5-5-2017

De Novo Repeats Construction: Methods and Applications

Chong Chu
chong.chu@engr.uconn.edu

De Novo Repeats Construction: Methods and Applications

Chong Chu, Ph.D.

University of Connecticut, 2017

ABSTRACT

Repeat elements are important components of eukaryotic genomes. The dropping cost of the second and third generation sequencing technologies provides opportunities to study repeat elements of hundreds of species and thousands of individuals of one species. Based on the quality of the assembled genomes, generally there are two obstacles for studying repeat elements: (1) For species with high repetitive or complex genomes that do not have high quality genomes assembled, how do we construct de novo repeat elements? (2) For species with high quality genomes assembled, how to detect mobile elements insertions (one type of repeat elements) of different individuals of the species? It is known that most of the gaps on draft genomes are caused by repeat elements, thus a following-up question is: (3) With the understanding of repeats on genome, can we better close the gaps on draft genomes?

To address the first problem that reference genomes are incomplete and often contain missing data in highly repetitive regions, we propose a method (called REPdenovo) to construct repeats directly from short sequencing reads. REPdenovo can construct various types of repeats that are highly repetitive and have low sequence divergence within copies. We show that REPdenovo is substantially better than existing methods both in terms of the number and the completeness of the repeat sequences that it recovers. The key advantage of REPdenovo is that it can reconstruct long repeats from short sequence reads. We apply the method to human data

and discover a number of potentially new repeats sequences that have been missed by previous repeat annotations.

Next we present an improved version of REPdenovo, which is able to reconstruct more divergent and lower frequency repeats from short sequencing reads. Comparing with the original REPdenovo, this improved approach uses more repeat-related k-mers. In addition, the new approach improves repeat assembly quality using a consensus-based k-mer processing method. We compare the performance of the new method with REPdenovo and RepARK on Human and Arabidopsis thaliana short sequencing data. The results show that the improved REPdenovo can assemble more complete repeats than REPdenovo (and also RepARK). We apply the improved REPdenovo on Hummingbird which has no known repeat library, and construct many repeat elements that are validated using PacBio long reads. Many of these repeats are likely to be true that are not in public repeat libraries.

To answer the second question, we develop a novel method (called REPdenovo-MEI) for detecting mobile element insertions (MEIs) with given reference genome and alignments of different individuals. Different from all existing tools, REPdenovo-MEI does not rely on any repeats library and can call MEIs efficiently and accurately. Besides calling out insertion sites, REPdenovo-MEI has a local assembly step to construct the inserted copy and a classification based approach for calling genotypes. In addition, the third-generation sequencing technology generates long reads of thousands of bases long, which usually is long enough to contain the whole repeat elements in the reads, thus can help to construct the MEIs completely. Thus, besides short reads, REPdenovo-MEI can also work with long reads to infer the inserted copies. Results on both simulated and real data show that REPdenovo-MEI outperforms existing tools on both accuracy and the number of constructed high divergent MEIs.

To solve the third problem of closing gaps on draft genomes, we propose a new method (called GAPPadder) that can sensitively close gaps for large and complex genomes. Different from existing approaches, GAPPadder collects more gap originated reads, especially repeat associated reads, and better utilize the information of different insert sizes of PE and MP reads. Finally, GAPPadder provides higher quality of local assembly with an extra contigs merging step. We show GAPPadder can close more gaps on one bacterial genome, Human chromosome 14 and Human whole genome. Besides closing gaps on draft genome assembled only from short sequence reads, GAPPadder can also be used to close gaps for draft genomes assembled with long reads. We show GAPPadder can close gaps on the bed bug genome and the Asian sea bass genome that are assembled partially and fully with long reads respectively. We also show GAPPadder is efficient in both time and memory usage.

De Novo Repeats Construction: Methods and Applications

Chong Chu

Master of Engineering, Beijing University of Chemical Technology, China, 2012

Bachelor of Science, Beijing University of Chemical Technology, China, 2009

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2017

APPROVAL PAGE

Doctor of Philosophy Dissertation

De Novo Repeats Construction: Methods and Applications

Presented by

Chong Chu, B.S. Computer Science, M.E. Computer Application Technology

Major Advisor _____
Yufeng Wu

Associate Advisor _____
Ion Mandoiu

Associate Advisor _____
Sanguthevar Rajasekaran

Associate Advisor _____
Dong-Guk Shin

University of Connecticut

2017

ACKNOWLEDGMENTS

First, I would like to express the deepest gratitude to my major advisor, Dr. Yufeng Wu, who has led my way during the past five years. I am so lucky to have the chance working with him. He guides and trains me in research step by step: From selecting topics, establishing milestones, developing methods, to writing papers & presenting works. I believe all these will benefit for my whole life. Without his supervision and constant help, my research and this dissertation would not have been possible. I can only hope to become a researcher of his caliber in the future.

I would like to thank my doctoral committee members, Dr. Ion Mandoiu, Dr. Sanguthevar Rajasekaran, and Dr. Dong-Guk Shin. Dr. Ion Mandoiu, I appreciate all your support for providing all kinds of computing resources for my work, and also the suggestions and helps in doing projects. Dr. Sanguthevar Rajasekaran, and Dr. Dong-Guk Shi, your guidance has been one of the most important sources that motivates me to do better research.

Next, to everyone in the group, and also the friends at UCONN. It was great sharing of ideas, pleasure, and progresses with all of you during last five years. Hope the research and the life go well for you all.

In special, to my girlfriend Xian Shao. It is lucky to meet you at UCONN. And I am gratitude for all your supports and understandings for my work and decisions.

And finally, last but by no means least, to my parents, my sister, and other members in our big family. You always respect my choice, and give me the biggest support in my life. Without you I will never make the decision to come to U.S. and get the degree.

To my family, I dedicate this dissertation.

The work was partially supported by research grants from NSF: IIS-0953563, IIS-1447711 and IIS-1526415.

Contents

Ch. 1. Introduction 1
1.1 Motivation and Challenges . . . 1
1.2 Overview . . . 4

Ch. 2. REPdenovo: Inferring *de novo* repeat motifs from short sequence reads 7
2.1 Introduction . . . 7
2.2 Background . . . 9
2.3 Methods . . . 10
2.3.1 Construction of raw contigs . . . 11
2.3.2 Assembly of raw contigs into long repeats . . . 13
2.3.3 Improving assembly quality by filtering . . . 15
2.3.4 Evaluation and comparison of methods . . . 16
2.4 Results . . . 17
2.4.1 Constructed human repeats are likely to be real . . . 18
2.4.2 Comparison to Repbase and RepeatMasker and finding novel repeats in human genome . . . 21
2.4.3 Comparison of assembly quality of REPdenovo and RepARK . 27
2.4.4 Running time . . . 29
2.5 Discussions and Conclusion . . . 30

Ch. 3. An Improved Approach for Reconstructing Consensus Repeats from Short Sequence Reads 33
3.1 Introduction . . . 33
3.2 Method . . . 35
3.2.1 Repeat assembly from frequent k-mers . . . 35
3.2.2 The difficulty of assembling highly divergent repeat regions . . 37

3.2.3 Mapping-based alignment for finding more repeat-related k-mers 38
3.2.4 K-mer polishing . . . 40
3.3 Results . . . 41
3.3.1 Comparison with Human and Arabidopsis data . . . 42
3.3.2 Comparison between the two versions of REPdenovo . . . 43
3.3.3 Repeat elements construction for Hummingbird . . . 45
3.4 Discussion . . . 47

Ch. 4. REPdenovo-MEI: De novo detection of mobile element insertions from sequence reads 50
4.1 Introduction . . . 50
4.2 Method . . . 53
4.2.1 High frequent k-mer library construction . . . 53
4.2.2 Calling candidate MEIs from short reads . . . 55
4.2.3 Constructing MEIs by local assembly . . . 57
4.2.4 Constructing and validating MEIs from long reads . . . 58
4.2.5 Call genotype of the MEIs . . . 59
4.3 Results . . . 61
4.3.1 Comparison on simulated data . . . 61
4.3.2 Comparison on real data . . . 62
4.3.3 Comparison on called out MEI divergence . . . 65
4.4 Discussion and Conclusion . . . 65

Ch. 5. GAPPadder: A Sensitive Approach for Closing Gaps on Draft Genomes with Short Sequence Reads 67
5.1 Introduction . . . 67
5.1.1 Gaps in draft genomes . . . 70
5.2 Methods . . . 72
5.2.1 High level approach . . . 72
5.2.2 Relevant reads originated from gap regions . . . 73
5.2.3 Gap closing procedure . . . 74
5.3 Results . . . 79
5.3.1 Comparison with existing tools . . . 80
5.3.2 Comparison on data with different insert sizes . . . 81
5.3.3 Time and memory usage . . . 83
5.3.4 Closing gaps on draft genomes assembled partially or fully with long reads . . . 84
5.4 Discussion and Conclusions . . . 86

Ch. 6. Conclusion 89

Bibliography 92

Chapter 1

Introduction

1.1 Motivation and Challenges

Most genomes, and in particular mammalian genomes, consist of large amounts of repeat elements. A repeat is a segment of DNA that appears multiple times in the genome in identical or near-identical form. There are many types of repeats (75; 76; 77). Transposable elements (TEs) or mobile elements (MEs) are perhaps the most well-known. They are believed to constitute 25% to 40% of most mammalian genomes (76; 77; 78; 79) and can amplify themselves in the genome using various mechanisms, typically involving RNA intermediates. In humans the most common TEs are Long Interspersed Elements (LINE-1s or L1s), Short Interspersed Element (SINEs), and Long Terminal Repeats (LTRs), comprising approx. 17%, 11% and 8% of the human genome, respectively. Other common repeat elements include various types of satellites. When repeat elements amplify from one place to another place and genome passed from one generation to another generation, variations including

SNP, short indels, or even large structural variations will happen on the copies. Thus as a result, even when one repeat is copied to another location, the two copies may not be exactly the same. And as the evolution proceeds, variations within the repeat copies will accumulate. Divergent rate is used to indicate the difference between the consensus of copies and the individual copy. In general, the higher the divergent rate, the older the copy is. Many studies have demonstrated that repeat elements contribute to genome evolution and genetic diversity (132; 133; 134; 135; 149; 152). In particular, (136) shows that MEs can provide active promoter, splice site or terminator features which can affect the expression, structure and function of nearby genes. MEs are also involved in the creation of transcriptional regulatory networks, and in the generation of chromosomal rearrangements (137; 138). Thus, it is important to identify and analyze repeat elements.

Taking advantages of the dropping cost and high throughput of the second and third generation sequencing technologies, hundreds of species and thousands of individuals of one organism are sequenced, which provides both opportunity and challenges to study repeat elements. The opportunity is that with the amount of sequencing data, repeat elements can be constructed and annotated, and thus can be analyzed in different applications. In general, the first step to characterize the genome is assembling the genomes, using assembly tools like (111; 119; 127; 128; 131). And then consensus repeats can be constructed from the draft genomes using tools like RepeatScout (89), PILER(31) and phRAIDER (90). Finally, the draft genomes can be annotated with the consensus repeats. The challenge is that, for many species, the genomes are not only large, but also repetitive and complex, which cause the assemblers failed to construct high quality draft genomes from sequencing data. Also, it is known that most of the missing parts (gaps) on the draft genomes are caused

by repeats. As a result, repeat elements either are missed from the low-quality draft genome, or are break into small pieces of contigs, and thus cannot be directly annotated and analyzed.

Another challenge arises when studying the still active TEs, which are usually called mobile element insertions (MEIs) or transposable element insertions (TEIs). These MEIs usually act as regulators in genomes, like regulating the chromatin structure and transcription, regulating the RNA processing, and affecting messenger RNA localization and translation (70). MEIs are still quite active in many species like maize, bacterias. In human, most of the TEs are non-active, and the most known still active ones are LINE1, Alu, and SVA. However, even though not many types of TEs are active in human, these active ones show important functions in many studies. For example, MEIs are found to be active in many types of cancers (144; 147). Recent studies also use MEIs as biomarkers to study the cell evolution in neural system (74). Thus, it is important to call the MEIs for individuals of one species. Several tools (144; 83; 84; 85; 93; 86; 92; 151; 153) have been developed for calling MEIs with short sequencing reads. All these tools follow the same strategy that: check the discordant reads whether can be aligned to existing repeat copies, and check the unmapped reads whether can be aligned to the consensus repeats. The strategy requires that there are annotated repeat copies and consensus library. It also requires prior knowledge that which repeats are still active. The challenge is that generally it is hard to get prior knowledge of which repeats are still active, especially for tissues or cell lines with high mutation rates. In addition, if the inserted copies have high divergent rate, it will be difficult to align the reads back to other copies or the consensus repeats. As a result, these MEIs will not be missed.

1.2 Overview

One solution for the first challenge is to construct the repeats directly from reads. One existing tool is RepARK (91). We propose a novel repeat assembly method, REPdenovo, performs *de novo* estimation of low-divergent and highly frequent repeats from sequence reads. Similar to RepARK, REPdenovo first identifies the frequent k-mers and then assembles these k-mers. This step leads to a set of repeat contigs (called raw contigs). Raw contigs are the final results of RepARK. However, raw contigs are often only fragments of complete consensus repeats. This is because repeats usually contain regions of higher sequence divergence than other regions. K-mers within higher divergence regions tend to have much smaller frequencies and thus may not be identified to be frequent. The frequent k-mer assembly only leads to segments that have low divergence (i.e. more conserved) in repeat copies. Therefore, assembly of frequent k-mers alone does not produce contigs spanning complete repeats. To address this issue, REPdenovo performs a second assembly step by connecting raw contigs into long repeats.

Another solution for the first challenge is to close the gaps on the draft genomes. But this is only for genomes assembled but with lots of gaps. It is known that gaps usually happen at regions that are high repetitive, duplicated, or of low coverage. For example, 45 new avian species have been sequenced and assembled recently in a comparative study of avian genomes (66). Draft genomes of 25 out of these 45 species have average N50 around 48 kb, which indicates the draft genomes are fragmented with many gaps. About 3,000 genes are likely missing or only partially annotated due to gaps. As a result, only 70% to 80% of the entire catalog of avian genes can be predicted, which may cause bias in downstream analysis. So, a more complete

genome not only helps to study repeats elements, but also leads to better annotation, less genotyping error and, concomitantly, a greater likelihood of identifying causal variation associated with traits (112). We develop a new approach called GAPPadder for closing gaps on draft genomes. Similar to existing tools, it also performs local assembly from reads that originate from gap regions. Different from existing tools, GAPPadder collects more reads relevant for gap closing, especially repeat-associated reads which are ignored by all the existing tools. Moreover, GAPPadder collects higher quality reads by utilizing more information with different insert sizes of pair-end (PM) and mate-pair (MP) reads. GAPPadder uses a different local assembly method for gap closing compared with existing methods. Existing methods often rely on local extension of contigs. GAPPadder, instead, performs a two-stage local assembly: it first assembles contigs in the gap and then generates higher quality local assembly of gap sequences by merging contigs.

To solve the second challenge, We propose a new approach for calling mobile elements insertions with sequencing data. Comparing with existing approaches, our approach does not require any repeat libraries or genome annotation files to call MEIs. The key idea is that instead of using a repeat library or genome annotation file to check whether reads come from repeats, we collect high frequent kmers, and check the reads against the high frequent kmers to decide whether the reads come from repeats. In addition, considering the divergence rate of repeats, a clustering algorithm is designed to involve those low frequent kmers, whose edit distances from high frequent kmers are within the given threshold. In this way, our approach can tolerate more variation and as a result can call out more diverged mobile element insertions. If long reads are available for the individuals, an optional step is designed to call out the inserted copies. For each called out candidate insertion site on the

reference, we first align the left and right flank regions beside the candidate site to the long reads. If the ME insertion do exist, then the two flank regions will be apart from each other when aligned on the long reads, and the part between them is the sequence of the ME Insertion. As there are usually more than one long read covering the site, we require at least N reads support the same sequence, where N is a user-specific parameter. What's more, besides identifying the sequences of inserted copies, this step can also improve the accuracy by filtering out the false positive ones.

Overall, in this thesis, we propose three methods to solve the three previous mentioned problems. In particular:

- De novo repeats construction directly from short sequencing reads. We propose an approach (called REPdenovo) to construct repeats directly from short sequencing reads. REPdenovo does not rely on any reference or any repeat libraries. Comparing with existing tools, REPdenovo can construct more complete repeats.

- De novo detect ME insertions with sequencing data. We propose a method to call ME insertions for different individuals of one species with the alignments. The method does not need any repeat libraries, and can work for high divergent ME insertions. To the best of our knowledge, there is no any computational tools that can do the same work.

- Closing gaps on draft genome. A gap closing approach is proposed to generate more complete draft genome. Different from existing tools, the proposed approach better utilizes the repeat associated reads, different insert size information, and high quality local assembly to close the gaps, which is more sensitive.

Chapter 2

REPdenovo: Inferring *de novo* repeat motifs from short sequence reads

2.1 Introduction

Identifying repeats in a genome is a long-standing research problem. There are many computational approaches and software tools for the analysis of repeat composition (80; 81; 82; 83; 85). One type of repeat analysis tools rely on curated repeat libraries to identify repeats. The most popular of these is RepeatMasker (80), which aligns genomic sequences to known consensus repeat sequences given by libraries such as Repbase (87) and Dfam (88). While RepeatMasker has been used extensively in the literature and led to many interesting discoveries, a limitation is that it needs a library of known repeat consensus sequences. Such repeat sequences are usually not available for many newly sequenced organisms. Alternatively, many existing methods

(81; 82; 83; 85; 148; 150) identify repeats by analysis of reference genomes. However, many genomes are poorly assembled, particularly in regions of high repeat content. Therefore, most existing methods can not find novel repeats that are not present in the curated library of repeats or in a reference genome. For organisms with little repeat annotation and without a good reference genome, there are few tools available for characterizing repeat content. Even for organisms such as humans with good reference genomes, there are often missing bases in regions of high repeat content. The human genome may therefore still harbor uncharacterized repeat elements.

In principle, finding repeats directly from sequence data may be appropriate for situations where there is no good reference genome or we want to find repeats that are not present in the reference genome. Recently, methods that analyze repeats based on sequence data start to appear. One such method is RepARK (91). RepARK can assemble repeats directly from sequence reads without reference genomes. However, experiments on RepARK show that there is still great room to improve the repeat assembly.

In this chapter, we present a new approach for *de novo* assembly of repeat elements, called REPdenovo. Similar to RepARK, REPdenovo constructs repeats from sequence reads directly and does not need a reference genome. REPdenovo aims at constructing repeats that have relatively high copy numbers and low sequence divergence within copies of the repeats. The repeats can be of various types, e.g. TE or satellites. The main advantage of REPdenovo is that it implements more accurate repeat assembly algorithms than RepARK. Using real data, we demonstrate that REPdenovo outperforms RepARK in terms of completeness and number of long repeats constructed. We also analyze sequence data from humans, and report potentially new human repeat elements missed by previous analyses. We also provide

supporting evidence which shows many of these repeats are likely to be real.

2.2 Background

There are several existing computational approaches for finding TEs from short sequence reads (92; 91). The method in (92) assumes a reference genome is available, and finds repeats from sequence reads using the reference. A major drawback is that there is no high-quality reference genomes for many organisms. In principle, one can use short reads to assemble a reference genome. However, repetitive regions are usually more difficult to assemble. This leads to reduced power for repeat analysis if one uses the assembled reference genome for the purpose of repeat finding.

There are also methods which directly assemble repeats from sequence reads. RepARK (91) is such a method developed recently for repeat elements assembly. RepARK is based on k-mer counting. K-mers are substrings of k nucleotides. As shown in Fig. 2.2.1, k-mer counting aims to count the occurrence of length-k substrings in all sequence reads. The result of k-mer counting is a vector OCC of size 4^k, where OCC_i is the number of times the i-th k-mer appears in the reads. For example, in Fig. 2.2.1, there is a single read. CGG appears two times while AAC and ACG appear once each. There exist efficient algorithms for k-mer counting, e.g. (94). RepARK uses an approach which reconstructs segments of repetitive regions directly from sequence reads by first counting the k-mers from the sequence reads and then assembling all frequent k-mers (whose frequencies exceed some fixed threshold) (91). The key idea is that k-mers in repeats may be more frequent than k-mers not in the repeats due to the high copy numbers of repeats. (91) showed that some contigs

assembled this way are fairly long and many contigs can be mapped to the reference genome. Here, a contig is a segment of assembled genomes. This is encouraging since it demonstrates that estimation of repeats such as TEs can be done *de novo* from raw short-read sequencing data. However, as we will show in this chapter, the method implemented in RepARK tends to only construct partial repeats. This may lead to considerable uncertainty when analyzing the evolution of repeat elements and to reduced detection rates of new repeat elements.

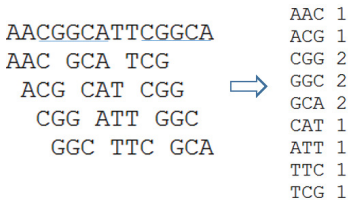

FIGURE 2.2.1: *Illustration of the k-mer counting. Long sequence: a sequence read. Length-3 sequence: k-mer. Here,* $k = 3$*. The table on the right shows the k-mer counting result.*

2.3 Methods

The new repeat assembly method, REPdenovo, performs *de novo* estimation of low-divergent and highly frequent repeats from sequence reads. Similar to RepARK (91), REPdenovo first identifies the frequent k-mers and then assembles these k-mers. This step leads to a set of repeat contigs (called raw contigs). Raw contigs are the final results of RepARK. However, raw contigs are often only fragments of complete consensus repeats. This is because repeats usually contain regions of higher sequence divergence than other regions. K-mers within higher divergence regions tend to have much smaller frequencies and thus may not be identified to be frequent.

The frequent k-mer assembly only leads to segments that have low divergence (i.e. more conserved) in repeat copies. Therefore, assembly of frequent k-mers alone does not produce contigs spanning complete repeats. To address this issue, REPdenovo performs a second assembly step by connecting raw contigs into long repeats. The key steps of REPdenovo algorithm are illustrated in Fig. 2.3.1 and are explained below:

1. Assembly of raw contigs from frequent k-mers.

2. Merging of raw contigs into larger contigs ideally representing the entire repeat motif. This step is conceptually analogous to the idea of merging contigs into scaffolds in regular genome-assembly.

