IDENTIFYING AND CHARACTERIZING TRANSPOSABLE ELEMENTS IN THE GENOME

Anwica Kashfeen

A dissertation submitted to the faculty at the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2022

ABSTRACT

Anwica Kashfeen: Identifying and Characterizing Transposable Elements in the Genome
(Under the direction of Leonard McMillan)

A large fraction of mammalian genome consists of transposable elements (TEs). These elements are segments of DNA that either move or are copied from one place in the genome to another. TEs are a significant source of genetic variation and are directly responsible for many diseases. It is difficult to identify, map, characterize, and determine the zygosity of TEs using current high-throughput short-read sequencing data because of their numerous copies in the genome. Existing approaches search for TE insertion (TEi) by aligning millions of mostly irrelevant short reads to either a reference genome or a TE sequence library. In this dissertation I describe two alignment-free novel TEi detection algorithms, ELITE and Frontier which outperform existing tools in several different categories. Both algorithms use *local-genome-assembly* where ELITE is template-dependent and Frontier is template-free. The key idea is to focus on identifying the boundary of TE insertion which contains partial TE and non-TE context. I use a MultiString Burrow Wheeler Transform, msBWT-based data structure to store and index all the reads from a high-throughput sequencing dataset and leverages additional data structures FM-index and Longest Common Prefix (LCP) to efficiently search for TEi boundaries. I show that combination of two methods can identify nearly all the Endogenous RetoVirus (ERV) insertions that are segregating in a population with more than 100 samples. These methods can also be used to identify very recent or *de novo* TE insertions. Moreover, characterization based on the sharing pattern of ERVis allows us to study phylogeny within a population.

To my family

ACKNOWLEDGEMENTS

I will be forever grateful for having a wonderful advisor like Leonard McMillan during my Ph.D. It would have been impossible to complete this dissertation without his guidance and support. His passion for solving both computational and biological problems always inspired me to do the same. Eventually, this leads me to this dissertation where I answered many biological questions by the means of computational methods. Other than being an excellent academic advisor, he also taught me ethics, kindness, humility and the list goes on.

I am fortunate to work with so many talented people from the Genetics department, especially Dr. Fernando Pardo Manuel de Villena. He encouraged me, believed in my work, and provided many biological insights to make it even better.

I am thankful for all my labmates, classmates, colleagues in the Ph.D. program. All of them, from time to time, provided various resources which filled the gaps in my educational background. They were also there for me as friends, conversational partners which made the seemingly difficult job quite easier.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

All organisms except viruses are made of one or multiple cells. For example, the average human body contains more than 40 trillion cells (Bianconi et al., 2013). Within almost every cell there is a complete blueprint that dictates many of the organism's function. This blueprint containing genetic information is called a *genome*. Non-virus genomes are encoded as a sequence in a special molecule called DNA. This sequence is composed entirely from a linear arrangement of four molecules or bases, Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). DNA incorporates many forms of redundancy that help to assure both its stability and robustness. It is double stranded where each strand stores the complementary information. The complement version of DNA arises from chemical bonds between the nucleotides of two DNA strands where Adenine always binds to Thymine, Cytosine always binds to Guanine. As a result of this two stranded structure, we typically refer the nucleotides as basepairs instead of bases. In addition to complimentary copies, DNA also strengthens and protects this genetic information in coiled structures called chromosomes. A typical human genome is about 3 billion basepairs long and and is distributed over 23 pairs of chromosomes. To put it simply, a genome is a long string that can be divided into multiple substrings called chromosomes where each chromosome is composed of 4 letters, A for Adenine, C for Cytosine, G for Guanine, and T for Thymine.

High-throughput sequencing also known as Next Generation Sequencing (NGS) is one of the significant inventions of the last century. Due to its parallel processing and high throughput, we are now able to sequence a large genome like the human in less than a day (Behjati and Tarpey, 2013). For contrast, first-generation sequencing technologies like Sanger sequencing (Sanger and Coulson, 1975) took more than a decade just to present a draft of the human genome. However, neither first nor second generation sequencing methods can output the whole genome at once.

Instead, they produce many short overlapping fragments called reads. While the First generation sequencers can produce reads longer than 500 bases, the reads produced by NGS are typically very short (within 100-150 bases). All these reads together are like pieces of multiple jigsaw puzzles where each puzzle corresponds to a different chromosome. And even within a chromosome, each piece has its particular place/position. Therefore, to reconstruct the original genome, one needs to put majority of the reads in the right place of the right chromosome and also in the right orientation. The overlapping nature of reads makes it even harder to find its correct position. However, there are several algorithms available to perform this genome reconstruction. These programs are called assemblers and the process is called genome assembly.

The typical genome assembly process starts with a short sequence of k nucleotides called a *seed kmer*. Usually these are known sequences or genes that are presumed to be unique substrings in the genome. Those seed k-mers are then extended in both directions as long as possible. However, the extension is terminated whenever a branch or repeated sequence is reached. A branch occurs when there are overlapping substrings with different sequences in the direction of extension. Repeats occur when a previously seen substring reappears during the extension, and are equivalent to finding a cycle in a graph. Repeats create ambiguity (i.e. loops) which many lead to many different branching paths. Therefore, it is common practice to mask repeats and resolve their connectivity via other experimental means such as optical mapping (Zhou et al., 2007). The original purpose of RepeatMasker (Smit et al., 1996) was to "mask" all the repeats in a genome so that assemblers and aligners can easily ignore them to focus on the non-repeated sequences. Ignoring repeats simplifies many analyses but hinders the chance to discover other interesting biological phenomena. Subsequent discoveries have shown that the majority of these repeats are not random, but follow some specific patterns. Some of the repeats in the genome are tandem, which refers to a repeat where a small substring (3-200 bases) is copied many times and many of the copies are adjacent to each other in the genome. On the other hand, there are also many repeats that are distributed randomly throughout the genome. The length of these non-tandem repeats vary depending on their type and can be as long as 10,000 bases composed of otherwise normal appearing sequence. A significant portion of these interspersed repeats are Transposable Elements (TEs) (McClintock, 1984). TEs are genetic sequences that have, or have had in the past a unique mobility feature that allows them to change their position in the genome, hence the name “transposable”. In the following sections, I will discuss more about these elements, why they are important, why they are hard to find, why the existing methods are not sufficient, and finally why do we need to develop a different approach to locate and identify these repeated sequences.

1.1 Transposable Elements

In the first half of the 20th century, scientists thought of genomes as a mostly static entities with a large, but stable organization, upon which relatively rare mutations accumulated over time. The most common type of mutation, which only affects 1 base (Consortium et al., 2015) is called a Single Nucleotide Polymorphism (SNP). SNPs are observed more frequently in non-coding or non-functional genomic regions (Barreiro et al., 2008). Due to the short and static nature of these mutations, they can easily be detected by performing alignments of sequenced read fragments to a known, and previously assembled reference genome. However, in the late 1940s Barbara McClintock discovered that the organization of genomes is far more dynamic than had been previously imagined. Certain genomic rearrangements are due to segments of DNA that either *jump* or are *copied* from one place in the genome to another. These mobile segments are now called *Transposable Elements* (TEs). When McClintock first introduced the concept of TEs in a maize genome, her peers were skeptical about these elements. Over the next several decades, other scientists confirmed the activity of TEs in other species like fungus, bacteria, etc. Eventually, her theory of a dynamic genome was accepted and McClintock was awarded the Nobel Prize for discovering TEs in 1983.

When sequencing become available, it has subsequently been discovered that a significant fraction of eukaryotic (cells with a nucleus) genomes sequences are composed of TEs and their vestiges. There are many types of TEs, and these can either be DNA-mediated or RNA-mediated.

While DNA-mediated TEs move in a cut-paste fashion, RNA-mediated ones employ a copy-paste mechanism of insertion that, over time, increases the overall size of the genome. About 45% of the human reference genome (Consortium et al., 2001), 37% of the mouse reference (Consortium et al., 2002), and 85% of maize genomes (SanMiguel et al., 1996) consist of TE-derived sequence. In comparison to SNPs, mutations due to TE insertions are believed to have more profound effects on the biological function (e.g. gene expression) (Uzunović et al., 2019) and phenotypic changes (traits) (Chiang et al., 2017). The overwhelming majority of TE insertions are largely detrimental to the organism and there is a strong selection pressure against passing new TE insertions on to future generations. Most of the inherited TE insertions in a genome eventually lose their ability to move, but their remnants remain in the genome and continue to pass on to the next generation. Recent studies suggest that many segregating structural variants in humans are due to these inactive TEs that were inserted a long time ago (Ebert et al., 2021).

1.2 Importance of Transposable Element

Consequences of Transposable Elements are far more significant other than just making a genome grow. For decades, scientists were skeptical about the ongoing TE activity and used to refer them as *"junk DNA"* or *"selfish DNA"*. However, recent research that depends on modern sequencing technology, have unveiled many consequences of recent TE insertions despite being canonically thought of as non-coding DNA. In this section, we discuss the importance of TE insertions in terms in the day-to-day function of the organism as well as it's impacts on future progeny.

- TE insertions in somatic cells (newly derived cells that emerge during normal mitotic cell divisions) are responsible for many diseases. Even though, insertions in somatic cells can not be passed to the offspring, they can be extremely deleterious. For example, many diseases like hemophilia A, neurofibromatosis, choroideremia, cholinesterase deficiency, Apert syndrome, Dent's disease, β-thalassemia, and Walker-Warburg syndrome are reported to be consequences of TE translocations (Wallace et al., 1991) (van den Hurk et al., 2003) (Muratani et al., 1991) (Chen et al., 2005) (Belancio et al., 2008) (Babushok and Kazazian Jr, 2007).

- TE insertions in germline can also cause genetic disease. Hundreds of TE insertions are discovered so far that are directly associated with disorders like Hereditary Breast and Ovarian Cancer (HBOC) syndrome (Teugels et al., 2005), as well Lynch syndrome (Li et al., 2020)(Hancks and Kazazian, 2016). Lynch syndrome causes colon cancer and increases the risk for many other type of cancers.

- Germline TE insertions also facilitate the study of TE biology and genome evolution. A deleterious insertion in germline is most likely to be purged quickly. On the other hand, non-deleterious TEis can be passed on from generation to generation and remain permanently in a population. Over the time, this accumulation of TEs makes the genome size bigger. Most importantly, germline TEs make it possible to track heritability and build phylogeny based on the shared insertion events. Population-based analysis can further be used to identify the the historical periods when a TE was actively mobile and periods where they were silenced.

1.3 Challenges in Finding Transposable Element Insertions

A large fraction of any eukaryotic genome is composed of transposable elements (TEs). However, they are not easy to locate from high-throughput-short reads for both computational and biological reasons. Most common challenges while locating TE insertions in genomes are stated below:

- Analyzing and distinguishing between the numerous copies of transposable elements in the genome is a computational challenge. A typical human genome which is sequenced at 30x coverage can have more than 500 million reads. Since TEs comprises almost 50% of the human genome, which indicates almost 250 million reads are associated with TEs.

Locating and analyzing TEs thus remain a computational challenge as we have to deal with millions of highly homologous reads. Unless there is an efficient algorithm to identify TE insertions, it will cost high in terms of calculation, CPU, and memory usage.

- Location of TE insertion can not be identified using typical variant searching techniques. SNPs and small INDELs can be easily detected by aligning all the reads to a reference genome. In this case, about 99% of the bases will align perfectly (Treangen and Salzberg, 2012) (Teissandier et al., 2019) due its short length. However, TEs affect more bases of a short read than SNPs or INDELs. In fact a single read is not enough to capture an entire insertion event. Moreover, to identify the position, it requires the same read to have some nonTE segment in addition to a TE segment. This creates a problem as length of TE sequence varies from 300 to 8000 basepairs, which is longer than typical short reads (100 - 150 bp).

- The biology of TEs also makes it hard to locate their inserted locations. Current TE locating methods are based some known TE templates found in online libraries like RepeatMasker (Smit et al., 1996) and RepBase (Bao et al., 2015). This libraries usually contain consensus TE sequences that were found in the assembled genomes of an organism. However, TE sequences vary between species and between separated populations. Most TEs are probably of viral origin. As a result, it is possible to have completely different types of TEs or viruses that are not found elsewhere in the tree of life. Thus, detecting unknown TE types is a significant challenge.

1.4 Limitations of Existing Approaches

Most existing approaches for detecting TE insertions (TEi) require that whole-genome short-reads be aligned to either a reference genome sequence or to a catalog of known TE sequences. Using a reference alignment to detect TEis presents several challenges. The TEis already in the reference genome tend to attract the bulk of TE-like sequence and *split-reads* (those that span a novel insertion point) are often incorrectly mapped to regions where differences are within alignment tolerances. Moreover, reference genomes are updated from time to time, thus the whole alignment process needs to be done for this new version. On the other hand, alignments of short-reads to TE catalogs depend on the inclusion of TE sequences that are generated either by collecting sequences from seminal discoveries (Bao et al., 2015) or by synthesizing consensus sequences for TE families (Wheeler et al., 2012) to get a match. Most importantly, it requires that a model for each TE type is known in advance, and the sequences of TE types vary significantly with TE classes and between species. If an isolated population contains a novel TE, chances are very low that they will be found in TE libraries.

Even the best available TE discovery algorithms are not well-suited for population-based TEi sharing analysis. As discussed previously, other than causing diseases, TE insertions are a common source of structural variation. Just like SNPs, TEis can be used to track ancestry and phylogeny in a given population. However, there exists only a few and incomplete pipelines that attempts to make a population-level analysis based on the TEi sharing pattern.

1.5 Thesis Statement

My research attempts to close many of these existing gaps in TE discovery methods using approaches that do not depend on genome alignments or TE libraries. These methods can aid in understanding the variations of the TE landscape within a population. I also hope to detect and differentiate recent and segregating TE insertion events to demonstrate that this activity is ongoing. My approach is summarized in the following thesis statement:

”It is possible to localize Transposable Element (TE) insertions in the genome from short reads by performing a local-genome-assembly around the insertion boundary within a given sample. Further, by combining all the TE insertions that exist in a population, each insertion can be characterized as fixed (present in everyone), segregating (present in a subset of samples), or de novo (present only in a sample or its biological twin). This characterization based on TEi

sharing pattern, can further be used for analyzing the evolutionary relationship among each individual in a population. "

1.6 Contributions

I introduce a pipeline that is well-suited for a population-based TEi sharing analysis. Current algorithms focus primarily on identifying TE insertions from an individual genome. Many clinical applications mainly focus on the TEis in somatic cells as they are more likely to cause diseases. However, just like SNPs or INDELs, TEis can also be treated like a genetic variant in a segregating population. This requires a TEi detection method to be fast enough so that it can be applied to hundreds of genomes presented as short-reads. Moreover, this method needs to have low false negatives and false positives for capturing the most reliable TEi sharing pattern. Therefore, keeping these two points in mind, we introduce the concept of *local-genome-assembly* in the process of locating TE insertion. We avoid the typical time and memory-consuming computational steps that require the alignment of billions of irrelevant short-reads. Instead, we focus mainly on the boundary of the TE insertion which is local to a particular insertion event. Even though *local-genome-assembly* is not an absolutely new idea, and already has been used in finding INDELs (Mose et al., 2019), we are the first to make it work for in the context of TE insertions.

We proposed two novel approaches for finding insertion of transposable elements. Both of our approaches, ELITE (chapter 3), and Frontier (chapter 4) work on the *split-read* principle. *split-reads* refer to the TEi boundaries that contain partial nonTE and partial TE sequence adjacent to each other. While typical assembly starts from an unique seed kmer and ends in a repeat, our local assembly does exactly the opposite. It starts from a TE or a highly repeated segment and ends in a segment that is nonTE or non-repeated. Even though both ELITE and Frontier use this same assembly technique, the key difference between them lies on the use of TE template. ELITE needs a template to starts with, then allows adequate mutations to capture distant and unknown TEs of the same type. On the other hand, Frontier is template-free and identifies all the instances of TE insertions using the pervasive nature of TE sequences. Since Frontier does not start with any TE template, it uses one classifier to detect the true TEi boundaries and another to classify the TE type for each true insertion.

The ability to identify presence and absence probes for each TEi also makes our pipeline unique from existing methods. After the initial TEi detection in an individual sample, we combine all the TEis in a population based on their location. Each event that takes place in the same location is considered as one single event. For each event, TE probes are run on all the samples in a given population to detect if there's any false negative. On the other hand, for the samples without TEi, we use the absent probes to further confirm the claim. Together with these TE present and absent probes, we annotate all the insertions in a population. Therefore, it provides a unique way to visualize how TE insertions are distributed in a population i.e., how many of them are present in every sample (fixed), how many are shared by only a subset of samples (segregating), and how many are unique (de novo) to a sample (or a few closely related samples). This TEi sharing pattern gives us an idea about the age of each TE insertions. For example, TEis that are present in everyone are most likely to be an old insertion which was passed on to all the descendants, whereas a *de novo* insertion is likely to have occurred within this or recent generations.

As proof of this concept, this dissertation also includes an extensive analysis of the TEi diversity within a population. This starts from the background materials to algorithm design, finding limitations of the proposed algorithm to improving it, applying the improved algorithm in an individual sample to extending it in a large population, detecting pattern of TEi sharing in a population to analyzing the phylogeny.

The structure of this dissertation is as follows: In chapter 2, I first provide additional background materials to understand this dissertation. In chapter 3 and chapter 4, I describe two different TE detection approaches where each is based on *local-genome-assembly*. In chapter 5, I demonstrate how these pipelines can be used to understand TE diversity in a large genetically varying population. In chapter 6, I document all the tools and resources to make them available to public for future research. In chapter 7, I conclude with discussions of some future possible research areas.

CHAPTER 2: BACKGROUND

In this chapter, I discuss the relevant previous work and terminologies related to this dissertation. Starting with the concept of transposable element insertion, their various types and characteristics that are associated with each insertion. Then I discuss the nature of whole-genome sequencing, and two different models that are currently applied on the sequenced data to find TE insertions. Finally, I describe the algorithms and data structures that my methods are based on.

2.1 Genome, Repeats, Transposable Element

A genome is a complete set of genetic instructions needed to build and maintain an organism. Most cells in a living organism have one or more copy these instructions that act as a blueprint for all the possible functions that cells perform. Instructions in genome are encoded in a molecule called DNA that is composed of four nucleic acid bases, Adenine, Cytosine , Guanine, Thymine which are typically represented by the letters A, C, G, and T respectively. Two strands of DNA, each carrying complementary information, together form a coil-like structure to maintain stability. From a computer science perspective, we can think of DNA as a long string composed from an alphabet of four symbols. Many genome sequences are divided into multiple segments called chromosomes, and the small differences in these sequences makes each individual's genome unique. A typical mammalian genome is composed of approximately 3 billion bases of DNA. However only 2% of genomes comprise a particular type of functional unit call genes. These genes provide the instructions for constructing biological materials, such as proteins and functional RNAs. The rest of the genome is full of different kinds of repeats. This organization is typical of most eukaryotic genomes — organisms whose cells have a nucleus, including plants and animals (Lander et al., 2001)(Waterston and Pachter, 2002)(Bennetzen, 2000).

These repeated elements can be categorized into different classes based on their structure and functionality. Some of them protect chromosomes from deterioration, i.e., telomeres, and some of them are just simple tandem repeats. In this dissertation I am primarily interested in a class of repeat, that is either currently capable of moving or copying itself within a genome, or had been able to do so historically. This repeat class is called Transposable Elements (TEs). Transposable elements alone account for to more than 90% of the repeats in human and mouse genomes. There are also different kinds of TEs depending on their insertion mechanism, which we describe in the next section.

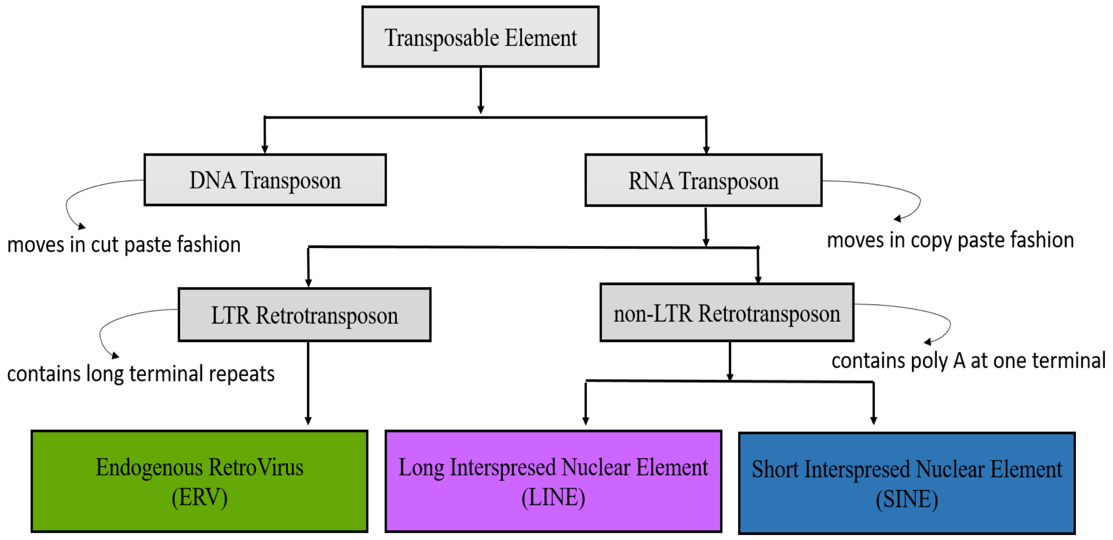

Transposable Element
DNA Transposon
RNA Transposon
moves in cut paste fashion
moves in copy paste fashion
LTR Retrotransposon
non-LTR Retrotransposon
contains long terminal repeats
contains poly A at one terminal
Endogenous RetroVirus (ERV)
Long Interspreded Nuclear Element (LINE)
Short Interspresed Nuclear Element (SINE)

Figure 2.1: Classification of Transposable Elements (TEs). TEs can be categorized into four primary types based on their structure and properties. DNA Transposons are sometimes called jumping genes because they can remove themselves from one place in the genome and reinsert themselves in a new place. A second family of TEs rely on the RNA transcription to propagate by making multiple copies. RNA transposons can be further subdivided according to structure. Enodegenous RetroVirus (ERV) are remnants from ancient viral infections that have been permanently integrated into the genome. ERVs are distinguished by Identical Long-Terminal Repeat sequences flanking both sides of the viral sequence. Another class of RNA transposon is a Long Interspersed Nuclear Element (LINE), which contain two key genes necessary for RNA transcription. A third class of RNA transposon, Short Interspersed Nuclear Element (SINE) is unable to copy itself independently, but instead relies on the two genes of active LINEs. LINEs and SINEs are the most common type of TE in both human and mouse consisting more than 40% of their genomes.

2.2 Types of Transposable Element

Transposable Elements (TEs) are segments of DNA that either move or are copied from one place in the genome to another. Different classes of TE are shown in figure 2.1. There are two types of TEs depending on their insertion mechanism. Some TEs move in a cut-and-paste manner, and are called DNA transposons. When they move, they leave their old place and are inserted to a new place. These are relatively rare events in mammals, but common in many plants. A second type of TE, is far more common in mammalian genomes relies on the RNA transcription to propagate and they move in a copy-and-paste fashion. These types of TEs are called RNA transposons. Due to their copy-and-paste nature, RNA transposons tend to increase genome size and their insertions can either damage or interfere with existing functional sequences. On rare occasions they can create entirely new function. To cover a large fraction of TEs, I thus decided to focus on finding and locating RNA transposons. This also allows me to find the old insertions still in the genome unless they were deleted. RNA transposons can further be classified into two categories: 1) LTR retrotransposons, 2) non-LTR retrotransposons. LTR retrotransposons contains identical *long terminal repeats* on both ends of the TE. The most common types of RNA-transposons in mammals are (i) Endogenous RetroViruses (ERV), (ii) Long Interspersed Nuclear Elements (LINEs) and (iii) Short Interspersed Nuclear Elements (SINEs). More than 99% of the TEs in human and mouse genomes fall into one of these three categories (Smit et al., 1996).

2.3 Insertion Mechanism of RNA Transposons

RNA transposons move in a copy-and-paste manner and follow common patterns when inserted into a host genome. At first it makes a staggered cut with a 5'-overhang in the target DNA, at the point where it inserts itself in the genome. This leads to a feature in which one small DNA segment at the insertion point ends up being copied on both sides of the insertion. The size of this small segment can vary for different TEs, but it is usually between 6-15 basepairs long and in the same orientation. These repeated small DNA segments are called Target Site Duplications

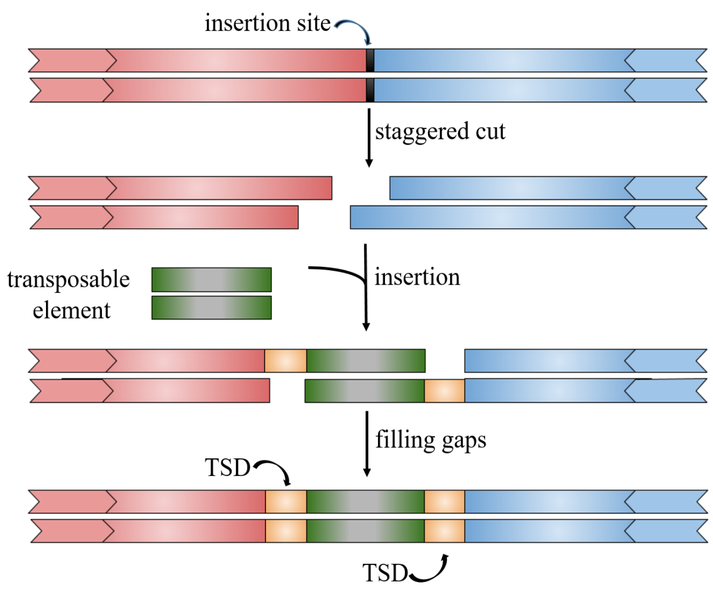


Figure 2.2: Insertion Mechanism of RNA Transposon: Transposable elements make a staggered cut in the target DNA leaving a two overhanging ends. Each strand of TE attaches itself with one of these ends. The remaining gap is then filled by duplicating a missing bases at the insertion site. The overall result is that each incorporated TE is enclosed between two repeated sequence, which are called Target Site Duplication (TSD).

(TSDs). This is separate from the LTR sequences of ERVs, which are part of the TE's sequence. TSDs are not part of a TE, but arise from the context of where a TE inserts itself into the DNA sequence, on both the proximal and distal end of the TE. TSDs are common to both LTR and non-LTR TEs. TSDs are analogous to open and closing parenthesis of the sequence. The same type of TE will have different flanking TSD repeats around it depending on where it is inserted in the genome. When transposon jumps to a new location, these flanking repeats do not move with it, but are just left in the genome as "foot prints". These TSDs are important feature of an insertion and facilitates identification and verifying recent insertions.

2.4 Short Read Sequencing

Conceptually a genome is multiple large chromosome strings composed of 4 bases (A, C, G, T). Each DNA molecule itself is composed of two redundant parts, a forward (5'-3') copy and a reverse-complemented copy that it is bonded to. When genomes are sequenced, DNA molecules are extracted from many thousands of cells. The DNA molecules are then independently chopped into suitably sized fragments before sequencing. Frequently, the forward and reverse complement strands of a fragment are sequenced together and are considered the ”paired-ends” of the fragment. Lastly, equal-sized substrings from each fragment end are sequenced, and these substrings are called “reads”. Together the two reads from the two-ends of a fragment are called “mates” and, when considered together they are called a “read pair”. Figure 2.3 shows an example of a short read-pair. This read pair consists of a read and its mate shown in pink and yellow respectively. The number of reads typically sequenced in a genome depends on the genome’s size and the desired coverage. For example a human genome sequenced with 150 base-pair reads with 30x coverage can have around half a billion reads once sequenced. Modern Next Generation Sequencing (NGS) technology is optimized for high throughput and accuracy so that it can be effectively applied to large genomes. In order to maximize accuracy and throughput, the length of the reads are typically limited to a range of 100-150 base pairs. Depending only on short reads creates ambiguities when analyzing repeated sequences. Note that, transposable elements (TEs) are not only repetitive in nature but also longer than typical short reads. That’s why identifying TEs requires more attention than any other genomic variants.

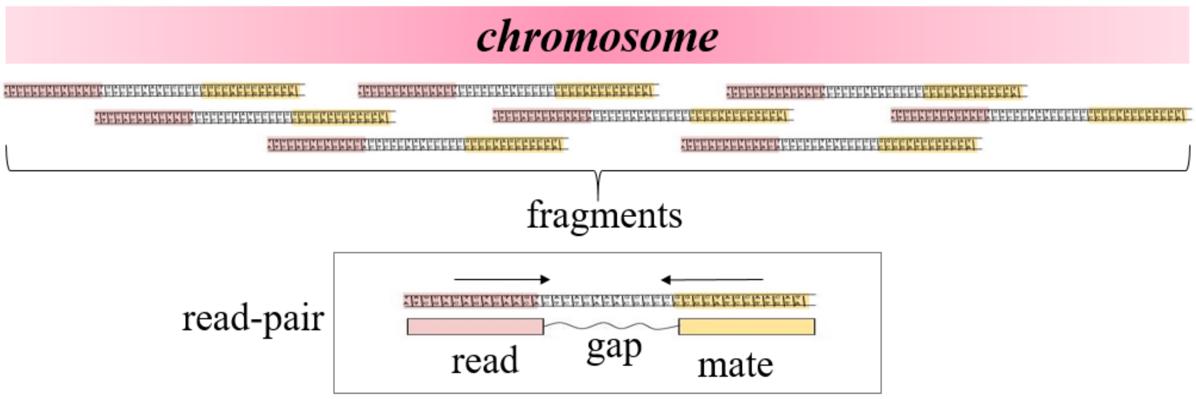


Figure 2.3: An example of a sequenced short read. Genome is chopped up into multiple overlapping fragments. For each read, a mate is sequenced from the other direction producing a read-pair. Read and pair are contiguous segments in the genome and have a gap in between them.

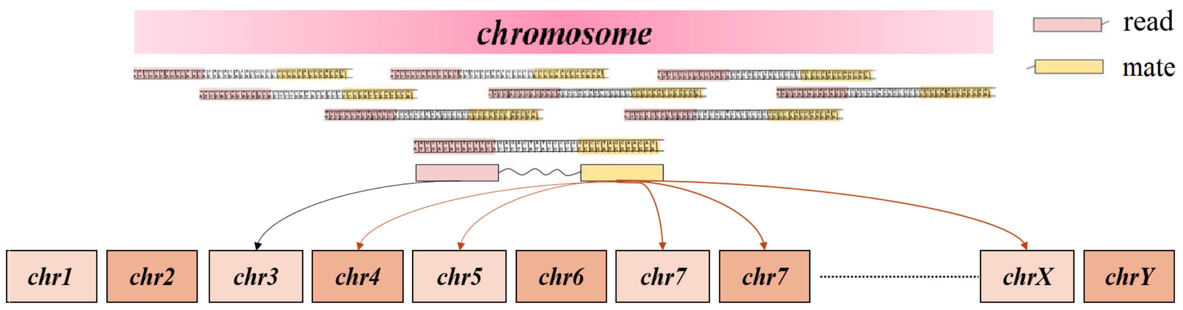


Figure 2.4: Example of a discordant read-pair, where read (pink) maps uniquely in chromosome 3, but its mate (yellow) maps multiple times in chromosome 4, 5, 7, and X.

2.5 Discordant Read-Pair

A read-pair is called discordant if at least one of the following is true: 1) one read maps uniquely to the reference genome while its mate maps to multiple places 2) both a read and its mate map uniquely but in the same orientation 3) a read and its mate map uniquely in the same position and in the same orientation, but their distance is more than the expected length (greater than fragment length). TEs are expected to be repeated throughout the genome, so DNA sequences of TE origin will tend to map to multiple places in the genome. At the same time, often a read's mate will map uniquely. Such read-pair falls under the first category of discordant. The uniquely mapped mate of a discordant pair can be used to infer a TE's location. Figure 2.4 shows an example of a discordant read-pair, where one read (shown in pink) maps uniquely at chromosome 3, but its mate (shown in yellow) maps to multiple positions (chromosome 3,5,7,and X) in the genome.

Many existing methods like MELT (Gardner et al., 2017), TEMP (Zhuang et al., 2014), RetroSeq (Keane et al., 2012) use discordant read-pairs to identify potential TE insertions. By running a BWA-like alignment tool on a sequenced dataset, all reads can be mapped to a known reference genome. For a large genome with 30x coverage, it often takes around 10 hours for BWA to finish an alignment. Even though, aligning is a time consuming step, the output of alignment is known to be useful for other analysis unrelated to TE discovery. However, since the alignment is based on a reference genome, the whole process needs to be repeated when a new genome version is released. Finding discordant read-pairs also heavily relies on accurate alignment tools. If any read contains more or less equal parts of TE and non-TE sequence, then depending on the alignment's tolerance level, it can either be mapped uniquely, multiple times, or not at all. In addition, the insertion location reported by discordant read-based methods can be significantly far from the actual insertion point.

2.6 Split-Read

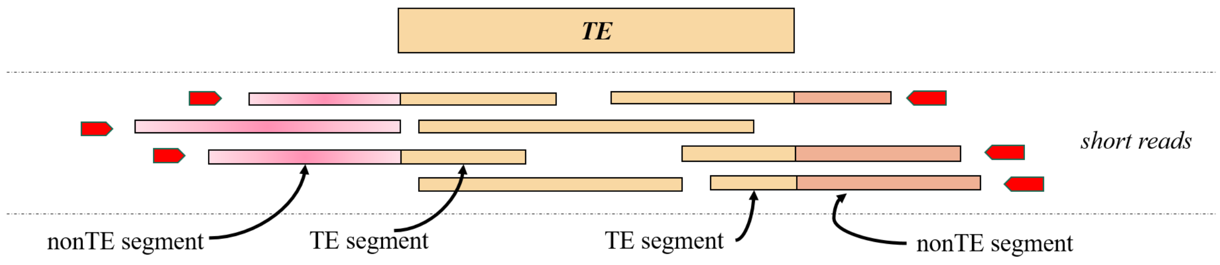


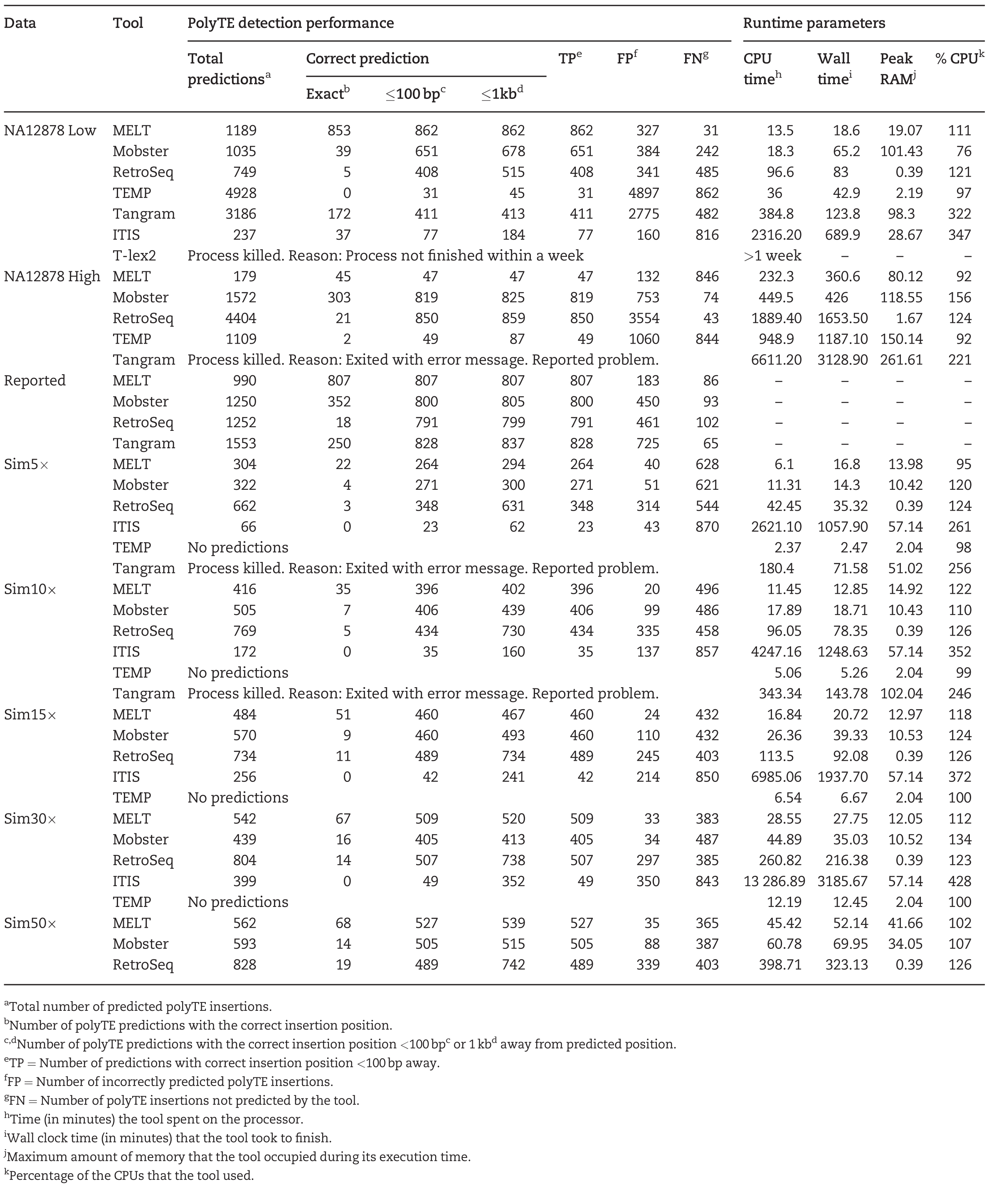
Figure 2.5: Example of split reads. Split-reads partially align with a known TE templates shown by the red arrows. Split-read contains boundary reads, which can be either proximal or distal TE boundary.

