CHARACTERIZATION OF SHORT TANDEM REPEATS USING LOCAL ASSEMBLY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Gülfem Demir
March 2017

Characterization of Short Tandem Repeats Using Local Assembly
By Gülfem Demir
March 2017

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Can Alkan(Advisor)

Ercüment Çiçek

Mehmet Somel

Approved for the Graduate School of Engineering and Science:

Ezhan Karaşan
Director of the Graduate School

ABSTRACT

CHARACTERIZATION OF SHORT TANDEM REPEATS USING LOCAL ASSEMBLY

Gülfem Demir
M.S. in Computer Engineering
Advisor: Can Alkan
March 2017

Tandem repeats are pieces of DNA where a pattern has multiple consecutive copies adjacent to itself. If the repeat unit (pattern) consists of 2 to 6 nucleotides, it can be referred to as a short tandem repeat or a microsatellite. There are many genetic diseases (such as huntington disease and Fragile-X syndrome) linked with STR expansions and because tandem repeats make up 3% of the sequenced human genome, STR detection research is significant.

STR variations have always been a challenge for genome assembly and sequence alignment due to their repetitive nature, sequencing errors, short read lengths, and high incidence of polymerase slippage at STR regions. Despite the information they carry being very valuable, STR variations have not gained enough attention to be a permanent step in genome sequence analysis pipelines. After the 1000 Genomes Project, which aimed to establish the most detailed genetic variation catalogue for humans, the consortium released only two STR prediction sets which are identified by two STR caller tools, lobSTR and RepeatSeq. Many other large research efforts have failed to shed light on STR variations.

The main aim of this study is to use sequence assembly methods for regions where we know that there is an STR, based on reference genome, and release a complete pipeline from sample's reads to STR genotype. The assembly problem we are dealing with in the scope of this thesis can be considered as local assembly, which is the assembly procedure of reads that maps to a small part of the genome. We will be focusing on two general assembly approaches that make use of graph data structures: de Bruijn graph (DBG) based methods that rely on a variant of k-mer graph, overlap-layout-consensus (OLC) methods that are based on an overlap graph. Even though sequence assembly is a well studied problem, there is not any work that uses assembly algorithms to characterize STRs. We demonstrate

that using sequence assembly on STR regions increases the true positive rate compared to state-of-art tools.

We evaluated the performance of three different local assembly methods on three different experimental settings: focusing on (i) genotype based performance, (ii) coverage impact, and (iii) evaluating pre-processing and including flanking regions. All these experiments supported our initial expectations on using assembly. Besides, we show that OLC based assembly methods bring much higher sensitivity to STR variant calling when compared to DBG based approach. This concludes that assembly with OLC is a better way for genotyping STRs according to our experiments.

Keywords: short tandem repeat, sequence assembly, next generation sequencing.

ÖZET

LOKAL DNA BİRLEŞTİRME METHODU İLE MİKROSATELLİTLERİN BULUNMASI

Gülfem Demir
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: Can Alkan
Mart 2017

Bitişik tekrarlar, genomda kısa nükleotid sekanslarının düzenli olarak tekrarlanmasıdır. Eğer tekrar eden kısım 2-6 bp uzunluğunda ise mikrosatellitler ya da kısa bitişik tekrarlar olarak adlandırılır. Mikrosatellitlerin kopya sayısının artmasıyla ilişkilendirilmiş birçok hastalık bulunmaktadır, bunlara örnek olarak Huntington hastalığı ve Frajil X sendromu gösterilebilir. Bu yüzden insan genomunun yüzde üçünü oluşturan bitişik tekrarların tespit edilmesi önemli bir araştırma alanıdır.

Mikrosatellit lokus yer alan varyantlar tekrarlı yapıları, dizileme sırasında meydana gelen hatalar, kısa DNA okumaları ve son olarak sıklıkla meydana gelen PCR hataları yüzünden genom birleştirme ve dizi hizalama için her zaman problem teşkil etmiştir. Büyük öneme sahip olmalarına rağmen mikrosatellitlerin bulunması hiçbir zaman DNA dizileme süreçlerinin kalıcı bir parçası olarak görülmemiştir. İnsan genomları arasındaki farklılıkların kataloglanmasını amaçlayan 1000 Genom Projesi'nin başlatılmasından sonra ilgili konsorsiyum sadece iki farklı mikrosatellit bulma methodunun sonucunu yayınladı (lobSTR ve RepeatSeq). Diğer büyük projelerin de mikrosatellit konusunu aydınlığa kavuşturmak için harcadığı çabalar başarısızlıkla sonuçlandı.

Bu çalışmanın ana amacı genom birleştirme yöntemlerini, referans genom dizilimden bildiğimiz mikrosatellit pozisyonları üzerinde kullanarak incelenen genomun referanstan farklılıklarını bulmaktır. Bunun için DNA okumalarını girdi olarak alıp çıktı olarak mikrosatellit kopya sayısını veren bir süreç geliştirilmiştir. Bu tez kapsamında uğraştığımız problem bütün genomdan ziyade lokal birleştirme olarak görülebilir, çünkü sadece mikrosatellit bölgesine karşılık gelen okumaları birleştirmeye çalışıyoruz. Bu çalışmada genom birleştirme için sıklıkla kullanılan iki farklı çizge yapısından yararlanıyoruz: de Bruijn and OLC. Genom birleştirme

bir çok çalışmanın bulunduğu bir alan olmasına rağmen, şu ana kadar mikrosatellit için kullanan bir çalışma bulunmamaktadır ve diğer mikrosatellit methodlarından daha iyi bir performans gösterdiği kanıtlanmaktadır.

Üç farklı genom birleştirme methodunu, yukarıda bahsedilen iki çizge yapısını kullanan, üç farklı deney modelinde inceledik. Deneyler genotip olarak farklı durumlarda, değişen teminatlarla ya da birleştirilen bölgeye komşu bölgelerin dahil edilmesi durumundaki farklılıkları incelemek için düzenlendi. Her birinin sonucu OLC çizge yapısını kullanan methodun mikrosatellit bulunmasındaki üstünlüğünü kanıtlamaktadır.

Anahtar sözcükler: mikrosatelit, genom birleştirme, yeni nesil dizileme.

To my family and

to my dearest uncle...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the primary aims of genomics studies is to characterize genetic variations and associate them with phenotypical changes including genetic diseases. Recently there has been substantial progress in detecting various types of genetic variations. Genome-wide association analyses have already identified thousands of genetic loci linked with human phenotypes, diseases, complex traits, and disorders. While many different types of genetic variations, such as single nucleotide polymorphisms (SNP), copy number variations (CNV), structural variations (SV), have been identified by these studies, short tandem repeat (STR) variations remain largely understudied.

For example, The 1000 Genomes Project, which aimed to establish the most detailed genetic variation catalogue for humans, analyzed 2,504 individuals from 26 populations and only reported SNPs, indels and a limited number of types of structural variations (i.e. deletions, small inversions, mobile element insertions, and tandem duplications) in detail. Six years after the start of the project, the consortium has released only two STR prediction sets which are identified by two STR caller tools, namely lobSTR and RepeatSeq. The 1000 Genomes Project and many other large research efforts have failed to shed a light on STR variations.

The major obstacle is the complex nature of STRs. Being a rich primary source

of genetic variation, single nucleotide changes are probably the simplest type and easiest to assay. On the other hand, STRs (microsatellites) are composed of a few nucleotides that are repeated several times. This structure causes a high mutation rate, which can reach 1/500 mutation per locus per generation. This is 200x higher than the rate of CNVs and 200,000x higher than the rate of *de novo* SNPs. Their hypervariability and ubiquity throughout the genome makes them hard to spot.

Despite being harder to identify, STRs are still highly utilized in human genetics applications since they serve as a major source of genetic polymorphism among individuals:

- **Forensics:** STR analysis is the *de facto* standard for constructing national public forensic DNA databases [4]. STRs usually have small number of alleles which increases the information entropy of a single STR. This means that a limited number of STRs can sufficiently identify a single individual. During late 1990s, the FBI Laboratory has established the CODIS set that only contains 13 STR loci, which later has been recognized as the standard for human identification.
- **Medical genetics:** STR mutations have been associated to more than 40 single-gene disorders [5], such as Huntington's disease and amyotrophic lateral sclerosis/frontotemporal dementia (ALS/FTD). In the case of ALS, the condition is triggered by the abnormal expansion of short repeat units [6]. In addition to single-gene disorders, microsatellites also contribute to various complex traits heritability.

Lastly, STR variations are important for population genetics studies, linkage analysis, and genetic anthropology.

1.1 Motivation and Health Relevance

Dentatorubral-pallidoluysian atrophy (DRPLA) is rare brain disorder that mainly impacts mental and emotional state, intellectual ability and causes uncontrollable muscle movements in the patient. It is associated with the expansion of CAG repeats over 49-88 repeats on the ATN1 gene. Similarly, the mutated AR gene with expanded CAG repeats (40 to 62 repeats) in the coding region is responsible for the pathogenesis of Spinal-bulbar muscular atrophy (SBMA or sometimes called Kennedy disease), in which loss of motor neurons affects the voluntary muscle movement in the face, mouth and throat [7]. DRPLA and SBMA are only two examples of biological effects of short tandem repeats. The fact that there are many more genetic diseases linked with STR expansions and such tandem repeats make up ˜3% of the sequenced human genome makes STR detection research even more significant [8].

Repetitive patterns is a major cause of ambiguity in genome assembly and sequence alignment, which may later be the root of inaccurate interpretations. Hence, STR variations, due to their repetitive nature, have always been a challenge for genome assembly and sequence alignment [9]. Mostly for that reason, STR variations are relatively unexplored and lack a large-scale analysis; whereas other types of variations (SNPs, CNVs, insertion and deletions, etc.) have been comprehensively cataloged in extensive studies [5, 10].

STR variation is a special case of indel. Although generic indel calling tools can be used to detect STRs, they are not specialized for that and not as performant as tools such as lobSTR and RepeatSeq, both of which are STR variant callers based on high-throughput sequencing data and split reads signature [11, 12]. However, there are only a limited number of tools available that have been developed specifically for detecting STR variations and to the best of our knowledge, none of them utilize local genome assembly methods during variant calling phase [13].

Most of STR variant callers try to identify variation by comparing a read sequence with a reference sequence. Since they expect reads to be longer than

repeat regions, this approach limits the detectable STR length significantly. Using local genome assembly information enables us to be able to identify longer STR variations.

There is much room for improvement in STR characterization, especially identification of *de novo* expansions and contractions, which is crucial for many fields in biology and its applications such as medical genetics, forensics, and population genetics. Our motivation is to improve the accuracy of STR copy number detection by making use of local genome assembly of reads.

1.2 Challenge

There are many STR regions that are hard to detect using Illumina reads, which are generally up to 150 base pairs in length. Even though sequencing machines producing longer reads are becoming popular, there are still issues about their costs and high indel error rates, which complicate detecting STRs. Furthermore, if the region is longer than the read length, aligners cannot map it uniquely.

Another crucial challenge is that STRs have several PCR stutter artifacts which results in STRs that have different size when compared to true underlying genotype.

Lastly, although there have been considerable effort in understanding the nature of sequencing errors, variant calling pipelines still suffer from them.

1.3 Contributions

In this study, we exploited local assembly methods to characterize short tandem repeats.

- We created a pipeline that goes from reads to STR genotype. We integrated

local assembly as a new step this pipeline.

- We demonstrated that using local sequence assembly on STR regions may help variant callers increase sensitivity.
- We evaluated assembly methods that make use of graph data structures, namely de Bruijn graph and overlap-layout-consensus based approaches.
- We analyzed significance of read coverage in STR detection.

1.4 Structure of the Thesis

The thesis is organized as follows:

- Chapter 1 summarizes the motivation behind this research, challenges, and contributions.
- Chapter 2 provides a brief background information on DNA sequencing, genome assembly, and short tandem repeats.
- Chapter 3 introduces the complete pipeline built for STR genotyping. We explain the process for each step in the pipeline, discuss related tools, and justify why they have been chosen over others.
- Chapter 4 presents three experiment setups and reports their results.
- Chapter 5 concludes the thesis by giving final remarks, discussing experiment results, and suggesting possible directions for future researches in this field.
- Appendix A contains complete experiment results in a tabular format.

Chapter 2

Background

In this chapter, we provide brief information about DNA sequencing, genome assembly challenges, graph based assembly methods, short tandem repeats, and STR variant callers.

2.1 DNA Sequencing

DNA sequencing is the procedure for determining the exact order of four nucleotides (A, C, G, T) that make up a DNA molecule. Fundamentally, there have been two approaches to DNA sequencing starting from late 1970s [14].

2.1.1 First Generation Sequencing

Introduced in the late 1970s, Maxam-Gilbert and Sanger sequencing were early DNA sequencing methods that have been taken up widely [14]. Although Maxam-Gilbert method has lost its appeal in the following years due to its technical complexity, Sanger sequencing is still heavily used.

Sanger method tries to replicate DNA replication reaction. It is able to sequence regions of DNA consisting up to 900 base pairs [15]. It provides unprecedentedly accurate and long reads. That is why Sanger sequencing is used in human reference genome although it's considerably slower and more expensive compared to more modern approaches.

2.1.2 Next Generation Sequencing

First generation sequencing methods have some limitations:

- Sequencing reactions cannot be parallelized, hence not scalable.
- The entire process is very expensive.
- Since reactions are costly and cannot be parallelized, the entire process is slow.

Next generation sequencing (NGS) approaches such as Illumina, Roche 454, and Ion torrent solve those issues. They provide highly scalable, faster and cheaper methods so that large quantities of DNA can be sequenced more quickly and cheaply. The cost of sequencing a human genome is reduced by 99% ($100m in 2001, $1245 in 2015) thanks to next generation sequencing [16]. In order to provide a cheaper and faster alternative, next generation sequencing methods compromise quality (more error prone) and read length (shorter reads).

