Frontier: Finding the Boundaries of Novel Transposable Element Insertions in Genomes

Anwica Kashfeen
anwica@cs.unc.edu
UNC Chapel Hill
Chapel Hill, NC, USA

Leonard McMillan
mcmillan@.cs.unc.edu
UNC Chapel Hill
Chapel Hill, NC, USA

ABSTRACT

Transposable Elements (TEs) are DNA subsequences that have historically copied themselves throughout a genome. Apart from constituting a large fraction of all eukaryotic genomes, TEs are a significant source of genetic variation and are directly responsible for many diseases. TEs are also one of the most difficult genomic regions to analyze. A typical approach for identifying TE insertions (TEi) involves the detection of *split-reads*, which requires checking if each read can be split into TE and non-TE parts. Identification of the TE part depends on a model for each distinct TE class, and these classes vary significantly both within and between species. Previous methods for detecting segregating TEis depend on template libraries and their computational cost increases with the number of templates. Here we propose a novel template-free method for identifying the split-reads with TEis called *Frontier*. We define Frontier as the boundary between highly repeated and normal genomic sequence. We leverage the pervasiveness of TE sequences to identify candidate reads that might include the boundary of an insertion. We then apply machine learning methods to further classify whether the read includes actual TE-like sequence. For each predicted TEi boundary we apply a second classifier to infer the corresponding TE type (LINE, SINE, ALU, ERV/LTR). Both classifiers achieve high precision (> .9), recall (> .8) and F1 score (> .8) when applied to real data. The resulting trained model, can detect and classify about 50 million frontier reads in less than an hour.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Applied computing** → **Computational biology**.

KEYWORDS

neural networks, split-read, msBWT, LCP, transposable element

ACM Reference Format:
Anwica Kashfeen and Leonard McMillan. 2018. Frontier: Finding the Boundaries of Novel Transposable Element Insertions in Genomes. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Repetitive DNA sequence is found throughout the genome in many species including plants, bacteria, and mammals[21][37][3]. Analyzing and distinguishing between the numerous copies present in the genome is a computational challenge, and, thus, is often ignored when analyzing genetic variation[30]. Current variant detection methods focus mostly on detecting Single Nucleotide Polymorphisms (SNPs) and small insertions or deletions (INDELS) as they are relatively easier to detect in high-throughput short-read sequencing data[35]. Moreover, repetitive DNA sequences introduce ambiguity when aligning and assembling short reads, especially in cases when the repeated element is longer than the read fragment[32][31]. Researchers often use programs like Repeatmasker [29] and Dust [26] to mask known repetitive regions before performing genomic analysis. Reportedly 45% of the mouse and 53% of the human genome is masked by Repeatmasker.

Masking or ignoring repeats simplifies many analyses but hinders the chance to discover other interesting biological phenomena. Insertions of a particular class of repetitive sequence, called Transposable Elements (TEs)[25], can lead to serious functional consequences, as they are mobile and frequently inserted within and near genes[36][34][27][7][2]. RNA-mediated TEs employ a copy-paste mechanism of insertion that, over time, increases the size of the genome. Moreover, recent studies suggest that many segregating structural variants in humans are due to TE insertions[10]. In comparison to SNPs, TE insertions are believed to have more pronounced effects on gene expression and phenotypic variation in some organisms[33].

Over the past two decades, many TE or SV detection tools have been developed. Most of these methods use discordant read pairs, split reads, or a combination of the two to identify non-reference TEs. Since a split read spans the boundary of a TE insertion, it can be used to precisely detect the insertion location and any associated Target Site Duplication (TSD). However, split-read TE detection requires comparing each read to one or more TE templates to detect a boundary between TE-like and non-TE-like sequences and is therefore computationally expensive. 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. Templates for TEs can be found in libraries like RepeatMasker, Dfam, Repbase but are limited to TE sequences identified in a 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 we developed a template-free TE discovery pipeline that automatically identifies all the spanning boundaries of TE insertions under the assumption that the target TE appears with high frequency in the genome.

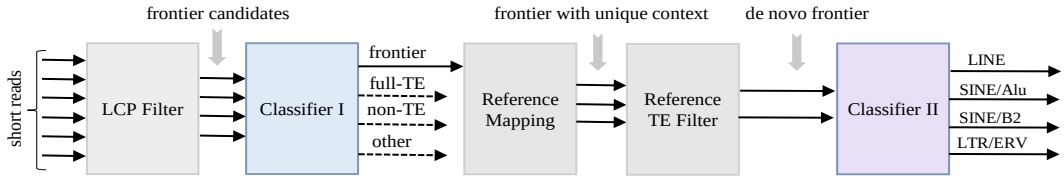


Figure 1: Frontier Pipeline: Initially we use the Longest-Common Prefixes (LCPs) of the suffix array from a full short-read sequencing dataset (represented as a multi-string Burrows-Wheeler Transform (msBWT)) to extract all substrings of reads where a sizable prefix is highly repeated, but, then appears as normal genomic coverage when the prefix is extended. 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), no repetitive sequence (non-TE), and repetitive sequence 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.

In an attempt to find TE insertions, we start with finding *Frontier* reads. Frontier reads include the boundaries of TE insertions, where on one side of the boundary the sequence is highly repeated, and the other side appears to have normal coverage. We introduce a template-free, data-driven approach by exploiting TE's repetitive property. We 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. We then consider sub-intervals of the unidentified repeat intervals to find extended sequences with normal genomic coverage. We call these *Frontier candidates*. However, there are certain cases where a non-frontier candidate might look like a true Frontier. Such as, 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*, we propose a deep learning model which takes filtered short-reads i.e., *Frontier Candidates* as inputs.

