Detecting genomic deletions from high-throughput sequence data with unsupervised learning

Xin Li[1, *] **and Yufeng Wu**[1]

[1]University of Connecticut, Department of Computer Science and Engineering, Storrs, 06269, USA
[*]xin.li@uconn.edu

ABSTRACT

Structural variation (SV), which ranges from 50 bp to ∼3 Mb in size, is an important type of genetic variations. Deletion is a type of SV in which a part of a chromosome or a sequence of DNA is lost during DNA replication. Three types of signals, including discordant read-pairs, reads depth and split reads, are commonly used for SV detection from high-throughput sequence data. Many tools have been developed for detecting SVs by using one or multiple of these signals. In this paper, we develop a new method called EigenDel for detecting genomic deletions. EigenDel first takes advantage of discordant read-pairs and clipped reads to get initial deletion candidates, and then it clusters similar candidates by using unsupervised learning methods. After that, EigenDel uses a carefully designed approach for calling true deletions from each cluster. We conduct various experiments to evaluate the performance of EigenDel on low coverage sequence data from 1000 Genomes Project. Our results show that EigenDel outperforms other major methods in terms of improving capability of balancing accuracy and sensitivity as well as reducing bias. EigenDel can be downloaded from https://github.com/lxwgcool/EigenDel.

Introduction

The differences in genetic compositions, which are relatively large in size (∼3 Mb or more) and mainly rare changes in the quantity and structure of chromosomes, are defined as microscopic structural variations[1]. With the development of molecular biology and DNA sequencing technology, smaller and more abundant alterations were observed. We define these variants, which range from ∼1 kb to 3 Mb in size, as submicroscopic structural variations[1]. Recently, they have widened to include much smaller events (for example, those >50 bp in length)[2]. The potential contribution of submicroscopic structural variants to human genetic variation and disease might be higher than that of microscopic variants, as they seem to occur at a higher frequency[1]. Deletion is a type of SVs in which a part of a chromosome is lost during DNA replication[3]. Small indels are the most common type of SVs[4]. Deletions may have significant phenotypic influence. Specifically, among genetic disorders annotated in some disease database, such as DECIPHER[5], 80% are caused by deletions[6].

In this paper, we focus on detecting germline deletions in submicroscopic structural variation from pair-end next-generation sequencing (NGS) reads. Germline mutation is the variation within germ cells, which can be passed on to offspring[7]. Read-pairs are the most common form of NGS data. Traditionally, three types of alignment-based signals are used for calling deletions, including discordant read-pairs, reads depth, and split reads[2]. Discordant read-pairs are the read-pairs that the mapped positions and/or orientation of two ends of pairs are inconsistent with reference genome. The read-pairs mapping too far apart are associated with deletions[2]. Read-depth-based approaches assume a random distribution in mapping depth and investigate the divergence from this distribution to highlight duplications and deletions. The deleted regions show reduced read depth when compared to wild-type regions[2]. Split reads are single reads that are mapped to the reference genome discontinuously as two or more segments[8]. The presence of a SV breakpoint is investigated on the basis of a split sequence-read signature breaking the alignment to the reference. A gap in the read is a marker of a deletion[2]. There are some limitations of those three signals. Discordant read-pairs may uncover structural variants but only give inexact positions of breakpoints. Split-read-based methods are inefficient in terms of time and memory, and they can have both high false positive and false negative rates. Read depths are not able to identify smaller events and poor at localizing breakpoints[2]. Moreover, de novo assembly is another common method in bioinformatics[9], which has also been used for detecting SVs. In principle, it allows for the detection of all forms of SVs. However, the application of this approach is still challenging due to the limited length of NGS reads[10].