For building a gold standard set, Sanger sequencing is still the way to go as it is much less error prone. However, for other genomics studies NGS is invaluable.

There are new sequencing studies that tackle these problems of next generation sequencing, which are usually referred as third generation. They are capable of generating longer reads compared to NGS methods. However, their error profile is still pretty variable, which is the main reason NGS still dominates the market share.

2.2 Genome Assembly

None of currently available sequencing methods (sequencing by synthesis, single-molecule real-time sequencing and chain termination method) can read whole genome in one go. Instead, they all read small pieces of DNA from whole genome. The number of bases they can read depends on the sequencing technology and machine, but it is between 20 and 30000 nucleotides [17]. For this reason, we need an assembly process that maps sequence data to a reconstruction of the target as a whole.

Genome assembly is at the bedrock of all genomics fields, as almost every upstream analyses such as calling variations between individuals, alignment of reads, and studying evolution, depend upon genome references. A human reference genome, which is a representative consensus of humans' set of genes with the help of DNA sequencing, first released in 2003 in the scope of Human Genome Project [18]. It is called reference because it actually does not represent single individual human genome; instead it serves as a haploid representation of different people's DNA sequences without unnecessary duplication of content. For instance, build 37 version of reference genome (released by the Genome Reference Consortium) is based on 13 different individuals' DNAs. On the other hand, while the efforts for getting better reference consensus assembly have been continuing, 2 personal genomes are released in 2007. As opposed to previous efforts, these references are assembled from single individuals, namely James Watson and Craig Venter, by saving the diploid nature of their genomes [19, 20].

Human Genome Project were carried out by five largest centers around the world: Sanger Institute, DOE's Joint Genome Institute, three NIH-funded centers at Baylor College of Medicine, Washington University School of Medicine, and Whitehead Institute. Even though it was the most comprehensive study in this field, there have been only five releases of human reference genome in a ten-year span, from 2003 to 2013 by Genome Reference Consortium (GRC). This gives an idea of how difficult genome assembly problem is. Still, there is always a constant improvement on each release. For example, in the previous version of

human genome reference (GRCh37) there were nearly 234 million N letters, which is used to represent sequence gap or unannotated regions. In the latest release (GRCh38) this has been reduced by 35% to nearly 150 million [21].

In genome assembly, two different methodologies can be distinguished as *de novo* assembly and mapping assembly. *De novo* assembly, which has been used in human reference genome studies, requires starting from a set of reads whereas the latter assumes existence of reference genome. It means that mapping assembly can start with mapping reads to substantial reference genome, which is considerably easier and requires less coverage.

To practice a *de novo* assembly, DNA fragments (parts of DNA strands as sequencing machines can handle) of the target sample are collected such that they cover the genome entirely. Then, these DNA fragments are aligned against each other and merged based on overlap information to get contiguous segments (contigs). Using contigs and paired end information between reads, scaffolds are formed. Scaffolds might contain gaps next to each other.Lastly, multiple scaffolds are joined so that they form a chromosome.

Table 2.1: Larger fragments are built from small sequenced parts

																								T	C	T	A	A	T	A	G	C	A	C	A	C	A	
																						G	C	T	T	C	T	A	T	T	A	C	C	A	C	A	C	
																						A	C	T	T	C	T	A	A	T	A	C	C	T	C	A	C	
																		C	T	C	C	G	C	T	T	C	T	A	A	T	A	C								
																	T	C	T	C	C	A	C	T	T	C	T	A	A	T	A	C								
																A	T	C	T	C	C	G	C	T	T	C	T	A	A	T										
										C	C	T	G	A	T	T	T	C	T	C	C	A	C	T	T	C														
										C	C	T	G	A	T	T	T	C	T	C	C	G	C	T	T	C	T	A	A	T	A									
				T	G	G	C	C	C	C	G	A	T	T	T	C	T	C	C	G	C																			
T	A	C	T	G	G	C	C	C	C	G																													
T	A	C	C	G	G	C	C	C	T	G	A	T																											
T	A	C	T	G	G	C	C	C	T																														
T	A	C	T	G	G	C	C	C	T	G	A	T	T	T	C	T	C	C	G	C	T	T	C	T	A	A	T	A	C	C	A	C	A	C	A		

On the other hand, in alignment based assembly, reads are aligned against a similar reference genome, typically from the same specie, using a read aligner that can tolerate variants up to some point. There are some other reference guided methods that use reference genome's structure to form scaffold from contigs,

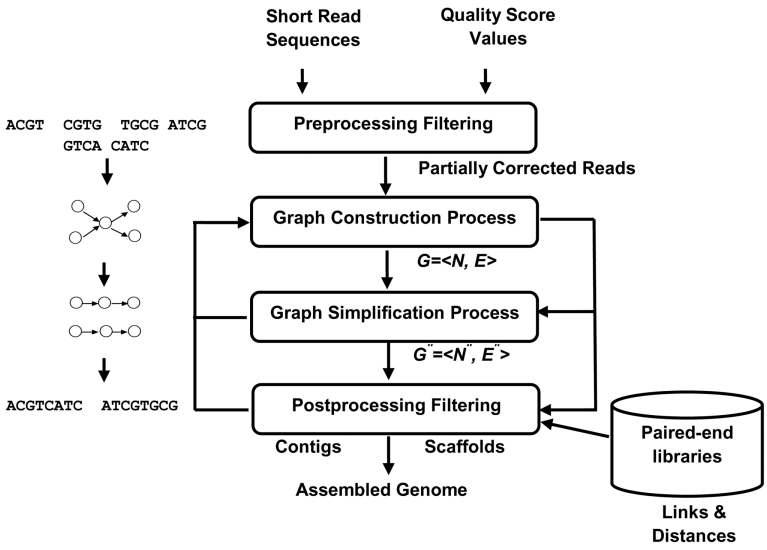

Figure 2.1: Schematic representation of the four stages of the next-generation genome assembly process (adapted from [2])

rather than using reference in the beginning of assembly procedure [22].

2.2.1 Challenges in Genome Assembly

Highly repetitive nature of human genome does not help assembly at all. According to a study conducted in 2011, even though widely accepted claim is that about half of the human genome consists of repetitive sequences, mostly transposable elements; the ratio of repetitive sequences in human genome, in fact, is much higher around 66-69% [23]. Genomics regions with exact copies can not be distinguished from each other, especially if repeat length is greater than the read length. Nevertheless, single reads that cover repetitive region and non-repetitive unique sequence next to it might help to make this problem easier. Paired end reads and information about average distance between pairs together are certainly useful for assembly as well [24].

Unfortunately, repetitiveness is not the only problem about sequence assembly. Depending on sequencing chemistry methods, various kinds of errors, such

as deletion, insertion, substitution or other platform-specific issues, may appear during the whole amplification and sequencing processes. Average error rates are below 0.4%, 1.78% and 13% for sequencing platforms Illumina, Ion Torrent and PacBio, respectively. More importantly the number of reads that do not contain any kind of error are 76.45% for Illumina MiSeq, 15.92% for Ion Torrent and 0% for PacBio [25]. This is where a need arises for alignment tools which is capable of tolerating sequencing errors. However, imperfect error tolerance algorithms would lead to other problems such as false positive joins. This problem occurs principally in reads from polymorphic repeats, which is highly important for STR discovery.

Sequence assembly might also be affected negatively by non-uniform coverage of sample genome and non-uniform distribution of fragment lengths.

Differences inherited from two parents can be presented as another barrier for a smooth sequence assembly, although most studies so far ignore ploidy. Since STRs are polymorphic and we aim to take the diploid nature of DNA into consideration, it requires and additional effort and more detailed analysis to do to it properly.

The assembly problem we are dealing with in the scope of this thesis can be considered as local assembly, which is the assembly procedure of reads that maps to an extremely small part of the genome.

2.2.2 Assembly Methods

NGS assemblers are usually based on three paradigms: (i) constructing contigs by a greed approach, (ii) de Bruijn graph (DBG) based method that rely on a variant of k-mer graph, and (iii) overlap-layout-consensus (OLC) methods that are based on an overlap graph [24]. In the scope of this thesis, we will be focusing on two general approaches that make us of graph data structures.

2.2.2.1 De Bruijn Graphs

k-mer is a substring of length k. k-mer graph, on the other hand, is a graph data structure in which each k-mer is a node and an edge between two nodes implies they are neighbors in the genome sequence. De Bruijn Graph method is based on k-mer graphs [26].

Eulerian walk is a path on a graph that visits every edge exactly once, which does not necessarily exist for every graph. Indeed, Eulerian walk is available on a directed graph if and only if it has at most two semi-balanced nodes (indegree differs from outdegree by 1) and all other nodes are balanced. As long as each k-mer belongs to one read, de Bruijn Graph is going to have an Eulerian walk. Eulerian walk on a de Bruijn Graph gives the reconstruction of original genome sequence [3].

In order to build the de Bruijn Graph for genome assembly, all reads are first chopped into k-mers and their adjacency relation is preserved on the edges. As seen on Figure 2.2, an edge on the de Bruijn Graph corresponds to an overlap (of length k-2) between two k-1-mers. More clearly, it corresponds to a k-mer from the initial sequence [27]. In the ideal scenario, assuming sequencing data is perfect (i.e. error free k-mers with full coverage), the de Bruijn Graph built from reads would contain an Eulerian path, which is trivial to find in linear time [28]. Hence, assembly problem becomes trivial.

Perfect sequencing data is not a good assumption to make, though. There are multiple issues that affect efficiency of DBG based assemblers and require additional computation:

- Repeats: There will always be complex repeat structures in the real data, which may cause multiple Eulerian walks on the de Bruijn Graph. It is not possible to infer the correct path if there are multiple branches. Therefore, DBG assemblers typically make use of reads and mate pairs to resolve branches caused by repetitive regions.

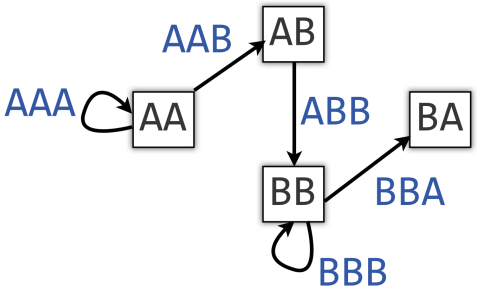

Figure 2.2: de Bruijn graph of AAABBBA with k-mer length 3 (adapted from [3])

- Sequencing errors: Faulty reads may lead to redundant nodes in the de Bruijn Graph. Assemblers usually include a pre-processing step for reads to eliminate erroneous data or they try to eliminate graph edges that is not supported above a threshold value (number of reads).

DBG based assemblers are mostly used on the short reads. Although it does not store individual reads, it keeps the whole sequence in memory, which can be exhaustive on large genomes. The most important benefits of this approach are speed and simplicity.

2.2.2.2 Overlap-Layout-Consensus Method

OLC works similar to DBG in terms of utilizing overlap information about reads [29]. However, it uses this information in a different way.

De Bruijn Graph method is counter-intuitive. Although the goal in DNA sequencing is to achieve longer reads, DBG splits those reads into even shorter k-mers. By doing this it loses connectivity information on reads. As reads get longer, more information is lost. OLC method builds an overlap graph instead where reads are represented as nodes and overlaps as edges [26].

OLC method is a three step approach:

1. Overlaps are computed explicitly by all-against-all, pairwise read aligning. This step is based on shared k-mers among reads and can be tuned with k-mer size, minimum overlap length and minimum percent threshold required to identify overlap parameters [27].

2. In the layout step, the aim to merge reads into contigs. This is achieved by removing transitively-inferable edges that skip one or two edges. This is, in fact, finding a Hamilton path, which is known to be an NP-hard problem. However, there are greedy algorithms that are good approximations and often used in this step.

3. Lastly, a consensus sequence is derived from a profile of the assembled fragments using multiple sequence alignment (MSA). Even though it is theoretically possible to exploit statistical methods and weighted voting strategies to obtain the optimal MSA, there is no known efficient way to compute it [26]. That is why this phase uses progressive pairwise alignments.

In OLC method, the number of nodes kept in the graph is linearly proportional to sequencing depth. It is much less memory intensive compared to DBG. Hence, OLC fits well with the low-coverage long reads; whereas DBG is more suitable for short reads with high coverage [26].

2.3 Tandem Repeats

Tandem repeats are pieces of DNA where a pattern has multiple consecutive copies adjacent to itself. They cover more than 10% of the whole genome [30]. If the repeat unit (pattern) consists of 2 to 6 nucleotides, it can be referred as a short tandem repeat or a microsatellite.

Despite the information they carry being very valuable, STR variations have

not gained enough attention to be a permanent step in genome sequence analysis pipelines. However, several approaches to call them from high throughput sequencing (HTS) data have been proposed in the recent past. Two approaches used in phase 3 data of 1000 Genomes Project (1KG) are lobSTR [11] and RepeatSeq [12], which are released in June 2012 and January 2013 respectively [31].

lobSTR accepts raw reads as input data and only aligns the ones which fully encompass repeat region with its upstream and downstream flanking regions. Aligning all reads using other mainstream aligners is the most cumbersome part of variant calling. By not doing so, lobSTR gains a big advantage. According to their experiments, lobSTR is remarkably faster than conventional aligners; 22 times against BWA, 70 times against Novoalign, and 1000 times against BLAST [11].

Three main steps of lobSTR's algorithm can be summarized as follows:

- Sensing is the part for (i) filtering reads such that only reads containing an STR region with its anchors are left and (ii) finding repeat unit itself. This procedure relies on a signal processing approach. Based on their reportings, they have the highest ratio of reads with non reference allele versus total informative reads.
- Alignment part is to find remaining reads' positions on the reference genome using their anchors and provided reference STR regions.
- Finally allelotype step reports allelic configuration of region using some statistical learning approach.