We 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. We transform the sequence of a *Frontier Candidate* into a matrix similar to a one-hot vector and feed it directly to the *Classifier*-I. We 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. We then filter all the frontiers that are also present in the reference. This provides a list of non-reference novel frontiers, which are then fed to the *Classifier*-II to predict the TE type. Here we included the 4 major classes of TEs present in mouse reference genome [28]. We use the short read dataset of a C57BL/6J (B6) strain, which is the basis for the mouse reference genome (mm10), to build training data for *Classifier*-I and *Classifier*-II. Using RepeatMasker's TEi annotation of the mouse reference, we labeled all the frontier reads of that dataset. After training both classifiers based on the B6 data, we used them to predict frontiers in 7 other mouse strains. We showed on average it can find more than 80% of the Frontiers with high precision (≈ 0.9). Additionally, once the classifier is trained, on average it takes less than an hour to classify all the frontier candidates.

2 RELATED WORK

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 [8], RepARK [20] and others use a program like Jellyfish [24] 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 [13], TEMP [38], RetroSeq [19] 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, these methods assume that there is a TE has been inserted somewhere around the uniquely mapped read. However, the insertion location reported by these methods can be significantly off (0-300bp). Alignment-based methods also tend to report a considerably high number of false positives. Moreover, these are heavily biased in favor of TE-templates present in the reference.

A third class of TE insertion methods are split-read-based, and can precisely detect the insertion location to the basepair. These methods find a "break-point" where a 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 the boundary of a TE, these tools like ELITE [18], Relocate2 [6], ITIS [16] rely on a known TE template in advance. Relocate2, ITIS 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 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 novel TE insertions because of their dependency on a known TE template.

The Frontier pipeline identifies repeats directly from short reads without templates based on the longest-common prefixes of adjacent entries of a suffix array. These repeated suffix-prefixes are then extended further to only keep the segments that have a normal coverage thus suggesting a single copy. This also ensures a higher chance that the extended segment might be mapped uniquely. We call these frontier candidates where repeated subsequence extends into a single copy sequence. These candidates are then run through a classifier to predict if they contain a TE and non-TE segment adjacent to each other – we call such sequences the actual Frontier. Finally, in a separate classifier, we 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 a TE class. Examples of such tools for classifying TE sequences include [9], [14] and [1]. Our pipeline's main contribution is to use machine learning to identify split-reads from the set of all reads in a sequenced dataset without the need of any TE-template.

3 BACKGROUND

We use two data structures called a multi-string Burrows-Wheeler Transform (msBWT) and Longest Common Prefix (LCP) array in the first stage of our discovery pipeline. The Burrows-Wheeler Transform (BWT) [5] was originally developed to compress text data. The transformation effectively determines the sequence of predecessors of all the sorted suffixes of a text, and it is a permutation of the original text. This BWT permutation can also be inverted to reconstruct the original text. The BWT 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 original text.

Beyond compressing data, the BWT has also been shown to be as efficient as a suffix-tree for performing substring searches. Using a light-weight auxiliary data structure called an FM-index[12]. 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 its essential properties. A BWT of a string collection is called a multi-string BWT. Holt [15] showed that assembling BWTs for multiple strings can be done incrementally by merging msBWTs. Another auxiliary BWT data structure called the LCP [17] 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 [11] [4]. We used Holt's approach [15] for building the msBWT and LCP, because of the supporting functions offered in the supplied API. The relationship between the BWT and LCP data structures and their use for finding repeat intervals is illustrated in figure2.

Text: GGGCATGGGGTAGGGGCAT$

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

Figure 2: 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.

4 METHOD

Given a short-read high-throughput sequence dataset, we aim to find novel *frontiers* i.e., the boundary of unannotated TE insertions. Since TEs are repeated throughout the genome, we expect any TE sequence to have unusually high coverage when compared to a regular genomic sequence. Therefore, we identify all the reads that contain a subsequence that appears with a very high count (repeat), followed by a subsequent sequence with a normal count (context). The boundary between this repeat/potential TE and context/non-TE segment is called the *Frontier* and reads containing this pattern are called *Frontier Candidates*. However, not all the repeats are TEs nor are all the contexts non-TE. Therefore to find the true *frontier*, we apply a machine learning classifier to distinguish these cases. Contexts of each frontier are then mapped to a reference genome to identify the chromosome and position of insertion. Once we know the position, we revisit the reference genome to check if there is an annotated TE nearby. If not we hypothesize an insertion, we annotate it as a *novel TEi*. For each novel TEi, we further classify it according to TE type (LINE, SINE, ALU, LTR/ERV) using a second classifier. The pipeline is shown in figure1.

Algorithm 1 Finding Frontier Candidate

```
procedure FindRepeat(K, , LCP, m)
    repeat = ∅
    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++
        low = i
        while chunk[i] ≥ K do
            i++
            count++
        high = i
        if count ≥ m − 1 then
            repeat = repeat ∪ new_repeat
                                ▷ m = minimum frequency of repeat
                                    ▷ K = length of repeat segment
```

4.1 Identifying Frontier Candidates

At first, we build two data structures i.e., msBWT and 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, we find all the k-mers that occurred 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. We 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, we break the LCP into smaller chunks and process each chunk in parallel, and finally merge the results. After finding all the genomic repeat intervals, the algorithm then proceeds to subdivide them into subintervals with normal coverage. Once again the LCP is scanned to find extended prefixes contained within the 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 and called *frontier candidates*.