Many methods have been developed for SV detection by using one or multiple signals mentioned above. CNVnator[11] uses reads depth to detect copy number variation. Pindel[12] is a split reads mapping-based tool, which uses an algorithm called pattern growth to report deletions with micro-insertions. Delly[13] uses split reads alignments to define the exact positions of SV breakpoints by aligning the split reads across the two regions linked by the discordant clusters, which are identified by discordant read-pairs. Lumpy[14] integrates multiple SV signals and uses different reads mappers to generate the alignments of normally aligned pair-end reads, split reads and discordant read-pairs respectively for SV detection. Machine learning,

including supervised learning and unsupervised learning, is widely used in many research fields in recent decades, such as RNA/DNA analysis[15]. Some tools, such as forestSV[16], extract the features from alignment signals and apply supervised learning method to find structural variations. Although many approaches have been developed for SV detection, there is no single method that outperforms others in every aspect for all types of dataset, especially in terms of balancing accuracy and sensitivity. In addition, for supervised-learning-based methods, since the benchmark repositories do not contain every SV for all individuals, the training data may contain many noises, which can significantly reduce the accuracy of prediction.

In this paper, we introduce a new method called EigenDel to call deletions in germline samples. EigenDel first takes advantage of discordant read-pairs and clipped reads to find initial deletion candidates, and then it clusters the similar candidates by using unsupervised learning methods based on reads depth. After that, EigenDel uses a carefully designed approach for detecting true deletions from each cluster. There are two major advantages of applying unsupervised-learning-based methods. First of all, unsupervised learning can discover hidden signals within dataset, and these hidden signals are significant for calling SV deletions. Secondly, unsupervised learning works without labeling training data, which is more adaptable than supervised learning. We compare EigenDel with other 5 widely used tools in terms of the capability of balancing accuracy and sensitivity. The results show EigenDel outperforms these existing methods.

Methods

High-level approach

EigenDel works with mapped sequence reads. Three statistic values, including average depth ($Depth_{avg}$), average insert size (Avg_{IS}), and standard deviation of insert size (STD_{IS}) are calculated at the beginning. After that, EigenDel processes each chromosome separately to call deletions. For each chromosome, EigenDel extracts discordant read-pairs and clipped reads from mapped reads. Then, the initial deletion candidates are determined by grouping nearby discordant read-pairs. Clipped reads are used to produce more accurate estimates of the left and right breakpoints of each deletion candidate. Since the depth of deletion regions should be significantly lower than wild-type regions, candidates with depth larger than average are discarded. Then, for the remaining candidates, EigenDel gets a number of features based on depth for each of them and applies unsupervised learning to classify these candidates into four clusters. Finally, EigenDel marks these clusters as good or bad and applies different strategies to keep true deletions from each cluster respectively. A good cluster means the majority candidates in this cluster are likely to be true deletions, while a bad cluster means the majority candidates are likely to be false. The details are illustrated in Figure 1.

EigenDel

Our new EigenDel method contains four parts: (a) collecting border-clipped reads and discordant read-pairs, (b) identifying deletion candidates, (c) extracting features from candidates, and (d) detecting true deletions with unsupervised learning. EigenDel first finds border-clipped reads and discordant read-pairs from different chromosomes respectively. Then, discordant read-pairs are used to locate deletion candidates, and the breakpoints of each deletion are adjusted by border-clipped reads. Next, the candidates are filtered by depth. This gives a shortened list of deletion candidates. After that, EigenDel collects multiple features related to depth for each remaining candidate and classifies them into four clusters by unsupervised learning. Each cluster is marked as either good or bad. The true deletions are collected by applying different strategies in each cluster.

Collecting border-clipped reads and discordant read-pairs

Bam file that contains alignment information of read-pairs is required by EigenDel. EigenDel uses Picard[17] to get Avg_{IS} and STD_{IS} from BAM file. Samtools[18] is used to calculate $Depth_{avg}$. Some reads are filtered right away, including unmapped reads, polymerase chain reaction (PCR) duplicate reads, reads with low quality, and non-primary alignment reads.

Since a deletion breaks the mapping relationship between reads and reference, two types of reads, including border-clipped reads and discordant read-pairs, are collected. Border-clipped reads are the reads clipped from either tail or head, and we call them as tail-clipped reads and head-clipped reads, which are considered to support the left and right breakpoints of a deletion respectively. Since the clipped part is expected to be from the other side of a deletion, we filter the border-clipped reads, whose clipped part is shorter than 15 bp. Discordant read-pairs that satisfy $Len_{IS} > Avg_{IS} + 3 * STD_{IS}$ are collected and used to locate the deletion candidates because the deletion event would enlarge the insert size of pair-end reads. Note that, since we only consider the deletion in submicroscopic structure variation, the discordant read-pairs with too large insert size are discarded. Usually, deletions do not cross different chromosomes. Therefore, we collect border-clipped reads and discordant read-pairs and identify deletion candidates for each chromosome separately.