The other method applied on 1KG, RepeatSeq, takes alignment files from any mainstream aligner as input and realigns them to get a higher chance for minimizing the number of mismatching bases. For each reference repeat region given, it first excludes reads that do not span across the entire repeat region, and then tries to estimate genotypes using Bayesian approach with the following features as prior information: reference length of repeat, repeat unit size and average base quality of read.

Although lobSTR and RepeatSeq have their own ways to handle aligning reads to reference genome, they both suffer from covering only STRs shorter than the read length. In fact, since they use reads that contain anchors, their size limitation is much more strict than the read length (see Chapter 4 for results).

Considering the facts that (i) some individuals or ancestries might have notably larger copy numbers than those seen in general population and (ii) having longer repeat regions might have some biologically critical consequences; limitations of these tools is important enough to mention. As an example, while most humans have 30 CGC repeats in FMRI gene, FMR1 alleles with copy numbers in between 55 and 200 are associated with neurodegeneration and ovarian insufficiency. Or alleles with more than 200 repeats associated with intellectual disability and autistic symptoms [8]. FMR1 gene is not the only one whose expansion can be correlated with abnormal phenotypes. Those cases are certainly off capabilities of lobSTR and RepeatSeq. Therefore, even if read length in sequencing machines have been increasing continuously, discovering longer STR regions is not straightforward and definitely requires more attention.

Chapter 3

Materials and Methods

In this chapter we provide detailed descriptions of each step in our STR characterization pipeline, as visualized in Figure 3.1.

3.1 Overview of the Approach

According to our literature survey, there are approximately 30 assembly tools which use different algorithms and data structures to optimize their resource consumption. A non-exhaustive chronological list of sequence assembly tools in large genomes would be as ABySS, SGA, SOAPdenovo version 2, Minia, DISCOVAR, and BCALM2 [32, 33, 34, 35, 36, 37] - from 2009 up to 2016. Each one has its own advantages over others, mostly focused on parallelization of work and advanced data structures to represent underlying assembly graph.

For example, ABySS was the first software to assembly whole human genome from short reads by distributing a de Bruijn graph across a cluster of nodes [32]. A more recent tool, Minia also uses de Bruijn graphs but reduces memory requirements using Bloom filter, which is a space efficient hash-based data structure to check existence of an element. According to their experiments, peak memory usage for Minia is 5.7Gb, whereas memory consumption of ABySS goes up to 336Gb

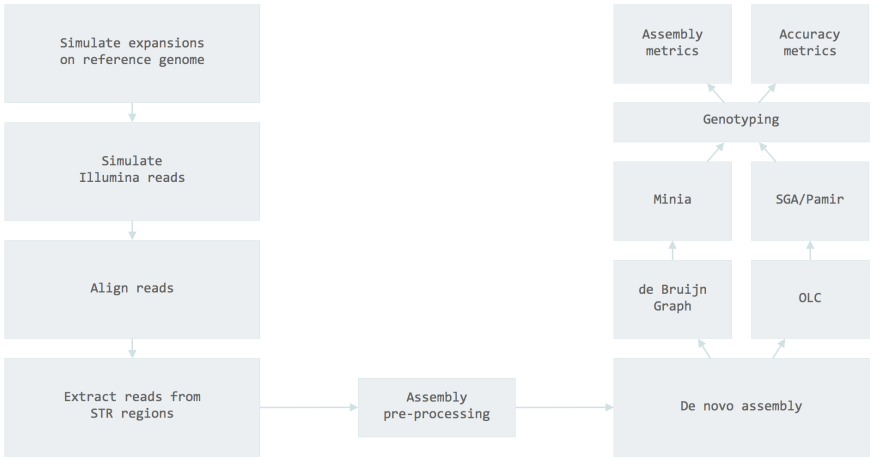

Figure 3.1: STR characterization pipeline

for *de novo* human genome assembly. However, this a trade-off; lower memory usage comes with consequences: execution times for the same two tools are 23h and 15h respectively. Even if they have some differences, percentage of contigs that they could align to reference with at least 95% identity only changes 0.4% between these two [36]. This conclusion applies to tools using overlap-layout-consensus based algorithms as well.

On the other hand, a recent study reveals that an OLC-based method is able to assemble a genome sequence with ten times higher N50[1] compared to a de Bruijn graph approach. Besides, mean length for generated contigs is also ten times longer while the number of contigs are only one-fifth [39]. Hence, accuracy rates or the capability to assembly a genome as a whole might differ a lot.

The main aim of this work is to use sequence assembly methods for regions where we know that there is an STR, based on reference genome, and release a complete pipeline from sample's reads to STR genotype. In the lights of information about genome assembly tools, we have selected three different assemblers to be integrated into our pipeline: SGA, Minia, and Pamir.

[1]The length of the smallest contig x, which makes the ratio of cumulative length of contigs from this length x to largest contig size covers at least 50% of the bases of the assembly. An assembler with high N50 size value is obviously considered to be a high quality assembler [38].

However, since we use assemblers only on relatively short regions, we do not expect choosing a specific assembler tool to have a prominent impact on the end result as long as they use the same underlying method. Local assembly is going to even out their advantages. We will discuss this issue further in the experiments section.

3.2 Simulating Expansions on Reference Genome

In this step, we extract STR regions from reference genome and alter their copy numbers them with simulated events.

3.2.1 Tandem Repeat Regions from TRF

As opposed to other variation callers for structural variations, SNPs or indels, short tandem repeat tools require an input file that contains STR regions in the reference genome. Regions are gathered by using the same reference version with alignment step. Finding tandem repeats in a sequence is a well studied problem in different domains. For example, pattern matching in long strings in computer science is a form of technically the same problem.

Most of the tools in the field use a two-phase approach: search and filtering. In the search phase, the intuitive approach would be dividing the entire sequence into subsequences and comparing every adjacent region with each other for various period sizes. However, this brute force approach is exponential in terms of time complexity. Considering the size of whole human genome with millions of base pairs, it's not a good solution. To avoid this problem, some tools use a heuristic or statistical approach that detects candidate repetitive loci by scanning over the sequence with a small window first, and then tries to find longer repeats [40].

In the second phase, filtering is done to identify and extract biologically significant repeats. Among multiple approaches, we have decided to use Tandem

Repeat Finder [30] for getting reference STR regions, as it has been the most popular tool used by many researchers in large projects so far.

We downloaded TRF output for chromosome 20 of GRCh37, ran with default parameters and filtered it based on our needs so that:

- Repeat matches are perfect, that is there is no change in repeat unit.
- Both copy number and repeat unit length are greater than 3.

In the end, we obtained 1963 STR regions from reference genome for chromosome 20, ready for random event simulation as the next step in the pipeline.

3.2.2 Random Homozygous/Heterozygous Event Simulation

After extracting 1963 STR regions from chromosome 20, we ran multiple versions of event simulations to generate synthetically expanded STR regions. The simulation accepts STR regions (TRF output) and reference genome (FASTA) as input and produces two inflated sequences together with metadata about expanded regions (updated end location, new copy number, genotype, etc).

Workflow in this step for each STR region is as follows:

- Randomly chose between homozygous or heterozygous genotype
- Randomly pick an expansion factor N between 1 and 30
- Identify the STR region in the reference genome
- Inflate the sequence by inserting N more repeat units

If the genotype is homozygous, both alleles have the same expansion, i.e. same sequence. If it is heterozygous, one allele might be the same as reference genome

while the other one has a random expansion, or they both can have different random expansions.

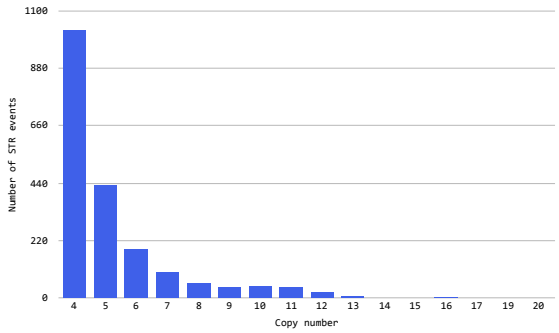

Figure 3.2: STR events per copy number (reference)

The distribution of inflated copy numbers is presented in Figure 3.2 and 3.3.

3.3 Simulation of Illumina Reads Using Mitty

It is quite common in bioinformatics studies to use synthetic reads in the pipeline mainly because either they believe there is no high quality gold standard set to compare results against or they are more keen to try out their methods on a more predictable dataset before jumping into more complicated cases with the real genome. Both reasons are applicable to our study.

Mitty is a tool that generates synthetic next-generation sequencing reads, mainly for Illumina platform [41]. It has five built in read models with changing error profiles, read lengths and fragment size distributions, which you can use out of the box to simulate reads. Besides, in the case that one of the existing models does not match your requirements, there is also an option to build an empirical model based on a provided alignment file.

The two most useful features of this software are:

- It has two different modules as read generation and read corruption, which

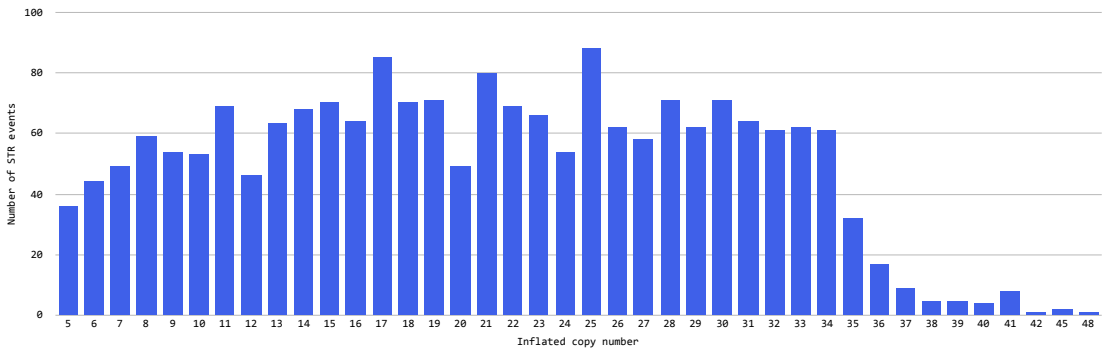

Figure 3.3: STR events per copy number (inflated)

allows you to work with non-corrupted versions of reads, as well.

- It has a "god aligner" module for providing the correct alignment file to eliminate errors that alignment tools might generate.

For all our simulation experiments, built in HiSeq X version 2.5 Garvan read model has been used with varying coverage depths. Some features of the model [41]:

- It has 1.92% mean error rate at first pair, whereas it is 7.46% for the second pair.
- The average fragment size distribution is 350, and it has a normal distribution.
- The read length is 150bp.

3.4 Aligning Reads Using BWA-MEM [1]

Capability to generate sequences grows exponentially faster than our capacity to analyze it, which requires having efficient and accurate tools to utilize such volume of data. Usually, aligning reads to a reference genome is the initial step

of bioinformatics pipeline and it can be seen as one of the most time consuming yet essential step.

In general, sequence alignment software tools work on pipeline of three steps:

1. Indexing the reference genome or the reads (often using heuristics)
2. Identifying the exact or near exact matches
3. Performing dynamic programming alignment to tolerate more complex cases

It would be more useful to divide sequence aligners into groups based on the way they approach the indexing step of this pipeline.

The hash-based methods first identify a subset of possible locations for reads in the reference genome using common k-mers shared by both reads and reference genome through the hash tables. This step is the "seed" part of seed-and-extend strategy. "Extend" strategy is performing more precise approaches (e.g. dynamic programming) to eliminate some of the seeds. Hash-based methods might also differ based on the fact that whether mismatches or indels are tolerated in the seed step [42].

Second group of aligners use Burrows-Wheeler Transform (BWT) based backtracking algorithm for indexing. They align entire set of reads (not just seeds of reads) against substrings sampled from the reference genome. To be able to search for reads efficiently, they build a data structure (such as suffix trees or FM index) that contains all suffixes of the reference genome sequence. This approach efficiently solves the issue with aligning to multiple identical copies in the reference region, where hash table-based approaches do not perform as well. However, keeping all suffixes of the reference genome requires massive amount of memory [42].

BWT is a reversible data compression algorithm, which is used in order to reduce the memory burden of storing all suffixes. Hence, this group of aligners

(BWA, Bowtie, SOAP2, etc.) are able to index very quickly with reasonable memory requirements [42].

In theory, our pipeline is applicable to any BAM file generated by a read aligner that supports soft-clipping. But for this study we have selected BWA-MEM because of its efficiency and comparable accuracy [1]. BWA team also suggested to use BWA-MEM in case the read length is longer than 70bp, which applies to our work.

3.5 Assembly

3.5.1 Assembly Pre-processing

Most of the established sequence alignment tools do not only report start position of reads in the genome but also provide supplementary information about the alignment. Supplementary information may include fields such as CIGAR (indicator of base alignment, deletion, insertion, etc.), MAPQ (mapping quality), and QUAL (query quality).

CIGAR string provides information about insertion and soft clipping events, which makes it useful for our pipeline.

Table 3.1: Alignment with attributes POS=5 and CIGAR=3M1I3M1D5M4S

Reference Position	1	2	3	4	5	6	7		8	9	10	11	12	13	14	15	16	17	18	19	20	21
Reference	C	C	A	T	A	C	T		G	A	A	C	T	G	A	C	T	A	A	A	A	A
Read					A	C	T	A	G	A	A	-	T	G	G	C	T	G	G	C	T	

CIGAR string is a sequence of events with base lengths. Each one indicates an event such as alignment match (M), insertion to the reference (I), soft-clipping (S), deletion from the reference (D), and skipped region from the reference (N).