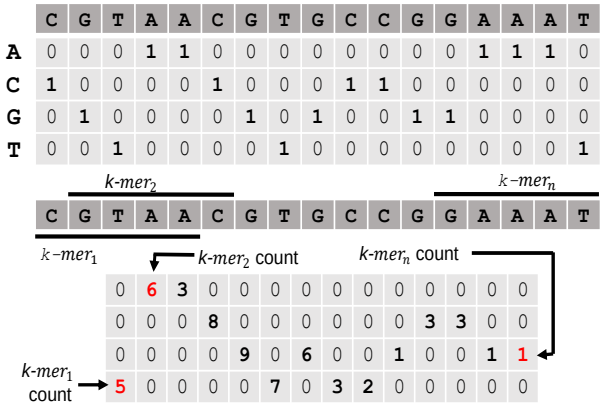


Figure 3: Input data representation: We use a modified one-hot vector encoding to represent input sequences of our classifiers where the *one* values are replaced by a log_2(k-mer count) of a subsequence centered around the represented base. This effectively trims the sequence by $k-1$. The k-mer counts used are the sum of the subsequence'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.

4.2 Identifying True Frontiers

In this step, we classify each read from the previous step to predict whether it is an actual Frontier or not (non-TE, Full-TE and others). As described earlier, a Frontier is a boundary between TE and non-TE context. Each read from previous steps has two adjacent parts, one where the repeat/TE side has high coverage and the other side has normal coverage. But mutations or interruption from other TEs or repeats can also cause a drop in k-mer count. Besides, common non-TE repeats like microsatellites, telomeric sequence, and tandem repeats are not TEs even though they also tend have very high coverage. Thus to specifically identify TEs, we build a machine learning model to filter frontier candidates. This model takes frontier candidates and k-mer frequency counts from overlapping windows from the candidate as input and outputs their classes i.e. whether true Frontier, TE, non-TE or other.

4.2.1 Input Data Preperation. Each input of our model consists of a read and its corresponding overlapping-kmer 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 we map each R-length sequence into a 4xR dimensional matrix for each read, where R is the read length and 4 rows correspond to the 4 bases. If at any position, the sequence in a read has A, then in that position, we put 1 in the A's row, and 0 in the others, using a so-called 1-hot encoding. The same goes for all the other bases. The second input we use is the overlapping *k-mer* count. We then divide the sequence into $R-k+1$ segments or k-mers and count the occurrence of each k-mer in the dataset using the msBWT. Then we 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 $R-k+1$ kmers from a

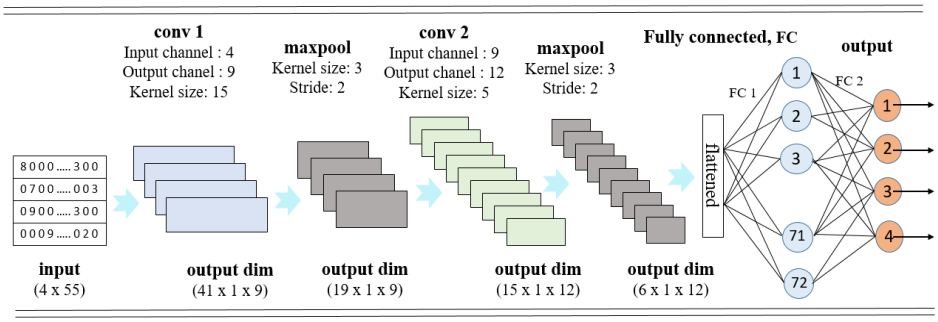


Figure 4: Architecture of deep-learning neural network used: A 4-layer convolutional neural network is used for classiying Frontier. First two layers are CNN, each of these are followed by a maxpool and a RelU. The flattened output from CNN 2 is then fed to a fully connected linear layer. Final layer's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

read of length R. Thus, we 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.

4.2.2 Model Description. We designed a 4-layer deep neural network to classify each frontier read. The first two layers are CNNs, followed by two linear layers. The architecture is shown in figure 4. The first CNN 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 $4xR - k + 1$ matrix for each read, is now divided into 4 vectors. Each of these 4 vectors is fed to each input channel independently. We use 1D convolution as we 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, representing the probability of being 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 repeat types of uncategorized reads fall into the other class. Even though we are only interested in the frontier class, suggesting that a binary classifier should work, in practice, we observed that having 4 classes improves the classification results 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 a hugely imbalanced dataset which is not preferred for a good classifier.

4.3 Mapping Frontier Context

After finding all the true frontiers, we then identify their positions in a reference genome. We determine the position of the context (i.e., the segment of the frontier that has normal genomic coverage) using bowtie2[22] (which is also BWT based). Mapping the Frontier's context allows us to find the TE's insertion location in terms of the reference genome's chromosome and position. We only keep the frontiers where the context maps uniquely. For each unique context, we consider the surrounding sequence from reference and check if it contains any annotated repeat/TE. If no repeat is found, then we annotate it as a Novel Transposable Element Insertion.

4.4 Identifying Frontier's TE type

The final step of our pipeline is to classify each *Novel frontier* based on its TE type. We 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. We 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. Using this classifier for any species other than mouse might require a change in the number of output classes, which can be trivially done by changing parameters in the network model.