Identifying deletion candidates

EigenDel first sorts all discordant read pairs based on the position of left mates. Then it groups nearby discordant read-pairs based on the positions of their left mates to get the range of deletion candidates. Two discordant read-pairs are grouped together

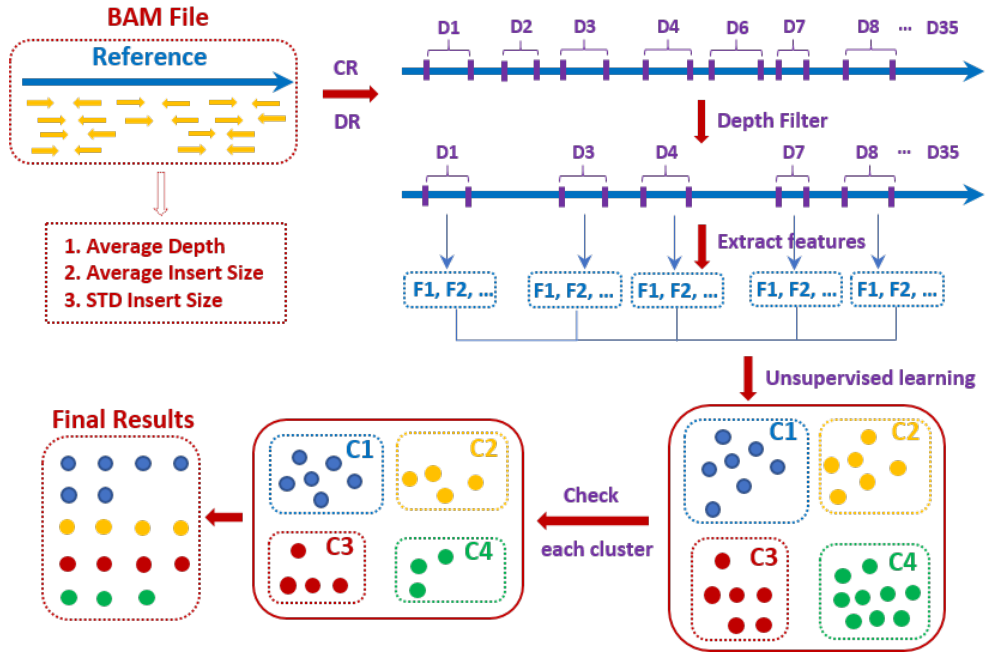

Figure 1. High-level approach. EigenDel takes mapped sequence reads (in the BAM format). Clipped reads (CR) and discordant reads (DR) are used to obtain deletion candidates (total 35 candidates in the figure, denoted as D1 to D35). Then, some candidates, such as D2 and D6, are discarded by the depth filter. There are 29 deletion candidates left, and EigenDel extracts features (F1, F2, . . .) for each of them. All remaining deletions are classified into four clusters named C1 to C4 through unsupervised learning. There are 7, 6, 6 and 9 deletions in clusters C1 (blue), C2 (yellow), C3 (red) and C4 (green) respectively. Finally, false deletions are removed from each cluster. There are 17 deletions left: 6 in C1, 4 in C2, 4 in C3 and 3 in C4. These 17 deletions are called as true deletions.

if the distance between their left mates is shorter than reads (e.g., 101 bp). Once all discordant read-pairs are grouped, each group represents a deletion candidate site. EigenDel discards candidate sites that are supported by only one discordant read-pair. The left and right boundary of each site come from the smallest mapping position of left mates and the largest position of right mates plus its alignment length respectively. Two candidate sites are merged if their boundaries are overlapped, and boundaries of the new merged site are updated. Then, EigenDel discards candidate sites that have no border-clipped reads. For each remaining site, the left breakpoint of deletion candidate comes from the largest mapping position of left mates plus its alignment length, while the right breakpoint is determined by the smallest mapping position of right mates. This roughly locates deletion candidate on the reference genome.