Soft-clipped sequence may occur in a partially mapped read such that start or

end portions of it is unmatched with the reference genome [43]. Aligner assigns a smaller penalty for soft-clipped regions compared to mismatching bases. This is extremely desirable for finding expansions as we do not want to miss them by penalizing too much. Hence, we make use of CIGAR strings and soft-clipping information when pre-processing data for assembly.

In general, assemblers accept FASTQ files that contain reads to assemble. To build reads data, we have applied following steps for each STR region:

- Using HTSlib [44] reads that map to the STR region were extracted. For each read, we have decided whether it is a reference supporter or not. This is based on its CIGAR string, the number of exact consecutive repeat units it has, and whether these consecutive repeats are placed at both ends of the read.
- For the STR region, if it has enough reference supporters, we have decided that at least one of the alleles has the same copy number with the reference and filtered all reference supporter reads so that we can get a better and clean assembly. Using all reads that map to the STR region would not be fair in the experiments since some STR callers use the reference copy number as a prior information to make better calls.
- Lastly, we have prepared a SAM and a FASTQ file containing reads that passed pre-processing filter.

3.5.2 SGA as the Assembler

SGA (String Graph Assembler) is one of the early adopters of string graph based approach for assembling short reads. In order to decrease memory usage, it utilizes a compressed index for set of sequence reads. Despite being one of the very firsts of its type in genome assembly, it is still widely used in many projects [33].

Each step in SGA pipeline is exposed as a different module, so that each one can be updated independently to keep up with new sequencing technologies or improved algorithms. This way, it also provides a flexibility to skip or rerun any step independently if needed.

For example, since we are dealing with very short regions, scaffolding is not necessary. Indeed, it did not give a different result than contigs in our experiments. That's why we skipped the scaffolding module that uses pairs of the reads to connect contigs to each other.

Submodules we have used in the scope of this work are:

- Preprocess: It filters or trims reads with low quality or ambiguous base calls. However, for our case paired-end mode and filtering based on quality are disabled. This is because we have already applied a filtering and we wanted to be fair to all assemblers in the experiments in case they do not have this module.

- Index: The FM-Index is a data structure that allows efficient searching of a compressed representation of a text and commonly used on top of Burrows-Wheeler transform of the text. BWT compression and FM-indexing together are useful especially for high coverage data. SGA uses them for indexing reads. We run indexing module twice in the pipeline, first with pre-processed raw reads and then with corrected reads.

- Correct: Having a built-in read correction bundle in an assembly software is practical. This bit can be applied to correct base calling errors using two different algorithms: (i) based on k-mer frequencies and (ii) based on inexact overlaps between reads. K-mer based correction, which is preferred in our work, is similar to what other assemblers do [33]. It basically marks a base call as trusted if that position is covered by a k-mer at least c times, where c being a parameter decided by SGA itself. For untrusted positions, it tries alternative bases so that it can replace them with a read of which all positions are trusted. If no alternative is found or there is more than

one alternative with same frequency, the read stays unchanged. We choose k for k-mers as 41 and 63 based on their suggestion for human genome [33].

From 1963 STR regions we worked on there were 173,342 reads in total. 76,090 of these reads were corrected by SGA Correct module. Based on SGA Error Correction module, only 56% of reads are error free, which is less than expected according to average error rates of Illumina platform. x-axis of Figure 3.4 shows edit distances between reads and their corrected versions; whereas y-axis represents the frequencies. Most of the reads have 1, 2, or 3 mismatches, while the highest edit distance with a read and its processed version is 11. More interesting comparison as a future work might be looking at the non-STR regions to compare the error rate reported by SGA.

- Filter: This step filters reads having exact matches (including reverse complements) before starting to build the graph.
- Overlap and Assemble: These last two parts construct the graph using filtered and corrected reads and try to find a path in the graph to report as a contig. When deciding overlapping reads, the parameter that defines whether reads are neighbours in the graph or not (minimum overlap required between two reads) is tuned as 70. Even if it does not apply to local assembly, their improvement on graph building, where they construct assembly graph locally around each read and merge them later instead of loading full graph, has many advantages for whole genome assembly. Based on parameters we have set, a path to be reported as a contig should at least have 90 bases across its nodes.

3.5.3 Minia as the Assembler

As opposed to modular approach of SGA, Minia is presented as a whole package. While this makes providing specific configuration for sub-steps quite complicated, it is much more straightforward to use the tool as a complete assembler.

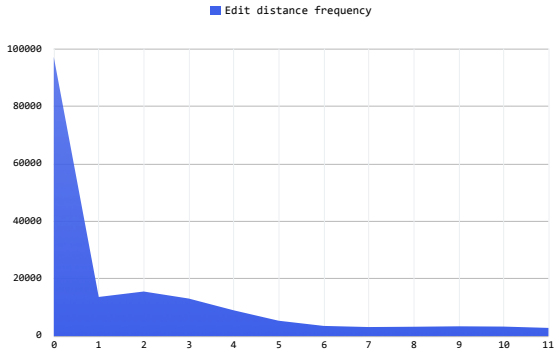

Figure 3.4: Frequency of edit distance between reads and their corrected versions

As mentioned in Section 2.2.2, de Bruijn graph based assemblers are absolutely less time consuming compared to Overlap-Layout-Consensus graph based methods. However, speed comes with an huge memory burden. A de Bruijn graph based assembler without any additional data structure or algorithmic optimization, requires nearly 15Gb of memory to store just the nodes alone when working on the human genome (k-mer size being 27) [35]. There have been studies to tackle memory consumption of de Bruijn graph, primarily focused on (i) enabling distributed computation on the graph to reduce memory usage per node, (ii) storing less data in the graph by omitting read locations and paired end information, (iii) greedy algorithms to compute local assemblies around related sequences [35]. Minia, on the other hand, makes use of Bloom filter to improve memory efficiency of de Bruijn graphs.

Bloom filter is a hash-based data structure that is specifically designed for checking existence of element in a set and to be space efficient. By its design, Bloom filter does not guarantee that element exists in the set even if lookup operation returns true. Whilst, a false return value indicates that element is definitely not in the set. Therefore, Bloom filter may have false positives, probability of which is directly proportional to the size of the set [45]. In order to overcome false positives coming from the Bloom filter, Minia utilizes an additional data structure to keep track of critical false positive k-mers.

After placing all k-mers to the Bloom filter, Minia iterates over each k-mer

and searches for its neighbours in the Bloom filter. If the Bloom filter returns negative, no edge is placed in between. If it returns positive, in order to avoid false positives it performs an additional lookup to auxiliary data structure. If it confirms the positive value, an edge is placed between k-mer nodes.

Once the graph is constructed, breadth first traversal returns contigs. Minia does not provide a module that supports scaffolding using paired end information.

According to the developers of Minia, optimal k-mer size is determined as 47 for chromosome 14 assembly with 100bp reads [35]. We decided k-mer size to be 63 since reads in our work are longer (150bp).

3.5.4 Pamir as the Assembler

Pamir is a software for novel insertion detection [46]. It popped into our radar because novel insertions can only be detected using *de novo* assembly. We have extracted custom assembly module to be able to use it as a standalone application.

Pamir's in-house assembly algorithm is based on OLC with an important tweak. It takes strand-specific information into account, which can be inferred from the mapping information. Since it knows which strand to pick, duplicate calculation for reverse complement can be avoided. That is why we thought that it would be reasonably more time efficient compared to conventional assembly methods. Experiments verified our initial view, which will be discussed further in Chapter 5.

3.6 Genotyping

Genotyping is the process of determining the genetic constitution of an individual, which has been inherited from the parents. Since human genome is diploid, we have two alleles for every part of our genome. The output of this step is going to

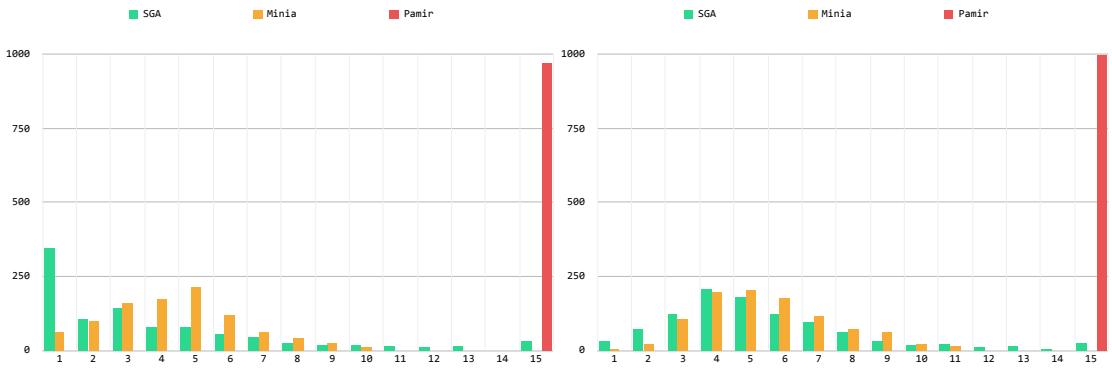

Figure 3.5: Produced number of contigs for homozygous and heterozygous events, respectively

be one of the following:

- Heterozygous: Each parent contributes different alleles. To call it as heterozygous, one of the alleles must be different from the reference allele. The other one is not necessarily different than reference allele.
- Homozygous: Parents contribute the same allele, which is always different from the reference allele.

As the last step in our pipeline, we perform genotyping on the reported contigs.

Ideally an assembler should produce a single contig for an homozygous event, and two contigs for an heterozygous event. Most of the time assemblers produce more contigs than the number of different alleles. Figure 3.5 reports the number of contigs produced by assembler in homozygous and heterozygous events, respectively. Figure 3.5 shows that all three assemblers have struggled resolving reads into the correct number of contigs while SGA performed relatively well. Pamir have always produced way too many contigs than 15, which is due to the fact that it does not apply post-processing. Minia has been more successful compared to Pamir, whilst it has failed to resolve regions that are larger than k-mer size.

In order to decide which contig is more accurate, we take sequences reported by reads into consideration. We first tried to align reads to all contigs and select

the contig with the highest mapping score using classic Smith–Waterman local alignment for each read. However, since Smith-Waterman local alignment did not scale well in terms of time complexity, we have decided to apply BWA-MEM after considering the facts explained in Section 3.4.

Upon obtaining the most supported two contigs, we infer genotype based on the following rules:

- If read support ratio of these contigs is too low, we conclude that it is an heterozygous event.
- If one of the contigs is supported by most of the reads, while the other one has a low support, we conclude that it is an homozygous event.

As a side note, depending on whether reference allele is available in the region after pre-processing filter or not, copy number for one of the alleles can be obtained directly.

Chapter 4

Experimental Results

In the experiments, we wanted include a calling method that does not depend on assembly as a comparison. Based on a recent study which evaluates non-assembly methods for detecting STRs, HipSTR, lobSTR, and RepeatSeq are leading the competition [47]. Although HipSTR outperformed others in their experiments, it is designed for population data and requires high coverage to detect *de novo* mutations, which makes it unsuitable for our study. Hence, we included lobSTR in our experiments.

There are three experiments presented in this chapter, each focusing on a different aspect:

- First experiment is to evaluate methods based on their performance on separate genotypes.
- Second one is to analyze how sequence coverage impacts assembly based callers.
- The last experiment assesses the importance of pre-processing and including flanking regions.

4.1 Genotype Based Performance

In this experiment we have simulated events based on 1963 STR regions retrieved from TRF with a sequencing coverage 60X. Each region was inflated by a random amount of copy numbers (between 1 and 30) and randomly assigned a genotype. In all heterozygous events, alleles coming from each parent have different copy number and both of them are different from reference allele.

True positive rates[1] of methods are reported as a distribution over repeat region size (i.e. copy number times repeat unit length). In order to tolerate randomness and volatility in short spans, events are grouped into bins based on repeat region size. Each bin contains events from a range of five size values. For instance, bin-20 contains all events having a repeat region longer than or equal to 20nt and shorter than 25nt.

Table 4.1: True positive rates for all events

	Homozygous			Heterozygous					Overall				
Tool	**Hit**	**Total**	**Hit Ratio**	**Hit**	**PHit**	**Total**	**Hit Ratio**	**PHit Ratio**	**Hit**	**PHit**	**Total**	**Hit Ratio**	**PHit Ratio**
Minia	130	970	0.134	0	0	993	0.000	0.000	130	130	1963	0.066	0.066
SGA	514	970	0.530	108	642	993	0.109	0.647	622	1156	1963	0.317	0.589
lobSTR	339	970	0.349	79	79	993	0.080	0.080	418	418	1963	0.213	0.213
Pamir	187	970	0.193	25	419	993	0.025	0.422	212	606	1963	0.108	0.309

In distribution charts, x-axis represents bins while y-axis represents true positive rate.

To report some statistics about the events:

- As presented in Table 4.1, genotype split in 1963 events is 970 homozygous to 993 heterozygous.
- Bin-115 is the largest bin containing 128 STR events (see Figure 4.4).
- Shortest and longest repeat regions are composed of 20nt and 220nt, respectively.