A split-read is a single read that can be broken into two non-overlapping segments where the two segments map to different places in the genome. In the context of TE insertion, split-read can be defined as the read where one segment contains TE sequence and the other segment contains non-TE sequence. Figure 2.5 shows an example with several split-reads that partially align with a given TE sequence. Since split-reads contain the boundary of TE they can be used for detecting accurate insertion locations. Usually paired-end reads are not required to identify potential TE insertions when using split-read approaches. However, the mate read can be useful for identifying the insertion location the TE is inserted into repetitive genomic regions.

Some existing methods for finding TE insertions like Relocate2 (Chen et al., 2017), ITIS (Jiang et al., 2015) use split-reads to find a breakpoint where a TE sequence is inserted. Typical split-read models first align all the reads from a sequenced dataset to a set of known TE templates rather than a reference genome as is case with discordant read-pair based methods. They identify those reads that contains partial TE and partial non-TE sequences. This alignment is not only time consuming but also useless for any other non-TE-related genomic analysis. Moreover, they rely on a known set of TE templates which is usually obtained from TE databases like RepeatMasker(Smit et al., 1996) or Dfam(Wheeler et al., 2012). Therefore, split-read models fail to identify any TE insertions where a TE template is not present in any databases. Split-read approaches tend to have high true-positive rates when compared to discordant read-pair based methods and they can predict the insertion locations much more accurately.

A recently published paper (Rishishwar et al., 2016) compared TE discovery and mapping methods. It benchmarked seven different packages including: MELT, Mobster, RetroSeq, TEMP, Tangram, ITIS, and T-lex2. The comparisons were done by using both real and simulated data. Figure 2.6, provides a summary of their results and observations. Among these, only T-lex2 is a tool that uses split-read-model exclusively for finding TE insertions. RetroSeq and TEMP use discordant read model exclusively. MELT, Tangram, ITIS, Mobster use a combination of both models.

The real dataset used in the evaluation was a female sample from the 1000 human genome project (Siva, 2008). This sample has been extensively analyzed and validated for benchmarking structural variations and TEi detection tools. For evaluation, the same sample was sequenced twice with both low and high coverage. For low coverage (5.7x), MELT has the highest True positives indicating high precision. ITIS has the lowest false positive indicating high recall rate, but it failed to predict more than 800 true insertions. MELT is also faster than rest of the tools in terms of both CPU and Wall-clock time. The only model in this analysis that uses split-reads exclusively, T-lex2, did not even finish running within a week. Therefore, researchers tend to favor of discordant-read-based models as they are faster compared to split-read models. Tools that use only discordant reads, are not good at predicting the exact location of TE insertions. However, MELT was good in finding the exact location by using split reads in combination with discordant ones. In the case of high genomic coverage (95.6x), MELT identified only a few TEis compared to RetroSeq and Mobster, which was similar to TEMP. Even though RetroSeq and Mobster had



Data	Tool	PolyTE detection performance							Runtime parameters			
		Total predictions[a]	Correct prediction			TP[e]	FP[f]	FN[g]	CPU time[h]	Wall time[i]	Peak RAM[j]	% CPU[k]
			Exact[b]	≤100 bp[c]	≤1kb[d]							
NA12878 Low	MELT	1189	853	862	862	862	327	31	13.5	18.6	19.07	111
	Mobster	1035	39	651	678	651	384	242	18.3	65.2	101.43	76
	RetroSeq	749	5	408	515	408	341	485	96.6	83	0.39	121
	TEMP	4928	0	31	45	31	4897	862	36	42.9	2.19	97
	Tangram	3186	172	411	413	411	2775	482	384.8	123.8	98.3	322
	ITIS	237	37	77	184	77	160	816	2316.20	689.9	28.67	347
	T-lex2	Process killed. Reason: Process not finished within a week							>1 week	–	–	–
NA12878 High	MELT	179	45	47	47	47	132	846	232.3	360.6	80.12	92
	Mobster	1572	303	819	825	819	753	74	449.5	426	118.55	156
	RetroSeq	4404	21	850	859	850	3554	43	1889.40	1653.50	1.67	124
	TEMP	1109	2	49	87	49	1060	844	948.9	1187.10	150.14	92
	Tangram	Process killed. Reason: Exited with error message. Reported problem.							6611.20	3128.90	261.61	221
Reported	MELT	990	807	807	807	807	183	86	–	–	–	–
	Mobster	1250	352	800	805	800	450	93	–	–	–	–
	RetroSeq	1252	18	791	799	791	461	102	–	–	–	–
	Tangram	1553	250	828	837	828	725	65	–	–	–	–
Sim5×	MELT	304	22	264	294	264	40	628	6.1	16.8	13.98	95
	Mobster	322	4	271	300	271	51	621	11.31	14.3	10.42	120
	RetroSeq	662	3	348	631	348	314	544	42.45	35.32	0.39	124
	ITIS	66	0	23	62	23	43	870	2621.10	1057.90	57.14	261
	TEMP	No predictions							2.37	2.47	2.04	98
	Tangram	Process killed. Reason: Exited with error message. Reported problem.							180.4	71.58	51.02	256
Sim10×	MELT	416	35	396	402	396	20	496	11.45	12.85	14.92	122
	Mobster	505	7	406	439	406	99	486	17.89	18.71	10.43	110
	RetroSeq	769	5	434	730	434	335	458	96.05	78.35	0.39	126
	ITIS	172	0	35	160	35	137	857	4247.16	1248.63	57.14	352
	TEMP	No predictions							5.06	5.26	2.04	99
	Tangram	Process killed. Reason: Exited with error message. Reported problem.							343.34	143.78	102.04	246
Sim15×	MELT	484	51	460	467	460	24	432	16.84	20.72	12.97	118
	Mobster	570	9	460	493	460	110	432	26.36	39.33	10.53	124
	RetroSeq	734	11	489	734	489	245	403	113.5	92.08	0.39	126
	ITIS	256	0	42	241	42	214	850	6985.06	1937.70	57.14	372
	TEMP	No predictions							6.54	6.67	2.04	100
Sim30×	MELT	542	67	509	520	509	33	383	28.55	27.75	12.05	112
	Mobster	439	16	405	413	405	34	487	44.89	35.03	10.52	134
	RetroSeq	804	14	507	738	507	297	385	260.82	216.38	0.39	123
	ITIS	399	0	49	352	49	350	843	13 286.89	3185.67	57.14	428
	TEMP	No predictions							12.19	12.45	2.04	100
Sim50×	MELT	562	68	527	539	527	35	365	45.42	52.14	41.66	102
	Mobster	593	14	505	515	505	88	387	60.78	69.95	34.05	107
	RetroSeq	828	19	489	742	489	339	403	398.71	323.13	0.39	126

[a]Total number of predicted polyTE insertions.
[b]Number of polyTE predictions with the correct insertion position.
[c,d]Number of polyTE predictions with the correct insertion position <100 bp[c] or 1 kb[d] away from predicted position.
[e]TP = Number of predictions with correct insertion position <100 bp away.
[f]FP = Number of incorrectly predicted polyTE insertions.
[g]FN = Number of polyTE insertions not predicted by the tool.
[h]Time (in minutes) the tool spent on the processor.
[i]Wall clock time (in minutes) that the tool took to finish.
[j]Maximum amount of memory that the tool occupied during its execution time.
[k]Percentage of the CPUs that the tool used.

Figure 2.6: Benchmarking and validation results done by (Rishishwar et al., 2016) for 7 existing TEi detection tools. Among these, RetroSeq, TEMP are discordant-read-based models, T-lex2 is split-read-based model. MELT, Tangram, Mobster, ITIS use discordant reads for initial prediction and split reads for some later validation.

high true positive rates, they also reported many false positives too. One tool, Tangram did not finish running for this high coverage dataset.

For the simulated data, ranging coverage from 5x to 50x, MELT performed consistently better than than other methods. With the increase in coverage, its true positve rate increased, and false negative rate decreased. For all datasets, MELT was the fastest. MELT was also better in predicting the exact insertion location. Other than the memory usage, which is quite low for RetroSeq, MELT in general outperformed all other tools.

2.7 MultiString Burrow Wheeler Transform, msBWT

The TEi location approach that I propose employs a specific sequence indexing data structure that is widely used in genomics call the Burrows-Wheeler Transform. The Burrows-Wheeler Transform (BWT) (Burrows and Wheeler, 1994) was originally developed to compress text data. The transformation effectively determines the sequence of predecessors of every sorted suffixes from a text, and it is, thus, a permutation of the original text. This BWT permutation can also be inverted to reconstruct the original text. The BWT permutation also leads to long runs of repeated characters in any text with redundancy. Therefore, simple compression techniques, like run-length encoding, can be used to compress the BWT of any compressible text. Beyond compressing data, the BWT has also been shown to be as efficient as a suffix-tree for performing substring searches. BWT searches use a light-weight auxiliary data structure called an FM-index (Ferragina et al., 2004). It is light-weight in the sense that it can be computed on the fly while accessing the BWT. The FM-index supports searching for all substrings of length k (k-mer) within a string in $O(k)$ time. The classic BWT was devised to compress a single string or text. It has since been extended to support collections of strings while maintaining the essential properties of the data structure. A BWT of a string collection is called a multi-string BWT. Holt (Holt and McMillan, 2014) showed that assembling BWTs for multiple strings can be done incrementally by merging msBWTs, and is thus suitable for parallelizing and can be used for divide-and-conquer approaches. Thus, msBWT is a great way to store large scale text data like short reads and it supports searching for common substrings shared by multiple reads efficiently.

2.8 Longest Common Prefix, LCP

The Longest Common Prefix, LCP is a second auxiliary data structure (Kasai et al., 2001) that can be built while constructing an msBWT. While an msBWT's FM-index allows for the traversal of all suffixes from each of its included strings, which we will refer to as reads, the LCP provides the length of the Longest Common Prefix between two adjacent suffixes in the suffix array implicitly represented by the msBWT. LCP and msBWT have a one-to-one correspondence with each other. That means i^{th} element of the LCP array stores the length of prefix shared by i^{th} and $(i+1)^{th}$ read suffix's predecessor symbol of the msBWT. There are many algorithms for creating msBWT and LCP in linear time (Egidi et al., 2019) (Bonizzoni et al., 2021). We used Holt's approach (Holt and McMillan, 2014) for building the msBWT and LCP, because of the supporting functions offered by the supplied API. The relationship between the BWT and LCP data structures and their use for finding repeat intervals is illustrated in figure 2.7.

2.9 Summary

Previous methods for identifying transposable element insertions have primarily relied on one of two methods discordant-mates or split-reads. Both methods require some form of alignment, either to a full genome for discordant mates or to a library of expected TE models for split-reads. Split-read models are solely template-based. Alignments with respect to TE templates do not serve any other purpose, and thus are often deleted once TE insertions are detected. None of the split-read model (RelocaTE2 (Chen et al., 2017), T-lex2 (Fiston-Lavier et al., 2014)) use any special data structure to process the reads to make it reusable for future. Therefore, this type of model is an unpopular choice for detecting TEis even though they tend to have higher true positive rates. In contrast, discordant-read TEi discovery models are quite popular among

Text: GGGCATGGGGTAGGGGCAT$

Index	**All permutations of GGGCATGGGGTAGGGGCAT$**	**sorted permutations (suffix array)**	**BWT**	**LCP**
0	GGGCATGGGGTAGGGGCAT$	$GGGCATGGGGTAGGGGCAT	T	0
1	$GGGCATGGGGTAGGGGCAT	AGGGGCAT$GGGCATGGGGT	T	1
2	T$GGGCATGGGGTAGGGGCA	AT$GGGCATGGGGTAGGGGC	C	2
3	AT$GGGCATGGGGTAGGGGC	ATGGGGTAGGGGCAT$GGGC	C	0
4	CAT$GGGCATGGGGTAGGGG	CAT$GGGCATGGGGTAGGGG	G	3
5	GCAT$GGGCATGGGGTAGGG	CATGGGGTAGGGGCAT$GGG	G	0
6	GGCAT$GGGCATGGGGTAGG	GCAT$GGGCATGGGGTAGGG	G	4
7	GGGCAT$GGGCATGGGGTAG	GCATGGGGTAGGGGCAT$GG	G	1
8	GGGGCAT$GGGCATGGGGTA	GGCAT$GGGCATGGGGTAGG	G	5
9	AGGGGCAT$GGGCATGGGGT	GGCATGGGGTAGGGGCAT$G	G	2
10	TAGGGGCAT$GGGCATGGGG	GGGCAT$GGGCATGGGGTAG	G	6
11	GTAGGGGCAT$GGGCATGGG	GGGCATGGGGTAGGGGCAT$	$	3
12	GGTAGGGGCAT$GGGCATGG	GGGGCAT$GGGCATGGGGTA	A	4
13	GGGTAGGGGCAT$GGGCATG	GGGGTAGGGGCAT$GGGCAT	T	3
14	GGGGTAGGGGCAT$GGGCAT	GGGTAGGGGCAT$GGGCATG	G	2
15	TGGGGTAGGGGCAT$GGGCA	GGTAGGGGCAT$GGGCATGG	G	1
16	ATGGGGTAGGGGCAT$GGGC	GTAGGGGCAT$GGGCATGGG	G	0
17	CATGGGGTAGGGGCAT$GGG	T$GGGCATGGGGTAGGGGCA	A	1
18	GCATGGGGTAGGGGCAT$GG	TAGGGGCAT$GGGCATGGGG	G	1
19	GGCATGGGGTAGGGGCAT$G	TGGGGTAGGGGCAT$GGGCA	A	0

Figure 2.7: An example illustrating relationships between the LCP, Suffix Array, and BWT shown for the sequence GGGCATGGGGTAGGGGCAT. Only the BWT and LCP are used to identify Frontier candidates, whereas the suffix array is implicit. Intervals of the suffix array where adjacent suffixes share a prefix of 3 or more bases include [4,6], [6,8], and [10,15]. These represent the substrings CAT, GCA, and GGG respectively. These intervals are determined in a single linear scan of the LCP. Of these, and assuming a repeat threshold of 4, only the interval [10,15] is sufficiently large to be considered a repetitive element. A subsequent rescan of this interval identifies four distinct prefixes of length 6 that appear two or fewer times (GGGCAT, GGGGCA, GGGGTA, and GGGTAG). These prefixes are easily recovered using the BWT. The memory requirements of the LCP and BWT are proportional to, and typically smaller than, the original sequence.

the biologists. The alignment required for identifying discordant reads are often used for other downstream analysis and and stored in a special alignment file called BAM. One discordant-read-based model, RetroSeq (Keane et al., 2012) was used to find landscape of TE insertions in 16 different types of mouse (Nellåker et al., 2012). MELT (Gardner et al., 2017) was used to find TEis in chimpanzee, ancient Neanderthal and Denisovan genomes, cancer genomes, and canines. However, discordant-read based methods have lower true positive rates than split-read-models. Moreover, their inability to predict the exact location may complicate the inference of TEi sharing patterns necessary for understanding the phylogeny of such events. Therefore, there is a need for TEi discovery and mapping pipelines that are highly sensitive like split-read-based models and efficient and convenient like discordant-read-based models. In the next two chapters (chapter 3 and 4), I provide new approaches and solutions to these problems. In chapter 5 I show how my proposed methods work effectively to analyze TEi diversity and sharing patterns in a large population.

CHAPTER 3: EFFICIENTLY LOCATING INSERTIONS OF TRANSPOSABLE ELEMENTS (ELITE)

A goal of this dissertation is to provide algorithms and methods capable of identifying complete catalogues of the transposable element (TE) insertions in a given set of genomes. Towards solving this problem, I describe and implement a tool called ELITE, which relies on a known TE templates to identify TE insertions within short-read sequence data sets that are within a user specified error tolerance of the given template (specified by edit-distance). Unlike typical alignment-based methods, ELITE is based on a *local-genome-assembly* approach. It uses an msBWT-based data structure to store and index all the reads from a high-throughput sequencing dataset and leverages a sampled FM-index to detect TEis efficiently. It starts with a highly conserved seed k-mer from a known TE template and uses the msBWT and FM-index to assemble genomic sequence around the seed. Assembled sequences which extend beyond the TE boundary, called the genomic *context* are then clipped and used to map the insertion location. In addition to finding TEis, ELITE also predicts the zygosity status of each insertion, and it also discovers unannotated TEs that are distantly related to the given target TE. Most importantly ELITE provides a summary set of TEi present and absent probes that can be used to identify segregating TEi patterns within a population.

3.1 Challenges and Motivation

Detecting TEis in high-throughput short reads is a challenge. Short reads are usually between 100-150 basepairs long, while TE sequences can be as long as several thousands of basepairs. Moreover, similar TE sequences from past insertions can appear thousands of times in the genome. Besides, many TEs have Long Terminal Repeats, where themselves repeats, that are

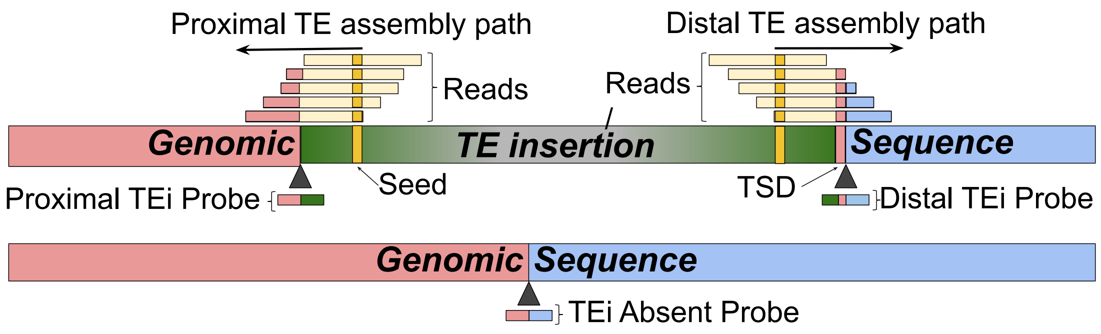


Figure 3.1: An overview of ELITE: ELITE first finds two highly conserved seeds from any annotated TE sequence. One near the proximal boundary, and the other near the distal boundary (shown in gold). It assembles sequence paths around the seeds from unaligned HTS read data using an FM-index. TE annotation is used to guide the path traversal in a depth-first search to estimate the TEi boundaries. The overhangs of the assembled sequences are used to map the TEi location. When possible, nearby insertions are merged informed by the sharing of a target-site duplication (TSD) sequence. Finally, up to three probes are constructed for use in testing other samples for a similar polymorphic TEi.

longer than a read's length. Thus, it is difficult to identify such TEs using only short-reads. Moreover, TE sequences are repeated in the genome many times. Due to the copy-paste mechanism of RNA transposon, all eukaryotic genomes contain numerous copies of the same TE. Therefore, same TE sequence can be found throughout the genome, but in different places. It is hard to identify the insertion context from all the different versions of TEs.

Various methods and tools have been developed, but all depend on specific knowledge and assumptions about either the TE's or the genome's sequences. Some tools aim to find the inserted sequence of the TE, and others are designed to localize TEs where the TE sequence is already known. One commonly used tool for identifying TEs is RepeatMasker (Smit et al., 1996). It masks all the repetitive sequences in the genome and provides a potential list of sequence referring to transposable element. Other databases such as Repbase (Bao et al., 2015) and dfam (Wheeler et al., 2012) also annotate TEs that are specific to a particular species. Many existing tools including ELITE, rely on TE sequence from Repbase as a template in order to detect and localize them in a given sample.

Others have developed methods that attempt to detect and map TEi sites in the genome. Most rely on a technique called alignment, which is applied to map reads to a reference genome or a

TE catalog (see Chapter 2). Some, RetroSeq(Keane et al., 2012), TEMP(Zhuang et al., 2014), MELT(Gardner et al., 2017) align all the short reads to a reference genome and identify discordant read pairs. These read pairs are those where one read is maps uniquely and the corresponding mate maps to multiple places in the reference. The non-unique read is then compared with a known library of TEs. This supports identifying insertions that are present in a sample but not in the reference. Detecting TEis by resolving discordant reads depend heavily on the read quality of the data and accuracy of the alignment method. Moreover, methods based on discordant read pairs are unable to report exact insertion site without any additional steps. As reported by (Rishishwar et al., 2016) the best TEi detection tools report a considerable number of false-positive results. The best performing algorithms have precision rates ranging from to 63% to 95% for simulated data, and between 25% to 73% for real data.

A second TE detection detection approach aligns short reads to a catalog of consensus TE sequences. This is considerably faster than alignment to a reference, but such approaches typically employ lower throughput aligners (i.e. BLAST(Altschul et al., 1990) or BLAT(Kent, 2002)) that allow for more mismatches than typically used for genome alignment(Ewing, 2015). These TEi detection tools attempt to find all the reads that contain some TE-like sequence. All these reads are then “split” to two parts: a non TE segment and a TE segment. The non-TE part of these split reads are then clustered, combined, and mapped to a reference genome to identify the insertion site. ITIS(Jiang et al., 2015) uses BWA(Li and Durbin, 2009), and RelocaTE2(Chen et al., 2017) uses BLAT to find sequence similarities between each read and the given TE. Split reads are then identified to pinpoint the insertion site. These methods are able to report the exact location of a TEi along with the TSD accurately by looking at the split reads from both sides of the insertion. Finding the exact location and nearby sequence is important to create primers for PCR verification.

TE detection tools based on alignments to a TE catalog have limitations too. ITIS looks for one class of TE at a time, whereas RelocaTE2 is designed to search for multiple TEs in the single run. It’s impractical to use RelocaTE2’s parallel search approach when an incremental search approach is required. Moreover, Repbase (Bao et al., 2015), dfam (Wheeler et al., 2012) or any other TE databases are constantly adding new TEs to their repositories. To find these, RelocaTE2 must repeat all the steps of it's alignment pipeline.

To avoid all these problems, I index whole-genome high-throughput sequence data using a msBWT (Bauer et al., 2011) (Cox et al., 2012) (Holt and McMillan, 2014). In general, msBWTs of short-read datasets are useful for other genomic applications. An msBWT compresses the data of raw fastq files while supporting efficient searches. Many researchers have used BWTs as an underlying data structure to perform local assembly (Simpson and Durbin, 2010) (Salikhov et al., 2013). Others have leveraged it for correcting errors in reads (Greenstein et al., 2015) (Wang et al., 2018). Mentaci et al. developed an extended version of BWT to distinguish between two given genomes (Mantaci et al., 2007). It is important to note that, the cost of building BWT is fixed, it only depends on the sequencing dataset. Once built, an msBWT is unaffected by different versions of genome builds like genome alignments of the same read set are.

Using an msBWT as the primary indexing data structure, I have developed an efficient pipeline, called ELITE, for identifying, mapping and characterizing Transposable Elements in the genome. Unlike previous alignment-based approaches, ELITE uses a targeted *local-genome-assembly-based* method. ELITE uses a branch-and-bound Depth-First-Search (DFS) algorithm for the assembly, which efficiently searches the entire read set of a high-throughput sequencing dataset represented as a multi-string Burrows-Wheeler Transform (msBWT) (Holt and McMillan, 2014), using an FM-index (Ferragina and Manzini, 2000). ELITE also finds other TEis that are more distantly related but share some conserved TE kmers. Allowing for this divergence enables ELITE to discover new TE families, that may not been annotated in any standard TE database (Bao et al., 2015) (Wheeler et al., 2012). ELITE also reports a summary of TE sharing within a given set of samples, which includes TEis that are present in all samples, TEis that are shared by only a subset of samples, and TEis that are unique to a specific sample. This is helpful for ascertaining phylogeny as well as treating TEis as genetic variants that may cause phenotypic differences.

3.2 ELITE Pipeline

Given a template TE, and a sample, I aim to find all the locations of that and similar TE classes in that sample. As a preprocess, I construct an msBWT of a whole-genome high-throughput sequencing dataset. An msBWT sorts all the reads alphabetically and then assigns an index to each read. An auxiliary data structure called FM-index is built on the fly (as a sideeffect of loading the mSBWT). Searching for any particular k-mer is done incrementally in reverse or suffix order. For example: to search ACT, it will first find all the read indices containing Ts, then CTs, and finally ACTs. I use this backward search approach to assemble TE sequences from an interior seed towards the TE's boundary. Searching is done for the two sides of a TE, which I call *proximal* and *distal* ends. The assembled sequence beyond the TE boundary, called the *context*, is then mapped to a reference genome to find the location of insertion. This TEi discovery phase for the proximal side is illustrated in the figure3.3. The execution order of the steps goes from right to left. After the discovery phase, several additional steps are taken which involves merging proximal and distal TEis, identifying their zygosity and creating different probes to assess TEi patterns in a population. All the steps are described in detail in the following subsections.

3.2.1 Choosing Seeds for Local Assembly

I use a seed-based search approach to perform local assembly where the seed is a highly repeated substring near the boundary of a known TE model. Multiple seeds can be used to allow some mutations in the seed itself. All the seeds, however, need to be within a certain distance from the TE's boundary because ELITE needs to have enough bases beyond the boundary to infer the context of insertion. To ensure this, I only consider the $kmer$ as seed whose distance from TE boundary is no more than half of a read's length. Among all these, I recommend a seed that occurs most frequently in the sample's genome. I identify two seeds, i.e., proximal and distal ends of the TE's sequence, where each one is closer to its respective boundary. Users are allowed, however, to provide their own preferred seeds. In that case too, two seeds are required for the two

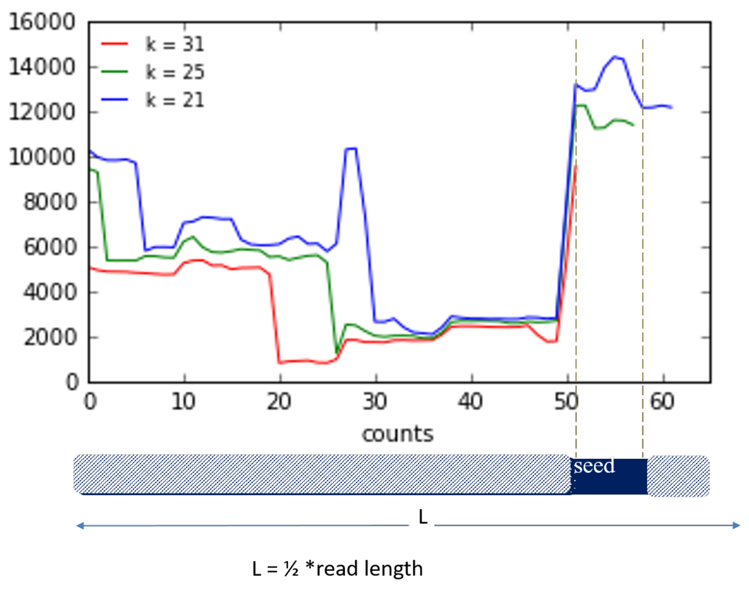


Figure 3.2: Criteria for choosing seeds from the TE boundaries: From the given TE template, I select a subsequence that is the first L bases from the TE as proximal TE. I then identify the kmer that occurs most in the target sample's msBWT.

boundaries. Figure 3.2 shows an example of choosing seed for the proximal TE boundary. Here I consider 3 different lengths for seed k-mers, where $k = 31, 25, 21$ represent the lengths. At first I choose a proximal TE which is of length L where $L = 1/2 * read\_length$. Then I divide the proximal TE into multiple overlapping segments of length k, and count their occurrence in the sample's msBWT. The kmer with the highest count is then selected as a seed. As we can see from the figure 3.2, count varies throughout the boundary. It also depends on different k. However, all of them have a peak at after the 50 bases of the proximal TE boundary. Therefore, I choose a kmer as a seed from that region shown by the dotted line.

3.2.2 Finding TEs by Assembling Seed

The first step of our TEi discovery phase finds all mutated versions of TE in a given sample. To allow for some mutations as well as the possiblity of capturing related, and perhaps unannotated, TEs, I find all the sequences that are less than a given edit distance away from the original

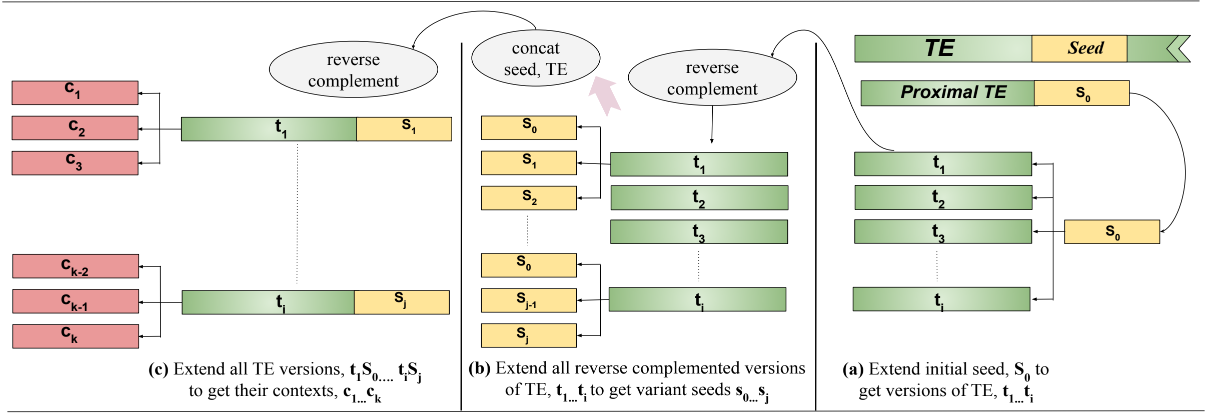

reverse complement
concat seed, TE
reverse complement
TE
Seed
Proximal TE
s_0
c_1
c_2
c_3
c_{k-2}
c_{k-1}
c_k
t_1
s_1
t_i
s_j
s_0
s_1
s_2
s_{j-1}
s_j
t_2
t_3
(c) Extend all TE versions, $t_1S_0 \ldots t_iS_j$ to get their contexts, $c_1 \ldots c_k$
(b) Extend all reverse complemented versions of TE, $t_1 \ldots t_i$ to get variant seeds $s_0 \ldots s_j$
(a) Extend initial seed, S_0 to get versions of TE, $t_1 \ldots t_i$

Figure 3.3: TEi Discovery phase: We take the proximal sequence of a given TE and select a kmer located distal to the TE's start as a seed. A seed whose distance from the TE start is less than half of the read length, is of suitable length and has the highly repeated in the data set is chosen. The seed is then extended to find all the potential versions of TE (shown in green). To incorporate any variants that may appear in the seed, I then extend all of the potential TEs in the opposite direction. This is accomplished in the BWT by searching for the extension's reverse complement (shown in yellow). Finally all the versions are further extended to get all the possible prefixes, which establishes their genomic context (shown in pink).

TE. I start with a seed as an input, which is our initial assembled sequence and use a depth-first-search algorithm to extend it further towards the boundary of the TE (figure 3.3a).

Using algorithm 1, ELITE first finds the range of indices for the seed k-mer using a sampled FM-index of the compressed msBWT (Holt and McMillan, 2014)(Ferragina and Manzini, 2000), where the range represents the number of occurrences of seed substring in the set of sequenced reads (specifically their interval in an implicit suffix array). This range, along with the seed k-mer, is used to initialize the recursive DFS used by the local assembly. At each step in the recursion, the algorithm adds a new possible base before the seed and updates the suffix array range ($newRange$) concerning the newly added base and then continues along this child's path in the recursion tree.

I introduce two threshold parameters to limit the depth of recursion. First $t1$ sets the minimum number of reads that contain evidence of TE. Second, $t2$ sets the maximum edit distance that is allowed in a TE sequence with respect to the original TE template. If at any place the value of $newRange$ is less than $t1$ or the value of $dist$ is greater than $t2$ we prune that path. This

Algorithm 1: Extending seed k-mer

```
EXTEND-Kmer($range, seed, newTE$)
    if $newTE$ Reaches TE Boundary then
        Add newTE to the TE list
    end
    for base in A, C, G, T do
        $newTE \leftarrow$ base + newTE
        $newRange \leftarrow$ findIndicesOfStr(base,(range))
        $dist \leftarrow$ editDistance(TE,newTE)
                                                    ▷ t1 and t2 are threshold parameters
        if newRange $> t1$ and dist $< t2$ then
            EXTEND-Kmer($newRange, seed, newTE$)
        end
    end
```

prunes from our search sequences that are not present in the short read dataset, as well as the sequences that differ too much from the given TE template.

At this point, all the versions end with the same seed sequence (The DFS traversal is in suffix order). To allow for variants in the seed, I then remove it from all the TE versions and continue assembling each version in the reverse direction towards the seed (figure 3.3b). I then assemble in that direction until the length of the seed is reached without exceeding the edit-distance threshold $t2$. Since searching in an msBWT using an FM-index is done by extending suffixes, finding a prefix is more straightforward than finding a suffix. Thus, when extending TE sequences in this second DFS pass, where the seed is removed, I conduct the search using the TE's reverse complement sequence, thus searching for alternative seeds that prefix. It works because of DNA's double-stranded structure where one strand contains the reverse complemented sequence of the other to form a double helix. Now to get back to the original strand, I again reverse complement the TE versions. I repeat the same assembly process for the distal side of a TE in the opposite direction and adjust the strand so that the traversal is always in suffix order.

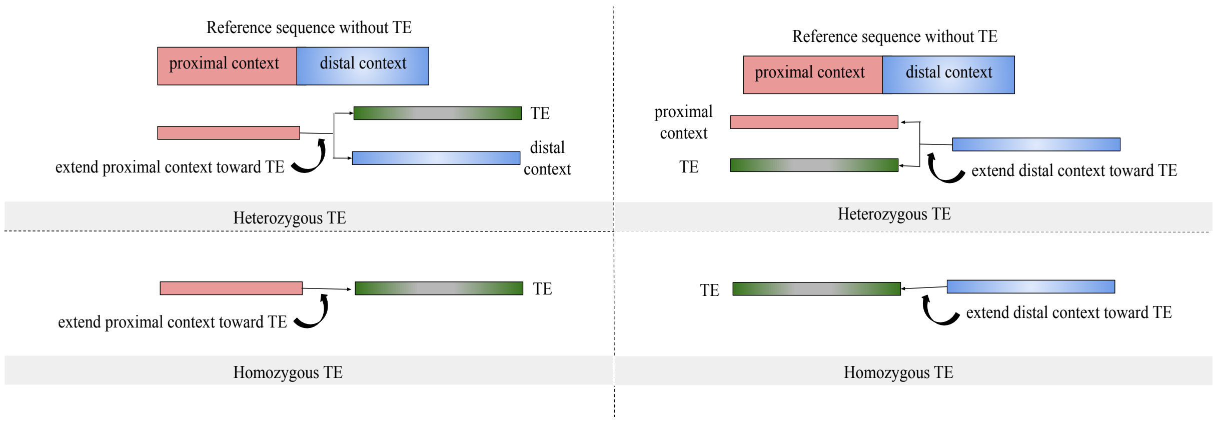
3.2.3 Mapping TE Insertion Sites

After finding all the versions of a TE, I then attempt to place them in the genome. I assemble the genomic context by extending the prefix and suffix of each discovered proximal and TE-like sequence, respectively, to find the corresponding genomic context (figure 3.3c). I use the same algorithm described in the previous subsection, but initialized with the suffix array range of each TE version found by the $ExtendKmer$ routine. I set the edit distance threshold $t2$ to a large value and the support threshold, $t1$ to zero to allow the DFS to extend until it runs out of reads containing the given TE version.

ELITE next maps all of the contexts sequences found (the additional bases fond by extending the TE versions) against the reference genome using bowtie2 alignment tool (Langmead and Salzberg, 2012)(Langmead et al., 2009) to get the chromosome and position. In this phase, ELITE only keeps the contexts that are uniquely mapped. Finally, it examines the reference sequence adjacent to the alignment position to assess whether it differs or is a close match (as determined by $t2$) to the TE sequence used to find the context originally. If the reference sequence adjacent to the mapped TE differs from the context-TE sequence combination, I consider the mapping a non-reference TEi.

3.2.4 Determining Zygosity

As a third step, for any non-reference TEi, ELITE determines if it is present in the homozygous or heterozygous state. To find the zygosity, we extend the mapped context sequence found in step 2 in the direction of TE, which differs depending on whether it was discovered from the proximal or distal side of the TE template. If I find sequence paths, one matching the expected TE, and a second similar to the reference sequence (as determined by the edit-distance parameter, $t2$) then I report the TEi as heterozygous. But, if the extended sequence only leads to the expected TE-like sequence, then we report it as homozygous. Usually, heterozygous TEis are indications of recent TE activity, although a similar pattern can be created by a duplication fol-



Reference sequence without TE
proximal context
distal context
TE
distal context
extend proximal context toward TE
Heterozygous TE
Reference sequence without TE
proximal context
distal context
proximal context
TE
extend distal context toward TE
Heterozygous TE
TE
extend proximal context toward TE
Homozygous TE
TE
extend distal context toward TE
Homozygous TE

Figure 3.4: Finding zygosity of a TE insertion

lowed by a TE insertion. This alternative explanation requires additional tests to validate and to distinguish. Such tests which are beyond the scope of this discussion.

Figure 3.4 shows the process of finding the zygosity of a TE insertion. Recall that, I get two probes from the discovery phase: proximal context + TE, and distal TE + context. By extending these contexts (excluding the TE part), I further determine the zygosity. For the proximal context (shown if left), I extend the context towards right which is direction to the proximal TE sequence. And for the distal context (shown if right), I extend the context to the left which is the direction of the distal TE sequence. In both cases, this extension must lead to at least one path which is the TE we already discover. However, it may lead to another path that is similar to the reference sequence (proximal context + distal context) given that reference does not have an insertion at that point. Here two paths from the extended contexts refers to a heterozygous insertion, and one path refers to a homozygous insertion.

ELITE predicts zygosity once a TEi is found. The method of finding TEi, even though is same for both homozygous or heterozygous insertion, coverage may affect the algorithm's outcome. For heterozygous insertion, the number of paths containing the context+TE sequence will goes down to almost half compared to a homozygous one. Therefore, to find the heterozygous TEis, like most existing algorithms, ELITE also requires short read dataset with good coverage.