5 EXPERIMENTS

We applied our Frontier pipeline to eight short-read sequencing datasets. This includes five classical inbred laboratory strains and three wild-derived strains. All samples were sequenced using either Illumina X10 or Illumina HiSeq4000 sequencers with paired-end reads. One sample, C57BL/6J incorporates multiple sequencing runs and is a mix of read lengths, 125-bp to 151-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 and corresponding LCP as described by Holt[15]. The details for each sample are shown in Table 1.

Type	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 1: Samples used for evaluating the performance of two Frontier classifiers. B6 sample C57BL/6J is also used for making training dataset. Trained models obtained from B6 were directly applied to other samples to find their frontiers and corresponding type.

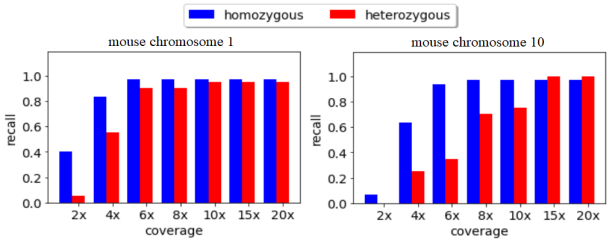


Figure 5: Recall rate of algorithm 1 on simulated data. The X-axis shows reads coverage, and Y axis shows the recall rate ranging from 0 to 1, Recall for homozygous and heterozygous frontier discovery is shown in the blue and red bar respectively. For both chromosome 1 and 10, the recall rate increases with increased coverage, specially for the cases where TEi was inserted heterozygous. However, the recall rate stablizes (≈ .9) when the coverage is around 15x.

5.1 Evaluation of Algorithm 1

We applied our algorithm 1 on both simulated and real datasets. For each dataset, we first found all the intervals in the msBWT that contain repeats. We divided the LCP array into multiple segments and sent each one to a different processor for finding repeats in parallel. We 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, we expect to see it around 10x40 = 400 times in short reads. After finding the repeat intervals, we looked for intervals that are adjacent to a context with normal coverage. We set the length of repeat and context to 45 and 30 respectively. So each Frontier candidate has a length of 75. We 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. We 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. We also built msBWT and LCP on this machine. We used simulated data for evaluating algorithm 1.

To estimate the recall rate of frontier candidate discovery, we ran 1 on simulated data where ground truth is known. We inserted 50 TEs into chromosome 1 and chromosome 10 of the mouse reference genome. For each insertion, we 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. We used Samtools [23] to simulate 150-bp reads at different coverages ranging from 2x to 20x. we ran 1 on both chromosomes to get a list of frontier candidates. To measure the recall rate, we checked if each ground truth frontier candidate is found by running algorithm 1. Since there are many old existing TEis already in the mouse genome, the algorithm found more candidates than the ones we inserted. That's why we calculated only the recall rate and skipped precision. Recall rates for both chromosomes are shown in figure5. As expected, frontier candidates with heterozygous TEi require higher coverage compared to homozygous TEis to be found. When the coverage was around 15x, our algorithm 5 was able to find 95% of the candidates irrespective of the zygosity.

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

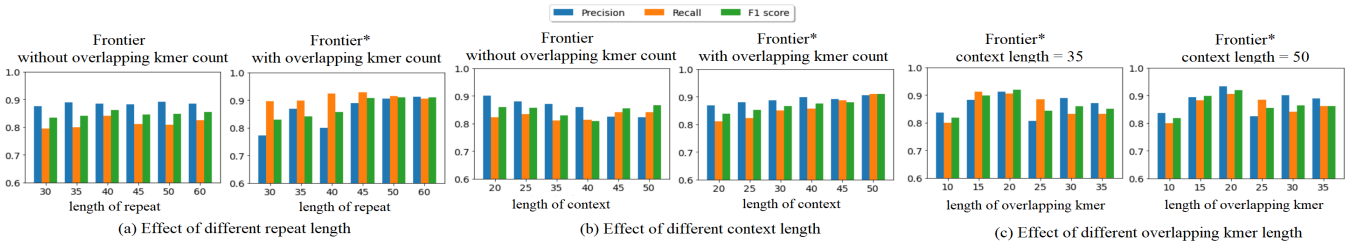
Table 2: Comparison between baseline and proposed Frontier classifiers. The Baseline performs poorly in general as compared to classifier that uses and LCP-filter. Frontier* performs the best by using the overlapping k-mer count. However, for a very large dataset, the overhead of counting all the overlapping k-mer is high. So we recommend using Frontier for a large dataset as it can still predict more than 84% of the frontiers with high precision (.884).

5.2 Building Frontier Training Dataset

For each frontier candidate, we searched for their position in the reference and checked if RepeatMasker has annotated any TEi in that region. We found a total of 33331533 candidates that either contain partial or full TE sequence. Candidates that contain partial TE sequence are in class (i): Frontier. We 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. We had a total of 895332 examples in class (i): frontier. There were many examples for class (ii),(iii), and (iv), but we sub-sampled them by randomly selecting 895332 examples to make our training dataset balanced. For each read, we 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. For our second classifier, we used RepeatMasker's annotation to label the 4 different TE types present in each Frontier.

