB to only 16.2 GB (approximately six times improvement). The size of the library object on the local disk dropped from 7.9 GB to only 1.8 GB (about four times smaller).

	Before	After improvement[a]								
k-mer size (bp)	25	13	15	17	19	21	23	25	27	29
Time (minutes)	25 hrs[b]	17	25	25	25	26	22	25	26	25
Memory (GB)	~100	10.8	16.5	17.4	19.3	16.4	14.6	16.2	15.5	21.8
File size (GB)	7.9	1.1	1.4	1.6	1.6	1.6	1.6	1.8	1.8	1.9

a The memory usage is calculated using TotalMem()-freeMem() functions during the running time. The actual memory usage should be smaller.

b Average time cost for analysis using initial GF-library takes 25 hours, comparing to only 25 mins for the same GF-library after the code improvement.

2.2.1.2. Other data structures

ChimeRScope loads the entire GF-library into RAM (Random-Access Memory) in the beginning of the analysis. Therefore, a minimum usage of 16 GB RAM is required before the analysis starts. To optimize the memory consumption, we tried several alternative indexing methods or data structures. However, none of these performed better than the HashMap.

Java *RandomAccessFile* class

Java RandomAccessFile class can be used to access the specific location of a random access file. It works as a file pointer and it can access the specific byte location of a file in *O(1)* complexity time (however slower than HashMap when number of entries are huge) without the need for loading the file into RAM. Using this strategy, we tried several data structures where

each *k*-mer can be mapped to a numeric value that is directly associated to the byte location of the corresponding gene IDs.

A data structure we applied to reduce the usage of RAM is a 'large bytes array'. We defined that for each *k*-mer, if that *k*-mer can be found in more than 100 transcripts, it is considered as a common *k*-mer and will not be used as a gene fingerprint. The maximum length of the gene IDs is 100*32-bits (each gene ID takes 32 bits), which is the maximum of 400 bytes per *k*-mer. In theory, for a *k*-mer of size of 17, there will be 4^{17} (~17.2 billion) possible *k*-mers (the actual number of 17-mers for human reference genome is around 62 million). It is necessary to reserve the maximum space for all possible *k*-mers for consistency. Now, for a given *k*-mer that ranked at *x* (can be easily calculated), we can get the related gene IDs by looking at position 400*x* in the file. However, the estimated size of the index file will be over one petabyte, which makes it unpractical to use.

We also tried to group *k*-mers and built separate index files within each group. This reduces total number of entries and at the same time, averages the maximum number of Gene IDs per *k*-mer by releasing unnecessary space reservations. One example we tried is to mask the last 4-mers for each 17-mer. Therefore, there will be 4^{13} = 67,108,864 entries and each entry contains information for 4^4 = 256 *k*-mers. However, it still takes at least $4*4^{17}$ (~68.7GB) even if in average, all *k*-mers were unique *k*-mers.

Next, we separated the GF-library file into two different files. One of the file is the index file that serves similar purpose as previous attempts. The difference is, instead of returning a list of gene IDs, it returns two numeric values. The first value is the start location of the *k*-mer while the second value is the size of the gene list. Then, these two values will be used to retrieve gene IDs from the second file. This data structure completely removes the necessity for space reservation. However, the sizes of these two files are still huge (each takes more than 200GB hard disk space).

Moreover, generating such index files also takes very long because of the calculation and large number of I/O operations (>20 hours).

Java *FileHashMap* class

Java FileHashMap class belongs to Map class. Instead of loading the whole Map into the memory, FileHashMap only keeps the hash keys in memory. The hash values are saved as serialized objects in a random access disk file. It will create two files, one with a *.ix extension (index for hash keys) and another with a *.db extension (hash values). The loading time for GF-library using FileHashMap takes less than 1 minute, comparing to ~17 mins for HashMap class. It also reduces the memory usage to only 4 GB. However, the downside of FileHashMap is the search time. The time cost for each searching query is 100 times slower than HashMap. For analysis involving millions of short reads, the time cost makes FileHashMap less affordable even with the improvement on memory use.

In Conclusion, considering all the advantages and disadvantages of the data structures tested, we chose HashMap class as the data structure for constructing the GF-library. The huge HashMap (GF-library) is generated by comparing all the *k*-mers for all transcripts, and saved to local disk using java ObjectOutputStream as a binary file. Moreover, the binary transformation of *k*-mers and gene IDs greatly improves the computational performance. Other data structures were not used for GF-library due to various issues such as data storage or searching speed.

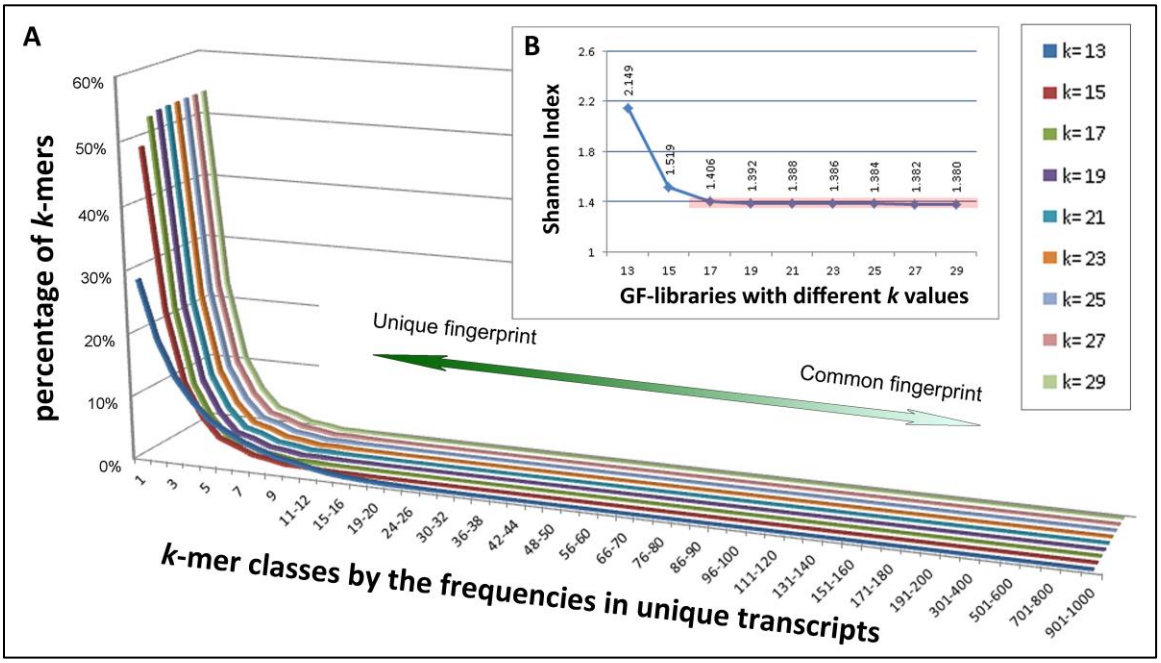
2.2.2. Determining the optimal *k*-mer length

The size of the *k*-mer will not only affect the size and the memory usage of the GF-library, it will also affect the sensitivities and specificities in the fusion gene prediction method. We used

the following criteria to optimize the best *k*-mer size. (1) *k* should not be too small, because a small *k* will generate redundant gene fingerprints; (2) *k* should not be too large because it generates too many unique *k*-mers resulting in a large search space. Also, a large *k*-mer will be more vulnerable to mutations (e.g., a SNP occuring in a *k*-mer at position 18 can affect a 19-mer but not a 17-mer). (3) *k* should be an odd number due to the design of the algorithm (*k*-mers were tracked using the index of the central nucleotides; hence, the inversion of the sequence will not alter the value of that index.

To decide the most optimized k value for human reference genome GRCh38/hg38 (with 38,834 mRNA transcripts), we plotted the *k*-mer composition for all *k*-mer libraries with all the odd *k* from 13 to 29 (Figure 2-4A). We defined that GF-libraries with higher level of discriminative *k*-mers (or lower Shannon Index [13]) are generally better for evaluating gene fingerprint sequences [34]. Shannon Index, known as Shannon's entropy, quantifies the uncertainty in predicting the species identity of an individual that is taken at random from the dataset [35] (initially described in ecological literature [36]). We calculated the Shannon Index using $H' = -\sum_{i=1}^{R} p_i \ln p_i$ (*i* refers to a *k*-mer type and p_i is the percentage of that -mer type. *R* is a collection of *k*-mer types that have been seen in no more than 100 transcripts) for these *k*-mer libraries, in order to measure the uncertainty in predicting the origins of the reads using different *k*-mer libraries. Higher the Shannon Index, more challenging it is to correctly predict the origins of a read. Results (Figure 2-4) have shown that *k*=17 is the most optimized *k*-mer size for human reference model (GRCh38/hg38) because it is the smallest *k* size that gives the equivalent levels of discriminative fingerprints (or similar low levels of Shannon Index) as those GF-libraries with larger *k* values.

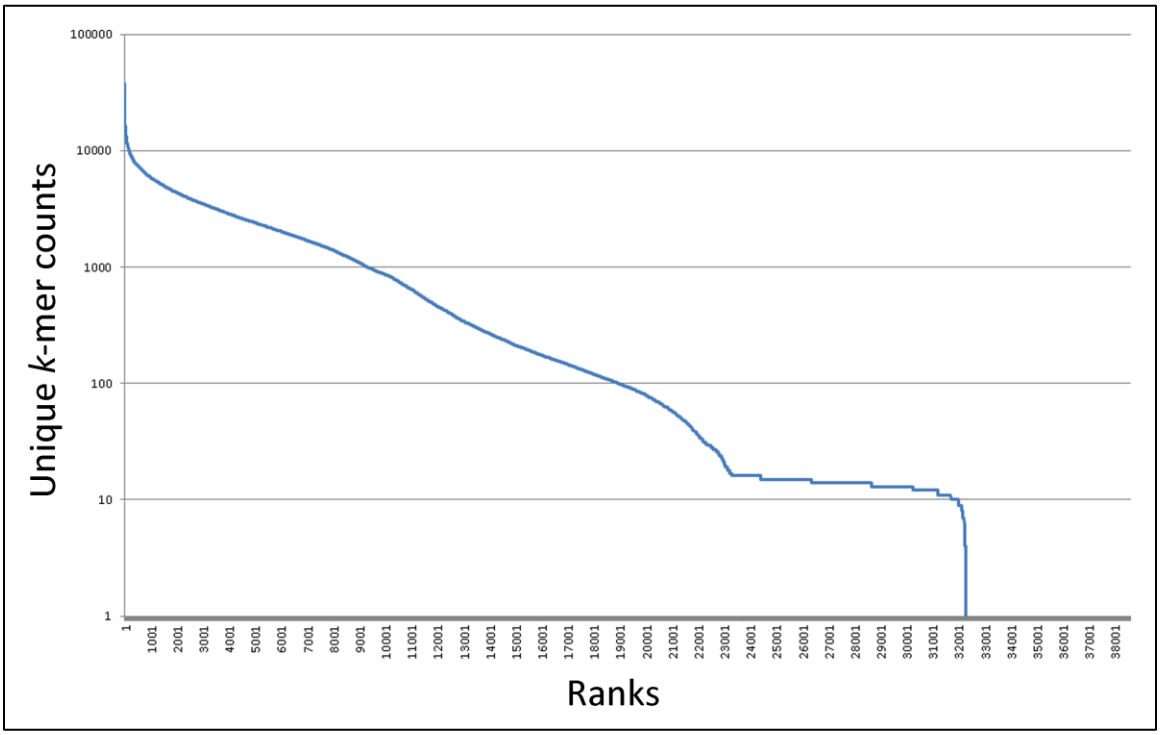
Figure 2-4. *k*-mer percentages for nine selected *k*-mer libraries (GRCh38/hg38, RefSeq mRNAs only). (A) The *k*-mer percentages for nine selected *k*-mer libraries (*k*=13, 15, 17, 19, 21, 23, 25, 27, 29) are plotted using 3D line chart, where each line in this chart represents a unique *k*-mer library. Libraries with larger *k* values are plots to the further side of the figure. The *x*-axis lists all *k*-mer classes (characterized by the number of transcripts that use the *k*-mer as a fingerprint) and *y*-axis shows the corresponding percentages of the *k*-mer class in the certain *k*-mer library. For instance, approximately 48% of the *k*-mers in the 15-mer library are unique fingerprint sequences (*y*=48% when *x*=1 and *k*=15). For each *k*-mer library, the majorities (~99%) of the *k*-mer classes are those discriminative *k*-mers with less than 10 associated genes. Overall, larger *k* often gives better *k*-mer library because it contains more discriminative *k*-mers (higher value towards left part of the *x*-axis). (B) Shannon Indices for all nine GF-libraries. GF-libraries for *k*=17 or larger have similar low levels (~1.400) of Shannon Indices (highlighted in the red box). Consequently, *k*=17 is the optimized *k*-mer size for GRCh38/hg38 because *k*=17 is the smallest *k*-mer size that gives highest levels of discriminative *k*-mers (or roughly lowest Shannon Index).



To conclude, we choose *k*=17 as the most optimized *k*-mer size for GRCh38/hg38 because it is the smallest *k* value that gives the highest levels of discriminative *k*-mers (or low Shannon index), comparing with the *k*-mer libraries with larger *k* (Figure 2-4).

We further evaluated the fingerprint sequences in the 17-mer library and plot the ranked distribution of all unique *k*-mers (Figure 2-5) for 17-mer GF-library. Results have shown that approximately 82% of all the transcripts (31501/38834=81%) have at least 10 unique fingerprint sequences. The figure also shows that more than 32,026 transcripts have at least one unique fingerprint sequence. Additionally, results from other discriminative *k*-mers (*k*-mers found in less than ten transcripts) have shown that more than 99.4% (38587/38834) of the transcripts contain at least one of these fingerprint sequences, suggesting 17-mer GF-library contains sufficient fingerprint sequences for most of the transcripts/genes.

Figure 2-5. The distribution of unique 17-mers across all transcripts. All transcripts are ranked based on the number of unique 17-mer fingerprints. Nearly half of the transcripts have at least 100 unique 17-mers. More than 32,000 transcripts have at least one unique fingerprint sequence.



2.3. ChimeRScope Scanner: identifying Fusion Event Supporting Reads from discordantly aligned reads

2.3.1. Extracting discordantly alignment reads

During the sequencing stage, most of the expressed RNAs (not only fusion transcripts) will be captured and sequenced into short reads. Because reads generated from normal transcripts will not support any fusion events, instead of searching fusion genes from the whole RNA-Seq datasets, we can align all the short reads against the normal reference genome, in order to filter out reads originated from normal transcripts. This will greatly reduce the search space to 10% of the total reads as only the discordant reads (reads that failed to align or aligned with conflicted insert-size) will be used for fusion gene prediction. Any aligners that support spliced alignment (e.g., TopHat/TopHat2 [33, 37], STAR [38]) for RNA-Seq short reads can be used in this step. For example, we can use Tophat2 to align paired-end reads (in *.fastq format) against reference genome (indexed genome, precisely). The aligned reads are then saved into a single file called *accepted_hits.bam*, whereas the unmapped reads are saved into a separate file named *unmapped.bam*. We can retrieve all discordant reads using SAMtools [39] from these two BAM files (Binary Sequence Alignment/Map format) based on the bitwise flag value of the reads. All the discordant reads can be converted back into the original fastq format using *bamtofastq* from BEDTools [40] repository. An example of bash script for this step is given as follows.

```
#!/bin/sh

## Set the path for required files, along with other necessary parameters
OUTDIR=[OUTPUT_DIRECTORY]    #Output directory
INDEX=[PATH_TO_INDEXED_GENOME]   #Bowtie1 index
READ1=[PATH_TO_FORWARD_READ] #Read 1 in paired-end read
READ2=[PATH_TO_REVERSE_READ] #Read 2 in paired-end read
THREADS=[NUMBER_OF_THREADS]  #Multi-threads

## alignment using tophat2 with bowtie1 index (just an example)
tophat2 -p $THREADS --bowtie1 --output-dir $OUTDIR $INDEX $READ1 $READ2
cd $OUTDIR

## Retrieve discordant reads (FLAG = 2)
## -f = INCLUDE; -F = EXCLUDE. -b = output binary format
samtools view -F 2 -b accepted_hits.bam > discordant.bam

## Combine discordant reads with other unmapped reads
samtools cat -o merged_unmapped.bam discordant.bam unmapped.bam

## Sort the merged BAM file based on read name
## bamToFastq requires the BAM to be sorted on read name
samtools sort -n merged unmapped.bam unmapped sorted

## Retrieve all reads with primary alignment (duplicates are removed)
## FLAG 256 means "not primary alignment"
samtools view -F 256 -b unmapped_sorted.bam > unmapped_sorted_primary.bam

## Convert all retrieved reads into fastq format.
## unmapped_1.fastq and unmapped_2.fastq are input reads for ChimeRScope
bamToFastq -i unmapped_sorted_primary.bam -fq unmapped_1.fastq -fq2 unmapped_2.fastq
```

2.3.2. Evaluating fingerprint sequences

As we mentioned earlier, Illumina's paired-end sequencing reads are in forward-reverse orientation. Therefore, for *k*-mers derived from paired-end reads, we will also need to check their reverse-complementary forms. Moreover, genomic variations like chromosomal inversion can also happen, which can inverse the direction of the genes in the genomic level. In this case, we should also check the reverse form (reverse of the original form) and the complementary form (reverse of the reverse complementary form) for these *k*-mers. In total, four different forms for each *k*-mer will be assessed, including original form (sequenced from template strand), reverse-complementary form (sequenced from non-template strand), reverse form (sequenced from template strand with an inversion), and complementary form (sequenced from non-template strand with an inversion).

ChimeRScope treats each gene fingerprint (*k*-mer) differently. A commonly observed gene fingerprint (shared in many transcripts) should have a small weightage score, whereas unique *k*-mers should be assigned with higher weightage scores because they are more discriminative. We also defined that for a common *k*-mer with more than 100 associated IDs, this *k*-mer will not be used as a fingerprint sequence. Using this idea, we calculated the weightage score of a given *k*-mer using the following scoring function. The final weightage score $w(x)$ is a normalized score that ranges from 0 to 1. The higher the score, the more unique the *k*-mer is.

$$w(x) = \frac{10^{1-\frac{min(x,100)}{100}} - 1}{10^1 - 1}, w \in [0,1)$$

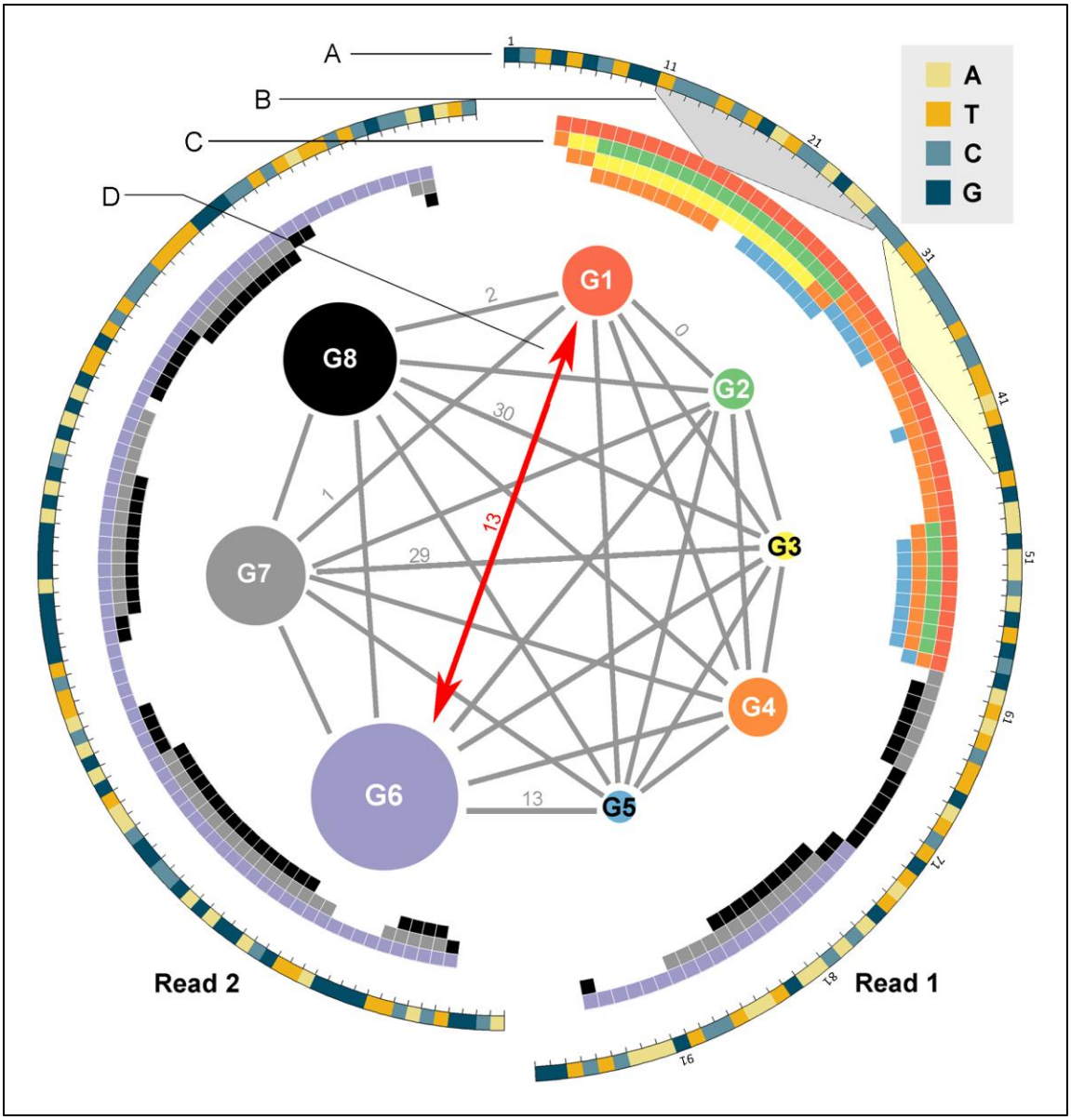
x = total number of unique IDs associated with the k-mer (and other variations)

We also tested the scoring function using Euler's number *e* as the base instead of 10. There is not much difference for the final result between these two different bases.