3. Verification and filtering of the assembled repeats. By aligning reads back to the constructed repeats and checking the read depth, some wrongly assembled repeats can be filtered out.

2.3.1 Construction of raw contigs

REPdenovo first constructs raw contigs directly from sequence reads by constructing a catalog of highly represented kmers, i.e. k-mers with frequencies over average k-mer frequency times a threshold value f_K. The default value of f_K is 10, which means the frequency of a frequent k-mer is over 10 times the average k-mer frequency. This step could be improved in the future by using k-mer probabilities that take nucleotide or di-nucleotide frequencies into account. The current implementation uses Jellyfish (94) for k-mer counting, although other k-mer counting algorithms can also be used.

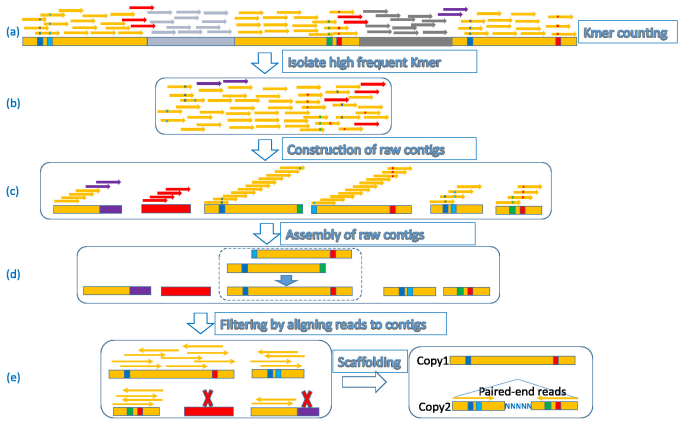

FIGURE 2.3.1: *Illustration of the main steps of REPdenovo. Thick bars: genomic sequences. Thin bars: k-mers. K-mer counting step: yellow parts are repeats (with some mismatches). Colored squares within thick bars: mutations (substitutions and indels) within repeats.*

Once frequent k-mers are identified, the next step of REPdenovo is assembling frequent k-mers into contigs (called raw contigs). This is done by treating the frequent k-mers as sequence reads and then assembling these k-mers by existing short read assembly tools. Currently, Velvet (131) is used for this step. REPdenovo implements several additional techniques for more accurate construction of raw contigs and classification of repeats. First, REPdenovo takes a "frequency-based assembly" approach. That is, REPdenovo does not assemble all frequent k-mers in one step as in RepARK (91). Instead, it groups and assembles k-mers with similar binned frequencies. A bin is a range of frequency based on a target frequency. By default, the range is [0.2f, 5f], where f is the target k-mer frequency. REPdenovo selects a number of evenly spaced target frequencies based on the collected k-mer frequencies from the reads as follows. REPdenovo starts from the k-mers in the highest frequency bin. Each time, REPdenovo assembles frequent k-mers within the current frequency bin. The range is then decreased (by default two times from the previous one) for the

next round in a way that there is overlap in ranges of two consecutive steps. Users can also change the ranges in the settings of REPdenovo. Frequency-based assembly may reduce assembly error under the assumption that k-mers from the same repeat tends to have similar frequencies.

We use multiple k values (e.g. 20, 30 and 40) for assembly. Raw contigs assembled from different k-values are then combined to form a single list of raw contigs.

2.3.2 Assembly of raw contigs into long repeats

Most current next-generation sequencing platforms, like Illumina, generate paired-end reads with length around 100bp. Paired-end reads allow users to sequence both ends of a fragment and generate high-quality, alignable sequence data.

Empirical results show that most assembled raw contigs tend to be close to 100 bp in length for real sequence reads. However, many repeats are much longer than 100 bp. For example, many L1 elements in humans are 5 kbp or longer. Thus, raw contigs alone usually do not give complete repeat consensus sequences. To address this problem, REPdenovo performs a second assembly step by connecting the raw contigs into long repeats as follows.

For the raw contigs, we build a directed contig graph G, which is similar to the overlap graph in sequence assembly (96). Each node in G corresponds to a raw contig. There is an edge from node v_1 to v_2 if there is significant overlap between contig v_1 and contig v_2. We say v_1 has significant overlap with v_2 if the length of the overlap between v_1 and v_2 is longer than a threshold value (15 bp by default) and the number of mismatches (substitutions and indels) is small (¡ 5% by default). The analysis is performed using standard pairwise sequence alignment based on dynamic

programming, e.g. (97). The overlap detection step allows errors in the overlapped regions of the raw contigs. This is because the overlapped regions of two connecting raw contigs usually don't match exactly. Performing the alignment for all $O(n^2)$ pairs can be slow when n is large. REPdenovo therefore only aligns pairs of raw contigs containing common length k_0 substrings. REPdenovo uses $k_0 = 5$ in the current implementation. Such preprocessing speeds up the computation significantly in practice.

Overlap alone may not be very reliable especially when the length of the overlap region is small. To allow an edge from v_1 to v_2 in G, REPdenovo also requires the existence of read pairs where one end maps (using "bwa mem" with default parameters) to v_1 and the other maps to v_2, when such read pairs are expected given the insert size of the library and the relative positions of v_1 and v_2 in the merged contig. We use the default settings of BWA for reads mapping.

Once the contig graph G is constructed, REPdenovo then searches for long paths between two nodes in G. Each path corresponds to an assembled long repeat. There are often cycles in G and it is usually impractical to enumerate all paths in G. To address the issue of cycles, REPdenovo first finds the strongly connected components in G. A strongly connected component (SCC) contains one or multiple nodes where any two nodes are mutually reachable. Suppose we treat a SCC as a node. Then G implies a new graph G', where the nodes of G' are the SCCs and there is an edge from SCC_1 to SCC_2 if a node in SCC_2 is reachable from a node in SCC_1 in G. The definition of SCC ensures G' is acyclic. We then run the standard topological sort algorithm (e.g. (98)) on each SCC (i.e. subgraph G_1 containing only nodes in the SCC_1). When G_1 is acyclic, topological sort arranges the nodes of G_1 in a linear topological order so that all edges of G_1 point to the same direction in the linear order.

That is, topological order means for every directed edge $u \rightarrow v$ from vertex u to vertex v in G, u is prior to v in the ordering. When G_1 contains cycles, topological sort can still run but it does not lead to a perfect topological linear order. Our experience shows that while cycles exist in G, G is often near-acyclic and the strongly connected components are usually small. Thus, we simply rely on the linear order produced by the topological sort algorithm even when the linear order is not strictly topological order. REPdenovo then enumerates (possibly a subset of) paths that traverse one or several components in a heuristic way. The path finding generally follows the topological order when traversing the graph. Within each component, REPdenovo takes a heuristic approach for finding a valid path that allows the traversal to find long paths. In particular, when there are multiple edges to follow from the current node during the path finding, edges that agree with the linear order and lead to the nearest node in the linear order are preferred. To avoid cycles, REPdenovo assumes each raw contig may only appear in a path at most once. That is, each path contains distinct nodes in G. REPdenovo only outputs the maximal paths in G (i.e. paths that are not sub-paths of another path). Empirical results show that this path finding approach works reasonably well in practice.

2.3.3 Improving assembly quality by filtering

To further improve assembly REPdenovo uses two filtering steps. First, before assembling the raw contigs, contigs that have no (or very low) sequence read coverage are removed. Second, we truncate a raw contig if its coverage is uneven. We align the raw reads back to the raw contigs using "bwa mem" (99) with "-a" option. Then, we calculate the coverage for each base of each raw contig. If the average coverage

is lower than a threshold value (2 by default), then this contig is considered to be wrongly assembled and is discarded. For example, in Part (e) of Fig. 2.3.1, the raw contig marked in red color is discarded since it has no mapped reads. Sometimes a contig has uneven coverage, which means parts of the contig have high coverage and other parts have very low coverage (lower than the threshold). Such contigs are truncated so that only the high coverage parts are kept. For example, in Part (e) of Fig. 2.3.1, the right part of the lower right raw contig (marked with the purple color) is truncated due to low read coverage.

2.3.4 Evaluation and comparison of methods

Throughout this chapter, we use NCBI Blast (the output of blastn with default cutoff parameters which is considered to be "significant hits") to compare a query sequence against a set of reference sequences. We define "matching cutoff" as the ratio between the length of the matched part (between the query sequence and the reference sequence) and the length of the query sequence. Notice that we use Blast searches in two different settings:

1. To compare a query sequence against a set of reference sequences such as the reference genome of a species or a set of estimated repeats (e.g. Blast a constructed repeat against Repbase). We call this use of Blast "Blast".

2. Sometimes we want to search for a query sequence in the entire known nucleotide database at NCBI. We call this use of Blast as "NCBI Blastn".

For a mapping between a repeat and the reference genome, we use a matching cutoff of 0.0 as default, which means we allow any matches that are considered to be

TABLE 2.3.1: Sequence reads information from four human individuals from the 1000 Genomes Project. # of reads: in millions. Coverage: average sequence depth per base.

Individual	Population	# of reads	Read length	Coverage
NA12889	CEU	229M	101	7.2
HG01890	ACB	394M	100	12.3
NA18641	CHB	254M	101	8.0
NA19206	YRI	172M	100	5.4

significant by Blast (with default parameters). For other matchings, unless otherwise stated, the default matching cutoff is at least 85%.

In order to evaluate the performance of REPdenovo on repeat assembly, we first analyze raw sequence data from a human individual: individual NA12889 from the 1000 Genomes Project (100). In the following, the repeats are assembled from the NA12889 reads, unless otherwise stated. To evaluate the consistency of REPdenovo, we also use REPdenovo to assemble repeats with three other 1000 Genomes individuals: HG01890, NA18641 and NA19206. See Table 2.3.1 for information on the reads data. See the Supplemental Materials for the source of data. We first compare two different k-mer frequency cutoff f_K values of 10 and 100 to evaluate the performance of REPdenovo on different parameters. We thereafter mainly use $f_K = 10$ unless otherwise stated.

2.4 Results

The results section is organized as follows: First, we evaluate the performance of REPdenovo using sequence reads from four human individuals. We show that the found

repeats are likely to be real and many are absent from the human reference genome. Then, by comparing with repeat annotations stored in existing repeat libraries and latest long human sequence reads, we identify and validate a set of potentially novel repeats in the human genome that are not included in existing repeat annotations. At last, we show REPdenovo outperforms RepARK in terms of the accuracy and completeness of the constructed repeats.

2.4.1 Constructed human repeats are likely to be real

Classification of constructed repeats. We use REPdenovo to construct consensus repeats from reads data of the human individual NA12889. For each repeat, we address the following two questions:

1. Is the repeat mappable to the human reference genome?

2. Can the repeat find homologs in a NCBI Blastn search?

In general, if a repeat can be mapped to the human reference, the repeat is more likely to be real (i.e. not introduced by assembly artifact). Since many existing methods rely on the reference genome in repeat analyses, reads identified from reference genomes are likely to be incorporated in such analyses (e.g., in Repbase). Novel repeats (i.e, not previously reported in humans) in contrast are not mappable to the human reference. However, inferred new repeats that do not map to the genome may in reality be assembly artifacts. In order to identify repeats that are more likely to be real, we search for homologs of the each consensus repeat using NCBI Blastn. If the consensus repeat is represented in the *nt* NCBI database it is unlikely to be an assembly artifact.

For NA12889, 5,479 out of 6,200 for $f_k = 10$, and 589 out of 669 for $f_k = 100$, estimated repeats from REPdenovo are mappable to the human genome. Furthermore, 190 out of 721 for $f_k = 10$, and 78 out of 80 for $f_k = 100$ can find high quality NCBI Blastn hits even when they are not mappable to the human reference (see Table 2.4.1 for details). This suggests that the vast majority of constructed repeats are real and not assembly artifacts. It also suggests the possibility that a significant number of human repeats may be absent from the current human genome reference.

TABLE 2.4.1: The number of classified repeats constructed by REPdenovo on four different human individuals for $f_K = 10$ and 100. Classified into: (i) mappable to the reference genome, (ii) unmappable to the reference but have NCBI Blastn hits, and (iii) unmappable to the reference and no NCBI Blastn hits. The repeats in (ii) and (iii) may potentially be previously unknown repeats.

f_K	Individuals	Mappable	Unmappable		Total
			NCBI Blastn hits	No NCBI Blastn hits	
10	**NA12889**	5,479	**190**	**531**	6,200
	NA18641	5,626	189	764	6,579
	NA19206	6,055	150	603	6,808
	HG01890	5,606	171	691	6,468
100	NA12889	589	78	2	669
	NA18641	610	83	8	701
	NA19206	646	57	11	714
	HG01890	609	80	6	695

Length distribution of matched repeats. Fig. 2.4.1 shows the length distribution of the human repeats that are either mappable to the reference or have good NCBI Blastn hits for $f_K = 10$ and 100. The solid bars show the matching length distribution for repeats that are mappable to the reference. The bars with patterns are for those not mappable to the reference but having good NCBI Blastn hits. Most repeats have matching ratios of 100% or nearly 100% (i.e. the entire repeat can be mapped) for

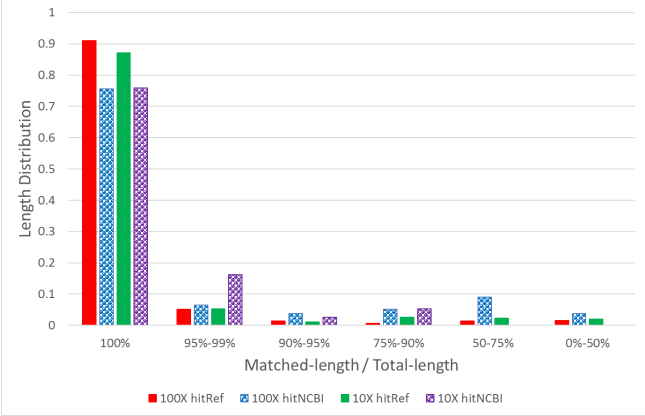

FIGURE 2.4.1: *Distribution of repeat matching lengths relative to their total length for* $f_K = 10$ *and* $f_K = 100$. *Solid bars: repeats mappable to the reference genome. Bars with patterns: repeats unmapped to reference and having NCBI Blastn hits. The figure shows the relative matching length as the mapping ratio (0%-100%), which is the ratio between the length of mapped part and total length of the repeat. A majority of constructed repeats can match fully to the reference genome or have NCBI Blastn hits.*

both reference matching and NCBI Blastn hits. That is, when a repeat is matched, it is quite likely the whole repeat can be matched to the reference genome or to the NCBI *nt* database. This suggests assembly accuracy of the constructed repeats may be high.

REPdenovo constructs repeats with high copy numbers and low sequence divergence. REPdenovo works better for low divergent than for high divergent repeats. To illustrate this, we show a distribution of constructed repeats as a function of copy number and repeat divergence in Fig. 2.4.2. We use matching cutoff 0.0 when comparing with Repbase repeats. To get the copy number and divergence of repeats, we use UCSC annotation (103), which utilizes a copy number generated by RepeatMasker. There are 1,119 human repeats in Repbase. Here, we only use 1,001 repeats out of all repeats which exist in the UCSC annotation. 283 out of the 1,001 repeats have a hit among the constructed repeats. From Fig. 2.4.2, it is clear that

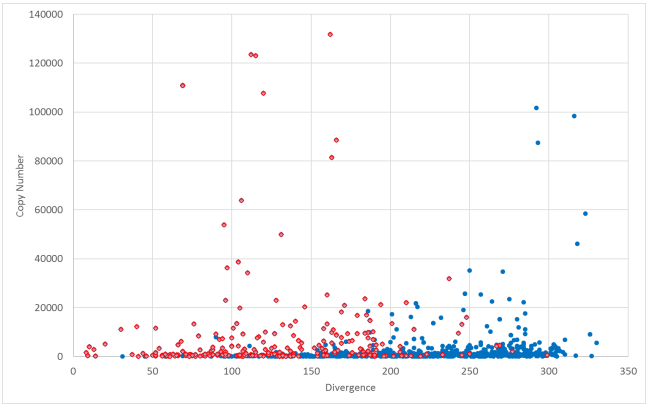

FIGURE 2.4.2: *Hits of Repbase repeats found by REPdenovo. X axis: divergence rate (mismatches per 1,000 bases) of repeats given by Repbase. Y axis: number of copies from the UCSC genome browser annotation. Dots: Repbase repeats. Red dots: hits found by REPdenovo. Blue dots: repeats not found by REPdenovo.*

most of the hits are on repeats of low divergence and high copy number.

Consistency of different human individuals. To evaluate the consistency of REPdenovo, we use REPdenovo to construct repeats from four human individuals using data from the 1000 Genomes Project. The results are shown in Table 2.4.1. It can be seen that the numbers of repeats in different categories are overall consistent for different individuals for both $f_K = 10$ and $f_K = 100$ cases. Differences might reflect the variations in repeats in different human individuals, differences in read quality and sequencing depth between individuals.

2.4.2 Comparison to Repbase and RepeatMasker and finding novel repeats in human genome

One of our main goals is finding novel human repeats that are previously unknown. For this purpose, we first map the constructed repeats to human reference. As expected, repeats that are mapped to the reference are more likely to find matches in Repbase. Repeats unmappable to human reference may be novel. We rely on two

means to validate whether a repeat is novel: (i) perform NCBI Blastn search: a repeat with good hits is likely to be real; (ii) compare with the latest long human reads: if a repeat matches the long reads data, it is likely to be real.

Repeats with Repbase hits and/or masked by RepeatMasker. Table 2.4.2 shows the overall results on the number of constructed repeats. For both mappable and unmappable repeats, first we examine whether the repeats can find matches in Repbase. Then we run RepeatMasker on the inferred repeats to examine whether the repeats can be classified into a particular type. Here, the matching cutoff for NCBI Blastn is 0.0, and is also 0.0 for both Repbase hits and "masked" by RepeatMasker. As expected, repeats mappable to the reference are more likely to find matching Repbase repeats, while unmappable repeats have no Repbase hits. RepeatMasker can give results on many repeats, although sometimes only small parts of repeats can be masked.

TABLE 2.4.2: Numbers of repeats that hit Repbase (with matching cutoff 0.0) and masked by RepeatMasker (with matching cutoff 0.0). We classify the repeats based on whether they are mappable to the human reference and whether they have matches in Repbase. Masked: RepeatMasker can classify the repeat. Unmasked: RepeatMasker cannot classify the repeat.

f_K	Mappable to reference				Unmappable to reference			
	Hit Repbase		No-hit Repbase		Hit Repbase		No-hit Repbase	
	Masked	UnMasked	Masked	UnMasked	Masked	Unmasked	Masked	Unmasked
10	3,426	26	1,239	788	0	0	683	38

Potentially novel repeats in human. There are 190 repeats for NA12889 in Table 2.4.1 that do not have significant hits on the human reference (GRCh37) but find significant and nearly complete hits in NCBI Blastn. In Table 2.4.3 (the first column) we give out a detailed statistic of these hits. Note that, one repeat may have more than one hits and here only the top one (blast with option "-max_target_seqs

1") is used. We believe that these 190 repeats are potentially novel repeats in human, although some may be assembly artifacts. We run Blast on these 190 repeats against the constructed repeats of the other three individuals. Out of the 190 repeats, 152, 156 and 157 repeats can find Blast hits on NA18641, NA19206 and HG01890, respectively. We also run RepeatMasker on these 190 repeats. Among the 190 repeats, 129 are masked, and among these, 101 are identified to be "Satellite repeat", 25 are "Simple repeat", and three are "centromeric". We note that RepeatMasker tends to mask the shorter ones among these 190 repeats. This is illustrated in Table 2.4.4. Note that fewer repeats are masked by RepeatMasker than those in Table 2.4.2 because here we require near complete matching when running RepeatMasker.

TABLE 2.4.3: Classification of the 190 un-mappable repeats with NCBI Blastn hits. The numbers in parentheses are the numbers of repeats in each category.

NCBI Blastn hit description	All (190)	Long read reference hits (164)	Masked (129)
Epstein-Barr virus	*2*	*0*	*0*
Homo FOSMID clone	8	8	4
Homo BAC clone	8	8	2
Homo related	35	33	29
Gossypium hirsutum clone	2	2	0
Haemonchus placei genome	51	51	50
Human herpesvirus	*19*	*0*	*0*
Onchocerca flexuosa genome	6	5	3
Protopolystoma xenopodis genome	18	18	5
Spirometra erinaceieuropaei genome	40	38	36
Toxoplasma gondii ME49	1	1	0

Compare novel repeats with long reads data. To further validate that the 190 repeats indeed are present in the human genome, we compare them against the Pacific-Bio long reads released in (104). These reads are generated from a human hydatidiform mole cell line (CHM1) from SMRT sequencing (104). As the error rate

TABLE 2.4.4: Length distribution of the 190 potentially novel repeats. The numbers in parentheses are the numbers of repeats in each category. For each range of repeat lengths, the number is the percentage of repeats falling in the range.

length	All (190)	Mappable to long reads reference		Masked by RepeatMasker	
		Yes (164)	No (26)	Yes (129)	No (61)
100-300	79.4	87.8	27.0	86.8	63.9
301-500	9.5	9.2	11.6	9.3	9.9
501-750	1.1	0.6	3.8	0.8	1.6
751-1,000	2.6	2.4	3.8	3.1	1.6
1,001-1,500	0.5	0	3.8	0	1.6
1,501-2,000	1.1	0	7.7	0	3.3
2,001-5,000	1.1	0	7.7	0	3.3
5,001-50,000	4.7	0	34.6	0	14.8

of the Pacific-Bio reads is usually high, we use the released error-corrected reads which are corrected by a multiple read alignment procedure. 164 and 161 out of the 190 repeats match one or more of the Pacific-Bio reads with matching cutoff 0.85 and 0.95, respectively.

There are two recently released human reference genomes based on Pacific-Bio reads, and we also Blast the 190 novel repeats against these. One reference is directly assembled from Pacific-Bio reads (104), while the other is based on extending the gaps in GRCh37 using Pacific-Bio reads (105). We get different results when blasting the repeats against these two references. For the directly assembled reference (104), we find 164 hits with matching cutoff 0.85, while for the patched reference (105) we only find 1 hit using the same cutoff. It is possible that the reference constructed by GRCh37 has missed some hard-to-assemble regions.