3.2.5 Merging Proximal and Distal TEis

ELITE independently considers the two sides of a TE during the first three assembly steps. In the next stage ELITE merges any non-reference TEi if there exists a pair of proximal and distal contexts that map to the same genomic position after adjusting for a TSD. ELITE also attempts to find the other side of non-reference TEis when a mapped TEi is discovered only from one side. In this case ELITE uses the genomic sequence adjacent to the context on the side of the detected TEi from the reference to once again guide a DFS to find any TE-like context adjacent to it. If ELITE finds a reference-like context flanked by the expected TSD followed by any sequence that is a significant edit distance from the originating context, the TEi is considered merged. For merging TEs in the reference, I find all the proximal-distal context pairs that are near the length of the TE plus or minus a gap parameter ($\leq$ 10bp). If a consistent TSD is observed, I merge the proximal-distal context pair. ELITE keeps all one-sided TEis if there is enough reads to support the insertion. TEis can be one-sided in one of two ways; if the context on the other side is unmappable due to being inserted into a genomic repeat, or if some subsequent insertion or deletion modified the missing context at the other side of the TEi.

3.2.6 Assessing TEi patterns in a population

A final optional feature of ELITE is that, once a TEi is identified and mapped, several targeted sequence probes (up to three) are generated to accelerate the testing of subsequent samples. When a TEi is discovered, one or more of three probe types are created (see Figure 3.1). The first is a TEi specific *proximal* probe for finding split-reads that contain the normal genomic context and the adjacent TE sequence. ELITE's proximal probes are constructed relative to the TE sequence's orientation, not the reference genome's orientation. The second *distal* probe includes genomic sequence at the other end of the insertion preceded by a TSD sequence and the distal TE sequence. The third *absence* probe represents the expected genomic sequence without the TEi. Ideally, corresponding proximal and distal probes are derived from the TE-like sequence found during the discovery phase. Absence probes are derived from reference genome and/or all the samples, which has some nonTE-like sequence in the same chromosome and position. The presence or absence of a TEi is confirmed by querying the TEprobe (*proximal* and *distal*) and *absence* probe, respectively. However, having both makes a sample heterozygous in that site. Currently no other TEi discover method attempts to construct absence probes even though they are very useful for population based TEi sharing analysis. Lets consider a case, where an algorithm found a TEi in every sample in a population except for one. Then it will just be automatically assumed that TEi is absent in that one particular sample. However, it's possible that the sample has a TEi which the algorithm fails to find. Therefore, its important to know what the sample has in that location if not the TEi. The presence of a nonTE-like sequence in that case can be used to prove that the algorithm's finding is indeed correct. And if the algorithm is wrong, then it can be fixed in this step. Typical methods do not attempt to find absence probe thus gives wrong idea when there's a false negative. To remove this limitation, in addition to finding TE presence probe, I also look for the absence probe for each TEi in a population.

3.3 Evaluating ELITE Pipeline

I applied ELITE to six short-read sequencing data sets from six mouse samples. These six samples were drawn from two common laboratory mouse strains. Having replicates of each strain allowed us to consider both consistency and differences within and between the two strains. The expectations are that inferences from the same strain should be mostly consistent. There should be few differences between samples from the same strain, so called biological replicates, and these differences would more likely be found in a heterozygous state. Differences in TEis that are consistent within strains but differ between strains and that are also homozygous suggests segregating TEis. I examined ELITEs TEi predictions in the context of these expected sharing patterns. I also show how this process could be applied to larger population to understand mouse phylogeny and population genetics. Sharing within strains, in addition to simulated read data, is used to estimate ELITE's error rate and error types. To further examine ELITE's error rate I also compared it to other independent resources for validation, including other TE detection tools, and the most recent genome annotations available. I report on the performance of ELITE applied to a synthetic dataaset where the truth is known. We use both discordant-read-based model and split-read-based models for a fair comparison. Several of ELITE's predictions were also validated at the bench using standard Polymerase Chain Reaction (PCR) based methods. Since PCR sequencing is expensive, I chose 9 insertions that were biologically more interesting.

Type	Strain	Sample	Number of reads	Read length	Size of msBWT
B6	C57BL/6J	m03636A	1,045,633,612	125/151	23,303,735,734
B6	B6N-Tyr^{c-Brd}/BrdCrCrl	m001	417,644,074	150	10,064,032,516
AJ	A/JCr	f001	741,355,602	150	16,094,491,291
AJ	A/JOlaHsd	m001	337,606,538	150	8,506,263,007
AJ	A/JOlaHsd	f015	556,997,844	150	10,928,292,735
AJ	A/J	f321	634,232,014	150	12,321,190,427

Table 3.1: Six real samples are used for evaluating ELITE. These are categorized into two types (B6 and AJ) based on their similarities in strain. All these were sequenced using either an Illumina X10 and Illumina HiSeq4000 sequencers with paired-end reads. One sample, C57BL/6J(m03636A) incorporates multiple sequencing runs and is a mix of read lengths, 125-bp to 150-bp, all others resulted from a single run (with multiple lanes) and used a uniform 150-bp read length. The Illumina sequencing data from each sample was used to construct an msBWT for each sample as described by (Holt and McMillan, 2014). Number of reads per sample and size of each msBWT (in byte) are also shown for each sample.

3.3.1 TE discovery in real data

Laboratory inbred mouse strains are widely used in biomedical research as their genomes are assumed to be fixed and reproducible. I examined six such mouse strains using ELITE to determine the variability of 6 Endogenous Retro Virus (ERV) TE types between them. The most commonly used mouse strain, C57BL/6J, is the basis for the mouse reference genome (GRCm38.68). It is also the primary source of existing TEi annotations (Smit et al., 1996). Thus, I ran ELITE on a C57BL/6J sample and a second related strain, B6N-Tyr^{c-Brd}/BrdCrCrl, to assess the degree of TE activity relative to the reference. I refer these two related samples as the B6 type.

TE	Length	Seeds
ERVB4_2B-LTR_MM	509	ATTGTCCTGATCTCTGAATTGGGCCTCTCCC TTTGGTTCTTAGGAGAAGGTCCCCTCGAGAC
ERVB7_1-LTR_MM	319	GTTCTGCCACGCCCACTGCTG CCCCGTCTAGATTCCTCTCTTACAGCTCGAG
IAPEY3C_LTR	304	TAACTGGTAAACAAGTAATGT AATAAACGTGTGCAGAAGGAT
MERVL_LTR	493	ATATTTGGAATGAACTACAAT ACTGTAAGTCATCA ATAAATT
RLTR1IAP_MM	351	CTGTGTTCTAAGTGGTAAACAAA TAATCTGC GAAGATTCTGGTCTGTGGTGTTCTTCCTGGC
RLTRETN_MM	322	ATTGTCCTGATCTCTGAATTGGGCCTCTCCC TTCTCTTCCAGGTTTCCAAAATGCCTTTCCA

Table 3.2: 6 different ERV templates to capture majority of the ERV insertions in 6 samples. Templates vary in lengths and sequence. Different seeds are used for each ERV, except for the proximal seed of ERVB7_1-LTR_MM and RLTRETN_MM.

Sample	TEis	Reference	Shared	Polymorphic	Private
C57BL/6J(m03636A)	11661	11519	8585	3072	4
B6N-Tyr^{c-Brd}/BrdCrCrl(m001)	11407	11264	8585	2818	4
A/JCr(f001)	11119	9086	8585	2529	5
A/JOlaHsd(m001)	10639	8673	8585	2047	7
A/JOlaHsd(f015)	11148	9099	8585	2550	13
A/J(f321)	11190	9148	8585	2591	14

Table 3.3: Total number of TEis found by ELITE in each sample for the six TE templates. A large fraction of these are also in reference. Total 8585 TEis are shared by all samples. More than 2000 TEis are polymorphic, meaning they are only present in a subset of these samples. A small fractions are private to only one sample indicating potentially recent TE movement.

I also ran ELITE on four additional samples from a second widely used lab strain, A/J. Two of the A/J samples are independent samples from the same vendor, which allows us to examine TEi pattern relative to that vendor. C57BL/6J(m03636A) incorporates multiple sequencing runs and is a mix of read lengths, 125-bp to 150-bp, all others resulted from a single run (with multiple lanes) and used a uniform 150-bp read length. The Illumina sequencing data from each sample was used to construct an msBWT for each sample as described by (Holt and McMillan, 2014).

C57BL/6J	B6N-Tyr^{c-Brd}	A/JCr	A/JOla(m001)	A/JOla(f015)	A/J(f321)	count
0	0	1	1	1	1	1789
0	0	1	0	1	1	74
0	0	0	1	1	1	1
1	1	1	1	1	1	113
1	1	1	0	1	1	8
1	0	1	1	1	1	3
0	1	1	1	1	1	2
1	1	0	1	1	1	1
0	0	0	1	1	0	45
0	0	1	0	0	1	35
1	1	0	0	0	0	10
1	1	0	1	0	0	2
1	1	0	0	0	1	1
0	0	1	1	0	0	2
0	1	1	1	0	0	2
0	0	0	0	1	1	1

Table 3.4: A comprehensive list of TEi sharing patterns for discovered TEs that do not appear in the reference sequence. The population structure of our samples allows us to cluster sharing patterns according to expectations. Expected sharing patterns are highlighted in green above. There are two primary groups in the population. One composed of the first two samples, which are ”B6-like”, and the second four samples, which are ”AJ-like”. The AJ-like samples can be further broken down into subpopulations according to vendor, which implies the possibly of sharing between the two A/JOla samples. Sharing patterns that do not match these expectations are likely indirect indicators of ELITE’s false-negative rate, which impacts recall, and its false-positive rate, which impacts precision. Possible false-negative TEis are indicated in pink. If these cells were non-zero the pattern would match its associated cluster’s sharing pattern. Possible false-positive TEis are indicated in light blue, which, if zero, would conform to an expected sharing pattern. A group of three unclustered and unexpected sharing patterns fill out the table. Combinations of multiple false-negatives and/or false-positives create these patterns.

In each sequenced sample, ELITE looked for TEis using six different TE templates i.e., ERVB7_1-LTR_MM, ERVB4_2B-LTR_MM, RLTRETN_MM, RLTR1IAP_MM, MERVL_LTR, and IAPEY3C_LTR obtained from Repbase. Separate seeds were found for each TE template as described previously. For each template, I selected two conserved kmers as seeds each of length 25 where one is from the proximal side, and the other is from the distal side. The seeds are located within 60-80 bases from the terminal end of the TE sequences. The TE templates and examples seeds are shown in table 3.2. Here I use the same naming conventions of RepBase. The proximal TE of ERVB7_1-LTR_MM and RLTRETN_MM have a high level of sequence similarities that leads us to use only one proximal seed. Using one seed for both ERV types also reduced the runtime of ELITE pipeline. Since, many TEs originate from the same type, it is also possible to identify many more classes of TEs by using a highly conserved seed. For all other ERV types, I chose one seed from the proximal TE, and one from the distal. All the samples were ran using the same seeds which are given in the table 3.2. The rows for each ERV is the proximal seed and the second row is the distal seed that we used for the B6 samples.

The edit distance threshold, $t2$ was set according to this offset to allow for no more than 1 edit per TE 8 bases, and the minimal read support threshold used was 4 for approximately $25\times$ - $30\times$ genome coverage. The number of mapped TEis per data set is shown in Table 3.3. These are broken down according to the number of TEis that are included in the reference sequence, those that are common to all six samples, or *shared*, and those that are shared by two to five samples, which I call *polymorphic*, and finally those that appear in only a single sample, which I call *private*.

Since the mouse reference genome is based on B6, I see a large fraction of TEi found in C57BL/6J and B6N-Tyr^{c-Brd}/BrdCrCrl are also present in reference. Table 3.4 breaks down the patterns of sharing detected between samples focusing only on those TEis absent from the reference genome. As expected, the single largest pattern of TEi sharing is between the four AJ samples. The second most common pattern of sharing is TEis that appear in all six samples, but do not appear in the reference. There is also significant sharing in subpopulations. In particular, between the two A/JOlaHsd samples, from a common vendor, and the A/JCr and A/J samples distributed from two separate vendors.

By analyzing the sharing patterns of TEis I also gain some insight into ELITE's error rate. The expected patterns of sharing are highlighted as green rows in Table 3.4. Many of the unexpected sharing patterns are shown clustered with their closest expected pattern and include a highlighted cell which we hypothesize is due to a specific error types. I indicate presumed

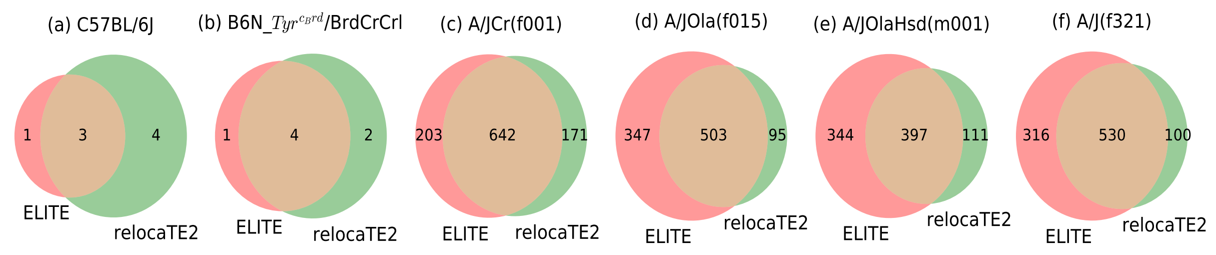


Figure 3.5: Summary of ELITE and RelocaTE2's TEi detections in real sequencing data. Sample types are shown in top and names are in parentheses. Sample name for the first two are ommited due to simplicity. For six different samples, I compare the number of TEis discovered by both tools. RelocaTE2 reports a TEi with high confidence if it is not present in reference and has a consistent TSD. Although, ELITE reports a TEi even if its only discovered from one side, but for a fair comparison, I consider here only the non-reference TEis where ELITE found both sides of a TEi along with the TSD. As we can see, for each sample, a large fraction of TEis are discovered by both tools. Since ELITE is less sensitive to the template TE sequence, it tends to find more TEis than RelocaTE2. I investigated several examples where ELITE failed to find a TEi reported by RelocaTE2. I found that unmappable context is one of the most common reasons RelocaTE2 succeeds by leveraging paired-end read information.

false negatives in red, and presumed false positives in blue. ELITE's false-negative error rate brings into question the validity of private TEi which is present only in one sample and has no sharing pattern. Thus, those TEis are best validated by external means, including PCR based experiments, and concordance with other TE detection tool, both of which I report on later. The last three sharing patterns (total 5 TEis) are likely a result of multiple errors. There are only two presumed false-negative TEis in the sample with the highest coverage, C57BL/6J. However it is the only sample with a mix of read lengths (125-bp, 151-bp), and the shorter reads may play a role in introducing that error. The single sample with the highest predicted error rate based on these anomalous sharing patterns is A/JOlaHsd(m001), and the type of error is dominated by an access of false negatives (74+8 false negatives vs 2 false positives). This is consistent with the fact that A/JOlaHsd(m001) has lower than usual coverage, which is what we believe drives the false-negative error rate of ELITE. This conjecture is tested by our experiments on simulated data sets later on.

3.3.2 Comparison of ELITE with other Split-Read-based Method

For comparison, I ran a recent transposable element detection tool, RelocaTE2 (Chen et al., 2017), on the same set of six sequenced samples. Just like ELITE, RelocaTE2 also requires TE template and finds insertions by identifying the split-reads. Recall split-reads are the ones that contains partial TE and partial non-TE sequence. RelocaTE2 is a recently published paper (PeerJ 2017), where authors showed their tool outperforms three other TE detection tools - ITIS (Jiang et al., 2015), TEMP (Zhuang et al., 2014), and RelocaTE (Robb et al., 2013) regarding precision, recall and runtime. Geneticists had already used the first version of this tool, RelocaTE (Robb et al., 2013) to assess TE activity in rice data. In addition to the good performance, this is a very well-documented tool, and its installation is fairly simple. For these reasons, I chose this split-read-based method to compare with ELITE.

In the six samples of our dataset, there is a high concordance between the TEi predictions of both tools. However, ELITE discovers a few more than RelocaTE2. RelocaTE2 fails to identify any TEi in the reference, and it misses many non-reference TEis where the TE sequence diverges too much from the template. A summary of RelocaTE2's and ELITE's TEi discoveries are depicted as Venn diagrams in Figure 3.5. Note that I am only comparing the non-reference predicted TEi's for ELITE since RelocaTE2 does not consider them. RelocaTE2 tends to perform better on samples that are B6 or samples that closely match the reference, such as C57BL/6J and B6N-Tyr^{c-Brd}/BrdCrCrl. In those samples, ELITE mostly fails in the mapping step. Mutation in any B6 sample with respect to reference is very rare. Therefore, there's rarely any variants in the part of TE's context. This helps RelocaTE2 to correctly map some TEis. However, other samples are not like B6, and they have relatively higher mutations. It's important to allow some edit distance in context too in order to locate TE in non B6 samples. That's why ELITE performed better in the 4 AJ samples.

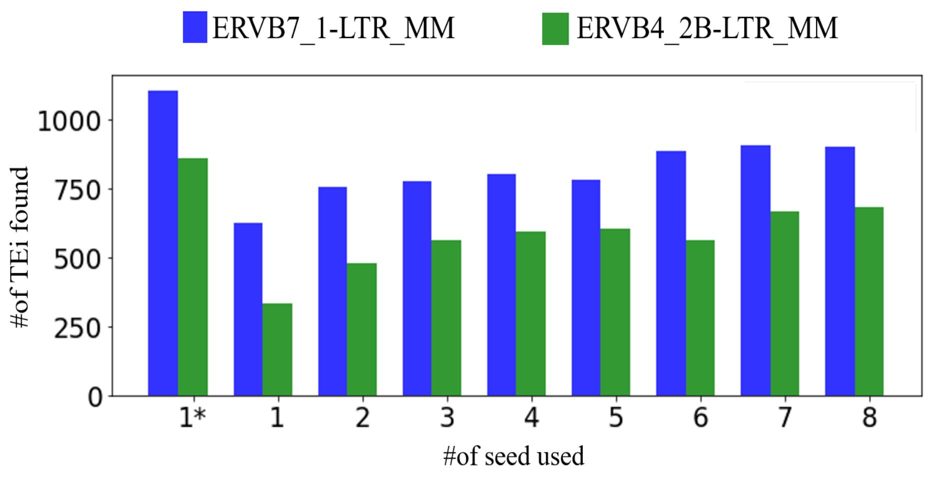


Figure 3.6: Effect of using multiple seeds: The left-most pair of bars show the number of TEi found by ELITE using the most conserved seed. I denote 1 conserved seed by 1*. The next eight pairs of bars show the number of TEi found using a different number of seeds ranging from one to eight. As we can see, increasing the number of seed results in more TEi discovery. However, a single conserved seed is often able to capture more TEis than eight random seeds. Thus I chose to use data-driven seed which I call the conserved seed.

3.3.3 Effect of Multiple Seeds

To examine the effect of using multiple seeds, in comparison to selecting a single *conserved* one I ran ELITE on A/JOla(f015) with a variable number of seeds (see Figure 3.6). For this experiment, we only looked for two TE templates, i.e., ERVB7_1-LTR_MM and ERVB4_2B-LTR_MM. I varied the number of seeds starting from one to eight. All the seeds had an equal length of 25 bases and started from the boundary of a TE template. Here I only considered the proximal TE and its seed. In case of more than one seed, all seeds were chosen to be equally distant from their neighbors. As you can see from the figure, one conserved seed is enough to find majority of the TE insertions for both of the TE templates. However, if I chose some random seeds, it is highly unlikely to find all the TEis. In fact 8 random seeds from ERVB7_1-LTR_MM find less TEis (≈ 850) than a conserved one (≈ 1000). This is also true for ERVB4_2B-LTR_MM. 8 random seeds from ERVB4_2B-LTR_MM find less TEis (≈ 750) than a conserved one (≈ 850). Therefore we recommend using a conserved seed for capturing more TEis.

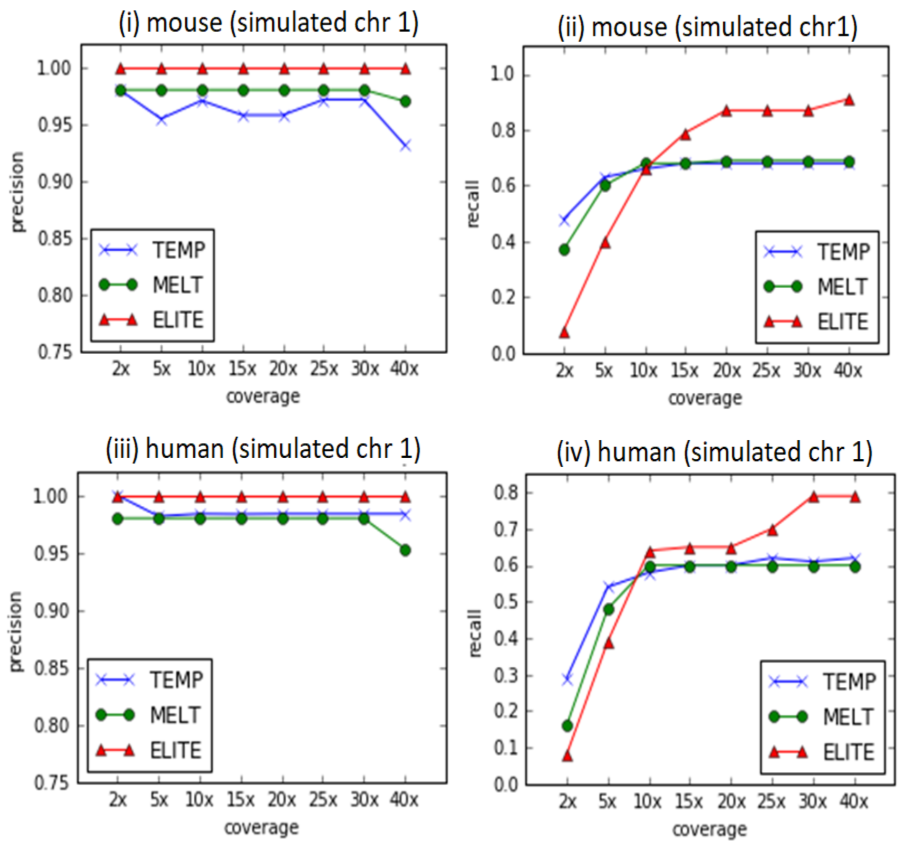
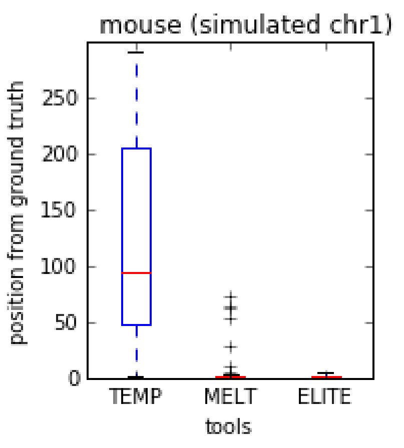


Figure 3.7: (i) and (iii): The precision rate of ELITE, MELT, and TEMP for TEi discovery in simulated mouse and human genome as a function of genomic coverage is shown in red, green and blue respectively. As we can see, ELITE did not produce a single false positive in either mouse or human genome resulting in a precision rate of 1. However, TEMP (in both human and mouse) and MELT (in mouse only) produced some false negatives when the coverage is around 40x. (ii) and (iv): The recall rate of three tools. For each tool, it increased with the increase in coverage Although, at low coverage, ELITE performed poorly compared to others, but caught up when the coverage is around 10x. At coverage higher than 10x, MELT and TEMP had many false negatives resulting in a reduced recall rate compared to ELITE.

3.3.4 Evaluation of ELITE on Simulated Data

To estimate the precision and recall of TEi discovery, I ran ELITE and two additional state-of-the-art TE detection tools, i.e., MELT and TEMP on simulated data. I inserted 100 AluY and 100 ERVB7_1-LTR_MM into chromosome 1 of human and mouse reference genome respectively. Location of these insertions is chosen randomly. Among the 100 TEis in mouse and human, 80 are inserted as homozygous and 30 are heterozygous to estimate ELITE's zygosity prediction



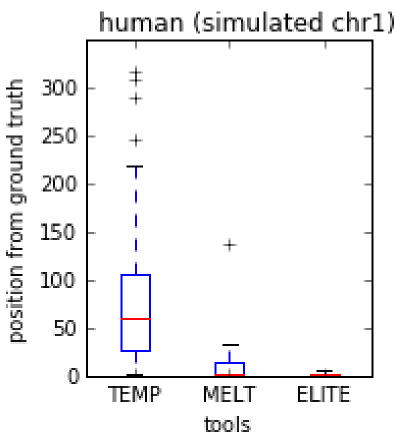


Figure 3.8: The absolute distance between predicted and actual position for each TEi at coverage 40x. On average, this distance for TEMP was around 100 and 50 bp in mouse and human respectively. MELT's predicted position was very close to the ground truth with few exceptions. However, ELITE outperformed both by finding all the TEis within 6bp resolution.

rate. I used Samtools (Li et al., 2009) to simulate 150-bp paired-end reads at different coverages ranging from 2x to 40x. For each set of simulated data, I built an msBWT index from paired-end reads as required by ELITE. Additionally, as a preprocessing step for both MELT and TEMP, I aligned all the short reads to the corresponding reference genome using BWA. I considered a TEi is found if it's within 500 bases of the ground truth location.

Figure 3.7 shows the precision and recall rate of ELITE, MELT and TEMP for different coverages ranging from 2x to 40x. From this figure, we can see that, ELITE has no false positive rate for either mouse or human genome. On the other hand, both MELT and TEMP's precision rate decreased with the increase in coverage. However, ELITE's recall rate heavily depends on the sample's coverage. With low coverage, the majority of the heterozygous insertions are missed by ELITE. However, as the coverage increases, recall rate for ELITE also increases and becomes stable when the it is around 15x. This strongly indicates that having 15x coverage to get higher recall rate is not random and using sequenced data with at least 15x coverage is enough for ELITE to identify more than 90% of the TE insertions. However, for MELT, and TEMP, we can get at most 60% of the insertions even with 40x coverage.

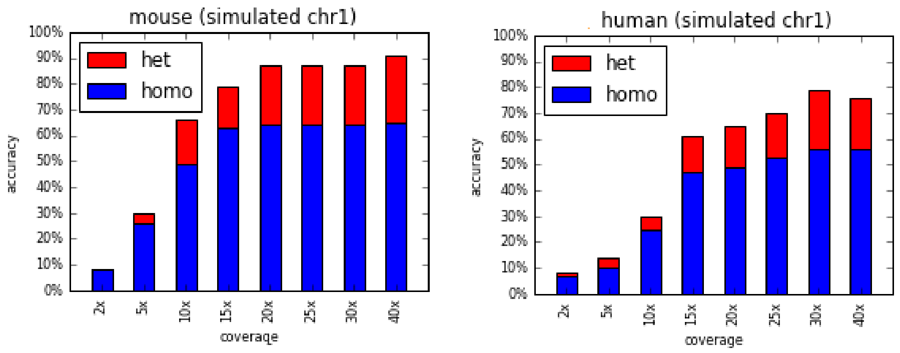


Figure 3.9: This figure shows the accuracy of zygosity prediction by ELITE in mouse and human respectively. Blue bars stand for homozygous TEi, and red bars stand for heterozygous TEi. Similar to recall, zygosity prediction accuracy also increased with the increase of coverage. At coverage 2x, ELITE failed to locate any heterozygous insertion. Other two tools do not have the zygosity prediction feature.

I also examined the distance from predicted position to the ground truth for each 3 tools. As we know TE insertion in a gene can disrupt or completely disable its functionality. It is also known to have consequences if it get inserted in a promoter region (around 500 bases before the start of a gene). That's why it is important to identify the exact location of a TE insertion to see if its around or inside e gene. Usually discordant-read-pair-based methods are not good at finding the exact location of TE insertion. However MELT has additional steps that further analyzes the discordant pairs once a TEi is found. It walks through all the discordant reads to find the exact TE boundary or insertion site. That's why MELT can predict almost accurately the location other than few exceptions. On the other hand, ELITE focuses of finding TE boundary, and always find the exact location with 100% reliability shown in figure 3.8.

Finally, I calculated ELITE's zygosity prediction rate. All these results are shown in figure3.9. Similar to recall rate, ELITE's zygosity prediction rate depends on coverage. Since heterozygous TEis have half of the coverage, it is more likely to be missclassified than a homozygous TEi. In my experiment run, ELITE never declares a homozygous TEi as heterozygous. However, at low coverage, it tends to missclassify some heterozygous TEis as homozygous.

3.3.5 Validation via PCR

In conjunction with my biolocical collaborators, I validated the presence, absence, and zygosity of the nine predicted TEis found in three of the sequenced samples for which I had available DNA (A/JCr(f001), A/JOla(f015), and B6N-Tyr^{c-Brd}/BrdCrCrl(m001)). Instead of choosing a random TE insertion for validation, I followed the following criteria to effectively measure ELITE's performance while providing information that is most useful for biologists. For example, I selected at least one TEi that is shared by everyone. I also selected one that is polymorphic in AJ and one that is polymorphic in B6. Finally, I chose some private TEis that were present only in one sample. To validate the zygosity predictions I chose four homozygous and five heterozygous TEis. Private TEis are biologically the most interesting because they indicate recent TE activity since they are absent in other closely related samples. Private TEis can also lead us to find the active copy of TE for further analysis. I also kept in mind that, insertions that are in genes are more likely to have functional consequences. Thus I selected more TEis that are private, and prioritized the ones that are within a gene. For the private TEis, I also made sure that each sample in a population either has a TE presence or a TE absence probe. In addition to these, I chose two one-sided TEis where my algorithm failed to find both proximal and distal ends. We did not design PCR assays to specifically target the predicted false positives shown in 3.4.

Assay Design: The proximal and distal genomic context probes of each TEi were used to further assemble the genomic sequence surrounding each TEi using the msBWT using the methods similar to those described in (Holt et al., 2016). For each TEi, a pair of primers flanking the TEi, one internal to TE and two within the genomic context of the TEi were designed. Two PCR products of different sizes (range 160-414 bp) can be synthesized from the primers. One using the two genomic context, and a second from the TE primer sequence and the closer of the two context primers. The presence or absence of these amplicons was used to determine the absence, presence and zygosity of the corresponding TEi predictions in a given sample. PCR reactions were imaged after running on a 2% agarose gel with ethidium bromide at 120V for 40-50 minutes.

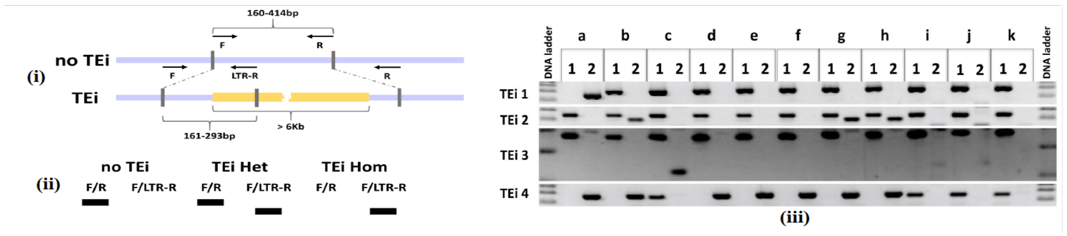


Figure 3.10: (i) Assay Design: For each TEi, we created three primer sequences i.e. forward F, reverse R, and LTR (ii) Representation of ideal DNA amplification results for each TEi zygosity: an amplification band in F/R indicates absence of TEi, a band only in F/LTR indicates presence of TEi in homozygous state, and a band both in F/R and F/LTR indicates presence of TEi in heterozygous state, (iii) PCR amplification result: For each sample, the PCR products from 1) Forward-Reverse and 2) Forward-LTR primers reactions are shown side by side. Reaction 1 detects the absence of the TEi and Reaction 2 detects the presence of the TEi. Sequenced samples: a) A/JCr(f001), b) A/JOlaHsd(f015), c) B6N-Tyr^{c-Brd}/BrdCrCrl(m001). Additional test samples: d) A/J(m93), e) A/JCrl(f002), f) A/JOlaHsd(m001), g) A/JOlaHsd(m002), h) A/JOlaHsd(f003), i) B6N-Tyr^{c-Brd}/BrdCrCrl(m001), j) C57BL/6J(m001), k) C57BL/6J(f002).TEi 1 is private and found only in A/JCr(f001) at Chromosome 2:58948253. Its presence is confirmed in that sample (a), but it does not appear in any other test sample including a second A/JCr (e). TEi 2 was found in A/JOlaHsd(f015), is also private, and was found in a heterozygous state at Chromosome 4:98626789. The assay (b) confirms this, as well as it being present and heterozygous in 2 of 3 test samples from the same vendor (g,h). TEi 3 is private and was found in B6N-Tyr^{c-Brd}/BrdCrCrl(m001) at Chromosome 1:125336107, where it is confirmed (c). It does not appear in the test sample from the same strain (i). TEi 4 was predicted to be shared among all A/J samples at Chromosome 1:28791918. It was confirmed in the two sequenced samples (a,b) and it also appears in all A/J test samples (d,e,f,g, and h). All four TEis were identified as non-reference, and they are not found in any of the reference-like test samples (j,k).

PCR results: There was perfect concordance between ELITE's predictions and PCR validation in all 27 assays (3 assays per TEi). I tested the presence, absence and zygosity for six of these TEis in eight additional unsequenced samples (A/J(m93), A/JCrl(f002), A/JOlaHsd(m001), A/JOlaHsd(m002), A/JOlaHsd(f003), B6N-Tyr^{c-Brd}/BrdCrCrl(m001), C57BL/6J(m001), and C57BL/6J_F002)(see figure 3.10). In all 48 of those PCR results suggested genotypes that are consistent with ELITE's TEi findings. In two of our assays, I confirmed the presence of a TEi discovered by ELITE, but it was not found by either MELT or TEMP. Thus, via another independent method we have confirmed 100% of 9 of ELITE predicted TEis. This supports our claims that ELITE has a low false-positive rate.

3.3.6 ELITE TEi confirmation via an AJ genome assembly

Recently a new AJ-specific full-genome assembly was announced (Lilue et al., 2018). I used this resource to test if any of the TEis predicted in any of our AJ samples, but absent in the standard mouse reference genome, were included in this *de novo* assembly. While there was no specific mention of any special assembly efforts for Transposable Elements, in a previous publication the same authors reported on TE differences between mouse strains using the same data(Keane et al., 2012). I first considered those TEis shared by all of our AJ samples. For each TEi, I queried for the context discovered by ELITE in the assembled AJ genome, and verified whether it was followed by the expected TE sequence (allowing for errors). More than 90% TEis reported in all AJ samples by ELITE were also found in the AJ reference genome. For those missing, the context itself was absent from the new genome, thus not allowing us to verify the presence or absence of the predicted TEi. It was never the case that I found a predicted TEi context in the new AJ reference, that was not adjacent to a TEi.

This full genome assembly, as mentioned previously, provides a means for estimating the rate at which private TEis found by ELITE are, in fact, not private, but instead due to the false-negatives. I considered only the new predicted TEis by ELITE (i.e. those not already in the reference) that appear in *any* AJ sample, 1544 of 2113 (73%) appear in the AJ genome. As before those missing are due to missing genomic context sequence. Of these, 1440 of the 1789 (80%) that ELITE found in *every* AJ sample appear in the AJ reference. Next, we considered the predicted false-negative cases from Table 3.4. Of the 75 predicted TEs that are shared by all but one AJ sample, 53 (71%) appear in the AJ reference, and all of the 53 were missing from the low coverage A/JOlaHsd(m001) sample. In a second set of 14 predicted false-negative TEis do not appear in the reference but appear in 5 of the 6 strains we sequenced, 7 (50%) appear in the AJ genome. Of those, 4 of the 8 missing from A/JOlaHsd(m001) were included. Overall, there is substantial agreement between the TEi's found by ELITE and those incorporated into the AJ reference genome.

3.3.7 Runtime Comparison

I measured the total time spent on each of the four AJ samples to discover TEis by ELITE, MELT, and TEMP (Table3.5). At first, I constructed msBWT of each sample for running ELITE. On the other hand, for running MELT and TEMP, I created bam files by aligning all the short reads of each sample to the mouse reference genome using bwa-mem. Each bam file was then sorted and indexed using samtools. I used 6 threads to run our msBWT construction pipeline whereas 30 threads were used for running all the preprocessing steps of MELT and TEMP due to its computational demand. These preprocessing steps for each tool were run on a machine with the following specification: Intel(R) Xeon(R) CPU E5-2643 v3 @ 3.40GHz, 6 cores, 256 GB memory. Running the actual TE searching tool after the preprocessing step does not require heavy computational resources. Hence I ran the rest of the steps on an Intel(R) Xeon(R) E5420 CPU, 4 cores, 2.50 GHz with 32 GB RAM. ELITE, MELT, and TEMP are written in Python, Java, and Perl, respectively.

Sample	ELITE	MELT	TEMP	BWT index	Alignment
A/JCr(f001)	55	95	312	581	507
A/JOlaHsd(m001)	42	72	238	178	267
A/JOlaHsd(f015)	32	55	200	403	407
A/J(f321)	45	77	360	408	514

Table 3.5: Total time required for each tool to locate six classes of TE in four AJ samples (in minutes). Data preprocessing time of each tool is shown in the last two columns, where the BWT index corresponds to ELITE, and Alignment timing corresponds to both MELT and TEMP.

As we can see from table 3.5, ELITE is about 1.7 times faster than MELT. TEMP is significantly slower than both ELITE and MELT. It is also apparent that the preprocessing step for each tool is the most computationally heavy step. However, even in this case, except for sample A/JCr(f001), creating a BWT index is always faster than typical alignment. Other than providing a faster way to detect TEi, msBWT is not biased to any reference genome and has many uses, including data compression, fast kmer query, local assembly, local alignment, error correction, etc (Simpson and Durbin, 2010)(Salikhov et al., 2013)(Greenstein et al., 2015)(Wang et al., 2018).