2.3.3. Detection of Fusion Event Supporting Reads (FESRs)

We model the problem of identifying the fusion genes from discordant paired-end reads into a problem in graph theory. Figure 2-6 shows an example of how we define a FESR based on its *k*-mer content. Details of the algorithm in mathematic terms will be shown later.

Figure 2-6. Strategies used in ChimeRScope for identification of FESRs: an example. (A) A discordant paired-end read (100bps x 2) that fails the alignment against the reference genome is plotted in a circular layout with each nucleotide base type represented by a unique color. (B) Four different variations (original, reverse, complementary, reverse complementary) for each possible ***k***-mer in the read (the 11th 17-mer for read 1 from base 11 to 27 is shown in this example) will be created and searched against the ***k***-mer library, in order to obtain, (C) a list of gene IDs that uses the corresponding ***k***-mer as fingerprints. Each block represents a ***k***-mer and each color here represents a unique gene ID. For example, four genes (G1: red, G2: green, G3: yellow, and G4: orange) are related to the 11^{th} 17-mer (from the 11^{th} nucleotide to 27^{th} nucleotide, as highlighted in grey region) and two genes (G1 and G4) are associated with the 29^{th} 17-mer (highlighted in light yellow). (D) A complete graph is drawn for all genes that are associated with both reads (in this figure there are 8 genes). Each vertex in the complete graph represents a unique gene with the size of the vertex proportional to the overall fingerprint weightage score (defined in the previous section) for that gene. The edge value between two genes is defined by the distance (denoted as *d*) between two closest fingerprints of the gene pair (only a few values are listed). Two genes with edge value less than the *k*-mer size will have at least *k-d* nucleotides overlap. Gene pairs with small distance values tend to be false positives due to the similar sequences (Chapter 2, subsection 2.4.1.2). Here, if we define that only those gene pairs with less than 5 bps overlap (or with the distance more than 17-5=12) can be valid fusion candidates, this read will be classified as a FESR that supports the fusion between G1 and G6 because G1 and G6 are two of the largest vertices with the edge value larger than 12.


A
B
C
D
A
T
C
G
G1
G2
G3
G4
G5
G6
G7
G8
2
0
30
1
13
29
13
Read 2
Read 1

Here we list all the notions we used in the following paragraphs for better explanations.

N :	The total number of the discordant reads
R_i :	The i[th] read, $i \in [1, N]$
L_i :	The length of R_i
J_i :	Total number of k-mers in R_i
k :	The size of the k-mer, should be the same as *k* defined in GF-library
$S_j(R_i)$:	The j[th] k-mer in R_i, $j \in [1, J]$
$H_j(R_i)$:	The list of transcripts that contains fingerprint sequence $S_j(R_i)$
Z_j :	The size of $H_j(R_i)$. $Z_j = size(H_j(R_i))$
$w_j(R_i)$:	The weightage score for $S_j(R_i)$. $w_j(R_i) = \frac{10^{1-\frac{min(Z_j,100)}{100}} - 1}{10^1 - 1}$
$U(R_i)$:	The list of all transcripts shown in the read R_i
x_i :	The size of the unique transcript ID list $U(R_i)$
$G_i(x_i)$:	An array of transcript IDs for R_i with the size of x_i
$M_i(x_i)$ or M :	A two dimensional square matrix ($M_i[x_i, x_i]$) that tracks overall weightage score for each transcript and the overlap relationship between each transcript pair.
D_i or $D_i(x_i)$:	D_i is the diagonal array of $M_i(x_i)$, It stores the overall weightage scores for all transcripts related to R_i. Can be calculated as $D_i[t] = M_i[t,t], t \in [0, x_i)$
P_i :	The gene fingerprint profile for R_i. P_i can be represented by a vector $P_i = (M_i(x_i), G_i(x_i))$
d :	The parameter that defines whether two fingerprints are considered as overlap or not.
$conf(i, G_x, G_y)$:	The confidence score of R_i that supports fusion event between G_x and G_y
$F(G_x, G_y)$:	The overall confidence score for fusion event between G_x and G_y

For each discordant paired-end read, we pose a sliding window of size *k* from its start position 1 to the last possible position $L_i - k + 1$, in order to generate all *k*-mers from i[th] read (R_i). For j[th] *k*-mer $S_j(R_i)$ in this read, we check its original form ($S_j^{[1]}$), reverse-complementary form ($S_j^{[2]}$), reverse form ($S_j^{[3]}$), and complementary form ($S_j^{[4]}$) against the GF-library, in order to obtain the list of transcript IDs associated with each variation ($H_j^{[1]}, H_j^{[2]}, H_j^{[3]}$, and $H_j^{[4]}$). Next, we calculate the union set of transcript IDs for this *k*-mer, represented by

$H_j(R_i) = \bigcup_{a=1}^{4} H_j^{[a]}(R_i)$. The weightage score of this *k*-mer $w_j(R_i)$ can be calculated using the scoring function we described before. Once all the *k*-mers have been processed, we can obtain the union set of all transcripts for this read using $U(R_i) = \bigcup_{j=1}^{L_i-k+1} H_j(R_i)$.

Next, we initiate the 2D square matrix $M_i(x_i)$, The row and column index can be projected to the same index for genes in $U(R_i)$. We only utilize half of the matrix $M_{a,b}(a \geq b)$. $M_{a,b}(a = b)$ or D_i is the diagonal array and it stores the quantitative value of overall fingerprint weightage score for all the transcripts identified in R_i . For each *k*-mer in R_i , we update the elements in D_i by adding the weightage score of the *k*-mer to these elements if the corresponding transcripts use this *k*-mer as a fingerprint sequence. Comparatively, $M_{a,b}(a > b)$ stores the distance (denoted as d) of two closest *k*-mers for each transcript pair after all the *k*-mers are parsed. If the distance is too small for the transcript pair, it suggests that these two transcripts share a common sequence of size $k - d$. ChimeRScope classifies such transcript pair as potential false positive (here we use transcript instead of gene for conveniently describing the problem, fusion genes found by ChimeRScope have to be between two different genes). The principle behind this can be described with the following example. Suppose we have $d = 1$ and $k = 17$ for gene X and gene Y in R_i , it means that these two genes share a common sequence of length 16. Suppose the read is sequenced from the cDNA of gene X only and there is a point mutation right after the "common sequence". There is a 1/3 of the chance that this mutation will change the *k*-mer originated from gene X into the *k*-mer derived from gene Y. Using the strategy we described in this subsection without the distance filter, this read will support a fusion event between gene X and gene Y while it is actually sequenced solely from gene X. If the distance value is less than the cut-off for a transcript pair, we define this transcript pair as an *overlapped pair* and filter it out.

The selection of the most optimized distance cut-off will be described in subsection 5.2 of this Chapter.

The problem of identifying FESR can be described as follows using mathematical terms.

Find the index of the maximum value in the diagonal array

$$m = \arg\max_{a}(M_{a,a}), a \in [0, x_i)$$

If there exists a value of s where

$$s = \arg\max_{a}(M_{a,a}), a \in \{a \mid (M_{a,m} \geq d) \,\&\, (a \neq m)\}$$

Here we have $M_{a,m} = M_{m,a}$ if $a < m$

Then, this read (ith read) supports fusion between G_m and G_s with a confidence score of $conf(i, G_m, G_s)$

$$conf(i, G_m, G_s) = \frac{4 \times M_{m,m} \times M_{s,s}}{J_i^2}$$

Basically, we try to find two largest values from the diagonal array where the corresponding transcript pair is not an overlapped pair. If there exists such pair, we define that this read is a FESR that supports the fusion event for this pair.

The evidence of why this confidence score function works the most appropriate for this problem is shown as follows.

The weightage score for a given k-mer ranged from 0 to 1.

$$w_j(R_i) \in [0,1)$$

which gives,

$$M_{m,m} + M_{s,s} \le \sum_{j=1}^{J_i} w_j(R_i) < J_i \;\Rightarrow\; 0 \le M_{m,m} \times M_{s,s} < (\frac{J_i}{2})^2$$

$$\Rightarrow\; 0 \le \frac{4 \times M_{m,m} \times M_{s,s}}{J_i^2} < 1$$

We define the confidence score

$$conf(i, G_m, G_s) = \frac{4 \times M_{m,m} \times M_{s,s}}{J_i^2} \;\Rightarrow\; 0 \le conf(i, G_m, G_s) < 1$$

It achieves the maximum value only and if only

$$M_{m,m} = M_{s,s} \approx \frac{J_i}{2}$$

The last equation shows a perfect fusion pattern with each end of the paired-end read (half of the total *k*-mer count) matched uniquely to each transcript. This is when the confidence score function achieves the maximum score. Decrease of the *k*-mer weightage score $w_j(R_i)$ (quality of the *k*-mer), or $M_{m,m}$ and $M_{s,s}$ on either/both sides (quantity of the *k*-mers) will lower the confidence score, which is also correlated with the significance of the fusion pattern.

2.4. ChimeRScope Sweeper: predicting fusion gene candidates

2.4.1. Summarization of the FESRs confidence scores

The confidence score for a given FESR can be calculated using the method described in the previous subsection. Once we identified all the FESRs, we group all the scores based on the supported fusion pair. We use an iterative function in java to summarize all the scores for each fusion gene candidate (Code shown below).

```
/*
* Summarize the final confidence score from a sorted list of FESR scores
* @param sorted float ArrayList
* @return final confidence score (float value ranged from 0 to 1)
*/
private float calculateScore(ArrayList<Float> sortedScoreList){
    float score = 0;
    boolean first = true;

    for (int i = 0; i < sortedScoreList.size(); i++) {
        if (first) {
            score = sortedScoreList.get(i);
            first = false;
            continue;
        }
        float tempScore;
        float newProb = sortedScoreList.get(i);
        if (newProb > score) {
            tempScore = score;
            score = newProb;
            newProb = tempScore;
        }
        score = score+(1-score)*newProb*(1-score)*newProb;

    }
    if (score>1) {
        score = 1;
    }
    return score;
}
```

This java function takes out the smallest FESR score from the list one at a time and updates this FESR score to the final confidence score. This function guarantees that, (1) the final score is always higher than the maximum FESR score. (2) The false positive fusion genes with large amounts of low quality FESRs will not have high confidence scores..

The final confidence score for a fusion gene ranges from 0 to 1. Any fusion gene with a score of more than 0.5 is considered as a true fusion gene candidate. The cut-off score 0.5 suggests the overall confidence for a fusion gene to be true when it receives, for example, 70% support in average from each fusion partner (0.7*0.7=0.49).

2.4.2. Filtering false positives

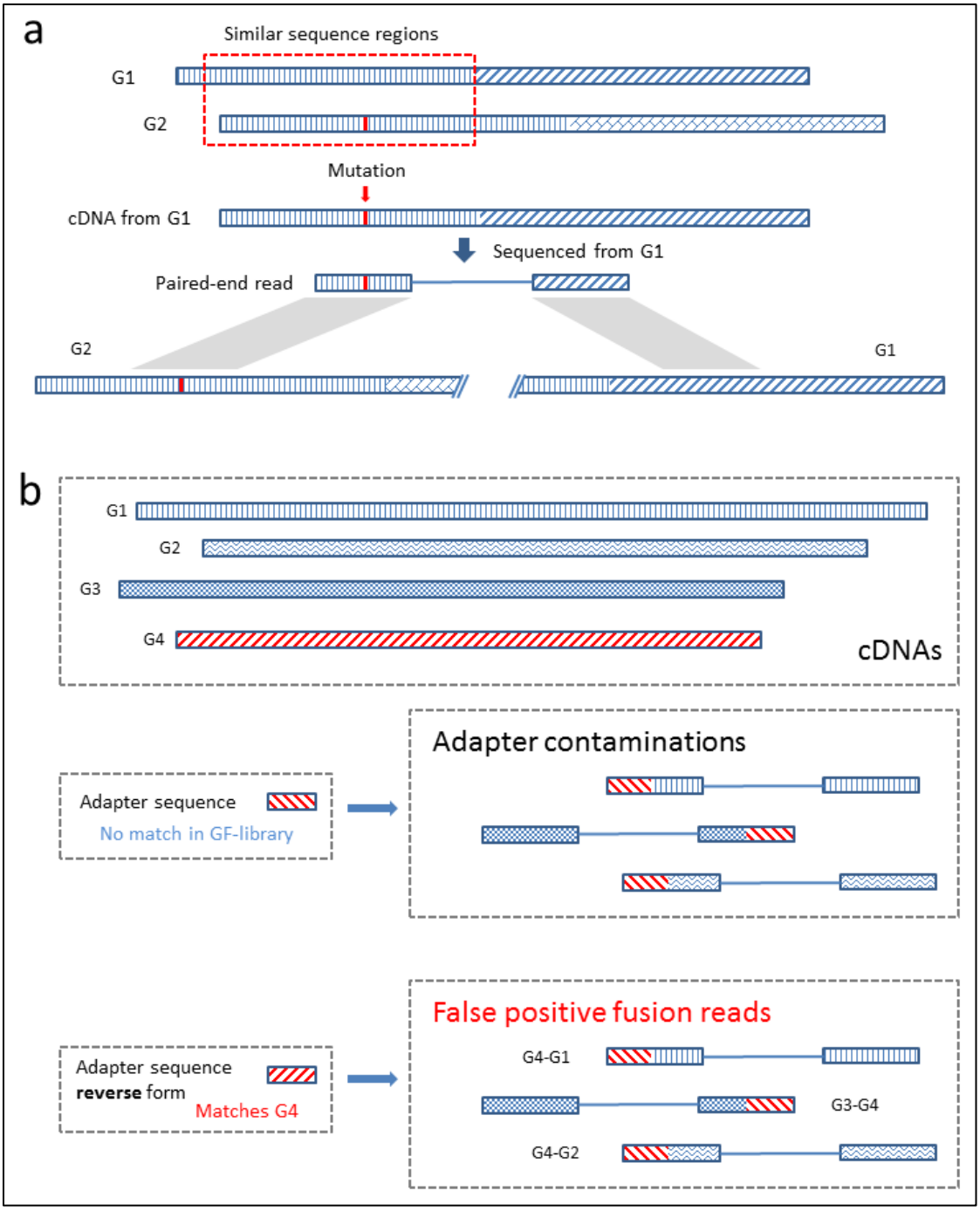
2.4.2.1. Fusion partners with similar sequence

In 3.3, we mentioned that genes with similar sequences can have higher chance to introduce false positives (Figure 2-7a), even with the right distance parameter. To avoid such false positives, we check the similarity of two sequences using the following methods.

We filter out the fusion gene if the fusion partners come from the same gene family. Genes from the same family often have similar functions. They can share homologous structures in certain regions like the functional domains. The HUGO Gene Nomenclature Committee (http://www.genenames.org/), known as HGNC, provides information of the genes and their gene families. ChimeRScope can take this optional information to filter out potential false positives caused by sequence similarity.

Two genes from different gene families can still have similar sequences. To address this, we align the fusion partner genes using Smith-Waterman algorithm (match = 2, mismatch = -3, gap open = -5, gap extension = -2). This parameter set for SW algorithm is the default parameter set used in NCBI BLAST (Basic Local Alignment Search Tool) for gapped alignment (blastn) and it gives the best result in most of the alignments. Two sequences with an alignment score of 200 or more is defined as similar pairs (because these two sequences need to have a sequence match of at least 100 bps) and will be filtered out in the final result. We choose the cut-off alignment score of 200 because the smallest length of the paired-end (PE) Illumina read is 100 bps (50*2). The maximum alignment scores for a 100bp PE read mapped to the similar regions of these two genes are also 200. Based on the predictions on the real RNA-Seq datasets, this cut-off filtered out most of false positives with similar sequences.

Figure 2-7. False positive fusion reads caused by genes with similar sequences and adapter contaminations. (a) An example of the false positives caused by similar sequence region. The reference sequences of G1 and G2 share a similar sequence near the 5' region. There are only a few nucleotides that are different (highlighted in red in G2). A paired-end read sequenced from the subject G1 cDNA contains a mutation that changes the fingerprint sequences in that region to the identical sequence of G2. This paired-end read, originated from G1 only, will now be classified as a fusion read because it supports the fusion between G1 and G2 due to that specific mutation. We filter out this type of false positives by checking if the reported fusion partners contain such similar regions. (b) An example of the false positives caused by adapter sequences. Paired-end reads can have adapter contaminations when the template is smaller than the length of the read or due to other technical issues. ChimeRScope checks four different variations of the *k*-mers, including the reverse form. Although adapter sequences are designed not to significantly match any of the gene sequences, the reverse forms of these adapter sequences can match to certain genes (G4 matches the reference form of the adapter sequence in this example). All the paired-end reads sequenced from the cDNA library with adapter contaminations can classified as fusion reads by ChimeRScope. We filter out this class of false positives by removing fusion genes involving partners with high counts because this kind of fusion partners tend to pair with a large number of genes.


a
Similar sequence regions
G1
G2
Mutation
cDNA from G1
Sequenced from G1
Paired-end read
G2
G1
b
G1
G2
G3
G4
cDNAs
Adapter sequence
No match in GF-library
Adapter contaminations
Adapter sequence
reverse form
Matches G4
False positive fusion reads
G4-G1
G3-G4
G4-G2

2.4.2.2. Conjoined genes

A conjoined gene is a gene that transcribed from two or more different genes (parent genes). These parent genes are often located close to each other on the same chromosome in the same orientation. A conjoined gene occurs either the transcription of the upstream gene is not properly terminated or if a transcript is not properly spliced; thus multiple exons from different genes are joined together to form the conjoined gene. Most of the conjoined genes are conserved [41] and not significant for tumorigenesis. Therefore, we implement a genomic distance filter (not to confuse with the distance parameter described in 3.3) in ChimeRScope. This filter marks the fusion genes into categories as "overlapped" (if two genes are overlapped at the genomic location), "<1k" (less than 1,000 bases), "<10k" " (less than 10,000 bases but more than 1,000 bases), "<100k" (10,000 bases to 100,000 bases), and "PASS" (more than 100,000 bases or on different chromosomes). Fusion genes from "PASS" class are considered as true fusion events. Fusion genes from "<100k" are also considered as true but they are further evaluated with caution. Others are classified as conjoined genes because they are too close to each other.