[1]Number of hits / Number of events

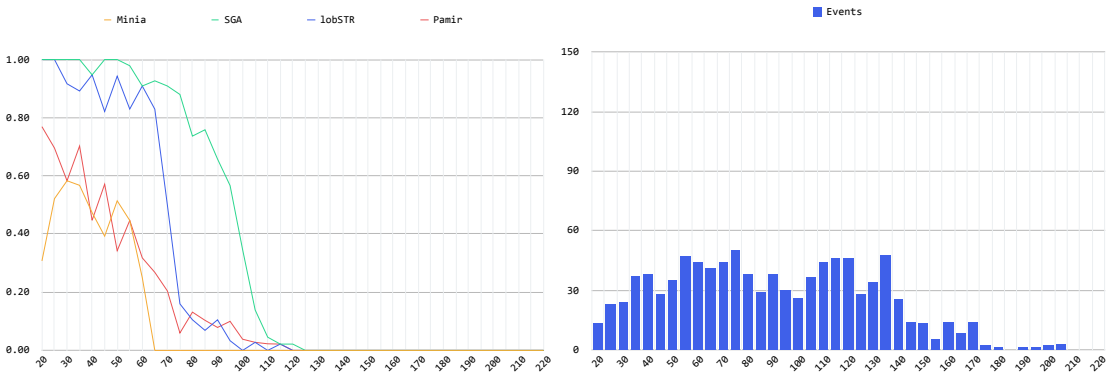

Figure 4.1: (Left) True positive rates for homozygous events vs. region length and (Right) number of homozygous events vs. region length

As seen in Figure 4.1, SGA is clearly the most successful at calling homozygous STRs, followed by lobSTR. It does not only cover the widest range in terms of repeat region length but also dominates other approaches in almost every bin, which reveals that read length is not a major barrier for SGA unlike lobSTR.

Figure 4.2 and Figure 4.3 present true positive hit and partial hit rates on heterozygous events. On a heterozygous STR event, if caller is able to determine copy number for both alleles, it is considered as a hit. If it calls only one of the events, it is a partial hit. SGA is still definitely more effective at detecting at least one of the alleles correctly. However, especially in shorter repeat regions, lobSTR performs better by a large margin at calling events on both alleles. For regions longer than 65nt, there is a significant drop in lobSTR's effectiveness.

In general, considering lobSTR's 0.349 hit ratio in homozygous events versus 0.080 in heterozygous cases, it has a substantial disadvantage in calling heterozygous STRs. Minia and Pamir have underperformed in all cases (see Figure 4.4)

Table 4.2: Genotype calls an all events

	Homozygous (970 events)		Heterozygous (993 events)	
Tool	**Correct Call**	**Correct Call Ratio**	**Correct Call**	**Correct Call Ratio**
Minia	474	0.489	560	0.564
SGA	725	0.747	517	0.521
lobSTR	359	0.370	76	0.077
Pamir	432	0.445	628	0.632

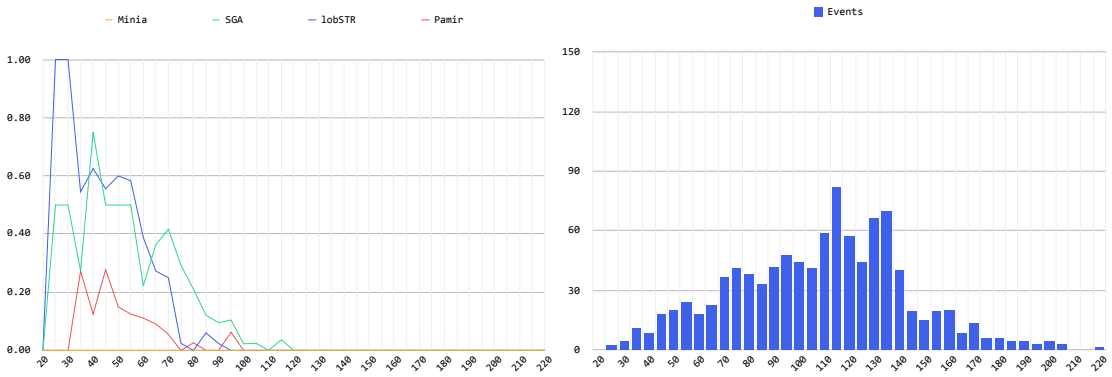

Figure 4.2: (Left) True positive rates for heterozygous events vs. region length and (Right) number of heterozygous events vs. region length

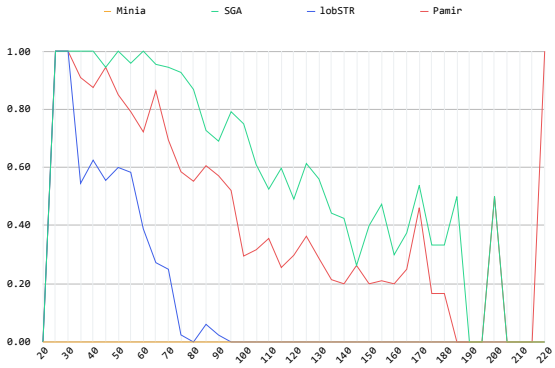

Figure 4.3: True positive rates (partial) for heterozygous events vs. region length

with similar true positive rates (0.066 and 0.108, respectively). Although all approaches have at least 4-times lower rates in heterozygous STRs, Minia (and de Bruijn graph) failed to detect even a single one.

We have also examined their abilities to annotate genotype of the STR region without taking the reported copy numbers into account. Genotype annotation accuracies are presented in Table 4.2. lobSTR reported only 76 of 993 heterozygous events as heterozygous, while Pamir is the best performing with 628. On the other hand, SGA is better at annotating homozygous regions. On the basis of this results, it is safe to say that assembly based methods are superior to lobSTR, which proves that discovering the existence of different alleles are easier with local assembly. To sum up, none of the methods help us genotype an STR region confidently.

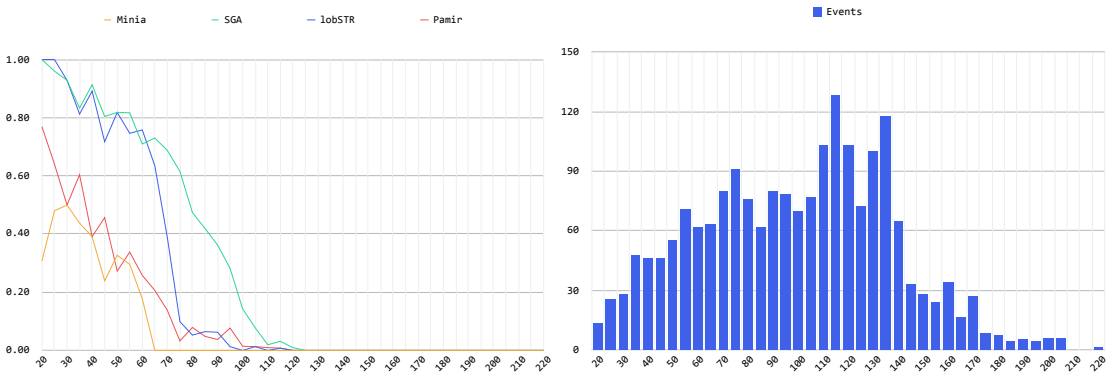

Figure 4.4: (Left) True positive rates for all events vs. region length and (Right) number of events vs. region length

4.2 Coverage Tests

There are 1963 simulated STR regions with a 966/997 split between genotypes (homozygous/heterozygous) in this experiment. However, for heterozygous events, this time one of the parents has the same allele with the reference region.

In this group of runs, we aimed to assess the impact of sequence coverage on assembly based callers. We simulated sequence data from the altered genome with multiple sequence haploid coverage, at 40X, 60X, and 80X. True positive rates for each caller with different coverages are reported in Figure 4.5, 4.6, 4.7, and 4.8.

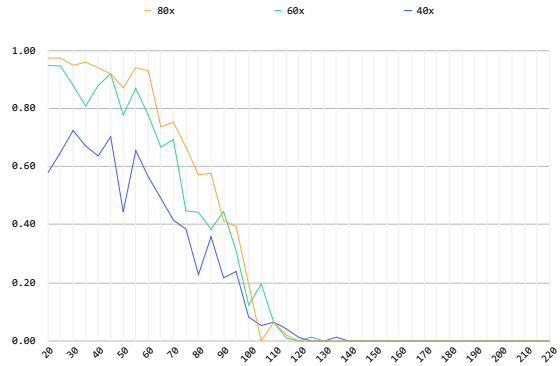

Figure 4.5: True positive rates of SGA with different coverages vs. region length

As anticipated, higher coverage helps all approaches perform better. However,

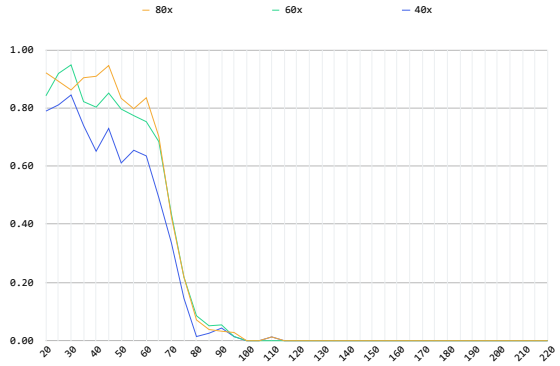

Figure 4.6: True positive rates of lobSTR with different coverages vs. region length

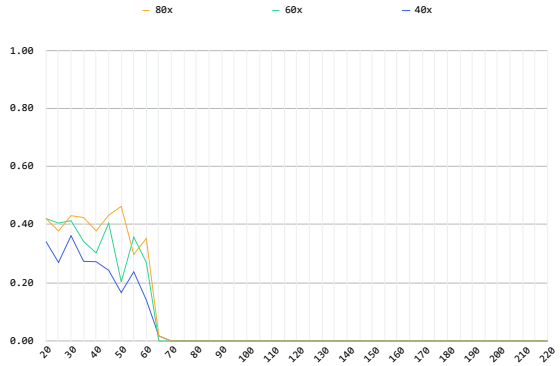

Figure 4.7: True positive rates of Minia with different coverages vs. region length

the percentage of increase in recall is different between 40X to 60X and 60X to 80X. It slows down, which suggests that we are close to a saturation coverage value. For example, SGA were able to call 1.39 times as more events with 60X coverage when compared to 40X coverage; whereas the rate grows 1.12 times when coverage went up from 60X to 80X.

Reported in Table 4.3, SGA and lobSTR handle low coverage relatively well. Pamir hits jump 300% percent with 60X coverage in comparison with 40X. Having higher coverage data is more crucial for assembly based methods. Hence lobSTR's efficiency is not dramatically affected by the coverage.

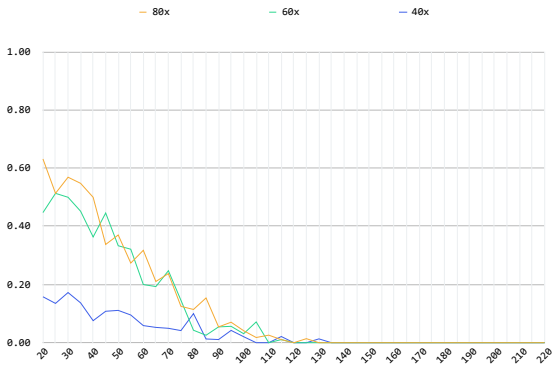

Figure 4.8: True positive rates of Pamir with different coverages vs. region length

De Bruijn graph fails once again. The range of region lengths that Minia is able to call is the smallest among all callers. It could not detect any STR region that is longer than 70nt.

Hit rate lines for Pamir at 60X and 80X coverage almost overlap in each bin, which signals that it does not depend on high coverage data.

Table 4.3: True positive rates for 40X, 60X, and 80X coverage

	40X		**60X**		**80X**	
Tool	**Hit**	**Hit Ratio**	**Hit**	**Hit Ratio**	**Hit**	**Hit Ratio**
Minia	142	0.072	194	0.099	224	0.114
SGA	568	0.290	793	0.404	892	0.455
lobSTR	487	0.248	589	0.300	615	0.313
Pamir	92	0.047	289	0.147	331	0.169

As the repeat regions get longer, it becomes harder to assemble them. Besides, higher coverage values increase the variation. Considering these two facts, as seen in Figure 4.5, for STR regions longer than 100nt, SGA performs worse with high coverage data (80X).

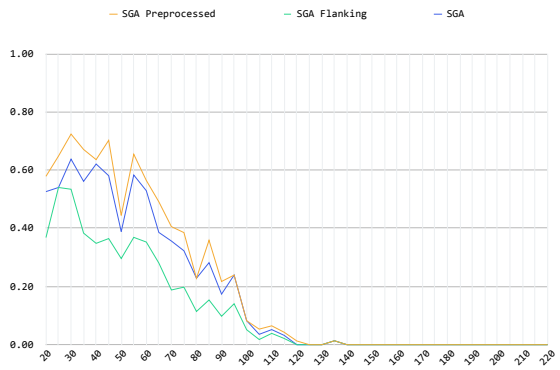

Figure 4.9: True positive rates of SGA with different setups vs. region length

4.3 Assembly Pre-processing and Flanking Regions

In this experiment, we have tested a single tool with a different pipeline and a configuration. SGA was clearly the best caller in all runs. We wanted to assess the possibility of increasing its sensitivity even more. The same set of STR regions is inputted to:

- An SGA pipeline with pre-processing, same as previous experiments. Pre-processing includes, as explained in Section assembly preprocess, checking reference supporter reads for STR regions and assigning reference copy number to at least one of the alleles and filtering out these reads from the assembly.
- A regular SGA pipeline without pre-processing step.
- A regular SGA pipeline with flanking regions included from both ends of the STR region.

True positive rates are reported in Figure 4.9. First of all, it is certain that including flanking regions didn't help STR variant calling at all and indeed it made it worse. This is probably caused by STR regions becoming more complicated as

they get longer. Reads from STR regions themselves cover the flanking regions enough to identify the region. Pre-processed assembly, on the other hand, gave SGA's performance a slight boost in every bin.

Chapter 5

Discussions

We addressed the problem of short tandem repeats not being a part of large scale projects due to characterization limits. We proposed an end-to-end solution for STR characterization using local assembly that is suitable for WGS pipelines.

We concluded that String Graph Assembler is better than both other assembly based tools and state-of-art STR callers in estimating repeat numbers. However, since it uses a relatively costly OLC approach, it is lagging far behind the rest of the methods in terms of time complexity. lobSTR and Pamir have comparable runtimes and are considerably faster than Minia. At this state, we greatly value correct variant detection over runtime complexity.