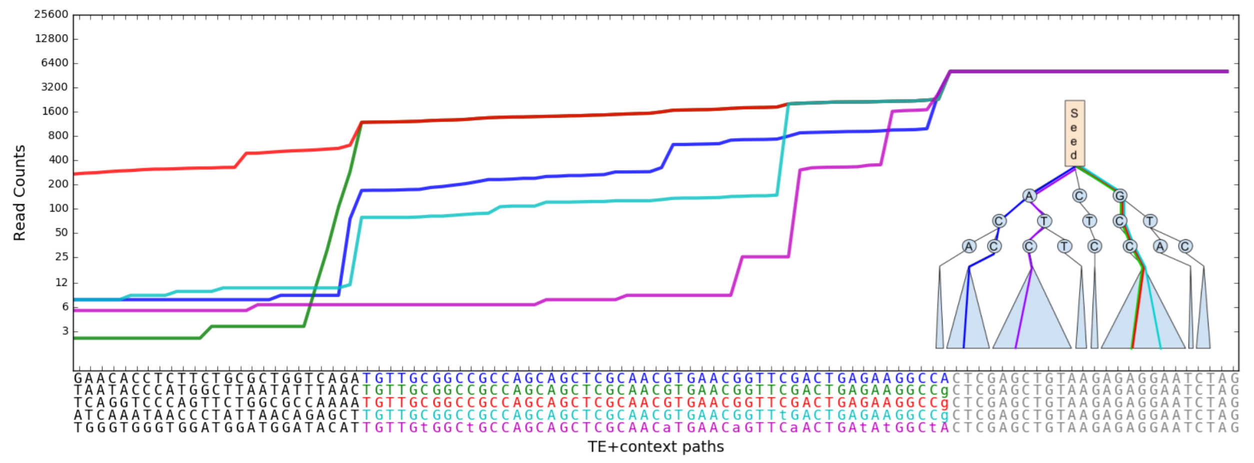


Figure 3.11: These plots illustrate a range of hypotheses generated during the initial discovery phase of ELITE. They show the read support for 5 out of more than 14000 TE-like sequence paths found during a single discovery pass using a 25-base seed from ERVB7 LTR. Each sequence is shown along the x-axis. This seed is shown in gray and is a suffix of all 5 sequences. Each TE sequence is shown in the same color as its plot, and its genomic context is the prefix shown in black. The y-values of the plot indicate the number of reads supporting each suffix extension of the seed for each TE-like sequence on a log scale. Generally, the number of reads supporting each TE-path is high until the context is reached.

3.4 Algorithm Complexity Analysis

The ELITE algorithm for finding TEs and the corresponding genomic contexts is essentially an exponential DFS algorithm. Given that a DNA sequence is consists of 4 bases, i.e., A, C, G, and T, in the worst case scenario, it will traverse all 4^r possible path to extend r bases before a seed. Fortunately, the genome is finite, and it's not possible for a genome to have all 4^r sequences when r is too large. In addition, msBWT allows us to look for any prefix of length r in $\mathcal{O}(r)$ time. Finding seed k-mer and extending it ($\mathcal{O}(k) + \mathcal{O}(r)$) are the only two operations that I use in my discovery phase.

The figure3.11 illustrates paths found during the discovery phase of ELITE. It follows five out of the more than 14000 paths explored during a search starting from a single seed from a TE called ERVB7_1-LTR_MM. The number of supporting reads for each suffix of the five paths is plotted. The DFS progresses from right-to-left in this plot. The support for the seed, shown in gray, includes 5050 reads. As the suffix is extended to the left the number of supporting reads is reduced. When a variant is reached, the tree branches and the DFS follows each path separately. Branching points near the root are illustrated in tree form under the right section of the plot. The paths taken by the five examples are overlaid on the tree. A sudden drop in read coverage along each of the five paths can be seen when a variant is found on the path. Typically, the read coverage remains high until the genomic context of the insertion point is reached (shown in black). Soon after that point, the read coverage falls into a typical range for the unique genomic sequence (between 4 to 8 reads for a 75-bp subsequence of a 150-bp read on a single strand at 30x coverage). On some occasions, the TE path does not fall into the typical range, such as seen in the red plot. These paths represent context adjacent to the TE's Long Terminal Repeat (LTR) sequence and cannot be uniquely mapped in the genome. There are other cases where a TE path cannot be mapped, as illustrated by the light blue path. While the context and TE combination appears only once in the genome, the context itself is repeated. Lastly, the purple path illustrates how ELITE finds TEi's that are significantly different than the TE template. Since ELITE does not attempt to align any part of the read once a seed is established, it is free to follow any promising path. Thus, it can find distantly related TEs that are highly diverged and perhaps unannotated. This distinction is one of ELITE's advantages over reference-alignment or TE-catalog-alignment methods. Also, this figure gives us an insight into using our approach to find any repeats by looking at the sudden drop of coverage.

3.5 Summary

I have developed a tool ELITE, that uses a novel *local-genome-assembly-based* algorithm to efficiently discover TE insertions. In addition to several independent validation methods, I also proved the legitimacy of ELITE's findings by showing its presence in real DNA. I showed different pattern of TEi discovered by ELITE within a population is highly consistent with their origin. Even though in this chapter I use a smaller population with only six samples, it is possible to run the whole pipeline in a much larger one. In chapter 5, I applied ELITE on a population that has more than a hundred samples. I discuss there in detail how ELITE is suitable for finding the

TEi segregation pattern. This helps to understand TE biology, the phylogeny of the population, and the evolution of genomes. ELITE's powerful feature of identifying TE presence and absence probes is useful for biologists in many ways. First, the probes can be used for creating primers for PCR validation, which I already showed in this chapter. Second, the polymorphic probes can also be used for Quantitative Trait Loci (QTL) mapping in a given population. Results produced by ELITE also led to interesting biological findings i,e,. many private TEis segregating in a closely related population is in the heterozygous state. They are thus providing stronger evidence for recent activity and being *de novo*. Overall, the large number of TEis found by ELITE indicate that TEs are a significant source of genetic variation that must be taken into consideration. However, in this chapter, I mainly look for TEis using six TE templates, where all six belong to a special class called *Endogenous RetroVirus, ERV*. As traditional TE database and literature suggest, there are other types of TEs and some more subclasses of ERV. That's why I extend our ELITE algorithm further to systematically find all the different types of TEis in a sample. Thus, in the next chapter 4, I describe a new pipeline Frontier, that can find any TE insertions from a short-read dataset and does not require any TE template. This template-free pipeline can potentially detect new TE subtypes that are still unknown and not annotated in any TE database.

CHAPTER 4: THE BOUNDARIES OF NOVEL TRANSPOSABLE ELEMENT INSERTIONS IN GENOMES, FRONTIER

In this chapter, I discuss a second pipeline *Frontier* for detecting and mapping transposable elements in short-read sequence data. It extends ELITE (described in chapter 3) in its ability to find novel TE types. ELITE and other split-read-based TE detection tools require templates to detect a TE boundaries. However, the sequences of TE types vary significantly within TE classes and between host species. Furthermore, the templates for TEs found in libraries like Repeat-Masker, Dfam, Repbase are typically limited to TE sequences identified in a single reference genome assembly. If an isolated population contains a novel TE, chances are very low that they will be found in these libraries. That's why I developed a template-free algorithm that automatically identifies all the spanning boundaries of TE insertions using a more general TE model. These boundary reads are similar to split-reads which contain both TE and non-TE segments. This method takes advantage of the repetitive nature of TEs by finding all the sub-sequences that are repeated more than a certain threshold in the genome and it uses two data structures, the ms-BWT and the LCP. These repeated sub-sequences are then extended further to identify potential candidates that contain the boundaries between highly repeated sequences and normal coverage, which I call the *Frontier*. However, this pattern of high-coverage adjacent to normal coverage is not unique to just TE repeats. To eliminate non-TE repeats, all candidates are then run through a classifier. This classifier identifies those candidates where the repeated sequence is more TE-like under the assumption that functional aspects of TE-like sequences are conserved. Finally, in a separate classifier, I infer the likely TE class of the TE portion from each frontier candidate. My experiments based on the mouse data reveal that a classifier that is trained on one type of mouse strain can be applied to other types. In fact, species that share similar types of transposable elements do not need separate classifiers, thus making the whole TEi detection process faster than any other existing methods. My classifier-based approach can also be extended to find other types of repeats like the boundaries of telomeres or DNA microsatellites.

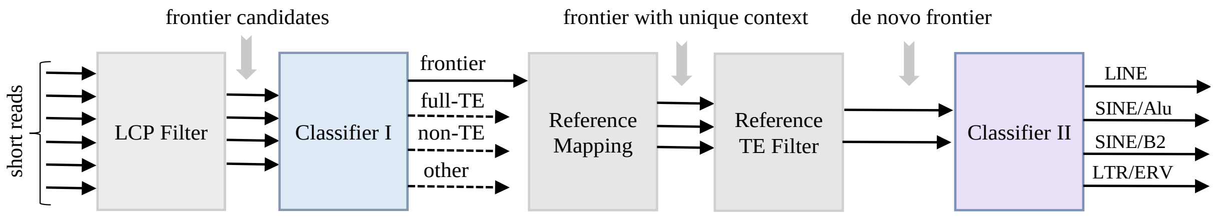


Figure 4.1: Frontier Pipeline: The Longest-Common Prefixes (LCPs) of the suffix array from a full short-read sequencing dataset (represented as a multi-string Burrows-Wheeler Transform (msBWT)) is used to extract all reads where a sizable substring is highly repeated, but, then drops to normal genomic coverage when it is extended to one side. These Frontier candidate reads are then fed to an initial *Classifier*-I to predict one of 4 different read classes, a true TEi boundary (frontier), a small variant within a TE (full-TE), non-repetitive sequence (non-TE), and repetitive sequences that are not TE (Other). Only the class type frontier i.e., reads with partial TE and non-TE segments are kept. Then two subsequent filters are applied, the first keeps only the unique contexts and the second identifies only the non-reference frontiers. Finally these non-reference frontiers are fed to the *Classifier*-II to predict the TE type of each frontier read.

4.1 Challenges and Motivation

Detecting repeats and/or Transposable Elements in high-throughput short reads is a challenge. Usually, TEs are considerably longer than a read length. Thus, each read contains only a fragment of a TE that can be mapped throughout the genome making it extremely difficult to determine how many, and where the TEs are in the genome. Some existing tools focus on identifying all the repeats/TE types in the genome while others focus on locating TE insertions. RepDeonovo (Chu et al., 2016), RepARK (Koch et al., 2014) and others use a program like Jellyfish (Marçais and Kingsford, 2011) to initially obtain a list of most frequent k-mers, then extend those that appear with high frequency to assemble potential TE subsequences, which are finally clustered and aligned to obtain consensus repeat sequences. These tools only report the different types of repeats present in a short read dataset and do not identify the insert location.

Other tools attempt to detect the location of TE insertions. Some, including MELT (Gardner et al., 2017), TEMP (Zhuang et al., 2014), RetroSeq (Keane et al., 2012) use discordant read-pairs to find TEs. A discordant read-pair occurs when one the read maps uniquely to the reference genome while its mate maps to multiple places. Since TEs are expected to be located throughout the genome, discordant read-pair methods assume that a TE has been inserted somewhere near the uniquely mapped read. The insertion location reported by these methods can be significantly off (0-300bp). Furthermore, alignment-based methods also tend to report a high number of false positives. Moreover, these methods are heavily biased in favor of TE-templates present in the reference, since the mate is expected to map to many locations.

A third class of TE detection methods are split-read-based. These methods can precisely detect the insertion location to the basepair by finding a “break-point” where the TE sequence is inserted. Since a short read cannot contain the whole TE, looking only at the sequence near TE boundaries serves the purpose. To find these TE boundaries, split-read-based tools like ELITE (Kashfeen et al., 2019), Relocate2 (Chen et al., 2017), T-lex2 (Fiston-Lavier et al., 2014), ITIS (Jiang et al., 2015) rely on a known TE template provided in advance. Relocate2, ITIS, T-lex2 aligns all the short read to the TE template. From the partially aligned reads, they clip the segments that do not match with the TE template and map them to a reference genome to find the insertion location. Often these clipped segments are not unique thus are ignored for the rest of the pipeline. ELITE also finds split-read by adopting a local genome-assembly approach seeded by a segment of a TE near its boundary. All methods for detecting TE insertions fail to find insertions of novel or highly variant TEs because of their dependency on a known TE template.

The Frontier pipeline identifies repeated genomic sequence directly from short reads without using templates based on the longest-common prefixes of adjacent entries of a suffix array. These repeated suffix-prefixes are then extended further and then filtered to keep only those segments that have a normal (i.e. non-repeated) coverage thus suggesting a single copy. This also ensures a higher chance that the extended segment might be mapped uniquely. I call highly repeated subsequences that, when extended in one direction, transition to a sequence typical of single coverage *frontier candidates*. These candidates are then run through a classifier to predict if they contain a TE and non-TE segment adjacent to each other – I call such sequences the *actual*

Frontier. Finally, in a separate classifier, I infer the likely TE class of the TE portion of each frontier. Even though machine learning is widely used throughout bioinformatics, previous work on transposable elements has focused primarily on this second stage of assigning sequence to a TE class. Examples of such tools for classifying TE sequences include da Cruz et al. (2020), Hoede et al. (2014) and Abrusán et al. (2009). Therefore it is important to have a pipeline like Frontier that not only attempts to find new TE types but also attempts to identify the locations of novel TE insertions.

4.2 Frontier Pipeline

In an attempt to find TE insertions my analysis pipeline considers reads from a given short-read high-throughput sequenced dataset. Since TEs are repeated throughout the genome, the analysis pipeline expects any TE sequence to have unusually high coverage when compared to a regular genomic sequence. Therefore, I identify all the reads that contain a subsequence that appears with a very high count (repeat), followed by an adjacent sequence that, when considered together, exhibit a normal count (context). I use two data structures, msBWT and LCP to identify repeat intervals that occur in a high-throughput short-read sequencing dataset above a given threshold. I then consider sub-intervals of the identified repeat intervals to find extended sequences with normal genomic coverage. I call these subsequences that are a mix of repeated and unique sequence *Frontier candidates*. However, there are certain cases where a frontier candidate might not be an actual TE insertion boundary. For example, any mutation in the TE sequence might cause the high count of TE segment to drop into a normal coverage range. There also exist boundaries of highly repeated sequences that are not part of any transposable element such as microsatellites and telomeric sequences. Therefore, to find the *True Frontier*, I propose a deep learning model which takes filtered short-reads, i.e. *Frontier Candidates*, as inputs.

I employ two classifiers, *Classifier*-I for predicting the true frontier and *Classifier*-II for predicting the TE type seen at the Frontier. Both classifiers use the same network structure with two convolutional layers followed by two fully connected linear layers. I transform the sequence of a

Frontier Candidate into a matrix similar to a one-hot vector and feed it directly to the *Classifier*-I. I next take the predicted frontiers from this classifier and map their context (i.e., the sequence on the side of the boundary where coverage drops to a normal level) to a reference genome to identify the unique contexts. I then filter out all the frontiers that are already present in the reference. This provides a list of non-reference novel frontiers, which are then fed to the *Classifier*-II to predict their TE type. In *Classifier*-II I included the 4 major classes of TEs present in mouse reference genome (Nellåker et al., 2012). Those are (i) LINE (ii) SINE/Alu (iii) SINE/B2 and (iv) ERV. The whole pipeline is illustrated in figure 4.1.

4.2.1 Identifying Frontier Candidates

At first, I build two data structures i.e., the msBWT and the LCP from a given whole-genome high-throughput sequencing read dataset. The msBWT can be used as an index for recovering entries of an implicit suffix-array and for performing k-mer searches within the collection of reads. The LCP is an array that contains the longest common prefix between two consecutive sorted read suffixes. The indices of msBWT and LCP array have a one-to-one correspondence. That is the i^{th} element of the LCP array will hold the length of prefix shared by i^{th} and $(i+1)^{th}$ read suffix in the msBWT. Using this LCP, along with msBWT, it is possible to find all k-mers that occur over given repeat threshold, m (typically 500 to 800 based on the read coverage).

A scan through the LCP can determine the intervals of any k-mer (prefix of the implicit suffix array entry) that is repeated at least m times. I achieve it by finding at least $m-1$ consecutive rows with an LCP value greater than k. The algorithm builds a list of all repeat intervals. Initially, if the value of LCP is less than k, then it goes to the next value until it finds one which is greater than or equal to k. It saves the index corresponding with this value as *low*. Then it continues sequentially until the LCP value is less than k. This index corresponds to the *high* end of the repeat interval. If $high - low \geq m - 1$ the (low,high+1) interval is saved to a list of genomic repeat intervals. It continues to find intervals until the end of the LCP array. However, a short read dataset can contain 100s of millions of reads depending on the organism and sequencing coverage. To overcome this problem, I break the LCP into smaller chunks and process each chunk in parallel, and finally merge the results. This step outputs all the repeats where a subset belongs to some TE classes.

Algorithm 2: Finding Frontier Candidate

```
FIND-Repeat($K, LCP, m$)
    $repeat = \emptyset$
    Select $n$ random indices $i$ from LCP where LCP[i] $> K$
    Divide LCP into chunks: $LCP[i_1 : i_2], ....LCP[i_{n-1}, i_n)]$
    for each chunk do
        $count = 0$
        while $chunk[i] < K$ do
            $i++$
        end
        $low = i$
        while $chunk[i] \geq K$ do
            $i++$
            $count++$
        end
        $high = i$
        if count $\geq m - 1$ then
            $repeat = repeat \cup range(low, high + 1)$
        end
                                        ▷ $m$ = minimum frequency of repeat
                                        ▷ $K$ = length of repeat segment
    end
```

After finding all the genomic repeat intervals, our next goal is to identify all reads where the repeat is adjacent to normal coverage. I use the same algorithm for finding repeat from LCP with some minor changes. Previously, when finding the repeats, I only considered LCP values that are greater than a certain threshold. However, when finding contexts, I keep only the LCP intervals that have both upper and lower bounds indicative of normal read coverage for k-mers of that length. Thus, for finding repeat's context, the algorithm then proceeds to subdivide repeat intervals into sub-intervals with normal coverage. Once again the LCP is scanned to find extended prefixes contained within genomic repeat intervals of length k+k' with normal coverage, typically below a coverage-dependent high-count threshold (15-20) and above a noise threshold (2). Finally, all the reads in these interval ranges are extracted using msBWT. I call these reads *frontier candidates*.

4.2.1.1 Input Data Preparation

Each input to our model consists of a read and its corresponding overlapping-k-mer count. However, to encode a read as an input we convert it to a numerical form. Since DNA can have only 4 bases, (A, C, G, or T) each R-length read is mapped into a 4xR dimensional matrix. I employ a so-called *1-hot encoding* where for each A of the read's sequence a 1 is placed in the A's row, and 0 in the others. Likewise for the other three bases. The second input encoding that I use is the overlapping *k-mer* count. I first divide the sequence into $R - k + 1$ segments or k-mers and count the occurrence of each k-mer across the entire dataset using the msBWT. Then I replace the 1s of the 1-hot encoding with the log_2 of the k-mer count. This trims k-1 bases from the sequence as we can only get $R - k + 1$ k-mers from a read of length R. Thus, I choose a k value large enough so that it is informative of actual coverage, but small enough so that the read is adequately sampled at many base positions.

4.2.1.2 Model Description

I designed a 4-layer deep neural network and trained it to classify each frontier read. The first two layers are convolutional network layers, followed by two fully-connected layers. This architecture is shown in figure 4.3. The first convolutional layer in our model has 4 input channels, 9 output channels, and a kernel of size 15. Recall our input data representation which is a $4 * (R - k + 1)$ matrix for each read, is now divided into 4 vectors. Each of these 4 vectors is fed to each input channel independently. I use 1D convolution as I have 1D vectors as input instead of matrix. The output of this layer is fed to a second CNN which has 9 input channels, 12 output channels, and a kernel of size 5. The output of this layer is then fed to a fully connected linear layer followed by another linear layer which predicts the class for each input. The dimension of this layer's output is the same as the number of classes and represents the probability of being

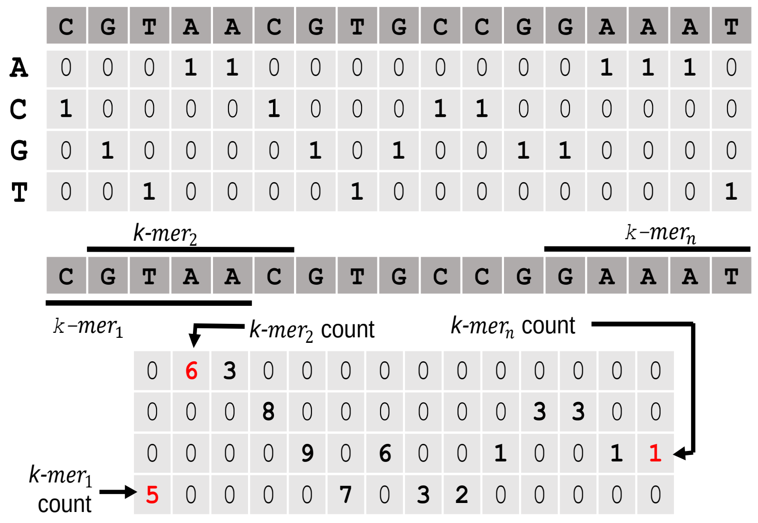

C G T A A C G T G C C G G A A A T
A 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0
C 1 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0
G 0 1 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0
T 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1
k-mer2
k-mern
k-mer1
k-mer2 count
k-mern count
0 6 3 0 0 0 0 0 0 0 0 0 0 0
0 0 0 8 0 0 0 0 0 0 3 3 0 0
0 0 0 0 9 0 6 0 0 1 0 0 1 1
5 0 0 0 0 7 0 3 2 0 0 0 0 0
k-mer1 count

Figure 4.2: Input data representation: I use a modified one-hot vector encoding to represent input sequences of our classifiers where the *one* values are replaced by a $log_2(k$-mer count) of a sub-sequence centered around the represented base. This effectively trims the sequence by $k-1$. The k-mer counts used are the sum of the sub-sequence's frequency in the reads of the NGS data set in both the forward and reverse-complement direction. They are computed using an FM-index of the msBWT. Since the k-mer is derived from an frontier candidate read from the dataset, it is guaranteed to appear at least one time. This encodes sequence motifs in combination with their genomic frequency.

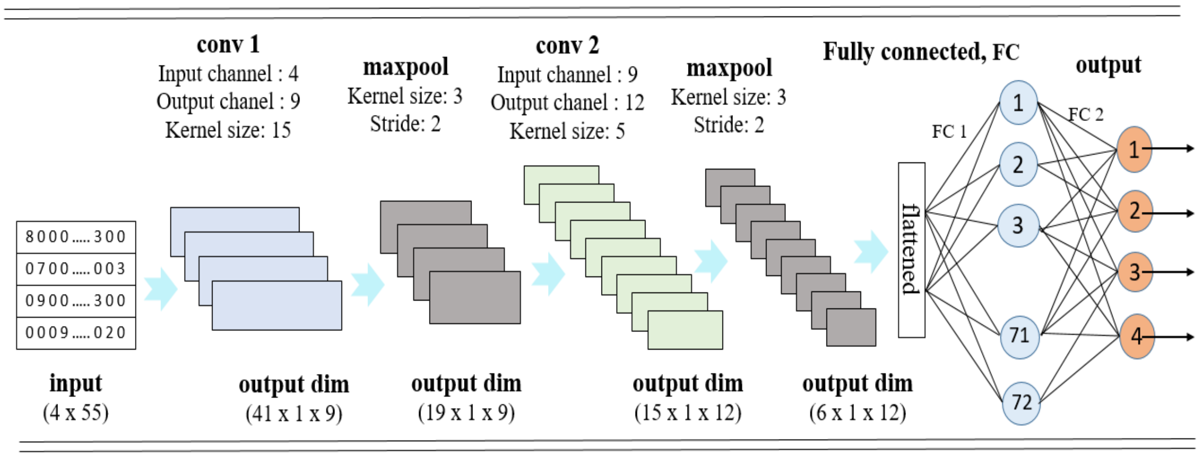


Figure 4.3: Architecture of deep-learning neural network used: A 4-layer convolutional neural network is used for classiying Frontier. First two layers are CNN, each of these are followed by a maxpool and a RelU. The flattened output from CNN 2 is then fed to a fully connected linear layer. Final layer's output is 4D for predicting probability for 4 classes.

in that class. We have 4 output classes i.e., i) Frontier ii) Full-TE iii) non-TE iv) others. A candidate is a Frontier if it contains both a TE and a non-TE segment. It's a Full-TE if all the bases are part of a TE. It's a non-TE if it contains no TE or any other kind of repetitive sequence. All other

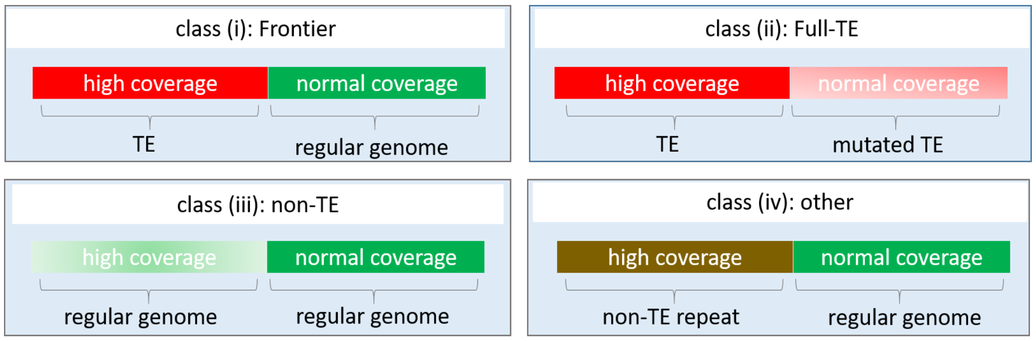


Figure 4.4: Different classes of our Frontier *Classifier*-I. class (i) is the true frontier, where high coverage repeat is next to a normal coverage genome. class(ii) belongs to full-TE where a mutation in TE resulted the drop in coverage from high to low. class(iii) belongs to non-TE where the high coverage portion came from regular genome, most possibly is from a duplicated region or from a gene family. class (iv) contains high coverage segments which came from nonTE repeat like telomere or DNA microsatellite.

repeat types of uncategorized reads fall into the other class. 4 Different classes of *Classifier*-I is shown in figure 4.4. Even though I am only interested in the frontier class, suggesting that a binary classifier might work, in practice, I observed that modeling 4 biologically relevant classes improves the classification accuracy for the frontier class. Also for a binary classifier, all of the many types of non-frontier reads fall into a single class. Since there are many examples in this class compared to frontier, it creates either an under-sampled set of negative examples or a hugely imbalanced dataset, neither is ideal for a good classifier. Thus I propose a 4 class classifier for predicting frontier. Three classes other than the *true frontier*, might be useful in future research if someone wants to annotate other, non-TE, types of repeats in the genome.

4.2.2 Mapping Frontier Context

After finding all the *true frontiers*, I then identify their positions with respect to a reference genome. Now, if a TE insertion is not present in the reference genome, then the frontier read will not be there either. However, the context part of the frontier must be in the reference genome unless other mutations happen exactly at the same place. Thus I map the position of the context using bowtie2(Langmead et al., 2009) (which is also BWT based). I set the parameters such that it allows 2 mutation in 30 bases. Mapping the Frontier's context allows us to find the TE insertion location in terms of the reference genome's chromosome and position. I only keep the frontiers where the context maps uniquely. For each unique context, I consider the surrounding sequence from reference and check if it contains any annotated repeat/TE. If no repeat is found, then I annotate it as a Novel Transposable Element Insertion. However it is obviously reasonable if someone wants to keep the frontiers that are also in reference. Sharing TEis with reference indicates the sample is similar to it, and which might be useful in other genomic analysis. Moreover, frontiers that don't have uniquely mapped contexts can also be kept for other kind of experiments. For example, it is possible that a context is duplicated in reference genome but not in a sample, thus fails to map uniquely.

4.2.3 Identifying Frontier's TE type

The final step of our pipeline is to classify each *Novel frontier* based on its TE type. I use the same network structure as the *Classifier*-I. Only difference is that the input vectors for *Classifier*-II do not incorporate the overlapping k-mer count and use the whole frontier read as input. I have 4 major TE types included in our model, those are LINE, SINE/Alu, SINE/B2, and LTR/ERV. These are the most frequent types of TEs seen in mouse genome and annotated by RepeatMasker (Nellåker et al., 2012). Using this classifier for any species other than mouse might require a to change in the number of output classes, which can be trivially done by changing parameters in the network model. This will also require to retrain the model using desired TE types.

Figure 4.5 shows the fraction of mouse and human genome masked by RepeatMasker. Back arc represents the unmasked/non-repeated regions. As we can see, in both cases, almost half of the genome (45% in mouse and 52.5% in human) are masked due to some repeated elements. Majority of the repeats are from three different kinds of transposable element, SINE (shown in blue), then ERV (shown in green), finally LINE (shown in purple). I broke down the larger class SINE into two groups for making our annotation more accurate.

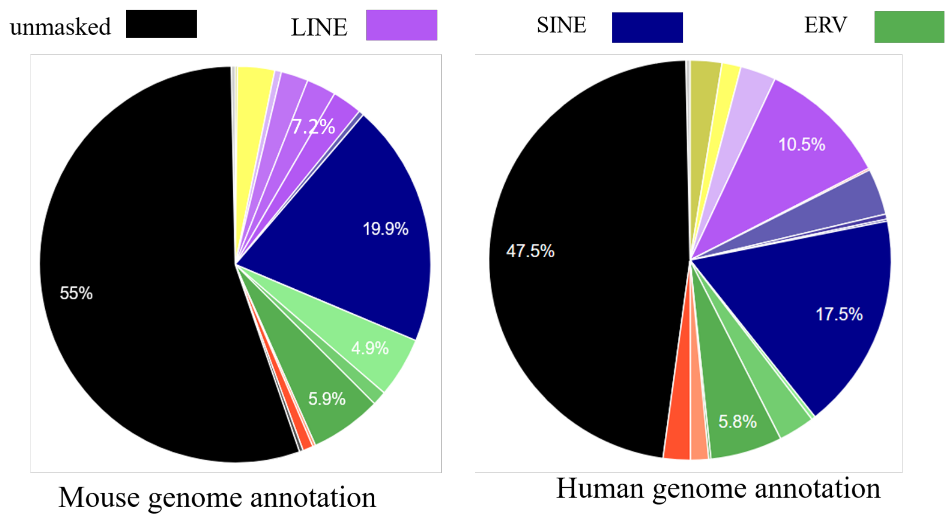


Figure 4.5: Fraction of different types of repeats including transposable elements in mouse and human genome. Majority of the repeats are from three different kinds of transposable element, SINE (shown in blue), then ERV (shown in green), finally LINE (shown in purple).

Name	Strain	Sample	Total reads	Read length
AJ	A/J	f321	634,232,014	150
B6	C57BL/6J	m03636A	1,045,633,612	125/151
129S1	129S1/SvlmJ	m157	586,539,366	150
NOD	NOD/ShiLtJ	m146	837,204,388	150
NZO	NZO/F111	m146	624,071,858	150
CAST	CAST/EiJ	m090	512,388,000	150
PWK	PWK/PhJ	f6114	481,626,592	150
WSB	WSB/EiJ	f111	429,564,492	150

Table 4.1: Eight samples were used for evaluating the performance of two Frontier classifiers. These 8 samples represent 8 widely used mouse strains. B6 sample C57BL/6J is based on mouse reference genome. It is also used for making training dataset. Trained models obtained from B6 were directly applied to other samples to find their frontiers and corresponding TE type.

4.3 Evaluation of Frontier Pipeline

I applied our Frontier pipeline to eight short-read sequencing datasets. This includes five classical inbred laboratory strains (AJ, B6, 129S1, NOD, NZO) and three wild-derived strains (CAST, PWK, WSB). All samples were sequenced using either Illumina X10 or Illumina HiSeq4000 sequencers with paired-end reads. The Illumina sequencing data from each sample was used to construct an msBWT and corresponding LCP as described by (Holt and McMillan, 2014). The details for each sample are shown in Table 4.1. Choosing these 8 strains were not random and facilitates some of my later analyses. Together they represent three subspecific origin of mouse. Those are: 1) M.m. domesticus (includes all the lab strains and WSB), 2) M.m. musculus (PWK), and 3) M. m. Castenous (CAST). A small phylogenetic tree can be built based on their subspecific origin where M. m. Castenous and M.m. musculus got separated from M.m. domesticus a long time ago. The detail tree is shown in figure 5.2. These 8 strains are also the ancestors/-founders of a large population called Collaborative Cross (Churchill et al., 2004). In the next section, we will discuss about the transposable element insertions from these founders that were passed on to their descendants CC samples.

4.3.1 Evaluation of Finding Frontier Candidate Algorithm

I applied our algorithm 2 on both simulated and real datasets. For each dataset, I first found all the intervals in the msBWT that contain repeats. I divided the LCP array into multiple segments and sent each one to a different processor for finding repeats in parallel. I consider a subsequence as a repeat if it's present at least 10 times in the sample's genome. For example, in a sample with coverage 40x, I expect to see it around 10x40 = 400 times in short reads. After finding the repeat intervals, I looked for intervals that are adjacent to a context with normal coverage. I set the lengths of repeat and context to 45 and 30 base pairs respectively. Thus, each Frontier candidate has a length of 75 bases. Using these parameters, I next applied our finding frontier candidate algorithm 2 on both simulated and read data. I ran this step of our pipeline on a machine with the following specification: Intel(R) Xeon(R) CPU E5-2643 v3 @ 3.40GHz, 6 cores, 256 GB memory. I also built msBWT and LCP on this machine. I later experimented with different parameters and compared their performances. However, the sum of the lengths of the repeated portion and context portion of a frontier read must be smaller than the total read size. In fact it is better if I get multiple candidates for the same TE boundary otherwise I might miss

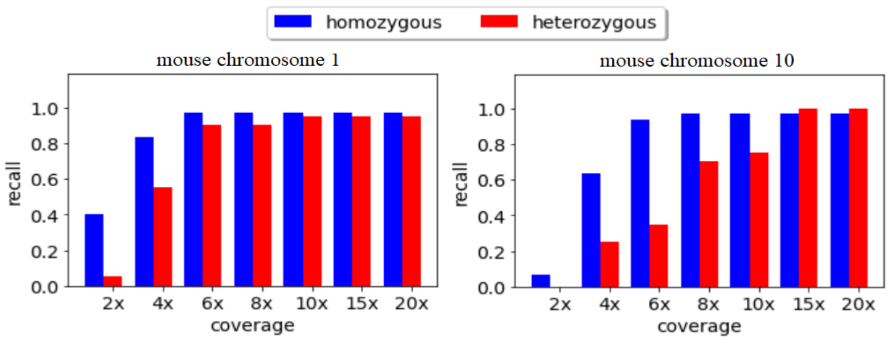


Figure 4.6: Recall rate of algorithm 1 on simulated data. The X-axis shows reads coverage, and Y axis shows the recall rate ranging from 0 to 1, Recall for homozygous and heterozygous frontier discovery is shown in the blue and red bar respectively. For both chromosome 1 and 10, the recall rate increases with increased coverage, specially for the cases where TEi was inserted heterozygous. However, the recall rate stablizes ($\approx .9$) when the coverage is around 15x.

some completely. However, for all the real data, I do not have any coverage-based threshold to determine if a segment is repeated. This needs to be done in case data has low coverage sample.

To estimate the recall rate of frontier candidate discovery, I ran algorithm 2 on simulated data where ground truth is known. I inserted 50 TEs into chromosome 1 and chromosome 10 of the mouse reference genome. For each insertion, I recorded the frontier sequence (last 45-mer of TE + first 30-mer of regular genomic context = 75-mer frontier candidate) as ground truth. For both chromosomes, of the 50 inserted TEis, 30 were inserted as homozygous and 20 were inserted as heterozygous. I used Samtools (Li et al., 2009) to simulate 150-bp reads at different coverages ranging from 2x to 20x. I ran algorithm 2 on both chromosomes to get a list of frontier candidates. To measure the recall rate, I checked if each ground truth frontier candidate is found by running algorithm 2. Since there are many old existing TEis already in the mouse genome, the algorithm found more candidates than the ones I inserted. That's why I calculated only the recall rate and skipped precision. Recall rates for both chromosomes are shown in figure 4.6. As we can see, for both chromosomes, recall rates for homozygous TEis are higher than the heterozygous ones when the coverage is low. When the coverage is only 2x, even the homozygous insertions are hard to be found. However recall rate increases with the increase in coverage. For chromosome 1, it became stable around 6x, which was not enough for chromosome 10. When the coverage was around 15x, my algorithm 4.6 was able to find 95% of the candidates irrespective of the zygosity or the chromosome.

Similarly for the real data shown in table 4.1, I also ran the Finding Frontier Algorithm 2. In this case, I do not have the ground truth, thus were not able to show the recall rate. However, B6 sample is identical to the mouse reference genome. Therefore, they should share all the TE insertions unless B6 has any very recent ones that occurred after the reference genome was assembled. I use the frontier candidates of B6 sample to further identify the true frontiers. I use a subset of the frontier candidates that are present in the mouse reference genome. Then with the help of an open source software RepeatMasker, I categorize each candidate if its true frontier.