2.4.2.3. Adapter contamination

In subsection 1.1 of this chapter, we mentioned that Illumina sequencers use adapter sequences to anneal all DNA fragments to the flow cell. It is possible that NGS reads can contain adapter sequences due to a variety of reasons [42]. Although adapter sequences are designed not to significantly align to any part of the genome (both forward and reverse strand), the reverse form of the adapter sequences can be part of the fingerprints for a subset of genes due to complementary matching. Reads sequenced from any expressed gene containing adapter contamination can exhibit a fusion pattern between the expressed gene and genes from that subset. Such fusion pairs are false positives introduced by adapter contamination (Figure 2-7b).

We developed a simple filter for false positives caused by adapter contamination. This filter calculates how many times one gene is paired with other genes in the raw result. If a gene paired

with other genes more than other 200 genes, fusion genes involving this gene are more likely to be false positives. This cut-off value is optimized based on predictions on real RNA-Seq datasets, in order to avoid the majorities of the false positives. We do not set the cut-off to smaller values to avoid false negatives.

2.4.2.4. Unannotated transcripts

Human genome is a well-studied genome with more than 20,000 well-annotated genes and more than 38,000 transcripts in NCBI RefSeq annotations. However, it still contains many uncharacterized regions and transcripts. During the development stage of ChimeRScope, we identified a group of false positives, many of which are also predicted as true fusion genes in some of the studies [43]. These fusion genes have perfect pattern of two distant genes fused together. However, BLAST search revealed that these predicted fusion transcripts can be aligned to regions corresponding to non-coding genes or predicted gene models with high similarities indicating that such genes fusions are likely to be false positives. To address this issue, we recommend that, for analysis on real RNA-Seq datasets using ChimeRScope, a BLAST search of the fusion sequence against human nucleotide collection is necessary to remove such false positives.

2.5. ChimeRScope Examiner: alignment module with graphical output

As an alignment-free method, the core algorithmic part of ChimeRScope is not capable of identifying the exact fusion site because *k*-mers are treated as individual fingerprint sequences without keeping their chromosomal location information. Therefore, we implemented a separate alignment module called *ChimeRScope_Examiner* (*Step 4* in Chapter 3, subsection 2.2) to find the fusion junction coordinates using sequence alignment. We also transformed the coordinates reported from alignment results into vector graphics (Figure 2-8) for better result interpretation.

2.5.1. Targeted alignment for fusion junction

ChimeRScope *Sweeper* outputs a list of fusion genes ranked by their confidence scores. It also outputs a binary file that contains all the FESR sequences and the supported fusion pairs. In *ChimeRScope_Examiner*, four different variations (original, reverse complementary, reverse, and complementary) for each FESR are aligned against the sequences of the corresponding fusion partners using Smith-Waterman algorithm. In other words, four different alignment attempts were made for one read and a fusion partner. We defined that each fusion partner can only be aligned to one form of the read. Therefore, we only keep the read form with the highest alignment score as the primary alignment result form for that gene. We used the same parameter set for Smith-Waterman scoring matrix (match=2, mismatch=-3, gap open=-5, gap extension=-2) for all comparisons. Each alignment result is saved in a single line of text which contains the following fields. These are *read name*, *gene name*, *transcript ID*, *original strand of the transcript*, *read 1 or read 2* (paired-end reads), *aligned strand*, *aligned direction* (inversed or not), *matched coordinates in the read*, *matched coordinates in the original strand of the transcript*, *alignment score*, and *unmatched regions*. We also classified FESRs into ***spanning reads*** and ***split reads*** (Chapter 2, subsection 1.2) from the alignment results. Spanning reads give an approximated region for fusion junction while split reads support the exact fusion boundary. We calculated the estimated fusion junction from each FESR and report the consensus fusion point for the fusion gene. For a small sequence (often less than 50 bps) near the fusion junction that can be mapped to both fusion partners, we resolve the fusion junction using following rules: If the overlapping sequence at the junction covers an exon-exon junction for one fusion partner, we split the sequence at the exon-exon junction site and assign each part of the sequence accordingly. The rationale for this selection is based on the fact that the post-transcriptional modifications of the fused pre-mRNAs follow the same RNA splicing mechanisms. The fusion gene is more likely to

fuse at the exon-exon junction if the fusion at the genomic level is not occurred in the middle of a coding region (exon) and the splice sites are intact. If the overlapped sequence does not span any exon junctions, we assign the part of the sequence to the fusion partner with less number of unmatched bases. If two fusion partners report the same number of unmatched bases, we split the common sequence into half and report the fusion junction accordingly. We also report the orientation and the strand of the fusion partners, along with the resolved fusion sequences near the fusion junction (+/- 100bp of the fusion junction).

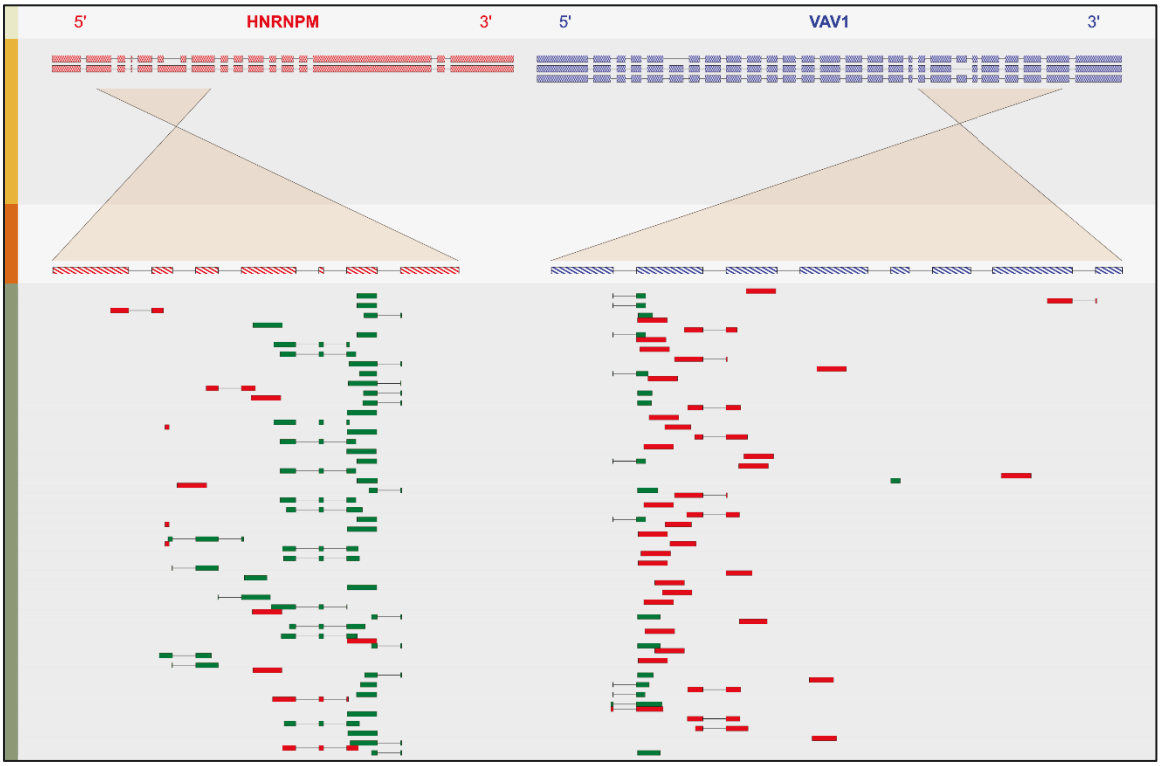
2.5.2. Scalable Vector Graphics images for fusion genes

For better understanding of the fusion patterns, we also transformed the target alignment results into SVG figures. Scalable Vector Graphics (SVG) is an XML-based vector image format. Unlike raster graphics that uses a dot matrix to represent shapes and colors, vector images like SVG use polygons to represent the objects. The objects in SVG images can also have different attributes such as stroke color, shape, thickness, and fill color. For instance, a simple rectangle in SVG is defined by its position (*x* and *y* in 2D coordinate system) and size (*width* and *height*). This object can be easily repositioned and transformed by wrapping it with a single line of code (a vector with parameters like *x*, *y*, *transform*, *scale*). This makes the objects in SVG images easily implementable and editable. Comparatively, similar objects in raster images often take larger space and can be edited only at the pixel level. Moreover, because vector images are not based on matrices of pixels, they can be infinitely resized without losing image quality. Therefore, we chose SVG as the standard format for ChimeRScope graphical report.

Figure 2-8 illustrates the layout of the SVG output generated by *ChimeRScope_Examiner*. The width/height ratio for the image is set to 4/3. *ChimeRScope_Examiner* calculates the exon length from the reference gene model. Then, we create boxes with difference sizes that are proportional to the length of the exons. We group these boxes and lines based on transcript names

for better editability of the objects. The start location of the group objects are set to the origin point [0, 0] and they will be transformed and moved to the right locations using vectors. The alternative transcripts for the same gene are aligned vertically based on the chromosomal location of the coding regions. The third track of the figure shows a maximum of 500bp region from each fusion partner near the fusion junction. We transformed this amplified region to better fit the width of the figure. This track will also be used as the standard ruler for the alignment results. We use two different colors for forward reads (red) and reverse reads (green). It is evident that both ***spanning reads*** and ***split reads*** are supporting the fusion event for HNRNPM-VAV1, as shown in track four of Figure 2-8.

Figure 2-8. Example SVG image for fusion gene HNRNPM-VAV1. This SVG figure is composed of four tracks separated by different background colors. The first track on the top shows the gene names of two fusion partners, along with their original orientation on the chromosomes. The second track lists all the transcripts from corresponding fusion partners. Each string of boxes represents a transcript that is involved in the fusion event. The boxes and the lines between the boxes represent exons and introns. The third track illustrates the amplified regions near the fusion junction for each fusion partner. The lines pointing from track two to track three show the orientation of the amplified region. If there is a cross between two lines, it means the amplified region is flipped over so that the fusion junction is between these two genes. The first three tracks use two different colors for fusion partner 1 (red) and 2 (blue). The forth track on the bottom lists the mapping results for a maximum of 71 paired-end reads. In this track, Forward reads are shown in red and reverse reads are filled with green color. Using the first transcripts from both genes as an example, the fusion junction is between the 3rd exon of HNRNPM and the 24th exon of VAV1. The SVG figure provides more information when opened with a web browser such as Chrome to help locate the fusion junction.



The recommended way of viewing this SVG output is to open the file using web browsers like Chrome because the SVG file contains several interactive contents. For instance, the exon number will be shown in the bottom of the image when users mouse-over to the exons in the second track of the image. Moreover, the read name for the corresponding paired-end read will also be displayed in the same region if users hover the mouse cursor to any of the reads shown in 4th track, making the SVG output more informative. The SVG file also shows the clear fusion junction and the direction of fusion in the two fusion partner genes. This SVG output file can be converted into publication-quality images using programs such as Photoshop or Inkscape.

2.6. Program and parameters optimizations

2.6.1. Computational cost and code optimization

The early development version of ChimeRScope took approximately ten minutes to load the GF-library into memory. It can process approximately 4000 reads per minute using multithreading option. For an average sized NGS dataset with millions of discordant reads, it took more than 10 hours and up to 100 GB RAM. In order to improve the performance, we optimized the algorithm by reducing number of iterations, optimizing data structures, and releasing unused variables in the program code. We describe these optimizations in the following paragraphs.

ChimeRScope was using a float square matrix to store the overall fingerprint scores for all transcripts. The overlap relationship is also stored in the same matrix. ChimeRScope (old version) uses several iterations to get the final updated matrix. The first iteration checks how many unique transcripts are shown in the read, in order to determine the size of the square matrix. The second iteration initiates the square matrix and calculates the overall fingerprint scores (diagonal array). The third iteration calculates the distance for each transcript pair and the last iteration determines the fusion gene from the matrix. We optimized the algorithm by separating the square matrix into a float array (denoted as A) and a numeric byte array (denoted as B). Array A stores the overall

fingerprint scores and the final size of A is the number of unique transcripts (denoted as *n*) shown in the read. Array B stores the pairwise distances of all the transcripts and the final size of B is (*n-1)*(n-2)/2*. In the optimized version, we do not pre-calculate the value of *n*. Instead, we define a preset size of array A and B. Whenever Java throws an IndexOutOfBoundsException (meaning the array container is smaller than what we want to use) for array A or B, we catch such exceptions and copy this array into an array with larger size (2X for array A and 4X for array B). This guarantees the least number of copy operations without reserving space that is too large that affects the performance. Using this idea, we omitted the need of the first iteration. We also combined the last three iterations into one iteration by updating all the arrays all at one for each *k*-mer we parsed. Overall, it reduced the total number of operations by about four times. We also decreased the memory usage of the square matrix (*n*n* floats) from about $32n^2$ bits into n*32+(n-1)*(n-2)/2*8=$4n^2$-12n+8, reducing about eight times even with small *n*.

We also optimized one of the most extensively used functions. As we mentioned earlier, we also check the complementary form of a given *k*-mer. The initial function for calculating complementary form of a *k*-mer checks each nucleotide (count as one operation) from the *k*-mer and put the complementary character (one operation for retrieving complementary char) into a new sequence (one operation for putting the char into new sequence). Therefore, there will be 3**k* operations each time when we retrieve the complementary form for a given *k*-mer (complementary operation happened before binary transformation). In the optimized version, we significantly reduce the number of operations by using bitwise operators. We took advantage of the bitwise operation that is directly supported by the processor. It operates the binary numerals at bits level. For example, the bitwise **NOT** (denoted by "~" in java) on a decimal value of 100 (01100100 in 8-bits binary) will return a decimal value of 155 (10011011 in 8-bits binary) because it performs logical negation on each bit (~01100100=10011011). In the new version of ChimeRScope, the complementary operation was performed after binary transformation. We optimized the binary transformation code chart with 'A' maps to '00', 'T' maps to '11', 'C' maps

to '01', and 'G' maps to '10'. Now, the complementary operation on the binary form of the *k*-mer is equivalent to the bitwise NOT operation (~'A' =~00=11='T' and ~'C' =~01=10='G', vice versa). This greatly reduces the operation time from 3*k to only 1 operation for each *k*-mer we processed.

Here we used a simulated dataset [28] to test the improvement in the performance. There are around 6000 discordant paired-end reads in this dataset. We also replicate the dataset to 10x, 100x, and 1000x of the original volume. Results have shown that the running time is significantly improved in the optimized version (10-100 times reduction). More importantly, it also scales better as the total number of reads increased. The estimated time for running an averaged size NGS dataset is less than 1 hour (the module looks for FESRs), which makes ChimeRScope much more attractive for massive data analysis.

Table 2-5. Speed costs on simulated datasets before and after code optimization. The time is calculated after excluding the constant GF-library loading time. Experiments using improved version take less than 40 GB RAM as against the dataset with the largest volumn (1000X). The total time for old version on 1000X volume dataset takes more than 24 hours and was killed due to Java OutOfMemoryError (specified 100 GB in that run). Therefore, it is marked as N/A (Not Applicable).

Volume	Read number	Old version	Improved version
1x	~ 6,000	00:01:20	00:00:04
10x	~ 60,000	00:13:48	00:01:06
100x	~ 600,000	02:53:17	00:02:18
1000x	~ 6,000,000	N/A	00:08:37

2.6.2. Distance cut-off for overlapped *k*-mer profiles

In subsection 2.3 of this chapter, we introduced a parameter called *distance*. It defines how close two fingerprint profiles are in the FESR and is calculated based on the index difference of two nearest *k*-mers. Small distance cut-off value tends to introduce more false positives caused by

gene pairs with similar sequences (Chapter 2, subsection 2.3). During the development stage of ChimeRScope, we have tested different distance cut-offs for the same simulated dataset [28] (contains 50 simulated fusions), using three different GF-library (k=15, 17 and 19). Results have shown that allowing 5 common nucleotides (distance = k-5) gives the best sensitivity and least false discovery rate irrespective of the GF-library (Table 2-6). It also confirms that GF-library with k=17 gives the best overall results, compared to other two GF-libraries.

Table 2-6 Detailed prediction statistics using three different GF-library (k=15, 17, and 19) with other parameter combinations on a simulated dataset with 50 true fusion genes, using the early development version of ChimeRScope. TP: true positive. FP: false positive. TPR: True Positive Rates, also known as sensitivity. TPR=TP/50. FDR: False Discovery Rate. FDR=FP/(TP+FP). The parameter *o* stands for *overlap* (*o=k-d*). Gene pairs with the fingerprint profiles with larger overlap size will be filtered out. The parameter *c* stands for read *number cut-off*. In this dataset, c=2 generally gives better result. Cells with TPR more than 90% and FDR less than 10% are highlighted in red. Results have shown that TRP=0.94, FDR=0.04 (k=17, o=5 or 6, c=2) gives the best result using f-score (f-score is described in detail in Chapter 3). Results have also shown that k=17 GF-library gives the best result using the right parameter set, comparing to that of other two GF-libraries.

		o	c=1	c=2	c=3	c=4	o	c=1	c=2	c=3	c=4	o	c=1	c=2	c=3	c=4
k=15	TP	0	48	46	44	39	1	48	46	44	39	2	48	46	44	39
	FP		113	37	21	15		91	31	16	12		84	31	17	12
	TPR		0.96	0.92	0.88	0.78		0.96	0.92	0.88	0.78		0.96	0.92	0.88	0.78
	FDR		0.7	0.45	0.32	0.28		0.65	0.4	0.27	0.24		0.64	0.4	0.28	0.24
	TP	3	48	46	44	39	4	48	46	44	40	5	49	46	44	40
	FP		67	21	13	8		45	18	10	8		32	14	8	7
	TPR		0.96	0.92	0.88	0.78		0.96	0.92	0.88	0.8		0.98	0.92	0.88	0.8
	FDR		0.58	0.31	0.23	0.17		0.48	0.28	0.19	0.17		0.4	0.23	0.15	0.15
	TP	6	49	45	44	38	7	44	40	37	32					
	FP		38	26	18	16		141	100	84	72					
	TPR		0.98	0.9	0.88	0.76		0.88	0.8	0.74	0.64					
	FDR		0.44	0.37	0.29	0.3		0.76	0.71	0.69	0.69					
k=17	TP	0	49	47	44	39	1	49	47	44	39	2	49	47	44	39
	FP		76	14	4	1		72	37	3	1		31	9	4	1
	TPR		0.98	0.94	0.88	0.78		0.98	0.94	0.88	0.78		0.98	0.94	0.88	0.78
	FDR		0.61	0.23	0.08	0.03		0.6	0.44	0.06	0.03		0.39	0.16	0.08	0.03
	TP	3	49	47	44	39	4	49	47	44	39	5	49	47	44	39
	FP		22	6	3	0		13	4	2	0		5	2	0	0
	TPR		0.98	0.94	0.88	0.78		0.98	0.94	0.88	0.78		0.98	**0.94**	0.88	0.78
	FDR		0.31	0.11	0.06	0		0.21	0.08	0.04	0		0.09	**0.04**	0	0
	TP	6	49	47	44	39	7	49	46	44	37	8	44	39	37	32
	FP		4	2	0	0		16	12	10	8		59	43	35	30
	TPR		0.98	**0.94**	0.88	0.78		0.98	0.92	0.88	0.74		0.88	0.78	0.74	0.64
	FDR		0.08	**0.04**	0	0		0.25	0.21	0.19	0.18		0.57	0.52	0.49	0.48
k=19	TP	0	48	46	44	39	1	48	46	44	39	2	48	46	44	39
	FP		14	3	0	0		10	3	0	0		9	3	0	0
	TPR		0.96	0.92	0.88	0.78		0.96	0.92	0.88	0.78		0.96	0.92	0.88	0.78
	FDR		0.23	0.06	0	0		0.17	0.06	0	0		0.16	0.06	0	0
	TP	3	48	46	44	39	4	48	46	44	39	5	48	46	44	39
	FP		6	2	0	0		5	2	0	0		5	2	0	0
	TPR		0.96	0.92	0.88	0.78		0.96	0.92	0.88	0.78		0.96	0.92	0.88	0.78
	FDR		0.11	0.04	0	0		0.09	0.04	0	0		0.09	0.04	0	0
	TP	6	48	46	44	39	7	48	46	44	39	8	48	46	43	38
	FP		4	2	0	0		4	2	0	0		5	2	1	1
	TPR		0.96	0.92	0.88	0.78		0.96	0.92	0.88	0.78		0.96	0.92	0.86	0.76
	FDR		0.08	0.04	0	0		0.08	0.04	0	0		0.09	0.04	0.02	0.03
	TP	9	44	40	37	33										
	FP		8	4	3	3										
	TPR		0.88	0.8	0.74	0.66										
	FDR		0.15	0.09	0.08	0.08										

To conclude, k=17 and d=12 (k-o=17-5=12) seems to be the best parameter set for fusion gene prediction analysis on human samples. Predictions from other datasets (discussed in Chapter 3) also shown high accuracies using these parameters.