5.3 Performance of *Classifier*-I on B6

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



(a) Effect of different repeat length (b) Effect of different context length (c) Effect of different overlapping kmer length

Figure 6: Effect of three different parameters on the performance of *Classifier*-I. X-axis shows the different values of parameters i.e., repeat lengths ranges from 30 to 60 for (a), context lengths ranges from 20 to 50 for (b), overlapping kmer lengths ranges from 10 to 35 for (c). For all three plots, Y-axis shows precision, recall and f1-score for the corresponding parameter values (possible ranges from 0 to 1). (a) For different repeat lengths, precision and f1 score of Frontier* (with overlapping kmer count) increases with the increase in repeat length and becomes stable when the length is around 45. No significant change is observed for frontier (without overlapping kmer count). (b) For different context lengths, precision of Frontier (without kmer count) decreases with the increase in context length. On the other hand, for frontier* (with overlapping kmer count), precision, recall, f1-score all improved as context length increases. (c) For different overlapping kmer lengths, frontier* (with overlapping kmer count) performs best when the length is around 15-20 bases irrespective of the length of context.

To obtain the result of our proposed classifier, we used the dataset described in 5.2. Here we 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, we 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 2 in terms of three metrics: precision, recall, and F1 score. Since this is a multi-class classifier, we 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.

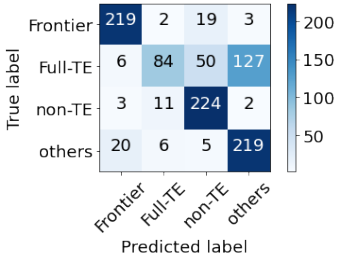


Figure 7: Confusion Matrix showing the result of running Classifier I on 1000 randomly selected candidates. Each row of this matrix shows the True labels, and each column shows the predicted labels. There are 243 Frontiers, and our classifier misclassifed 24 examples in total. It also misclassified many Full-TEs as others. The classification errors of Full TEs have no impact since we only process further those reads identified as Frontier. Many of these appear to be older and highly mutated TE insertions.

We also provide a confusion matrix to demonstrate the classifier's performance in predicting other classes. The matrix is shown in figure 7. Here we 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. We investigated this small 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, we achieve a recall rate >.9 for the other classes and we care only for the candidates classified as Frontier in this step.

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

We have three parameters in our pipeline that initially were chosen in an ad-hoc fashion. One of these is the length of the repeat portion of a frontier candidate. Recall that in the first step of our pipeline 4.1, we 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 choice for repeat length, we varied it ranging from 30 to 60 base pairs and observed the classifier's performance. We ran our two separate models where one considered the overlapping k-mer count and the other one did not. Figure 6(a) shows the effect of different repeat lengths on our classifier. The length of overlapping k-mer counts was kept fixed at 21. For different repeat lengths, we ran our classifier and calculated its precision, recall, and F1 score. Even though we did not see any significant change in the performance of Frontier (without k-mer count), the precision and f1 score for Frontier* decreased with the decrease in candidate length.

Another parameter used while searching for frontier candidates was the context length. Initially, we set the context length to 30. But to see if the length of context affects the classifier's performance, we ran the same model with different context lengths. The performance comparison is shown in figure 6(b). The Frontier* classifier that uses overlapping k-mer count performed better and recall rate increased with the increase in context length. For this experiment, we kept fixed the length of overlapping kmer to 21 and repeat length to 45. From our experiment, we observed that, frontier* discovered more true frontiers and achieved the best recall rate. We believe this is due to incorporating the overlapping k-mer counts as a signal of the expected variable coverage across a Frontier read.

The last parameter we varied is the length of overlaping 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 we used overlapping kmer count as *Classifier*-I's input in 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, we 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, we vary its value ranging from 10 to 35 by keeping the context length fixed. For each different k, we then ran the *Classifier*-I. The observed results are shown in figure 6(c).

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

We used our trained model from *Classifier*-I to predict frontiers of 7 other non-B6 samples. At first, we used the LCP-filter to identify the frontier candidates by using algorithm 1. We randomly selected 50000 candidates from each sample and ran *Classifier*-I. Here we applied *Frontier* classifier that does not use overlapping kmer count. For validation, we obtained the class label by querying the *candidate* in RepeatMasker because we cannot compare it with the reference. For this experiment, we 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 we 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 our training data used only frontier candidates from a B6 sample, we 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 3.

5.6 Performance of *Classifier*-II

After identifying the *frontiers*, we applied several steps to identify non-reference Novel TE frontiers in a sample. These frontiers were then classified based on their TE type. Our 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, we randomly selected 1000 frontiers from each sample. Our 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/EiJ, PWK/PhJ, and WSB/EiJ) are expected to differ more significantly from the reference than the four common laboratory strains (A/J, 129S1/SvImJ, NOD/ShiLtJ, NZO/HlLtJ), as they likely share more recent common ancestors.

Strain	Sample	Precision	Recall	F1 Score
A/J	f321	0.833	0.795	0.814
129S1/SvlmJ	m157	0.907	0.810	0.855
NOD/ShiLtJ	m146	0.923	0.766	0.837
NZO/F111	m146	0.939	0.705	0.805
CAST/EiJ	m090	0.921	0.761	0.833
PWK/PhJ	f6114	0.855	0.691	0.764
WSB/EiJ	f111	0.952	0.800	0.870

Table 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.

In the confusion matrices shown in 8, we 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 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.