There are 531 repeats in Table 2.4.1 for NA12889 that cannot map to the reference and have no NCBI Blastn hits. We also Blast these 531 repeats against the corrected long reads. 18 and 16 out of the 531 repeats match at least one long read with

matching cutoff 85% and 95%, respectively. Thus, there appears indeed to be a small number of novel valid repeats even among the ones that do not have a significant NCBI Blastn hit.

Overall, the fact that a majority of repeats match at least one Pacific-Bio read with at least 95% identity across the entire repeat sequence provides additional support for our belief that the majority of inferred reads are real and are not assembly artifacts. However, we note that this does not completely rule out the possibility of assembly errors.

Further analysis of potentially novel repeats in Human. Among the 26 repeats with no long reads hits, but with significant NCBI Blastn hits, the average NCBI Blastn identity is 99.5%. Average coverage (alignment length/repeat length) is 98.5%. The largest E-value of the 26 repeats is 2.0e-49 and 17 repeats have E-value reported to be 0.0. We also note that the average length of the 26 repeats is 7,988 bp, while the average length of all 190 repeats is 1,266 bp (see Table 2.4.4). This may suggest that the long repeats are poorly assembled even in the Pacific-Bio reference genome.

We classify all 190 potentially new repeats according to whether they find matches in the new reference sequence or are masked by RepeatMasker in Table 2.4.3. The general pattern mirrors that for the 26 repeats with not long read hits. However, we now observe an increased proportion of repeats that previously have been classified as human, for example to sequenced BAC clones that are not incorporated into an assembly. We also observe an increase in hits to specific blood parasites, particularly *Haemonchus placei* and *Spirometra erinaceieuropaei*.

We see that a substantial proportion of the hits are viral. The EBV virus is not surprising as it has been used to transfect the sequenced cell lines. Matches to other viruses (e.g., herpes virus) may be caused by contamination of the sequencing

libraries or the cell cultures or by homology with the transfecting virus. We note that three herpesvirus have high sequence homology with parts of two EBV virus. We also note that there is no hits to the long reads reference for the repeats among the viruses. This supports our hypothesis that these repeats in fact reflect homolgy with transfecting viruses or contamination.

The remaining hits are all blood parasites. It is likely that these are real repeats that have been incorporated into the parasite assemblies by error, as sequencing of parasites typically is based on samples contaminated with the host DNA, which can be removed by filtering. However, if the host reference sequence is missing the sequence motif, such filtering may fail. Repeats motif not present in the reference genome of the host and not caught by RepeatMasker are likely to fall into this category.

We take a closer look at the 26 repeats that do not hit the new long reads reference. Table 2.4.5 shows the top hits from NCBI Blastn for these 26 repeats, and the number of repeats masked (all are masked as "Simple repeats") by RepeatMasker for each type. The numbers inside parentheses are the number of occurrences of the repeats of the specific types.

TABLE 2.4.5: Classification of the 26 (out of the 190 potentially novel repeats) repeats that have no Blast hits on long reads reference. The numbers in parentheses are the numbers of repeats in each category.

NCBI Blastn hit description	All (26)	Masked (18)
Homo sapiens isolate satellite	2	2
Human herpesvirus	19	11
Onchocerca flexuosa genome	1	1
Spirometra erinaceieuropaei genome	2	2
Epstein-Barr virus	2	2

2.4.3 Comparison of assembly quality of REPdenovo and RepARK

We compare REPdenovo to RepARK repeat assemblies by comparing both to the repeats represented in Repbase (87). In the following, "hits" refer to constructed repeats that are represented in Repbase, and we use the following metrics to compare REPdenovo to RepARK:

1. The number of Repbase hits with $> 85\%$ sequence identity across the length of the Repbase represented repeat sequence.

2. Average Repbase coverage. For a Repbase hit, this is the average fraction of the Repbase repeat covered by the assembled sequence. For a single position of a hit, there can be multiple assembled repeats covering it. When calculating the average coverage, we use the set of non-overlapping assembled repeats that achieve the largest coverage. This statistic can be computed by a simple greedy algorithm.

3. Average Repbase coverage by the longest assembled repeat. One repeat in Repbase may be covered by several constructed repeats. When calculating the average coverage, we choose the longest one. This statistic is used to examine how well the methods can construct long repeats (not just fragments of repeats).

4. N50 of the assembled repeats.

We run REPdenovo and RepARK with same kmer frequency and assembly parameters (both use velvet (131) as the assembler). The results of REPdenovo and RepARK on individual NA12889 are given in Table 2.4.6. REPdenovo outperforms

TABLE 2.4.6: Assembly quality comparison of REPdenovo and RepARK. N: the number of assembled contigs. N_h: the number of complete Repbase hits from the N repeats (with 85% coverage cutoff). $\overline{C}$: average coverage of hits. C_m: maximum coverage of hits by single assembled repeats. N50: N50 of assembled repeats.

f_K	Method	N	N_h	$\overline{C}$	C_m	N50
10	REPdenovo	6,200	**91**	0.88	0.53	**3,141**
	RepARK	7,894	**1**	0.74	0.06	**116**

RepARK in all quality metrics. In particular, REPdenovo can assemble longer repeats while RepARK tends to assemble smaller fragments of repeats. REPdenovo also achieves higher average coverage of Repbase repeats than RepARK. The N50 of REPdenovo assembled repeats is about 27 times of that of RepARK assembled repeats. As an example, we use one Repbase repeat, AluYd3, for an illustration. This is shown in Fig. 2.4.3, generated by mapping the assembled repeats on Repbase repeats and then visualizing with the program IGV (106). The length of the AluYd3 repeat is approximately 270 bp. In this case REPdenovo almost assembles the complete repeat while RepARK only assembles two small fragments. This illustrates the stark difference in completeness and length of constructed repeats between the two methods as measured by N50 and other statistics.

To further compare the performance of REPdenovo and RepARK, we run REPdenovo and RepARK on four 1000 Genomes individuals: NA12889, HG01890, NA18641 and NA19206. We use Blast to identify matches in Repbase and investigate how well long repeats are constructed with matching cutoff t_L.

Table 2.4.7 shows the repeat assembly performance on all four individuals (including NA12889). We can see that results from REPdenovo overall keep consistent as the matching cutoff t_L changes from 0.0 to close to 1.0 (by default, t_L is chosen

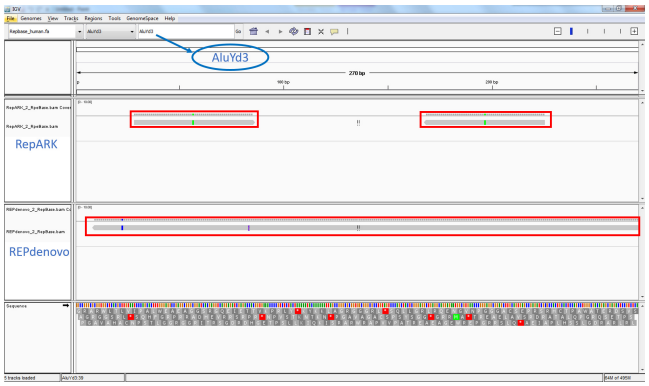

FIGURE 2.4.3: *Assembled repeats matching AluYd3 (a Repbase repeat) by REPdenovo (bottom panel) and RepARK (top panel). The matched assembled repeats are shown on their mapped positions where the AluYd3 consensus repeat sequence serves as the reference.*

to be 0.85). Also, REPdenovo outperforms RepARK in terms of the completeness and the number of long repeats constructed. This is benefit from the assembling raw contigs and filtering steps. As copies of a repeat are diverged from each other, lots of short pieces (of copies) will be assembled because of the variations on the copies, and assembling these raw contigs will help to construct the complete repeats, while RepARK only reports these pieces. Thus, REPdenovo works better for constructing more diverged repeats. However, it is still possible for REPdenovo to wrongly assemble contigs, even though there is a filtering step.

2.4.4 Running time

For a small genome and low-coverage sequence data, REPdenovo usually runs fast. It takes about 19 hours (including the k-mer counting time) to analyze the human individual NA12889 (read length 100bp with read depth 7.2X) on a 3.2 GHz eight core Xeon X5482 computer with 32G of memory.

TABLE 2.4.7: The number of repeats in Repbase that match (over the minimum threshold t_L) one *de novo* repeat. The numbers outside and side the parentheses are REPdenovo and RepARK results, respectively. t_L: matching cutoff.

f_K	t_L	NA12889	HG01890	NA18641	NA19206
10	0.95	86 (1)	81 (0)	84 (0)	84 (1)
10	0.85	91 (1)	87 (0)	94 (0)	94 (1)
10	0.75	102 (1)	100 (1)	104 (1)	102 (3)
10	0.5	141 (9)	138 (10)	135 (12)	139 (12)
10	0.0	295 (278)	307 (294)	312 (293)	313 (296)

2.5 Discussions and Conclusion

In this chapter, we develop REPdenovo, a new repeat analysis approach that reconstructs repeat sequences from raw sequence reads and does not rely on prior knowledge of repeats. REPdenovo can be applied to genomes that have been sequenced but for which no good reference genomes and repeat annotations are available. REPdenovo improves upon a previous approach, RepARK, by providing better assemblies of repeat consensus sequences in terms of completeness and number of long repeats constructed, as demonstrated by our analyses of human annotated repeats. REPdenovo can assemble full (or nearly full) repeat consensus repeats, while RepARK usually only produces small fragments of long repeats (see Table 2.4.6 and Table 2.4.7). This is especially important for downstream analyses of the identified repeat sequences. While REPdenovo may only identify recently expanded repeat families, these are also the families that are of greatest interest in comparative studies, as older repeats tend to be shared among species.

Like most bioinformatics tools, REPdenovo requires specification of several pa-

rameters, which can significantly affect the results. The most important parameter is the relative k-mer frequency cutoff f_K, which specifies the lowest k-mer frequency that a k-mer is considered to be frequent and assembled. The default value of f_K is 10, which means the frequency of a frequent k-mer is over 10 times of the read depth. When a higher value is used for f_K, fewer repeats will be assembled. Also, the running time of REPdenovo will increase when f_K decreases. Another important parameter is L_K, the length of k-mers. There is no rigorous way for choosing a single value of L_K. Shorter k-mers are more robust against variations within repeats but may give less accurate assemblies due to ambiguity in assembly process. Longer k-mers may give more accurate assemblies but may miss some segments that contain more variations within the repeats. Thus, REPdenovo uses multiple L_K values when assembling raw contigs, while RepARK only uses a fixed L_K value (i.e. 30). Table 2.5.1 shows the number of repeats in Repbase that match (over the minimum threshold 0.85) one *de novo* repeat for different L_K. The results show that different L_K values may generate different sets of repeats, and combining these repeats may provide more accurately assembled repeats.

TABLE 2.5.1: The number of repeats in Repbase that match (over the minimum threshold 0.85) one *de novo* repeat for different k-mer length L_K. By default, REPdenovo use different k-mer length (29,39, and 49) together, and its result is marked as "Combined".

L_K	21	29	39	49	59	69	79	89	99	Combined
Hit Repbase	13	49	61	71	75	71	54	57	46	91

We applied the method to human data and identified 190 potentially new repeats. We note that top Blast hits are non-human for some REPdenovo assembled repeats. For example, the top two hits for one assembled human repeat from NA12889 are for Onchocerca Flexuosa (a deer parasite) and Protopolystoma Xenopodis (an amphibian

parasite). We also find the assembled repeats that have top Blast hits (with 100% coverage) on Onchocerca Flexuosa and Protopolystoma Xenopodis exist in the other three human individuals as well. One explanation is that there are homologous repeats with high sequence identity between humans and the parasites, perhaps because these are sequences that have jumped genomically, through unknown mechanisms, between hosts and parasites. A more likely explanation is that these are repeats caused by human contamination in the parasite sequencing projects.

Moreover, the newly available long Pacific-Bio reads provided additional support that the novel human repeats we constructed may indeed be real. For the 190 potentially novel human repeats, 129 repeats are masked by RepeatMasker to be mostly simple repeats or satellite repeats. Further studies are needed to find the types of the remaining repeats.

Chapter 3

An Improved Approach for Reconstructing Consensus Repeats from Short Sequence Reads

3.1 Introduction

Constructing consensus repeats is necessary for genome annotation, repeat masking, and repeats evolution studies. There are tools designed for constructing consensus repeats from reference or draft genomes, like RepeatScout (89), PILER(31) and phRAIDER (90). One limitation of these tools is that they all require high quality draft or reference genomes to construct consensus repeats. However, for complex (e.g. highly repetitive) genomes or genomes from some recently sequenced species, there are only low quality assembled genomes available and often no existing annotated repeat libraries. Thus, it is useful to develop tools for analyzing repeats directly from short reads, without the need of high quality reference or draft genomes. Recently, we

developed REPdenovo (50), a computational approach for constructing repeats directly from short sequence reads. However, REPdenovo doesn't work well for highly divergent or low copy number repeats.

In the chapter, we propose an improved method for reconstructing repeat elements from short reads. Similar to the original REPdenovo, our new method also finds and assembles these highly frequent k-mers to form consensus repeat sequences. Here are the two main improvements over the original REPdenovo:

- Our new method uses more repeat-related k-mers than the original REPdenovo for repeat assembly, and can assemble longer consensus repeats.
- Our new method runs a randomized algorithm to generate more accurate consensus k-mers than the original REPdenovo. This improves the quality of the assembled repeats.

Comparing to the original REPdenovo and RepARK, our new method can construct more fully assembled repeats in Repbase on both Human and Arabidopsis data, especially for higher divergent, lower copy number and longer repeats. We also apply the new method on Hummingbird data, which has no existing repeat library. Most of the repeats constructed by our new method for Hummingbird can be fully aligned to PacBio long reads. Many of these repeats are long. More than half of the Hummingbird repeats are masked by RepeatMasker, which suggests that our assembly works well. Moreover, many of the assembled repeats are likely to be novel because there are no matches in RepBase, which suggests these may be present in only Hummingbird or its close related species.

3.2 Method

Similar to the original REPdenovo, our new method assembles consensus repeats directly from sequence reads. The high-level procedure is shown in Figure 4.2.1. In the following, we first provide a brief description on the repeat assembly procedure with frequent k-mers that is used by the original REPdenovo. We then illustrate the key technical problems that make the original REPdenovo perform poorly on highly divergent and low copy number repeats. We present two approaches that are implemented by our new method. These approaches allow better construction of highly divergent or low copy number repeats.

3.2.1 Repeat assembly from frequent k-mers

For completeness, we provide a brief introduction on repeat assembly from frequent k-mers. Repeats usually have many copies in the genome. For low divergent and high copy number repeats, k-mers generated from copies of the same repeat at the same position will be identical with high probability. Thus the frequencies of such k-mers will be higher than those of k-mers from non-repetitive regions. Thus, with given cutoff (say n times of the average k-mer frequency, where n can be viewed as the copy number), these highly frequent k-mers from repeats can be identified, while the less frequent k-mers will be discarded since they are unlikely to come from repeats. Now if we view the repeats as "genomes" and the frequent k-mers are the "reads" as in genome assembly, the repeats can be assembled from these frequent k-mers using standard genome assembly tools such as Velvet (131). This is the key observation of RepARK and the original REPdenovo. However, in practice complete consensus repeat sequences can rarely be assembled in this way. This is because the

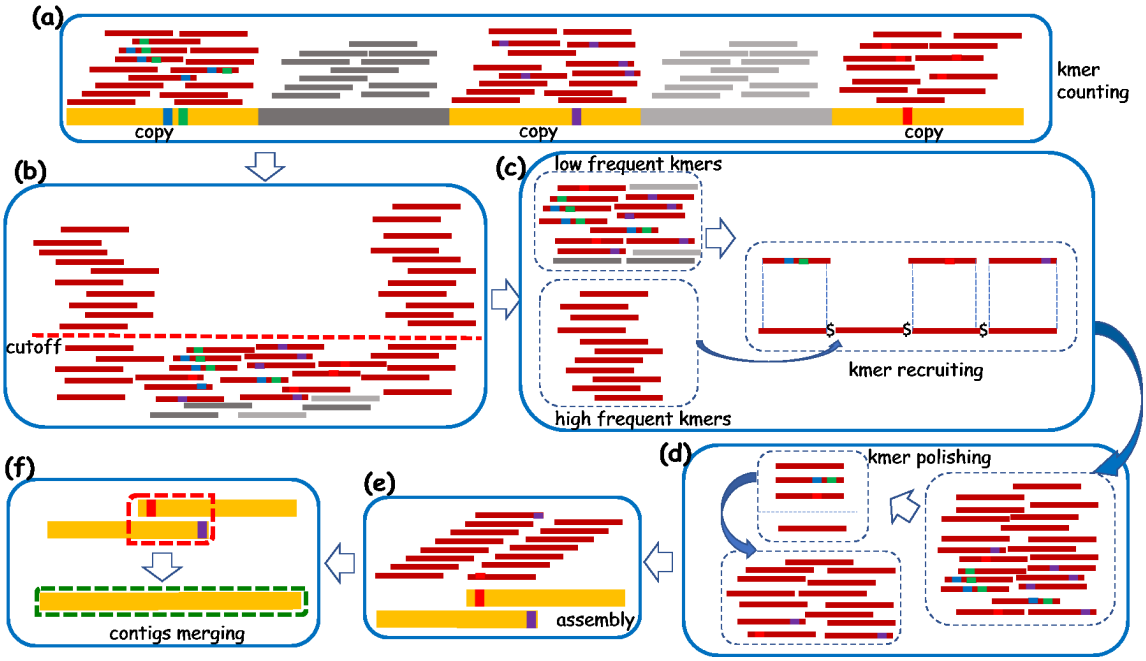

FIGURE 3.2.1: High-level procedure of improved repeat construction. Thick bars: genomic sequences. Yellow thick bars: repeat copies. Colored squares within thick bars: mutations (substitutions and indels) within repeats. Thin bars: k-mers. There are six main steps. (a) K-mer counting for the reads. (b) Find the highly frequent k-mers and k-mers with intermediate frequencies according to a user-specified cutoff on k-mer frequency. (c) Find repeat-related k-mers by aligning those k-mers of intermediate frequencies to highly frequent k-mers. (d) Improve k-mer quality with a consensus-based approach. (e) Assemble the improved k-mers. (f) Merge contigs that have reliable prefix-suffix overlap.

variations on repeat copies and also read errors make the repeat copies divergent from the consensus. As the result, even for low divergent repeats, usually only short contigs can be directly assembled. Figure 3.2.2 (b) shows one such situation. The improvement made by the original REPdenovo is that it performs a second-round assembly: it tries to assemble short contigs to form longer consensus repeats based on reliable prefix-suffix matches of the contigs. Refer to (50) for more details.

3.2.2 The difficulty of assembling highly divergent repeat regions

The main problem with the original REPdenovo is that it cannot fully assemble highly divergent or low copy repeats. Even when a long repeat is overall of low divergence, there may still be regions with high divergence rate. In this case, the original REPdenovo also cannot assemble the high divergent regions within a repeat. There are two reasons. First, when the repeat divergence rate is high or the copy number is low for a region, k-mers originated from the this region will likely be of low frequency and thus are discarded. As a result, when repeat assembly is performed, only fragments of repeats will be obtained since k-mers from the highly divergent regions are missing. Moreover, even though some k-mers from highly divergent regions are present in repeat assembly, it is still challenging to assemble whole repeats. This is because contigs may break at regions with sequence variations. In Figure 3.2.2 we show two examples to illustrate these two issues. First, we obtain highly frequent (at 10 times of the average k-mer frequency) 30-mers from real reads of one human individual NA19239. Then we align these 30-mers to the human consensus repeats released in Repbase. Two alignment cases on repeats "LTR2B" and "LTR10C" are shown in Figure 3.2.2 through IGV (106). The left alignment is for "LTR2B" with length 490bp. Apparently, there are two gaps with very low or no k-mer mapped, which will cause the assembly of this repeat to have at least 3 segments. The right alignment is for "LTR10C" which has more variations (the colored bars). When there are variations, genome assemblers e.g. Velvet (131) usually construct contigs that are short and fragmented.

In order to assemble repeats with higher divergence rates, we need to find more

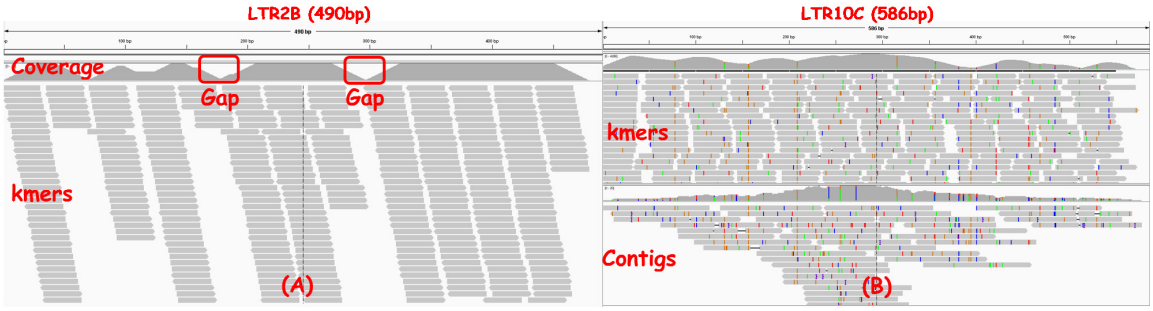

FIGURE 3.2.2: Illustration of two example repeats that are not fully constructed by the original REPdenovo. Highly frequent 30-mers of one human individual NA19230 are aligned to the human consensus repeats in Repbase. The left part (a) shows the alignments on repeat "LTR2B". Two gaps are formed when 30-mers originated from highly divergent regions have low frequencies due to repeat copy divergence. The right part (b) shows the alignments on repeat "LTR10C". The colored bars are variations on copies. The assembled contigs are fragmented because the 30-mers are of highly divergence.

repeat-related k-mers that originate from highly divergent repeat regions. In the following, we first describe a new method for finding such less frequent repeat-related k-mers. We then use these repeat-related k-mers to improve the quality of the assembled repeats.