3. Conclusions

In this chapter, we discussed the strategies used in most alignment-based fusion gene prediction methods and the drawbacks of alignment-based methods in general. We also discussed the detailed algorithmic part of our alignment-free method, namely ChimeRScope. We optimized the algorithm in different aspects and to greatly improve the performance of ChimeRScope. The parameter set, k=17 and d=12 (d=k-5), is set as default values for analysis on human datasets because it gives the best result on tested datasets.

Chapter 3: COMPARISONS WITH OTHER FUSION GENE DETECTION METHODS

1. Introduction

In NGS, all expressed transcripts are captured and sequenced, including those from chimeric fusion genes. These sequenced transcripts are then transformed into digital data, known as short reads. The number of short reads related to fusion genes is correlated with various factors such as the sequencing depth, the length and the expression level of the fusion genes. Besides, among all the reads derived from fusion transcripts, only reads that cover the fusion boundaries can be considered as direct evidences for the fusion genes (described as *fusion reads* in this study). Therefore, we estimated that only less than 0.001% of the total reads support the fusion gene junctions, irrespective of all factors described above. This poses a huge computational challenge to precisely identify this miniscule fraction of reads from tens of millions of total NGS reads.

Currently, published fusion gene prediction methods use different approaches to mine the fusion reads among millions of short reads. Many of the fusion genes have been identified and validated using these methods. However, mutations and sequencing errors near the fusion junctions hinder them from identifying these fusion genes. Recent studies [24, 29, 30] and reviews [32, 44] on fusion gene prediction methods have shown that SOAPfuse [24], FusionCatcher [29], and JAFFA [30] are generally the best methods for fusion gene prediction among all the publicly available tools. Therefore, we chose to compare the accuracy of ChimeRScope against these three methods in this study.

2. Materials and methods

2.1. Datasets

We have chosen two types of the datasets for assessing the accuracies of the selected fusion gene prediction methods. The first dataset type is the simulated datasets. Simulated datasets are

constructed with reads simulated from known number of artificially generated fusion genes. The advantages and disadvantages of simulated datasets are described as follows. Simulated datasets are often used to evaluate how good an algorithm works theoretically, because the characteristics of the datasets can be precisely controlled. They provide the precise number of true fusion genes, which is very difficult to get from real NGS datasets. However, because of the complexity of the transcriptome and potential unidentified patterns of the fusion genes, simulated datasets are generally considered as clean datasets and therefore, will introduce less number of false positives. In this study, we collected several simulated datasets containing known artificial fusion genes from previously published papers. One simulated dataset, originally released by FusionMap research group [28], consists a total of 50 synthetic fusions (namely *50_pos_set*). There are 114,418 paired-end read (2x75bp), with 8,600 simulated fusion reads that support 50 fusion events with RPKM levels ranged from 0.23 to 407.96 [28]. A review paper published in 2013 [32] compared the sensitivity and false discovery rate for six of the popular fusion gene detection tools using this *50_pos_set*. We chose this dataset for our study because it is one of the first publicly available simulated datasets and has also been used in many other studies [30, 32, 44]. Apart from *50_pos_set*, we also used another simulated datasets from the latest research paper [44] that comprehensively evaluated the performance of 15 different fusion gene prediction algorithms.

The second group of simulated datasets, namely *comp_sim_set*, consisted of 15 different datasets with three different read lengths (2x 50bp, 75bp, and 100bp) and five different coverage depths (5X, 20X, 50X, 100X, and 200X) for each of them. There are the same 150 simulated fusion genes in each dataset. We analyzed these datasets and carried out head-to-head comparisons using the same F-measure [44] for performance assessment. We did not use simulated datasets directly generated from SOAPfuse [24], FusionCatcher [29], and JAFFA [30] groups to avoid bias. We also did not generate simulated datasets of our own for the same reason.

However, there are several limitations of using the simulated datasets. Simulated datasets might contain biologically insignificant fusion events like read-throughs. Besides, the distribution of the mutations and sequencing errors in simulated datasets are often defined as simple distributions such as multinomial distribution. In this case, simulated datasets might not reflect the real state of fusion genes in cancer samples. To address these issues, we used several different real RNA-Seq datasets. We have tested ChimeRScope, along with SOAPfuse, FusionCatcher and JAFFA on seven published datasets that were obtained from four different breast cancer cell lines (namely *BC_CL*. Table 3-1) with 27 experimentally validated fusion genes [19]. We also tested these tools on Nature Killer (NK) cell lines (*in-house datasets*) and did experimental validation on all ChimeRScope predictions with commonly reported fusion genes. The *in-house datasets* were downloaded from NCBI Short Reads Archive [45]. It consists of four different RNA-Seq datasets (KHYG1, NKYS, NK92-PMIG, NK92-PRDM1), which can be divided into three different cell lines (KHYG1, NKYS and NK92). NK92 cell line samples used for sequencing were transduced with either PMIG, a control vector, or a vector to knock-down PRDM1, a known tumor suppressor. We only used the normal/non-transduced NK92 RNA in the experimental validation step for those fusion genes predicted from NK92 samples because the original vector treated NK92 RNA used in the transcriptome sequencing was not available.

Table 3-1. Description of the real RNA-Seq datasets from 4 breast cancer cell lines and the validated fusion genes. 27 fusion genes were validated across all four breast cancer cell lines. The gene ID highlighted in red, ENSG00000236127, is deprecated and no longer in the current EnsEMBL database. All gene symbols are the latest official gene symbols.

	Cell lines	Read length (bp)	Read number	Validated fusions [19]
Breast cancer cell lines	BT474	50	21,423,697	ACACA-STAC2, RPS6KB1-SNF8, VAPB-IKZF3, ZMYND8-CEP250, RAB22A-MYO9B, SKA2-MYO19, DIDO1-TTI1, STARD3-DOK5, LAMP1-MCF2L, GLB1-CMTM7, CPNE1-PI3
	KPL4	50	6,796,443	BSG-NFIX, PPP1R12A-SEPT10, NOTCH1-NUP214

	MCF7	50	8,409,785	BCAS4-BCAS3, ARFGEF2-SULF2, RPS6KB1-VMP1
	SKBR3	50	18,140,246	TATDN1-GSDMB, CSE1L-ENSG00000236127*, RARA-PKIA, ANKHD1-PCDH1, CCDC85C-SETD3, SUMF1-LRRFIP2, TBC1D31-ZNF704, CYTH1-EIF3H, DHX35-ITCH, NFS1-PREX1

* ENSG00000236127 is not a valid identifier in the latest EnsEMBL database (deprecated identifier).

2.2. Detailed analysis pipelines

As we stated earlier, we mainly focused on comparing ChimeRScope against SOAPfuse, FusionCatcher and JAFFA. The analysis pipelines for a selected tool on simulated datasets and real datasets are mostly the same. In this section, we will describe these pipelines in details.

ChimeRScope: Here we listed the complete ChimeRScope pipelines for all datasets. All the steps are labeled separately. The differences among the pipelines for different datasets are shown in Table 3-2. All major parameters are also listed, although most of the time the default values are used.

Step 0 – ChimeRScope *Builder*: Generate GF-library

We build the GF-library with *k* equals to 17 from human reference build GRCh38/hg38.

```
## Create GF-library with selected k-mer size
java -Xmx40g -jar ChimeRScope.jar Builder
    -i mRNAs.fa \ # all mRNA sequences in fasta
    -o $GF_LIB \ # output directory for GF-library, with prefix
    -k 17 \ # k-mer size. default: 17
```

Step 1 – Retrieve discordant reads (Chapter 2, subsection 2.3)

Step 2 – ChimeRScope *Scanner*: Identify FESRs

```
## Identify FESRs from discordant reads
java -Xmx40g -jar ChimeRScope.jar Scanner
   -fq1 unmapped_1.fq \ # discordant reads, forward
   -fq2 unmapped_2.fq \ # discordant reads, reverse
   -lib $GF_LIB \ # GF-library
   -k 17 \ # size of the k-mer used in GF-library
   -o ./ChimeRScope_Out \ # output directory
   -t 1 \ # number of threads
   -d 12 # distance cut-off. Recommend d=k-5
```

Step 3 – ChimeRScope *Sweeper*: Summarize FESRs scores with filters

```
## Summarize FESR scores for fusion candidates
java -Xmx40g -jar ChimeRScope.jar Sweeper \
   -i chimeRScope_out/raw_FESRs.txt \ # Output from previous step
   -cc 0.2 \ # confidence score cut-off for FESR to be true
   -mc 0.36 \ # Default. True fusion must have at least 1 read with score>0.36
   -lib $GF_LIB \ # library path with prefix
   -rn 2 \ # read number cut off
```

Step 4 – ChimeRScope *Examiner*: Graphical output and analysis of the fusion sequences

```
java -jar ChimeRScope.jar Examiner \
   -i ./ChimeRScope_Out/raw_FESRs.txt_rawFusions.txt \ # Output by Sweeper
   -lib $GF_LIB \ # library path with prefix
   -indir ./ChimeRScope_Out \ # mRNA sequences
   -alignOut ./ChimeRScope_Out/blastResult.txt \ # alignment result
   -o chimeRScope_ExaminerOut # other output (e.g. svg figures)
```

Step 5 – BLAST fusion sequences against human nucleotide collection

To summarize, we did not apply any filters for simulated datasets (including the last BLAST step) because of the limitations of simulated datasets we described earlier. We reduced the *rn* parameter to 2 because some of the simulated datasets have less than five reads for some true fusions. Besides, the step for extracting discordant reads from alignment step is not used for

comp_sim_set because they are not mixed with background reads. For analysis on real RNA-Seq datasets, we applied all the recommended filters. For *in-house datasets*, we also increased the *mc* parameter to 0.5 (at least one FESR should have a weightage score of more than 0.5, which suggests that around 70% (0.7*0.7=0.49) of the *k*-mer composition belongs to two fusion partners.) in order to get high confident fusions.

Table 3-2 Detailed pipelines used in four different datasets. Cells highlighted in red are different from the default values.

		50_pos_set	*comp_sim_set*	*BC_CL*	*In-house datasets*
Step 0		GF-library with k=17, one time generation			
Step 1		√	×	√	√
Step 2		√	√	√	√
Step 3	rn	2	2	5	5
	hf	×	×	√	√
	mc	0.36(0.6*0.6)	0.36(0.6*0.6)	0.36(0.6*0.6)	0.49(0.7*0.7)
	Other filters	×	×	√	√
Step 4		√	√	√	√
Step 5		×	×	√	√

SOAPfuse: We ran the SOAPfuse (version 1.26) pipeline using the default parameters. The version of the reference database is GRCh38 release version 79. Other reference files are prepared under SOAPfuse instructions.

For the *in-house datasets*, we applied the similar filters for SOAPfuse using "awk" in Linux.

```
cat SAMPLE.final.Fusion.specific.for.genes |
    awk '($2!=$7 ||
    ($2==$7 && ($4-$9>100000 || $9-$4>100000))) &&
    $11+$12>=5 {print}'
```

Explaination:

$2!=$7: two genes are on different chromosomes (for read-through filter)

$2==$7 && ($4-$9>=100000 || $9-$4>=100000): If two genes are on the same chromosome, the distance of these two genes should be more than 100,000 bp

$11+$12>=5: read number no less than 5.

FusionCatcher: FusionCatcher (version 0.99.4d beta) by default uses three different aligners. They are Bowtie [46], Blat [47], and STAR aligner [38]. We failed to set up STAR aligner for FusionCatcher due to the version conflict. Therefore, we disabled the use of STAR in FusionCatcher pipeline. Other parameters used in the pipeline are set as default.

We applied similar filters for FusionCatcher on the *in-house datasets* as well.

```
cat final-list_candidate-fusion-genes.txt |
    awk '$5>=5 {print}' |
    grep -v readthrough
```

Explaination:

$5>=5: read number no less than 5

grep –v readthrough: filter out all read-throughs.

JAFFA: JAFFA (version 1.06) uses three different modes for fusion gene prediction. Assembly mode is used for paired-end reads with 50 bps (involves assembly). Direct mode is used for reads with longer length (>100 bps) because the read is long enough for direct mapping. Hybrid mode is used for reads with length of around 70 bps. Both assembly and mapping will be used in Hybrid mode. We analyzed all the datasets using different modes based on the read length. Similar filters were also applied for the *in-house datasets*.

```
cat jaffa_results.csv | sed -e 's/,/\t/g' |
    awk '$8+$9>=5 && ($7=="Inf" || $7>=100) {print}' |
    grep -v LowConfidence
```

Explaination:

$8+$9>=5: read number no less than 5

$7=="Inf" || $7>=100: on different chromosome or no less than 100k bp

grep –v LowConfidence: filter out all lowConfidence fusion genes

2.3. Statistical measurements

Here we list all the statistical terms we used for accuracy measurements. For real RNA-Seq datasets, because the total number of true fusion genes is unknown, it is not applicable to calculate the sensitivity. Besides, we could not calculate the specificity for datasets like *BC_CL* also, because many predicted fusions have not been validated to be false. Therefore, we only calculate statistics like *TP*, *FP*, and *precision*.

TP: True positive. Positive instances correctly classified as true.

FP: False positive. Negative instances incorrectly classified as true. Also known as Type I error.

TN: True negative. Negative instances correctly classified as false.

FN: False negative. Positive instances incorrectly classified as false. Known as Type II error.

Sensitivity or *recall*: TP/P = TP/(TP+FN). It measures the ability of a method to correctly classify a positive instance. Also known as *true positive rate* (TPR).

Specificity: TN/N=TN/(TN+FP). It measures the ability of a method to correctly classify a negative instance. Equivalent to *true negative rate* (TNR).

Precision: also known as *positive predictive value* (PPV). PPV=TP/(TP+FP). It measures the proportion of the positive results reported by a method that are truly positive.

FDR: *False Discovery Rate*. *FDR=FP/(TP+FP)=1-PPV*. It is used to measure type I errors.

F-measure: Also known as F_1*-score*, or *F-score*. It is the hormonic mean of *precision* and *recall*. $F_1 = 2 \times \frac{precision \times recall}{precision + recall}$. It measures the overall accuracy of a method.

3. Results

Simulated dataset 1 (50_pos_set)

To avoid the analysis bias on other methods, we compared the prediction performance of ChimeRScope against the best reported results (either from this study or from other published studies, if applicable). The performance of the selected methods on *50_pos_set* is shown in Table 3-3. We failed to run JAFFA on *50_pos_set* in this study. However, the statistics reported in this table is directly obtained from JAFFA publication [30] and can be considered as the best prediction results for JAFFA. In summary, ChimeRScope achieves the highest F-score by predicting 47 true positives without reporting any false positives. Even after applying all the filters, ChimeRScope still reported the best results. Here, we also listed the impact of the filters used in FusionCatcher. With all filters enabled, only 3 false positives were removed at the cost of

removing 16 true positives. Moreover, the highest f-score reported from "other methods" group is only 0.83 by FusionMap, which is lower than ChimeRScope and other three major methods.

Table 3-3. Detailed prediction results on *50_pos_set*. This dataset contains in total of 50 simulated fusion genes. Our method, ChimeRScope, predicts 47 true positives without reporting any false positives. ChimeRScope achieves the highest f-score with highest sensitivity (recall = 94%) and lowest FDR (0%). Even with all filters enabled, the performance of ChimeRScope on this dataset is still better than other methods, with a second highest f-score of 0.95.

50_pos_set		Study	TP	FP	Precision	Recall	FDR	F-score	Note	Best score
Major methods	ChimeRScope	This study	47	0	1.00	0.94	0.00	**0.97**	No filtering	**0.97**
		This study	45	0	1.00	0.90	0.00	**0.95**	With all filters*	
	SOAPfuse	This study	38	1	0.97	0.76	0.03	**0.85**		0.85
		PMID:26019724	37	1	0.97	0.74	0.03	0.84		
	JAFFA	PMID:26019724	44	0	1.00	0.88	0.00	**0.94**	JAFFA-hybrid	0.94
		PMID:26019724	39	0	1.00	0.78	0.00	0.88	JAFFA-assembly	
		PMID:26019724	34	0	1.00	0.68	0.00	0.81	JAFFA-direct	
	FusionCatcher	This study	31	0	1.00	0.62	0.00	0.77	Final result	0.94
		This study	47	3	0.94	0.94	0.06	**0.94**	Raw result	
Other methods	FusionMap	PMID:23815381	40	6	0.87	0.80	0.13	**0.83**		0.83
	FusionFinder	PMID:23815381	41	10	0.80	0.82	0.20	**0.81**		0.81
	tophat-fusion	PMID:26019724	27	0	1.00	0.54	0.00	**0.70**		0.70
		PMID:23815381	40	73	0.35	0.80	0.65	0.49		
	MapSplice	PMID:23815381	39	23	0.63	0.78	0.37	**0.70**		0.70
	deFuse	PMID:23815381	32	4	0.89	0.64	0.11	0.74		0.81
		PMID:26019724	34	0	1.00	0.68	0.00	**0.81**		
	FusionHunter	PMID:23815381	20	4	0.83	0.40	0.17	**0.54**		0.54

* Two fusion genes were filtered out by similarity filter (PRKCA&USP49 score: 384; FKTN&SCAI: 367). However, the F-score is still higher than others (0.95 is the second highest score)

Simulated dataset 2 (comp_sim_set)

We also analyzed *comp_sim_set* using four primary methods and compared our results against the original study for this dataset [44]. We carefully checked the gene name alias for all fusion pairs to make sure the reported numbers are the most current ones. We failed to run

JAFFA on datasets with 50bp and 75bp read length due to computational issues caused by JAFFA assembly mode. Nevertheless, considering the results for JAFFA on 100bp datasets as the best results it can get on 50bp and 75bp datasets may not put JAFFA as one of the best tools for *comp_sim_set* (Table 3-4). During the analysis stage, we also carefully inspected all the simulated genes. For *comp_sim_set*, all the fusion genes are simulated using sequences from EnsEMBL database. We aligned the fusion junction sequences against each fusion partner using both EnsEMBL and RefSeq version. We found out that 15 fusion genes could not align to RefSeq version (Table 3-5). Because ChimeRScope GF-library is built only based on RefSeq sequences, it will be impossible for ChimeRScope to report any of these fusion genes. Therefore, we also reported the normalized results by setting the total number of fusion genes to 135. The final results are shown in Table 3-6. ChimeRScope reports the highest F-scores in 13 out of 15 datasets. Notably, ChimeRScope has significantly higher f-scores on datasets with low coverage depth (5X), suggesting better performance on fusion genes with low expression levels. More importantly, the variation of the f-scores reported by ChimeRScope is subtle (max=0.957 and min=0.905) across different datasets irrespective of the read length and coverage depth, indicating a cosistently good performance on all datasets.