After that, border-clipped reads that satisfy the situations below are used to update the left and right breakpoints of deletion candidate in each site. Specifically, tail-cliped reads and head-clipped reads are viewed to contribute to left and right breakpoint respectively. For the left breakpoint, the distance between it and tail-clipped reads should be shorter than Avg_{IS}. If the tail-clipped read is the second mate, its insert size should be close to Avg_{IS}, and the mapping position of its first mate should be close to the left boundary of current site. If the tail-clipped read is the first mate, the mapping position of its second mate should be near the right boundary of current site. Once all qualified tail-clipped reads are collected, EigenDel only consider the best clipped positions that are supported by the largest number of tail-clipped reads. Multiple best clipped positions may be obtained, and the largest one is used to update the left breakpoint. Note we do not update it if the best clipped positions are only supported by one tail-clipped reads. There are four major differences during the updating of right breakpoint. First, the position of head-clipped reads should be near the right breakpoint. Second, if the head-clipped read is the second mate, the mapping position of its first mate should be near the left boundary of current site. If the head-clipped read is the first mate, its insert size should be around Avg_{IS}, and the mapping position of its second mate should be close to the right boundary of current site. Third, the smallest best clipped positions supported by the largest number of head-clipped reads are selected to adjust the right breakpoint. Figure 2 shows the details.

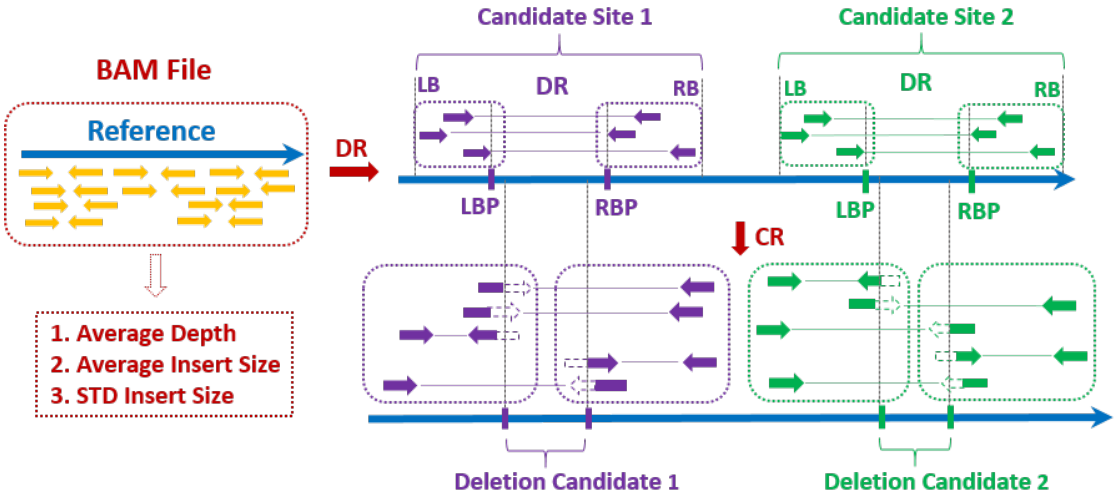

Figure 2. Identifying deletion candidates. Discordant read-pairs (DR) and border-clipped reads (CR) are collected from BAM file. Two deletion candidate sites, including candidate site 1 (purple) and candidate site 2 (green), are identified by DRs. Each site contains 3 DRs. Left boundary (LB) and right boundary (RB) are used to present the range of site. Left breakpoint (LBP) and right breakpoint (RBP) are used to describe the deletion candidate in current site. 5 CRs are contained by site 1, which are used to adjust LBP and RBP. Deletion Candidate 1 refers to the potential deletion in site 1. Site 2 contains 5 CRs, and its potential deletion is shown as Deletion Candidate 2.