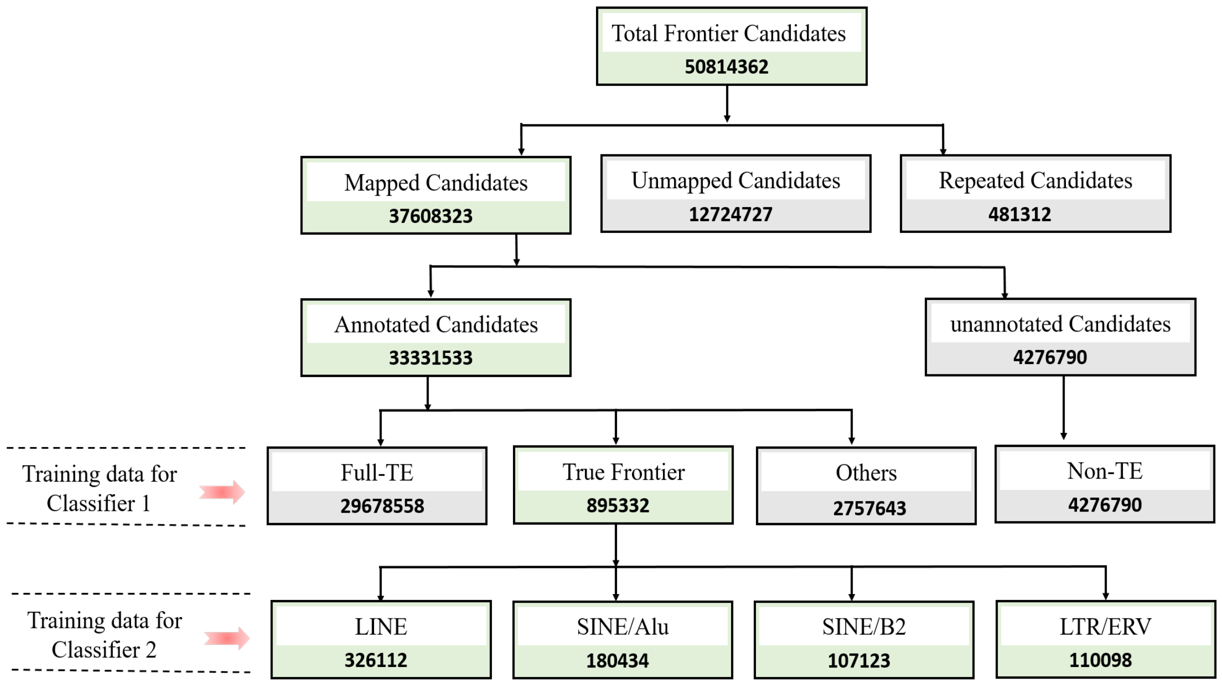


Figure 4.7: This figure shows the steps to build training data for both classifiers. At first, I found Frontier candidates in a B6 sample (The mouse reference genome is based on this strain). All the candidate reads are then mapped using bowtie2 to find their location. Each of these locations was cross checked with RepeatMasker's verify if they had been annotated as a repeat element. If nothing was found or annotated, then it was kept as a training example for the non-TE class of *Classifier*-I. If it was annotated, then I again check if its a TE or some other kind of repeat. If its not a TE, then it is used as a training example for others. If was a TE, then I checked how many bases of these reads belong to TE. If more than 65 bases of a read contains TE, then it is used as an example of the full-TE class, otherwise its kept as a training example for the True Frontier class.

4.3.2 Training Dataset for Frontier

For each frontier candidate, I searched for its position in the reference and checked if it is annotated as a TEi by RepeatMasker. I found a total of 33,331,533 candidates that either contain partial or full repeat sequence. Candidates that contain partial TE sequences are in class (i): Frontier. I only used the ones where the length of TE and non-TE segment in a candidate are at least 25. Then candidates whose entire sequence is a part of a TE sequence are in class (ii): Full-TE. Candidates with no TE/repeat segments are in class (iii): non-TE and all the other kinds of candidates with different kinds of non-TE repeats are in class (iv): others. I had a total of 895,332 examples in class (i): frontier. There were many examples for class (ii),(iii), and (iv). I randomly subsampled 895,332 examples to make my training dataset balanced. For each read, I computed the k-mer frequency for overlapping 21-mers from that read using the msBWT along with FM-index. These counts were used as a part of input in addition to the sequence as described in 4.2.1.1. For my second classifier, I used RepeatMasker's annotation to label the 4 different TE types present in each Frontier.

Model	LCP-filter	Precision	Recall	F1 score
Baseline	No	.493	.557	.53
Frontier	Yes	.884	.841	.862
Frontier*	Yes	.913	.906	.909

Table 4.2: Comparison between a baseline, described in section 4.3.3, and the proposed Frontier classifiers. The Baseline performs poorly in general compared to a classifier that uses an LCP-filter. Frontier* performs the best by using the overlapping k-mer count. However, for a very large dataset, the overhead of counting all the overlapping k-mer is high. So I recommend using Frontier for a large dataset as it can still predict more than 84% of the frontiers with high precision (.884).

4.3.3 Performance of *Classifier*-I on B6

To evaluate the performance of the frontier classifier, *Classifier*-I, I compared it with a typical sequence-based classifier. Since there was no attempt made previously to classify Frontier, I built a baseline classifier that did not use LCP-filter. It takes the read sequence directly from the short read dataset instead of the frontier candidate as input. Note that *Frontier* classifiers use the frontier candidates obtained from LCP-filter where each read contains partial repeat and partial normal coverage genomic sequence. Since baseline does not use this filter, it cannot be trained with the same frontier candidates. Thus I made another training data for the baseline classifier. Using RepeatMasker's annotation of TEi in the reference genome, I identified random 50000 places of TEi. I made examples of the frontier class by adding 35 bases nonTE sequence after a 45-mer TE segment. For the class full-TE, I made data by taking 75-mers entirely from a TE sequence. For the class non-TE, I took 75-mers that were outside any annotated TEi. Then finally for the last class (other), I chose random 75-mers that did not fall into any of the previously mentioned three classes. Using 90% examples from this dataset, I trained the baseline classifier and tested its performance on the rest 10%.

To evaluate the result of the classifier, I used the dataset described in 4.3.2. Here I also used 90% of the examples for training and the rest 10% for testing. To see how the overlapping kmer affects the performance of the classifier, I ran two separate models, where one used the overlapping kmer count (*Frontier**) and the other did not (*Frontier*). However, all three models, including the baseline have the same network structure with two CNNs and 2 Linear Layers. The comparison of the three models is given in table 4.2 in terms of three metrics: precision, recall, and F1 score. Since this is a multi-class classifier, I report the metrics values for the class Frontier which is our main interest. We can see both classifiers that used frontier candidates as input achieved high precision, recall, and F1 score when compared to the baseline. Overlapping kmer count also helped to get better precision.

I also provide a confusion matrix to demonstrate the classifier's performance in predicting other classes. The matrix is shown in figure 4.8. Here I showed the result of Frontier* that used overlapping k-mer count. A randomly sampled set of 1000 examples were used to create this matrix. All 4 classes have approximately equal numbers of examples. As we can see, 127 (47%) out of 267 Full-TEs are misclassified as others and 50 (18%) as non-TE. I investigated this small

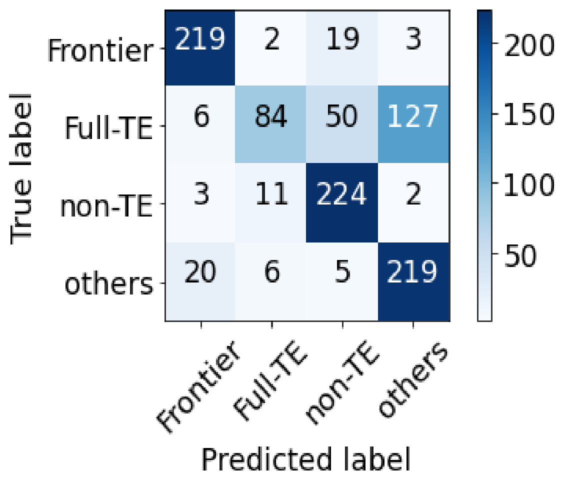


Figure 4.8: Confusion Matrix showing the result of running *Classifier*-I on 1000 randomly selected candidates. Each row of this matrix shows the True labels, and each column shows the predicted labels. There are 243 Frontiers, and my classifier misclassifed 24 examples in total. It also misclassified many Full-TEs as others. The classification errors of Full TEs have no impact since I only process further those reads identified as Frontier. Many of these appear to be older and highly mutated TE insertions.

set of data and observed that most of the Full-TEs in those cases are very old and have many mutations that diverge significantly from their original sequence. However, I achieve a recall rate > 0.9 for the other classes and I care only for the candidates classified as Frontier in this step.

4.3.4 Effect of different parameters on the Performance of *Classifier*-I

There are three parameters in our pipeline that initially were chosen in an ad-hoc fashion. First, the length of repeat in a frontier candidate (= 45), then the length of context in a frontier candidate (= 30), and finally the length of overlapping kmer (= 21) for input sequence encoding. In this section, I consider if varying the values of these parameters significantly impacts the performance of *Classifier*-I.

4.3.4.1 Effect of Repeat Length

Recall that in the first step of our pipeline 4.2.1, I searched the msBWT and LCP to find all the repeat segments where the length of repeat segment $K = 45$. To examine the effect of this

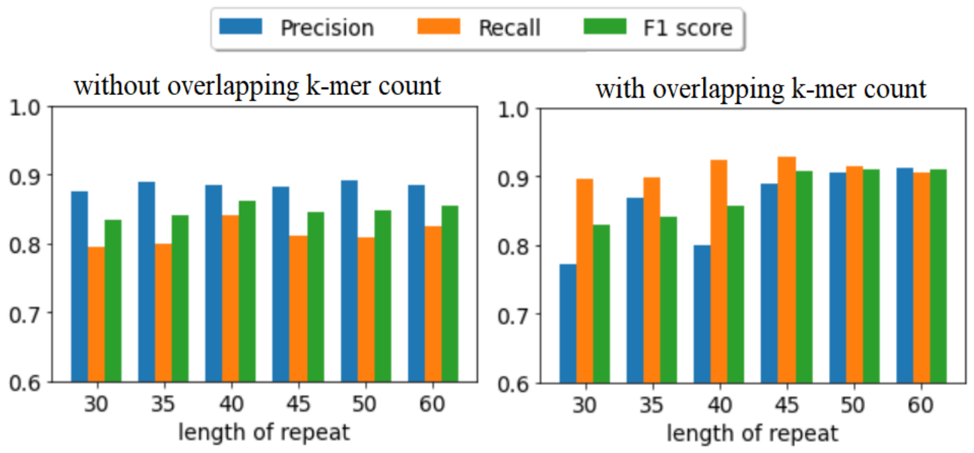


Figure 4.9: Effect of different repeat length (ranging from 30 to 60) on classifier's performance. X-axis shows the repeat length and Y-axis shows the values of precision, recall and f1 score which ranges from 0 to 1. Precision and F1 score of Frontier* increases with the increase in repeat length and becomes stable when the length is around 50.

choice for repeat length, I varied it ranging from 30 to 60 base pairs and observed the classifier's performance. I ran our two separate models where one considered the overlapping k-mer count and the other one did not. Figure 4.9 shows the effect of different repeat lengths on our classifier. Other two parameters, the length of context and the length of overlapping k-mer were kept fixed at 30 and 21 respectively. For different repeat lengths starting from 30 to 60, I ran both versions of our *Classifier*-I and calculated their precision, recall, and F1 score. I did not see any significant change in the performance of Frontier (without k-mer count) for different repeat lengths. But it affects the precision and f1 score for Frontier*. In fact, for length of 30, Frontier*'s precision rate drops lower than Frontier. Effect of repeat length almost gets nullified when its more than 50. It is conceivable that increasing repeat length helps classifier to distinguish the nonTE repeat with real TEs. However, I cannot make the repeat length too big because the whole repeat and context must fit in a single short-read.

4.3.4.2 Effect of Context Length

Another parameter used while searching for frontier candidates was the context length. Initially, I set the context length to 30. But to see if the length of context affects the classifier's per-

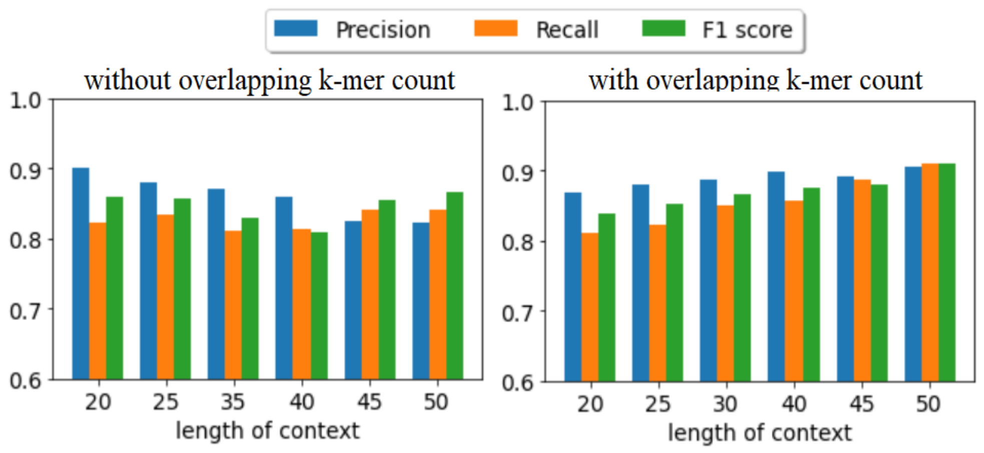


Figure 4.10: Effect of different context lengths (ranging from 20 to 50) on classifier performance. X-axis shows the length and Y-axis shows the values of precision, recall and f1 score which ranges from 0 to 1. For the Frontier classifier without k-mer count, precision decreases with the increase in context length as opposed to both recall and f1 score. On the other hand, for the Frontier* classifier with k-mer count, there was a steady increase in the discovery rate.

formance, I ran the same model with different context lengths. The performance comparison is shown in figure 4.10. The Frontier* classifier that uses overlapping k-mer count performed better and recall rate increased with the increase in context length. For this experiment, I kept fixed the length of overlapping kmer to 21 and repeat length to 45. From my experiment, I observed that, frontier* discovered more true frontiers and achieved the best recall rate. I believe this due to incorporating the overlapping k-mer counts as a signal of the expected variable coverage expected across a Frontier read.

4.3.4.3 Effect of Overlapping kmer

The last parameter I varied is the length of overlapping kmer. This only applies to Frontier* (with overlapping kmer count). The idea behind including this count is to capture changes in coverage for portions of the read. Sometimes, only the sequence itself is not sufficient enough to say if it's a TE nor a non-TE sequence. A mutation in a TE subsequence can drop its count from high to normal. This may lead the classifier to miss-classify the TE as a Frontier. But if that TE sequence is broken down into small overlapping k-mers, then the k-mers without the SNP will still have a high count. That's why I used overlapping kmer count as *Classifier*-I's input in

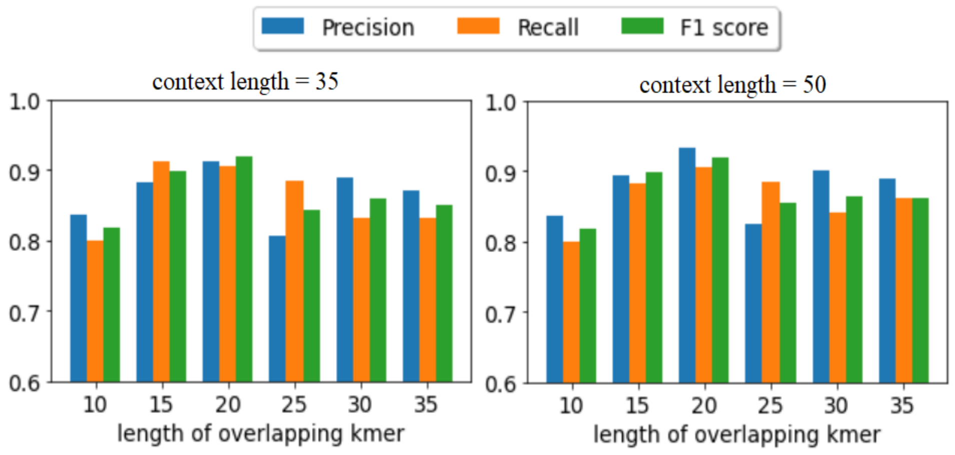


Figure 4.11: Effect of overlapping kmer length (ranging from 10 to 35) on classifier's performance. X-axis shows the kmer length and Y-axis shows the values of precision, recall and f1 score which ranges from 0 to 1. I ran this experiment twice for two different context lengths (35 and 50). For both cases, the classifier worked best when the overlapping kmer length is around 15-20 bases.

addition to the sequence. However, if the value of k is too low, then it will be all over the genome, thus will lose its purpose. On the other hand, if the value is too high, I will lose more bases from the sequence because the length of the input depends on k. To see how the value of k affects the performance of our classifier, I vary its value ranging from 10 to 35 by keeping the context length fixed. For each different k, I then ran the *Classifier*-I. The observed results are shown in figure 4.11.

4.3.5 Performance of *Classifier*-I on Other non-B6 Datasets

I next used our trained model from *Classifier*-I to predict frontiers of 7 other non-B6 samples. At first, I used the LCP-filter to identify the frontier candidates by using algorithm 2. I randomly selected 50000 candidates from each sample and ran *Classifier*-I. Here I applied *Frontier* classifier that does not use overlapping kmer count. For validation, I obtained the class label by querying the *candidate* in RepeatMasker because I cannot compare it with the reference. For this experiment, I sub-sampled 50000 candidates as its expensive to get the repeatMasker's annotation. If RepeatMasker identifies a TE in a candidate, then it can be either class(i): Frontier or class(ii): Full-TE. If the length of a TE segment in a candidate is less than 50 and greater than

25 then we label them as Frontier, otherwise, it's a full-TE. If a candidate does not contain any repeat-like sequence then I label it as the class(iii): non-TE. The rest of the reads are considered as a class(iv): others, which includes non-TE like repeats and their boundaries. Even though my training data used only frontier candidates from a B6 sample, I achieved a precision rate > 0.9 for the majority of the other samples with different strains. The f1 score is also more than 80% except for the wild strain PWK. The details are listed in table 4.3.

4.3.6 Performance of *Classifier*-II

After identifying the *frontiers*, I applied several steps to identify non-reference Novel TE frontiers in a sample. These frontiers were then classified based on their TE type. My current model includes the 4 major TE families found in the mouse reference genome (LINE/L1, SINE/Alu, SINE/B2, and LTR/ERV). More TE classes can be added based on the organism if required but the classification performance tends to degrade if too many subclasses are included. To analyze the performance of *Classifier*-II, I randomly selected 1000 frontiers from each sample. My TE classification *Classifier*-II model was trained on examples from the mouse reference genome as labelled by RepeatMasker. The landscape of TE insertions in the remaining 7 strains is expected differ from the reference, although some sharing is also expected. In particular, the three wild-derived mouse strains (CAST, PWK, and WSB) are expected to differ more significantly from the reference than the four common laboratory strains (AJ, 129S1, NOD, NZO), as they likely share more recent common ancestors.

In the confusion matrices shown in 4.12, I compared the result of our TE-type classifier (*Classifier*-II) to predictions made by RepeatMasker for the high-repeat 45-mer portion of the frontier. Recall that the HMM used by Repeatmasker is based on TEs found in the mouse genome reference (mm10). Thus, for the sequenced sample B6 the predicted labels from RepeatMasker can be considered ground truth, whereas for other samples they should be considered as yet another classifier. In the case of B6 we selected a balanced set of 4 TE types based on RepeatMasker's classification. For other samples we selected 1000 frontier sequences at random. The resultant

Name	Precision	Recall	F1 Score
AJ	0.833	0.795	0.814
129S1	0.907	0.810	0.855
NOD	0.923	0.766	0.837
NZO	0.939	0.705	0.805
CAST	0.921	0.761	0.833
PWK	0.855	0.691	0.764
WSB	0.952	0.800	0.870

Table 4.3: Performance of *Classifier*-I on other non-B6 sample: average precision ($\approx$ .9) is as high as B6 but the f1-score ($\approx$ .8) and recall rates ($\approx$ .75) are lower than what we saw on B6. Note that the test data here are from a completely different strain than the training data.

confusion matrices suggest that our TE-type classifier (*Classifier*-II) has high precision ($> 98\%$) in the B6 case, and is highly consistent ($> 93\%$) with Repeatmasker for other cases. The most common inconsistency is when our classifier calls SINE/B2 where RepeatMasker predicts an LINE/L1. These inconsistencies are probably caused by the variable length polyadenylation signal that is common at the 3' insertion boundary of these two element TYPES.

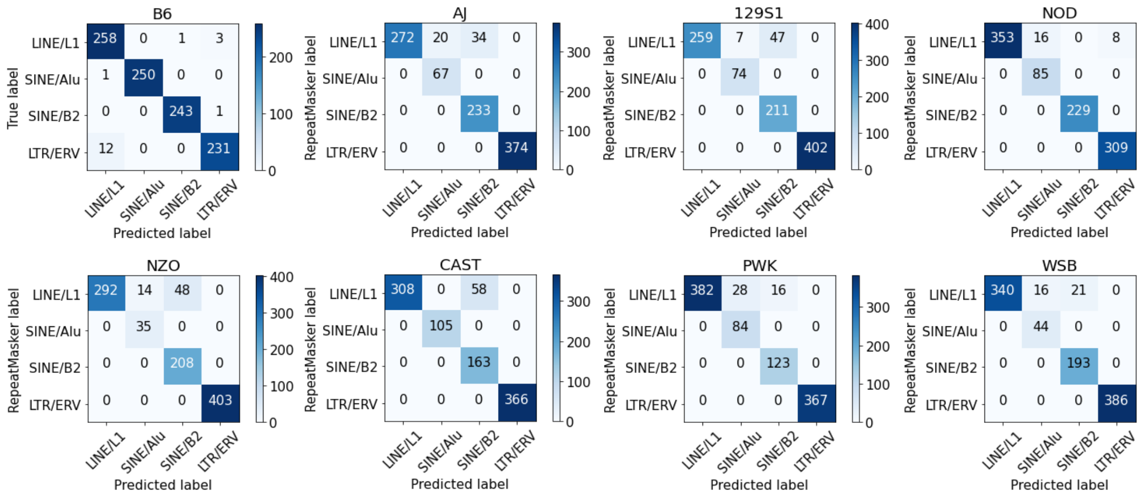


Figure 4.12: Confusion matrices showing the result of running our TE classifier *Classifier*-II on 1000 frontier sequences from samples representing 8 mouse strains. I compare our predicted labels (columns) to those of RepeatMasker (rows). RepeatMasker annotates TE insertions in the mouse reference genome (mm10) which is based on the mouse strain C57BL/6J for which the sample B6 is an example. Thus, I considered Repeatmasker as ground truth for B6, a balanced set of the four TE types. For all other 7 samples I merely compare my predictions with those made by RepeatMasker for consistency. As we can see, the diagonal values are significantly larger than the others indicating an overall consistency of more than 93%.

Type	Name	Total Reads	Frontier Candidates	Unique Frontiers
Lab Strain	AJ	634232014	51526031	5146368
Lab Strain	129S1	586539366	46433170	4327522
Lab Strain	NOD	837204388	52153639	5639526
Lab Strain	NZO	624071858	51263358	5086113
Wild Strain	CAST	512388000	44730127	2217820
Wild Strain	PWK	481626592	44633988	2254541
Wild Strain	WSB	429564492	36597602	2825752

Table 4.4: The result of running the first two steps of Frontier pipeline. I ran both Frontier and Frontier* on the candidates. To avoid mapping twice, I found the unique frontier candidates first, then ran both versions of the *Classifier*-I to find the true unique frontiers. Overall I found more candidates and unique frontiers for the lab strains compared to the wild strains.

4.3.7 Finding Novel Frontiers

I applied the Frontier pipeline to 7 common laboratory mouse strains sequenced at 30x coverage with 150bp reads. I first constructed msBWTs and LCPs for a single sample of each strain. Then using these two data structures, I identified frontier candidates. As before, I considered all subsequences with 45-mer repeats appearing in more than 400 reads (more than 16 times the expected coverage) that when extended by 30 bases appear as normal coverage (15 reads). Then I ran *Classifier*-I (without overlapping kmer count) to predict the true frontiers from these candidates. Each sample generated more than 40 million frontier candidates. Wild strains (CAST, PWK, WSB) have less candidates compared to the Lab strains (AJ, 129S1, NOD, NZO). Higher coverage in NOD also resulted in higher number of candidates.

Even though the next step of our pipeline is to run *Classifier*-I, for this experiement, I swap the *Reference Mapping* step with the *Classifier*-I. This will not affect our result, but help us to save time as I ran both *Frontier* and *Frontier** on the same candidates. For each frontier candidate, I then aligned the 30-base context to the mouse reference genome (mm10) to find the location of TE insertion. I kept only those contexts that mapped uniquely in the genome. Around 10% of the contexts were mapped uniquely for the lab strains , 7% for the wild strains. The result of running these two steps are shown in Table 4.4.

Name	True Frontier	Total TEis	Novel TEis	LINE/L1	SINE/Alu	SINE/B2	LTR/ERV
AJ	1168218	181912	2200	671	249	480	800
	1269084	364787	2426	692	229	490	1015
129S1	1055816	165809	2196	621	265	519	791
	1114146	317637	2439	643	244	494	1058
NOD	1456247	216913	2938	1030	393	680	835
	1467838	390509	3288	1115	350	715	1108
NZO	1147893	179681	2240	730	227	519	764
	1232494	357466	2425	754	199	516	956
CAST	521986	92515	4240	1419	595	922	1304
	577882	180417	4536	1605	556	902	1473
PWK	516958	92328	4683	1640	614	1055	1374
	571320	181673	4903	1797	518	1032	1516
WSB	639532	107628	2175	647	208	433	877
	729109	221525	2243	659	196	421	976

Table 4.5: Results from running the last 3 steps of Frontier pipeline on 7 samples. These 7 samples represents 7 common mouse strains. For each sample, I ran two versions of *Classifier*-I, Frontier and Frontier*. Two rows of each sample shows the result obtained from Frontier, and Frontier* respectively. For each sample, Frontier* predicts more true frontiers than Frontier. This leads to more total TEis, novel TEis. Novel TEis are further broken down according to their predicted TE type. I observed more novel TEis in CAST and PWK as compared to the other samples even though there were TEis that match the reference for TEis in these strains. The large number of TEis in A/J 129S1, NOD, and NZO in contrast to the relatively smaller number of novel TEis suggests that these laboratory strains share more TEis in common with the mouse reference genome.

On the set of unique frontiers, I ran both versions of our *Classifier*-I: Frontier and Frontier*. For each sample, around 22% unique candidates are classified as true frontier by frontier. On the other hand, frontier* that uses coverage information predicts 24% of the unique candidates as true frontier. I next merged all aligned contexts with consistent 45-mer prefixes within 100 bases of each other. This gives us the total number of TE insertions in a sample. From table 4.5, we can see that, in general, A/J, 129S1, NOD, NZO and WSB have more TEis than the two wild strains (CAST, PWK). Frequently there are two nearby contexts that represent the two sides adjacent to the insertion. If an insertion is also present in the reference genome, then it is possible that we are double counting the TEis as the two sides of a TE can be 10,000 bases apart. However it does not matter as the next step of my pipeline will eventually filter all the reference TEis. For this filtering, I checked if there was an annotated TE in the reference near each of the insert locations. If no TEi was present nearby, I reported those as novel TE insertions. For each sample, I found more than 2000 novel TEis that are not annotated in the reference genome. Even though, there are less TEis in CAST and PWK, I saw more novel TEis in these samples. Finally, for these novel insertions, I applied *Classifier*-II to predict the type of TE. For each sample, I found more LINE and ERVs using Frontier* and more SINEs using Frontier.

These analyses suggest that there are a considerable number of non-reference TE insertions (TEis) in these 7 common mouse strains. For each sample, LINEs and LTRs are the most common type of TE compared to SINEs. As expected the four laboratory strains (A/J, 129S1, NOD, NZO) tend to share more TEis with the reference, which is based on a laboratory strain B6, than the two wild-derived mouse strains (CAST, PWK). The degree of shared TEis with the reference is indicated by the ratio of Novel TEis to the number of Total TEis. Even though, WSB is a wild-derived strain, the number of reference TEis in WSB is similar to the other 4 lab strains. This is due to their sub-specific origins, where all lab strains and WSB are from M.m. domesticus. On the other hand, the primary sub-species of CAST and PWK are M.m. castenous and M.m. musculus respectively.

Even though the total number of TEis in CAST and PWK is far less than the other 4 strains, they tends to have more novel (non-reference) TEis. This is also consistent with the commonly known mouse phylogeny based on Single Nucleotide Polymorphisms (SNPs). Both CAST and PWK separated from the lab mouse strains about half a million years ago. Therefore, any TEi that are inserted in CAST or PWK since their separation will not not be in the reference genome and vice versa. Only exception is the NOD, which has more novel TEis compared to rest of the lab strains. Again, this is due to its high coverage. To recover the TEis that Frontier missed due to low coverage can be easily solved by adjusting the threshold in our LCP filter. For each sample, I had a hard threshold of 400, which is the number of read that contains the repeat segment. Therefore, I recommend using a coverage-based threshold so that I can find majority of the frontiers as coverage is sample-specific like the thresholds.

4.4 Runtime of Frontier Pipeline

The runtime of the Frontier TEi discovery pipeline is dominated by the first step of finding Frontier candidates. This step uses msBWT and corresponding LCP to find all reads with high-count 45 mers that when extended by 30 bases appear as normal coverage. For a short read dataset with 30x coverage and 150 bp reads, it took approximately 3 hours to find all the frontier candidates. The combined time required for building msBWT and LCP takes about 7 hours. Overall my pipeline complets in approximately 10 hours. However, one could justifiably not consider the construction time of msBWT and LCP as a fixed cost. Both of these data structures have many uses other than finding repeats. Moreover, they are useful as a lossless data compression that can be efficiently queried. And, the msBWT and LCP construction time is comparable to the sequence alignments used by other algorithms.

4.5 Summary

I proposed a novel template-free approach for finding Transposable Element insertions by identifying Frontier/split-reads and using machine learning. Unlike alignment-based methods, my learned model can be applied to a sequenced short read dataset to quickly identify all the true frontiers. As my model uses overlapping kmer counts in addition to sequence, it can find more novel TE insertions than sequence-based classifiers. Identifying novel TE insertions can be used to detect rare mutations in somatic cells that are often associated with tumorigenesis. Whereas, identifying segregating TE insertions in the germline has the potential to uncover new classes of genetic variants. In addition to finding TE insertions, our template-free method *Frontier* will be useful to identify novel TE templates. Currently, most researchers depend on databases like RepeatMasker, Dfam or RepBase for known TE templates. These databases only include the TEs that were once found active in a sample. Over the time, TEs can become dormant, and remain permanently in the genome and become active after a long period. Some samples can also have a different type of active TE which is not covered by these databases. Frontier becomes handy in those cases by enabling us to find all the sample-specific TE templates.

The idea behind *Frontier* was to provide a comprehensive catalog of TE insertions in a short read dataset. However, in comparison to ELITE 3, *Frontier* falls short, with regard to recall, when a TE template is known in advance. This is because of my assumption behind the frontier finding algorithm, which is, TE sequence has always high count/coverage. At present, I can only search for repeats that occur more than a given threshold. Whenever TE insertions are mutated, the mutated versions may not have a high count near the insertion point. Therefore, in the future, I plan to modify our repeat searching algorithm to allow for some mutations from a highly repeated k-mer found elsewhere in the dataset. As discussed before, Frontier is still a better option finding TE templates. That's why, in my next chapter 5, I discuss how the combination of these two pipelines can be used to comprehensively catalog TE insertions in a population. I mainly focus on one type of TE which is called Endogenous RetroVirus (ERV). For this TE type, I collect all the templates using Frontier, and feed it to ELITE as input. Finally, I catalog all the ERV insertions in a population called Collaborative Cross (CC) which has more than 100 samples.

CHAPTER 5: DIVERSITY OF TRANSPOSABLE ELEMENT INSERTIONS IN A POPULATION

The goal of this chapter is to show how my algorithms for identifying Transposable Element insertions (TEis) can be used to assess their diversity in a population. These algorithms ELITE and Frontier, from chapters 3 and 4 respectively, alone can only be used to identify TEis in a given sample. With respect to whether a detected TEi segregates (different than rest of the individuals in a population), they can, at best, determine whether a TEi is in the given reference genome or not. However, collectively looking at the TE insertions in a larger population and their patterns of sharing facilitates phylogenetic and genetic analysis. Typically, only simple mutations such as Single Nucleotide Polymorphisms (SNPs) and Insertions/Deletions (INDELs) are used in these types of analyses. On the other hand, the mutation rate of TE insertions can vary significantly over time based on epochs of TE activity, how many active TEs are in the genome, and their rate of suppression. In order to understand these important biological and evolutionary processes, I applied ELITE with some additional features to the problem of population-based genetic analysis. To understand the TE inheritance and diversity patterns, I used a genetic reference mouse population called the Collaborative Cross (CC). In the CC all the strains are derived from 8 common ancestors. For each sample in this population, I ran ELITE using 9 TE templates specific a particular type of TE i.e., Endogenous Retrovirus (ERV). ERVs contribute almost 30% of the transposable elements in mouse and comprise about 10% of the reference genome. They are also the most studied active TEs known to be present is mouse genome. Thus, focusing on ERVs increases the chance of finding multiple recent insertion events, key for conduction genetic analyses. I chose ELITE over Frontier because ELITE can comprehensively find all the TE insertions when a template is already known. Another unique feature of ELITE is that it returns probe sequences for detecting both the presence and the absence of an insertion, which can then be genetically mapped across a population. Therefore, I use CC population to validate where TEis segregating in the founder are located in the genome and shared by subsets of descendants. Further we can also classify whether each identified ERV is fixed (present in every sample), segregating (present in a subset of samples) or *de novo* (present in a single strain, suggesting a new or recent insertion).

5.1 Datasets For Analyzing TE Diversity

In this section, I describe the datasets used for analyzing TE diversity pattern in a population. For this study, I used ELITE to identify TE insertions directly from high-coverage short-read sequencing data. All the sequenced short reads correspond to a single samples from an inbred genetic reference population. Since, ELITE is a template-based method, I also collected a set of ERV TE templates based on Frontier-candidates classified as Frontier*/ERV repeats by Frontier. In the following I describe details of the two data sets that were used in our analysis. First I characterize the Collaborative Cross (CC) genetic-reference population, then I discuss the ERV templates used. Both of these play a significant role in our analysis of TE diversity.

5.1.1 Population: *Collaborative Cross*

In this chapter, we characterize the ERV landscape in a population called Collaborative Cross (CC). The CC is a large panel of recombinant inbred mouse strains derived from a common set of 8 inbred laboratory mouse strains, which are A/J, C57BL/6J, 129S1/SvImJ, NOD/ShiLtJ, NZO/HlLtJ, CAST/EiJ, PWK/PhJ, and WSB/EiJ, which are referred to as the CC founders. In the remainder of this chapter I will use the following abbreviated names to distinguish these strains respectively: AJ, B6, 129S1, NOD, NZO, CAST, PWK, and WSB. By convention, the genotypes of genomic segments from these 8 strains are also often denoted by AA, BB, CC, DD, EE, FF, GG, HH respectively. Among these founders, five are classical inbred strains (AJ, B6, 129S1, NOD, NZO), and 3 wild-derived inbred strains (CAST, PWK, WSB). Together, these founders capture about 90% of the known genetic diversity present in laboratory mice (Roberts

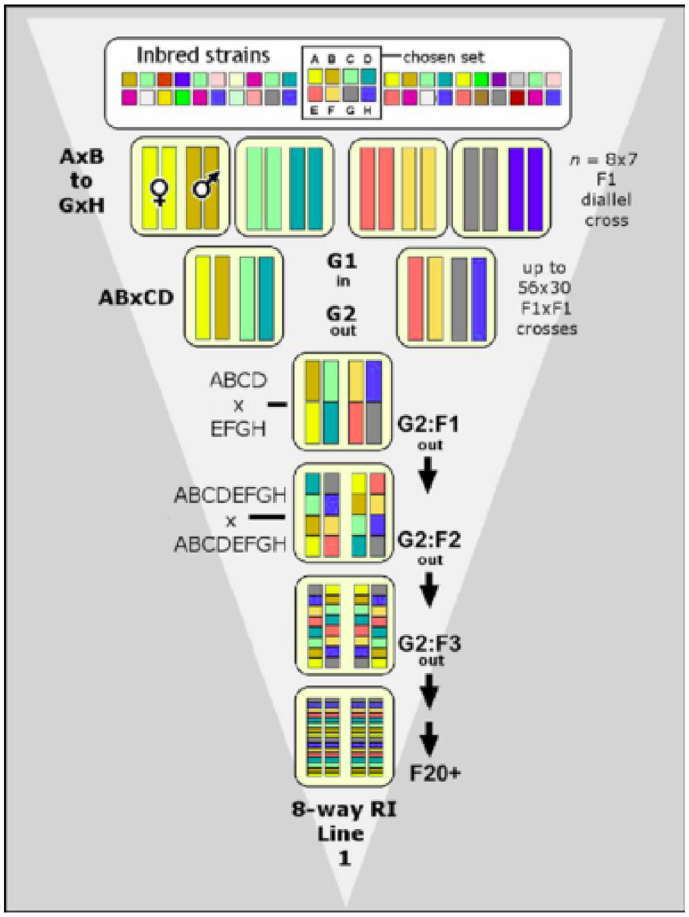


Figure 5.1: The funnel breeding scheme used to derive each CC strain from 8 common inbred mouse strains. The 8 founder haplotypes are represented by A for A/J, B for B6, C for 129S1, D for NOD, E for NZO, F for CAST, G for PWK, H for WSB. An independent breeding funnel like the one shown above was set up for each extant CC strain. After 20+ generations of breeding, they arrived at the current CC population. This figure is from Churchill et al. (2004)

et al., 2007). Three different subspecies of mouse are also represented in this set of founders. Those are *M. musculus Domesticus* (The dominate subspecies of all the common lab strains and WSB), *M. musculus Casteneus* (largely due to the wild-derived CAST strain), and *M. musculus Musculus* (largely due to the wild-derived PWK strain). However, both wild-derived and lab strains have small intervals of introgression from other subspecies (Yang et al., 2007). The standard accepted mouse phylogeny between these founder strains, based on SNPs (single nucleotide polymorphisms) is shown in 5.2.

An unique breeding scheme was adopted so that each CC sample in this population is independent, meaning that there is no common ancestor more recent than the original 8 founders.