3.2.3 Mapping-based alignment for finding more repeat-related k-mers

We now focus on assembling repeats that have higher divergences and/or lower copy numbers than those constructed by the original REPdenovo. Many k-mers from highly divergent regions may have relatively low frequencies. These k-mers are then discarded and are not included for repeat assembly. A main observation is that these discarded repeat-related k-mers usually have high sequence identity with some repeat-related k-mers with high-frequency. Recall that k-mers with high frequencies are likely to come from some repeats. Thus, if a k-mer is similar to some highly frequent k-mer,

this is an indication that this k-mer is also related to some repeat. Therefore, we can compare the sequences of all the discarded k-mers with the highly frequent k-mers. If a discarded k-mer has reliable prefix-suffix match with some highly frequent k-mer, this k-mer should be kept for repeat assembly. However, direct comparison of all pairs of lower frequency and high-frequent k-mers using dynamic programming is infeasible empirically. This is because the number of lower frequency k-mers can be very large (usually in millions), and there can also be many highly frequent k-mers.

To develop a practical method, we take the following "mapping-based alignment" approach. The key idea is creating a "reference k-mer genome" by concatenating all the high-frequent k-mers. We then view the less frequent k-mers as "reads". The reads mapping tool BWA (99) is used to align these "reads" to the reference k-mer genome. The mapped k-mers are kept for repeat assembly. This is shown in step (c) of Figure 4.2.1. This approach works because we only want to find k-mers that have high sequence similarity with some high-frequent k-mers. Our experience shows that reads mapping tools work well for this purpose. The main benefit of mapping-based alignment is that it allows small insertions and deletions, and thus can find more repeat-related k-mers. We only consider the lower frequency k-mers that are of intermediate frequency (by default three times or more over the read depth). This not only speeds up the computation and also reduces false positives. This is because k-mers from highly divergent parts of repeats still tend to have frequencies higher than average. BWA "mem" is used with option "-T" to set the minimum score for the alignments. Since we want to avoid false positives, we use no penalty for mismatch, gap open, gap extension and mismatch, and set $k-5$ as the minimum score by default for reads mapping. This step can be performed iteratively if users want to construct more fully constructed repeats. Note that this step may introduce some unrelated

k-mers and lead to false positives in repeat assembly. Thus, there is a trade-off in determining how many times this step is run. The mapped k-mers are merged with the highly frequent k-mers and are used as input of the next step.

3.2.4 K-mer polishing

As illustrated in Figure 3.2.2 (b), k-mers from repeat copies with variations can often only be assembled to form short contigs. If there are only mismatches on the two k-mers from the same position of two repeat copies, most positions of the two k-mers are still the same. We call these two k-mers "end-to-end" matched. Now suppose there is a single inserted (or deleted) base at the beginning of a k-mer. Then k-mers started from the insertion (or deletion) will be "end-to-end" matched with the k-mer from the other copy that is one base left (or right). "End-to-end" match can be used to generate the consensus k-mers. Consensus k-mers can be more reliable to use for repeat assembly for highly divergent repeats.

Given the merged k-mers (highly frequent and also the mapped intermediate frequent k-mers) generated by mapping-based alignment, a randomized algorithm is used to generate the "end-to-end" matches. For two k-mers with length k, we randomly pick h bases from the same positions of the two k-mers. If the chosen h-mers are the same, then the two k-mers will be considered as "end-to-end" matched. This procedure runs for n times to guarantee the two "end-to-end" matched k-mers are grouped together. Here, we require at least one match between the two h-mers out of n times to group the two k-mers. Given the values for n, k, h, the probability p of

two k-mers being grouped is:

$$p = 1 - (1 - \frac{\binom{k-e}{h}}{\binom{k}{h}})^n$$

Here e is the allowed edit distance between two "end-to-end" matched k-mers. By default, the value of e is set to 1, that is, we allow one mismatch, insertion or deletion in one k-mer. This is reasonable because usually k is not large (less than 100).

When matched k-mers are found, we use a weighted voting method for constructing the consensus k-mer. For each position, each k-mer votes for one of the four possible bases with weight f, where f is the frequency of the k-mer. The base with maximum votes is chosen as the base at that position. The maximum vote out of all positions is considered as the final frequency of the consensus k-mer. This step is implemented in the popular map-reduce way for efficient processing: first we partition the k-mer file into several parts, and then run the polishing step for each part. Finally we merge the results of each partition.

3.3 Results

To evaluate the performance of the new method, we compare it against the original REPdenovo and RepARK on Human and Arabidopsis thaliana data. These two species are well studied and have good quality of annotation, which can provide benchmark for our comparison. We use the repeat libraries of these two species released in Repbase (87) as the benchmark. Short sequence reads of one human individual NA19239 from the 1000 Genomes Project (28) is used. The read depth is around 6X with read length 100bp. For Arabidopsis thaliana, the F1 sample released in (26)

is used with read depth 10X and read length 250bp. We compare the divergence rate, copy number and repeat length of the constructed repeat elements. To get the divergence rate and copy number of repeats, we use UCSC annotations (41), which utilizes copy numbers generated by RepeatMasker. We also apply the new method to infer the repeat elements of Hummingbird. There is no existing repeat library for Hummingbird, but there are recently sequenced PacBio long reads (36) which can be used to validate the constructed repeats. For Hummingbird, we use the short sequence reads released in the GeneBank (accession number SRR943146), where the average coverage is around 20X with read length 101bp.

3.3.1 Comparison with Human and Arabidopsis data

We evaluate the performance of the two versions of REPdenovo and RepARK by comparing the assembled repeats from these tools with the consensus repeats released in Repbase. There are 1,119 and 525 consensus repeats for Human and Arabidopsis thaliana respectively. In the following, "hits" refers to constructed repeats that are present in Repbase. We use the following metrics previously used in(50) to compare the two versions of REPdenovo to RepARK:

1. The number of Repbase hits with $> 85\%$ sequence identity across the length of the Repbase consensus repeat sequence.

2. Average Repbase coverage. For a Repbase hit, this is the average fraction of the Repbase repeat covered by the assembled sequence. We use the set of non-overlapping assembled repeats that achieve the largest coverage.

3. Average Repbase coverage by the longest assembled repeat. One repeat in

Repbase may be covered by several constructed repeats. When calculating the average coverage, we choose the longest one.

In Table 3.3.1 we show the detailed comparison of the three methods on human and Arabidopsis thaliana data. Besides the three metrics, we also show the number of repeats in Repbase that are partially (no identity threshold requirement) constructed. The results show that both versions of REPdenovo outperform RepARK on both the number of hit Repbase repeats and the average covered repeat length. In comparison, the original REPdenovo fully constructs 89 (out of the 220 hits) and 11 (out of the 68 hits) repeats in Repbase for human and Arabidopsis respectively. And the new version of REPdenovo fully reconstructs 108 (out of the 332 hits) and 24 (out of the 102 hits) repeats in Repbase for Human and Arabidopsis respectively. Therefore, our new method significantly outperforms the original REPdenovo in terms of the number of fully constructed repeats.

Note that the C_{avg} and C_m values for the original REPdenovo are slightly larger than the improved version. This is mainly because the new method reports more repeats than the original REPdenovo. Our experience shows that the new method tends to construct copies of the same repeat with different variations and thus construct more repeats in general than the original REPdenovo. We provide more information in the Discussion section.

3.3.2 Comparison between the two versions of REPdenovo

In Section 3.2 we show the new method can find more k-mers originated from the repeat regions than the original REPdenovo. As a result, it can construct not only higher diverged regions, but also less frequent repeat elements than the original REP-

TABLE 3.3.1: Comparison between the two versions of REPdenovo and RepARK on both Human and Arabidopsis thaliana data. REPdenovo*: the new method. N: the total number of repeats constructed. N_h and N_0 are the number of hit Repbase repeats with at least 85% and 0% similarity respectively. C_{avg}: the average Repbase coverage which indicates the average percent of a repeat in Repbase is covered by the constructed repeats. C_m: the average Repbase coverage by the longest assembled repeat.

Species	Methods	N	N_h	N_0	C_{avg}	C_m
Human	REPdenovo*	6192	108	332	0.61	0.49
	REPdenovo	4648	89	220	0.66	0.55
	RepARK	2046	1	168	0.34	0.21
Arabidopsis	REPdenovo*	808	24	102	0.42	0.31
	REPdenovo	508	11	68	0.46	0.34
	RepARK	632	8	59	0.33	0.21

denovo. As shown in Figure 3.3.1, there are a number of low divergent but low copy number repeats that are only constructed by our new method.

In Figure 3.3.1, we show the comparison between the two versions of REPdenovo on the divergence rate and copy number of the constructed repeats. The bullet circle points are the repeats constructed by both versions, while the empty circle points are the repeats only constructed by the new method. Note that 211 repeats (out of the 332 repeats) are shown in the figure. This is because out of the 332 repeats only 211 can find the divergence rate and copy number information from the UCSC annotations (mainly because the IDs do not match between Repbase and RepeatMasker). The results show that most of the repeats only constructed by the new method have higher divergence rates and are less frequent.

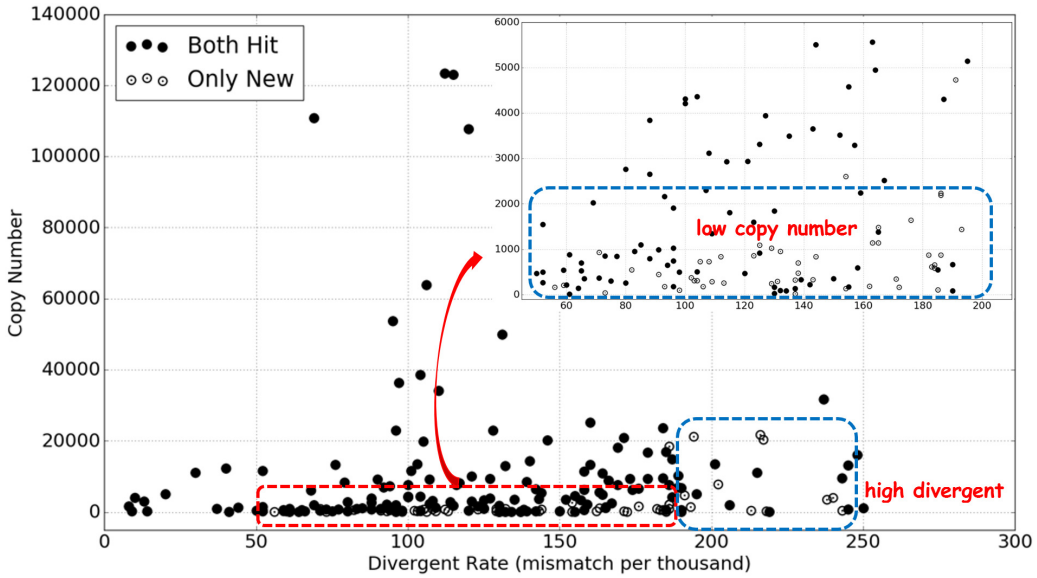

FIGURE 3.3.1: Comparison of the fully constructed repeats in Repbase for the two version of REPdenovo. Bullet circles: hit Repbase repeats constructed by both versions of REPdenovo. Empty circles: hit Repbase repeats constructed only by the new version. Figure in the left-up corner is zoomed in the red rectangle region. There are 154(out of all the 220) bullet circles and 57 empty circles. Most of these 57 ones fall in higher divergent and lower copy number regions (the regions of blue rectangles).

3.3.3 Repeat elements construction for Hummingbird

Our new method can be used to construct repeat elements for species that have no existing repeat libraries or no high quality reference genomes. We apply our new method on the Hummingbird data. 2,406 repeats are constructed. Because there is no existing repeat library of Hummingbird to compare with, we cannot directly validate the constructed repeats. Generally, long reads are long enough to cover most of the repeats which provides a way to check whether the assembled repeats are real. Thus, we run NCIB Blast on the constructed repeats to the error-corrected PacBio long reads of Hummingbird. Out of the 2,406 repeats, 1,617 are almost fully aligned (with similarity larger than 85%). Among these, 1,406 are perfectly fully aligned. This indicates that most of the constructed repeats are likely to be real. In Figure

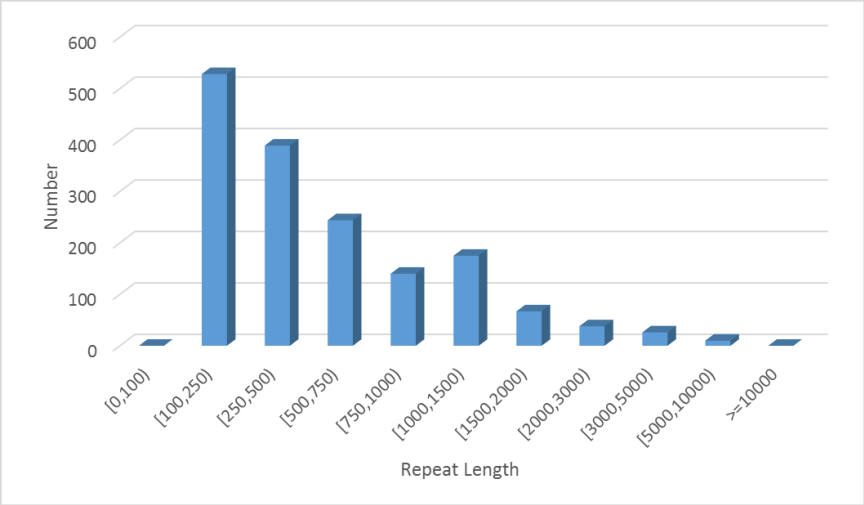

FIGURE 3.3.2: Length distribution of the selected constructed Hummingbird repeats.

3.3.2 we show the length distribution of the 1,617 constructed repeats. Most of the repeats are shorter than 1,500bp. There are 64 repeats longer than 2,000bp.

To further analyze the constructed repeats, we run RepeatMasker on the 1,617 repeats. In general, RepeatMasker relies on an external repeat library to mask the repeats, which means it will not work for Hummingbird which has no existing repeat library. However, homologous copies of repeats usually exist in multiple species. In this study, we use the "Vertebrate (Other than below)" from RepeatMasker as the DNA source to mask the constructed repeats. Out of the 1,617 repeats 928 are masked and the rest 628 ones are unmasked. Detailed information are shown in Table 3.3.2. Note that one repeat may have several regions and the regions may be of different repeat families. Thus one repeat may be reported for several times with different regions and repeat families. For the statistic in Table 3.3.2, the row marked as "Unique" only counts those repeats with one unique masked repeat family, while the row marked as "Dup." allows one repeat counted for more than once. Many of the repeats are masked as "LINE", which is supported by the known fact that

TABLE 3.3.2: Masked repeats of the 1,617 validated using long reads. For one repeat, RepeatMasker may report several hits depending on whether the repeat is composed of regions of different repeat types. "Unique" only counts those repeats with one unique masked repeat family, while "Dup." allows one repeat counted more than once.

Category	LINE	SINE	LTR	Retroposon	Satellite	Simple_repeat	Other
Unique	371	0	139	0	10	98	36
Dup.	557	6	244	0	19	216	134

"LINE" repeats widely exist in vertebrate. We believe the 628 unmasked repeats are possibly Hummingbird-only or its close relatives, because they are of high frequency and fully aligned to long reads but have no hits on the "Vertebrate" general library.

3.4 Discussion

In this chapter, we propose an improved method for reconstructing repeat elements directly from short sequence reads. Our new method is able to collect more repeat-related k-mers. Results on both Human and Arabidopsis data show that the new method can fully construct more repeats in Repbase than the original REPdenovo and RepARK, especially for repeats of higher divergence rates and lower copy number. In Figure 3.4.1, we show the comparison of the two versions of REPdenovo on constructing one sample repeat "LTR2B", which is mentioned in section 3.3.1. The original REPdenovo generates three pieces of the repeat, while the new version constructs the whole repeat. We also apply the new method on Hummingbird data and assemble 1,619 repeats that can be validated from PacBio long reads. Many of these repeats are likely to be novel (i.e. previously not present in RepBase). We note that long sequence reads (e.g. PacBio reads) may provide new data for repeat analysis.

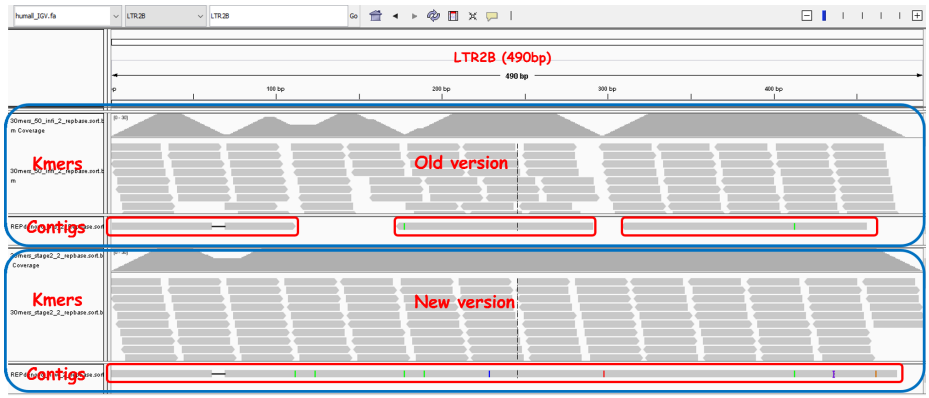

FIGURE 3.4.1: Comparison between the two versions of REPdenovo on constructing one sample repeat "LTR2B". The old version generates three pieces of the repeat, while the new version constructs the whole repeat.

We believe that our method can still be useful for repeat analysis especially for longer repeats even when long reads are available. For example, our method assembles 64 Hummingbird repeats that are longer than 2,000 bp, which can be difficult to analyze even with long reads.

The new method reports more repeats than the original REPdenovo. There are two main reasons for this increase. First, many repeats are of high divergence rate and many constructed contigs are just fragments of one repeat. As more repeat-related k-mers are used in assembly, many previously uncovered regions are constructed, although many are just fragments of the repeat. The other source of more repeats by the new method is that many repeats are just copies of the same repeat consensus. To evaluate how many constructed repeats are from the same repeat consensus, we design the following copy cluster algorithm: First, we check the pairwise similarity between each two repeats, and if the similarity is larger then threshold (by default 0.85), we view the two repeats are of the same group. Then a union find set algorithm is used to cluster the repeats. We apply the clustering on the 6,192 constructed repeats of

human individual NA19239, and 3,196 groups are reported. Therefore, the number of constructed repeats can be greatly reduced when related copies are removed.

Chapter 4

REPdenovo-MEI: De novo detection of mobile element insertions from sequence reads

4.1 Introduction

In human genome, most of the TEs are inactive now, except some like Alu, L1, SVA and HERV. We generally call these still active ones as mobile element insertions (MEIs). Recent studies show that MEIs are found active in several types of cancers (147; 144). Other studies show MEIs are active in neural system and possibly contribute to several types neural diseases (146; 143; 139). All these indicates it is still quite necessary to call out MEIs to help further understand the mechanisms behind.

The fast development of sequencing technologies, especially the development of second and third generation sequencing techniques, provide big opportunities and also challenges to study MEIs. With the amount of sequencing data generated, several

tools (144; 83; 84; 85; 93; 86; 92) have been developed to call ME insertions and deletions for individuals. MELT (86) is the tool used to call ME insertions in the 1000 genomes project (101; 102). MELT first collects all discordant pairs from a WGS alignment, and aligns them to provided MEI library using a sequencing aligner. It then goes across the reference genome and creates a list of putative MEIs based on total read support at each putative site. MELT will then merge the initial MEI calls across the individuals provided, and determine specific information and exact breakpoints for each of the putative MEIs. Finally, all sites are genotyped by a likelihood approach. Other tools use more or less the similar strategy, like RetroSeq (83) also collects the discordant reads and one mapped the other unmapped reads. For discordant reads it checks with a pre-annotated file to see whether one read of the pair is aligned in a repeat region. And for one mapped the mate unmapped reads, RetroSeq aligns the unmapped reads against a repeat library to check whether the unmapped read is originated from a repeat. TEMP (92) uses the exact same strategy as RetroSeq, and in addition it checks the number of clipped reads at the candidate breakpoints to improve the accuracy. T-lex (85; 93) also check the similarity between the unmapped reads and a ME annotation library to call the sites, besides that it uses the ME structure features like tail-A, TSDs to improve the accuracy. There are two main obvious drawbacks for all these tools: First, all these tools require a repeat consensus library (and a file of repeat annotation on the reference) to check whether the collected reads come from a repeat insertion or not. However, for most of the species there are no or only incomplete repeat libraries available, which constrains the widely usage of these tools. Second, even for some well studied species that have repeat libraries for using, as all these tools needs to check the reads similarity against the repeat consensuses, they will miss most of the high divergent ME insertions.

In this chapter, we propose a new approach called REPdenovo-MEI for calling mobile elements insertions with sequencing data. Comparing with existing tools, the major differences are:

- REPdenovo-MEI does not require any repeat libraries or genome annotation files to call MEIs. Instead of using a repeat library or genome annotation file to check whether reads comes from repeats, we collect high frequent k-mers, and check the reads against the high frequent k-mers to decide whether the reads come from repeats. In addition, considering the divergence rate of repeats, a clustering algorithm is designed to involve those low frequent k-mers, whose edit distances from high frequent k-mers are within the given threshold. In this way, our approach can tolerate more variations and as a result can call out more diverged mobile element insertions.

- REPdenovo-MEI provides an option to call out the inserted sequences from long reads. To the best of our knowledge, there is no other tool that can work with hybrid data. In addition, if no long reads provided, REPdenovo-MEI provides a two-stage local assembly approach that can construct the inserted copies accurately.

- A classification based approach is designed to call the genotypes of the MEIs. With more features collected and the advantages of machine learning approaches, the proposed approach can call out the genotypes accurately and efficiently.

- We show REPdenovo-MEI outperforms other tools on both simulated and real data.