Table 3-4 F-scores for selected methods on 15 simulated datasets (*comp_sim_set*). The highest f-scores are highlighted in red for each dataset. ChimeRScope achieves significantly higher f-scores for datasets with low coverages (5X in 50bp, 75bp, and 100bp), suggesting that ChimeRScope is more sensitive to fusion genes with low expression level.

comp sim set (TP=150)	50bp					75bp					100bp				
	5X	20X	50X	100X	200X	5X	20X	50X	100X	200X	5X	20X	50X	100X	200X
ChimeRScope	**0.898**	0.905	0.899	0.864	0.861	**0.898**	0.900	0.908	0.906	0.900	**0.890**	0.907	0.908	0.909	0.909
SOAPfuse	0.830	0.890	0.863	0.871	0.858	0.833	0.883	0.873	0.871	0.874	0.844	0.895	0.887	0.883	0.887
SOAPfuse*	0.801	**0.923**	**0.938**	**0.918**	**0.945**	0.828	**0.925**	**0.925**	**0.932**	**0.932**	0.840	**0.940**	**0.932**	**0.939**	**0.935**
FusionCatcher	0.565	0.746	0.762	0.757	0.766	0.675	0.745	0.779	0.792	0.792	0.741	0.785	0.758	0.777	0.781
FusionCatcher*	0.337	0.842	0.886	0.890	0.894	0.687	0.850	0.875	0.884	0.891	0.678	0.873	0.887	0.891	0.891
JAFFA	0.052	0.605	0.667	0.678	0.678	0.101	0.461	0.597	0.619	0.616	0.441	0.675	0.698	0.693	0.693
JAFFA*	-	-	-	-	-	-	-	-	-	-	0.579	0.857	0.867	0.867	0.867
EricScript	0.454	0.680	0.669	0.519	0.612	0.568	0.759	0.763	0.695	0.773	0.593	0.782	0.793	0.779	0.788
chimerascan	0.672	0.743	0.744	0.730	0.724	0.631	0.751	0.821	0.744	0.752	0.565	0.745	0.746	0.730	0.737
PRADA	0.077	0.512	0.534	0.534	0.534	0.101	0.505	0.545	0.543	0.543	0.204	0.498	0.538	0.545	0.543
deFuse	0.284	0.551	0.634	0.588	0.750	0.281	0.608	0.566	0.741	0.744	0.192	0.611	0.502	0.629	0.779
FusionMap	0.039	0.485	0.611	0.619	0.623	0.343	0.535	0.691	0.627	0.650	0.343	0.635	0.691	0.684	0.658
TopHat-Fusion	0.165	0.406	0.457	0.453	0.460	0.295	0.417	0.498	0.491	0.509	0.312	0.444	0.488	0.488	0.507
MapSplice	0.144	0.414	0.443	0.493	0.514	0.249	0.426	0.468	0.495	0.512	0.241	0.406	0.416	0.488	0.486
BreakFusion	0.741	0.756	0.743	0.739	0.735	0.624	0.729	0.730	0.730	0.731	0.408	0.619	0.692	0.707	0.704
SnowShoes-FTD	0.039	0.039	0.039	0.039	0.039	0.026	0.026	0.039	0.039	0.039	0.026	0.039	0.039	0.039	0.039
FusionQ	0.428	0.542	0.560	0.415	0.176	0.462	0.579	0.642	0.558	0.197	0.471	0.495	0.552	0.651	0.443
ShortFuse	-	0.780	0.777	0.776	0.785	0.587	0.748	0.775	0.783	-	-	-	-	-	-

* Marked rows are the results reported by our study. Stats for other rows are reported from this study [44].

Table 3-5 15 fusion genes simulated from EnsEMBL sequences with incompatible partners. If a fusion gene is simulated from a region where RefSeq defined that as a non-coding region while EnsEMBL does the contrary, we named the corresponding fusion partner as an "incompatible partner". *Comp_sim_set* contains 150 fusion genes simulated from EnsEMBL sequences. 13 of the fusion genes are found as fusion genes with incompatible partners. The other two genes, STAG3L1 and KIAA1875, were defined as genes with no protein products in the latest RefSeq build when we performed the study. Therefore, we classified these two genes as incompatible partners as well.

Index	Gene1	Gene2	Gene1_ensID	Gene2_ensID	Incompatible partner
1	AFTPH	HOMER1	ENSG00000119844	ENSG00000152413	HOMER1
2	C6orf130	ASPH	ENSG00000124596	ENSG00000198363	ASPH
3	CAPN2	TMEM223	ENSG00000162909	ENSG00000168569	TMEM223
4	CCDC88C	STAG3L1	ENSG00000015133	ENSG00000205583	STAG3L1[a]
5	FAM92A1	HNRNPL	ENSG00000188343	ENSG00000104824	FAM92A1
6	FGFR4	NREP	ENSG00000160867	ENSG00000134986	NREP
7	GPR128	MLLT6	ENSG00000144820	ENSG00000108292	GPR128
8	IPO5	MS4A6A	ENSG00000065150	ENSG00000110077	IPO5
9	KIAA1328	EXOSC7	ENSG00000150477	ENSG00000075914	EXOSC7
10	MAPK10	KIAA1875	ENSG00000109339	ENSG00000179698	KIAA1875[b]
11	MYH10	GPR155	ENSG00000133026	ENSG00000163328	MYH10
12	SFTPA2	THOC2	ENSG00000185303	ENSG00000125676	THOC2
13	SLC19A1	LLPH	ENSG00000173638	ENSG00000139233	SLC19A1
14	TBX22	RBM48	ENSG00000122145	ENSG00000127993	RBM48
15	USP45	COBL	ENSG00000123552	ENSG00000106078	USP45

[a] STAG3L1 is a pseudogene in the current RefSeq build. We believe fusion genes involving pseudogenes are functionally not important and are excluded in the GF-library.

[b] KIAA1875 was considered as a non-coding gene (updated in Mar.15th, 2015) and was later replaced with WDR97 (a coding gene). However, it is not included in the latest RefSeq database when we performed the analysis.

Table 3-6. F-scores for 4 major methods on *comp_sim_set*, with 15 incompatible fusion genes removed. The results are calculated from 135 effective fusion genes. ChimerRScope achieves the highest F-scores in 13 out of 15 datasets. Only SOAPfuse reports higher f-scores in 100X_50bp and 200X_50bp, with marginal increases of 0.001 and 0.034 in F-scores, respectively.

comp sim set (TP=135)	50bp					75bp					100bp				
	5X	20X	50X	100X	200X	5X	20X	50X	100X	200X	5X	20X	50X	100X	200X
ChimeRScope	**0.948**	**0.954**	**0.947**	0.908	0.905	**0.948**	**0.949**	**0.957**	**0.954**	**0.947**	**0.940**	**0.957**	**0.957**	**0.957**	**0.957**
SOAPfuse	0.807	0.922	0.935	**0.913**	**0.943**	0.836	0.921	0.921	0.929	0.925	0.840	0.942	0.928	0.932	0.929
FusionCatcher	0.357	0.856	0.894	0.895	0.899	0.692	0.855	0.878	0.888	0.896	0.707	0.884	0.891	0.891	0.891
JAFFA	-	-	-	-	-	-	-	-	-	-	0.609	0.849	0.856	0.856	0.856

Cancer RNA-Seq datasets (breast cancer cell lines)

We analyzed the 4 different breast cancer cell lines (*BC_CL*) using ChimeRScope, SOAPfuse, JAFFA, and FusionCatcher. We also compared our results against this study [44]. Results (Table 3-5) have shown that ChimeRScope achieves the best performance with successfully identifying 22 true fusion events. ChimeRScope also predicts the highest number of fusion genes in all 4 cell lines. Besides, among all predicted fusion genes from ChimeRScope results, another 8 fusion genes are also validated by other studies [43, 48, 49]. However, we only compare the true fusion genes reported from the original study [19] for consistency of the comparison in Table 3-6.

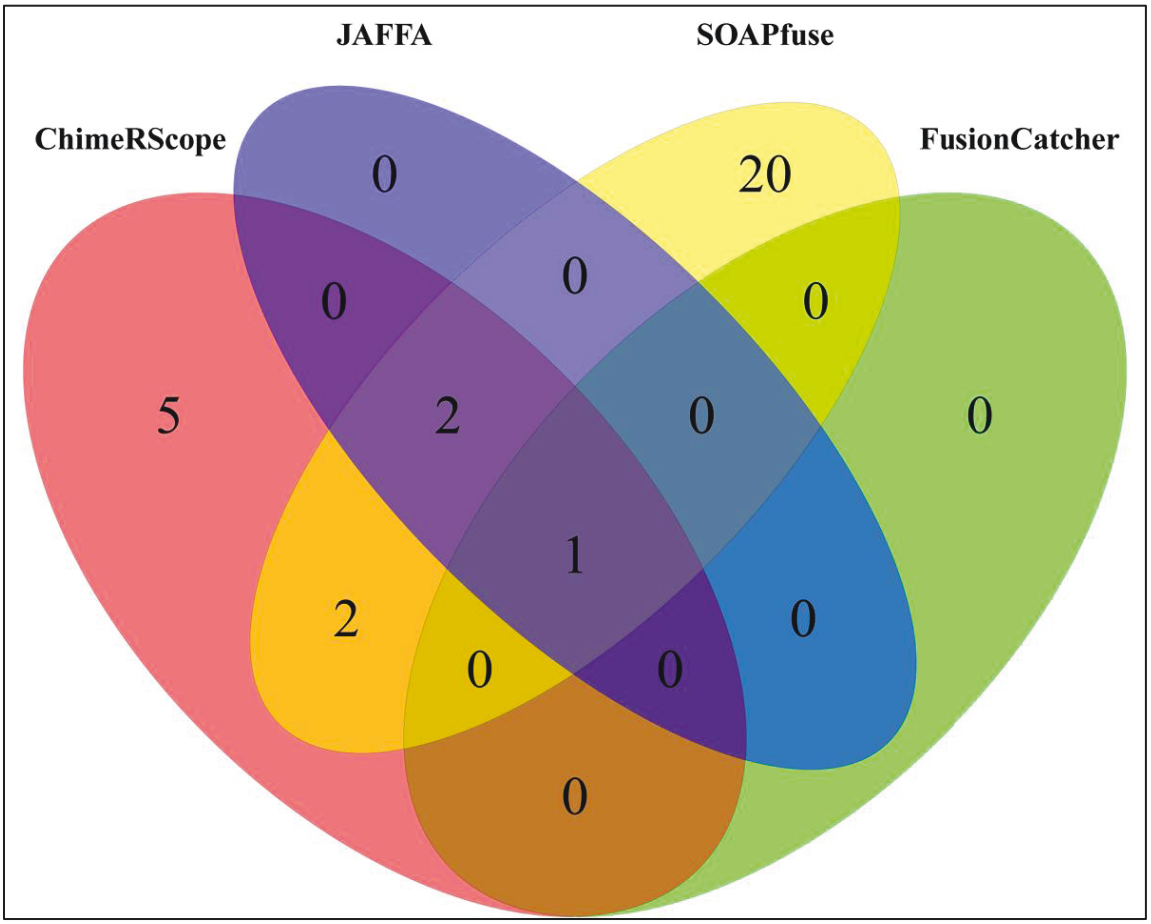
Table 3-7. Predictions on breast cancer cell lines for selected methods. The numbers in parenthesis next to the cell line names are the total number of validated fusion genes in each cell line. ChimeRScope reports the highest number of fusion genes in all four cell lines. Overall, ChimeRScope reports the highest number of true positives by identifying 22 true fusion genes out of 26 validated fusion genes.

BC_CL	BT474(11)		SKBR3(10)		MCF7(3)		KPL4(3)		Total TP	Total Prediction
	TP	Total Prediction	TP	Total Prediction	TP	Total Prediction	TP	Total Prediction		
ChimeRScope*	**10**	**24**	**6**	**24**	**3**	**14**	**3**	**4**	**22**	**66**
SOAPfuse	9	35	6	19	2	6	3	8	**20**	**68**
SOAPfuse*	7	26	6	11	3	7	3	4	**19**	**48**
fusionCatcher	9	31	6	24	2	7	2	5	**19**	**67**
fusionCatcher*	5	13	5	7	3	7	3	3	**16**	**30**
JAFFA	8	15	5	9	2	6	2	2	**17**	**32**
JAFFA*	7	17	4	9	3	10	3	5	**17**	**41**
EricScript	8	31	4	37	2	10	2	5	**16**	**83**
chimerascan	9	50	5	33	2	22	3	10	**19**	**115**
PRADA	7	23	4	7	2	4	2	3	**15**	**37**
deFuse	9	57	6	50	2	16	2	12	**19**	**135**
FusionMap	4	76	0	0	2	50	0	6	**6**	**132**
TopHat-Fusion	9	28	3	31	1	9	2	5	**15**	**73**
MapSplice	8	27	4	15	2	6	2	5	**16**	**53**
BreakFusion	8	636	4	676	2	387	1	239	**15**	**1938**
SnowShoes-FTD	8	12	4	5	2	2	1	1	**15**	**20**
FusionQ	-	-	-	-	2	199	2	258	**4**	**457**
ShortFuse	8	19	7	15	1	5	3	4	**19**	**43**

* Marked rows are the results reported by our study. Stats for other rows are reported from a previous study [44].

In-house cancer RNA-Seq datasets (lymphoma cell lines) with experiment validation

Figure 3-1. Venn diagram of all fusion genes reported by the four tested methods on lymphoma cell lines. In total, 30 unique fusion genes were reported. ChimeRScope and SOAPfuse predicted the most number of common fusion genes (five genes), with two of them exclusively reported by these two methods. The results reported by FusionCatcher have the lowest overlaps with other methods (only one gene).



We used paired-end RNA-Seq datasets from three natural killer (NK) lymphoma cell lines that we have published earlier [39]. We analyzed these RNA-Seq datasets using ChimeRScope, SOAPfuse, FusionCatcher and JAFFA. In total, ChimeRScope predicted 10 unique fusion genes, compared with 25 unique fusion genes by SOAPfuse, three by JAFFA and only one by

FusionCatcher (Figure 3-1). Among all tested methods, 30 unique fusion genes were predicted, including five that were reported by at least two of the four methods. Further analysis on fusion genes predicted by SOAPfuse has shown that *BOLA2B&SMG1P2*, *TVP23C&CDRT4*, and *DSCR4&DSCR4-IT1* are directly associated with well-annotated read-through mRNAs (with *NM* IDs in NCBI Reference Sequence database). This type of predictions is classified as false fusion event by ChimeRScope and hence is filtered out. In addition, we failed to design primers for *ORC6&PLEKG4B* and *MAPK8&NMU* due to low complexity regions. Therefore, these five fusion genes predicted by SOAPfuse are excluded from our experimental validation.

Table 3-8. Eleven fusion genes selected for experimental validation on NK cell lines.

Cell line	Fusion Gene	Fusion read counts*			
		ChimeRScope	JAFFA	FusionCatcher	SOAPfuse
KHYG1	*PEX2&YWHAZ*	34	22	(3)	49
KHYG1	*ARIH2&PRKAR2A*	(3)	(2)	(2)	5
KHYG1	*CTSC&RAB38*	(2)	(0)	(5)	10
KHYG1	*PRKCH&FLJ22447*	(0)	(0)	(1)	6
NKYS	*LRRFIP1&RBM44*	10	(0)	(64)	192
NKYS	*RPL14&SRP14*	7	(0)	(0)	(0)
NK92	*C15orf57&CBX3*	6	(6)	(1)	10
NK92	*DAB2&FRYL*	48	33	30	59
NK92	*LEP&SND1*	92	51	(60)	121
NK92	*LRRC37A3&NSF*	5	(0)	(0)	(4)
NK92	*MAST2&METTL21A*	8	(0)	(0)	(0)
NK92	*NCOR2&UBC*	5	(3)	(5)	(4)
NK92	*NPIPB5&SMG1*	15	(0)	(0)	(0)
NK92	*PTMA&NPM1*	(9)	(0)	(0)	13

Tool	Unique prediction	Test size	TP	FP	Sensitivity	FDR	F-score
ChimeRScope	10	10	10	0	0.714	0	0.833
SOAPfuse**	25	20	9	11	0.643	0.55	0.529
JAFFA	3	3	3	0	0.214	0	0.353
FusionCatcher	1	1	1	0	0.071	0	0.133

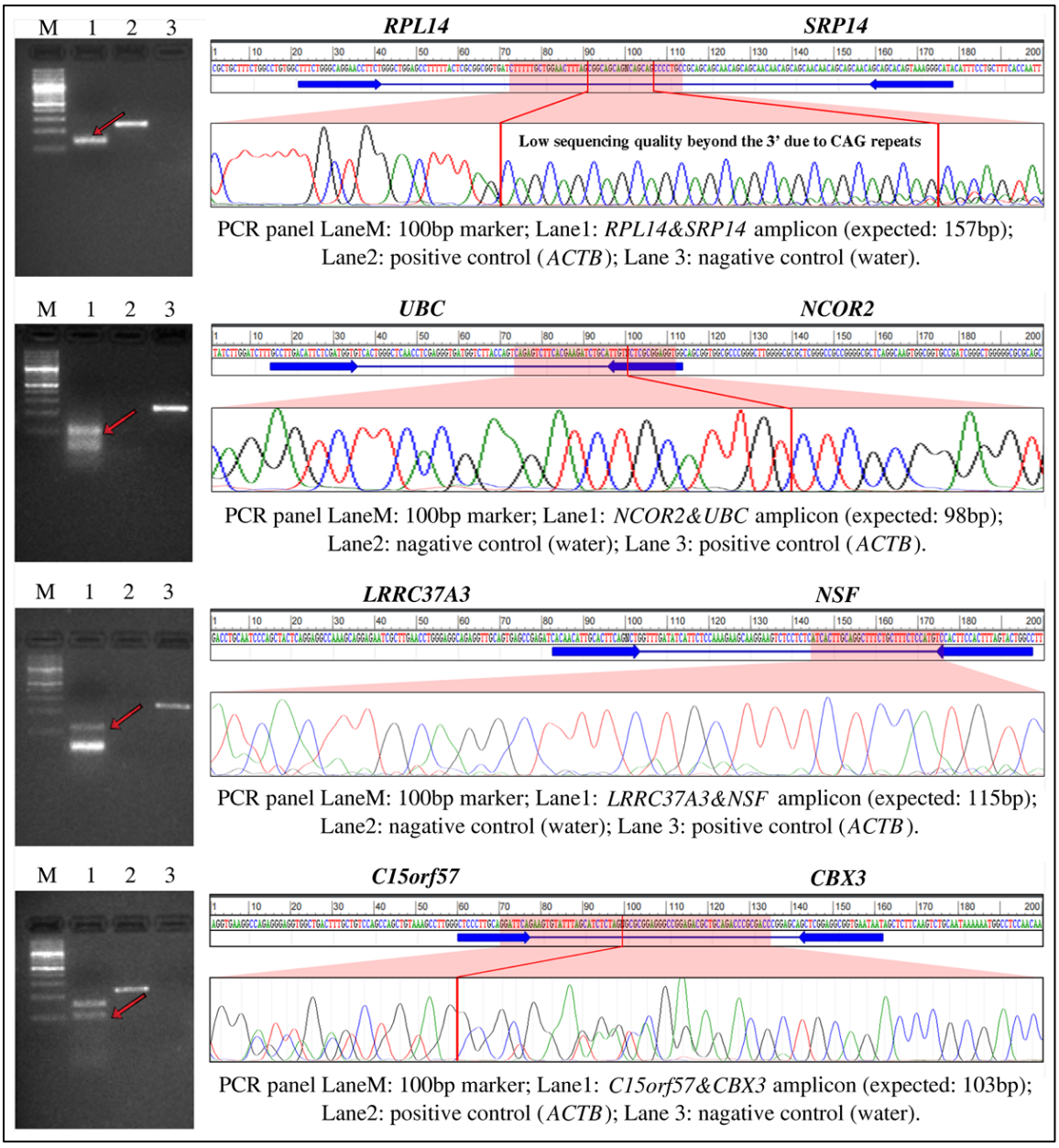
* The number of identified fusion reads for each fusion gene identified by each method is listed in the corresponding cells. Cells with parenthesis indicate that the fusion genes were filtered out by the corresponding tools and thus not reported in their final results.

** Five fusion genes predicted by SOAPfuse were excluded from the validation list because either the complete fusion sequence were associated with well annotated read-through mRNAs or the specific primer binding sites were not available due to repeated nucleotide sequences.

Of all the fusions predicted by different algorithms, we designed primers for 25 unique fusion genes and confirmed 14 fusion genes (56%; 14 out of 25 tested fusions) by RT-PCR and Sanger sequencing (Table 3-8). Due to space limitations, we chose to show validation results from only four fusion genes that are predicted by ChimeRScope with relatively lower number of fusion reads (five to seven FESRs, see Table 3-8). Figure 3-2 highlights the PCR results, primer target regions, Sanger sequencing chromatograms, and the exact fusion junctions marked by red lines (except for *RPL14&SRP14* and *LRRC37A3&NSF*) for these four fusion genes. In total, ChimeRScope predicted 10 fusion genes from the lymphoma cell lines and we were able to experimentally validate all of these predictions. Thus, there are no false positives predicted by our method (FDR: 0%). However, our method missed four true positives that were predicted by other methods, hence the sensitivity of this method is at 71.4% (10 out of 14), which is the highest among all methods tested. Comparatively, among the 20 tested fusion genes from SOAPfuse predictions, only nine fusion genes were experimentally confirmed (sensitivity is 64.3% and FDR is 55%). All the fusion genes reported by JAFFA (three fusions) and FusionCatcher (one fusion) are also predicted by ChimeRScope and SOAPfuse. Therefore, both JAFFA and FusionCatcher achieve 100% precision rate, but with only 21.4% and 7.1% sensitivities, respectively. Overall, ChimeRScope reported the best F-score (0.833) for this dataset, compared with 0.529 for SOAPfuse, 0.353 for JAFFA and 0.133 for FusionCatcher (Table 3-8).