5.7 Finding Novel Frontiers

We applied our pipeline to 7 to mouse genomes sequenced at 30x coverage with 150bp reads. We first constructed msBWTs and LCPs for each. Then using these two data structures, we identified frontier candidates. As before, we 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 we ran *Classifier*-I (without overlapping kmer count) to predict the true frontiers from these candidates. On average, each sample generated around 50 million frontier candidates and less than 10% of these were predicted as a true frontier. For each predicted true frontier, we then aligned the 30-base context to the mouse reference genome (mm10) to find the location of TE insertion. We kept only those contexts that mapped uniquely in the genome. In our dataset, approximately one-third of the contexts of the predicted true frontiers mapped to a unique genomic position. We next merged all aligned contexts with consistent 45-mer prefixes within 100 bases of each other. Frequently there are two nearby contexts that represent the two sides adjacent

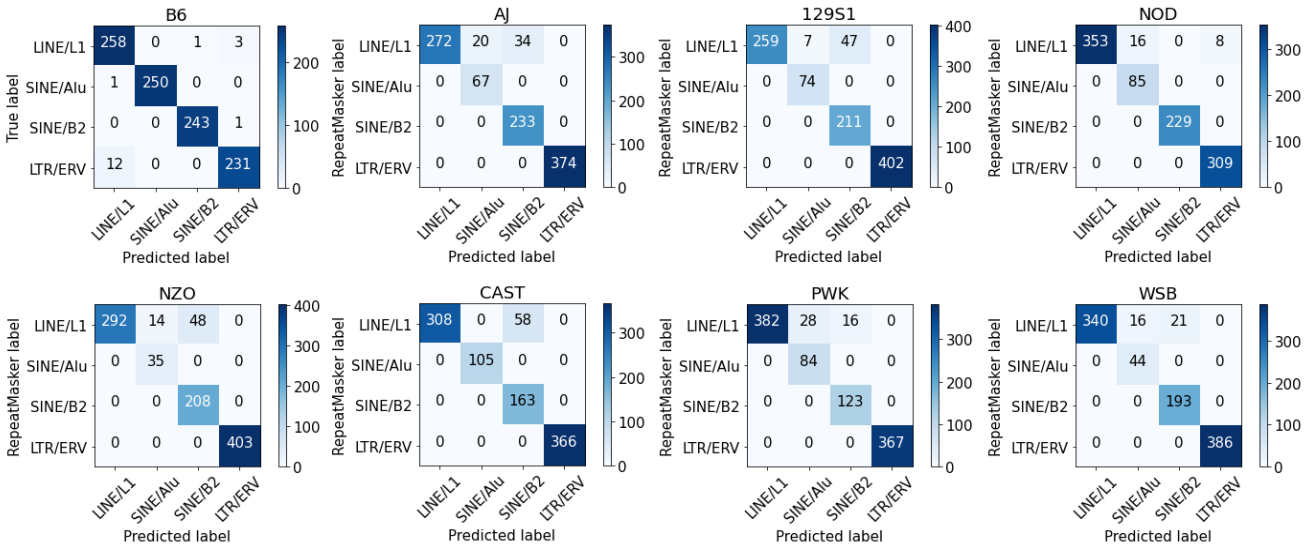


Figure 8: Confusion matrices showing the result of running our TE classifier *Classifier*-II on 1000 frontier sequences from samples representing 8 mouse strains. We compare our predicted labels (columns) to those of RepeatMasker (rows). RepeatMasker annotates TE insertions in the mouse reference genome (mm10) which is based on the mouse strain C57BL/6J for which the sample B6 is an example. Thus, we considered Repeatmasker as ground truth for B6, a balanced set of the four TE types. For all other 7 samples we merely compare our 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%.

to the insertion. We next checked if there was an annotated TE in the reference near each of the insert locations. If no TEi was present nearby, we reported those as novel TE insertions. For each sample, we found around 4000 novel TEis that are not annotated in the reference genome. Finally, for these novel insertions, we applied *Classifier*-II to predict the type of TE.

These analysis suggest that there are a considerable number of non-reference TE insertions (TEis). Details are shown in table 4. Overall we found more than 18000 novel TEis which includes 1318 LINE, 457 SINE/ALU, 923 SINE/B2, and 1677 LTR/ERV insertions. As expected the four laboratory strains (A/J, 129S1/SvImJ, NOD/ShiLtJ, NZO/HlLtJ) tend to share more TEis with the reference (also based on a laboratory strain B6) than the three wild-derived mouse strains (CAST/EiJ, PWK/PhJ, and WSB/EiJ). The degree of shared TEis with the reference is indicated by the ratio of Novel TEis to the number of merged positions.

6 RUNTIME

The runtime of our frontier discovery pipeline is dominated by the first step of finding 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 3 hours to find all the frontier candidates. The time required for building msBWT and LCP took about 7 hours. Overall our pipeline was completed in 10 hours. However, we should not consider the construction time of msBWT and LCP as they are a fixed initial cost. Additionally, both of these 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.

7 CONCLUSION

We proposed a novel approach for finding de novo Transposable Element insertions based on identifying Frontier/split-reads and using machine learning. Unlike the typical alignment-based approach, our pipeline does not depend on any known TE template. 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.

There are a lot of areas for future improvements to our Frontier-based TE discovery approach. For example, at present, we can only search for the repeats that occur more than a given threshold. But sometimes TE insertions are mutated during retrotranscription and the mutated version may not have a high count near the insertion point. Therefore, in the future, we plan to modify our repeat searching algorithm to allow for some mutations from a highly repeated k-mer found elsewhere in the dataset.