4.2 Method

Depending on whether long reads are provided or not, there are different steps to call the MEIs. As shown in Figure 4.2.1, we first construct the high frequent k-mer library to replace the consensus repeat library and genome annotation files used by other tools. This step is the major difference for REPdenovo-MEI from other tools. Then, we call out the candidate sites from the short read alignments with the constructed high frequent k-mer library. With the collected reads for each candidate MEI, we design a two-stages local assembly step to construct the inserted sequences, which can be used to classify and further "validate" the called out candidate MEIs. If long reads are provided, users can skip the two-stages local assembly step and construct the inserted sequences by aligning the left and right flank sequences to the long reads. This usually can help to construct more completed inserted sequences. Then, with the "validated" called out MEIs, REPdenovo-MEI provides a classification based genotype calling step. In the following sections, we show the detailed procedures of each step.

4.2.1 High frequent k-mer library construction

A key step of REPdenov-MEI is to construct the high frequent k-mer library to replace the annotation files and consensus repeat libraries used in other tools. The main observation is that k-mers originated from repeats are usually of high frequency, especially for low divergent and high copy number repeats. Thus, with the given raw reads, we first do k-mer counting using tool KMC2 (114). Then, k-mers with frequency larger than "f" (by default 10) times the average coverage are parsed out as the initial high frequent k-mers. One major issue for only using these initial

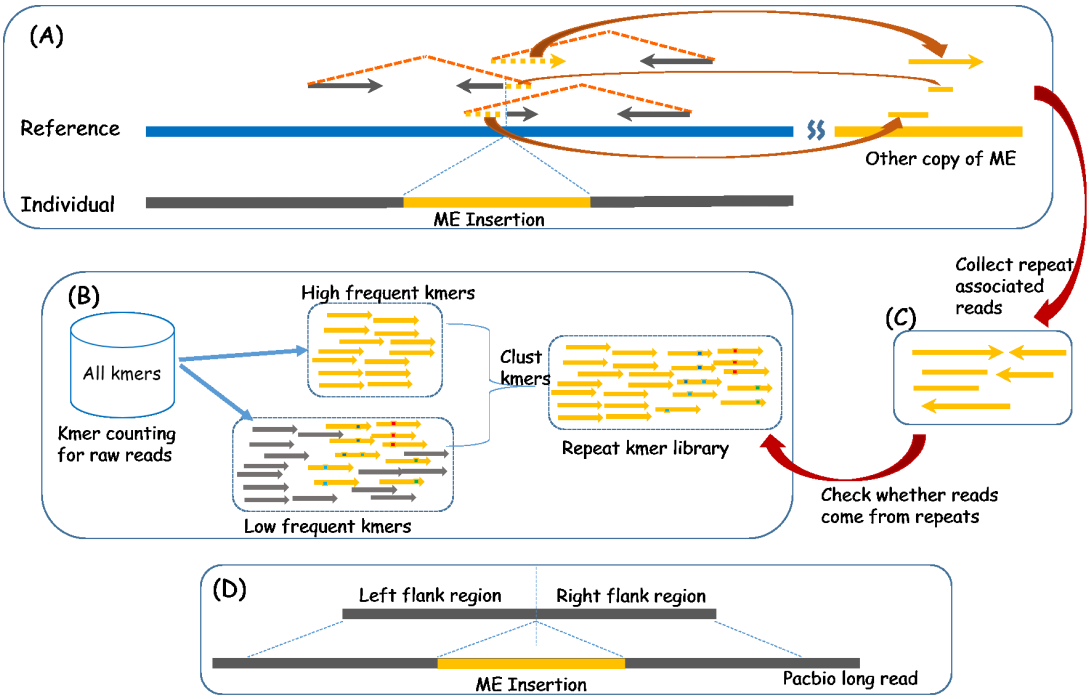

FIGURE 4.2.1: ***Pipeline of REPdenovo-MEI for calling MEIs:*** ***(A)*** *Collect repeat associated reads by parsing the alignment files. Clipped reads and discordant reads whose mate is uniquely mapped around the candidate sites are collected.* ***(B)*** *Construct the repeat kmer library. First do kmer counting for the raw reads, and isolate the high frequency k-mers. Then a cluster algorithm is designed to involve the low frequent k-mers that within the given edit distance from the high frequent k-mers.* ***(C)*** *Check the collected reads against the repeat kmer library to decide whether the reads are originated from the repeats.* ***(D)*** *Align the two side flank regions to the long reads to get out the sequences of the ME insertions. This is an optional step that only run when long reads are available.*

high frequent k-mers is that some repeats are of high divergent or with low copy number, and as a result k-mers originated from copies of these repeats will be of low frequency, thus are filtered out. We observe that even though these k-mers are of low frequent, they are usually within small edit distance from the high frequent k-mers. Thus, a randomized algorithm is designed to recruit these low frequent but originated from repeats k-mers. For two k-mers (with length k) are of high and low frequent respectively, we randomly pick h bases from the same positions of the two k-mers. If the chosen h-mers are the same, then the low frequent k-mer will be recruited. This procedure runs for n times to guarantee the low frequent k-mer is recruited. Here, we require at least one match between the two h-mers out of n times to recruit the low frequent k-mer. Given the values for n, k, h, the probability p of the low frequent k-mer being recruited is:

$$p = 1 - (1 - \frac{\binom{k-e}{h}}{\binom{k}{h}})^n$$

Here e is the allowed edit distance between the two k-mers. By default, the value of e is set to 1, that is, we allow one mismatch, insertion or deletion in one k-mer. This is reasonable because usually k is not large (less than 100).

The initially high frequent k-mers with the recruited low frequent k-mers together construct the final high frequent k-mer library, which will be used to call the candidate MEIs in the following step.

4.2.2 Calling candidate MEIs from short reads

As shown in Fig. 4.2.1 (A), when align the reads to the reference genome, if there is one MEI happen at one site, then three types of reads will be collected near this site: **(i) Discordant paired-end reads**: For two paired-end reads, if one read "A" of

the pair originates from the MEI, while its mate "B" doesn't, then when aligning this pair to the reference, read "B" will be mapped to the flank region, and read "A" will be mapped to other repeat copies, which is either in the same chromosome far from read "B" or in a different chromosome. Thus, they become discordant pair. To collect this pair of reads, we first require read "B" should be with high mapping quality (by default with mapping quality 60). Then we cut read "A"to overlap k-mers (by default concatenate k-mers shift by 1 base), and check each k-mer against the high frequent k-mer library generated in the previous step. If most (by default over 75%)of the k-mers find match in the high frequent k-mer library, then we view read "A" is originated from a repeat copy. In this way, we collect the discordant paired-end reads.

(ii) One mapped and its mate unmapped: For the paired-end reads mentioned in the type (i) section, if read "A" is originated from a MEI that has somatic mutations happen, and if the mutation rate is high, then this read with high probability will become unmapped. But, because our high frequent k-mer library is constructed by recruiting low frequent k-mers, k-mers from this unmapped read will still find hit in the k-mer library. Thus, if most (by default over 75%) of the k-mers from this unmapped reads find hit in the high frequent k-mer library, and its mate read is mapped with high mapping quality (by default 60), then we collect this one mapped and the mate unmapped read pair.

(iii) Clipped mapped reads: For one read, if part of the read comes from the MEI, while the other part is from the flank region, then when aligned to the reference, the read will become clipped-mapped. To distinguish from other clipped-mapped reads caused by other reasons, we check the clipped part against the high frequent k-mer library to see whether most (by default over 75%) of the k-mers can find hit in the

library. If most part of the clipped region find hits in the high frequent k-mer library, then we collect this clipped mapped read, otherwise drop it.

In the implementation, we first go through the alignment file, and collect all the clipped positions. And if some positions are only several (by default 10) bases away, then these sites will be merged to the one with the largest number of clipped reads. Then we keep the sites with number of supported clipped reads over given threshold (this is user-specific parameter with default value one third of the average coverage). For each of these parsed sites, we collect the first two types of reads, and if the sum of the number of these two types of reads is also larger than the same threshold, then this site will be viewed as a candidate MEI site. We view the MEI sites as independent from each other, and thus the calling step can be implemented in parallel. The first round of candidate sites callings is implemented in parallel chromosome by chromosome. And the second round is implemented in parallel site by site.

4.2.3 Constructing MEIs by local assembly

One general question for MEI analysis is to know what types are the MEIs or which families they belong to. One straightforward way to do this is constructing the sequence of each MEI and comparing with consensus repeats with known family information. We design a two stage local assembly approach to construct the MEIs with given collected reads from the previous step.

For the first stage, for each candidate MEI, we first use KMC2 (114) to convert the collected reads to k-mers, then we use Velvet (131) to assemble the k-mers to contigs. This step is similar to the repeat assembly approach developed in (50). For

the second stage, for each candiate MEI, we feed in the contigs generated in the first stage, and perform a contig merging step to get more complete sequence. Similar to the general genome assembly problem, contig merging can be performed based on prefix-suffix overlap between two contigs. We use the contig merging procedure in (50), which was originally developed for merging contigs for the repeat construction problem. Refer to (50) for more details on this procedure. For some regions of the MEIs, even though we have collected reads that fall into these regions, there may not be enough reads covering these regions. As a result, when we perform local assembly for each MEI, only short contigs (with little overlap with other contigs) are obtained for these regions. A simple solution is that we can view these reads as contigs and include them in the contigs merging step. To improve the merging efficiency and accuracy, we only use the reads whose mate are uniquely (by default with mapping quality 60) mapped.

If the users only focus on some specific type of repeat families and have consensus repeats of these repeats, then this step can also be used as a validation step. For each candidate MEI, we can blast the constructed sequence against the consensus sequences, and if we found significant hits, then we will be more confirm about the candidate site, and can directly called out as "validated" one.

4.2.4 Constructing and validating MEIs from long reads

Long reads are usually long enough to contain lots of types of repeats, which provides an option to construct more complete MEI sequences comparing to constructing by local assembly from short reads. With the called out candidate MEI sites, REPdenovo-MEI provides an option to construct the MEI sequences from long reads.

For each candidate MEI site, we first get the left and right flank regions (by default with length 300 bp) from the reference genome, and then we align these two flank sequences to the long reads with tools like Minimap (145). For one MEI site, if both the left and right flank sequences are fully mapped to one long read and these two flank sequences are aligned apart from each other, then with high probability the middle part between the flank regions on the long read can be viewed as the MEI sequence. If more than n (by default 3) long reads support the same MEI site, and the MEI sequences are similar to each other, then this site will be viewed as a "validated" site. As mentioned in the previous section, if users can provides consensus repeats, then REPdenovo-MEI can blast MEI sequences against these consensus sequences to report more "validated" MEI sequences.

4.2.5 Call genotype of the MEIs

REPdenovo-MEI also provides an option to call the genotypes of each MEI. We design a classification based approach to call the genotypes. The approach is based on the observation that the number of each type of collected reads are different for different genotypes. Similar approaches have been successfully used in our previous work(142). The current version is mainly developed for human which is diploid, but the approach can be extended to work for species with multiple haploids. So here, we use 0, 1, 2 to represent homozygous non-insertion, heterozygous insertion and homozygous insertion. In general, there are two main steps to call genotypes with given MEI sizes: First, we parse features for each site. The numbers of the three types reads mentioned in section 4.2.2 are quite different for different genotypes. Besides these three types of reads that indicate the existence of the MEIs, there is another

type of reads denotes the absence of the MEIs: **(iv) Fully mapped reads**: If the site is not a MEI site, reads will fully mapped at the site. So, for each MEI site, we collect these four features.

Second, we perform model training and genotype calling. All features are quantified using the sequence reads for the specific individual and the MEIs for all individuals. Let K be the number of features for a MEI. We obtain a length-K vector. We treat each such vector as a point in a K dimensional space. There are total NM points, where N is the number of diploid individuals and M is the number of deletions. Intuitively, points with the same genotype values tend to resemble each other. So, we expect to see three categories of points, which correspond to three possible genotypes: 0, 1 and 2. We use the support vector machine to perform classification. In particular, we use LibSVM (141) to train models using provided training data, and then use the trained SVM model to call genotypes of test data. The model training usually takes two steps. First, all training data is labeled and then scaled. When there is no validated data or labeled data provided, we can use simulated data to train the model. A simulator is released in our program. Second, grid search and 10-cross validation are used to find the optimal parameters, and default kernel function is used in this procedure. Then we use the trained model to call the genotypes of the test data. Also for training and testing, a "-b" option can be used in LibSVM to generate the probability for each genotype.

4.3 Results

To evaluate the performance of REPdenovo-MEI, we compare it against Melt and RetroSeq on both simulated and real data. To test on simulated data, we remove some known annotated TEs from the reference genome to construct a benchmarked one, then we align reads to this revised "reference genome", and thus these removed TEs will be converted to insertions. Then we compare the three tools on the number of called out "benchmarked" MEIs. To evaluate the performance of the three tools on real data, we use the high and low coverage short reads released by Illumina and the 1000 genomes project respectively. Then long reads are used to validate the called out MEIs. In addition, we also do further comparison of the three tools on the divergence of the called out "validated" MEIs. We show the detailed steps in the following three sections.

4.3.1 Comparison on simulated data

As human genome is well studied, and the reference genome is also of high quality. So first we compare REPdenov-MEI against Melt and RetroSeq on simulated human data. To generate the simulation data, we first need to construct a benchmarked "reference genome", and then we can align the reads to this revised "reference genome" to generated the simulated alignment data, which are the input of all the three tools. As human reference genome is well annotated, we can directly remove some annotated TEs from the reference genome to construct the revised one. Then we can test how many these removed one are called out by all these three tools to evaluate their performances. We use the annotation file on human reference GRCh38, which is generated by RepeatMasker. As it is known that L1 and Alu repeats are still active

in human genome, we pick two sub-family L1PA8 and AluJr to generate the revise reference genome. We parse out 4,163 copies of L1PA8 with length larger than 400 bp, and 94,910 copies of AluJr with length larger than 150 bp from the annotation file. One important thing is we must make sure the individual, from whom we sequence the reads, contains the repeat copies, thus when we move the copies from the reference genome, the remove copies will become insertions when align the reads to the revised "reference genome". To check whether the individual contains the repeat copies, we first align the reads of the individual to the original reference genome, and check whether there are reads clipped at the boundaries of each repeat copy, and if no clipped reads found, then we view the individual contains the repeat copy. Thus, finally 630 and 1,016 copies of L1PA8 and AluJr are validated in this way and randomly picked out respectively. So overall 1,646 repeat copies are manually removed to construct the benchmarked "reference genome". Human individual NA19239 is picked as the selected individual. High coverage (60X) short sequencing reads of NA19239 from Illumina are used to align to the benchmarked "reference genome". In addition, to evaluate the performance of different tools on low coverage data, we down-sample the 60X data to 15X to generate low coverage data. Table 4.3.1 shows the comparison of REPdenovo-MEI against Melt and RetroSeq on the high and low coverage data. It shows that REPdenovo-MEI called out more MEIs than Melt and RetroSeq on both high and low coverage data.

4.3.2 Comparison on real data

We compare REPdenovo-MEI with Melt and RetroSeq on three human individuals NA19238, NA19239 and NA19240 (forming a trio). Both high and low coverage data

TABLE 4.3.1: Compare REPdenovo-MEI against Melt and RetroSeq on simulated data. 1,646 repeat copies are removed from the human reference genome to be simulated as MEIs in the revised "reference genome". Two datasets of coverage 60X and 15X from NA19239 are aligned to this revised genome.

Data	Coverage	Melt	RetroSeq	REPdenovo-MEI short
NA19239 high	∼60X	1,132	900	1,364
NA19239 low	∼15X	754	839	1,098

of the three individuals are used, where the high coverage data are released in the Illumina platinum genomes, and the low coverage data are released in the 1000 genomes project. The coverage is around 60X and 10X for high and low coverage datasets respectively. For Melt and RetroSeq, annotation files and consensus repeats are needed. We use the annotation file generated by RepeatMasker on human reference GRCh38. And consensus repeats are parsed from Repbase. We use error corrected Pacbio long reads released in (140) to validate the called out MEIs. To do validation, for each candidate MEI, we first parse out the left and right flanks regions (by default with length 300 bp), then we align the two flank regions to long reads using Minimap, and if more than n (by default 3) long reads support the same MEI, then this MEI site survives for the next round validation. Then next, we parse out the n copies from the long reads, and run RepeatMasker for these copies. In this way, each copy from the long read will be labeled one (or more, or unmasked) repeat family, and if all the copies support the same repeat family, then we view the called out MEI is a "true" MEI.

TABLE 4.3.2: Comparison of REPdenovo-MEI with Melt and RetroSeq on real data. "called" indicates how many MEIs are reported by each tool. "validated" represents the number of MEIs are validated from Pacbio long reads. For REPdenovo-MEI, "short" represents the results are generated only from short reads, while "hybrid" means the results are generated with both short and long reads.

Cov.	Individual	Melt		RetroSeq		REPdenovo-MEI		
		called	validated	called	validated	short	hybrid	validated
High	NA19238	1,772	341	18,424	200	2,487	660	504
	NA19239	1,729	306	20,017	222	2,446	626	475
	NA19240	1,685	327	15,749	189	2,473	643	486
Low	NA19238	49	7	194	1	71	11	9
	NA19239	51	8	117	0	73	14	14
	NA19240	30	2	131	0	76	12	11

Table 4.3.2 shows the performance of REPdenovo-MEI, Melt, and RetroSeq on the high and low sequencing data. "called" indicates how many MEIs are reported by each tool. "validated" represents the number of MEIs are validated from Pacbio long reads. For REPdenovo-MEI, "short" represents the results are generated only from short reads, while "hybrid" means the results are generated with both short and long reads. On high coverage data, RetroSeq reports lots of false positives, while Melt and REPdenovo-MEI report much less. REPdenovo-MEI calls out much more "validated" MEIs than RetroSeq and Melt. With the filtering step from long reads, the "hybrid" REPdenovo-MEI reports much less false positives than the "short" one. The performance of REPdenovo-MEI, Melt and RetroSeq are all not well on low coverage data, although REPdenovo-MEI called out more "validated" MEIs than Melt and RetroSeq.

TABLE 4.3.3: Comparison of REPdenovo-MEI against Melt and RetroSeq on the number of called out MEIs of different divergent rates. On the NA19239 data, 306, 222, and 475 MEIs are validated for Melt, RetroSeq and REPdenovo-MEI respectively. "Divergent rate" is generated from the mismatch rate that reported by RepeatMasker when blast the repeat copies against the consensus.

Divergent rate	Melt	RetroSeq	REPdenovo-MEI
0.0-1.0	137	76	139
1.0-5.0	138	75	173
5.0-10.0	14	23	50
>10.0	17	48	113

4.3.3 Comparison on called out MEI divergence

On the NA19239 data, 306, 222, and 475 MEIs are validated for Melt, RetroSeq and REPdenovo-MEI respectively. We give further analysis on these validated MEIs of where the differences come from. For each "validate" MEI, we use the mismatch rate reported by RepeatMasker as their divergent rate. And thus, we can check the number of called out MEIs at different divergent rates. In table 4.3.3, we show the number of validated MEIs for REPdenovo-MEI, Melt and RetroSeq on different divergent rates. The results show that REPdenovo-MEI calls out much more higher diverged MEIs than Melt and RetroSeq.

4.4 Discussion and Conclusion

In this chapter, we propose a novel approach called REPdenovo-MEI for calling MEIs from sequencing data. Different from all the existing tools, REPdenovo-MEI does not rely on any annotation files or repeat libraries to call MEIs. In addition, REPdenovo-

MEI can work with both short and long reads, while to the best of our knowledge, no other tools can do the same work. REPdenovo-MEI provides a two-stage local assembly step to construct the MEI sequences when only short reads provided. It also provides a classification based genotype calling step to call genotypes of the called out MEIs. Experiments on both simulated and real data indicates REPdenovo-MEI outperforms Melt and RetroSeq on sensitivity and specificity, especially on calling high diverged MEIs.

The work is an ongoing project that many more experiments, such as experiments on genotype calling and validation, novel MEIs calling on other species, and evolution analysis on different human populations, will be done in the future.

Chapter 5

GAPPadder: A Sensitive Approach for Closing Gaps on Draft Genomes with Short Sequence Reads

5.1 Introduction

With the fast developing high-throughput sequencing technologies, de novo genome assembly from sequence reads has become a major application of sequencing technologies. So far many genome assembly software tools have been developed, including e.g. (131; 111; 128; 119). As sequence data from many species is becoming increasingly more available, draft genomes of many species have been assembled. Furthermore, more recent sequencing technologies such as long reads sequencing are expected to lead to even more assembled genomes with better quality than before.

Despite all these exciting developments, it is still challenging to obtain complete

genomes with the current technologies and assembly tools, especially at regions that are highly repetitive or have low coverage. At present, most assembled genomes contain gaps. For relatively complex genomes, only draft genomes which usually contain a large number of gaps are available. A more complete genome is highly desirable since it leads to better annotation, less genotyping error and easier identification of causal variation associated with traits (112) than a genome with many gaps.

With the development of the third generation sequencing technology, long reads from different platforms, like Pacific Biosciences, Illumina TruSeq, Oxford Nanopore, have been developed. With the help of these new technologies, the quality of the assembled draft genomes is greatly improved. In general, long reads are used in two ways to help to improve the draft genome assembly: 1) Long reads are used to scaffold the contigs and fill the gaps on the draft genomes assembled from high coverage short reads. 2) Long reads are directly used to assemble the draft genomes. Due to the high error rates of long reads, read depth is required to be high to guarantee the quality of genomes assembled directly from long reads, and thus sequencing cost can be high. In comparison, for scaffolding contigs and closing gaps with long reads, the coverage is usually not required to be very high. However, there are still gaps on the draft genomes even assembled with long reads, especially for draft genomes initially assembled high coverage short reads and then improved with long reads. Thus, it is still needed to close the gaps on draft genomes assembled with long reads. At present, short sequence reads are still the most available sequence reads. Thus, it is important to develop methods that can close gaps on draft genomes with short sequence reads that are readily available.

Several tools have been developed for closing gaps on draft genomes with short reads. GapCloser is a stand-alone tool in the SOAPdenovo (120) package. It performs

several iterations of base extension steps using the reads aligned to specific regions. GapFiller (110) implements a method that finds read pairs with one end aligned within a contig and its mate partially aligned to the draft genome and partially located in a region identified as a gap. These partially aligned reads are used to close the gap through sequence overlapping. Sealer (125) generates pseudo long reads from paired-end sequence reads by filling the unknown sequences between read pairs using the redundancy in sequence coverage, and then the pseudo long reads are used to fill the gaps. While these approaches have been used to close gaps in assembled genomes, these tools still cannot close many gaps (especially those originated in more complex genomic regions, e.g. repeats).