Extracting features from candidates

We calculate average depth for each deletion candidate in the region between left and right breakpoints. Since a deletion may lead to significantly lower reads depth than wild-type region, the candidates with depth larger than $Depth_{avg}$ are discarded. EigenDel is designed for detecting deletions in germline samples, and it assumes that the organism under study is diploid. That is, EigenDel does not consider the situation where ploidy can change (in, e.g. tumor samples). For diploid organism, there are two types of deletions, including homozygous and hemizygous deletions. Hemizygous deletion refers to the loss of one allele, whereas homozygous (biallelic) deletion refers to the loss of both alleles identified by allele-specific analysis in clinical samples[19]. For homozygous deletions, the deletions occur in both copies. Thus, ideally, there is no reads within the deletion, and the depth should be equal to 0. For hemizygous deletion, since it is single copy deletion, the depth should be roughly equal to 50% of $Depth_{avg}$. In practice, however, situations is less clear cut. In order to allow mapping errors and inaccurate positions of breakpoints, we identify 4 coverage ranges, namely T_0 , T_1 , T_2 and T_3, as shown in Table 1, to describe the internal structure of each deletion candidate.

Coverage	Range
T_0	0
T_1	$[0, Ceil(Depth_{avg} * 0.25))$
T_2	$[Ceil(Depth_{avg} * 0.25), Ceil(Depth_{avg} * 0.5)]$
T_3	$(Ceil(Depth_{avg} * 0.5), Ceil(Depth_{avg})]$

Table 1. Coverage ranges for feature collection

T_0 refers to the perfect case of homozygous deletions (i.e., read depth is 0). T_1 refers to the case of homozygous deletions allowing reads mapping errors and inaccurate boundaries. T_2 refers to the case of hemizygous deletions with the same tolerance as T_1. T_3 refers to the range that contains both true and false deletions. We use (D_0, L_0), (D_1, L_1), (D_2, L_2) and (D_3, L_3) to present the internal structure of each deletion candidate. L_i stands for the total length of all positions that fall into T_i (may be non-consecutive), and D_i is the average depth of the range of L_i. Then, we use the length of current deletion, the distance between left and right breakpoints, to normalize L_i. We record the normalized result as LN_i. Therefore, $LN_i (i = 0, 1, 2, 3)$ are used as 4 independent features to present each deletion candidates. Figure 3 illustrates the approach.

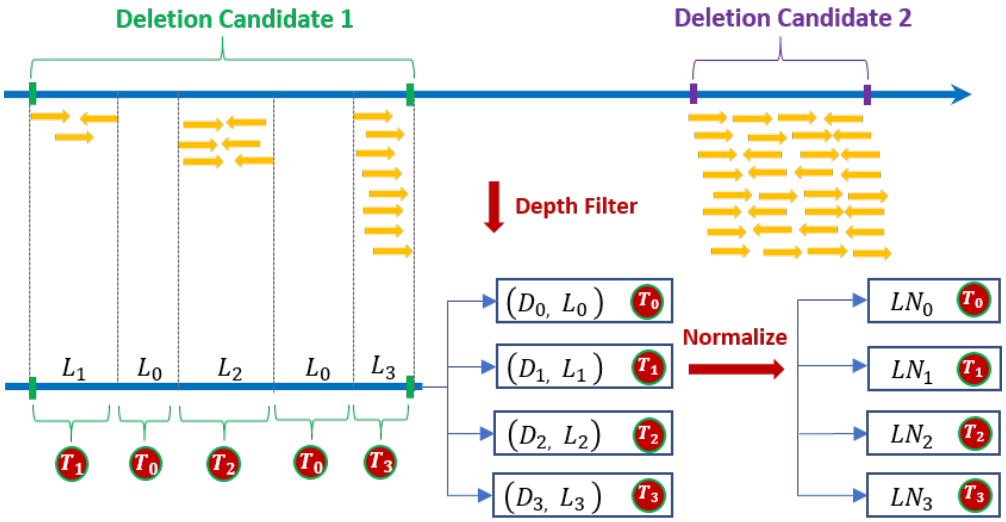

Figure 3. Feature extractions from deletion candidates. Two deletion candidates are identified by discordant reads. "Deletion Candidate 2" is discarded after depth filter because its depth is larger than $Depth_{avg}$. For "Deletion Candidate 1", 5 ranges are identified by T_i. L_i and D_i are the total length and the average depth of the range defined by T_i respectively. Each L_i is normalized by the length of "Deletion Candidate 1", and the normalized results are recorded by LN_i. Therefore, the internal structure of "Deletion Candidate 1" is presented by $LN_i (i = 0, 1, 2, 3)$.