REFERENCES

[1] György Abrusán, Norbert Grundmann, Luc DeMester, and Wojciech Makalowski. 2009. TEclass—a tool for automated classification of unknown eukaryotic transposable elements. *Bioinformatics* 25, 10 (2009), 1329–1330.
[2] Victoria P Belancio, Dale J Hedges, and Prescott Deininger. 2008. Mammalian non-LTR retrotransposons: for better or worse, in sickness and in health. *Genome research* 18, 3 (2008), 343–358.
[3] Jeffrey L Bennetzen. 2000. Transposable element contributions to plant gene and genome evolution. *Plant molecular biology* 42, 1 (2000), 251–269.
[4] Paola Bonizzoni, Gianluca Della Vedova, Yuri Pirola, Marco Previtali, and Raffaella Rizzi. 2021. Computing the multi-string BWT and LCP array in external memory. *Theoretical Computer Science* 862 (2021), 42–58.
[5] Michael Burrows and David Wheeler. 1994. A block-sorting lossless data compression algorithm. In *Digital SRC Research Report*. Citeseer.
[6] Jinfeng Chen, Travis R Wrightsman, Susan R Wessler, and Jason E Stajich. 2017. RelocaTE2: a high resolution transposable element insertion site mapping tool for population resequencing. *PeerJ* 5 (2017), e2942.

Strain	Sample	Frontier Candidates	True Frontier	Unique Context	Merged Positions	Novel TEis	LINE/L1	SINE/Alu	SINE/B2	LTR/ERV
A/J	f321	51526031	3228795	1168218	181912	2200	671 (42)	249 (2)	480 (41)	800 (259)
129S1/SvlmJ	m157	46433170	2955226	1055816	165809	2196	621 (40)	265 (2)	519 (35)	791 (242)
NOD/ShiLtJ	m146	52153639	3871542	1456247	216913	2938	1030 (104)	393 (6)	680 (81)	835 (274)
NZO/HlLtJ	m146	51263358	3187737	1147893	179681	2240	730 (40)	227 (1)	519 (28)	764 (229)
CAST/EiJ	m090	44730127	2478107	521986	92515	4240	1419 (74)	595 (4)	922 (53)	1304 (203)
PWK/PhJ	f6114	44633988	2477698	516958	92328	4683	1640 (73)	614 (6)	1055 (64)	1374 (216)
WSB/EiJ	f111	36597602	2089009	639532	107628	2175	647 (33)	208 (0)	443 (20)	877 (174)

Table 4: Results from running the full Frontier pipeline on 7 common mouse strains. This table shows the number of novel TEis found in each sample, which is further broken down according to their predicted TE type. The numbers in parentheses indicate TE insertions detected from both sides (i.e. the unique genomic context both before and after a single TE insertion). We observed more novel TEis in CAST and PWK as compared to the other samples even though there were fewer merged positions that match the reference for TEis in these strains. The large number of merged positions in A/J 129S1, NOD, and NZO in contrast to the relatively smaller number of novel TEis suggests that these common laboratory strains share more TEis in common with the mouse reference genome.

[7] Jian-Min Chen, Peter D Stenson, David N Cooper, and Claude Férec. 2005. A systematic analysis of LINE-1 endonuclease-dependent retrotranspositional events causing human genetic disease. *Human genetics* 117, 5 (2005), 411–427.

[8] Chong Chu, Rasmus Nielsen, and Yufeng Wu. 2016. REPdenovo: inferring de novo repeat motifs from short sequence reads. *PloS one* 11, 3 (2016), e0150719.

[9] Murilo Horacio Pereira da Cruz, Douglas Silva Domingues, Priscila Tiemi Maeda Saito, Alexandre R Paschoal, and Pedro Henrique Bugatti. 2020. TERL: classification of transposable elements by convolutional neural networks. *bioRxiv* (2020).

[10] Peter Ebert, Peter A Audano, Qihui Zhu, Bernardo Rodriguez-Martin, David Porubsky, Marc Jan Bonder, Arvis Sulovari, Jana Ebler, Weichen Zhou, Rebecca Serra Mari, et al. 2021. Haplotype-resolved diverse human genomes and integrated analysis of structural variation. *Science* 372, 6537 (2021).

[11] Lavinia Egidi, Felipe A Louza, Giovanni Manzini, and Guilherme P Telles. 2019. External memory BWT and LCP computation for sequence collections with applications. *Algorithms for Molecular Biology* 14, 1 (2019), 1–15.

[12] Paolo Ferragina, Giovanni Manzini, Veli Mäkinen, and Gonzalo Navarro. 2004. An alphabet-friendly FM-index. In *International Symposium on String Processing and Information Retrieval*. Springer, 150–160.

[13] Eugene J Gardner, Vincent K Lam, Daniel N Harris, Nelson T Chuang, Emma C Scott, William S Pittard, Ryan E Mills, Scott E Devine, 1000 Genomes Project Consortium, et al. 2017. The Mobile Element Locator Tool (MELT): population-scale mobile element discovery and biology. *Genome research* (2017), gr–218032.

[14] Claire Hoede, Sandie Arnoux, Mark Moisset, Timothee Chaumier, Olivier Inizan, Veronique Jamilloux, and Hadi Quesneville. 2014. PASTEC: an automatic transposable element classification tool. *PloS one* 9, 5 (2014), e91929.