State-of-art STR callers that do not make use of local assembly, expect STR regions to be covered with flanking regions. All experiments supported that for reads of length 150 base pairs, discoverable STR region is up to 70 base pairs for them. This may be overcame with the help of new long-read sequencing methods, though. However, OLC based assembly methods will definitely benefit from longer reads, more than other STR callers or de Bruijn Graph based approaches [26].

We observed from coverage tests that higher coverage boosts the sensitivity of every approach. However, it does not linearly improve the number of STR regions

detected, which indicates a potential upper bound of sequencing read lengths of 150bp.

5.1 Future Work

There are two main directions that we can take to further improve assembly based STR calling pipelines.

First, to improve false negative rate we can include one end anchored reads, which are the reads where only one end-read is mapped onto the reference genome, using paired-end information. Currently, we use reads that map to an STR region. However, since STR regions are usually much shorter than the fragment size, there is not going to be a case where both paired-ends do not map to the reference genome due to an expansion. Hence, one end anchored reads should be sufficient to call STR variations.

Secondly, to achieve an accurate repeat number estimation we can improve alignment quality of sequence reads. This not only applies to assembly based pipelines but also applies to tools depending on alignment step, such as lobSTR. A recent study proposes a dynamic programming approach for the realignment step, where repeat patterns in STR regions are given as prior knowledge, and repeat patterns are used multiple times in the realignment process in order to get more accurate STR containing reads [48]. Since assembly based pipeline that we propose is an alignment based approach, re-alignment should help get more accurate calls.

Bibliography

[1] H. Li, "Aligning sequence reads, clone sequences and assembly contigs with bwa-mem," *arXiv preprint arXiv:1303.3997*, 2013.

[2] S. El-Metwally, T. Hamza, M. Zakaria, and M. Helmy, "Next-generation sequence assembly: four stages of data processing and computational challenges," *PLoS Comput Biol*, vol. 9, no. 12, p. e1003345, 2013.

[3] D. Gifford, "Foundations of Computational and Systems Biology: Genome Assembly." Available: `https://ocw.mit.edu/courses/biology`, 2014.

[4] P. Gill, "Role of short tandem repeat dna in forensic casework in the uk-past, present, and future perspectives," *Biotechniques*, vol. 32, no. 2, pp. 366–385, 2002.

[5] T. Willems, M. Gymrek, G. Highnam, D. Mittelman, Y. Erlich, . G. P. Consortium, *et al.*, "The landscape of human str variation," *Genome research*, vol. 24, no. 11, pp. 1894–1904, 2014.

[6] K. Doi, T. Monjo, P. H. Hoang, J. Yoshimura, H. Yurino, J. Mitsui, H. Ishiura, Y. Takahashi, Y. Ichikawa, J. Goto, *et al.*, "Rapid detection of expanded short tandem repeats in personal genomics using hybrid sequencing," *Bioinformatics*, vol. 30, no. 6, pp. 815–822, 2014.

[7] P. Kozlowski, K. Sobczak, and W. J. Krzyzosiak, "Trinucleotide repeats: triggers for genomic disorders?," *Genome medicine*, vol. 2, no. 4, p. 29, 2010.

[8] K. Usdin, "The biological effects of simple tandem repeats: lessons from the repeat expansion diseases," *Genome research*, vol. 18, no. 7, pp. 1011–1019, 2008.

[9] T. J. Treangen and S. L. Salzberg, "Repetitive dna and next-generation sequencing: computational challenges and solutions," *Nature Reviews Genetics*, vol. 13, no. 1, pp. 36–46, 2012.

[10] . G. P. Consortium *et al.*, "An integrated map of genetic variation from 1,092 human genomes," *Nature*, vol. 491, no. 7422, pp. 56–65, 2012.

[11] M. Gymrek, D. Golan, S. Rosset, and Y. Erlich, "lobstr: a short tandem repeat profiler for personal genomes," *Genome research*, vol. 22, no. 6, pp. 1154–1162, 2012.

[12] G. Highnam, C. Franck, A. Martin, C. Stephens, A. Puthige, and D. Mittelman, "Accurate human microsatellite genotypes from high-throughput resequencing data using informed error profiles," *Nucleic acids research*, p. gks981, 2012.

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

[14] J. M. Heather and B. Chain, "The sequence of sequencers: The history of sequencing dna," *Genomics*, vol. 107, no. 1, pp. 1–8, 2016.

[15] O. Morozova and M. A. Marra, "Applications of next-generation sequencing technologies in functional genomics," *Genomics*, vol. 92, no. 5, pp. 255–264, 2008.

[16] N. H. G. R. Institute, "The Cost of Sequencing a Human Genome." Available: `https://www.genome.gov/sequencingcosts/`. Last visited on March 2017.

[17] S. Goodwin, J. D. McPherson, and W. R. McCombie, "Coming of age: ten years of next-generation sequencing technologies," *Nature Reviews Genetics*, vol. 17, no. 6, pp. 333–351, 2016.

[18] F. S. Collins, M. Morgan, and A. Patrinos, "The human genome project: lessons from large-scale biology," *Science*, vol. 300, no. 5617, pp. 286–290, 2003.

[19] E. Singer, "Craig venter's genome," *Technology Review*, vol. 1, 2007.

[20] E. Singer, "The $2 million genome," *Technology Review*, vol. 1, 2007.

[21] Y. Guo, Y. Dai, H. Yu, S. Zhao, D. C. Samuels, and Y. Shyr, "Improvements and impacts of grch38 human reference on high throughput sequencing data analysis," *Genomics*, 2017.

[22] G. G. Silva, B. E. Dutilh, T. D. Matthews, K. Elkins, R. Schmieder, E. A. Dinsdale, and R. A. Edwards, "Combining de novo and reference-guided assembly with scaffold_builder," *Source code for biology and medicine*, vol. 8, no. 1, p. 23, 2013.

[23] A. J. de Koning, W. Gu, T. A. Castoe, M. A. Batzer, and D. D. Pollock, "Repetitive elements may comprise over two-thirds of the human genome," *PLoS Genet*, vol. 7, no. 12, p. e1002384, 2011.

[24] K. M. SteinBerg, V. A. Schneider, C. Alkan, M. J. Montague, W. C. Warren, D. M. Church, and R. K. Wilson, "Building and improving reference genome assemblies," *Proceedings of the IEEE*, vol. 105, no. 3, pp. 422–435, 2017.

[25] M. A. Quail, M. Smith, P. Coupland, T. D. Otto, S. R. Harris, T. R. Connor, A. Bertoni, H. P. Swerdlow, and Y. Gu, "A tale of three next generation sequencing platforms: comparison of ion torrent, pacific biosciences and illumina miseq sequencers," *BMC genomics*, vol. 13, no. 1, p. 341, 2012.

[26] J. R. Miller, S. Koren, and G. Sutton, "Assembly algorithms for next-generation sequencing data," *Genomics*, vol. 95, no. 6, pp. 315–327, 2010.

[27] Y. He, Z. Zhang, X. Peng, F. Wu, and J. Wang, "De novo assembly methods for next generation sequencing data," *Tsinghua Science and Technology*, vol. 18, no. 5, pp. 500–514, 2013.

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

[29] Z. Li, Y. Chen, D. Mu, J. Yuan, Y. Shi, H. Zhang, J. Gan, N. Li, X. Hu, B. Liu, *et al.*, "Comparison of the two major classes of assembly algorithms: overlap–layout–consensus and de-bruijn-graph," *Briefings in functional genomics*, vol. 11, no. 1, pp. 25–37, 2012.

[30] G. Benson *et al.*, "Tandem repeats finder: a program to analyze dna sequences," *Nucleic acids research*, vol. 27, no. 2, pp. 573–580, 1999.

[31] IGSR, "Short Tandem Repeats added to the 1000 Genomes Release #ASHG14." Available: `http://www.internationalgenome.org/announcements/short-tandem-repeats-added-1000-genomes-release-ashg14-2014-10-18/`. Last visited on March 2017.

[32] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, and I. Birol, "Abyss: a parallel assembler for short read sequence data," *Genome research*, vol. 19, no. 6, pp. 1117–1123, 2009.

[33] J. T. Simpson and R. Durbin, "Efficient de novo assembly of large genomes using compressed data structures," *Genome research*, vol. 22, no. 3, pp. 549–556, 2012.

[34] R. Luo, B. Liu, Y. Xie, Z. Li, W. Huang, J. Yuan, G. He, Y. Chen, Q. Pan, Y. Liu, *et al.*, "Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler," *Gigascience*, vol. 1, no. 1, p. 18, 2012.

[35] R. Chikhi and G. Rizk, "Space-efficient and exact de bruijn graph representation based on a bloom filter," *Algorithms for Molecular Biology*, vol. 8, no. 1, p. 22, 2013.

[36] N. I. Weisenfeld, S. Yin, T. Sharpe, B. Lau, R. Hegarty, L. Holmes, B. Sogoloff, D. Tabbaa, L. Williams, C. Russ, *et al.*, "Comprehensive variation discovery in single human genomes," *Nature genetics*, vol. 46, no. 12, pp. 1350–1355, 2014.

[37] R. Chikhi, A. Limasset, and P. Medvedev, "Compacting de bruijn graphs from sequencing data quickly and in low memory," *Bioinformatics*, vol. 32, no. 12, pp. i201–i208, 2016.

[38] M. M. Abbas, Q. M. Malluhi, and P. Balakrishnan, "Assessment of de novo assemblers for draft genomes: a case study with fungal genomes," *BMC genomics*, vol. 15, no. 9, p. S10, 2014.

[39] Y. Cherukuri and S. C. Janga, "Benchmarking of de novo assembly algorithms for nanopore data reveals optimal performance of olc approaches," *BMC genomics*, vol. 17, no. 7, p. 507, 2016.

[40] K. G. Lim, C. K. Kwoh, L. Y. Hsu, and A. Wirawan, "Review of tandem repeat search tools: a systematic approach to evaluating algorithmic performance," *Briefings in bioinformatics*, p. bbs023, 2012.

[41] SBG, "Mitty: Seven Bridges Genomics aligner/caller debugging and analysis tool." Available: `https://github.com/sbg/Mitty`. Last visited on March 2017.

[42] J. Shang, F. Zhu, W. Vongsangnak, Y. Tang, W. Zhang, and B. Shen, "Evaluation and comparison of multiple aligners for next-generation sequencing data analysis," *BioMed research international*, vol. 2014, 2014.

[43] S. Suzuki, T. Yasuda, Y. Shiraishi, S. Miyano, and M. Nagasaki, "Clipcrop: a tool for detecting structural variations with single-base resolution using soft-clipping information," *BMC bioinformatics*, vol. 12, no. 14, p. S7, 2011.

[44] samtools, "HTSlib: C library for high-throughput sequencing data formats." Available: `https://github.com/samtools/htslib`. Last visited on March 2017.

[45] B. Raphael and J. Tang, *Algorithms in Bioinformatics: 12th International Workshop, WABI 2012, Ljubljana, Slovenia, September 10-12, 2012. Proceedings*, vol. 7534. Springer, 2012.

[46] S. C. B. Lab, "Pamir: Insertion Discovery Tool for Whole Genome Sequencing Data." Available: `https://bitbucket.org/compbio/pamir`. Last visited on March 2017.

[47] T. Willems, D. Zielinski, A. Gordon, M. Gymrek, and Y. Erlich, "Genome-wide profiling of heritable and de novo str variations," *bioRxiv*, p. 077727, 2016.

[48] K. Kojima, Y. Kawai, K. Misawa, T. Mimori, and M. Nagasaki, "Str-realigner: a realignment method for short tandem repeat regions," *BMC genomics*, vol. 17, no. 1, p. 991, 2016.