Detecting true deletions with unsupervised learning

So far, EigenDel collects a list of deletion candidates that are identified by discordant reads, and then the candidates are refined by clipped reads. After that, some candidates are filtered by depth filter. However there are still many false positives. For example, some false deletions may appear in the coverage range T_3, which is from 50% $Depth_{avg}$ to $Depth_{avg}$. In addition, since the real data is noisy, it is challenging to handle some abnormal alignment situations (e.g. reads mapping error), which may change the real depth of candidates. Moreover, inaccurate breakpoints may bring the irrelevant range into deletion candidates. This may shrink the depth difference among homozygous deletion, hemizygous deletion and wild-type range. Therefore, using simple thresholds alone is not be able to filter many false positives.

Supervised learning approach is one choice to solve this issue. By taking advantage of benchmark callset, the deletions can be labeled and used to train a model to predict true deletions for testing samples. However, since the public datasets do not contain all structure variations for each individual, it is hard to label the true negative deletions for training. This leads to the failure of correct prediction of true deletions by the model. Moreover, some species may not have benchmark structural variants dataset, which prevents us from labeling training data.

In order to call true deletions from noisy candidates, EigenDel applies unsupervised learning. The key idea is that different types of deletion candidates tend to cluster due to shared features. That is, the same types of true (homozygous or hemizygous) deletions tend to be similar in features (e.g., depth profile within the deletions). Similarly, same types of false positives may share some similar internal structure patterns based on reads depth. Thus, it is possible to use unsupervised learning to separate different types of deletions into different clusters. Moreover, since unsupervised learning does not need labeled samples for training, it is more flexible than supervised learning, especially for the species without good benchmark dataset.

Based on the features described in the previous step, EigenDel uses two steps to perform unsupervised learning. It first applies principle component analysis (PCA), followed by hierarchical clustering[20]. Scikit-learn[21] is used in our implementation. Since true deletions should be either homozygous or hemizygous, two dimensions could express all different types of true deletions. Thus, we apply PCA to all candidates and choose the top two principle components to represent each deletion. This is also good for visualization. Then, all deletion candidates are classified into four clusters based on their top two principle components through hierarchical clustering. Those clusters are expected to present perfect homozygous deletions, homozygous deletions with error tolerance, hemizygous deletions with error tolerance, and the mix of heterozygous deletions and wild-type. Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all samples. The leaves are the clusters with only one sample[21]. We use agglomerative clustering object provided by Scikit-learn Python package, which performs a hierarchical clustering using a bottom-up approach: each candidate starts in its own cluster, and clusters are successively merged. There are several advantages of hierarchical clustering. First, it does not

need to select the initial node. Second, hierarchical clustering shows the relationship among the samples in a cluster. Third, it is not sensitive to the shape of cluster (e.g. k-means prefers spherical cluster), which makes it adaptable for different dataset. The Euclidean metric and ward (sum of squares of deviations) are used to implement hierarchical clustering.

Once four clusters are generated, they are marked as either good or bad. A good cluster means the majority of candidates in this cluster are true deletions, while the bad cluster means the majority of deletions in this cluster are false. Here is the definition of good and bad cluster. First, for a true deletion, ideally, $\sum_{i=0}^{2} L_i$ should be equal to the whole length of deletion. In another words, $\sum_{i=0}^{2} LN_i$ should close to 1. Considering the influence of reads mapping error and inaccurate breakpoints, we define a true deletion should have $\sum_{i=0}^{2} LN_i \geq 0.7$. Suppose there are N deletion candidates in one cluster, we collect three values, including LN_0, LN_1 and LN_2, for each of them. After that, all deletions in the current cluster are sorted by three rounds based on $LN_i (i = 0, 1, 2)$ respectively. We record the sorted result in each round, and store them as SR_0, SR_1 and SR_2. As a result, each SR_i contains all N deletions in the current cluster, which are sorted by LN_i from small to large. Then, we calculate three statistic values for each SR_i, including average of LN_i (Avg_{LN_i}), standard deviation of LN_i (STD_{LN_i}) and average of top half deletions with the highest LN_i ($THAvg_{LN_i}$). If $\sum_{i=0}^{2} THAvg_{LN_i} \geq 0.7$, we define this cluster as a good cluster, otherwise it is bad.