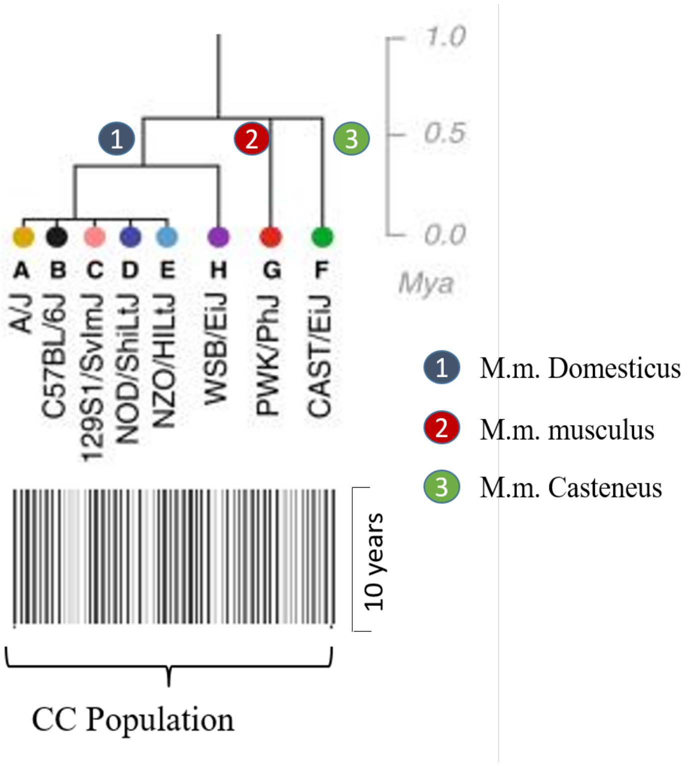


Figure 5.2: Consensus mouse phylogeny of the 8 CC founders: Three subspecies are represented (1) M.m. domesticus, (2) M.m.musculus and (3) M.m.casteneus. CAST, PWK, and WSB are wild-derived mouse strains. Of these CAST and PWK are predominantly different subspecies the other 6. Their wild caught founders had been separated from the other mouse lineages for more than .5 million years ago. These two subspecies are *M.m. musculus* and *M.m. castaneus* respectively. The genomes of the lab strains and WSB belong primarily to *M.m. domesticus* subspecies. The 5 lab strains were derived from a small population of fancy pet mice in the early 1900s and are primarily from the subspecies *M.m. domesticus* like the wild-derived strain WSB. Furthermore, many of the inbred strains show signs of introgression from one of the other subspecies.

This combination of a known founder set and a documented pedigree makes the CC uniquely powerful for studying genetic drift due to recent mutations. There are several advantages of choosing this particular population. Such as:

1. The CC is a genetic mapping population that is especially suitable to biomolecular traits. This allows us to confirm the presence/absence and position of any segregating ERV insertion.

2. The presence of multiple biological and technical replicates helps to identify the false positive and false negative rates. This is especially important in low coverage genomic regions.

3. The genetic diversity and different subspecific origins of the CC founders allow us to place our TEi findings within a phylogenetic context.

4. It is also possible to measure the mutation rates for different ERVs.

5. This population is already a popular resource widely used in research. The genomes of CC are known, replicable, and stable(Consortium, 2012).

5.1.2 TE Templates: *Endogenous RetroVirus (ERV)*

I use ERV templates to identify TE insertion sharing patterns within the CC population. In our lab, we initially found a CC strain (CC055M) who had observable phenotype that was different than other CC strains because of an ERV insertion. This motivated my interest in finding more ERV insertions in-particular trying to determine if any TEs are active. In an initial prototype of ELITE, I used the ERV template that was found in CC055M, which was ERVB7_1-LTR_MM according to a TE database called RepBase (Bao et al., 2015). Later I added 5 more ERV templates from RepBase, those are: ERVB4_2B-LTR_MM, IAPEY3C_LTR, MERVL_LTR, RLTR1IAP_MM, and RLTRETN_MM. These 5 templates were chosen quite randomly to get some initial understanding of ERV landscape in CC population.

However, after running Frontier 4 on the 8 founder sets, I discovered that there are many other types of TEs other than those 6 ERVs initially considered. Therefore, a vast majority of those TEs most likely be present in the CC too. For phylogeny analysis, this is an unique opportunity to see how these TE insertions are passed on to the next generations including the recently developed CC. Therefore, I revisited the frontier pipeline to select a new set of templates to supplement my initial set. As discussed before, Frontier automatically identifies TE insertions without templates. After finding the insertions, it then classifies the TE type into 4 major categories: 1.

LINE, 2. SINE, 3.Alu, 4. ERV. However most TE types, except for the ERV, do not have well-structured and stable TE boundaries. LINE, SINE and ALUs have variable number of poly A at their 3' insertion site. Also, it is common for non-ERV TEs to have their 5' sequence truncated, which makes it even harder to identify the insertion location to the single base-pair level. This is another reason why I focused on ERV insertions for this study. ERVs also represent almost 10% of the mouse genome and are known to be the most active type of TE in mice.

To prepare TE templates, I collected all the TE sequences from Frontier's output where the inserted TE type is ERV. Then with the help of two available online TE libraries (Repbase and RepeatMasker), I clustered all those TE into 9 different ERV consensuses. These 9 consensuses are finally used as templates which represent all the ERVs in the founders of our CC population. Even though there are many other types of ERV, from each template I chose seeds in a way that automatically covers all the related ERV types. In fact, the proximal TE of ERVB7_1-LTR_MM and RLTRETN_MM have a high level of sequence similarity that leads me to use a single proximal seed for both ERV types. Using one seed for both types also reduced the runtime of ELITE pipeline. For all the other ERVs, I chose one seed from the proximal LTR sequence, and one from the distal LTR sequence.

5.2 Building Probes For Genetic Mapping

Recall that, ELITE outputs two different types of probes for each insertion i.e., 1) TE presence probe, 2) TE absence probe. For each TEi, ideally there would be two presence probes, one for the proximal end (context + proximal TE) and one for the distal end (distal TE + context). However, contexts are not always unique, thus sometimes results in one-sided TEi. In One sided TEis, only one context, either proximal or distal maps uniquely. In those cases, ELITE was unable to construct the TE probe on the unmapped end. But it is possible to identify the distal TE probe when proximal TE probe is known or vice versa. Previously ELITE was also unable to find TE absence probe when the TE is present in the reference genome. Finding the absence probes is trivial in a sample if the insertion is heterozygous or TEi is not in the reference (because a single

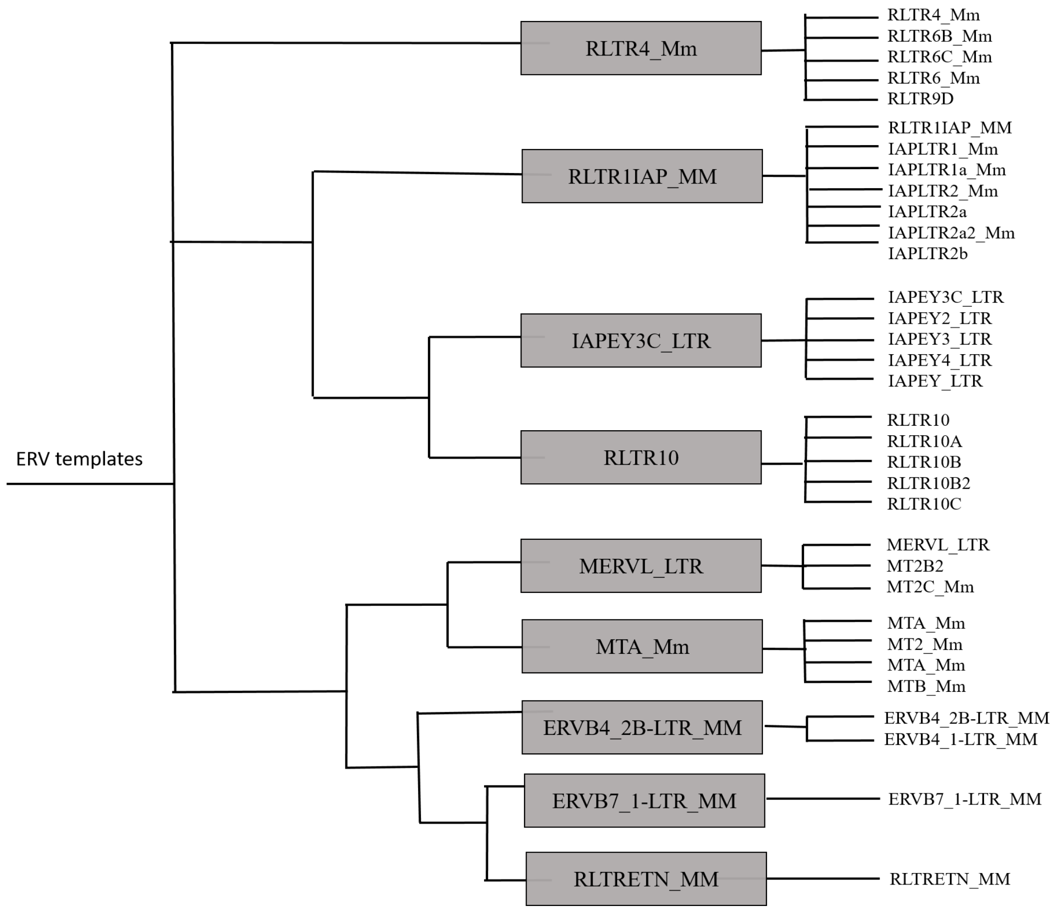
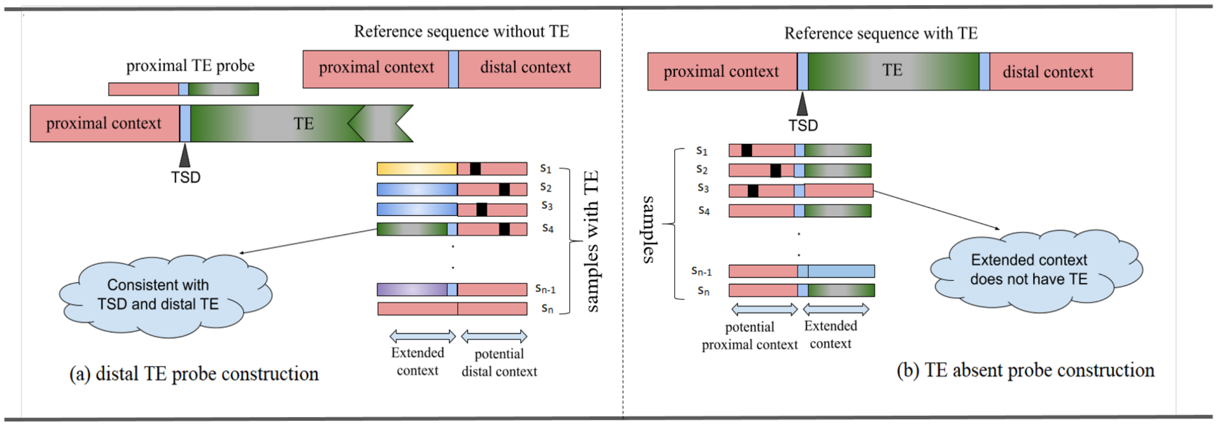


Figure 5.3: ERV templates used for running ELITE on Collaborative Cross population: There are 9 main ERV templates from I selected proximal and distal seed. However, after running Frontier on CC founders, I saw around 20 different type of ERV insertions. To make our ELITE pipeline efficient, I combined multiple templates into one. This combination is based on the sequence similarities, and done by creating a phylogeny tree. This process reduced the ERV templates to 9 distinct ones.

sample has instaces of both sequences). It is also possible to identify absence probes using a large population where TEi is absent in at least one sample. Therefore, I revisit the ELITE pipeline and extended it further to get more probes which I later use for genetic mapping.

5.2.1 Finding TE present probes

If ELITE fails to identify both proximal and distal TE probes, I make a second attempt using the expected genomic context obtained from the reference genome. This is possible only when the TEi is not present in the reference. If a proximal TE present probe is missing, then I grab



Reference sequence without TE
proximal TE probe
proximal context
distal context
proximal context
TE
TSD
s1
s2
s3
s4
sn-1
sn
samples with TE
Consistent with TSD and distal TE
Extended context
potential distal context
(a) distal TE probe construction
Reference sequence with TE
proximal context
TE
distal context
TSD
samples
s1
s2
s3
s4
sn-1
sn
Extended context does not have TE
potential proximal context
Extended context
(b) TE absent probe construction

Figure 5.4: (a) TE present probe construction: this figure shown how I construct the distal TE probe when the proximal probe is known and TEi is not in reference. At first, I use the reference genome to see how a non-TE sequence look like. Then in all the samples with TEi, I look for any sequence that looks like the distal context. After that, I extend the distal context towards the TE. Finally the extended context which is consistent with the distal TE and TSD are used as distal TE probe. (b) TE absent probe construction when TEi is present in reference. At first, in all the samples without TEi, I look for any sequence that is similar to the proximal context. I then extend the context, and keep only the one that does not contain TE and the distal context is consistent with the distal TE probe.

the sequence (i.e., expected proximal context) right before the distal context from the reference. Then I look for this expected proximal context in the given short read dataset using the samples's msBWT. I allowed for a maximum 2 mutation per 25 bases. Once the proximal context is found, I extend it to the right to get the proximal TE sequence. If the context is not unique, there will be multiple paths. However, I only keep the one that is consistent with the given TE template and target site duplication (TSD).

5.2.2 Finding TE absent probes

Whereas a TE presence probe confirms the presence of an insertion, a TE absence probe confirms its absence. ELITE will not find a TE presence probe in a sample for two reasons (a) the TEi is not actually present (b) algorithm has limitations. If a TE presence probe is missing because of the later one, then it's not accurate to say that sample does not have TEi. Thus to say in confidence that a TEi is absent, I need a TE absence for each sample who does not have it. Finding TE absent probe is trivial if the reference genome does not have the same TEi. Samples who don't have the same TEi may not have the same absent probe due to mutations (SNPs or INDELs) in in the nearby genomic context. Therefore, I extract the absent probe from the reference and look for that in our given short reads by allowing some mutations. Finding absent probe when the TEi is present in reference is not as straight-forward. However in a population setting, if at least one sample does not have the TEi, then it is possible to identify the absent probe. To do this, I extend all the contexts (both proximal and distal) towards the TE sequence. Samples missing the TEi, will have regular genomic context instead of TE. Thus I use that as the absent probe.

5.3 Combine TEis in a population

Since ELITE runs on individual samples, it is important to aggregate all the results from the given population to understand the TEi diversity patterns. I, at first, consider all the different locations where a TEi is found in any sample. We assign a unique TEi_id for each location. I have at least one TE present probe for each insertion. However there might be more than one. Sometimes TEs or nearby contexts get mutated independent of the insertion, so the exact probe might not work for all the samples who have the insertion at the same place. However, having insertion in the same place corresponds to the same event. Thus, for one TEi_id, it is possible to have multiple TE probes. I also have implemented this unique feature which tells us how a probe looks like in the absence of an insertion. Absence probe in any sample confirms that the sample indeed lacks the TE insertion, and rejects the possibility of false negative with proof. It is possible to have no absent probe for a TEi_id if all the samples in a population have the same corresponding TEi. Therefore, for each TEi_id, unless it is possible to have more than one probe. Using both TE present and absent probe, I produce a genotype-like call for each sample for each TEi_id annotating whether it has the insertion or not. For each location, I also provide some additional information like the Strain Diversity Pattern (SDP) based on founder set, gene annotation i.e. whether the insertion is intergenic or within a gene, etc. Below I provide a detailed list of all the fields that are associated with each TE insertion:

1. **TEi_id:** An unique id for each independent insertion. If more than one sample has a TEi exactly at the same place, then are not independent. If fact, it suggests that, this insertion was inherited from a shared common founder. Thus we have a single TEi_id based on the insertion location.

2. **chromosome, position, strand:** The position in the genome where the insertion was found. I use the location directly from ELITE's output which can identify the exact location of the insertion. Additionally, strand tells us the orientation of the ERV, whether it was inserted in the forward or the reverse DNA strand.

3. **ref_te:** A flag ref_te indicating whether the insertion is also present in the reference genome. This field allows us to quickly filter all the insertions that are already annotated in the reference. As discussed previously, the mouse reference genome is based on the B6 strain. Therefore, if any B6 sample has a non-reference insertion, that probably indicates a recent event.

4. **before and after:** Before and after is the 25-mer probe that is present before and after the inserted TE sequence. These are potentially the proximal and distal contexts. However, one single TEi can have multiple proximal or distal contexts. So I use only the contexts that are from reference. This is consistent with the fact that our TEi' location is also based on the reference genome.

5. **state:** It represents the TEi's state based on the sharing pattern. If an insertion is present in every sample in a population, then I call it *fixed*. If it's present in a subset of samples, then I call it *segregating*. Finally if it is present only in one sample or a set of samples who are biological replicates, then I call it *de novo*.

6. **genotype call:** Genotype calls are represented by one of the following bits: N,0,1,H. Our calls are based on the TE present and absent probe. If a sample has both types of probes, then I call it an H (H for heterozygous). If it only has a TE present probe, then it is homozy-

gous for the TE and denoted by 1. If it only has a absent probe, then its denote by a 0. If a sample is missing both TE present and absent probe, then I call it an N. A sample can have N if the genomic sequence around that location is deleted in that sample. It can also be due to low coverage or high mutation rate around that area.

7. **SDP:** SDP referes to Strain diversity pattern. I am showing the diversity patterns of TE insertions in the 8 CC founders. So this is a 8 bit pattern for 8 CC founder strains AJ, B6, 129S1, NOD, NZO, CAST, PWK and WSB respectively. Each bit can be 0,1,N or H. This is similar to the genotype calls as mentioned before.

5.4 ELITE in the Collaborative Cross

My goal was to identify all the ERV insertions in CC population. Using Frontier's output, I collected all the TE templates where the TE type is ERV/LTR. I found total of 9 different ERV classes. From these classes I derived ERV templates for each end of the LTR. Using these templates, I ran ELITE on each sample of the CC population. I followed all the steps of ELITE pipeline as described in 3. First I built an msBWT for each sample. Then I ran ELITE with some adjustment to its parameters. Recall that, ELITE has two threshold parameters: $t1$ and $t2$ where $t1$ is the minimum number of reads containing the TE probe, and $t2$ is the maximum edit distance that is allowed in a TE sequence with the original TE template. While running it on CC, we set $t1 = 4$, and $t2 = 1$ mutation per 10 bases. I then picked two seeds from each ERV template: one seed from the proximal and one from the distal template. By extending all the seeds, I found all the different versions of ERVs. Finally by extending the ERVs, I found their genomic contexts. In this case, I only extended the ERVs by 25 base, i,e,. context length is set to 25. I then filtered all of the sequences where the context does not uniquely map to reference genome. Using these unique contexts, I annotated all the locations in terms of (chromosome and position) where at least one sample has an ERV insertion. Then for each location, I found the alternate absent probe for the samples who don't have the ERV insertion. Some ERV insertions were present in all the samples and, thus, don't have an ERV absent probe. There are also some cases where a sample doesn't have any ERV present or absences probes. This could be caused by either a genomic deletion or a highly divergent variant. Finally, I merged all of the positions between samples based on their location proximity. For each position, I annotate the presence or absence of ERV in each sample.

Overall, we found a total 40,458 ERV insertions (ERVis) in 101 samples including CC samples and their founders. Among these ERVis, some were present in every sample. We call these fixed ERVis in the CC population. Some of the ERVis are present in a subset of samples, which are called segregating. Finally, I found some insertions that were private to only one sample or shared by only biological or technical replicates, which we call *de novo*. Biological replicates are the samples that were derived from the same founder, similar to monozygotic twins. On the other hand, technical replicates are actually the same samples that was sequenced multiple times. Technical replicates are useful for validating ELITE's findings, because all the insertions in one sample should be present in its technical replicates.

Table 5.1 shows the total number of ERV insertions we found in CC population using ELITE. I used 9 different ERV templates that I obtained from Frontier. I further categorized the number of insertions as fixed, segregating and *de novo*. Fixed ERVis are present in every CC strain including its 8 founders. This indicates that fixed ERVis are old meaning that insertions took place a long time ago when the founders had a common ancestor. Those insertions were passed on to their descendants including the CC founders and the CC. Around 36% of the total ERVis are fixed in CC. However, this also depends on the ERV type. MTA_mm contributes to 70% of the fixed ERVis whereas RLTR4_MM has none. For all ERVs other than MTA_mm and ERVB4_2B-LTR_MM, there are more segregating ERVis than fixed ones for each template. More than 33% of the segregating ERVis came from RLTR1IAP_MM and 30% from MTA_mm. The number of *de novo* also depends on the ERV template. ERVB7_1-LTR_MM contributes the majority of *de novo* insertions. In fact, it has comparable number of fixed (204) and *de novo* (166) insertions.

ERV	Fixed	Segregating	*de novo*	Total
ERVB4_2B-LTR_MM	559	376	0	935
ERVB7_1-LTR_MM	204	674	166	1,044
IAPEY3C_LTR	631	1,304	0	1,935
MERVL_LTR	1,650	2,833	35	4,518
MTA_mm	10,375	7,723	153	18,251
RLTR10	514	3,110	19	3,643
RLTR1IAP_MM	270	8,470	100	8,840
RLTR4_MM	0	307	16	323
RLTRETN_MM	387	558	24	969

Table 5.1: Total number of ERV insertions in CC population found by ELITE. I broke down the numbers based on 9 different ERV types that were used as templates. The number of ERV insertions varies for different templates. MTA_mm is the most common ERV type, which alone has about 50% of the total ERV insertions. On the other hand, RLTR4_MM has only 323 ERVis. I also categorized the total number of ERVis based on their sharing pattern. Majority of the ERVis for ERVB4_2B-LTR_MM and MTA_mm are present in every sample thus are fixed in CC. On the other hand, ERVB7_1-LTR_MM, RLTR1IAP_MM and RLTR4_MM have more segregating ERVis which are shared by only a subset of samples. The most recent events or *de novo* also varies depending on the ERV type. Almost 40% of the ERVB7_1-LTR_MM insertions are *de novo*. However, there are some ERVs like ERVB4_2B-LTR_MM and IAPEY3C_LTR, where I found no *de novo* insertions, indicating that they have not been recently active.

This indicates ERVB7_1-LTR_MM is the most active ERV in CC population because all of these TE insertion events took place during or after the breeding of CC.

5.5 Segregating ERV insertions in CC

Segregating insertions are present only in a subset of samples. The sharing pattern of this type of insertion allows us to assign a phylogeny to each insertion. Since all the samples of the CC population came from 8 common founders, the segregation pattern of ERV insertions in the CC should correlate with the SNP segregation pattern seen in the founders. The differences in the lengths of tree branches might help to identify when (in time) and where (in what subspecies branches) the ERV type was active. Therefore, to understand this sharing pattern, I build phylogenetic tree based on the sharing patterns of ERV insertions in each founder. I have already seen that different ERVs have different sharing patterns. That's why I chose two templates ERVB7_1-

LTR_MM and RLTRETN_MM to see how they differ in terms of segregation. The first template, ERVB7_1-LTR_MM, has 1044 ERVis, and among them 674 are segregating. On the other hand, RLTRETN_MM has a total of 969 ERVis, and 558 of those are segregating.

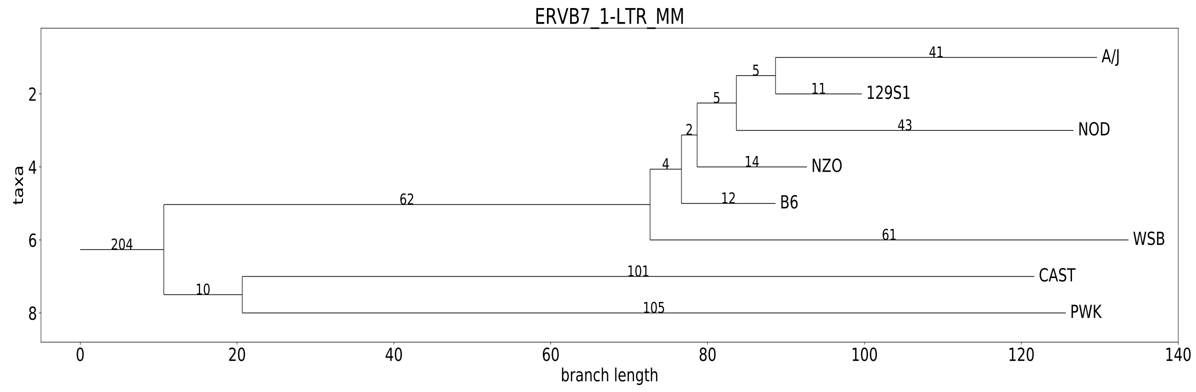


Figure 5.5: Phylogenetic tree based on the presence of ERVB7_1-LTR_MM insertions in the 8 CC founder strains. This tree is constructed using maximum parsimony, which allows minimum mutations. A total 204 ERVis are present in every founder, as shown by the root label, which is not shown to scale. Among the 674 segregating ERVis, 476 are supported by this tree. As we can see, CAST and PWK are well separated from the other 6 strains, which is consistent with the separation of the subspecies. CAST and PWK share 10 ERVis suggesting that they branched together from the *Mus Musculus* common lineage before separating into the *M.m castaneous* and *M.m. musculus* subspecies. On the other side of the tree, WSB is also separated from the common lab strains. Among the lab strains, we see longer leaf branches in A/J and NOD as compared to B6, 129S1 and NZO. However, CAST and PWK both have even longer leaf branches, suggesting they have many insertions that are not present in any other strains. The relatively long leaf nodes of this ERV tree suggests that ERVB7_1-LTR_MM has been, and continues to be active in *Mus Musculus*. Indeed, if the branch lengths were corrected for time, it would appear that the rate of activity has been increasing.

I show the segregation pattern of ERV insertions for ERVB7_1-LTR_MM in figure 5.5. Here sharing patterns are based on the ERV insertions in CC samples. Since they came from the 8 common founders, I use their founder/origin at the locus of insertion for understanding the diversity. These founders represent 8 mouse strains (A/J, B6, 129S1, NOD, NZO, CAST, PWK, WSB). This figure is consistent even if I use only the 8 founder strains, which is another validation of ELITE's findings. Other than the root, all the branches are drawn according to scale. There are 62 ERVis that are present in all the lab strains and WSB. Since WSB is also from the same subspecific origin (M.m. domesticus) as the lab strain (A/J, 129S1, B6, NOD, NZO), they have many common insertions as expected. On the other hand, CAST and PWK who got separated from the rests about half a million year ago, share 10 common insertions. All the lab strains share 4 ERVis. There are also some small branches under the lab strains, like there are 4 ERVis that are common to A/J and 129S1, 5 ERVis are common to A/J, 129S2 and NOD, 2 ERVis are common to A/J, 129S2, NOD and NZO. The leaf branches show the ERVis that are present only in that founder. Lab strains have less private ERVis compared to CAST, PWK and WSB. Among the lab strains, A/J and NOD have more ERVis than B6, NZO and NOD.

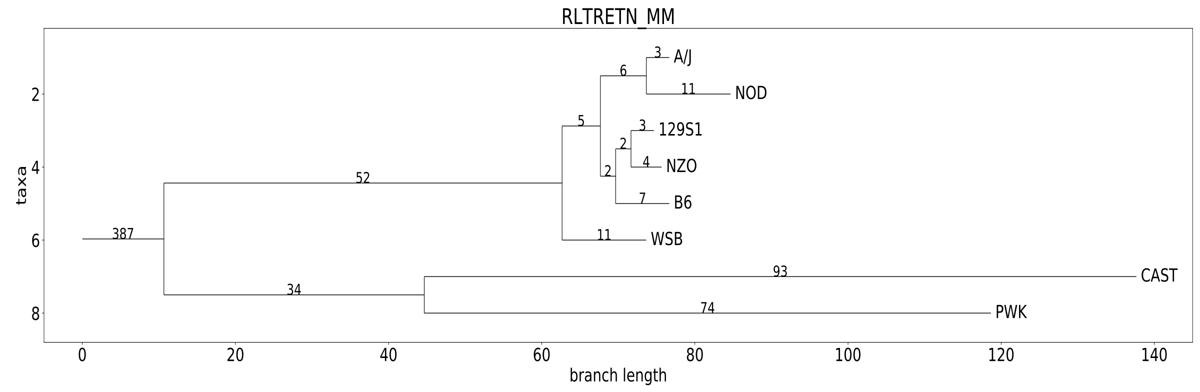


Figure 5.6: Phylogenetic tree based on the presence of RLTRETN_MM insertion in 8 CC founder strains. Tree is constructed using maximum parsimony, which allows minimum mutations. Total 387 ERVis are present in every founder, shows by the root. Among the 558 segregating ERVis, 296 are supported by the tree. As we can see, CAST and PWK are well separated from the other 6 strains. Together they share 34 ERVis. On the other side of the tree, WSB is also notably separated from the common lab strains. Among the lab strains, we see longer leaf branches in A/J and NOD compared to B6, 129S1 and NZO. However, CAST and PWK both have far longer leaf branches, suggesting they have many ERV insertions that are not present in any other strains.

Next I examine the segregation pattern of ERV insertions of RLTRETN_MM as shown in figure 5.6. Similar to the previous tree 5.5, the sharing patterns here are also based on the ERV insertions in 8 CC founders that represent 8 different mouse strains. Even though the topology of the tree looks similar to the previous one, the leaf branches here are considerably smaller. CAST and PWK have 93 and 74 private ERVis respectively. On the other hand, all the strains have less than 10 ERVis private to them. However, all the strains with M.m. domesticus subspecific origin (lab strains and WSB) shares 52 ERVis. Again CAST and PWK share 34 ERVis which were likely inserted after the separation of *M.m. domesticus* some 500,000 years ago, but before the separation of the *M.m castaneous* and *M.m. musculus*. With the exception of the internal branches among the common lab strains, both of the ERV-based phylogenetic trees are topologically consistent. However, the RLTRETN_MM branches are considerably shorter than those of the ERVB7_1-LTR_MM tree. This suggests that ERVB7_1-LTR_MM has been more recently active. Both ERV trees are consistent with the with the common mouse phylogeny based on SNPs, and they aid in resolving details such as the shared ancestry of *M.m castaneous* and *M.m. musculus*.

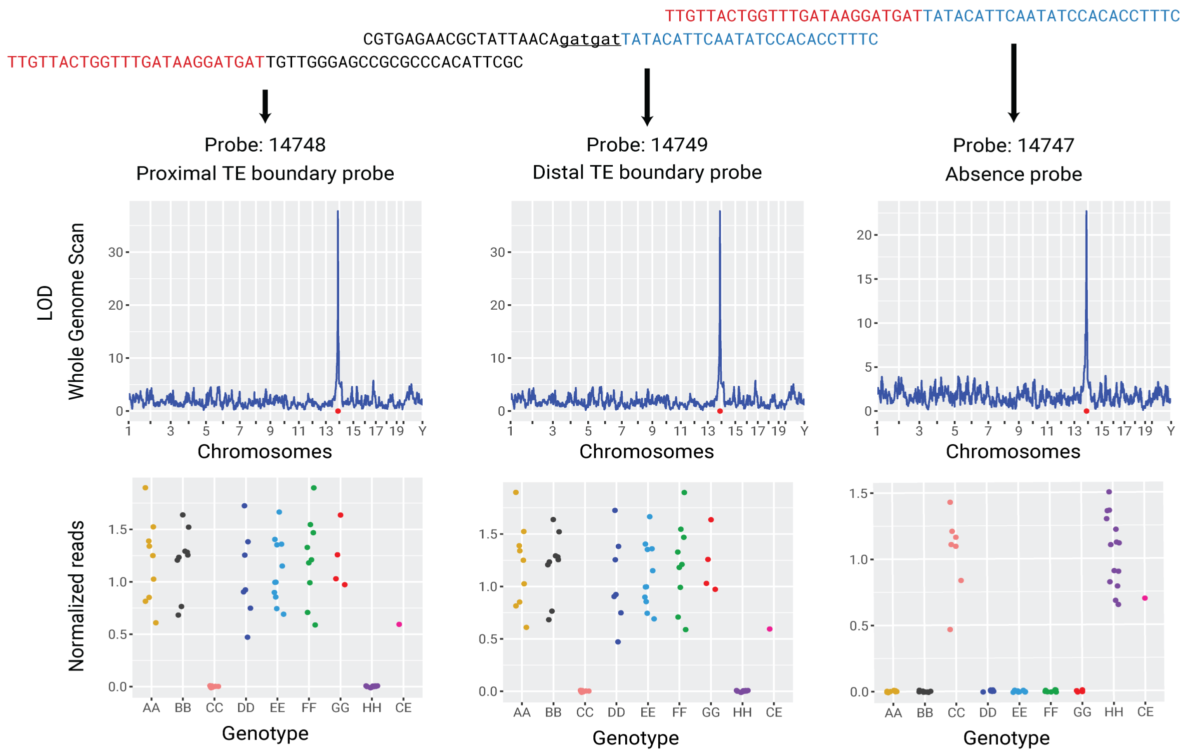


Figure 5.7: An example of QTL mapping for TEi_id 14747. This TEi_id has 3 probes, a proximal TE probe, a distal TE probe and an absence probe. For all the probes, LOD scores got a pick at the end of chromosome 13. This is also ELITE's predicted insertion location. Three plots at the bottom shows the allele frequency for those probes at that locus. Except for CC and HH, all the other haplotypes have high allele frequency for both proximal and distal TE probes. On the other hand, CC and HH both have high allele frequency for the absence probe, and the rests have zero. This indicates, TEi is present in everyone except for CC and HH. Absence of TEi in CC and HH is further confirmed by the QTL of the absence probe.

5.6 Validating Segregating ERV insertions

Genetic mapping (i.e., quantitative trait loci (QTL) mapping) was used to verify ELITE's findings. QTL mapping allows us to identify genomic regions that are correlated to the genetic segregation patterns of a trait. We mapped QTLs by detecting which genetic segregation pattern (based on a set of SNP markers) correlated to the patterns of segregating TEis. For our CC population, I used the SNP markers from a well-known, widely used genotyping array called GigaMUGA (Morgan et al., 2016). Markers in this array are distributed throughout the genome, and are capable of distinguishing the haplotypes of the 8 CC founder strains. In addition to confirming the location of TE insertion, QTL mapping also allows us to verify the list of samples who should and should not have an insertion based on their founder haplotype at in the region of the insertion. First, the insertion location of TE is expected to match a QTL peak. Second, the allele effects observed in the region of the QTL peaks are expected to verify strains where the ERV was detected or not as well as the predicted founder haplotypes.

I independently map the probe counts for each probe. Then I combined all the probes that are from the same insertion event. However before doing QTL mapping, I applied several filters on both TEi, TE probes. These filters are not mandatory and biologists can apply or remove any filter based on their type of analysis. Here are post-processing steps applied to the output of ELITE:

1. We perform QTL mapping for each probes of a segregating TEi. Similar to TEi, segregating probes are the ones where a subset of samples have the probe, and the rest don't have it. To be able to apply QTL, there has to be more than one group who are genetically different. Sometimes, it happens that a TEi is present in every sample, and it corresponds to the same TE present probe. These fixed and non-segregating probes are filtered before QTL mapping.

2. We normalized the probe counts for each sample based on its read coverage. These normalized counts are used for QTL mapping instead of the absolute count. If the normalized

count does not corresponds to a single copy, the ERV is removed from further consideration. Note that this does not imply that the ERV insertion is a false prediction, but it suggests that mapping will be ambiguous. Note that our selection of probes verifies that the "context" portion is unique in the reference, but that might not be the case in other genomes. To simplify our analysis, we did not considering any Copy Number Variations (CNVs) and only focusing on the locations that is unique for each sample in the population.

3. Any TEi probe on chromosome Y was filtered. Since Y chromosome does not recombine, we cannot map the exact location of a TEi. Additionally, chromosome Y is only present in male samples, and only a fraction of our population is male. Therefore it was ignored in this analysis.

4. We used normalized probe counts as the observed trait for QTL mapping. QTL returns LOD scores (Log of Odds) for the whole genome (specifically at the markers' position) which indicates the chances of having a QTL at that position. If the LOD score at ELITE's predicted location is sufficiently high, the probe was kept. For our analysis, we considered a QTL as likely if its LOD was greater than 8 and less than 200.

5. We removed any TEi_id that does not have a TE probe. Because of all the previous filtering steps mentioned above, it might happen that all the TE present probes of a particular TEi got filtered and removed. Thus that TEi_id no longer represents any insertion event.

Figure 5.7 shows an example of QTL mapping. ELITE found an ERV insertion of type RLTR1IAP_MM, denoted by TEi_id 5098, at chromosome 13 position 108559536. For this insertion, ELITE generated two TE present probes, one from the proximal side: TTGTTACTGGTTTGATAAGGATGATTGTTGGGAGCCGCGCCCACATTCGC (probe_id 14748) and one from the distal side: CGTGAGAACGCTATTAACAgatgatTATACATTCAATATCCACACCTTTC (probe_id 14749). For the same TEi one absent: TTGTTACTGGTTTGATAAGGATGAT TATACATTCAATATCCACACCTTTC (probe_id: 14747) is also predicted by ELITE. Two TE present probes share a common target-site-duplication (TSD) from the genomic context, gatgat, which is adjacent to the TE's LTR sequence. The distal TE probe's TSD is shown in lowercase. Using the normalized counts as traits, we performed QTL mapping on the samples of CC population. Counts were normalized by their coverage as mentioned earlier. As we can see from the figure, all the probes have a high LOD score at the end of chromosome 13 around 108 megabases, which is consistent with ELITE's prediction. Allele frequencies in both TE probe also suggest that, ERVi is present in every founder haplotype except C and H. On the other hand, the allele frequency of the absent probe shows that ERVi is absent in C and H. Therefore, all three probes independently confirm the ERV insertion predicted by ELITE.

	Total number of probes	112683
	Segregating Probes	80900
Filter 1	Potential CNV	-464
Filter 2	Duplicate probes	-745
Filter 3	Chromosome Y	-43
	Probes left for validation	79648
	QTL validated probes	74443

Table 5.2: Summary of ELITE's probes validation: ELITE reported a total of 112683 probes. However, QTL mapping is limited to validating only segregating traits, which left 80,900 probes. After applying additional filters, we arrived at a total 79,648 probes and among these 74,443 (93.5 %) were validated by QTL mapping. The remaining 6,457 were removed for variety of reasons, Some did not fall witihn the LOD confidence interval, others did not map to the location predicted by ELITE, and others still did not segregate among founders as expected.