In this chapter, we develop a new approach called GAPPadder for closing gaps on draft genomes. Similar to tools such as GapCloser and GapFiller, GAPPadder also performs local assembly from reads that originate from gap regions. The following are the main features of GAPPadder and also differences between GAPPadder and the existing methods.

- GAPPadder uses more information about the gaps contained in sequence reads than existing methods. GAPPadder collects more reads relevant for gap closing, especially repeat-associated reads which are ignored by all the existing tools. Moreover, GAPPadder collects higher quality reads by utilizing more information with different insert sizes of pair-end (PM) and mate-pair (MP) reads.

- GAPPadder uses a different local assembly method for gap closing compared with existing methods. Existing methods often rely on local extension of contigs. GAPPadder, instead, performs a two-stage local assembly: it first assembles contigs in the gap and then generates higher quality local assembly of gap

sequences by merging contigs.

We compare GAPPadder with existing approaches using real sequence data from staphylococcus aureus, human chromosome 14 from GAGE (126), and whole genome sequencing data (with PE and MP reads) of one human individual NA12878 from Illumina. These genomes are assembled from short reads only. We show GAPPadder can close more gaps than GapCloser, GapFiller and Sealer with these short sequence reads. Besides these draft genomes assembled with only from short reads, we also compare GAPPadder with GapCloser on two draft genomes assembled with long reads: the bed bug draft genome assembled with hybrid short and long reads and the Asian sea bass draft genome directly assembled from long reads. We show many gaps can be fully closed and extended by GAPPadder and GapCloser, and GAPPadder closes much more than GapCloser on the hybrid assembled bed bug genome.

5.1.1 Gaps in draft genomes

De novo assembly of reads produces contigs. Contigs are then further linked with paired-end (PE) or mate-pair (MP) reads to form scaffolds. Scaffolds contain multiple gaps, whose lengths are estimated from the insert sizes of PE or MP reads. In general, extension of contigs stops at sites with repetitive regions, heterozygous alleles, sequencing errors or low read coverage (122; 130). Gaps can be mainly classified to three types. The most common type is the repeat-associated gap. Repeat is a piece of DNA which may have multiple copies in the genome. Note that these copies may differ slightly from each other. There are different types of repeats, including LINE, SINE, LTR elements, DNA transposon, satellites, etc. Repeat-associated gaps can be categorized to be satellite-associated, dispersed low divergent repeats-associated,

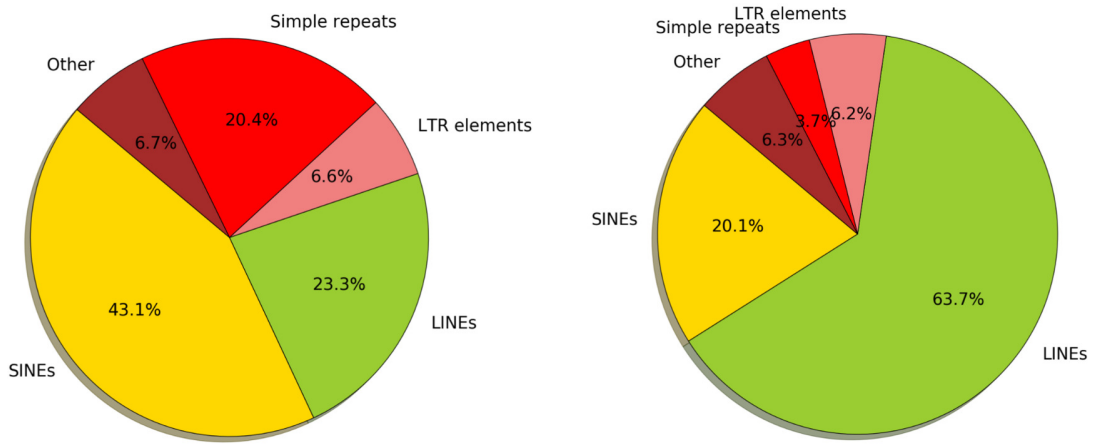

FIGURE 5.1.1: *Percentage of the masked gap sequences of each type of repeats. 3,934 gaps are extracted from the draft genome of human chromosome 14 that is released in GAGE. By aligning the flanking sequences to the human reference, we extract the gap sequences. We use RepeatMasker to get the types of repeats (e.g. LINE, SINE, LTR elements, etc) of these gap sequences. One gap may be masked to multiple repeat types. The left part shows the percentage of masked gaps for each repeat type. The right part shows the percentage of masked bases for each repeat type.*

and tandem repeats-associated. We show the results of masking the gap regions on chromosome 14 of human using RepeatMasker (129) in Fig. 5.1.1. To get the gap regions of the draft genome of chromosome 14, which is assembled by ALLPATH-LG and released in GAGE, we align the flanking regions of the reference genome, and thus get the benchmarked gap sequences (i.e. sequences from the reference genome that are missing in the draft genome). One can see that over 90% of the gaps are masked as repeat-associated gaps. Therefore, to develop gap closing methods, it can be very useful to consider the implications of repeats on gaps.

5.2 Methods

5.2.1 High level approach

In this chapter, we propose GAPPadder for closing gaps on draft genomes, which greatly improves the sensitivity. The key idea is that GAPPadder utilizes more information (i.e. relevant reads originated from the gaps) contained in the sequence reads for gap filling. For example, GAPPadder collects the repeat-associated reads, which are ignored by all existing approaches. Our main observation is that reads originated from repeat-associated gaps may be mapped to other copies of the same repeat contained in the genome. Therefore, the two ends of these read pairs may be discordantly mapped (i.e. mapping positions of the two ends are much farther away from each other than expected on the same chromosome or even located at different chromosomes). GAPPadder also uses multi-mapped reads near these reads because they may also be useful for the assembly of gap sequences, especially when the collected reads are of low coverage. GAPPadder utilizes the long insert size reads or mate-pair (MP) reads to collect high quality reads. Another important step in GAPPadder is that it performs two-stage local assembly for each gap: it first assembles contigs from relevant reads in the gap; then it merges these contigs to construct long gap sequences. The main observation is that assembled gap sequences usually are in the form of relatively short segments (contigs) due to positions with errors or variations. These contigs overlap but are usually not assembled by standard assembly methods into longer sequences due to mismatches between contigs. The merging step implemented in GAPPadder allows the merging of these contigs to form long (sometimes complete) gap sequences.

5.2.2 Relevant reads originated from gap regions

Similar to several existing methods, GAPPadder starts by finding relevant reads that originate within each gap. In this chapter, we are mainly concerned with paired-end (PE) or mate-paired (MP) reads. When aligning the reads back to the draft genome using tools e.g. BWA (118), four types of read pairs can be considered to originate from the gap regions. All these read pairs are located near the gap under consideration. This is shown in Figure 5.2.1.

(i) One end mapped and its mate unmapped. For a read pair, suppose the left (respectively right) read is aligned (by default with mapping quality greater than 30), and the alignment position is within $m + 3v$ distance from the left (respectively right) breakpoint of the gap. Then this end is called the anchored read. Here m and v are the mean and standard derivation of insert size respectively. Further suppose the mate of the anchored read is unmapped. Then the unmapped read comes from the gap region with high probability.

(ii) Discordant reads caused by repeats or duplicate segments. If one read of a pair comes from the gap region, then when aligning the read back to the draft genome, this read will be unmapped. However, if the gap region comes from a repeat region and there are other copies of the repeat that are already included in the draft genome, then this read may be aligned to another repeat copy. As a result, both ends of the pair will be mapped, but become discordant (with insert size outside the range $[m - 3v, m + 3v]$) or are mapped to different chromosomes. This kind of reads may originate within the gap and may help the assembly of gap sequences. Besides the discordant reads, multi-mapped reads (by default with mapping quality 0) near the discordant reads are also useful for assembly. This is because if the gap is repeat-

associated, these multi-mapped reads from the copy of the same repeat can be useful, especially when collected reads have low coverage.

(iii) Reads clipped at the breakpoints of the gaps. For the reads overlapping the breakpoints, parts of the reads will be aligned to the draft genome, and the other parts will be clipped. Clipped reads are useful to extend the assembled regions from collected reads to both sides of the flanking regions of gaps. This allows the assembled gap sequences to be positioned in the draft genome.

(iv) Both reads of a pair are unmapped reads. When the gap is long enough, then both ends of a pair likely originate within the gap region. As a result, when aligning reads back to the draft genome, both reads will be unmapped. Such unmapped reads may play an important role if the insert size is short and the gap is long. In this situation, it is difficult to find anchored reads. As a result, the middle part of the gap will not be filled using reads with anchor. We note that unmapped reads may be just due to reads errors and thus irrelevant for gap filling. The challenge is that we do not know which unmapped reads indeed originate from some gap, and if so, which gap they originate. We will explain how to address this problem in the following sections.

Most existing tools use only the type-iii reads, while GAPPadder uses all four types of reads.

5.2.3 Gap closing procedure

As shown in Fig. 5.2.1, there are five steps of GAPPadder. We process each gap in the draft genome independently. First, we collect the first three types of reads that may originate from a gap. Second, we perform local assembly of the collected reads of each gap. This generates (usually short) contigs that are segments of the gap

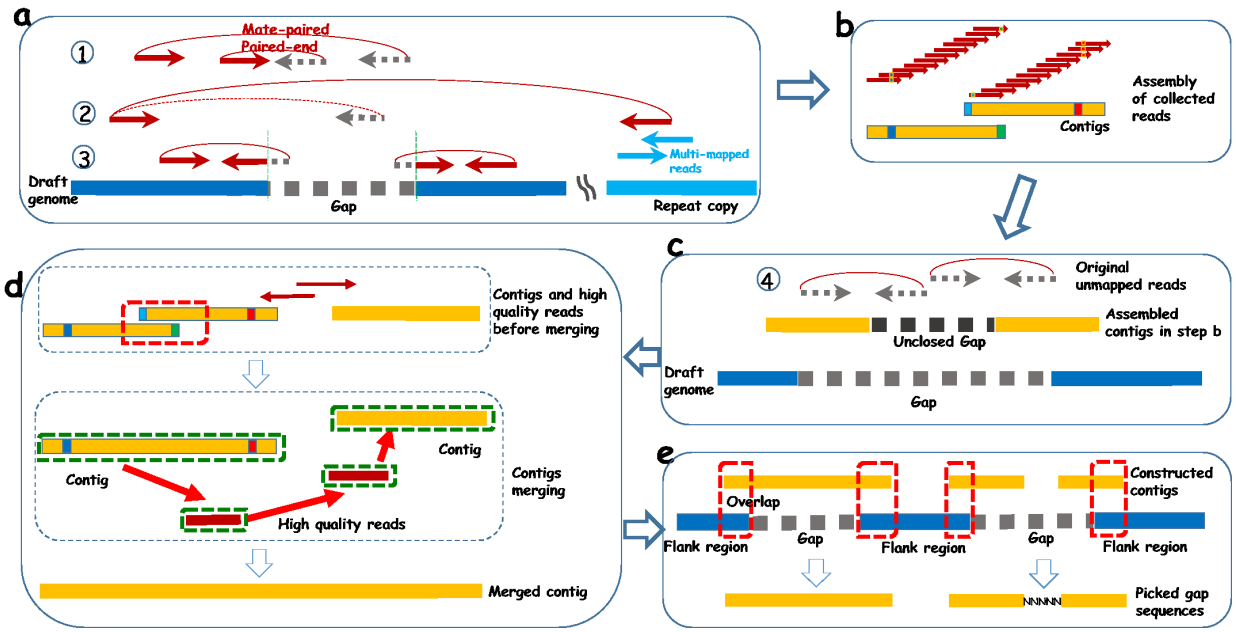

FIGURE 5.2.1: ***Pipeline of GAPPadder for closing gaps:*** ***(a)*** *Align reads back to the draft genome and collect the first three types of relevant reads originated from the gaps.* ***(b)*** *Perform local assembly with the collected reads, and merge contigs overlapping with each other to generate more complete gap sequences.* ***(c)*** *Align the unmapped reads to the constructed contigs of each gap, and collect the aligned (also their mate) reads.* ***(d)*** *Merge the contigs to form more complete assembly.* ***(e)*** *Gap sequences are obtained by aligning the merged contigs to the flanking regions of the gaps.*

sequences. Then, we align the unmapped reads to the constructed contigs and collect the aligned (also their mate) reads. We merge the contigs to form more complete assembly using a customized designed algorithm. Here, the high quality reads are treated as short contigs and is used for contig merging. Finally, we fill the gaps by aligning the merged contigs to the flanking regions of the gaps.

Collection of gap-associated reads

GAPPadder allows PE or MP reads of different insert sizes. For each group of reads of one specific insert size, we collect reads separately and then all these reads are used together for gap closing. To collect reads for one specific insert size, we first align the reads back to the draft genome using BWA.

We search for type-i reads that are mapped within $m + 3v + l$ (where l is the read length) distance from the breakpoints, and their mate reads are unmapped. The mapped reads are used as anchor, and the unmapped mate reads are used for gap closing. Here we consider all possible anchor reads, even when their mapping quality scores are low.

For type-ii reads, we search for reads in the region $[b_1 - m - 3v - l, b_2 + m + 3v + l]$, where b_1 and b_2 are the breakpoint positions of the gap. If a read A falls in this region but its mate read B is aligned outside the region, and also the mapping quality of read B is 0, then read B is considered to be type-ii. Also, suppose read B is aligned at position p, then we also use the reads whose mapping quality is 0 and aligned within the region $[p - d/2, p + d/2]$, where d is the gap length. This is because a read with mapping quality 0 is with high probability to be a multi-mapped read.

For type-iii reads, the assembly quality at the end of contigs is usually low. Thus,

when collecting reads clipped at breakpoints, we set some slack value (by default 20bp) to allow some distance between the clip position and the breakpoints of the gaps. Note that one read may satisfy the conditions of more than one gaps. And if this happens, we let the read to be used for all the related gaps.

Out of these collected reads, we define those reads whose mates (anchor reads) are uniquely mapped as high quality reads. Here, if the mapping quality of a read is equal to 60, then the read is considered to be uniquely mapped. In other words, we believe that with high probability these reads are from the specific gap region. We also collect the unmapped reads which will be used in the third step.

Local assembly of collected reads

This is the first stage of our two-stage local assembly approach. Once the reads are collected, we perform local assembly with the reads of each gap. KMC2 (114) is used to convert the reads to k-mers, then Velvet (131) is used to assemble the kmers to contigs. This step is similar to the repeat assembly approach developed in Chu *et al.* (50).

Collection of type-iv reads with the constructed contigs

From the previous steps, we construct contigs for each gap from the collected reads. If the insert size is shorter than the gap length, then both reads of a read pair may be unmapped. Such unmapped reads can be useful to construct longer contigs. This is important when there are paired-end and mate-pair reads of different insert sizes, but the coverage of mate-pair reads is low. Another case is when the mate-pair reads are initially used for scaffolding, but not for gap filling, and thus the coverage is usually

not high. In these situations, mate-pair reads can help to cover the regions where paired-end reads will never reach.

The challenge here is that we do not know which read pair comes from which gap, since they are unmapped. To solve this problem, we first collect all the unmapped reads. Then we align all the unmapped reads to the constructed contigs of each gap using BWA. By collecting the mapped reads, we collect the originally unmapped reads (now aligned to contigs of each gap) and their mate reads for each gap. Note that after the first-round assembly, we exclude those gaps that have been fully closed (see section 5.2.3 for details) from consideration. Then we only collect the unmapped reads for those not fully constructed.

Merging contigs

This is the second stage of the two-stage local assembly approach. The previous steps often generate more than one contigs for each gap. In order to obtain a complete gap sequence, GAPPadder performs a contig merging step. Similar to the general genome assembly problem, contig merging can be performed based on prefix-suffix overlap between two contigs. We use the contig merging procedure in Chu *et al.* (50), which was originally developed for merging contigs for the repeat construction problem. Refer to Chu *et al.* (50) for more details on this procedure. As mentioned in section 5.2.1, for some regions of gaps, even though we have collected reads that fall into these regions, there may not be enough reads covering these regions. As a result, when we perform local assembly for these gaps, only short contigs (with little overlap with other contigs) are obtained for these regions. A simple solution is that we can view these reads as contigs and include them in the contigs merging step. To

improve the merging efficiency and accuracy, we only use the reads with high quality scores that cannot be aligned to the constructed contigs.

Finishing gap sequence assembly

After contig merging, for each gap, there can be several constructed sequences. Most of these sequences are pieces of the repeats or wrongly assembled. So we need to identify the right one. We first check whether the whole gap is constructed. To identify the fully constructed ones, for each gap we get the two flanking sequences of the gap (by default 300bp for each). Then we align the two flanking sequences to the constructed contigs of the gap. If the left flanking sequence overlap with the left (right) side of the contig and the right (left) flanking sequence overlap with the right (left) side of the contig, and the two overlaps are of the same orientation (both are reverse complementary or both not), then we choose the contig as the gap sequence. If more than one contigs are found, we choose the longest one. In our experiments, we notice that for most of the filled gaps, there is usually only one satisfying these conditions. If complete gap sequences cannot be found, we choose the one that covers the gap the most.

5.3 Results

We compare GAPPadder with GapCloser, GapFiller and Sealor on datasets of three draft genomes of different sizes and with known reference sequences: staphylococcus aureus, human chromosome 14 and human whole genome. Data of staphylococcus aureus and human chromosome 14 are from GAGE (126). We choose the draft genome

assembled by ALLPATH-LG. For staphylococcus aureus, two groups of high coverage data of different insert sizes are used. While for the human chromosome 14, three groups of data of different insert sizes are used. The data with long jump library is of very low coverage. The human whole genome (NA12878) high-coverage PE and MP sequence reads are from Illumina. The draft genome of NA12878 is released in (115), which is assembled by ALLPATH-LG. Detailed information of the four datasets are given in the supplemental materials.

5.3.1 Comparison with existing tools

In Table 5.3.1, we show the results of GAPPadder and the other three tools on staphylococcus aureus, human chromosome 14 and human whole genome. Note that Sealer only runs well on short insert size data. For data with very long insert size, it can be extremely slow. So when running Sealer on the human whole genome data, we do not use the long insert size data. Detailed commands and parameters of running each tool are provided in the supplementary materials. The results show that GAPPadder outperforms the other three tools on the three datasets. For S. aureus and H. chromosome 14 datasets, GAPPadder closes more gaps than the other three tools. For the human whole genome datasets, GapCloser runs out of memory (on a server with 256G memory) and GapFiller did not finish after running for more than 725 hours. In comparison, GAPPadder and Sealer respectively close 130,371 and 110,876 gaps out of the 220,318 gaps.

To show the effect of the repeat-associated reads, we run a revised version of GAPPadder that does not use these repeat-associated reads for gap filling. This "streamlined" version of GAPPadder closes 1,103 gaps, much less than the original

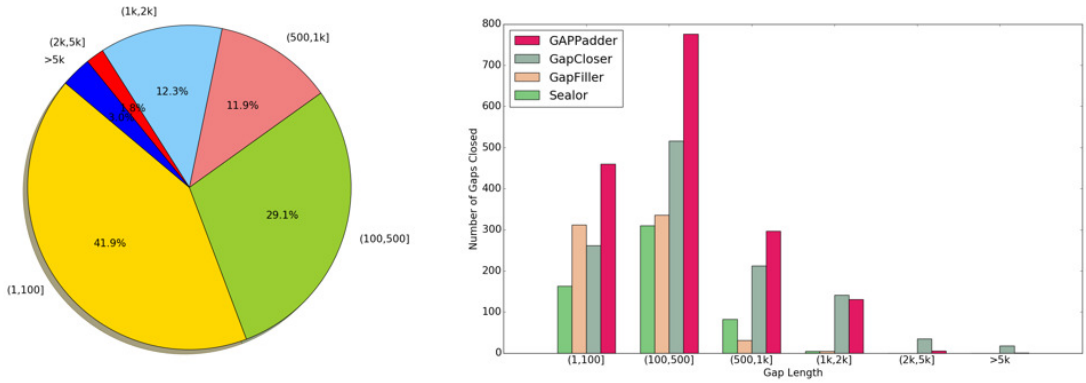

FIGURE 5.3.1: ***Comparison on different gap length:*** *The left part shows the distribution of gap length of all the 3,934 gaps on the draft genome of human chromosome 14. Over 80% of the gaps are smaller than 1 kb, and over 95% of the gaps are smaller than 2 kb. The right part shows the number of fully closed gaps of the four tools on different ranges of gap length.*

version of GAPPadder, which closes 1,670 gaps. This indicates that repeat-associated reads are indeed useful for gap closing.

In Fig. 5.3.1, we compare the four tools on different ranges of gap lengths of the closed gaps. The left part shows the distribution of gap length of all the 3,934 gaps on the draft genome of human chromosome 14. Over 80% of the gaps are shorter than 1k, and over 95% of the gaps are smaller than 2k. The right part shows the number of fully closed gaps of the four tools on different ranges of gap length. GAPPadder significantly outperforms the other three tools on gaps shorter than 1 kb, while GAPCloser performs slightly better on gaps longer than 1 kb.

5.3.2 Comparison on data with different insert sizes

For assembling draft genomes, usually data of different insert sizes are provided. Paired-end reads of long insert size or mate-paired reads can be helpful for closing (especially long) gaps on the draft genomes. Because of the different strategies used,

TABLE 5.3.1: Comparison of the four tools on three datasets: S. aureus, human chromosome 14 and whole human genome for NA12878, whose draft genomes have 23, 3,934, and 220,318 gaps respectively. Overall, GAPPadder closes more gaps than the other three tools on the these datasets.

Species	Gap Num.	Methods	Gaps fully closed
S. aureus	23	GAPPadder	9
		GapCloser	2
		GapFiller	1
		Sealer	2
H. chrom 14	3,934	GAPPadder	1,670
		GapCloser	1,184
		GapFiller	732
		Sealer	559
NA12878	220,318	GAPPadder	130,371
		GapCloser	Out of memory
		GapFiller	Not finished (After 725 hrs)
		Sealer	110,876

TABLE 5.3.2: Comparison of the four tools on different insert size data. Three groups of data of insert sizes (180, about 2,500 and about 35 kb) of human chromosome 14, and their combination are used for comparison. Results are given for reads with 180bp insert size only, and reads with 2,500bp insert size only and combined reads (with 180bp, 2,500bp and 35kb insert sizes).