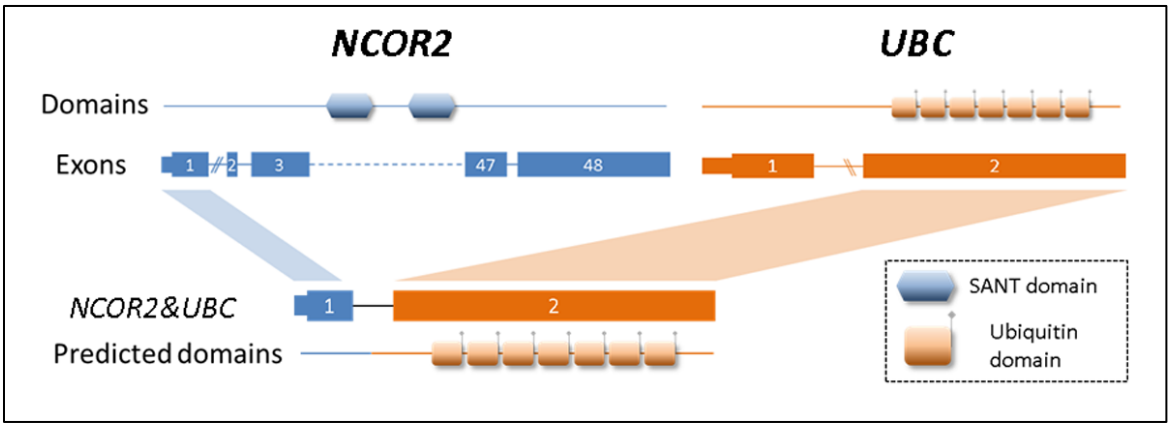
Figure 3-2. PCR and Sanger sequencing results for four fusion genes with low number of FESRs from ChimeRScope predictions. For each fusion gene track, the left panel is the PCR panel and the right panel displays the predicted fusion sequence and the primer binding site, along with the Sanger sequencing chromatogram. Specifically, each PCR image has four lanes for a 100bp ladder marker, the fusion gene amplicon with the band of the matched product pointed by a red arrow, the positive control (actin beta, or *ACTB*) and negative control (water). The right panel shows the name of the fusion partners, the predicted fusion junction sequence (100bp upstream and downstream,

separated by the wildcard "N"), the binding sites of the primer pair used in the PCR panel, the chromatogram for the highlighted region (mostly the fusion junction, if applicable). The PCR experiments and the Sanger sequencing results confirmed the existences of these four genes in the NK cell lines. We were unable to resolve the fusion junctions for *RPL14&SRP14* and *LRRC37A3&NSF* due to the poor Sanger sequencing data quality. Therefore, the exact fusion junctions of these two fusions were not marked in the chromatograms.



Literature searches for all experimentally validated fusion genes have suggested that some of the chimeric transcripts are potentially oncogenic. For instance, a fusion gene that is only predicted by ChimeRScope, *NCOR2&UBC* (Table 3-8), has also been reported previously in CLL patients [40]. *NCOR2* is a nuclear receptor corepressor that interacts with members of MAPK-signaling [41], Notch and NF-kappa-B pathways [42]. The altered expression of this gene is associated with cell cycle progression and apoptosis in multiple cancers [43, 44]. Figure 3-3 illustrates the fusion model of *NCOR2&UBC* with the predicted [45] functional domains in the resulting chimeric protein. This chimeric transcript combines the first exon of *NCOR2* and the second exon of *UBC*, creating a new transcript with the loss of the SANT (named after switching-defective protein 3 or *SWI3*, adaptor 2 or *ADA2*, nuclear receptor co-repressor or *N-CoR*, transcription factor IIIB or *TFIIIB*) domain that is responsible for chromatin-remodeling and transcription regulation [46, 47]. Another validated fusion gene, *LRRC37A3&NSF* is predicted only by ChimeRScope method (Table 3-8). This fusion involves a gene named N-ethylmaleimide sensitive factor (*NSF*). Studies have shown that *NSF* directly interacts with *CD28* [48], a gene responsible for T-cell activation and survival. Although triggering of human NK cells by *CD80* and *CD86* (ligands of *CD28*) seems to be independent of *CD28* [49], the absence of *CD28* expressions in NK cell lines [49] could be the result of the *LRRC37A3&NSF* fusion event. Other exclusive ChimeRScope's predictions like *MAST2&METTL21A* and *NPIPB5&SMG1* are kinase fusions [50] that are more likely to have oncogenic functions in cancer because they involve kinases like *MAST2* (microtubule associated serine/threonine kinase 2) and *SMG1* (nonsense mediated mRNA decay associated PI3K related kinase). Lastly, the only fusion gene predicted by all four methods, *DAB2&FRYL*, includes a potential tumor suppressor gene named *DAB2* (Disabled homolog 2) which has been found to be associated with tumorigenesis in different cancers [51-53]. These fusion genes mentioned above warrants further investigation to confirm their specific roles in tumorigenesis.

Figure 3-3. The fusion model of *NCOR2&UBC* and the predicted functional domains. This fusion gene is fused between the first exon of *NCOR2* (3' end) and the second exon of *UBC* (5' end). The SANT domain from *NCOR2* and the ubiquitin domains from *UBC* are plotted to the approximate position of the corresponding exons. Because the exons with the SANT motif sequence are not included in *NCOR2&UBC*, the predicted domains of the *NCOR2&UBC* fusion gene only contain the ubiquitin domains from *UBC*.



4. Discussion and conclusions

We tested the predictive power of ChimeRScope against SOAPfuse, JAFFA, and FusionCatcher on simulated datasets and cancer datasets. Comparison results have clearly demonstrated that ChimeRScope has better performance on all tested simulated datasets and real RNA-Seq datasets. Notably, results on simulated datasets have shown that alignment-based methods rely largely on the technical aspects like read length and sequencing depth of the RNA-Seq datasets, whereas alignment-free methods like ChimeRScope are less likely to be affected by these factors because it gives constantly superior F-scores across all datasets with different read length and coverage depth.

For analysis on real RNA-Seq datasets, a BLAST search of the fusion sequences is recommended for removing false positives and biologically insignificant fusion genes. For instance, the fusion gene reported in the original SOAPfuse paper, GATSL1-GTF2I, can be aligned to several non-coding RNAs (NR_002206.3 and NR_003580.2). ChimeRScope also

found a similar fusion gene named GATSL2-GTF2I in the breast cancer samples. This class of fusion genes should be excluded in the downstream analysis because it should not be considered as a fusion event.

We have successfully amplified all the 10 predicted fusion products with the exact amplicon size. We also confirmed the exact fusion junctions from the Sanger sequencing for eight out of 10 predictions, but unable to do so for two of the fusion genes (*RPL14&SRP14* and *LRRC37A3&NSF*) that have faint PCR bands (Figure 3-2). However, these two fusions were still considered as true fusion events based on the specific amplicon size and the alignment evidences between the predicted fusion sequences and the Sanger sequencing results. Specifically, the fusion junction of the *RPL14&SRP14* fusion contains CAG repeats, which could also affect the Sanger sequencing quality near the 3' end of the repeat region. We were unable to resolve the fusion junction due to the CAG repeats, thus the exact fusion junction is not marked for this fusion in Figure 3-2. For *LRRC37A3&NSF*, we were only able to design the primer pair with the forward primer spanning the fusion junction (Figure 3-2). Therefore, the Sanger sequencing result generated from the forward primer does not span the fusion junction. Due to the poor Sanger sequencing quality observed in the first 15 to 40 bases, the chromatogram of *LRRC37A3&NSF* (Figure 3-2) only shows the comparison between the predicted fusion sequence roughly 40bp downstream of the forward primer binding site and the Sanger sequencing result. We were not able to obtain high quality Sanger sequencing result from the reverse primer, thus the Sanger sequencing result that covers the fusion junction was not available for this fusion gene. Nevertheless, the PCR result shows the band with the exact amplicon size. Additionally, the forward and the reverse primer are very specific to *LRRC37A3* and *NSF*, respectively. Since the Sanger sequencing result shows significant match with the 3' gene (*NSF*), we believe that this fusion gene is also a true fusion event.

Among four of the validated fusion genes that are missed by ChimeRScope, ChimeRScope still reports FESRs for three of those fusions, but filters them out due to the stringent filters it uses to remove false positives. For example, the preliminary result from ChimeRScope *Scanner* shows that ChimeRScope identified nine FESRs for *PTMA&NPM1*. However, some of these FESRs have very low weightage score and was not considered as valid FESRs due to insufficient fingerprint sequences (possibly caused by evenly distributed sequence variations). Allowing a couple of mismatches when comparing the *k*-mers could potentially improve the sensitivity of our method to detect such fusion genes.

Here we conclude that, ChimeRScope has superior advantages over other popular tools for fusion gene prediction based on all tested datasets. Also, ChimeRScope relies less on the technical aspect of the datasets (read length, sequencing depth, etc.) or the expression levels of fusion genes, comparing to other alignment-based methods. This also suggests that the alignment-free methods, in addition to providing alternative ways of analyzing NGS data, can also address the issues that we often encounter using alignment-based methods.

Chapter 4: ADDITIONAL TOOLS TO ADD THE ANALYSIS (LOCAL GALAXY SERVER)

1. Overview

Many of the bioinformatics tools for biomedical research only run on linux-based servers using command lines. They also have other tool dependencies, some of which are lacking proper update procedures and installation documentations. For example, fusionCatcher [29] has over 15 different dependencies. Installing some of these dependencies also require root privileges, which are not always feasible for end users. Installation of python modules also involves changing environmental variables like python library path. Therefore, it can be extremely hard for researchers with limited programming and system administrative expertize to install and use these tools.

To facilitate the usage of ChimeRScope for different user groups, we also implemented an online Graphical User Interface (GUI) data analysis server for ChimeRScope using the Galaxy Server platform. Researchers can submit the data analysis jobs to this server online using a web browser and all the parameters can be set using check-boxes and dropdown menus. Below, we discuss the GUI of ChimeRScope in detail.

2. Introduction to Galaxy server

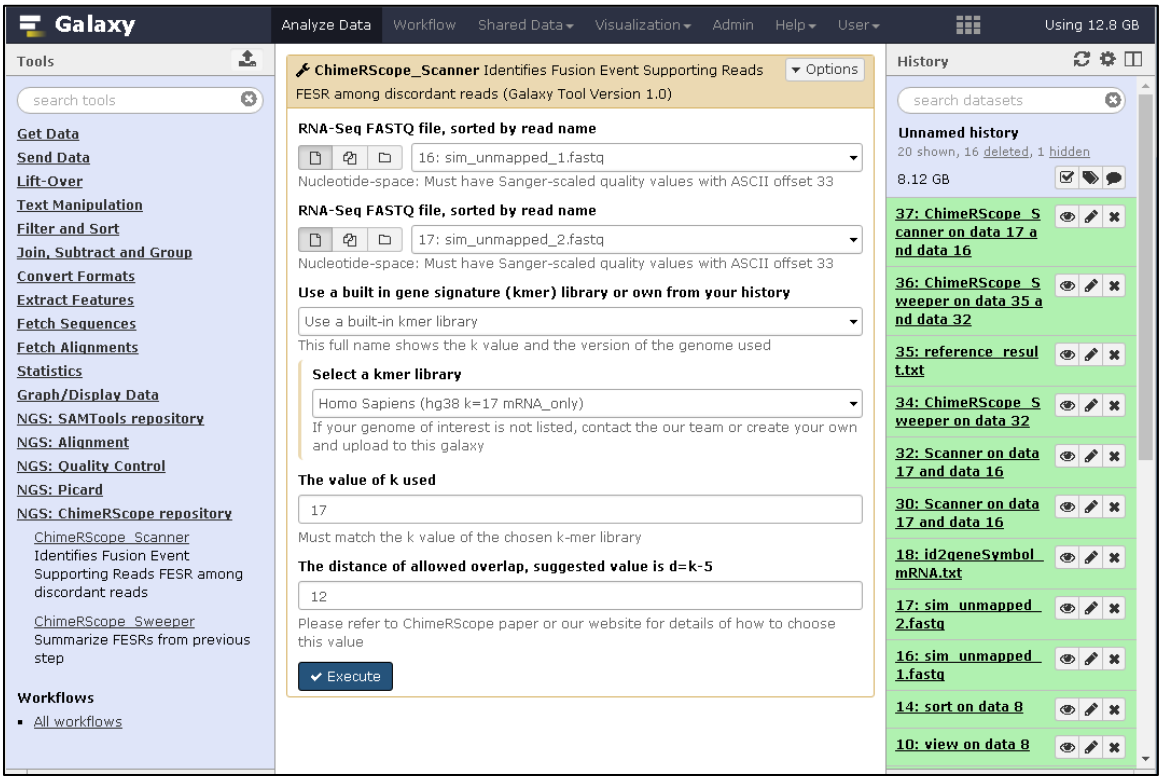
Galaxy server [67] is an open, web-based platform with large collection of bioinformatics tools installed, and accessible to the research community for local installation. It is initially developed by the Nebrutenko lab in the Center of Comparative Genomics and bioinformatics at Penn State and the Taylor lab at Emory University. Galaxy server provides Graphic User Interface (GUI) support for installed bioinformatics tools, thereby making data analysis more accessible for users with limited programming experience. Moreover, Galaxy server also includes

a workflow system. Users can easily construct the workflows from all tools in the Galaxy server. This makes the analysis in the Galaxy server more reproducible. Apart from that, Galaxy server is also transparent. Users can share their workflow and analysis histories with other colleagues.

The main Galaxy server (https://usegalaxy.org/) currently is built at the Texas Advanced Computing Center, with support from the National Science Foundation. It is a free, public resource with no encryption on the data. Therefore, it is suggested to build a local Galaxy instance for local usage. The Galaxy team provides detailed instructions on how to build a local Galaxy server and install customized tools. We followed these instructions and successfully build a local Galaxy server.

Figure 4-1 shows a screenshot of our local Galaxy server at UNMC. The left panel lists all the installed tools, the central panel displays the data analysis parameters for the currently selected tool and the right panel displays the history of the workflow. When a tool is selected (e.g., ChimeRScope_Scanner in Figure 4-1), menus for all the input parameters are listed in the central display panel. As we mentioned earlier, ChimeRScope_Scanner takes two fastq files (discordant paired-end reads), GF-library, and several other parameters as input, and outputs a list of Fusion Event Supporting Reads. In the example shown in Figure 4-1, we have uploaded two fastq files, namely *sim_unmapped_1.fastq* and *sim_unmapped_2.fastq*. We have also added several built-in GF-libraries in our Galaxy server. However, users can also upload their own *k*-mer libraries into the Galaxy server using the uploading application. We have set all recommended parameters as the default values, however, users have the ability to alter the input parameters and customize their jobs. Once a job is submitted to the Galaxy server, selected output files will be listed in the history panel on the right. Users can view these files by clicking the view bottom (first bottom with the “eye” symbol). If a job fails, the error log will also be listed in the history panel. Users can also use the workflow system to automate the analysis pipeline. The workflows and analysis results can be easily accessed by other authorized users using the “Share Data” application.

Figure 4-1. The layout of our local Galaxy server at UNMC. All the installed tools are grouped and listed in the left panel. The history panel on the right part lists all the datasets involved in the analysis, including the input files, intermediate files, and output files. The central panel in this example displays all the parameters used for running the ChimeRScope (see Chapter 3, subsection 2.2).



3. Local Galaxy server installation

We installed the Galaxy instance on our local bioinformatics server (named *metastasis*) at UNMC. The metastasis server has 48 cores and 128 GB memory with CentOS Linux version 7.2.1511, which provides sufficient computational power for the users. We create a separate administrator account named *galaxy* for our Galaxy server. All the following steps are processed under the user *galaxy*, unless mentioned specifically. We changed the Galaxy home directory to

the root directory of the Galaxy repository to maintain consistency for the file paths if we want to migrate the whole Galaxy directory to other servers.

Galaxy server requires python version 2.6 or 2.7. We installed the stable python version 2.7.5 on *metastasis*. To protect the integrity of the Galaxy environment, we also used the '*virtualenv'* tool to create an isolated Python environment solely for our Galaxy server. We downloaded the latest Galaxy source code from *https://github.com/galaxyproject/galaxy/* and installed it on the *metastasis* server. We used *nginx* for the proxy server of our Galaxy instance, as suggested by the Galaxy group for server security issue. Galaxy server by default uses SQLite for quick development. Unfortunately, SQLite does not handle concurrency. To make our Galaxy server more efficient, we replaced SQLite with PostgreSQL for better data and job handling.

4. Galaxy server structure

Galaxy server is a sophisticated system that offers a variety of functions. It is crucial to understand the structure of the Galaxy server if we want to customize our own Galaxy server. Here we list all the essential directories for our Galaxy server and highlight the main purposes of these directories (Table 4-1).

Table 4-1. Key directories and the main purposes of the directories for the Galaxy server.

Galaxy server path	**Purpose**
~/galaxy-dist/database/files/	Stores all input/output datasets
~/galaxy-dist/referenceFiles/	Stores all reference files, including *k*-mer libraries.
~/galaxy-dist/tool-data/	Configuration files for all registered tools and built-in reference data. (text files)
~/galaxy-dist/scripts/	System scripts for Galaxy server (python files).
~/galaxy-dist/tools/	Default path for installed tools and the corresponding python wrapper scripts
~/shed_tools/	Path for customized tools in the local Galaxy server
~/galaxy-dist/config/	Galaxy server configuration directory. Including server configuration, data types, etc. (xml files)

Galaxy server is mainly implemented in python and XML. For a customized tool to be usable in a Galaxy server, developers also need to implement specific wrapper script for that tool. XML stands for eXtensible Markup Language and was designed to be both human- and machine-readable. The XML wrapper script redirects the input files and output files to relative file paths. All the parameters set by users on the Galaxy server page will also be parsed to the selected tool.

Here we show an example of how to install a customized tool in the Galaxy server. We registered *ChimerScope* by adding a tool section called *ngs:_ChimeRScope* under the *toolbox* section in the tool shed configuration file (*shed_tool_conf.xml*).