Figure 5.2 shows the summary of QTL mapping on ELITE's probes. There are total 112,683 probes reported by ELITE. I eliminated some probes that are not segregating. First, probes that became fixed in CC population. Second, probes that belongs to *de novo* insertions. Therefore, I only keep only the segregating probes for QTL, which gives us 80,900 probes. Then I apply filter 1, which removed 464 probes that contain potential copy number variation (CNVs). Here normalized probe counts are used to find CNVs where a sample has count higher than the expected coverage. Then I remove 745 probes are duplicated which are left due to merging error. Next, I apply filter 3 to remove 43 probes that are in Y chromosome. All the filters left me with 79,648 probes. QTL was run on these probes and 74,443 were consistent with ELITE's findings. After that, I remove any singleton probe. These 74,443 probes represent 15,000 distinct ERV insertions in CC. So overall among the 19,922 TEis made up by 74,443 probes, 15,000 are validated using QTL. Moreover, allele frequencies are also consistent with the samples who are supposed to have ERVi in that location according to ELITE.

5.7 *De novo*

So far I have talked about the insertions that are shared by some samples in the CC population with a common founder. However, new mutations can occur in any CC line that are not inherited by any founder. I call this new mutations *de novo*. In the context of CC, we can define *de novo* as an insertion that is present only in one sample or shared by its biological replicates from the same CC line. In CC population, de novo means either the insertion took place as recently as when the sample was born or 10 years ago. Since its been 10 years since the breeding of CC, any *de novo* in CC can not be older than that thus can be referred to as recent events.

ERV	Total	*de novo*
ERVB4_2B-LTR_MM	935	0
ERVB7_1-LTR_MM	1044	166
IAPEY3C_LTR	1935	0
MERVL_LTR	4518	35
MTA_mm	18251	153
RLTR10	3643	19
RLTR1IAP_MM	8840	100
RLTR4_MM	323	16
RLTRETN_MM	969	24

Table 5.3: Total number of *de novo* ERV insertions in CC population found by ELITE. We broke down the numbers based on 9 different ERV templates. The number of *de novo* varies for different templates. Almost 40% of the ERVB7_1-LTR_MM insertions are de novo. However, there are some ERVs like ERVB4_2B-LTR_MM and IAPEY3C_LTR, I found no *de novo* insertion.

De novo insertion can take place in a germline or in a somatic cell. Here we are only interested in germline *de novo* because my analysis focuses on population genetic. Since somatic insertions cannot take place in the reproductive cells, usually the count of TE probe reads for somatic cell is far less than germline. This was taken care by setting a high threshold while running ELITE. Somatic *de novos* are also highly likely to be heterozygous. Moreover, for a subset of samples, I also have their “twins” or biological replicates from the same CC line. ERVis that are only present in a single CC line and absent in everyone else are also considered as germline *de novo*. It is also guaranteed that, these *de novo*, shared by replicates are in germline.

I found total 513 potential germline *de novos* in CC population. Number of *de novos* depends both on ERV template and sample. Some ERVs tend to have more *de novos* (ERVB7_1-LTR_MM), where some have none (ERVB4_2B-LTR_MM and IAPEY3C_LTR). Figure 5.8 shows the total number of ERVB7_1-LTR_MM *de novos* I found in CC population. Green bars in the figure represents the homozygous *de novo* and red represents heterozygous *de novo*. The samples sorted by the average number of *de novos* in their corresponding CC lines, and then by the total *de novos* in that particular sample. We chose ERVB7_1-LTR_MM becasue it has the highest number of *de novos* (166) compared to all the ERVs we assesed. From the figure we can see that, some CC lines (CC055, CC027, CC026) have more de novos than others. Both CC055, CC055M3799_UNC_NYGC and CC055F5017_UNC_UNC has 9 de novos each. For example, CC005M4714_UNC_NYGC also has 9 *de novos*, and its replicate CC005F1000_JAX_NYGC has 6. On the other hand, there are 32 samples that don't have a single *de novo*. 19 samples has only 1 and 13 have only 2 *de novos*. Many of the *de novos* are found in heterozygous state (shown in red), suggesting very recent insertions that are still segregating. The number of de novos can also be used to calculate mutation rate in the population. Since the number of *de novo* events are different for different ERV templates, we can also calculate how mutation rate varies with respect to ERVs. It is also possible to find the underlying cause of having more de novos for certain ERV types which we will discuss later.

5.8 Validating De novo Insertions

Validating *de novo* insertion in real sample where the ground truth is not known is not straightforward. It can be done using QTL mapping as the probes for de novos are not segregating. Be-

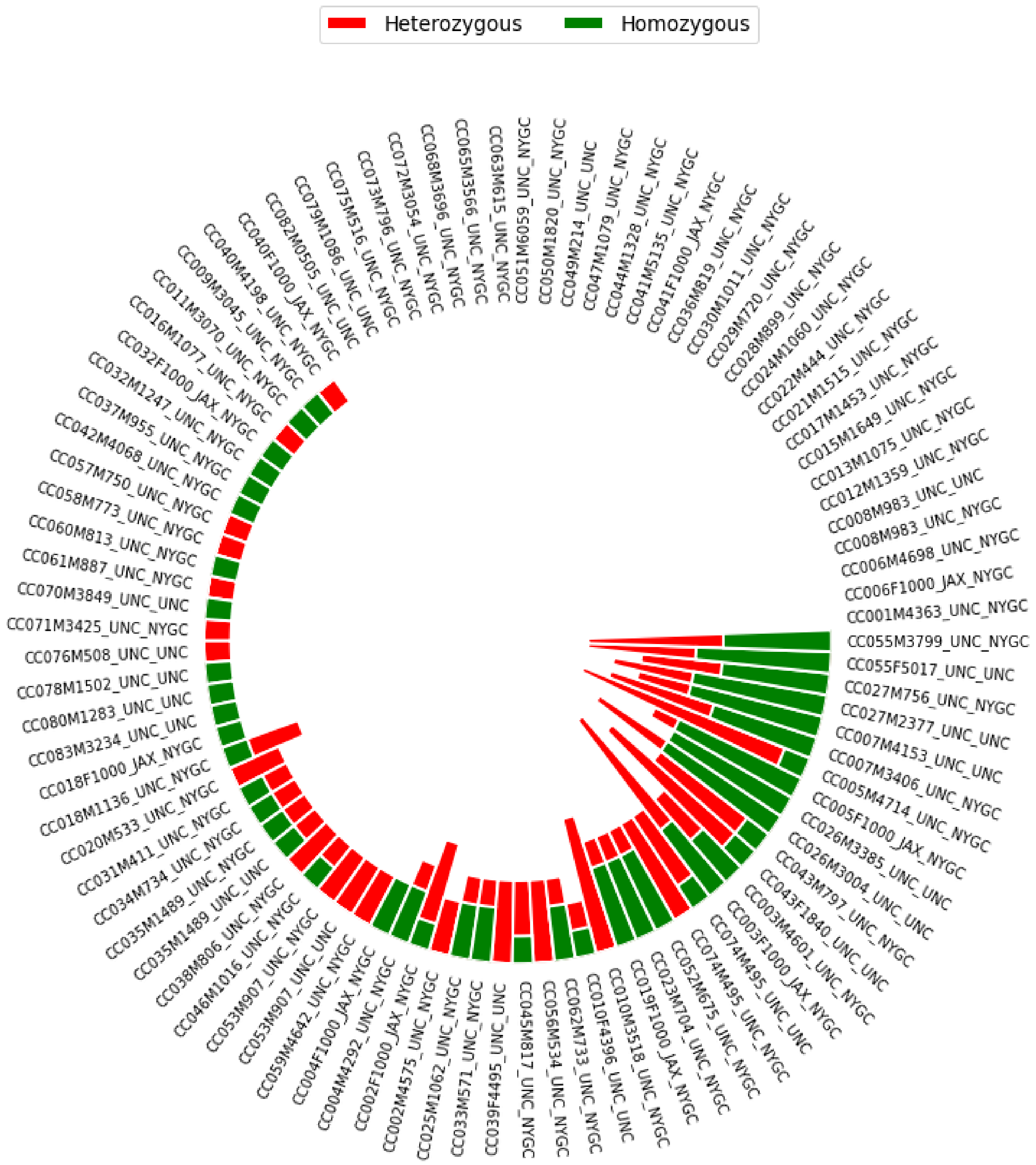


Figure 5.8: Number of *de novos* per CC sample. Green and red represent homozygous and heterozygous ERVis respectively. Almost one third of the samples have no *de novo*. On the other hand ome samples have more *de novo* like CC055M3799_UNC_NYGC and CC027M2377_UNC_UNC. Other samples from the same CC line CC055F5017_UNC_UNC, CC027M756_UNC_NYGC also have high nunmber of *de novo*. Majority of the de novos are in heterozygous state suggesting very recent insertion.

cause *de novo* TE probes are present only in one sample, or multiple samples from the same CC strain. However, I showed two different approaches for validating *de novos*. First one is based on the available data I have, which is comparing *de novos* that are shared between biological replicates. Second one is based on pure biological PCR-based method which is done by our collaborators at the Department of Genetics.

5.8.1 Validation using Replicates

The CC population provides a great opportunity to validate some of the *de novo* insertions. Unlike the segregating ones, de novos are not shared between all the samples with the same genotype. However, samples that came from the same CC line may share some *de novos*. Because breeding of CC started 10 years ago and I call any insertion as *de novo* if it is less than 10 years old. Therefore, I compare *de novo* ERVB7_1-LTR_MM insertions between biological replicates to eliminate the chance of false positives. Since I run ELITE individually on each sample, it is impossible to TE insertion exactly at the same position in two different dataset. Therefore, I use the biological replicates to show how a subset of the TEis are consistent within each other.

There are total 15 CC lines for which we have technical replicates. In table 5.4, I show the number of *de novos* in individual CC sample and the common *de novos* present in all biological replicates. As we can see, around 50% of the de novos are common to two replicates. However, CC003F1000_JAX_NYGC is the only exception who shares only 1 out of 8 *de novos* with its replicate. On the other hand, both samples from CC055 which have the highest number of de novo, share 8 out of 9 insertions. There are two CC lines, CC006 and CC041, have no de novo insertions at all. Overall in these 15 samples, there are 89 distinct *de novo* insertions and 36 are shared between biological replicates. These 36 insertions, found independently in two samples from the same CC lines proves that they were true insertions. Moreover, they are recent as their founder does not have that TEi.

De novos, those are not shared between biological replicates do not necessarily mean that they are false positive. Using only sequence data it is not possible to find the ground truth. Thus in later section, I, with the help of some biologists, validated a subset of *de novo* that are not shared between biological replicates.

So far I have talked about the biological replicates only. However, I also have technical replicates for 4 different CC strains. Technical replicates are basically the same sample but sequenced twice in two different machines. Even though technical replicates have the same DNA, their coverage is different. For example the coverage of CC074M495_UNC_NYGC is 32 whereas its only 8 for CC074M495_UNC_UNC. However, the total number of *de novo* in both dataset in 4, and all 4 are common to each other. This is same for the 3 other samples. All these samples have the exact same number of *de novo* as their technical replicates. This consistency proves ELITE's high true positive and true negative rate.

5.8.2 Validation using PCR-based method

I also validated all the de novos present in two samples from the CC027 strain. At first, we ran ELITE on CC027M756_UNC_NYGC. Due to high number of *de novos* found in this sample, I added another sample CC027M2377_UNC_UNC from the same strain. Both of the samples, CC027M2377_UNC_UNC and CC027M756_UNC_NYGC has 19 total *de novo* insertions, and 8 of them are present in both. Overall there are $19+19-8 = 30$ distinct de novo insertions combined two CC027. Among the 19 in CC027M756_UNC_NYGC, 7 are from ERVB7_1-LTR_MM, 1 is from MERVL_LTR, 7 are from RLTR1IAP_MM and 4 are from RLTRETN_MM. On the other hand, among the 19 in CC027M2377_UNC_UNC, 8 are from ERVB7_1-LTR_MM, 1 is from MERVL_LTR, 3 are from RLTR1IAP_MM and 7 are from RLTRETN_MM.

I had perfect concordance between ELITE's predictions and PCR validation in all 90 assays. Each TEi has 3 assays, forward (F), reverse (R) and (LTR-R). An amplification band only in F/R indicates absence of TEi, a band only in F/LTR indicates presence of TEi in homozygous state, and a band both in F/R and F/LTR indicates presence of TEi in heterozygous state. I tested the presence, absence and zygosity for all 30 of these TEis in 75 additional unsequenced CC027 samples. All 90 of those PCR results suggest genotypes that are consistent with ELITE's TEi findings. Though there are many TEis that were not present in any other CC027 except for the one I used as de novo insertion varies between samples. Thus, I have confirmed 100% of 30 of ELITE predicted de novo TEis. This supports our claims that ELITE has a low false-positive rate.

5.9 ERV insertion by Genomic Region

Even though genome is full of transposable elements, not all of them are active. Due to mutation in the TE sequence or the genome's own mechanism of suppression, majority of TEs in mouse or human genome lost their ability to move. These inactive TEs have no functional consequence thus are passed on to generation to generation and become fixed. However, active TEs can still move, and depending on the insertion location, it can be highly deleterious. For, example if a TE gets inserted into or near a gene, (which is the functional part of a genome), it can completely disable the gene's functionality. On the other hand, if a TE is not deleterious, most probably it will stay in the genome, and over the time become fixed with in future generation. Therefore, I annotate all the ERV insertion by their genomic region. I categorize genomic region by the proximity with respect to a gene. There are 4 different kinds of genomic regions. They are listed below:

1. **Exon:** Exon is the part of functional DNA. Even though we stated before, gene is the functional part of a genome, some segments of genes does not really get translated into protein. Sequence of genome that actually encodes protein is called Exon. Every gene starts and ends with an Exon.

2. **Promoter:** Promoter is a sequence that is within 1000 bases of transcription binding site. Simply put, its a sequence that is needed to turn a gene on or off. Even though its not inside the gene, or does not encode protein, its ability to turning the gene off can result serious consequences.

3. **Intron:** Intron is a part of a gene. However it does not get translated into protein. However, it is an integral part of gene expression regulation which affects phenotype. A gene can

contain several exons, with several introns in between them. Introns are also known as intagenic region.

4. **Intergenic:** The region outside the gene is called intergenic. Neither do they encode protein, nor they regulate gene expression. Their functionality are still unknown and are often referred to as "dark matter". Intergenic region comprises round 50% of a human genome.

I annotate the genomic region of each ERV insertion as defined above. Genes can overlaps with each other. Therefore it is possible to have a region which belongs to an exon of some gene but intron or promoter to some other gene. In cases of multiple regions, I chose the one which can have more deleterious effect. Based on the ERVi sharing pattern, I calculate the fraction of insertions that are in each genomic region (shown in table 5.6.) As I can see from the table, Most of the Fixed insertions are intergenic (81%). Intergenic regions are non-functional thus genome most likely does not resists if an insertion takes place there. This is quite similar to the segregating ones, except that the number is little lower than fixed. Around 73% the segregating ERVis are intergenic, 25% are in intron. In the case of *de novo*, 2.3% insertions are in exon which may at first looks like a very small fraction. However, compared to the fixed (.3%), and segregating (.2%) ERVis, de novo insertions are more likely to be found in exon.

5.10 Summary

I use our template-based tool ELITE for finding ERV insertions in each sample in the CC population including the founders. To get the templates, I also use my template-free pipeline Frontier. Frontier automatically classifies the TE types, and TE types that belong to ERV class are used here. To analyze the result on a population basis, I added some new features to our original pipeline. First, I build some additional probes for each TEi which I later use for genetic mapping. Then, I provide a genome-wide TEi sharing pattern between all samples by combining the individual insertions. Based on the TEi sharing pattern, I annotate each insertion's state whether it's fixed, still segregating or a de novo. I observed 36% ERVis are present in everyone and became fixed in the CC population. Moreover, 62% of ERVis are still segregating, and 75% of those are verified by QTL mapping. Using this large CC population with known pedigree also allowed me to identify the most recent events - *de novos*. And all the de novos in CC027 were also verified using PCR method. Overall a large number of ELITE's predictions are validated. I have not specifically try to validate any fixed ERVis. Fixed ERVis are present in every sample and their founders including B6. Now reference genome is based on B6, therefore I just verified if all the fixed insertions are already annotated in reference genome. This way I have a list of curated insertions in CC population.

Many researchers in the field of genetics and biology use the CC for various kind of analysis. Annotation of ERV insertions in this population will facilitated their research in a great way. I, also in collaboration with the Genetics department of UNC Chapel Hill perform many experiments. Most of these are purely biological. However, I describe some of the findings with adequate background so that the impact of my algorithmic pipelines can be understood and valued. Among the many interesting findings, I saw that, the ERV segregation or sharing pattern in CC population is consistent with the commonly believed mouse phylogeny. I can also estimate the age of ERV type. Some ERVs like ERVB4 are very old that most of the insertions became fixed. On the other hand, some ERVs are still active in CC, contributing to many recent events or *de novo*. Moreover, insertions that are not deleterious mostly found in intergenic (nonfunctional) region and became fixed in CC. *De novo* has highest number of exonic ERVis. These potential deleterious ERVs are usually purged quickly from a population. Overall I show that my algorithms work on a large genetically diverse population like CC. Therefore, it can be used to catalog and annotate ERVis in any other population. Moreover, I provide a complete database of the ERVis that are found in CC which serves a great resource for the researchers who use this population for various analyses.

Strain	Sample	*de novo*	Common
CC002	CC002F1000_JAX_NYGC	4	2
	CC002M4575_UNC_NYGC	2	
CC003	CC003F1000_JAX_NYGC	8	1
	CC003M4601_UNC_NYGC	4	
CC004	CC004M4292_UNC_NYGC	3	2
	CC004F1000_JAX_NYGC	2	
CC005	CC005F1000_JAX_NYGC	6	3
	CC005M4714_UNC_NYGC	9	
CC006	CC006M4698_UNC_NYGC	0	0
	CC006F1000_JAX_NYGC	0	
CC007	CC007M4153_UNC_UNC	7	5
	CC007M3406_UNC_NYGC	8	
CC010	CC010M3518_UNC_NYGC	5	1
	CC010F4396_UNC_UNC	2	
CC018	CC018F1000_JAX_NYGC	1	1
	CC018M1136_UNC_NYGC	3	
CC026	CC026M3004_UNC_UNC	8	5
	CC026M3385_UNC_UNC	5	
CC027	CC027M2377_UNC_UNC	8	4
	CC027M756_UNC_NYGC	7	
CC032	CC032F1000_JAX_NYGC	1	1
	CC032M1247_UNC_NYGC	1	
CC040	CC040M4198_UNC_NYGC	1	0
	CC040F1000_JAX_NYGC	0	
CC041	CC041F1000_JAX_NYGC	0	0
	CC041M5135_UNC_NYGC	0	
CC043	CC043M797_UNC_NYGC	5	3
	CC043F1840_UNC_UNC	7	
CC055	CC055M3799_UNC_NYGC	9	8
	CC055F5017_UNC_UNC	9	

Table 5.4: Comparing *de novo* ERVB7_1-LTR_MM insertions between biological replicates to eliminate the chance of false positives. Since I am calling any insertion that is less than 10 years old as *de novo*, in that sense insertions that are shared between biological replicates are also *de novo*. For the strains I have replicates, around 50% of the *de novos* are shared. However, CC003F1000_JAX_NYGC is the exception who shares only 1 out of 8 *de novos* with its replicate. On the other hand, both samples from CC055 which have the highest number of *de novo*, share 8 out of 9 insertions.

Strain	Sample	de novo	Common
CC008	CC008M983_UNC_NYGC	0	0
	CC008M983_UNC_UNC	0	
CC035	CC035M1489_UNC_NYGC	2	2
	CC035M1489_UNC_UNC	2	
CC053	CC053M907_UNC_UNC	2	2
	CC053M907_UNC_NYGC	2	
CC074	CC074M495_UNC_UNC	4	4
	CC074M495_UNC_NYGC	4	

Table 5.5: Comparing de novo ERVB7_1-LTR_MM insertions between technical replicates. Technical replicates are basically the same sample but sequenced in two different places. Here all the 4 samples are sequenced twice, in UNC and in New York. Therefore, they must share all the insertions whether de novo or not. As we can see, exactly the same number of de novos are present in each technical replicate of the same CC line, and they share all of them.

Genomic region	Total	Fixed	Segregating	*de novo*
Intergenic	77%	81%	73%	55.4%
Intron	22%	18%	25.5%	41.5%
Promoter	.9%	.8%	.9%	.8%
Exon	.3%	.2%	.3%	2.3%

Table 5.6: Genomic regions of ERVs classified by their segregation pattern in the CC population. ERVs are classified into one of four genomic regions depending on their genomic positions. For ERVs that potentially belong to multiple contexts, they were assigned to the most biologically significant based in the following schema: $exon > promoter > intron > intergenic$. Percentages are calculated within segregation groups.

CHAPTER 6: TOOLS AND RESOURCES

In this chapter, I document all the TEi related tools and resources created as part of this research for future use. Chapter 3 and 4 focus on describing the algorithms and their performances. In this chapter I give step-by-step instructions for running these tools on short read data. I also describe several web-tools for visualizing ERV insertions in the CC population. Finally, I provide links to download all the resources including source codes, and ERVi annotations in CC.

6.1 ELITE

ELITE is a tool for efficiently locating insertions of transposable elements in the genome. It is a template-based tool which uses a conserved k-mer as seed to locally assemble the insertion boundary. After extracting the insertion context from the boundary, a third party tool, bowtie2 is used to map it with respect to a reference genome. Therefore, to run my pipeline, there's some additional dependencies that I need to consider. In this section, I will briefly discuss how to run ELITE on short read dataset.

Source: Source codes and sample data can be downloaded from github:

`https://github.com/Anwica/ELITE`.

Pre-requisite for running ELITE:

1. Install msBWT from here:

   `https://github.com/holtjma/msbwt`

2. Install bowtie2 from here:

   `http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml`

3. Install Levenshtein python package from here:

   `https://pypi.org/project/python-Levenshtein/`

4. Build msBWT from short-read, follow instructions from here:

   `https://github.com/holtjma/msbwt`

5. Build bowtie2 index from reference genome, follow instructions from here:

   `http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml`

Prepare Input Data:

1. Create a .csv file with the following information for each sample: (a sample file sample_list.csv is given in the sample_data folder)

   - **sample name**: name of you sample
   - **bwtfile**: directory where the sample's msBWT is located
   - **threshold**: minimum number of supporting split-reads with TE

2. Create a .csv file containing the TE templates: (a sample file TEseq.csv is given in the sample_data folder)

   - **my_id**: name of the TE template
   - **TE**: sequence of the TE
   - **start_seed**: seed kmer (lengths between 21-35) near the TE's proximal boundary [empty string if you want ELITE to find the optimal seed]
   - **end_seed**: seed kmer (lengths between 21-35) near the TE's distal boundary [empty string if you want ELITE to find the optimal seed]

3. Reference genome: (a sample genome Reference.fa is given in root folder)

   - The reference genome of the sample species

4. Required parameters for running ELITE:

    - **sample_list**: path to the .csv file containing the sample's information
    - **te_file**: path to the .csv file containing the TE templates information
    - **reference**: path to the reference genome file
    - **species**: name of your bowtie index
    - **C**: length of TE's context which is the segment of non-TE sequence right next to a TE sequence [C = 25 recommended]
    - **T**: length of proximal and distal TE length which is the segment of TE sequence right next to a context [C + T must be less than half of read length]
    - **K**: length of seed [25 recommended]

a sample script runELITE.sh is given to run ELITE using the sample data. It will automatically create bowtie index if not already built. You must use the same reference genome for building bowtie and running all the steps of ELITE. For running ELITE, simply modify runELITE.sh with appropriate parameters and execute it.

6.2 Frontier

Frontier is pipeline that automatically detect the boundary of TE insertions without any template. Frontier at first identifies all the repeats directly from a sequences dataset, assuming that all TEs are repeated. Then it extends the repeat to find boundaries where repeat ends in a normal genomic coverage. Using a deep-learning classifier, frontier further classifies each boundaries where they are true frontier or not. For the true frontiers, there is a second classifier which further predicts the TE type in the repeated segment.

Source: Source codes and trained models can be downloaded from github:

`https://github.com/Anwica/Frontier`.

Pre-requisite for running Frontier:

1. Install msBWT from here:

   `https://github.com/holtjma/msbwt`

2. Install pytorch from here:

   `https://pytorch.org/`

3. Build msBWT and LCP from short-read Follow instructions from here:

   `https://github.com/holtjma/msbwt`

Running Frontier.py

1. Required parameter for running Frontier:

   - **kmerSize**: length of repeat, default is 45
   - **msBWTdir**: path to the msBWT directory
   - **tmp**: path to tmp directory

Running CreateCountMatrix.py

1. Required parameter for running CreateCountMatrix.py :

   - **frontierfile**: List of frontier candidates obtained from Frontier.py
   - **bwtfile**: Path to the bwt file
   - **countfile**: Desired Path and Name of the countfile

Running RunFrontierClassifier1_.py

1. Required parameter for running RunFrontierClassifier1_.py :

   - **frontierfile**: List of frontier candidates obtained from Frontier.py
   - **countfile**: Path to the countfile obtained by running CreateCountMatrix.py
   - **trainedmodel**: Path to the trained model for finding true frontier.
   - **truefrontier**: Desired Path to the output that keeps only the true frontier sequences

Running RunFrontierClassifier2.py

1. Required parameter for running RunFrontierClassifier2.py :

    - **truefrontier**: Path to the output that keeps only the true frontier sequences
    - **trainedmodel**: Path to the trained model for for classifier 2.
    - **truefrontier_classified**: Desired Path to the true frontiers with TE type classified

6.3 Web tools and resources

We have an website: `http://csbio.unc.edu/TEs/index.py` to make all the resources easily accessible to anyone who is interested in ERV insertions in Collaborative Cross Population. As described in chapter 5, I ran ELITE on CC population using 9 different ERV templates. I documented everything throughout the process and put them on our website. Three main tabs we have are as follows:

6.3.1 Summary

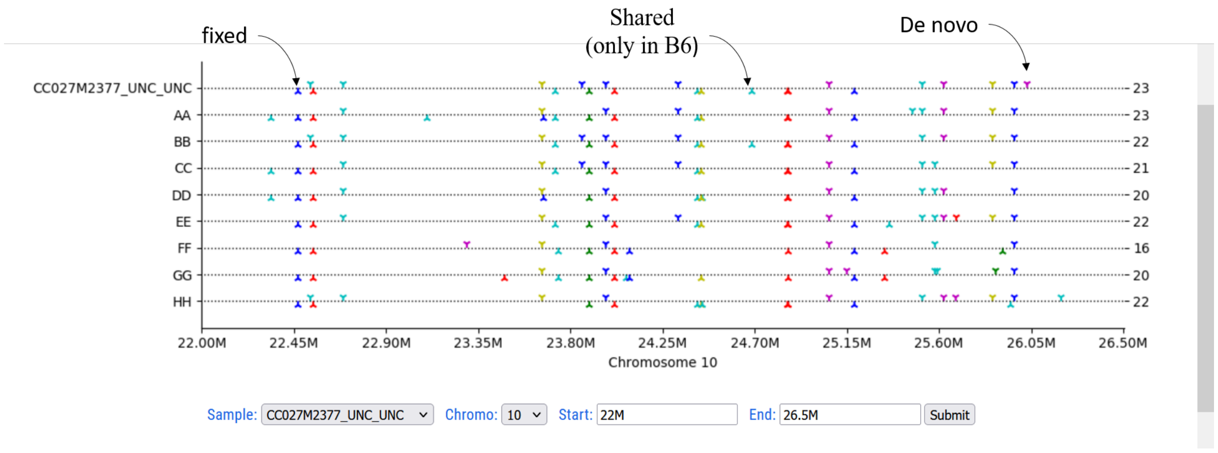
This tab shows several summary statistics we found useful after running ELITE. The url is `http://csbio.unc.edu/TEs/index.py?run=Summary.index`. Some of them are briefly described below:

- Number of TEis per ERV templates. Those are also broken down into types like whether they are present in reference genome, or still segregating.
- genomic region of each TEi based on their segregation pattern. Here segregation pattern refers to fixed, segregating and de novo. For each, we list how many of these are intergenic, and in gene's exon, intron and promoter.
- Number of de novo ERV insertions per sample for each of the 6 used templates. We also show how many of the de novos are homozygous and how many are heterozygous.

- Phylogenetic trees for each ERV templates based on the insertion sharing pattern. Here I only include the insertions that are either fixed or segregating.

6.3.2 TEiVewer

I developed an webtool, TEiViewer, for visualizing ERV insertions in CC population. It's like UCSC (Kuhn et al., 2013) or ensembl (Fernández and Birney, 2010) genome browser. In this genome browser, user can select a CC sample and a range in a chromosome to view all the insertions that are present in that CC. In addition to CC, I also show all the ERVis in 8 CC founders to understand the inheritance pattern of the insertions. If a sample is NOD (DD) in a region, most likely it will have all the TEis that are also in NOD unless a deletion event occurred at that DD. The url to this viewer is `http://csbio.unc.edu/TEs/index.py?run=TEiViewer.index`



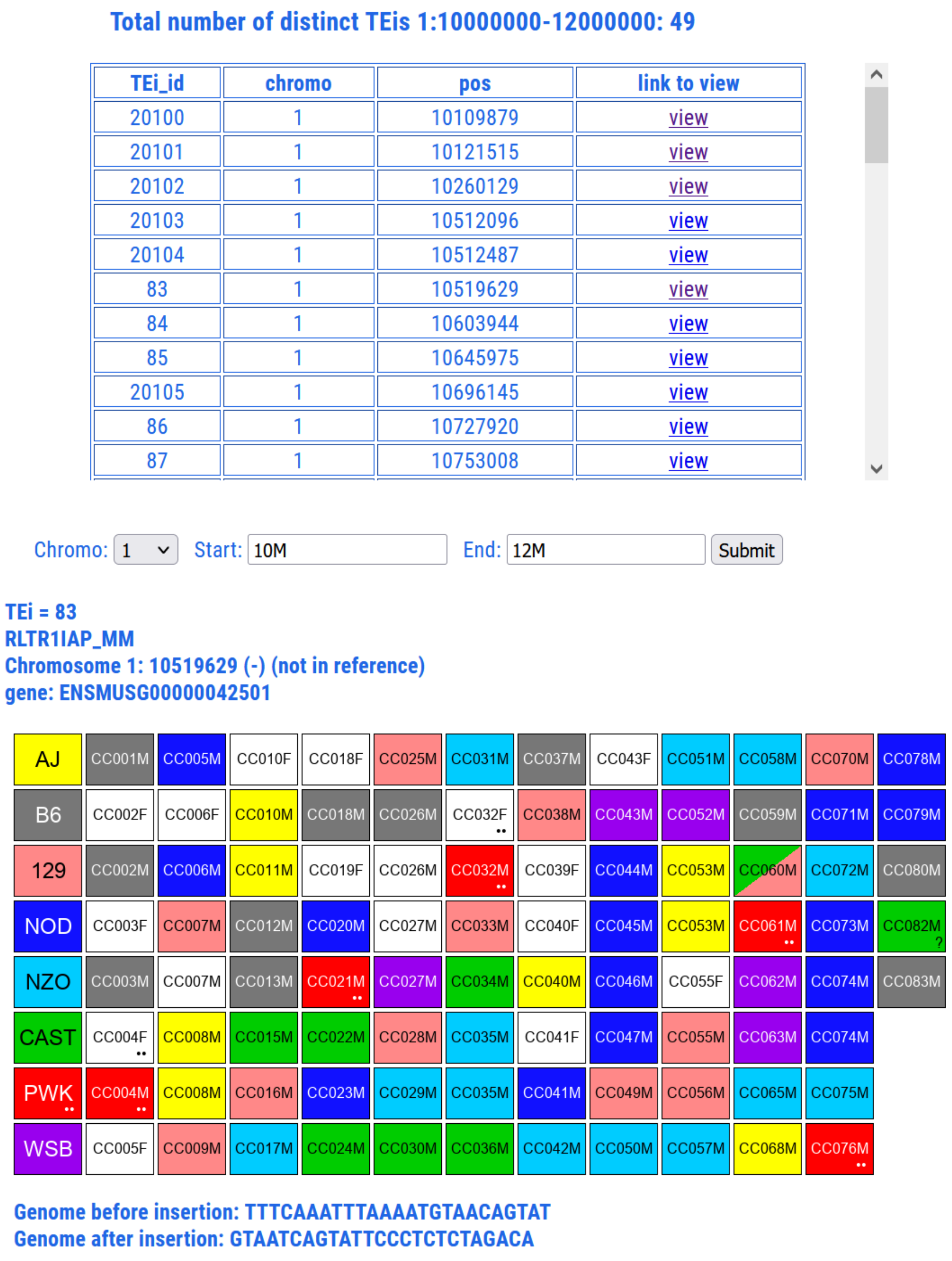
Sample: CC027M2377_UNC_UNC Chromo: 10 Start: 22M End: 26.5M

Figure 6.1: TEiViewer: An webtool for visualizing TE insertions in a population. Here I show the TEis that are present in CC027M2377_UNC_UNC in chromosome 10 starting from 22M to 26.5M. Around 22.45M, CC027 has an insertion which is present in all the other founders. At around 24.7M, CC027 has a TEi, which is only present in B6 (BB) sample. This means CC027 inherited this region along with the TEi from B6. Finally at around 26.05M, CC027 has a TEi that is not present in any of the 8 founders. Therefore, that TEi was not inherited from the ancestors and considered as a recent event or *de novo*.

Figure 6.1 shows a snapshot of our TEiViewer. As shown in the bottom, I have a dropdown list from which one can choose one CC sample. Beside that, there's another dropdown list for choosing a chromosome. Then I have two text inputs, Start and End, which refers to the starting and ending position of the selected chromosome. When user hits the return button, I show that particular samples genome by marking all the ERVis in that given range of that chromosome. I also show the ERVis that are present in all the 8 founders. After selecting CC027M2377_UNC_UNC as a sample, we can see that there's total 23 ERVis in that region. Founder AJ (AA) also has 23 ERVis, but those are not the ones that CC027 has. The first ERVi here which is around at 22.45M is present in AA. This is shared with CC and DD, but not with CC027. The next ERVi (shown in blue) is present in everyone and became fixed in CC population. The next one (shown in cyan), is shared between CC027,BB, and HH. This tells us, CC027 is either BB or HH in that region from which it inherited the insertion. It's unclear whether CC027 is BB or HH until we look at the ERVi at 24.7M. Here, CC027 has an insertion that is only present in CC027. Therefore, this is a great way to track the inheritance using TEi sharing pattern. It also tells us the ERVis that are very recent or de novo. For example, at around 26.05M, CC027 has an insertion, which is absent in all the 8 founders. Thus, this insertion took place after the breeding of CC population.

6.3.3 TEiDisplay

I developed an webtool, TEiDisplay, for visualizing individual ERV insertions in CC population. We also include the UNC founders so that one can understand the segregation pattern among the CC samples. My display tool takes a range in a chromosome as input. It then list all the TEi_id representing ERV insertions. For each TEi_id, some additional information are provided so that user can pick a single TEi to look in detail. The link "view" takes user to a new page where for each UNC founders and CC samples, I annotate whether it has TEi or not. I also indicate if it's in homozygous or heterozygous state. Homozygous TEis are shown by two dots, while heterozygous are shown by one dot. Each founder is color coded like this: yellow in for AJ, black is for B6, pink is for 129S1, dark blue is for NOD, light blue is for NZO, green is for CAST, red is for PWK, and purple is for WSB. CC samples are also color coded like the founders. Now I already know all CC samples genotype i.e., for a given position the founder from which they inherited the DNA sequence. Color for each CC is based on that inherited founder. CC can have different founder at different places, thus different color code.



Total number of distinct TEis 1:10000000-12000000: 49

TEi_id	chromo	pos	link to view
20100	1	10109879	view
20101	1	10121515	view
20102	1	10260129	view
20103	1	10512096	view
20104	1	10512487	view
83	1	10519629	view
84	1	10603944	view
85	1	10645975	view
20105	1	10696145	view
86	1	10727920	view
87	1	10753008	view

Chromo: 1 Start: 10M End: 12M Submit

TEi = 83
RLTR1IAP_MM
Chromosome 1: 10519629 (-) (not in reference)
gene: ENSMUSG00000042501

AJ	CC001M	CC005M	CC010F	CC018F	CC025M	CC031M	CC037M	CC043F	CC051M	CC058M	CC070M	CC078M
B6	CC002F	CC006F	CC010M	CC018M	CC026M	CC032F ••	CC038M	CC043M	CC052M	CC059M	CC071M	CC079M
129	CC002M	CC006M	CC011M	CC019F	CC026M	CC032M ••	CC039F	CC044M	CC053M	CC060M	CC072M	CC080M
NOD	CC003F	CC007M	CC012M	CC020M	CC027M	CC033M	CC040F	CC045M	CC053M	CC061M ••	CC073M	CC082M ?
NZO	CC003M	CC007M	CC013M	CC021M ••	CC027M	CC034M	CC040M	CC046M	CC055F	CC062M	CC074M	CC083M
CAST	CC004F ••	CC008M	CC015M	CC022M	CC028M	CC035M	CC041F	CC047M	CC055M	CC063M	CC074M	
PWK ••	CC004M ••	CC008M	CC016M	CC023M	CC029M	CC035M	CC041M	CC049M	CC056M	CC065M	CC075M	
WSB	CC005F	CC009M	CC017M	CC024M	CC030M	CC036M	CC042M	CC050M	CC057M	CC068M	CC076M ••	

Genome before insertion: TTTCAAATTTAAAATGTAACAGTAT
Genome after insertion: GTAATCAGTATTCCCTCTCTAGACA

Figure 6.2: display

6.3.4 Downloads

I made downloadable spread sheet available online annotating all the ERV insertions in CC. These can be found here: `http://csbio.unc.edu/TEs/index.py?run=Downloads.index`. I provide necessary glossary to understand the output spreadsheet, which can also be downloaded from there.