Methods	Insert size		
	180	2,283 to 2,803	Combined
GAPPadder	862	1,481	1,670
GapCloser	1,142	216	1,184
GapFiller	484	173	732
Sealer	468	308	559

the performance of different tools differs significantly on datasets of different insert sizes. To evaluate the performance of the four tools on different insert size datasets, we compare the four tools on the human chromosome 14 datasets with only short insert size data, only long insert size data, and combined data with short and long insert size. The results are shown In Table 5.3.2. With data with short insert sizes, GapCloser performs the best. But with only long inset size data, GAPPadder significantly outperforms the other three tools. For comparison on the combined dataset with reads of both short and long insert sizes, GAPPadder performs the best.

5.3.3 Time and memory usage

All four tools are benchmarked on a 64-core server with AMD 6380 CPU @2.499 GHz and 256 GB RAM. To compare the time and memory usage of these four tools, we benchmark the four tools on the human chromosome 14 datasets. When running Sealer we set the maximum allowed memory to 40G, and other parameters are set

as suggested by its manual. For GapCloser, we use the default parameters. For GapFiller, the parameter for the number of iterations to run is set to be 5. See the supplementary materials for more detailed information of running the tools. In terms of running time, GapCloser, GapFiller, Sealer and GAPPadder take 30m 32s, 424m 47s, 160m 23s, and 85m 12s respectively. For memory usage, GapCloser takes 7.8G at the peak, Sealer takes 40G (as set in the parameter), while GapFiller and GAPPadder take less than 2G memory. Therefore, GapCloser is the most efficient one among the four tools, but it requires more memory. GAPPadder is slightly slower than GapCloser but uses much less memory.

5.3.4 Closing gaps on draft genomes assembled partially or fully with long reads

Although long reads help to improve the draft genome assembly, large number of gaps may still remain in the draft genome, especially for the draft genome originally assembled from short reads and then improved from long reads. To evaluate the performance of GAPPadder on two draft genomes that are partially and fully assembled with long reads, we run GAPPadder on two draft genomes: 1) The bed bug cimex lectularius draft genome (released in (69)) which is assembled with hybrid data of both short and long reads, 2) Asian sea bass draft genome (released in (68)) that is purely assembled from high coverage PacBio long reads. The bed bug genome is initially assembled with 73x coverage Illumina short reads using ALLPATHS-LG (111) assembler. And then Illumina Moleculo kit is used to sequence long reads with average length 3,500bp, which is used to improve the initial assembled draft genome. However, even for the improved draft genome, there are still many gaps. In the fi-

nal released assembled genome, there are 118,821 gaps, out of which 97,251 gaps are larger than 100bp. We run GAPPadder and GapCloser to close the gaps. As some gaps are really small (just several bases), and to evaluate the power of different tools we only focus on these 97,251 gaps that are larger than 100bp. Three sets of Illumina short reads with insert sizes of 185, 367, and 3,000bp and coverage of 34x, 12x, and 7x respectively are used for gap closing. GAPPadder reports 19,476 gaps are fully closed and 52,879 gaps are partially extended, while GapCloser reports 3,299 and 2,417 are fully closed and partially extended respectively. To validate the fully closed and partially extended gaps, for each closed gap sequence, we extract the left and right flank regions of length 150bp each, and concatenate them with the gap sequence. Then we align the reads back to the concatenated sequences and check whether there are reads clipped at the joint regions. If enough (by default 10) reads are fully mapped at the joint regions and over 95% of the bases (for extended ones, excluding the not-filled regions) of each sequence at least have 10 reads covered, then we call this gap is validated. Otherwise it is not. For GAPPadder, 14,925 fully closed gaps and 37,802 partially extended gaps are validated in this way. While for GapCloser, 2,737 fully closed and 20 partially extended gaps are validated. In table 5.3.3 we show the comparison.

For the Asian sea bass draft genome, it is primarily assembled from 90x PacBio data and then scaffolded using transcriptome data. From the release draft genome, 110 gaps are extracted and all of them are larger than 100bp. We run GAPPadder and GapCloser to close the gaps. Two sets of Illumina short paired end reads with insert sizes of 500bp and 750bp, read length 100bp, and total coverage 80x are used for closing the gaps on the draft genome. For GAPPadder, 6 and 47 gaps are reported to be fully closed and partially extended respectively. We use the same validation

TABLE 5.3.3: Evaluation of GAPPadder and GapCloser on closing gaps for bed bug draft genome. The draft genome is initially assembled with high coverage short reads, and then improved with long reads. GAPPadder fully closes 14,925 out of reported 19,476 gaps and extends 37,802 out of reported 52,879 gaps. As a comparison, GapCloser fully closes 2,737 (3,299 are reported) gaps and extends 20 (2,417 are reported) gaps.

Category		GAPPadder	GapCloser
Fully closed	Reported	19,476	3,299
	Validated	14,925	2,737
Extended	Reported	52,879	2,417
	Validated	37,802	20

approach as used in validating the gap sequences of the bed bug genome, and 3 fully closed and 13 partially extended gaps are validated in this way. For GapCloser, 46 and 41 are reported to be fully closed and partially extended respectively, and 6 and 1 out of the fully closed and partially extended gaps are validated by the the same way.

5.4 Discussion and Conclusions

In this chapter, we propose a sensitive approach for closing gaps on draft genomes with paired-end reads and mate-paired reads. Empirical results show that when both short and long insert size data are provided, our tool GAPPadder outperforms GapCloser, GapFiller and Sealer. This is likely due to the fact that GAPPadder uses more reads (especially the repeat-associated reads) to close the gaps which are ignored by all other tools. Besides that, GAPPadder takes advantage of long insert size data and performs a two-stage local assembly approach to construct more complete gap

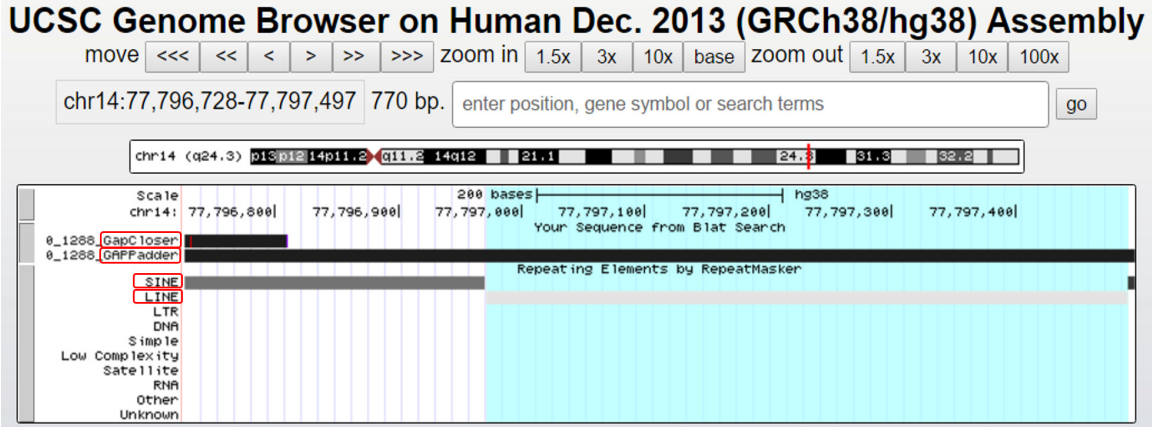

FIGURE 5.4.1: *One example gap closed by GAPPadder but failed by other three tools: one gap on the chromosome 14 draft genome is fully closed by GAPPadder, but other three tools fail to fully close it. The gap is about 770bp long, and from the UCSC genome browser we can see it is composed by part of a SINE copy and part of a LINE copy. GapCloser only extends by a short length on the left part. GapFiller and Sealer fail to fill this gap.*

sequences. In Figure 5.4.1, we show the comparison of the four tools on closing one example gap, which is about 770bp long on chromosome 14. GapCloser only extends a little on the left part. GapFiller and Sealer even have no extension at all, and thus are not shown in the UCSC Genome Browser. In comparison, GAPPadder fully closes the gap. One possible reason is the gap is composed by part of a SINE copy and part of a LINE copy as shown in the UCSC genome browser. The repeat-associated reads used by GAPPadder provide enough coverage for assembling the gap region.

However, when only using reads with short insert size for closing the gaps on human chromosome 14, GAPPadder does not performs as well as GapCloser. One reason is that GAPPadder relies on the contigs constructed in the first step to collect both unmapped reads. If the insert size is small, then the collected reads mainly come from the two ends of the gap, and thus the middle part will be difficult to construct in the following steps. In comparison, GapCloser uses an iterative strategy which can

gradually extend the contigs. This indicates that tools are designed with different strategies, and users should choose a tool based on the kind of data.

For draft genomes directly assembled from high coverage long reads, often the draft genomes contain far less gaps than those assembled from short reads. One reason is for less complex genomes, chromosome level contigs are directly assembled which do not need to do scaffold and gap closing. Second, very little scaffolding tools are developed for these near completed draft genomes, thus even gaps exists, they are not reported in the released draft genomes. Nonetheless, we observe that there can still be gaps within draft genomes that are directly assembled from long reads. Our results indicate that our GAPPadder tool can still be useful in the age of long reads genome assembly.

One possible future research on gap filling is incorporating long reads to close the gaps on the draft genome. Direct assembly of long reads usually requires the coverage should be high enough to get a high quality draft genome, which usually leads to high sequencing cost. Although low coverage of long reads cannot provides a high quality draft genome, it may help to close the gaps on the draft genome generated from short reads, especially for the long duplicate-associated or repeat-associated gaps.

Chapter 6

Conclusion

In this thesis, we propose four methods to address three computational questions related to genome repeats: 1) How do we construct de novo repeats with short sequencing reads? 2) How to detect and construct mobile element insertions with short and long sequencing reads? 3) How can we better close the gaps on draft genomes with short sequencing reads, especially those gaps caused by repeats?

To address the first question, we propose a method called REPdenovo to construct de novo repeat motifs directly from short reads. REPdenovo constructs consensus repeats by assembling high frequent k-mers. And then a contigs merging step is designed to merge fragments to complete repeats. We compare REPdenovo with RepARK on Human data. Results show that REPdenovo outperforms RepARK not only on the number of constructed repeats, but also on the completeness of the constructed repeats. In addition, we also construct 190 potentially novel repeats that do not exist in human reference genome, but find full hits in Pacbio long reads and NCBI nucleotide database. Further, we propose an improved version of REPdenovo. Comparing to

the original version, the improved version has two main advantages: First, it recruits more low frequent k-mers by an alignment based approach, benefit from which more low copy number repeats are constructed. Second, it performs a random algorithm to polish k-mers, as a results of which more high divergent repeats are able to be assembled. Experiments on both Human and Arabidopsis data show that the improved version construct more higher divergent and low copy number repeats than the original version. We run the improved REPdenovo on Hummingbird data and construct 1,617 repeats that can be validated from Pacbio long reads. Overall, our proposed approaches not only can help to construct consensus repeats for complex genomes which do not have available high quality genomes, but also can help to find novel repeats from high quality genomes like human reference genome.

To solve the second problem, we propose a novel method called REPdenovo-MEI to call mobile element insertions (MEIs) from both short and long reads. Comparing with existing methods, REPdenovo-MEI does not rely on any annotation files or consensus repeats to call MEIs. Instead, it constructs a high frequent k-mer library to check whether a read is from a repeat or not. REPdenovo-MEI also has a two-stage local assembly approach to construct the inserted sequences. And if long reads are provided, instead of the local assembly step, REPdenovo-MEI provides an option to call out the inserted sequences directly from long reads. In addition, REPdenovo-MEI provides a classification based genotype calling step to call genotypes of the called out MEIs. Experiments on both simulated and real data show that comparing to existing tools like Melt and RetroSeq, REPdenovo-MEI is able to call out more MEIs accurately and efficiently, especially those high diverged ones.

To address the third question, a novel method (called GAPPadder) is proposed to closing gaps on draft genomes from short sequencing reads. Comparing to exist-

ing tools like GapCloser, GapFiller and Sealer, GAPPadder collects more reads to construct the gap sequences, especially those repeats associated reads. This is based on the observation that most of the gaps on assembled genomes are caused by repeats. While all of the existing tools ignore these reads, GAPPadder collects them to improve the gap sequence assembly. Besides, GAPPadder better utilizes datasets of different insert sizes. In addition, GAPPadder performs a two-stage local assembly to assemble more complete gap sequences. Experiments on one bacterial genome, Human chromosome 14 and Human whole genome show that GAPPadder outperforms GapCloser, GapFiller and Sealer on both accuracy and efficiency. This is true not only on genomes assembled from short reads, but also on genomes assembled with long reads. We run GAPPadder on one bed bug genome that assembled from short reads and then improved from long reads, and 14,925 (out of 97,251) gaps are fully closed. Genomes assembled purely from high coverage long reads usually have much less gaps. But our experiments on the Asian sea bass genome show that still some gaps are able to be closed by GAPPadder.

Even though the proposed three computational questions are solved by the methods we designed, new questions emerge with the development of new technologies. For example, it is known that long reads provides opportunities to construct more complete repeats. However, with the high error rates, how to efficiently and accurately call out the repeats from the long reads? As many works have shown that linked reads sequenced by 10X Genome provide opportunities to construct higher quality draft genomes with low cost. Are there any differences to closing the gaps for the draft genomes assembled from linked reads? These will be the direction of our future works.

Bibliography

[1] Batzer MA, Deininger PL. Alu repeats and human genomic diversity. Nature Review Genetics. 2002;3:370–379.

[2] Kazazian, Haig H. Mobile Elements: Drivers of Genome Evolution. Science. 2004;303:1626–1632.

[3] Cordaux R, Batzer MA. The impact of retrotransposons on human genome evolution. Nature Review Genetics. 2009;10:691–703.

[4] SanMiguel P, Tikhonov A, Y K Jin ea. Nested retrotransposons in the intergenic regions of the maize genome. Science. 1996;274:765–768.

[5] Kazazian HH, Moran JV. The impact of L1 retrotransposons on the human genome. Nat Genet. 1998;19:19–24.

[6] Smit AF, Hubley R, Green P. RepeatMasker Open-4.0; 2013-2015.

[7] Witherspoon DJ, Zhang Y, Xing J, Watkins WS, Ha H, Batzer MA, et al. Mobile element scanning (ME-Scan) identifies thousands of novel Alu insertions in diverse human populations. Genome research. 2013;23(7):1170–1181.

[8] Nakagome M, Solovieva E, Takahashi A, Yasue H, Hirochika H, Miyao A. Transposon Insertion Finder (TIF): a novel program for detection of de novo transpositions of transposable elements. BMC bioinformatics. 2014;15(1):71.

[9] Keane TM, Wong K, Adams DJ. RetroSeq: transposable element discovery from next-generation sequencing data. Bioinformatics. 2013;29(3):389–390.

[10] Fiston-Lavier AS, Carrigan M, Petrov DA, Gonzalez J. T-lex: a program for fast and accurate assessment of transposable element presence using next-generation sequencing data. Nucleic acids research. 2011;39(6):e36–e36.

[11] Jurka J, Kapitonov VV, Pavlicek A, Klonowski P, Kohany O, Walichiewicz J. Repbase Update, a database of eukaryotic repetitive elements. Cytogenetic and genome research. 2005;110(1-4):462–467.

[12] Wheeler TJ, Clements J, Eddy SR, Hubley R, Jones TA, Jurka J, et al. Dfam: a database of repetitive DNA based on profile hidden Markov models. Nucleic acids research. 2013;41(D1):D70–D82.

[13] Koch P, Platzer M, Downie BR. RepARK - de novo creation of repeat libraries from whole-genome NGS reads. Nucleic acids research. 2014;42:e80.

[14] Zhuang J, Wang J, Theurkauf W, Weng Z. TEMP: a computational method for analyzing transposable element polymorphism in populations. Nucleic acids research. 2014;42(11):6826–6838.

[15] Marçais G, Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics. 2011;27(6):764–770.

[16] Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome research. 2008;18(5):821–829.

[17] Myers EW. Towards simplifying and accurately formulating fragment assembly. J of Comp Biology. 1995;2:275–290.

[18] Durbin R, Eddy S, Krogh A, Mitchison G. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge, U.K.: Cambridge University Press; 1999.

[19] Cormen T, Leiserson C, Rivest R. Introduction to Algorithms. Cambridge, MA.: MIT press and McGraw Hill; 1992.

[20] Li H, Durbin R. Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics. 2009;25(14):1754–1760.

[21] 1000 Genomes Project Consortium. An integrated map of genetic variation from 1092 human genomes. Nature. 2012;491:56–65.

[22] Rosenbloom KR, Armstrong J, Barber GP, Casper J, Clawson H, Diekhans M, et al. The UCSC Genome Browser database: 2015 update. Nucleic Acids Research. 2015;43:D670–681.

[23] Berlin K, Koren S, Chin CS, Drake JP, Landolin JM, Phillippy AM. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nature Biotechnology. 2015;33:623–630.

[24] Chaisson MJ, Huddleston J, Dennis MY, Sudmant PH, Malig M, Hormozdiari F, et al. Resolving the complexity of the human genome using single-molecule sequencing. Nature. 2015;517(7536):608–611.

[25] Robinson JT, Thorvaldsdóttir H, Winckler W, Guttman M, E S Lander ea. Integrative genomics viewer. Nature biotechnology. 2011;29:24–26.

[26] Chin, C.S., Peluso, P., Sedlazeck, F.J., Nattestad, M., Concepcion, G.T., Clum, A., Dunn, C., O'Malley, R., Figueroa-Balderas, R., Morales-Cruz, A., et al.: Phased diploid genome assembly with single-molecule real-time sequencing. Nature Methods 13(12), 1050–1054 (2016)

[27] Chu, C., Nielsen, R., Wu, Y.: Repdenovo: Inferring de novo repeat motifs from short sequence reads. PloS one 11(3), e0150719 (2016)

[28] Consortium, .G.P., et al.: An integrated map of genetic variation from 1,092 human genomes. Nature 491(7422), 56–65 (2012)

[29] Cordaux, R., Batzer, M.A.: The impact of retrotransposons on human genome evolution. Nature Review Genetics 10, 691–703 (2009)

[30] Deorowicz, S., Kokot, M., Grabowski, S., Debudaj-Grabysz, A.: Kmc 2: Fast and resource-frugal k-mer counting. Bioinformatics 31(10), 1569–1576 (2015)

[31] Edgar, R.C., Myers, E.W.: Piler: identification and classification of genomic repeats. Bioinformatics 21(suppl 1), i152–i158 (2005)

[32] Jr., H.H.K.: Mobile elements: Drivers of genome evolution. Science 303, 1626–1632 (2004)

[33] Jurka, J., Kapitonov, V.V., Pavlicek, A., Klonowski, P., Kohany, O., Walichiewicz, J.: Repbase Update, a database of eukaryotic repetitive elements. Cytogenetic and genome research 110(1-4), 462–467 (2005)

[34] Koch, P., Platzer, M., Downie, B.R.: RepARK - de novo creation of repeat libraries from whole-genome NGS reads. Nucleic acids research 42, e80 (2014)

[35] Ye, N., Zhang, X., Miao, M., Fan, X., Zheng, Y., Xu, D., Wang, J., Zhou, L., Wang, D., Gao, Y., et al.: Saccharina genomes provide novel insight into kelp biology. Nature communications 6 (2015)

[36] Korlach, J., Gedman, G., Kingan, S., Chin, J., Howard, J., Cantin, L., Jarvis, E.D.: De novo pacbio long-read and phased avian genome assemblies correct and add to genes important in neuroscience research. bioRxiv p. 103911 (2017)

[37] Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics 25(14), 1754–1760 (2009)

[38] Mills, R.E., Bennett, E.A., Iskow, R.C., Devine, S.E.: Which transposable elements are active in the human genome? Trends in genetics 23(4), 183–191 (2007)

[39] Price, A.L., Jones, N.C., Pevzner, P.A.: De novo identification of repeat families in large genomes. Bioinformatics 21(suppl 1), i351–i358 (2005)

[40] Robinson, J., Thorvaldsdóttir, H., Winckler, W., Guttman, M., E.S. Lander, e.a.: Integrative genomics viewer. Nature biotechnology 29, 24–26 (2011)

[41] Rosenbloom, K.R., Armstrong, J., Barber, G.P., Casper, J., Clawson, H., Diekhans, M., Dreszer, T.R., Fujita, P.A., Guruvadoo, L., Haeussler, M., et al.: The ucsc genome browser database: 2015 update. Nucleic acids research 43(D1), D670–D681 (2015)

[42] Schaeffer, C.E., Figueroa, N.D., Liu, X., Karro, J.E.: phraider: Pattern-hunter based rapid ab initio detection of elementary repeats. Bioinformatics 32(12), i209–i215 (2016)

[43] Smit, A., Hubley, R., Green, P.: Repeatmasker open-4.0. 2013–2015. Institute for Systems Biology. http://repeatmasker. org (2015)

[44] Wheeler, T.J., Clements, J., Eddy, S.R., Hubley, R., Jones, T.A., Jurka, J., Smit, A.F., Finn, R.D.: Dfam: a database of repetitive dna based on profile hidden markov models. Nucleic acids research 41(D1), D70–D82 (2013)

[45] Zerbino, D.R., Birney, E.: Velvet: algorithms for de novo short read assembly using de bruijn graphs. Genome research 18(5), 821–829 (2008)

[140] Berlin, K., Koren, S., Chin, C.-S., Drake, J. P., Landolin, J. M., and Phillippy, A. M. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, **33**(6), 623–630.

[110] Boetzer, M. and Pirovano, W. (2012). Toward almost closed genomes with gapfiller. *Genome biology*, **13**(6), 1.

[111] Butler, J., MacCallum, I., Kleber, M., Shlyakhter, I. A., Belmonte, M. K., Lander, E. S., Nusbaum, C., and Jaffe, D. B. (2008). Allpaths: de novo assembly of whole-genome shotgun microreads. *Genome research*, **18**(5), 810–820.