[15] James Holt and Leonard McMillan. 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*. ACM, 464–471.

[16] Chuan Jiang, Chao Chen, Ziyue Huang, Renyi Liu, and Jerome Verdier. 2015. ITIS, a bioinformatics tool for accurate identification of transposon insertion sites using next-generation sequencing data. *BMC bioinformatics* 16, 1 (2015), 72.

[17] Toru Kasai, Gunho Lee, Hiroki Arimura, Setsuo Arikawa, and Kunsoo Park. 2001. Linear-time longest-common-prefix computation in suffix arrays and its applications. In *Annual Symposium on Combinatorial Pattern Matching*. Springer, 181–192.

[18] Anwica Kashfeen, Harper B Fauni, Timothy A Bell, Fernando Pardo-Manuel de Villena, and Leonard McMillan. 2019. ELITE: Efficiently Locating Insertions of Transposable Elements. In *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*. 183–189.

[19] Thomas M Keane, Kim Wong, and David J Adams. 2012. RetroSeq: transposable element discovery from next-generation sequencing data. *Bioinformatics* 29, 3 (2012), 389–390.

[20] Philipp Koch, Matthias Platzer, and Bryan R Downie. 2014. RepARK—de novo creation of repeat libraries from whole-genome NGS reads. *Nucleic acids research* 42, 9 (2014), e80–e80.

[21] Eric S Lander, Lauren M Linton, Bruce Birren, Chad Nusbaum, Michael C Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William FitzHugh, et al. 2001. 2001. Initial sequencing and analysis of the human genome. *Nature* 409, 6822 (2001), 860–921.

[22] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. 2009. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome biology* 10, 3 (2009), R25.

[23] H Li, B Handsaker, A Wysoker, T Fennell, J Ruan, N Homer, G Marth, G Abecasis, and R Durbin. [n.d.]. 692 (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 25, 16 ([n. d.]), 2078–693.

[24] Guillaume Marçais and Carl Kingsford. 2011. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* 27, 6 (2011), 764–770.

[25] Barbara McClintock. 1984. The significance of responses of the genome to challenge. (1984).

[26] Aleksandr Morgulis, E Michael Gertz, Alejandro A Schäffer, and Richa Agarwala. 2006. A fast and symmetric DUST implementation to mask low-complexity DNA sequences. *Journal of Computational Biology* 13, 5 (2006), 1028–1040.

[27] Koji Muratani, Toshikazu Hada, Yoshihiro Yamamoto, Tadashi Kaneko, Yoshihisa Shigeto, Toru Ohue, Junichi Furuyama, and Kazuya Higashino. 1991. Inactivation of the cholinesterase gene by Alu insertion: possible mechanism for human gene transposition. *Proceedings of the National Academy of Sciences* 88, 24 (1991), 11315–11319.

[28] Christoffer Nellåker, Thomas M Keane, Binnaz Yalcin, Kim Wong, Avigail Agam, T Grant Belgard, Jonathan Flint, David J Adams, Wayne N Frankel, and Chris P Ponting. 2012. The genomic landscape shaped by selection on transposable elements across 18 mouse strains. *Genome biology* 13, 6 (2012), 1–21.

[29] Arian FA Smit, Robert Hubley, and P Green. 1996. RepeatMasker.

[30] Tim Stuart, Steven R Eichten, Jonathan Cahn, Yuliya V Karpievitch, Justin O Borevitz, and Ryan Lister. 2016. Population scale mapping of transposable element diversity reveals links to gene regulation and epigenomic variation. *elife* 5 (2016), e20777.

[31] Aurélie Teissandier, Nicolas Servant, Emmanuel Barillot, and Deborah Bourc'his. 2019. Tools and best practices for retrotransposon analysis using high-throughput sequencing data. *Mobile DNA* 10, 1 (2019), 1–12.

[32] Todd J Treangen and Steven L Salzberg. 2012. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics* 13, 1 (2012), 36–46.

[33] Jasmina Uzunović, Emily B Josephs, John R Stinchcombe, and Stephen I Wright. 2019. Transposable elements are important contributors to standing variation in gene expression in Capsella grandiflora. *Molecular biology and evolution* 36, 8 (2019), 1734–1745.

[34] José AJM van den Hurk, Dorien JR van de Pol, Bernd Wissinger, Marc A van Driel, Lies H Hoefsloot, Ilse J de Wijs, L Ingeborgh van den Born, John R Heckenlively, Han G Brunner, Eberhart Zrenner, 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 (2003), 268–275.

[35] José Luis Villanueva-Cañas, Gabriel E Rech, Maria Angeles Rodriguez de Cara, and Josefa Gonzalez. 2017. Beyond SNPs: how to detect selection on transposable element insertions. *Methods in Ecology and Evolution* 8, 6 (2017), 728–737.

[36] Margaret R Wallace, Lone B Andersen, Ann M Saulino, Paula E Gregory, Thomas W Glover, and Francis S Collins. 1991. A de novo Alu insertion results in neurofibromatosis type 1. *Nature* 353, 6347 (1991), 864.

[37] Robert H Waterston and Lior Pachter. 2002. Initial sequencing and comparative analysis of the mouse genome. *Nature* 420, 6915 (2002), 520–562.

[38] Jiali Zhuang, Jie Wang, William Theurkauf, and Zhiping Weng. 2014. TEMP: a computational method for analyzing transposable element polymorphism in populations. *Nucleic acids research* 42, 11 (2014), 6826–6838.