6.4 Summary

This chapter is written mainly for the users who wish to use our pipelines. Easy step-by-step instruction will also help future researchers who want to make any improvement to the algorithms. I also provide the results of running our pipelines on collaborative cross population. I include detail information for each insertion including all the probes (both TE present and absent) and their counts in each samples. Since many biologists use this CC population for many different kind of analysis, I created several webtools to easily view the diversity pattern of ERV insertions.

CHAPTER 7: DISCUSSION AND CONCLUSION

As part of my research into the exploration of the variability of Transposable Elements in intraspecific genomes I developed two different analysis algorithms and implemented pipelines for using them. Both algorithms rely on local-genome-assembly to look for transposable element insertions in the genome from short reads. Both pipelines assemble only the boundaries of insertions locally and find the context of the insertion. Since short-reads are smaller than a full TE sequence, it is possible to find the boundary unambiguously instead of a full inserted sequence with context. Some other TEi detection tools, like split-read models also work like that. However, their initial alignment with respect to the given TE templates is computationally expensive. On the other hand, instead of alignments, both of our pipelines use msBWT to store and retrieve short read data. Not only msBWT saves space, it also makes the local assembly much easier than typical methods. The way msBWT works, that is searching by suffix is perfect for assembling a genomic segment base by base. Even when a TE template is unknown, we show that msBWT can be jointly used with another data structure the LCP, to identify any repeated segments in the genome. Using deep learning classifier works great to finally filter the segments that are actually the boundary of transposable element insertions along with their type.

Looking only the insertions boundary eliminates a lot of repeated segments that are inside a TE. Those sequences do not help in finding insertion locations but create significant memory overhead. We strongly believe that our pipelines can be used in any kind of large scale short read data to identify TEis by mainly focusing on the insertion boundaries. However, there are still scope of improvements. Therefore, in this final chapter, we briefly discuss the summary of our results along with some existing limitations and potential improvements to our algorithms.

We also discuss some possible future research directions that can be pursued by extending our existing pipelines.

7.1 ELITE

Our first pipeline ELITE shows a promising result when compared to other template-based TE searching tools. We compare our results with both discordant-pair-based models and split-read-based models ((Gardner et al., 2017), (Zhuang et al., 2014), (Chen et al., 2017)) and show that ELITE's precision and recall rate are more than 30% and 5% higher respectively given a good coverage. Moreover, ELITE also outperforms those in predicting zygosity and the insertion location to the base pair. ELITE is computationally far less expensive and 2x faster than typical TE searching tool. The msBWT data structure also makes it feasible to run ELITE multiple times on a sample with different TE templates and across genome build versions. All these together makes ELITE a perfect choice for identifying diversity of TE insertions in a large scale population.

7.1.1 Limitations of ELITE

- One drawback of ELITE is that it does not perform equally well for all types of TEs. For example there are 3 main types of the TEs in human and mouse genome: LINEs, SINEs and ERVs. Our experiments show that ELITE performs better for ERV type as it has a more structured boundary. On the other hand, LINEs and SINEs are often truncated on the 5' end, and have variable length of poly A's at 3' end. Because of these wide range of variations in both 5' and 3' ends, it becomes difficult to identify the exact insertion boundary.

- Another limitation of ELITE is the requirement of TE template. Just like any other existing TE searching tools, ELITE needs a TE template to find the insertions of that TE type. Even though ELITE can identify TE insertions that are divergent to the given TE template, it needs a TE seed to start with. A seed is a sub-sequence from a known TE sequence that is conserved (meaning repeated numerous times in the genome). Even though current version of ELITE cannot automatically identify seeds from short reads, our second pipeline Frontier 4 is specifically designed to do that.

7.2 Frontier

Our second pipeline Frontier introduces a novel idea of detecting transposable element insertions directly from short reads without any given TE template. Frontier bypasses the need of a TE template with the assumption that TE sequence is repeated throughout the genome. Therefore, it uses a repeat finding algorithm at first to list all the candidate reads where repeated segments end in a normal coverage. Then a deep learning classifier is used to finally identify the reads that belong to TEi boundaries only. This makes it possible for Frontier to find TE types that are completely unknown. Moreover, it is not also biased towards any type of TE. Unlike ELITE, Frontier is able to find all classes of TE, LINEs, SINEs, and ERVs. In comparison to template-based TE searching tools, Frontier performs similar to ELITE as long as the TE segment is highly repeated in the genome.

7.2.1 Limitations of Frontier

- Our current version of Frontier considers a segment as TE if its repeated more than a certain threshold in the genome. This is a hard threshold and the exact sequence needs to be present over than threshold parameter. However, sometimes TE sequences get mutated and the mutated version may not have very high count. Thus some TE sequences gets ignored by Frontier as their count is below the hard threshold. This can be fixed by allowing some edit distances in the repeated segment which may need some additional data structure along with msBWT and LCP.

- Frontier's *Classifier*-II predicts only high level TE types like LINEs, SINEs and ERVs. However, each TE type can further be classified into subtypes. For example there are several types of ERVs in mouse like ERVB7_1-LTR_MM, MTA_mm, RLTR1IAP_MM etc. To correctly identify all the names of the subtype present in a given frontier, it is essential to revisit the design of the neural network. There are a couple of available deep neural network-based classifiers that try to predict these subtypes which we can implement to refine this process.

7.3 Future Research Ideas

In this section, I discuss future research directions that can be pursued. Some of the ideas are related to improving the tools that I developed. I also discuss some other research problems than might be easily solved by using my existing pipelines. These ideas are open for everyone. Anyone interested in this topic are also welcome to further develop these ideas.

7.3.1 TEis as markers

Currently SNP-based markers are commonly used as *markers* in a genome that differentiate samples from a reference genome of the same species. These markers can be used to map inherited regions in the genome. Thus, we can use markers to identify the potential ancestor in particular regions of genome. In chapter 5, we used markers for QTL mapping. QTL mapping was able to tell us if TE insertions are consistent with the genotypes indicated by some previously verified markers at a specific locus. Those markers were extensively verified before being make commercially available to public. However, to identify novel markers based on SNPs or INDELs, it requires alignment of reads to a reference genome and finding the places where it has some variants. Moreover, typical SNP or small INDEL based probes have smaller resolution (1-3 bases) compared to TEi.

Therefore, I propose a new type of markers TEis, which can be obtained just by running an efficient computational tool, ELITE. TEis also have high resolution compared to SNPs since TE sequences are more than hundred basepairs long. Using TEi as marker can be done easily since ELITE also outputs the TE present probe. We can identify the differences just by querying the TE probes. The number of TEis found by ELITE is also richer than many arrays, or marker data set. For example, recently developed microarray at our lab called MiniMUGA contains 11,000 markers, which is less than a quarter of the ERVis in CC. Recall that, I only looked at the ERV insertions in CC, thus there's still a lot of possibilities to identify more TEis using LINE and SINE templates. Therefore a comprehensive list of all TEis can make a very rich marker data set.

I also discovered several cases in the CC population where TEis could differentiate between founders when classical SNPs cannot. In some places of the genome, SNP markers cannot distinguish between closely related mouse strains. This problem is known as identity by descent (IBD). However, ELITE discovered several ERV insertions within these regions, which can be used unambiguously to resolve IBD. Figure 7.1 shows an example, where ELITE can be used to resolve an IBD. In this figure, a TEi is shown that is found in chromosome 9 at position 31071802. Among the 8 CC founders, AJ and 129 have that insertion in homozygous condition shown by two dots. All the CC samples are color-coded based on the sequence inherited from the founders. Therefore, CC samples that are AJ in that region are shown in yellow, and 129 are shown in pink. As you can see, all the yellow and pink ones have the insertion except for CC065 and CC080. I investigated CC065 in particular and found that it's actually B6 in that region, that's why it did not have any insertion. However, GigaMUGA microarray was unable to detect this difference. Thus TEis are in some cases, more informative than typical arrays.

7.3.2 LINE-mediated Duplication

We can tweak our Frontier classifier to identify some other biological interesting phenomena. One such example is called LINE mediated Duplications (LiMeD). It is a recently discovered process which does not exactly follow the typical copy-paste mechanism of RNA transposon. In the typical case, when a LINE moves, it makes a copy of itself and only the copied version gets inserted to a new place. Sometimes whole thing cannot be copied to the new place and the

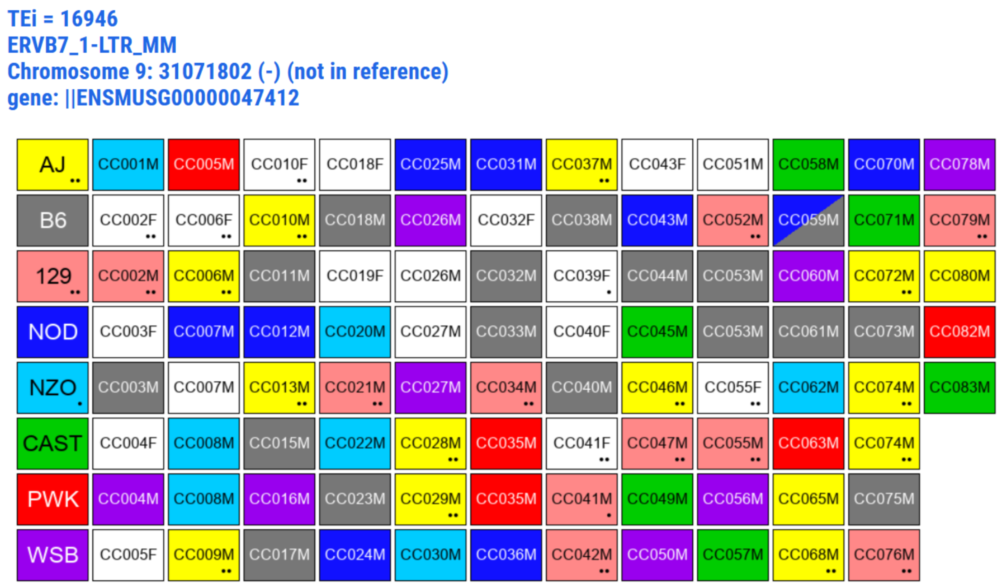

TEi = 16946
ERVB7_1-LTR_MM
Chromosome 9: 31071802 (-) (not in reference)
gene: ||ENSMUSG00000047412
AJ CC001M CC005M CC010F CC018F CC025M CC031M CC037M CC043F CC051M CC058M CC070M CC078M
B6 CC002F CC006F CC010M CC018M CC026M CC032F CC038M CC043M CC052M CC059M CC071M CC079M
129 CC002M CC006M CC011M CC019F CC026M CC032M CC039F CC044M CC053M CC060M CC072M CC080M
NOD CC003F CC007M CC012M CC020M CC027M CC033M CC040F CC045M CC053M CC061M CC073M CC082M
NZO CC003M CC007M CC013M CC021M CC027M CC034M CC040M CC046M CC055F CC062M CC074M CC083M
CAST CC004F CC008M CC015M CC022M CC028M CC035M CC041F CC047M CC055M CC063M CC074M
PWK CC004M CC008M CC016M CC023M CC029M CC035M CC041M CC049M CC056M CC065M CC075M
WSB CC005F CC009M CC017M CC024M CC030M CC036M CC042M CC050M CC057M CC068M CC076M

Figure 7.1: Example of an IBD region: TEi is found in most of the samples with AJ (shown in yellow) and 129 (shown in pink) in chromosome 9 at position 31071802. Homozygous TEis are shown by two small black dots. However, CC065 even though is AJ at that region, does not have a TEi. Several later investigates finally confirm that, there were not enough markers in the current array to distinguish between AJ and B6. Thus in a small region, where CC065 transtioned from AJ to B6 was not captured. However, it is clear from this TEi that CC065 is not AJ in that region.

copied version is clipped partially. However, recent research suggests that, , there are cases where a LINE copies a small segment of genomic sequence near the 3” boundary of the new LINE from elsewhere in the genome. That segment along with the LINE is inserted to a new place In our original Frontier 4 pipeline, we first identified the sequence where read count goes from high to normal. Here we need to find reads where count goes from high to normal then goes up to twice the normal coverage. Learning this pattern of counts is perfect for classifiers like deep learning. Thus we can add another class in the Frontier’s *Classifier*-I, to predict the reads that belong to Line-mediated duplication.

7.3.3 Transfer Learning: Mouse to Human

Researchers mainly use the mouse as a model genome and research organism. One of our pipelines, Frontier uses deep learning classifiers to predict TE insertions which is based on mouse model too. However, given the current advancement in the field of deep learning, it might be

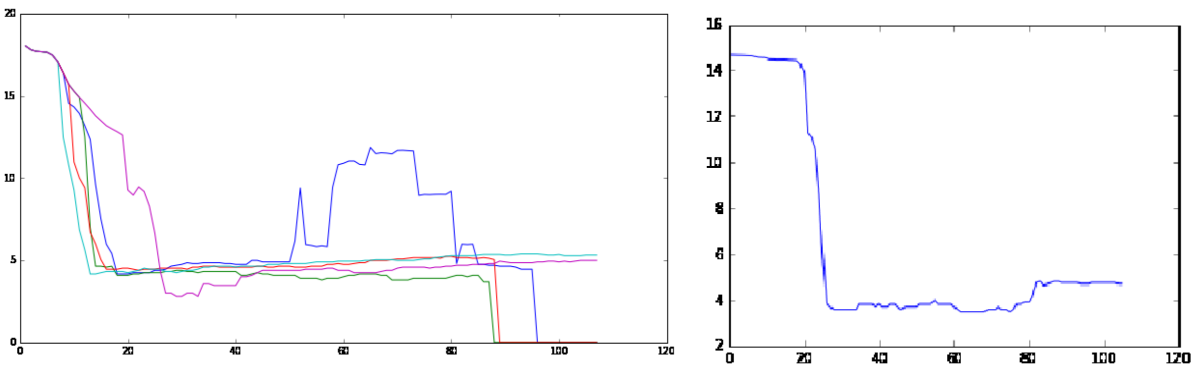


Figure 7.2: Comparison between LiMed and typical LINE insertion. For both LiMed and typical LINE insertion, we broke down the frontier read into consecutive 21-mers, and plotted their counts in the msBWT. Here left and right figure shows the 21-mer counts for typical and LiMed respectively. As you can see, in typical insertion, count drops from high to normal as the sequence goes from TE to normal coverage genome (shown in left). On the other hand, for LiMed (shown in right) count drops from high to normal and then again goes up. Initially when 21-mers are still from the LINE sequence, the count resembles many copies. Then when the count goes to normal, it resembles one copy and finally when the count again goes up it resembles two copies.

possible to use transfer learning to pass the knowledge gained from mouse to human. Transfer learning is a technique where an already trained network model helps to build a new trained model where both models aim to solve a similar kind of problem with slight distinction. The idea is to use completely or partially the parameters of a trained model in a different network. Training is the most computationally expensive part of a deep learning network, thus reusing it will accelerate the whole process.

In our frontier pipeline, the first classifier predicts the true frontiers from a list of potential frontier candidates or TEi boundaries. We used the mouse reference genome to train our model. There are three main reasons behind using mouse for this. First, mouse is highly similar to human genome. Similar to mouse, LINEs, SINEs and ERVs comprise majority of the transposable elements in the human genome. Second, mice are easy to breed in a research setting and takes only 2/3 years to get a new generation. Easy breeding facilitates population-based research which was also our main objective. Third, the mouse reference genome is very well annotated in terms of TEis. Researchers are mostly interested in mouse genome because of the first two reasons. However, we took the full advantage of the CC population population as a resource available at

UNC during my dissertation work. This also made us familiar with many additional resources related to CC. Moreover, well annotated mouse genome is helpful to get a good trained model. Therefore, a TEi boundary prediction model based on mouse genome seems perfect to be reused in identifying TEis in human.

There are several works where transfer learning is used to transfer knowledge from mouse to human successfully. For example, (Stumpf et al., 2020) shows that, trained model obtained from mouse can be used to classify cell types in human genome with zero shot learning. Therefore, we believe it is possible to use the trained model of Frontier's *Classifier*-I to successfully predict Transposable Element insertions in human.

7.3.4 TEi annotations in non B6 samples

RepeatMasker (Smit et al., 1996) currently annotates TEi that are in a reference genome. However, for the mouse, the reference genome only represents one inbred sample (B6), and we have discovered many TEis that are not in B6. Additionally there are also many TEis that are only present in B6. Many existing tools (Keane et al., 2012),(Gardner et al., 2017) cannot even find TEis that are in reference. These methods are useful if we only want to find new insertions. However, for phylogeny analysis or tracking TEi sharing pattern or understanding TEi effects on gene expression or function, we need to comprehensively find all the TEis irrespective of their type.

In table 7.1, we show how other CC founders are different than B6. As we can see, AJ, 129, NOD, NZO each have around 2000 ERVis that are not present in reference. In WSB, this number is little bit higher (2715). In CAST and PWK, the number of non-reference ERVis is more than 4000. On the other hand, there are around 3000 ERVis that are present in reference but absent in AJ, 129S1, NOD, NZO. In CAST and PWK, more than 9000 reference ERVis are absent.

Therefore, it is clear that, other mouse strains are different than the B6 reference. Even though lab strains are quite similar to each other in a larger evolutionary context (Yang et al., 2009), still a significant amount of ERVis in B6 are absent in others, also others have many non-

Type	Total ERVis	non-reference ERVis	absent reference ERVis
AJ	26146	1862	2928
129S1	26284	1972	2900
NOD	26122	1908	2998
NZO	26237	1944	2919
CAST	21921	4011	9302
PWK	21966	4016	9262
WSB	25404	2715	4523

Table 7.1: Lab strains AJ, 129, NOD, NZO are quite similar to reference, but still have around 2000 non-reference ERVis, whereas CAST, PWK has more than 4000. On the other hand, around 3000 reference ERVis are not present in all the lab strains, whereas this number is too high (> 9000) for CAST and PWK.

reference ERVis. These numbers are even higher for the types that are not lab strains, that are not classical lab strains, like WSB, CAST, and PWK. Thus, it is important to annotate ERVis in all the commonly used mouse strains. We also believe that, it is possible to do that by extending our current pipeline.

7.4 Conclusions and Final Remarks

Transposable elements (TEs) comprise a significant portion of any mammalian genome. Most of the research done so far has focused on the TE insertions (TEis) that are in somatic cells as they cause life-threatening diseases like cancer. However, germline TEis are also equally important as they are passed on to the next generation leading to genetic polymorphism. The majority of the TEis in the human and mouse genome are inactive, but still get inherited by the descendants. Therefore, similar to other variants like Single Nucleotide Polymorphism (SNPs) or small Insertion or Deletions (INDELs), it is possible to track the phylogeny in a population-based of the TEi sharing pattern. Typical variants are sometimes less powerful than TEis as they avoid repetitive regions. On the other hand, TEis are by default around repeats and have more bases to support a single event.

During my Ph.D., I focus on analyzing the diversity of TEi sharing patterns in a population. Even though there exist many algorithms for finding TEis, none of those are suitable for population-based analysis. All of those also use some sort of alignment either to a reference genome or to a TE template. Discordant-read-based algorithms which use align reads to a reference genome typically have low true positive rates. On the other hand, split-read-based algorithms which align reads to a TE template create huge computational overhead as they are not usable for any other kind of genomic analysis. Moreover, none of these can identify TEis without a given template. Template-dependent algorithms fail to find novel TE that are moving in a genome. That's why, I, at first focused on designing algorithms to eliminate several limitations of existing approaches.

Unlike existing alignment-based algorithms, I use a novel local-genome-assembly-based approach for finding the TE insertions. I showed that assembly can be performed locally starting from a repetitive region to a region with normal coverage. My approach on looking only at the TEi boundaries also avoids many irrelevant short reads which saves both time and space. I also chose data structures such as msBWT and LCP that are well-suited for finding repeats/TE or doing assembly. The cost of building these data structures are fixed and can be used in various genomic analysis other than TEis.

Even though both of my algorithms (ELITE and Frontier) are based on local-genome-assembly, ELITE is template-dependent and Frontier is template-free. Both ELITE and Frontier starts from a high coverage genomic region and continues to assemble locally towards a normal coverage one. I show that ELITE outperforms existing state-of-the-art template-dependent TEi detection algorithms in terms of precision, recall and runtime. I also designed a second algorithm Frontier which eliminates the need of any TE templates. To my knowledge, Frontier is the first algorithm out there that is template-free which finds TE insertions directly from short reads. The novel idea behind Frontier's algorithm is to use TE's well known characteristics of being repetitive to identify all the repeat boundaries. Then I use a deep learning classifier to finally pinpoint only the boundaries that belongs to TE insertions.

I also demonstrated how ELITE and Frontier together can be used to analyze the diversity of TEi in a large population. I show that my algorithms are capable of finding the location of

TE insertions almost to the basepair level. This avoids any ambiguity during merging TEis and ensures correct annotation of TEi's state no matter whether it is fixed, segregating, or de novo. By using the segregating TEi, I was also able to build phylogenetic trees showing the similarities and dissimilarities between 8 common mouse strains. All the trees I made using the TEi sharing pattern, are consistent with the established mouse phylogeny. The majority of the segregating TEis (more than 75%) are also validated using QTL mapping. The invalidated TEis are mostly due to mapping errors which can be fixed using appropriate mapping parameters.

My population-based analysis also reveals many interesting biological phenomena. For example, I found that, not all TEs are active in the Collaborative Cross population. While some of them became dormant half a million year ago, some of them continue to be active within the last 10 years (the time-point at which individual CC strains were separated from each other). This activity of TEis also varies between individual genome. Some samples tend to have more active TEis than others. This opens a new research path for the biologists who wants to investigate the underlying reason for a TE being active in an individual's genome. Using my TEi annotations, a group of geneticist already discovered that certain haplotype in chromosome 13 results in having more active ERVB7_1-LTR_MM.

I document all the TEi related tools and resources created as part of this research for future use. All my codes are available on github. I provide necessary instructions for running both ELITE and Frontier in github. This instructions can also be found in chapter 6 of this dissertation. I also made sure at least one person other than my lab group can successfully run these codes. I also created several web-tools for visualizing ERV insertions in the CC population. All these tools can be viewed under the website I created: `http://csbio.unc.edu/TEs/index.py`. Other than data visualization, I added the results of running ELITE Collaborative Cross Population as a form of spreadsheet. These can be downloaded from the website too. I hope, an easy access to these resources will benefit anyone who is interested in ERV insertions in CC Population.

I am certain that my algorithms will work better with the advancement of sequencing technology. Right now, I use short-reads, but in near future long-read sequencing will become more affordable. It will help to find the TE insertions that are nested into another TEs which will reduce the false negatives. On the other hand, Discordant-read-based methods, who try to map the whole read and its pair cannot possibly do any better if not worse. Because the longer the read, the harder it becomes to map, as the mutation rate is unpredictable in real data. Longer read will also help to identify different internal viral sequences that are enclosed by two long terminal repeats (LTRs) of any ERVs. Using my ExtendKmer 1, one can find all the versions of viral sequence that are next to an LTR. Therefore, in future there are many areas where my algorithms and pipelines can be used for better understanding of transposable elements in the genome.

REFERENCES

Abrusán, G., Grundmann, N., DeMester, L., and Makalowski, W. (2009). Teclass—a tool for automated classification of unknown eukaryotic transposable elements. *Bioinformatics*, 25(10):1329–1330.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

Babushok, D. V. and Kazazian Jr, H. H. (2007). Progress in understanding the biology of the human mutagen line-1. *Human mutation*, 28(6):527–539.

Bao, W., Kojima, K., and Kohany, O. (2015). Repbase update, a database of repetitive elements in eukaryotic genomes. mob dna. 2015; 6: 11.

Barreiro, L. B., Laval, G., Quach, H., Patin, E., and Quintana-Murci, L. (2008). Natural selection has driven population differentiation in modern humans. *Nature genetics*, 40(3):340–345.

Bauer, M. J., Cox, A. J., and Rosone, G. (2011). Lightweight bwt construction for very large string collections. In *Annual Symposium on Combinatorial Pattern Matching*, pages 219–231. Springer.

Behjati, S. and Tarpey, P. S. (2013). What is next generation sequencing? *Archives of Disease in Childhood-Education and Practice*, 98(6):236–238.

Belancio, V. P., Hedges, D. J., and Deininger, P. (2008). Mammalian non-ltr retrotransposons: for better or worse, in sickness and in health. *Genome research*, 18(3):343–358.

Bennetzen, J. L. (2000). Transposable element contributions to plant gene and genome evolution. *Plant molecular biology*, 42(1):251–269.

Bianconi, E., Piovesan, A., Facchin, F., Beraudi, A., Casadei, R., Frabetti, F., Vitale, L., Pelleri, M. C., Tassani, S., Piva, F., et al. (2013). An estimation of the number of cells in the human body. *Annals of human biology*, 40(6):463–471.

Bonizzoni, P., Della Vedova, G., Pirola, Y., Previtali, M., and Rizzi, R. (2021). Computing the multi-string bwt and lcp array in external memory. *Theoretical Computer Science*, 862:42–58.

Burrows, M. and Wheeler, D. (1994). A block-sorting lossless data compression algorithm. In *Digital SRC Research Report*. Citeseer.

Chen, J., Wrightsman, T. R., Wessler, S. R., and Stajich, J. E. (2017). Relocate2: a high resolution transposable element insertion site mapping tool for population resequencing. *PeerJ*, 5:e2942.

Chen, J.-M., Stenson, P. D., Cooper, D. N., and Férec, C. (2005). A systematic analysis of LINE-1 endonuclease-dependent retrotranspositional events causing human genetic disease. *Human genetics*, 117(5):411–427.

Chiang, C., Scott, A. J., Davis, J. R., Tsang, E. K., Li, X., Kim, Y., Hadzic, T., Damani, F. N., Ganel, L., Montgomery, S. B., et al. (2017). The impact of structural variation on human gene expression. *Nature genetics*, 49(5):692–699.

Chu, C., Nielsen, R., and Wu, Y. (2016). Repdenovo: inferring de novo repeat motifs from short sequence reads. *PloS one*, 11(3):e0150719.

Churchill, G. A., Airey, D. C., Allayee, H., Angel, J. M., Attie, A. D., Beatty, J., Beavis, W. D., Belknap, J. K., Bennett, B., Berrettini, W., et al. (2004). The collaborative cross, a community resource for the genetic analysis of complex traits. *Nature genetics*, 36(11):1133–1137.

Consortium, C. C. (2012). The genome architecture of the collaborative cross mouse genetic reference population. *Genetics*, 190(2):389–401.

Consortium, G. P., Auton, A., Brooks, L., Durbin, R., Garrison, E., and Kang, H. (2015). A global reference for human genetic variation. *Nature*, 526(7571):68–74.

Consortium, I. H. G. S. et al. (2001). Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860.

Consortium, M. G. S. et al. (2002). Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520.

Cox, A. J., Bauer, M. J., Jakobi, T., and Rosone, G. (2012). Large-scale compression of genomic sequence databases with the burrows–wheeler transform. *Bioinformatics*, 28(11):1415–1419.

da Cruz, M. H. P., Domingues, D. S., Saito, P. T. M., Paschoal, A. R., and Bugatti, P. H. (2020). Terl: classification of transposable elements by convolutional neural networks. *bioRxiv*.

Ebert, P., Audano, P. A., Zhu, Q., Rodriguez-Martin, B., Porubsky, D., Bonder, M. J., Sulovari, A., Ebler, J., Zhou, W., Mari, R. S., et al. (2021). Haplotype-resolved diverse human genomes and integrated analysis of structural variation. *Science*, 372(6537).

Egidi, L., Louza, F. A., Manzini, G., and Telles, G. P. (2019). External memory bwt and lcp computation for sequence collections with applications. *Algorithms for Molecular Biology*, 14(1):1–15.

Ewing, A. D. (2015). Transposable element detection from whole genome sequence data. *Mobile DNA*, 6(1):24.

Fernández, X. M. and Birney, E. (2010). Ensembl genome browser. In *Vogel and Motulsky's Human Genetics*, pages 923–939. Springer.

Ferragina, P. and Manzini, G. (2000). Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 390–398. IEEE.

Ferragina, P., Manzini, G., Mäkinen, V., and Navarro, G. (2004). An alphabet-friendly fm-index. In *International Symposium on String Processing and Information Retrieval*, pages 150–160. Springer.

Fiston-Lavier, A.-S., Barrón, M. G., Petrov, D. A., et al. (2014). T-lex2: genotyping, frequency estimation and re-annotation of.

Gardner, E. J., Lam, V. K., Harris, D. N., Chuang, N. T., Scott, E. C., Pittard, W. S., Mills, R. E., Devine, S. E., Consortium, . G. P., et al. (2017). The mobile element locator tool (melt): population-scale mobile element discovery and biology. *Genome research*, pages gr–218032.

Greenstein, S., Holt, J., and McMillan, L. (2015). Short read error correction using an fm-index. In *Bioinformatics and Biomedicine (BIBM), 2015 IEEE International Conference on*, pages 101–104. IEEE.

Hancks, D. C. and Kazazian, H. H. (2016). Roles for retrotransposon insertions in human disease. *Mobile DNA*, 7(1):1–28.

Hoede, C., Arnoux, S., Moisset, M., Chaumier, T., Inizan, O., Jamilloux, V., and Quesneville, H. (2014). Pastec: an automatic transposable element classification tool. *PloS one*, 9(5):e91929.

Holt, J. and McMillan, L. (2014). Constructing Burrows-Wheeler transforms of large string collections via merging. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 464–471. ACM.

Holt, J. M., Wang, J. R., Jones, C. D., and McMillan, L. (2016). Improved long read correction for de novo assembly using an fm-index. *bioRxiv*, page 067272.

Jiang, C., Chen, C., Huang, Z., Liu, R., and Verdier, J. (2015). Itis, a bioinformatics tool for accurate identification of transposon insertion sites using next-generation sequencing data. *BMC bioinformatics*, 16(1):72.

Kasai, T., Lee, G., Arimura, H., Arikawa, S., and Park, K. (2001). Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Annual Symposium on Combinatorial Pattern Matching*, pages 181–192. Springer.

Kashfeen, A., Fauni, H. B., Bell, T. A., Pardo-Manuel de Villena, F., and McMillan, L. (2019). Elite: Efficiently locating insertions of transposable elements. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 183–189.

Keane, T. M., Wong, K., and Adams, D. J. (2012). Retroseq: transposable element discovery from next-generation sequencing data. *Bioinformatics*, 29(3):389–390.

Kent, W. J. (2002). Blat—the blast-like alignment tool. *Genome research*, 12(4):656–664.

Koch, P., Platzer, M., and Downie, B. R. (2014). Repark—de novo creation of repeat libraries from whole-genome ngs reads. *Nucleic acids research*, 42(9):e80–e80.

Kuhn, R. M., Haussler, D., and Kent, W. J. (2013). The ucsc genome browser and associated tools. *Briefings in bioinformatics*, 14(2):144–161.

Lander, E. S., Linton, L. M., Birren, B., Nusbaum, C., Zody, M. C., Baldwin, J., Devon, K., Dewar, K., Doyle, M., FitzHugh, W., et al. (2001). 2001. initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921.

Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with bowtie 2. *Nature methods*, 9(4):357.

Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome biology*, 10(3):R25.

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

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R. (2009). 692 (2009). the sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–693.

Li, Y., Salo-Mullen, E., Varghese, A., Trottier, M., Stadler, Z. K., and Zhang, L. (2020). Insertion of an alu-like element in mlh1 intron 7 as a novel cause of lynch syndrome. *Molecular Genetics & Genomic Medicine*, 8(12):e1523.

Lilue, J., Doran, A. G., Fiddes, I. T., Abrudan, M., Armstrong, J., Bennett, R., Chow, W., Collins, J., Collins, S., Czechanski, A., et al. (2018). Sixteen diverse laboratory mouse reference genomes define strain-specific haplotypes and novel functional loci. *Nature genetics*, page 1.

Mantaci, S., Restivo, A., Rosone, G., and Sciortino, M. (2007). An extension of the burrows–wheeler transform. *Theoretical Computer Science*, 387(3):298–312.

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

McClintock, B. (1984). The significance of responses of the genome to challenge.

Morgan, A. P., Fu, C.-P., Kao, C.-Y., Welsh, C. E., Didion, J. P., Yadgary, L., Hyacinth, L., Ferris, M. T., Bell, T. A., Miller, D. R., et al. (2016). The mouse universal genotyping array: from substrains to subspecies. *G3: Genes, Genomes, Genetics*, 6(2):263–279.

Mose, L. E., Perou, C. M., and Parker, J. S. (2019). Improved indel detection in dna and rna via realignment with abra2. *Bioinformatics*, 35(17):2966–2973.

Muratani, K., Hada, T., Yamamoto, Y., Kaneko, T., Shigeto, Y., Ohue, T., Furuyama, J., and Higashino, K. (1991). Inactivation of the cholinesterase gene by alu insertion: possible mechanism for human gene transposition. *Proceedings of the National Academy of Sciences*, 88(24):11315–11319.

Nellåker, C., Keane, T. M., Yalcin, B., Wong, K., Agam, A., Belgard, T. G., Flint, J., Adams, D. J., Frankel, W. N., and Ponting, C. P. (2012). The genomic landscape shaped by selection on transposable elements across 18 mouse strains. *Genome biology*, 13(6):1–21.

Rishishwar, L., Mariño-Ramírez, L., and Jordan, I. K. (2016). Benchmarking computational tools for polymorphic transposable element detection. *Briefings in Bioinformatics*, 18(6):908–918.

Robb, S. M., Lu, L., Valencia, E., Burnette, J. M., Okumoto, Y., Wessler, S. R., and Stajich, J. E. (2013). The use of relocate and unassembled short reads to produce high-resolution snapshots of transposable element generated diversity in rice. *G3: Genes, Genomes, Genetics*, pages g3–112.

Roberts, A., Pardo-Manuel de Villena, F., Wang, W., McMillan, L., and Threadgill, D. W. (2007). The polymorphism architecture of mouse genetic resources elucidated using genome-wide resequencing data: implications for qtl discovery and systems genetics. *Mammalian Genome*, 18(6):473–481.

Salikhov, K., Sacomoto, G., and Kucherov, G. (2013). Using cascading bloom filters to improve the memory usage for de brujin graphs. In *International Workshop on Algorithms in Bioinformatics*, pages 364–376. Springer.

Sanger, F. and Coulson, A. R. (1975). A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *Journal of molecular biology*, 94(3):441–448.

SanMiguel, P., Tikhonov, A., Jin, Y.-K., Motchoulskaia, N., Zakharov, D., Melake-Berhan, A., Springer, P. S., Edwards, K. J., Lee, M., Avramova, Z., et al. (1996). Nested retrotransposons in the intergenic regions of the maize genome. *Science*, 274(5288):765–768.

Simpson, J. T. and Durbin, R. (2010). Efficient construction of an assembly string graph using the fm-index. *Bioinformatics*, 26(12):i367–i373.

Siva, N. (2008). 1000 genomes project. *Nature biotechnology*, 26(3):256–257.

Smit, A. F., Hubley, R., and Green, P. (1996). Repeatmasker.

Stumpf, P. S., Du, X., Imanishi, H., Kunisaki, Y., Semba, Y., Noble, T., Smith, R. C., Rose-Zerili, M., West, J. J., Oreffo, R. O., et al. (2020). Transfer learning efficiently maps bone marrow cell types from mouse to human using single-cell rna sequencing. *Communications biology*, 3(1):1–11.

Teissandier, A., Servant, N., Barillot, E., and Bourc'his, D. (2019). Tools and best practices for retrotransposon analysis using high-throughput sequencing data. *Mobile DNA*, 10(1):1–12.

Teugels, E., De Brakeleer, S., Goelen, G., Lissens, W., Sermijn, E., and De Grève, J. (2005). De novo alu element insertions targeted to a sequence common to the brca1 and brca2 genes. *Human mutation*, 26(3):284–284.

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.

Uzunović, J., Josephs, E. B., Stinchcombe, J. R., and Wright, S. I. (2019). Transposable elements are important contributors to standing variation in gene expression in capsella grandiflora. *Molecular biology and evolution*, 36(8):1734–1745.

van den Hurk, J. A., van de Pol, D. J., Wissinger, B., van Driel, M. A., Hoefsloot, L. H., de Wijs, I. J., van den Born, L. I., Heckenlively, J. R., Brunner, H. G., Zrenner, E., et al. (2003). Novel types of mutation in the choroideremia (chm) gene: a full-length l1 insertion and an intronic mutation activating a cryptic exon. *Human genetics*, 113(3):268–275.

Wallace, M. R., Andersen, L. B., Saulino, A. M., Gregory, P. E., Glover, T. W., and Collins, F. S. (1991). A de novo alu insertion results in neurofibromatosis type 1. *Nature*, 353(6347):864.

Wang, J. R., Holt, J., McMillan, L., and Jones, C. D. (2018). Fmlrc: Hybrid long read error correction using an fm-index. *BMC bioinformatics*, 19(1):50.

Waterston, R. H. and Pachter, L. (2002). Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420(6915):520–562.

Wheeler, T. J., Clements, J., Eddy, S. R., Hubley, R., Jones, T. A., Jurka, J., Smit, A. F., and Finn, R. D. (2012). Dfam: a database of repetitive dna based on profile hidden markov models. *Nucleic acids research*, 41(D1):D70–D82.

Yang, H., Bell, T. A., Churchill, G. A., and Pardo-Manuel de Villena, F. (2007). On the subspecific origin of the laboratory mouse. *Nature genetics*, 39(9):1100–1107.

Yang, H., Ding, Y., Hutchins, L. N., Szatkiewicz, J., Bell, T. A., Paigen, B. J., Graber, J. H., de Villena, F. P.-M., and Churchill, G. A. (2009). A customized and versatile high-density genotyping array for the mouse. *Nature methods*, 6(9):663–666.

Zhou, S., Herschleb, J., and Schwartz, D. C. (2007). A single molecule system for whole genome analysis. *Perspectives in Bioanalysis*, 2:265–300.

Zhuang, J., Wang, J., Theurkauf, W., and Weng, Z. (2014). Temp: a computational method for analyzing transposable element polymorphism in populations. *Nucleic acids research*, 42(11):6826–6838.