[112] Chaisson, M. J., Wilson, R. K., and Eichler, E. E. (2015). Genetic variation and the de novo assembly of human genomes. *Nature Reviews Genetics*.

[50] Chu, C., Nielsen, R. and Wu, Y. (2016). REPdenovo: Inferring De Novo Repeat Motifs from Short Sequence Reads. *PloS One*, **11**(3), e0150719.

[114] Deorowicz, S., Kokot, M., Grabowski, S., and Debudaj-Grabysz, A. (2015). Kmc 2: Fast and resource-frugal k-mer counting. *Bioinformatics*, **31**(10), 1569–1576.

[115] Gnerre, S., MacCallum, I., Przybylski, D., Ribeiro, F. J., Burton, J. N., Walker, B. J., Sharpe, T., Hall, G., Shea, T. P., Sykes, S., *et al.* (2011). High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, **108**(4), 1513–1518.

[116] Gordon, D., Huddleston, J., Chaisson, M. J., Hill, C. M., Kronenberg, Z. N., Munson, K. M., Malig, M., Raja, A., Fiddes, I., Hillier, L. W., *et al.* (2016). Long-read sequence assembly of the gorilla genome. *Science*, **352**(6281), aae0344.

[117] Kosugi, S., Hirakawa, H., and Tabata, S. (2015). Gmcloser: closing gaps in assemblies accurately with a likelihood-based selection of contig or long-read alignments. *Bioinformatics*, **31**(23), 3733–3741.

[118] Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, **25**(14), 1754–1760.

[119] Li, R., Zhu, H., Ruan, J., Qian, W., Fang, X., Shi, Z., Li, Y., Li, S., Shan, G., Kristiansen, K., *et al.* (2010). De novo assembly of human genomes with massively parallel short read sequencing. *Genome research*, **20**(2), 265–272.

[120] Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., Liu, Y., *et al.* (2012). Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, **1**(1), 1.

[121] Manolio, T. A., Collins, F. S., Cox, N. J., Goldstein, D. B., Hindorff, L. A., Hunter, D. J., McCarthy, M. I., Ramos, E. M., Cardon, L. R., Chakravarti, A., *et al.* (2009). Finding the missing heritability of complex diseases. *Nature*, **461**(7265), 747–753.

[122] Miller, J. R., Koren, S., and Sutton, G. (2010). Assembly algorithms for next-generation sequencing data. *Genomics*, **95**(6), 315–327.

[123] Myers, E. W. (1995). Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, **2**(2), 275–290.

[125] Paulino, D., Warren, R. L., Vandervalk, B. P., Raymond, A., Jackman, S. D., and Birol, I. (2015). Sealer: a scalable gap-closing application for finishing draft genomes. *BMC bioinformatics*, **16**(1), 230.

[126] Salzberg, S. L., Phillippy, A. M., Zimin, A., Puiu, D., Magoc, T., Koren, S., Treangen, T. J., Schatz, M. C., Delcher, A. L., Roberts, M., *et al.* (2012). Gage: A critical evaluation of genome assemblies and assembly algorithms. *Genome research*, **22**(3), 557–567.

[128] Simpson, J. T., Wong, K., Jackman, S. D., Schein, J. E., Jones, S. J., and Birol, I. (2009). Abyss: a parallel assembler for short read sequence data. *Genome research*, **19**(6), 1117–1123.

[129] Smit, A. F., Hubley, R., and Green, P. (1996). Repeatmasker open-3.0.

[130] Treangen, T. J. and Salzberg, S. L. (2012). Repetitive dna and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics*, **13**(1), 36–46.

[66] Zhang, G., Li, C., Li, Q., Li, B., Larkin, D. M., Lee, C., Storz, J.F., Antunes, A., Greenwold, M. J., Meredith, R. W., et al. Comparative genomics reveals insights into avian genome evolution and adaptation. *Science*, 346(6215):1311–1320, 2014.

[131] Zerbino, D. R. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, **18**(5), 821–829.

[68] Jeffrey A Rosenfeld, Darryl Reeves, Mercer R Brugler, Apurva Narechania, Sabrina Simon, Russell Durrett, Jonathan Foox, Kevin Shianna, Michael C Schatz, Jorge Gandara, *et al.* Genome assembly and geospatial phylogenomics of the bed bug cimex lectularius. *Nature communications*, 7, 2016.

[69] Shubha Vij, Heiner Kuhl, Inna S Kuznetsova, Aleksey Komissarov, Andrey A Yurchenko, Peter Van Heusden, Siddharth Singh, Natascha M Thevasagayam, Sai Rama Sridatta Prakki, Kathiresan Purushothaman, *et al.* Chromosomal-level assembly of the asian seabass genome using long sequence reads and multi-layered scaffolding. *PLoS Genet*, 12(4):e1005954, 2016.

[70] Reyad A Elbarbary, Bronwyn A Lucas, and Lynne E Maquat. Retrotransposons as regulators of gene expression. *Science*, 351(6274):aac7247, 2016.

[71] Eunjung Lee, Rebecca Iskow, Lixing Yang, Omer Gokcumen, Psalm Haseley, Lovelace J Luquette, Jens G Lohr, Christopher C Harris, Li Ding, Richard K Wilson, et al. Landscape of somatic retrotransposition in human cancers. *Science*, 337(6097):967–971, 2012.

[72] Zuojian Tang, Jared P Steranka, Sisi Ma, Mark Grivainis, Nemanja Rodić, Cheng Ran Lisa Huang, Ie-Ming Shih, Tian-Li Wang, Jef D Boeke, David Fenyö, et al.

Human transposon insertion profiling: Analysis, visualization and identification of somatic line-1 insertions in ovarian cancer. *Proceedings of the National Academy of Sciences*, page 201619797, 2017.

[73] Eunjung Lee, Rebecca Iskow, Lixing Yang, Omer Gokcumen, Psalm Haseley, Lovelace J Luquette, Jens G Lohr, Christopher C Harris, Li Ding, Richard K Wilson, et al. Landscape of somatic retrotransposition in human cancers. *Science*, 337(6097):967–971, 2012.

[74] Gilad D Evrony, Eunjung Lee, Bhaven K Mehta, Yuval Benjamini, Robert M Johnson, Xuyu Cai, Lixing Yang, Psalm Haseley, Hillel S Lehmann, Peter J Park, et al. Cell lineage analysis in human brain using endogenous retroelements. *Neuron*, 85(1):49–59, 2015.

[75] Batzer MA, Deininger PL. Alu repeats and human genomic diversity. Nature Review Genetics. 2002;3:370–379.

[76] Kazazian, Haig H. Mobile Elements: Drivers of Genome Evolution. Science. 2004;303:1626–1632.

[77] Cordaux R, Batzer MA. The impact of retrotransposons on human genome evolution. Nature Review Genetics. 2009;10:691–703.

[78] SanMiguel P, Tikhonov A, Y K Jin ea. Nested retrotransposons in the intergenic regions of the maize genome. Science. 1996;274:765–768.

[79] Kazazian HH, Moran JV. The impact of L1 retrotransposons on the human genome. Nat Genet. 1998;19:19–24.

[80] Smit AF, Hubley R, Green P. RepeatMasker Open-4.0; 2013-2015.

[81] Witherspoon DJ, Zhang Y, Xing J, Watkins WS, Ha H, Batzer MA, et al. Mobile element scanning (ME-Scan) identifies thousands of novel Alu insertions in diverse human populations. Genome research. 2013;23(7):1170–1181.

[82] Nakagome M, Solovieva E, Takahashi A, Yasue H, Hirochika H, Miyao A. Transposon Insertion Finder (TIF): a novel program for detection of de novo transpositions of transposable elements. BMC bioinformatics. 2014;15(1):71.

[83] Keane TM, Wong K, Adams DJ. RetroSeq: transposable element discovery from next-generation sequencing data. Bioinformatics. 2013;29(3):389–390.

[84] Fereydoun Hormozdiari, Iman Hajirasouliha, Phuong Dao, Faraz Hach, Deniz Yorukoglu, Can Alkan, Evan E Eichler, and S Cenk Sahinalp. Next-generation variationhunter: combinatorial algorithms for transposon insertion discovery. *Bioinformatics*, 26(12):i350–i357, 2010.

[85] Fiston-Lavier AS, Carrigan M, Petrov DA, Gonzalez J. T-lex: a program for fast and accurate assessment of transposable element presence using next-generation sequencing data. Nucleic acids research. 2011;39(6):e36–e36.

[86] To be determined Gardner et al. Paper

[87] Jurka J, Kapitonov VV, Pavlicek A, Klonowski P, Kohany O, Walichiewicz J. Repbase Update, a database of eukaryotic repetitive elements. Cytogenetic and genome research. 2005;110(1-4):462–467.

[88] Wheeler TJ, Clements J, Eddy SR, Hubley R, Jones TA, Jurka J, et al. Dfam: a database of repetitive DNA based on profile hidden Markov models. Nucleic acids research. 2013;41(D1):D70–D82.

[89] Alkes L Price, Neil C Jones, and Pavel A Pevzner. De novo identification of repeat families in large genomes. *Bioinformatics*, 21(suppl 1):i351–i358, 2005.

[90] Carly E Schaeffer, Nathaniel D Figueroa, Xiaolin Liu, and John E Karro. phraider: Pattern-hunter based rapid ab initio detection of elementary repeats. *Bioinformatics*, 32(12):i209–i215, 2016.

[91] Koch P, Platzer M, Downie BR. RepARK - de novo creation of repeat libraries from whole-genome NGS reads. Nucleic acids research. 2014;42:e80.

[92] Zhuang J, Wang J, Theurkauf W, Weng Z. TEMP: a computational method for analyzing transposable element polymorphism in populations. Nucleic acids research. 2014;42(11):6826–6838.

[93] Anna-Sophie Fiston-Lavier, Maite G Barrón, Dmitri A Petrov, and Josefa González. T-lex2: genotyping, frequency estimation and re-annotation of transposable elements using single or pooled next-generation sequencing data. *Nucleic acids research*, 43(4):e22–e22, 2015.

[94] Marçais G, Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics. 2011;27(6):764–770.

[95] Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome research. 2008;18(5):821–829.

[96] Myers EW. Towards simplifying and accurately formulating fragment assembly. J of Comp Biology. 1995;2:275–290.

[97] Durbin R, Eddy S, Krogh A, Mitchison G. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge, U.K.: Cambridge University Press; 1999.

[98] Cormen T, Leiserson C, Rivest R. Introduction to Algorithms. Cambridge, MA.: MIT press and McGraw Hill; 1992.

[99] Li H, Durbin R. Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics. 2009;25(14):1754–1760.

[100] 1000 Genomes Project Consortium. An integrated map of genetic variation from 1092 human genomes. Nature. 2012;491:56–65.

[101] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015.

[102] Peter H Sudmant, Tobias Rausch, Eugene J Gardner, Robert E Handsaker, Alexej Abyzov, John Huddleston, Yan Zhang, Kai Ye, Goo Jun, Markus Hsi-Yang Fritz, et al. An integrated map of structural variation in 2,504 human genomes. *Nature*, 526(7571):75–81, 2015.

[103] Rosenbloom KR, Armstrong J, Barber GP, Casper J, Clawson H, Diekhans M, et al. The UCSC Genome Browser database: 2015 update. Nucleic Acids Research. 2015;43:D670–681.

[104] Berlin K, Koren S, Chin CS, Drake JP, Landolin JM, Phillippy AM. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nature Biotechnology. 2015;33:623–630.

[105] Chaisson MJ, Huddleston J, Dennis MY, Sudmant PH, Malig M, Hormozdiari F, et al. Resolving the complexity of the human genome using single-molecule sequencing. Nature. 2015;517(7536):608–611.

[106] Robinson JT, Thorvaldsdóttir H, Winckler W, Guttman M, E S Lander ea. Integrative genomics viewer. Nature biotechnology. 2011;29:24–26.

[107] Albertin, C. B., Simakov, O., Mitros, T., Wang, Z. Y., Pungor, J. R., Edsinger-Gonzales, E., Brenner, S., Ragsdale, C. W., and Rokhsar, D. S. (2015). The octopus genome and the evolution of cephalopod neural and morphological novelties. *Nature*, **524**(7564), 220–224.

[108] Alkan, C., Sajjadian, S., and Eichler, E. E. (2011). Limitations of next-generation genome sequence assembly. *Nature methods*, **8**(1), 61–65.

[109] Berlin, K., Koren, S., Chin, C.-S., Drake, J. P., Landolin, J. M., and Phillippy, A. M. (2015). Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, **33**(6), 623–630.

[110] Boetzer, M. and Pirovano, W. (2012). Toward almost closed genomes with gapfiller. *Genome biology*, **13**(6), 1.

[111] Butler, J., MacCallum, I., Kleber, M., Shlyakhter, I. A., Belmonte, M. K., Lander, E. S., Nusbaum, C., and Jaffe, D. B. (2008). Allpaths: de novo assembly of whole-genome shotgun microreads. *Genome research*, **18**(5), 810–820.

[112] Chaisson, M. J., Wilson, R. K., and Eichler, E. E. (2015). Genetic variation and the de novo assembly of human genomes. *Nature Reviews Genetics*.

[113] Chapman, J. A., Ho, I., Sunkara, S., Luo, S., Schroth, G. P., and Rokhsar, D. S. (2011). Meraculous: de novo genome assembly with short paired-end reads. *PloS one*, **6**(8), e23501.

[114] Deorowicz, S., Kokot, M., Grabowski, S., and Debudaj-Grabysz, A. (2015). Kmc 2: Fast and resource-frugal k-mer counting. *Bioinformatics*, **31**(10), 1569–1576.

[115] Gnerre, S., MacCallum, I., Przybylski, D., Ribeiro, F. J., Burton, J. N., Walker, B. J., Sharpe, T., Hall, G., Shea, T. P., Sykes, S., *et al.* (2011). High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proceedings of the National Academy of Sciences*, **108**(4), 1513–1518.

[116] Gordon, D., Huddleston, J., Chaisson, M. J., Hill, C. M., Kronenberg, Z. N., Munson, K. M., Malig, M., Raja, A., Fiddes, I., Hillier, L. W., *et al.* (2016). Long-read sequence assembly of the gorilla genome. *Science*, **352**(6281), aae0344.

[117] Kosugi, S., Hirakawa, H., and Tabata, S. (2015). Gmcloser: closing gaps in assemblies accurately with a likelihood-based selection of contig or long-read alignments. *Bioinformatics*, **31**(23), 3733–3741.

[118] Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, **25**(14), 1754–1760.

[119] Li, R., Zhu, H., Ruan, J., Qian, W., Fang, X., Shi, Z., Li, Y., Li, S., Shan, G., Kristiansen, K., *et al.* (2010). De novo assembly of human genomes with massively parallel short read sequencing. *Genome research*, **20**(2), 265–272.

[120] Luo, R., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., Liu, Y., *et al.* (2012). Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*, **1**(1), 1.

[121] Manolio, T. A., Collins, F. S., Cox, N. J., Goldstein, D. B., Hindorff, L. A., Hunter, D. J., McCarthy, M. I., Ramos, E. M., Cardon, L. R., Chakravarti, A., *et al.* (2009). Finding the missing heritability of complex diseases. *Nature*, **461**(7265), 747–753.

[122] Miller, J. R., Koren, S., and Sutton, G. (2010). Assembly algorithms for next-generation sequencing data. *Genomics*, **95**(6), 315–327.

[123] Myers, E. W. (1995). Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, **2**(2), 275–290.

[124] OConnell, J., Schulz-Trieglaff, O., Carlson, E., Hims, M. M., Gormley, N. A., and Cox, A. J. (2015). Nxtrim: optimized trimming of illumina mate pair reads. *Bioinformatics*, **31**(12), 2035–2037.

[125] Paulino, D., Warren, R. L., Vandervalk, B. P., Raymond, A., Jackman, S. D., and Birol, I. (2015). Sealer: a scalable gap-closing application for finishing draft genomes. *BMC bioinformatics*, **16**(1), 230.

[126] Salzberg, S. L., Phillippy, A. M., Zimin, A., Puiu, D., Magoc, T., Koren, S., Treangen, T. J., Schatz, M. C., Delcher, A. L., Roberts, M., *et al.* (2012). Gage: A critical evaluation of genome assemblies and assembly algorithms. *Genome research*, **22**(3), 557–567.

[127] Simpson, J. T. and Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data structures. *Genome research*, **22**(3), 549–556.

[128] Simpson, J. T., Wong, K., Jackman, S. D., Schein, J. E., Jones, S. J., and Birol, I. (2009). Abyss: a parallel assembler for short read sequence data. *Genome research*, **19**(6), 1117–1123.

[129] Smit, A. F., Hubley, R., and Green, P. (1996). Repeatmasker open-3.0.

[130] Treangen, T. J. and Salzberg, S. L. (2012). Repetitive dna and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics*, **13**(1), 36–46.

[131] Zerbino, D. R. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, **18**(5), 821–829.

[132] Cemalettin Bekpen, Tomas Marques-Bonet, Can Alkan, Francesca Antonacci, Maria Bruna Leogrande, Mario Ventura, Jeffrey M Kidd, Priscillia Siswara, Jonathan C Howard, and Evan E Eichler. Death and resurrection of the human irgm gene. *PLoS Genet*, 5(3):e1000403, 2009.

[133] Steven A McCarroll, Alan Huett, Petric Kuballa, Shannon D Chilewski, Aimee Landry, Philippe Goyette, Michael C Zody, Jennifer L Hall, Steven R Brant, Judy H Cho, et al. Deletion polymorphism upstream of irgm associated with altered irgm expression and crohn's disease. *Nature genetics*, 40(9):1107–1112, 2008.

[134] Jeffrey A Bailey, Ge Liu, and Evan E Eichler. An alu transposition model

for the origin and expansion of human segmental duplications. *The American Journal of Human Genetics*, 73(4):823–834, 2003.

[135] Jinchuan Xing, Yuhua Zhang, Kyudong Han, Abdel Halim Salem, Shurjo K Sen, Chad D Huff, Qiong Zhou, Ewen F Kirkness, Samuel Levy, Mark A Batzer, et al. Mobile elements create structural variation: analysis of a complete human genome. *Genome research*, 19(9):1516–1526, 2009.

[136] Keiko Akagi, Jingfeng Li, and David E Symer. How do mammalian transposons induce genetic variation? a conceptual framework. *Bioessays*, 35(4):397–407, 2013.

[137] Michael Cowley and Rebecca J Oakey. Transposable elements re-wire and fine-tune the transcriptome. *PLoS Genet*, 9(1):e1003234, 2013.

[138] Benoît Chénais, Aurore Caruso, Sophie Hiard, and Nathalie Casse. The impact of transposable elements on eukaryotic genomes: from genome size increase to genetic adaptation to stressful environments. *Gene*, 509(1):7–15, 2012.

[139] J Kenneth Baillie, Mark W Barnett, Kyle R Upton, Daniel J Gerhardt, Todd A Richmond, Fioravante De Sapio, Paul M Brennan, Patrizia Rizzu, Sarah Smith, Mark Fell, et al. Somatic retrotransposition alters the genetic landscape of the human brain. *Nature*, 479(7374):534–537, 2011.

[140] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology*, 33(6):623–630, 2015.

[141] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[142] Chong Chu, Jin Zhang, and Yufeng Wu. Gindel: accurate genotype calling of insertions and deletions from low coverage population sequence reads. *PloS one*, 9(11):e113324, 2014.

[143] Nicole G Coufal, José L Garcia-Perez, Grace E Peng, Gene W Yeo, Yangling Mu, Michael T Lovci, Maria Morell, K Sue OShea, John V Moran, and Fred H Gage. L1 retrotransposition in human neural progenitor cells. *Nature*, 460(7259):1127–1131, 2009.

[144] Eunjung Lee, Rebecca Iskow, Lixing Yang, Omer Gokcumen, Psalm Haseley, Lovelace J Luquette, Jens G Lohr, Christopher C Harris, Li Ding, Richard K Wilson, et al. Landscape of somatic retrotransposition in human cancers. *Science*, 337(6097):967–971, 2012.

[145] Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, page btw152, 2016.

[146] Alysson R Muotri, Vi T Chu, Maria CN Marchetto, Wei Deng, John V Moran, and Fred H Gage. Somatic mosaicism in neuronal precursor cells mediated by l1 retrotransposition. *Nature*, 435(7044):903–910, 2005.

[147] Zuojian Tang, Jared P Steranka, Sisi Ma, Mark Grivainis, Nemanja Rodić, Cheng Ran Lisa Huang, Ie-Ming Shih, Tian-Li Wang, Jef D Boeke, David Fenyö, et al. Human transposon insertion profiling: Analysis, visualization and identifi-

cation of somatic line-1 insertions in ovarian cancer. *Proceedings of the National Academy of Sciences*, page 201619797, 2017.

[148] K. Hou, W. c. Feng, and S. Che. Auto-tuning strategies for parallelizing sparse matrix-vector (spmv) multiplication on multi- and many-core processors. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, May 2017.

[149] K. Hou, H. Wang, and W. c. Feng. Delivering Parallel Programmability to the Masses via the Intel MIC Ecosystem: A Case Study. In *2014 43rd International Conference on Parallel Processing Workshops*, pages 273–282, Sept 2014.

[150] K. Hou, H. Wang, and W. C. Feng. AAlign: A SIMD Framework for Pairwise Sequence Alignment on x86-Based Multi-and Many-Core Processors. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 780–789, May 2016.

[151] Kaixi Hou, Weifeng Liu, Hao Wang, and Wu-chun Feng. Fast Segmented Sort on GPUs. In *Proceedings of the 2017 International Conference on Supercomputing*, ICS '17. ACM, 2017.

[152] Kaixi Hou, Hao Wang, and Wu-chun Feng. ASPaS: A Framework for Automatic SIMDization of Parallel Sorting on x86-based Many-core Processors. In *Proceedings of the 29th ACM on International Conference on Supercomputing*, ICS '15, pages 383–392, New York, NY, USA, 2015. ACM.

[153] Kaixi Hou, Hao Wang, and Wu-chun Feng. Gpu-unicache: Automatic code generation of spatial blocking for stencils on gpus. In *Proceedings of the ACM Conference on Computing Frontiers*, CF '17. ACM, 2017.