~/galaxy-dist/config/shed_tool_conf.xml

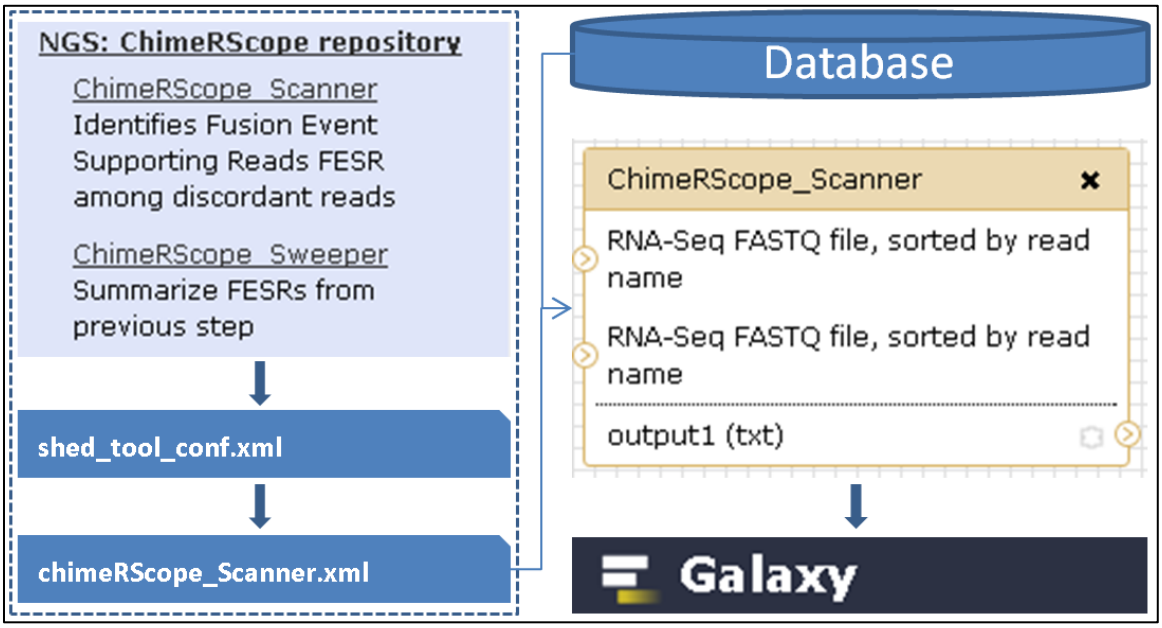
```
<toolbox tool_path="~/shed_tools>
    ...
    <section id="ngs:_ChimeRScope" name="NGS: ChimeRScope repository" version="1.0">
      <tool file="localTools/chimeRScope/chimeRScope_Scanner.xml" />
      <tool file="localTools/chimeRScope/chimeRScope_Sweeper.xml" />
    </section>
</toolbox>
```

ChimeRScope repository is registered under *~/shed_tools/localTools/chimeRScope/*. A simple screenshot of the ChimeRScope wrapper script is shown below.

```
~/shed_tools/localTools/chimeRScope/chimeRScope_Scanner.xml

<tool id="ChimeRScope_Scanner" name="ChimeRScope_Scanner" version="1.0">
    <requirements>
        <requirement type="package" version="1.0">ChimeRScope</requirement>
    </requirements>

    <description>
        Identifies Fusion Event Supporting Reads FESR among discordant reads
    </description>
    <command>
        #set kmer_lib_path = ''
        #if $klibSource.selectklibSource == "buildin":
            #set kmer_lib_path = $klibSource.prebuild.fields.path
        #else:
            #set kmer_lib_path = $klibSource.ownFile
        #end if

        java -Xmx20g -jar ~/shed_tools/localTools/chimeRScope/ChimeRScope_Scanner.jar
            -galaxy
            -fq1 "${input1}"
            -fq2 "${input2}"
            -k ${k_value}
            -d ${d_value}
            -lib ${kmer_lib_path}
            -t 10
            -o chimeRScope_out

    </command>
    <inputs>
        ...
        <param name="k_value" type="integer" value="17" />
        <param name="d_value" type="integer" value="12" />
    </inputs>
    <outputs>
        <data name="output1" format="txt" from_work_dir="chimeRScope_out/FESRs.txt"/>
    </outputs>
</tool>
```

The command for running *ChimeRScope_Scanner.jar* is shown in the *command* section of the XML file. The variables used in the command line are defined in the *inputs* section. We also registered the latest GF-libraries for human reference build GRCh38/hg38 in the Galaxy server database. Users can choose the built-in GF-libraries by selecting the right GF-library in the dropdown menu from the webpage, or select GF-library files from the history datasets (defined as *kmer_lib_path* in the *command* section). Once a job is submitted by the user, this java task will be run on the local server. We only wrapped the most important file, namely *FESRs.txt*, for the downstream analysis (defined in the *outputs* section). Direct interactions between the Galaxy webserver and the database are shown in Figure 4-2.

Figure 4-2 Interactions of the local Galaxy server/database using ChimeRScope_Scanner as an example. Tools are installed via XML wrapper scripts. A data analysis job is submitted by users from the Galaxy webserver. Input files and variables submitted through the website are parsed into command lines to the data analysis server. Once the job is done, the output will be sent back to the Galaxy website for users.



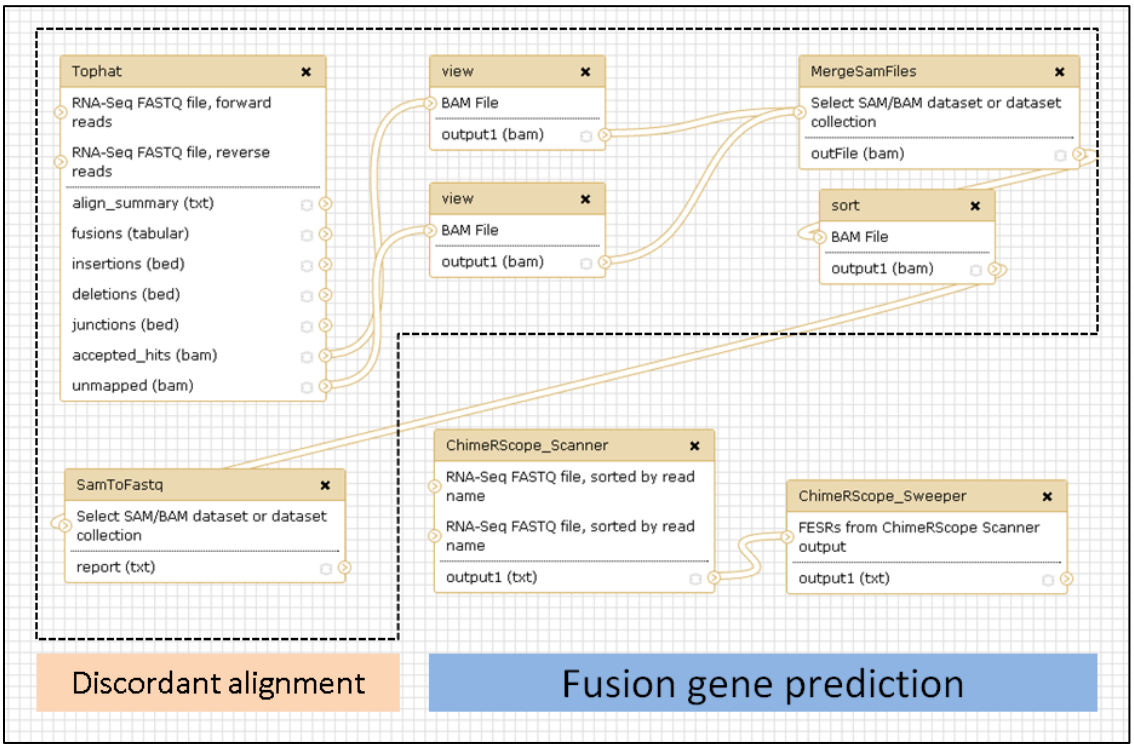
5. Workflows in Galaxy server

Recent advances in the sequencing technologies make it affordable to sequence hundreds or thousands of samples in a single research project. Consequently, these advances also posed a big challenge for the handling and analysis of these massive datasets. Tools designed for bioinformatics data analysis can take anywhere from minutes to days to finish in a single run. It is recommended to automate the whole data analysis pipeline, in order to maximize the computational power of the data analysis server and to prevent idle time when a downstream job is not submitted immediately after the upstream job is finished. Moreover, the data analysis in the same study often follows the same pipeline for the same data type. Analyzing and monitoring hundreds of data analysis jobs at the same time can be a tedious task that requires lots of human attention. Bioinformaticians often write a separate wrapping script using scripting languages (e.g.,

bash, perl, and python) to automate the pipeline, which requires advanced knowledge on programming, system I/O, etc. Alternatively, the Galaxy server platform helps to develop automated data analysis pipelines for routine tasks to enable users to run hundreds of jobs with minimal human supervision.

Using the Galaxy server, users can drag and drop the required tools in the workflow editor. Then, the workflow can be constructed by linking the output ports of the upstream tools to the input ports of the downstream tools. Figure 4-3 shows how the ChimeRScope workflow looks like in our local Galaxy server. Users can modify all the parameters and reference datasets before the analysis starts. This workflow can be now considered as a single tool. By specifying two fastq files as inputs, this workflow will automatically process all the steps in the workflow and outputs a list of predicted fusion genes once the pipeline is finished.

Figure 4-3. The complete ChimeRScope workflow on the local Galaxy server. The workflow starts with the discordant reads from the alignment step using Tophat. Then, the discordantly aligned paired-end reads are extracted from *accepted_hits.bam*, and unmapped reads are extracted from *unmapped.bam*. The output from the previous step are merged together. Then, the output bam file will be sorted and converted back to paired-end fastq files. The unmapped paired-end reads serve as input files for the fusion gene prediction pipeline. The final output is a text file that lists the predicted fusion genes with other values (e.g., confidence scores, number of FESRs).



We analyzed the *50_pos_set* using the regular command line version of ChimeRScope and the workflow in the Galaxy server (with the same versions of the tools) to check the consistency of the output. We compared the final output files using *diff* command in bash. Results have shown that there is no difference between these two versions of the workflow.

6. Limitations and conclusions

Galaxy server is a great resource for bioinformatics data analysis. The Graphical User Interface makes the data analysis much easier for researchers with limited programming expertise.

The sharing utility also makes the analysis transparent and the workflow system makes the analysis reproducible. However, there are still limitations for using Galaxy server for large-scale data analysis.

Galaxy server is less flexible on the data analysis part in certain situations. For example, the sequencing read quality control step is necessary for NGS data analysis. We often use FastQC [68] to check the stats on sequencing errors, GC content, possible adapter contamination, etc. It generates both text reports and graphical reports. For large-scale data analysis using Galaxy server, users will need to manually check each graphical report to obtain different parameters for the sequence trimming step. On the other hand, using the command lines, researchers can write a single script that parses all the text reports to retrieve the parameter sets.

Galaxy server is less flexible for resource allocation. Galaxy server is an efficient data analysis server; however, users often do not have permissions to alter the computational resources for each job. For instance, *ChimeRScope_Scanner* in our local galaxy server will reserve 20GB of memory for the Java virtual machine, irrespective of the dataset size. This can cause *java.lang.OutOfMemoryError* on large datasets. Administrators cannot set it to reserve more memory because it will be a waste of computational resources for smaller datasets. It is also not recommended to set the memory usage parameter to a publicly accessible variable for security issues.

The data transfer rate is also one of the biggest bottlenecks for using the Galaxy server on large-scale data analysis. Downloading data from public databases (denoted as D) to the local computer (denoted as C) and uploading them to the Galaxy server (denoted as S) takes two operations for each dataset (D to C, then C to S), and the transferring speed is often limited to the bandwidth of the client C. Comparatively, for data analysis using command line on Linux servers, the data can be downloaded directly to the server (D to S). In this way, it takes only one operation and the transferring speed is irrelevant to the client bandwidth.

In conclusion, the Galaxy server implemented for ChimeRScope offers an alternative approach for fusion gene analysis. It offers GUI for researchers with limited programming experience. It is best suitable for smaller project (often less than 100 sequencing samples) with medium size datasets (paired-end reads, less than 10GB). Although the wrapper scripts for ChimeRScope can be easily modified by server administrators for large-scale data analysis, we still recommend using the command-line version for large-scale data analysis projects for better project management experience, result integration and interpretation.

Chapter 5: PROJECT SUMMARY AND FUTURE DIRECTIONS

Fusion genes are a class of chimeric transcripts often observed in different types of cancers that can have significant impact on the tumorigenesis and cancer progression. Rapid advances in the next generation sequencing technologies make it more feasible to generate transcriptomics data (RNA-Seq data) from cancer patients and identify fusion genes. We developed a novel alignment-free method named ChimeRScope for fusion gene discovery by evaluating the fingerprint sequences from RNA-Seq paired-end reads. Comparison results against other popular fusion gene detection tools show that ChimeRScope has better performance on all tested datasets that include both simulated and real RNA-Seq datasets. Notably, results on simulated datasets have shown that alignment-based methods rely largely on the technical aspects like read length and sequencing depth of the RNA-Seq datasets, whereas alignment-free methods like ChimeRScope are less likely to be affected by these factors as indicated by consistently superior F-scores across all datasets.

1. Configuration and installation

1.1. Summary

Java is a platform-independent language that typically compiles to the Java Virtual Machine (JVM). Theoretically, java programs can be executed on any operating systems with Java Runtime Environment (JRE) support. Its ability to move easily between different computer systems has been one of the most significant advantages of Java. ChimeRScope suite, implemented in Java Standard Edition 7, inherits this advantage. Users can download and set up JRE (version 1.7 or higher) from its official website at www.java.com. No extra installation steps are required for ChimeRScope.

ChimeRScope requires several reference files. The detailed instruction of how to prepare the files is shown below (using human reference genome build 38 with 17-mers as an example).

Step 1: Open UCSC table browser at
https://genome.ucsc.edu/cgi-bin/hgTables

Step 2: Select following options in the UCSC Table Browser

clade: Mammal **genome:** Human **assembly:** Dec. 2013 (GRCh38/hg38)
group: Genes and Gene Predictions **track:** RefSeq Genes
table: refGene
region: genome
output format: GTF - gene transfer format
output file: hg38.gtf
file type returned: plain text

Step 3: Extract all mRNA entries on chr1-22 chrX, chrY, ChrM from the gtf file using awk on linux.

```
cat hg38.gtf | awk 'length($1)<=5 { print }' | grep "\"NM_" > hg38_mRNAs.gtf
```

Step 4: Download hg38.fa.gz from
http://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/

Step 5: Unzip the gzip file and create all mRNAs sequences from the downloaded gtf file and fasta file (use gtf_to_fasta from TopHat package).

```
gunzip hg38.fa.gz
$TOPHAT_PATH/gtf_to_fasta hg38_mRNAs.gtf hg38.fa hg38_allmRNA.fa
```

Step 6: Download ID conversion files from UCSC table browser

clade: Mammal **genome:** Human **assembly:** Dec. 2013 (GRCh38/hg38)
group: Genes and Gene Predictions **track:** RefSeq Genes
table: refGene

region: genome

output format: selected fields from primary and related tables

output file: refseq2gs.txt

file type returned: plain text

Click get output **-> Under** Select Fields from hg38.refGene**, Only select** name **and** name2 **-> output is saved as** refseq2gs.txt

Step 7: Download gene family information from HGNC database at http://www.genenames.org/ (Optional/Recommended)

Click Downloads **->** Custom Downloads **-> Under** Curated by the HGNC**, only select** RefSeq IDs **and** Gene Family ID **-> Uncheck all boxes under** Downloaded from external sources **->** submit **-> save output as** geneFamily.txt

Step 8: Run ChimeRScope builder. E.g., output library directory is ./GF_lib/homo_sapiens**. It creates** hg38_mRNA_k17.ids**,** hg38_mRNA_k17.loc**, and** hg38_mRNA_k17.lib **after it is done.**

```
java -jar ChimeRScope.jar Builder \
        -i hg38_allmRNA.fa \
        -k 17 \
        -id refseq2gs.txt \
        -o ./GF_lib/homo_sapiens/hg38_mRNA_k17
```

Final step: Compile files

Link other files to the library directory with the same prefix

```
ln   -s   hg38_allmRNA.fa   ./GF_lib/homo_sapiens/hg38_mRNA_k17.fa
ln -s geneFamily.txt ./GF_lib/homo_sapiens/hg38_mRNA_k17.gf
```

Under ./GF_lib/homo_sapiens/, it should contain 5 files with the same prefix. They are,

```
hg38_mRNA_k17.fa
hg38_mRNA_k17.gf
hg38_mRNA_k17.ids
hg38_mRNA_k17.lib
hg38_mRNA_k17.loc
```

The preparation of the reference files for ChimeRScope is a one-time operation that usually takes less than one hour with 40GB memory. We also provide pre-compiled files for the users on the ChimeRScope homepage at https://github.com/ChimeRScope/ChimeRScope/wiki/ChimeRScopeManual.

1.2. Limitations and future directions

ChimeRScope aims to identify biologically significant fusion genes. We defined that fusion genes involving non-coding RNAs are less likely to produce functional fusion proteins or maintain their intended regulatory functions. Besides, tests on simulated datasets show that GF-libraries created from total RNAs can introduce more false positive fusion genes. Therefore, we recommend that the GF-library should be constructed only from mRNAs sequences for best prediction results. In RefSeq sequence database, the RefSeq IDs for mRNAs (start with "*NM_*") and non-coding RNAs (start with "*NR_*") are easily distinguishable. Comparatively, the transcript IDs for coding RNAs and non-coding RNAs in EnsEMBL database all start with "*ENSG*". For ChimeRScope GF-libraries, we prefer to use RefSeq sequences because it is easier to extract all the mRNA sequences from RefSeq annotation. Another reason for the preference on RefSeq annotation is that, RefSeq is a collection of non-redundant, curated RNA models, whereas

EnsEMBL database includes RNA sequences from different sources like GENCODE [69], CDDS database [70], automatically-annotated pseudogenes, and non-coding RNAs. The redundant sequences and predicted RNAs in EnsEMBL can introduce more false fusion genes (Chapter 4 BC_CL result). Moreover, the number of coding genes in EnsEMBL (~56,000) is more than twice as that for the RefSeq database (~25,000). In this case, the memory cost using the EnsEMBL reference database will be doubled. Considering all the points described above, the current version of ChimeRScope only supports the use of RefSeq annotations.

The choice of the gene annotation database can hinder some users from using ChimeRScope. However, we still believe that fusion gene prediction should be based on more conserved mRNA database like RefSeq for the best prediction results. We are not planning to offer direct support for EnsEMBL sequences. However, we will modify the code in the future release of ChimeRScope so that EnsEMBL sequences can also be used without causing any technical issues.

2. Prediction performance

2.1. Summary

In sequence comparison analysis, alignment-based methods will perform better than alignment-free methods if the target sequences are similar. Most of alignment-based fusion gene prediction methods identify fusion gene candidates by aligning RNA-Seq short reads against the normal reference genome. In cancer research, RNA-Seq reads are sequenced from cancer genome/transcriptome, where the genomes are high perturbed compared to normal ones. Comparing reads derived from cancer genome against normal reference genome will prevent alignment-based methods from getting reliable alignment results, especially for the highly perturbed regions where fusion events occur.

As we discussed earlier, ChimeRScope is less likely to be affected by genetic variations. Similarly, ChimeRScope is also tolerant to sequencing errors. Therefore, ChimeRScope does not take the sequencing quality scores into consideration. We believe that, the decrease of the sequencing quality near 3' of the reads (often observed in Illumina RNA-Seq data) will have little impact to the prediction results because only a few of *k*-mers will be affected. Removing reads with overall low quality scores (e.g., average score of all bases lower than 30) is not required for ChimeRScope because such reads are more likely to occur near repetitive regions and polymer regions. *k*-mers generated from these regions will have low or zero weightage scores. Nevertheless, low quality reads are often excluded after the quality control step, because it is a part of the standard NGS data analysis pipeline.

The core algorithm used in ChimeRScope is an alignment-free algorithm based on *k*-mers to represent the gene fingerprints. The comparison results on both simulated datasets and cancer RNA-Seq datasets all suggested the better performance of ChimeRScope against other popular methods. Moreover, results on the simulated datasets show that ChimeRScope consistently performs better than other methods in datasets with different length and coverage depth, suggesting the unique advantage of ChimeRScope over other alignment-based methods.

2.2. Limitations and future directions

For analysis on real RNA-Seq datasets, we recommend users to manually perform a BLAST search of the fusion sequences against the NCBI non-coding sequence database collections. This is because the non-coding RNA sequences are not included in the GF-library. Occasionally, some of the expressed non-coding RNAs are transcribed from coding exons of two different mRNAs. Reads sequenced from this type of non-coding RNAs can exhibit fusion patterns for the

corresponding mRNAs. For instance, the fusion gene reported in the original SOAPfuse paper, GATSL1-GTF2I, can be aligned to several non-coding RNAs (NR_002206.3 and NR_003580.2). ChimeRScope also found a similar fusion gene named GATSL2-GTF2I in the breast cancer samples. This class of fusion genes should be excluded in the downstream analysis because they should not be considered as cancer specific chimeras.

This manual filtering step can be tedious for projects with hundreds of samples. Here we present several alternatives to minimize the need of this extra step. **(1) Incorporate a BLAST database into ChimeRScope suite.** Tools such as SOAPfuse have built-in BLAST databases. However, several drawbacks are apparent. Firstly, it takes longer time to install and configure the database. Some of the key scripts for BLAST also require root privilege and only a server administrator can install it. Moreover, as a heuristic method, BLAST sometimes can also fail to filter out certain FP fusion genes (e.g., false positives with highly repetitive sequences near the fusion junction). Because we designed ChimeRScope suite as a clean, platform-independent package with minimum database dependencies, we did not incorporate BLAST database into the ChimeRScope package. **(2) Updating the list of known false positives.** ChimeRScope is very sensitive to sequences with fusion patterns, no matter they are true fusion events or not. A false positive fusion gene can be introduced from non-coding genes that can be mapped to two different genes separately, or known germline insertions in a gene where the inserted sequences show homologies to other genes, or other unannotated transcripts with similar fusion patterns, all of which are recurrently shown across different samples (including normal samples). ChimeRScope constantly classify these “normal” transcripts into false positives. These false positives can be manually filtered out in the last filtering step but it requires extra work. Alternatively, we can constantly update the list of the false positives that we already know from the previous analysis results. If a false positive fusion gene is reported in the new samples, it is automatically classified as a false positive in the final output and we will also provide the reason why this fusion is a false

positive. In summary, we plan to choose the latter option to facilitate the use of ChimeRScope because the results can be more accurate without incorporating the BLAST databases.