Appendix A

Experimental Results

Table A.1: True positive rates for all events (Section 4.1) - 1

	Minia				SGA				lobSTR				Pamir			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
20	0	0	1	0.000	1	0	0	1.000	1	0	0	1.000	1	0	0	1.000
22	1	0	2	0.333	3	0	0	1.000	3	0	0	1.000	3	0	0	1.000
23	2	0	2	0.500	4	0	0	1.000	4	0	0	1.000	2	0	2	0.500
24	1	0	4	0.200	5	0	0	1.000	5	0	0	1.000	4	0	1	0.800
26	2	0	3	0.400	5	0	0	1.000	5	0	0	1.000	3	0	2	0.600
27	5	0	6	0.455	11	0	0	1.000	11	0	0	1.000	9	0	2	0.818
28	4	0	4	0.500	7	1	0	0.875	8	0	0	1.000	4	2	2	0.500
29	1	0	0	1.000	1	0	0	1.000	1	0	0	1.000	0	0	1	0.000
30	3	0	4	0.429	7	0	0	1.000	7	0	0	1.000	1	2	4	0.143
31	4	0	3	0.571	7	0	0	1.000	6	0	1	0.857	6	0	1	0.857
32	2	0	1	0.667	3	0	0	1.000	3	0	0	1.000	3	0	0	1.000
33	2	0	1	0.667	3	0	0	1.000	2	0	1	0.667	1	0	2	0.333
34	3	0	5	0.375	6	2	0	0.750	8	0	0	1.000	3	2	3	0.375
35	3	0	6	0.333	6	3	0	0.667	7	0	2	0.778	6	2	1	0.667
36	2	0	5	0.286	5	2	0	0.714	6	0	1	0.857	4	2	1	0.571
37	1	0	1	0.500	2	0	0	1.000	2	0	0	1.000	2	0	0	1.000
38	6	0	7	0.462	12	1	0	0.923	12	0	1	0.923	8	1	4	0.615
39	9	0	8	0.529	15	2	0	0.882	12	0	5	0.706	9	2	6	0.529
40	1	0	2	0.333	3	0	0	1.000	3	0	0	1.000	1	0	2	0.333
41	1	0	2	0.333	3	0	0	1.000	2	0	1	0.667	1	1	1	0.333
42	5	0	8	0.385	13	0	0	1.000	12	0	1	0.923	5	1	7	0.385
43	10	0	10	0.500	18	1	1	0.900	19	0	1	0.950	8	2	10	0.400
44	1	0	6	0.143	5	1	1	0.714	5	0	2	0.714	3	2	2	0.429
45	0	0	1	0.000	0	1	0	0.000	0	0	1	0.000	1	0	0	1.000

Table A.2: True positive rates for all events (Section 4.1) - 2

	Minia				SGA				lobSTR				Pamir			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
46	3	0	5	0.375	7	0	1	0.875	8	0	0	1.000	3	1	4	0.375
47	3	0	12	0.200	13	2	0	0.867	13	0	2	0.867	8	6	1	0.533
48	4	0	13	0.235	15	2	0	0.882	8	0	9	0.471	6	3	8	0.353
49	1	0	4	0.200	2	3	0	0.400	4	0	1	0.800	3	2	0	0.600
50	3	0	7	0.300	9	1	0	0.900	8	0	2	0.800	3	2	5	0.300
51	7	0	18	0.280	17	8	0	0.680	20	0	5	0.800	5	7	13	0.200
52	7	0	7	0.500	14	0	0	1.000	13	0	1	0.929	5	2	7	0.357
53	0	0	2	0.000	2	0	0	1.000	2	0	0	1.000	0	2	0	0.000
54	1	0	3	0.250	3	1	0	0.750	2	0	2	0.500	2	1	1	0.500
55	5	0	13	0.278	15	3	0	0.833	12	0	6	0.667	4	8	6	0.222
56	2	0	9	0.182	9	1	1	0.818	8	0	3	0.727	3	3	5	0.273
57	1	0	0	1.000	1	0	0	1.000	1	0	0	1.000	0	0	1	0.000
58	5	0	8	0.385	12	0	1	0.923	10	0	3	0.769	6	1	6	0.462
59	8	0	20	0.286	21	7	0	0.750	22	0	6	0.786	11	4	13	0.393
60	0	0	10	0.000	7	3	0	0.700	6	0	4	0.600	3	2	5	0.300
61	0	0	3	0.000	2	1	0	0.667	2	0	1	0.667	0	1	2	0.000
62	1	0	5	0.167	5	1	0	0.833	5	0	1	0.833	2	1	3	0.333
63	6	0	14	0.300	15	2	3	0.750	16	0	4	0.800	5	2	13	0.250
64	4	0	19	0.174	15	7	1	0.652	18	0	5	0.783	6	5	12	0.261
66	0	0	20	0.000	16	3	1	0.800	15	0	5	0.750	6	3	11	0.300
67	0	0	28	0.000	19	8	1	0.679	17	0	11	0.607	5	10	13	0.179
68	0	0	11	0.000	8	1	2	0.727	5	0	6	0.455	1	3	7	0.091
69	0	0	4	0.000	3	1	0	0.750	3	0	1	0.750	1	1	2	0.250
70	0	0	10	0.000	9	1	0	0.900	6	0	4	0.600	3	1	6	0.300
71	0	0	32	0.000	23	5	4	0.719	18	0	14	0.563	5	9	18	0.156
72	0	0	23	0.000	11	10	2	0.478	3	0	20	0.130	3	10	10	0.130
73	0	0	1	0.000	1	0	0	1.000	0	0	1	0.000	0	0	1	0.000
74	0	0	14	0.000	11	3	0	0.786	4	0	10	0.286	0	3	11	0.000
75	0	0	27	0.000	19	7	1	0.704	4	0	23	0.148	0	9	18	0.000
76	0	0	18	0.000	9	6	3	0.500	3	0	15	0.167	0	7	11	0.000
77	0	0	1	0.000	1	0	0	1.000	0	0	1	0.000	0	0	1	0.000
78	0	0	15	0.000	7	7	1	0.467	0	0	15	0.000	2	3	10	0.133
79	0	0	30	0.000	20	6	4	0.667	2	0	28	0.067	1	5	24	0.033
80	0	0	20	0.000	12	4	4	0.600	0	0	20	0.000	2	8	10	0.100
81	0	0	2	0.000	1	1	0	0.500	0	0	2	0.000	0	1	1	0.000
82	0	0	13	0.000	7	4	2	0.538	1	0	12	0.077	2	2	9	0.154
83	0	0	29	0.000	9	13	7	0.310	2	0	27	0.069	1	5	23	0.034
84	0	0	12	0.000	7	3	2	0.583	1	0	11	0.083	1	4	7	0.083
86	0	0	14	0.000	8	4	2	0.571	3	0	11	0.214	1	2	11	0.071
87	0	0	28	0.000	12	8	8	0.429	1	0	27	0.036	2	8	18	0.071
88	0	0	16	0.000	5	7	4	0.313	0	0	16	0.000	0	8	8	0.000
89	0	0	4	0.000	1	1	2	0.250	0	0	4	0.000	0	2	2	0.000
90	0	0	13	0.000	6	3	4	0.462	1	0	12	0.077	0	3	10	0.000
91	0	0	24	0.000	7	10	7	0.292	1	0	23	0.042	2	7	15	0.083

Table A.3: True positive rates for all events (Section 4.1) - 3

	Minia				SGA				lobSTR				Pamir			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
92	0	0	19	0.000	7	8	4	0.368	1	0	18	0.053	0	6	13	0.000
93	0	0	3	0.000	1	0	2	0.333	0	0	3	0.000	0	2	1	0.000
94	0	0	21	0.000	8	4	9	0.381	2	0	19	0.095	1	6	14	0.048
95	0	0	16	0.000	6	6	4	0.375	0	0	16	0.000	2	6	8	0.125
96	0	0	18	0.000	5	9	4	0.278	1	0	17	0.056	0	8	10	0.000
97	0	0	3	0.000	3	0	0	1.000	0	0	3	0.000	0	1	2	0.000
98	0	0	16	0.000	3	10	3	0.188	0	0	16	0.000	3	4	9	0.188
99	0	0	25	0.000	5	8	12	0.200	0	0	25	0.000	1	3	21	0.040
100	0	0	18	0.000	2	13	3	0.111	0	0	18	0.000	0	3	15	0.000
101	0	0	2	0.000	1	1	0	0.500	0	0	2	0.000	0	1	1	0.000
102	0	0	10	0.000	1	3	6	0.100	0	0	10	0.000	1	1	8	0.100
103	0	0	23	0.000	3	9	11	0.130	0	0	23	0.000	0	6	17	0.000
104	0	0	17	0.000	3	6	8	0.176	0	0	17	0.000	0	2	15	0.000
106	0	0	26	0.000	3	7	16	0.115	0	0	26	0.000	1	4	21	0.038
107	0	0	24	0.000	2	7	15	0.083	1	0	23	0.042	0	5	19	0.000
108	0	0	20	0.000	1	8	11	0.050	0	0	20	0.000	0	3	17	0.000
109	0	0	7	0.000	0	2	5	0.000	0	0	7	0.000	0	1	6	0.000
110	0	0	15	0.000	0	4	11	0.000	0	0	15	0.000	0	4	11	0.000
111	0	0	39	0.000	1	9	29	0.026	0	0	39	0.000	1	9	29	0.026
112	0	0	16	0.000	1	5	10	0.063	0	0	16	0.000	0	3	13	0.000
113	0	0	3	0.000	0	1	2	0.000	0	0	3	0.000	0	1	2	0.000
114	0	0	30	0.000	0	12	18	0.000	0	0	30	0.000	0	4	26	0.000
115	0	0	39	0.000	1	12	26	0.026	1	0	38	0.026	0	5	34	0.000
116	0	0	20	0.000	0	11	9	0.000	0	0	20	0.000	1	3	16	0.050
117	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
118	0	0	29	0.000	1	8	20	0.034	0	0	29	0.000	0	6	23	0.000
119	0	0	39	0.000	2	15	22	0.051	0	0	39	0.000	0	7	32	0.000
120	0	0	21	0.000	0	4	17	0.000	0	0	21	0.000	0	3	18	0.000
121	0	0	4	0.000	0	1	3	0.000	0	0	4	0.000	0	1	3	0.000
122	0	0	24	0.000	1	4	19	0.042	0	0	24	0.000	0	3	21	0.000
123	0	0	25	0.000	0	7	18	0.000	0	0	25	0.000	0	6	19	0.000
124	0	0	29	0.000	0	12	17	0.000	0	0	29	0.000	0	4	25	0.000
126	0	0	15	0.000	0	8	7	0.000	0	0	15	0.000	0	4	11	0.000
127	0	0	35	0.000	0	10	25	0.000	0	0	35	0.000	0	6	29	0.000
128	0	0	19	0.000	0	7	12	0.000	0	0	19	0.000	0	6	13	0.000
129	0	0	3	0.000	0	2	1	0.000	0	0	3	0.000	0	0	3	0.000
130	0	0	26	0.000	0	9	17	0.000	0	0	26	0.000	0	5	21	0.000
131	0	0	33	0.000	0	12	21	0.000	0	0	33	0.000	0	7	26	0.000
132	0	0	20	0.000	0	8	12	0.000	0	0	20	0.000	0	4	16	0.000
133	0	0	3	0.000	0	1	2	0.000	0	0	3	0.000	0	0	3	0.000
134	0	0	18	0.000	0	7	11	0.000	0	0	18	0.000	0	3	15	0.000
135	0	0	28	0.000	0	6	22	0.000	0	0	28	0.000	0	1	27	0.000
136	0	0	26	0.000	0	8	18	0.000	0	0	26	0.000	0	4	22	0.000
137	0	0	2	0.000	0	0	2	0.000	0	0	2	0.000	0	0	2	0.000

Table A.4: True positive rates for all events (Section 4.1) - 4

	Minia				SGA				lobSTR				Pamir			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
138	0	0	20	0.000	0	7	13	0.000	0	0	20	0.000	0	4	16	0.000
139	0	0	42	0.000	0	10	32	0.000	0	0	42	0.000	0	6	36	0.000
140	0	0	9	0.000	0	3	6	0.000	0	0	9	0.000	0	2	7	0.000
141	0	0	5	0.000	0	1	4	0.000	0	0	5	0.000	0	2	3	0.000
142	0	0	18	0.000	0	4	14	0.000	0	0	18	0.000	0	0	18	0.000
143	0	0	23	0.000	0	5	18	0.000	0	0	23	0.000	0	2	21	0.000
144	0	0	10	0.000	0	4	6	0.000	0	0	10	0.000	0	2	8	0.000
146	0	0	7	0.000	0	2	5	0.000	0	0	7	0.000	0	3	4	0.000
147	0	0	10	0.000	0	0	10	0.000	0	0	10	0.000	0	0	10	0.000
148	0	0	11	0.000	0	1	10	0.000	0	0	11	0.000	0	1	10	0.000
149	0	0	5	0.000	0	2	3	0.000	0	0	5	0.000	0	1	4	0.000
150	0	0	4	0.000	0	1	3	0.000	0	0	4	0.000	0	0	4	0.000
151	0	0	8	0.000	0	1	7	0.000	0	0	8	0.000	0	1	7	0.000
152	0	0	1	0.000	0	1	0	0.000	0	0	1	0.000	0	0	1	0.000
153	0	0	2	0.000	0	1	1	0.000	0	0	2	0.000	0	1	1	0.000
154	0	0	13	0.000	0	2	11	0.000	0	0	13	0.000	0	1	12	0.000
155	0	0	2	0.000	0	1	1	0.000	0	0	2	0.000	0	0	2	0.000
156	0	0	3	0.000	0	1	2	0.000	0	0	3	0.000	0	0	3	0.000
157	0	0	4	0.000	0	1	3	0.000	0	0	4	0.000	0	0	4	0.000
158	0	0	6	0.000	0	2	4	0.000	0	0	6	0.000	0	1	5	0.000
159	0	0	9	0.000	0	4	5	0.000	0	0	9	0.000	0	3	6	0.000
160	0	0	6	0.000	0	1	5	0.000	0	0	6	0.000	0	0	6	0.000
161	0	0	5	0.000	0	0	5	0.000	0	0	5	0.000	0	0	5	0.000
162	0	0	7	0.000	0	1	6	0.000	0	0	7	0.000	0	1	6	0.000
163	0	0	5	0.000	0	0	5	0.000	0	0	5	0.000	0	0	5	0.000
164	0	0	11	0.000	0	4	7	0.000	0	0	11	0.000	0	3	8	0.000
166	0	0	5	0.000	0	1	4	0.000	0	0	5	0.000	0	0	5	0.000
167	0	0	3	0.000	0	0	3	0.000	0	0	3	0.000	0	0	3	0.000
168	0	0	3	0.000	0	0	3	0.000	0	0	3	0.000	0	0	3	0.000
169	0	0	5	0.000	0	2	3	0.000	0	0	5	0.000	0	2	3	0.000
171	0	0	6	0.000	0	2	4	0.000	0	0	6	0.000	0	2	4	0.000
172	0	0	6	0.000	0	1	5	0.000	0	0	6	0.000	0	2	4	0.000
173	0	0	7	0.000	0	2	5	0.000	0	0	7	0.000	0	1	6	0.000
174	0	0	8	0.000	0	2	6	0.000	0	0	8	0.000	0	1	7	0.000
175	0	0	2	0.000	0	0	2	0.000	0	0	2	0.000	0	0	2	0.000
176	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
178	0	0	3	0.000	0	1	2	0.000	0	0	3	0.000	0	0	3	0.000
179	0	0	2	0.000	0	1	1	0.000	0	0	2	0.000	0	1	1	0.000
181	0	0	2	0.000	0	0	2	0.000	0	0	2	0.000	0	0	2	0.000
182	0	0	1	0.000	0	1	0	0.000	0	0	1	0.000	0	1	0	0.000
183	0	0	1	0.000	0	1	0	0.000	0	0	1	0.000	0	0	1	0.000
184	0	0	3	0.000	0	0	3	0.000	0	0	3	0.000	0	0	3	0.000
187	0	0	2	0.000	0	2	0	0.000	0	0	2	0.000	0	0	2	0.000
188	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000