ChimeRScope does not allow any mismatches when comparing the fingerprint sequences, which might be the main reason why some of the true fusion genes in the tested datasets were not predicted by this method. The regular approach for allowing one mismatch when comparing *k*-mers can increase the processing time of each *k*-mer for a maximum of 3^k times and may also increase false positives. Hashing algorithm called SimHash [71, 72] might be useful to solve this computational issue. SimHash is an efficient algorithm that can be used to find similar fingerprints within a certain Hamming distance. It creates similar hash values for strings with similar sequences. We plan to explore the applicability of using SimHash in the future releases of ChimeRScope (ChimeRScope currently uses HashMap) for fusion gene detection.

ChimeRScope achieves better prediction accuracies overall when compared to other popular tools. However, there is still room for improvement. For example, FPs caused by large insertions (mentioned in the previous paragraph) can be filtered out by alignment-based methods with adjusted scoring matrices that allow gapped alignment. Currently, the alignment module of ChimeRScope (Examiner) only uses a fixed substitution matrix for targeted alignment (Chapter 4, subsection 2.1), which rarely allows the identification of long insertions (>50bp). To improve, we can test different substitution matrices and output the best alignment results for each targeted alignment so that this class of FPs will be automatically recognized.

The current version of ChimeRScope can only report fusion sequences near the fusion junctions in a range that is limited to the read length and the insert size of the paired-end reads. One obvious limitation is that ChimeRScope is not capable of accurately predicting the complete fusion gene sequences because reads covering regions other than the fusion junctions will not be analyzed. Ultimately, it will be less accurate for ChimeRScope to predict the functional impact of the fusion event (e.g., predict if a frameshift is involved so that the fusion protein might not be

translated or the functional domains might be changed). To overcome this limitation, we can extract the consensus nucleotide sequences for each fusion partner from the Sequence AlignMent Files (SAM or Binary SAM files, generated by aligners like TopHat) and combine them with the predicted fusion junction sequences.

Another improvement that might be used to increase the overall prediction accuracies rather than directly fix any limitations is to adopt machine learning, particularly Artificial Neural Network (ANN) [73] or evolutionary algorithms [74]. Information such as the expression levels of each fusion partner, sequencing quality scores, mutations near the fusion junctions, and known structural variations can be transformed as vectors and they might be directly associated with the confidence level of the fusion genes. We can download cancer cell lines RNA-Seq datasets and construct training and testing datasets using all the experimentally validated fusion genes found in these cancer cell lines. Therefore, all the fixed parameters used in the current version of ChimeRScope can be optimized automatically for each predicted fusion gene, which eventually, should lead to higher sensitivities and lower false discovery rate.

3. Epilogue

The current version of ChimeRScope has shown superior performance on a variety of tested datasets. We will regularly update ChimeRScope to fix bugs and improve the accuracy. We are maintaining the ChimeRScope wiki page at https://github.com/ChimeRScope/ChimeRScope/wiki to keep track of the version changes, updates and FAQs. We will also create ChimeRScope mailing lists to provide support for the ChimeRScope user community.

REFERENCES

1. **http://globocan.iarc.fr/Pages/fact_sheets_cancer.aspx.**
2. Futreal PA, Coin L, Marshall M, Down T, Hubbard T, Wooster R, Rahman N, Stratton MR: **A census of human cancer genes.** *Nat Rev Cancer* 2004, **4:**177-183.
3. Ungefroren H, Sebens S, Seidl D, Lehnert H, Hass R: **Interaction of tumor cells with the microenvironment.** *Cell Commun Signal* 2011, **9:**18.
4. Pakneshan S, Salajegheh A, Smith RA, Lam AK: **Clinicopathological relevance of BRAF mutations in human cancer.** *Pathology* 2013, **45:**346-356.
5. Dhillon AS, Hagan S, Rath O, Kolch W: **MAP kinase signalling pathways in cancer.** *Oncogene* 2007, **26:**3279-3290.
6. Chapman PB, Hauschild A, Robert C, Haanen JB, Ascierto P, Larkin J, Dummer R, Garbe C, Testori A, Maio M, et al: **Improved survival with vemurafenib in melanoma with BRAF V600E mutation.** *N Engl J Med* 2011, **364:**2507-2516.
7. Nowell PC: **The minute chromosome (Phl) in chronic granulocytic leukemia.** *Blut* 1962, **8:**65-66.
8. **http://www.cancer.gov/.**
9. Barros-Silva JD, Paulo P, Bakken AC, Cerveira N, Lovf M, Henrique R, Jeronimo C, Lothe RA, Skotheim RI, Teixeira MR: **Novel 5' fusion partners of ETV1 and ETV4 in prostate cancer.** *Neoplasia* 2013, **15:**720-726.
10. Wang LC, Swat W, Fujiwara Y, Davidson L, Visvader J, Kuo F, Alt FW, Gilliland DG, Golub TR, Orkin SH: **The TEL/ETV6 gene is required specifically for hematopoiesis in the bone marrow.** *Genes Dev* 1998, **12:**2392-2402.
11. Panagopoulos I, Strombeck B, Isaksson M, Heldrup J, Olofsson T, Johansson B: **Fusion of ETV6 with an intronic sequence of the BAZ2A gene in a paediatric pre-B acute lymphoblastic leukaemia with a cryptic chromosome 12 rearrangement.** *Br J Haematol* 2006, **133:**270-275.
12. Parker BC, Annala MJ, Cogdell DE, Granberg KJ, Sun Y, Ji P, Li X, Gumin J, Zheng H, Hu L, et al: **The tumorigenic FGFR3-TACC3 gene fusion escapes miR-99a regulation in glioblastoma.** *J Clin Invest* 2013, **123:**855-865.
13. **https://en.wikipedia.org/wiki/Imatinib.**
14. Rehman FL, Lord CJ, Ashworth A: **Synthetic lethal approaches to breast cancer therapy.** *Nat Rev Clin Oncol* 2010, **7:**718-724.
15. Glenn TC: **Field guide to next-generation DNA sequencers.** *Mol Ecol Resour* 2011, **11:**759-769.
16. Zimmerman E: **50 Smartest Companies: Illumina.** *MIT Technology Review* 18 February 2014.
17. Shin J, Ming GL, Song H: **Decoding neural transcriptomes and epigenomes via high-throughput sequencing.** *Nat Neurosci* 2014, **17:**1463-1475.

18. Maher CA, Kumar-Sinha C, Cao X, Kalyana-Sundaram S, Han B, Jing X, Sam L, Barrette T, Palanisamy N, Chinnaiyan AM: **Transcriptome sequencing to detect gene fusions in cancer.** *Nature* 2009, **458:**97-101.

19. Edgren H, Murumagi A, Kangaspeska S, Nicorici D, Hongisto V, Kleivi K, Rye IH, Nyberg S, Wolf M, Borresen-Dale AL, Kallioniemi O: **Identification of fusion genes in breast cancer by paired-end RNA-sequencing.** *Genome Biol* 2011, **12:**R6.

20. Smith TF, Waterman MS: **Identification of common molecular subsequences.** *J Mol Biol* 1981, **147:**195-197.

21. Sboner A, Habegger L, Pflueger D, Terry S, Chen DZ, Rozowsky JS, Tewari AK, Kitabayashi N, Moss BJ, Chee MS, et al: **FusionSeq: a modular framework for finding gene fusions by analyzing paired-end RNA-sequencing data.** *Genome Biol* 2010, **11:**R104.

22. McPherson A, Hormozdiari F, Zayed A, Giuliany R, Ha G, Sun MG, Griffith M, Heravi Moussavi A, Senz J, Melnyk N, et al: **deFuse: an algorithm for gene fusion discovery in tumor RNA-Seq data.** *PLoS Comput Biol* 2011, **7:**e1001138.

23. Li Y, Chien J, Smith DI, Ma J: **FusionHunter: identifying fusion transcripts in cancer using paired-end RNA-seq.** *Bioinformatics* 2011, **27:**1708-1710.

24. Jia W, Qiu K, He M, Song P, Zhou Q, Zhou F, Yu Y, Zhu D, Nickerson ML, Wan S, et al: **SOAPfuse: an algorithm for identifying fusion transcripts from paired-end RNA-Seq data.** *Genome Biol* 2013, **14:**R12.

25. Francis RW, Thompson-Wicking K, Carter KW, Anderson D, Kees UR, Beesley AH: **FusionFinder: a software tool to identify expressed gene fusion candidates from RNA-Seq data.** *PLoS One* 2012, **7:**e39987.

26. Kim D, Salzberg SL: **TopHat-Fusion: an algorithm for discovery of novel fusion transcripts.** *Genome Biol* 2011, **12:**R72.

27. Wang K, Singh D, Zeng Z, Coleman SJ, Huang Y, Savich GL, He X, Mieczkowski P, Grimm SA, Perou CM, et al: **MapSplice: accurate mapping of RNA-seq reads for splice junction discovery.** *Nucleic Acids Res* 2010, **38:**e178.

28. Ge H, Liu K, Juan T, Fang F, Newman M, Hoeck W: **FusionMap: detecting fusion genes from next-generation sequencing data at base-pair resolution.** *Bioinformatics* 2011, **27:**1922-1928.

29. Nicorici D, Satalan M, Edgren H, Kangaspeska S, Murumägi A, Kallioniemi O, Virtanen S, Kilkku O: **FusionCatcher–a tool for finding somatic fusion genes in paired-end RNA-sequencing data.** 2014.

30. Davidson NM, Majewski IJ, Oshlack A: **JAFFA: High sensitivity transcriptome-focused fusion gene detection.** *Genome Med* 2015, **7:**43.

31. Ding L, Wendl MC, McMichael JF, Raphael BJ: **Expanding the computational toolbox for mining cancer genomes.** *Nat Rev Genet* 2014, **15:**556-570.

32. Carrara M, Beccuti M, Cavallo F, Donatelli S, Lazzarato F, Cordero F, Calogero RA: **State of art fusion-finder algorithms are suitable to detect transcription-induced chimeras in normal tissues?** *BMC Bioinformatics* 2013, **14 Suppl 7:**S2.

33. Kim D, Pertea G, Trapnell C, Pimentel H, Kelley R, Salzberg SL: **TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions.** *Genome Biol* 2013, **14:**R36.

34. Burge C, Campbell AM, Karlin S: **Over- and under-representation of short oligonucleotides in DNA sequences.** *Proc Natl Acad Sci U S A* 1992, **89:**1358-1362.

35. **https://en.wikipedia.org/wiki/Diversity_index.**

36. MacArthur RH, MacArthur JW: **On Bird Species Diversity.** *Ecology* 1961, **42:**594-598.

37. Trapnell C, Pachter L, Salzberg SL: **TopHat: discovering splice junctions with RNA-Seq.** *Bioinformatics* 2009, **25:**1105-1111.

38. Dobin A, Davis CA, Schlesinger F, Drenkow J, Zaleski C, Jha S, Batut P, Chaisson M, Gingeras TR: **STAR: ultrafast universal RNA-seq aligner.** *Bioinformatics* 2013, **29:**15-21.

39. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Genome Project Data Processing S: **The Sequence Alignment/Map format and SAMtools.** *Bioinformatics* 2009, **25:**2078-2079.

40. Quinlan AR, Hall IM: **BEDTools: a flexible suite of utilities for comparing genomic features.** *Bioinformatics* 2010, **26:**841-842.

41. Prakash T, Sharma VK, Adati N, Ozawa R, Kumar N, Nishida Y, Fujikake T, Takeda T, Taylor TD: **Expression of conjoined genes: another mechanism for gene regulation in eukaryotes.** *PLoS One* 2010, **5:**e13284.

42. Lindgreen S: **AdapterRemoval: easy cleaning of next-generation sequencing reads.** *BMC Res Notes* 2012, **5:**337.

43. Greger L, Su J, Rung J, Ferreira PG, Geuvadis c, Lappalainen T, Dermitzakis ET, Brazma A: **Tandem RNA chimeras contribute to transcriptome diversity in human population and are associated with intronic genetic variants.** *PLoS One* 2014, **9:**e104567.

44. Liu S, Tsai WH, Ding Y, Chen R, Fang Z, Huo Z, Kim S, Ma T, Chang TY, Priedigkeit NM, et al: **Comprehensive evaluation of fusion transcript detection algorithms and a meta-caller to combine top performing methods in paired-end RNA-seq data.** *Nucleic Acids Res* 2015.

45. **http://trace.ncbi.nlm.nih.gov/Traces/sra/?study=SRP049695.**

46. Langmead B, Trapnell C, Pop M, Salzberg SL: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome.** *Genome Biol* 2009, **10:**R25.

47. Kent WJ: **BLAT--the BLAST-like alignment tool.** *Genome Res* 2002, **12:**656-664.

48. Kangaspeska S, Hultsch S, Edgren H, Nicorici D, Murumagi A, Kallioniemi O: **Reanalysis of RNA-sequencing data reveals several additional fusion genes with multiple isoforms.** *PLoS One* 2012, **7:**e48745.

49. Asmann YW, Hossain A, Necela BM, Middha S, Kalari KR, Sun Z, Chai HS, Williamson DW, Radisky D, Schroth GP, et al: **A novel bioinformatics pipeline for identification and characterization of fusion transcripts in breast cancer and normal cell lines.** *Nucleic Acids Res* 2011, **39:**e100.

50. Babiceanu M, Qin F, Xie Z, Jia Y, Lopez K, Janus N, Facemire L, Kumar S, Pang Y, Qi Y, et al: **Recurrent chimeric fusion RNAs in non-cancer tissues and cells.** *Nucleic Acids Res* 2016, **44:**2859-2872.

51. Yasuda T, Tsuzuki S, Kawazu M, Hayakawa F, Kojima S, Ueno T, Imoto N, Kohsaka S, Kunita A, Doi K, et al: **Recurrent DUX4 fusions in B cell acute lymphoblastic leukemia of adolescents and young adults.** *Nat Genet* 2016, **48:**569-574.

52. Ye J, Coulouris G, Zaretskaya I, Cutcutache I, Rozen S, Madden TL: **Primer-BLAST: a tool to design target-specific primers for polymerase chain reaction.** *BMC Bioinformatics* 2012, **13:**134.

53. Nikolaus D. Obholzer BJH, Dan-Avi Landau, Nathalie Pochet, Aviv Regev, Catherine Wu: **Development of a cancer transcriptome analysis toolkit: identification of gene fusions in chronic lymphocytic leukemia. .** *Cancer Research* 2015.

54. **http://www.ncbi.nlm.nih.gov/gene/6924.**

55. Nacu S, Yuan W, Kan Z, Bhatt D, Rivers CS, Stinson J, Peters BA, Modrusan Z, Jung K, Seshagiri S, Wu TD: **Deep RNA sequencing analysis of readthrough gene fusions in human prostate adenocarcinoma and reference samples.** *BMC Med Genomics* 2011, **4:**11.

56. Liu S, Tsai WH, Ding Y, Chen R, Fang Z, Huo Z, Kim S, Ma T, Chang TY, Priedigkeit NM, et al: **Comprehensive evaluation of fusion transcript detection algorithms and a meta-caller to combine top performing methods in paired-end RNA-seq data.** *Nucleic Acids Res* 2016, **44:**e47.

57. Liu X, Tang J, Wang M, Ma Q, Wang Y: **Visual detection and evaluation of latent and lytic gene expression during Epstein-Barr virus infection using one-step reverse transcription loop-mediated isothermal amplification.** *Int J Mol Sci* 2013, **14:**23922-23940.

58. Heller M, Watts JD, Aebersold R: **CD28 stimulation regulates its association with N-ethylmaleimide-sensitive fusion protein and other proteins involved in vesicle sorting.** *Proteomics* 2001, **1:**70-78.

59. Wilson JL, Charo J, Martin-Fontecha A, Dellabona P, Casorati G, Chambers BJ, Kiessling R, Bejarano MT, Ljunggren HG: **NK cell triggering by the human costimulatory molecules CD80 and CD86.** *J Immunol* 1999, **163:**4207-4212.

60. Blackmore JK, Karmakar S, Gu G, Chaubal V, Wang L, Li W, Smith CL: **The SMRT coregulator enhances growth of estrogen receptor-alpha-positive breast cancer cells by promotion of cell cycle progression and inhibition of apoptosis.** *Endocrinology* 2014, **155:**3251-3261.

61. Ghoshal P, Nganga AJ, Moran-Giuati J, Szafranek A, Johnson TR, Bigelow AJ, Houde CM, Avet-Loiseau H, Smiraglia DJ, Ersing N, et al: **Loss of the SMRT/NCoR2 corepressor correlates with JAG2 overexpression in multiple myeloma.** *Cancer Res* 2009, **69:**4380-4387.

62. Eisold M, Asim M, Eskelinen H, Linke T, Baniahmad A: **Inhibition of MAPK-signaling pathway promotes the interaction of the corepressor SMRT with the human androgen**

receptor and mediates repression of prostate cancer cell growth in the presence of antiandrogens. *J Mol Endocrinol* 2009, **42:**429-435.

63. Espinosa L, Ingles-Esteve J, Robert-Moreno A, Bigas A: **IkappaBalpha and p65 regulate the cytoplasmic shuttling of nuclear corepressors: cross-talk between Notch and NFkappaB pathways.** *Mol Biol Cell* 2003, **14:**491-502.

64. Xie XM, Zhang ZY, Yang LH, Yang DL, Tang N, Zhao HY, Xu HT, Li QC, Wang EH: **Aberrant hypermethylation and reduced expression of disabled-2 promote the development of lung cancers.** *Int J Oncol* 2013, **43:**1636-1642.

65. Xie Y, Zhang Y, Jiang L, Zhang M, Chen Z, Liu D, Huang Q: **Disabled homolog 2 is required for migration and invasion of prostate cancer cells.** *Front Med* 2015, **9:**312-321.

66. Tong JH, Ng DC, Chau SL, So KK, Leung PP, Lee TL, Lung RW, Chan MW, Chan AW, Lo KW, To KF: **Putative tumour-suppressor gene DAB2 is frequently down regulated by promoter hypermethylation in nasopharyngeal carcinoma.** *BMC Cancer* 2010, **10:**253.

67. Giardine B, Riemer C, Hardison RC, Burhans R, Elnitski L, Shah P, Zhang Y, Blankenberg D, Albert I, Taylor J, et al: **Galaxy: a platform for interactive large-scale genome analysis.** *Genome Res* 2005, **15:**1451-1455.

68. **http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc.**

69. Harrow J, Frankish A, Gonzalez JM, Tapanari E, Diekhans M, Kokocinski F, Aken BL, Barrell D, Zadissa A, Searle S, et al: **GENCODE: the reference human genome annotation for The ENCODE Project.** *Genome Res* 2012, **22:**1760-1774.

70. Pruitt KD, Harrow J, Harte RA, Wallin C, Diekhans M, Maglott DR, Searle S, Farrell CM, Loveland JE, Ruef BJ, et al: **The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes.** *Genome Res* 2009, **19:**1316-1323.

71. Sood S, Loguinov D: **Probabilistic near-duplicate detection using simhash.** In *Proceedings of the 20th ACM international conference on Information and knowledge management*. pp. 1117-1126. Glasgow, Scotland, UK: ACM; 2011:1117-1126.

72. Charikar MS: **Similarity estimation techniques from rounding algorithms.** In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. pp. 380-388. Montreal, Quebec, Canada: ACM; 2002:380-388.

73. Lancashire LJ, Lemetre C, Ball GR: **An introduction to artificial neural networks in bioinformatics--application to complex microarray and mass spectrometry datasets in cancer studies.** *Brief Bioinform* 2009, **10:**315-329.

74. Fraser AS: **Monte Carlo analyses of genetic models.** *Nature* 1958, **181:**208-209.