Table A.5: True positive rates for all events (Section 4.1) - 5

	Minia				SGA				lobSTR				Pamir			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
189	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
190	0	0	4	0.000	0	0	4	0.000	0	0	4	0.000	0	0	4	0.000
191	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
196	0	0	4	0.000	0	0	4	0.000	0	0	4	0.000	0	0	4	0.000
201	0	0	3	0.000	0	1	2	0.000	0	0	3	0.000	0	1	2	0.000
202	0	0	3	0.000	0	1	2	0.000	0	0	3	0.000	0	1	2	0.000
205	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
206	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
208	0	0	4	0.000	0	0	4	0.000	0	0	4	0.000	0	0	4	0.000
220	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000	0	1	0	0.000

Table A.6: True positive rates for all events (Section 4.2) - 1

	80x				60x				40x			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
20	11	0	0	1.000	10	1	0	0.909	5	4	2	0.455
22	4	0	0	1.000	4	0	0	1.000	2	0	2	0.500
23	9	1	0	0.900	10	0	0	1.000	5	4	1	0.500
24	13	0	0	1.000	12	1	0	0.923	10	2	1	0.769
26	7	0	0	1.000	7	0	0	1.000	5	2	0	0.714
27	14	1	0	0.933	13	2	0	0.867	9	4	2	0.600
28	10	0	0	1.000	10	0	0	1.000	6	2	2	0.600
29	5	0	0	1.000	5	0	0	1.000	4	0	1	0.800
30	5	0	0	1.000	4	1	0	0.800	0	4	1	0.000
31	25	0	0	1.000	21	4	0	0.840	23	2	0	0.920
32	11	1	0	0.917	12	0	0	1.000	9	2	1	0.750
33	1	0	1	0.500	1	0	1	0.500	2	0	0	1.000
34	13	1	0	0.929	13	1	0	0.929	8	3	3	0.571
35	22	0	0	1.000	19	3	0	0.864	19	3	0	0.864
36	9	0	0	1.000	8	1	0	0.889	7	2	0	0.778
37	0	1	0	0.000	1	0	0	1.000	1	0	0	1.000
38	13	0	0	1.000	11	2	0	0.846	7	5	1	0.538
39	26	1	1	0.929	20	5	3	0.714	15	10	3	0.536
40	13	1	0	0.929	13	1	0	0.929	8	6	0	0.571
41	2	0	0	1.000	2	0	0	1.000	1	1	0	0.500
42	14	1	0	0.933	13	2	0	0.867	9	5	1	0.600
43	14	1	0	0.933	11	4	0	0.733	9	6	0	0.600
44	19	1	0	0.950	19	1	0	0.950	15	5	0	0.750
46	22	2	0	0.917	21	3	0	0.875	16	7	1	0.667
47	25	4	0	0.862	28	1	0	0.966	18	10	1	0.621
48	16	0	0	1.000	14	2	0	0.875	13	3	0	0.813
49	5	0	0	1.000	5	0	0	1.000	5	0	0	1.000
50	7	0	0	1.000	6	1	0	0.857	5	1	1	0.714
51	15	1	1	0.882	16	1	0	0.941	8	6	3	0.471
52	9	0	3	0.750	7	2	3	0.583	2	7	3	0.167
53	1	0	0	1.000	1	0	0	1.000	1	0	0	1.000
54	15	1	1	0.882	12	4	1	0.706	8	7	2	0.471
55	27	1	0	0.964	26	2	0	0.929	20	8	0	0.714
56	12	1	0	0.923	11	2	0	0.846	8	4	1	0.615

Table A.7: True positive rates for all events (Section 4.2) - 2

	80x				60x				40x			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
57	1	0	0	1.000	1	0	0	1.000	0	0	1	0.000
58	16	0	2	0.889	14	3	1	0.778	11	3	4	0.611
59	23	0	1	0.958	21	2	1	0.875	16	5	3	0.667
60	20	0	1	0.952	15	5	1	0.714	11	6	4	0.524
61	1	0	0	1.000	1	0	0	1.000	1	0	0	1.000
62	12	0	0	1.000	8	2	2	0.667	5	6	1	0.417
63	23	1	2	0.885	21	3	2	0.808	15	7	4	0.577
64	23	1	1	0.920	21	4	0	0.840	16	6	3	0.640
66	10	4	0	0.714	9	5	0	0.643	6	8	0	0.429
67	13	5	2	0.650	15	3	2	0.750	12	5	3	0.600
68	15	1	0	0.938	9	5	2	0.563	8	7	1	0.500
69	4	3	0	0.571	5	1	1	0.714	2	2	3	0.286
70	18	1	3	0.818	18	1	3	0.818	8	7	7	0.364
71	33	4	2	0.846	26	8	5	0.667	15	12	12	0.385
72	8	2	1	0.727	7	2	2	0.636	7	3	1	0.636
73	5	1	2	0.625	8	0	0	1.000	5	3	0	0.625
74	12	4	5	0.571	11	8	2	0.524	7	9	5	0.333
75	20	1	4	0.800	13	5	7	0.520	13	9	3	0.520
76	12	2	4	0.667	7	6	5	0.389	7	5	6	0.389
77	1	0	0	1.000	1	0	0	1.000	0	0	1	0.000
78	13	5	0	0.722	5	10	3	0.278	6	10	2	0.333
79	18	9	7	0.529	17	12	5	0.500	11	13	10	0.324
80	4	3	4	0.364	3	5	3	0.273	3	4	4	0.273
81	3	0	0	1.000	1	0	2	0.333	1	0	2	0.333
82	10	1	3	0.714	6	4	4	0.429	3	6	5	0.214
83	13	4	6	0.565	13	5	5	0.565	4	8	11	0.174
84	10	6	3	0.526	8	6	5	0.421	5	7	7	0.263
85	1	0	0	1.000	0	0	1	0.000	1	0	0	1.000
86	12	3	5	0.600	12	4	4	0.600	6	6	8	0.300
87	24	7	6	0.649	13	16	8	0.351	16	15	6	0.432
88	7	4	6	0.412	5	5	7	0.294	5	8	4	0.294
89	1	1	1	0.333	0	2	1	0.000	0	1	2	0.000
90	7	4	3	0.500	10	2	2	0.714	5	3	6	0.357
91	12	7	13	0.375	13	10	9	0.406	8	9	15	0.250

Table A.8: True positive rates for all events (Section 4.2) - 3

	80x				60x				40x			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
92	6	6	2	0.429	5	6	3	0.357	1	8	5	0.071
93	2	1	0	0.667	1	1	1	0.333	1	2	0	0.333
94	11	8	10	0.379	12	7	10	0.414	5	8	16	0.172
95	10	5	7	0.455	8	6	8	0.364	8	5	9	0.364
96	6	6	4	0.375	6	4	6	0.375	2	5	9	0.125
98	2	4	4	0.200	3	4	3	0.300	3	4	3	0.300
99	10	6	7	0.435	5	10	8	0.217	4	11	8	0.174
100	5	8	8	0.238	2	10	9	0.095	1	10	10	0.048
101	0	1	2	0.000	0	1	2	0.000	0	1	2	0.000
102	7	5	9	0.333	4	6	11	0.190	0	9	12	0.000
103	6	14	13	0.182	3	17	13	0.091	6	14	13	0.182
104	1	8	10	0.053	3	5	11	0.158	1	7	11	0.053
106	0	9	10	0.000	6	7	6	0.316	0	9	10	0.000
107	0	9	11	0.000	4	6	10	0.200	3	7	10	0.150
108	0	10	3	0.000	1	9	3	0.077	0	10	3	0.000
109	0	1	3	0.000	0	1	3	0.000	0	1	3	0.000
110	1	4	5	0.100	0	4	6	0.000	0	4	6	0.000
111	2	20	7	0.069	3	19	7	0.103	4	16	9	0.138
112	2	4	12	0.111	2	6	10	0.111	0	6	12	0.000
113	0	0	3	0.000	0	0	3	0.000	0	0	3	0.000
114	0	8	9	0.000	0	7	10	0.000	1	5	11	0.059
115	2	13	16	0.065	0	15	16	0.000	2	12	17	0.065
116	0	7	10	0.000	0	6	11	0.000	2	7	8	0.118
117	0	1	1	0.000	0	1	1	0.000	0	1	1	0.000
118	0	8	7	0.000	0	7	8	0.000	0	7	8	0.000
119	0	16	13	0.000	1	14	14	0.034	0	14	15	0.000
120	0	8	8	0.000	0	8	8	0.000	1	7	8	0.063
121	0	2	2	0.000	0	1	3	0.000	0	2	2	0.000
122	0	6	14	0.000	0	7	13	0.000	0	7	13	0.000
123	0	14	9	0.000	0	12	11	0.000	0	15	8	0.000
124	0	7	8	0.000	0	7	8	0.000	0	5	10	0.000
126	0	11	10	0.000	0	11	10	0.000	0	11	10	0.000
127	0	13	14	0.000	1	12	14	0.037	0	14	13	0.000
128	0	8	12	0.000	0	7	13	0.000	0	8	12	0.000

Table A.9: True positive rates for all events (Section 4.2) - 4

	80x				60x				40x			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
129	0	3	1	0.000	0	3	1	0.000	0	3	1	0.000
130	0	8	10	0.000	0	8	10	0.000	0	8	10	0.000
131	0	13	11	0.000	0	12	12	0.000	0	13	11	0.000
132	0	9	6	0.000	0	9	6	0.000	0	7	8	0.000
133	0	1	2	0.000	0	1	2	0.000	0	1	2	0.000
134	0	6	10	0.000	0	6	10	0.000	0	6	10	0.000
135	0	4	18	0.000	0	4	18	0.000	0	5	17	0.000
136	0	4	6	0.000	0	5	5	0.000	1	4	5	0.100
137	0	2	0	0.000	0	2	0	0.000	0	2	0	0.000
138	0	8	8	0.000	0	6	10	0.000	0	7	9	0.000
139	0	14	10	0.000	0	12	12	0.000	0	11	13	0.000
140	0	3	4	0.000	0	3	4	0.000	0	3	4	0.000
141	0	3	1	0.000	0	3	1	0.000	0	3	1	0.000
142	0	3	7	0.000	0	2	8	0.000	0	3	7	0.000
143	0	4	9	0.000	0	4	9	0.000	0	4	9	0.000
144	0	4	9	0.000	0	3	10	0.000	0	3	10	0.000
145	0	1	1	0.000	0	1	1	0.000	0	1	1	0.000
146	0	3	1	0.000	0	2	2	0.000	0	2	2	0.000
147	0	4	5	0.000	0	4	5	0.000	0	2	7	0.000
148	0	3	2	0.000	0	3	2	0.000	0	3	2	0.000
149	0	2	1	0.000	0	2	1	0.000	0	2	1	0.000
150	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
151	0	3	3	0.000	0	3	3	0.000	0	3	3	0.000
152	0	3	1	0.000	0	2	2	0.000	0	3	1	0.000
153	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
154	0	4	6	0.000	0	2	8	0.000	0	3	7	0.000
155	0	0	2	0.000	0	0	2	0.000	0	0	2	0.000
156	0	1	2	0.000	0	1	2	0.000	0	0	3	0.000
157	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
158	0	1	5	0.000	0	1	5	0.000	0	1	5	0.000
159	0	5	4	0.000	0	4	5	0.000	0	5	4	0.000
160	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
161	0	1	2	0.000	0	1	2	0.000	0	1	2	0.000
162	0	2	2	0.000	0	2	2	0.000	0	2	2	0.000

Table A.10: True positive rates for all events (Section 4.2) - 5

	80x				60x				40x			
Region Size	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**	**Hit**	**PHit**	**Miss**	**Recall**
163	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
164	0	3	2	0.000	0	3	2	0.000	0	3	2	0.000
166	0	5	5	0.000	0	6	4	0.000	0	6	4	0.000
167	0	2	1	0.000	0	2	1	0.000	0	2	1	0.000
168	0	3	1	0.000	0	3	1	0.000	0	3	1	0.000
169	0	3	6	0.000	0	3	6	0.000	0	3	6	0.000
170	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
171	0	1	1	0.000	0	1	1	0.000	0	1	1	0.000
172	0	1	3	0.000	0	2	2	0.000	0	2	2	0.000
173	0	3	2	0.000	0	3	2	0.000	0	3	2	0.000
174	0	3	1	0.000	0	3	1	0.000	0	3	1	0.000
176	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
177	0	3	0	0.000	0	3	0	0.000	0	3	0	0.000
178	0	1	1	0.000	0	1	1	0.000	0	1	1	0.000
179	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
180	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
181	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
182	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
184	0	1	2	0.000	0	2	1	0.000	0	0	3	0.000
187	0	1	1	0.000	0	1	1	0.000	0	1	1	0.000
188	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
189	0	0	2	0.000	0	0	2	0.000	0	0	2	0.000
190	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
192	0	1	0	0.000	0	0	1	0.000	0	0	1	0.000
194	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
196	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000
199	0	2	0	0.000	0	2	0	0.000	0	2	0	0.000
202	0	1	2	0.000	0	1	2	0.000	0	1	2	0.000
203	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
207	0	1	0	0.000	0	1	0	0.000	0	1	0	0.000
208	0	3	2	0.000	0	3	2	0.000	0	3	2	0.000
211	0	0	1	0.000	0	0	1	0.000	0	0	1	0.000