Once a cluster is marked as either good or bad, we use LN_i, which is associated with the largest $THAvg_{LN_i}$, as the principle feature of current cluster to find the true deletions. We assume the distribution of LN_i follows empirical rule. Therefore, the majority of deletion candidates should be in the range $[Avg_{LN_i} - STD_{LN_i}, Avg_{LN_i} + STD_{LN_i}]$, since $Pr(\mu - 1\sigma \leq X \leq \mu + 1\sigma) \approx 0.6827$. Two thresholds, including T_{low} and T_{high}, are defined as shown in Table 2.

Threshold	Value
T_{low}	$Avg_{LN_i} - STD_{LN_i}$
T_{high}	$Avg_{LN_i} + STD_{LN_i}$

Table 2. Thresholds of detecting true deletions

For a good cluster, the deletions are discarded if $LN_i < T_{low}$ and $\sum_{j=0}^{2} LN_i (j \neq i) < T_{low}$. For a bad cluster, the deletions are kept if $LN_i > T_{high}$ or $\sum_{j=0}^{2} LN_i (j \neq i) > T_{high}$. Finally, all remaining deletions in each cluster are called as true deletions. The details are shown in Figure 4.

Results

We use 1000 Genome Project[22] Phase3 dataset as the benchmark. Only the deletions in the Phase3 callset of the 1000 Genomes Project are viewed as true deletions. Five existing deletion detecting tools are used for comparision with EigenDel. These include Pindel, CNVnator, GASVpro[23], Delly and Lumpy. We directly use BAM files provided by 1000 Genome Project Phase3 dataset to run the experiments. For some tools that require separate reads files as input, such as Lumpy, we dump reads from BAM files. In addition, since high coverage datasets usually have big size and high cost, consume large computing resources and take long time for analyzing, we use low coverage datasets in all experiments.

The main purpose of the comparison is evaluating the performance of balancing accuracy and sensitivity by various tools. We first calculate accuracy and sensitivity of each tool. After that, F1 score is used as the main statistic for comparison. Traditionally, the F1 score is defined as $2 \times \frac{Precision \times Recall}{Precision + Recall}$. In our case, since there is no true negative, and all non-true positives are viewed as false positives, the precision and recall are equal to accuracy and sensitivity respectively. So the F1 score is equal to $2 \times \frac{Accuracy \times Sensitivity}{Accuracy + Sensitivity}$[24]. We compare F1 score based on different samples and different chromosomes in one sample respectively. A method with low bias means it can get the highest F1 score in both majority of those samples and majority of chromosomes in one sample. Our results show that EigenDel is the best method to balance accuracy and sensitivity with low bias in all testing cases.

NA12878

The individual NA12878 of the 1000 Genomes Project has been studied by many researchers. We use the low coverage BAM file (20121211) of NA12878 from the 1000 Genomes Project Phase3 dataset in this comparison. The average depth of this BAM file is 5.26. It contains the aligned reads of SRR622461, which contains 92,459,459 pair-end reads. The reads length in this sequence library is 101 bps. There are 1982 deletions of NA12878 reported in the released Phase3 structural variation callset. These deletions are from 23 different chromosomes. We calculate accuracy and sensitivity of each tool, and then get the F1 score of the whole genome and each chromosome respectively.

The results are illustrated in Figure 5 (A), supplemental materials (Table S1) and Figure 6 (C.1 and C.2). Figure 5 (A) shows that EigenDel has the highest F1 score for NA12878. Supplemental materials (Table S1) show that EigenDel has higher F1 score than others in the majority of chromosomes: Pindel (23/23, i.e., EigenDel is better than Pindel on 23 out of 23 chromosomes), CNVnator (23/23), GASVpro (23/23), Delly (16/23) and Lumpy (14/23). Figure 6 (C.1 and C.2) shows an example of the

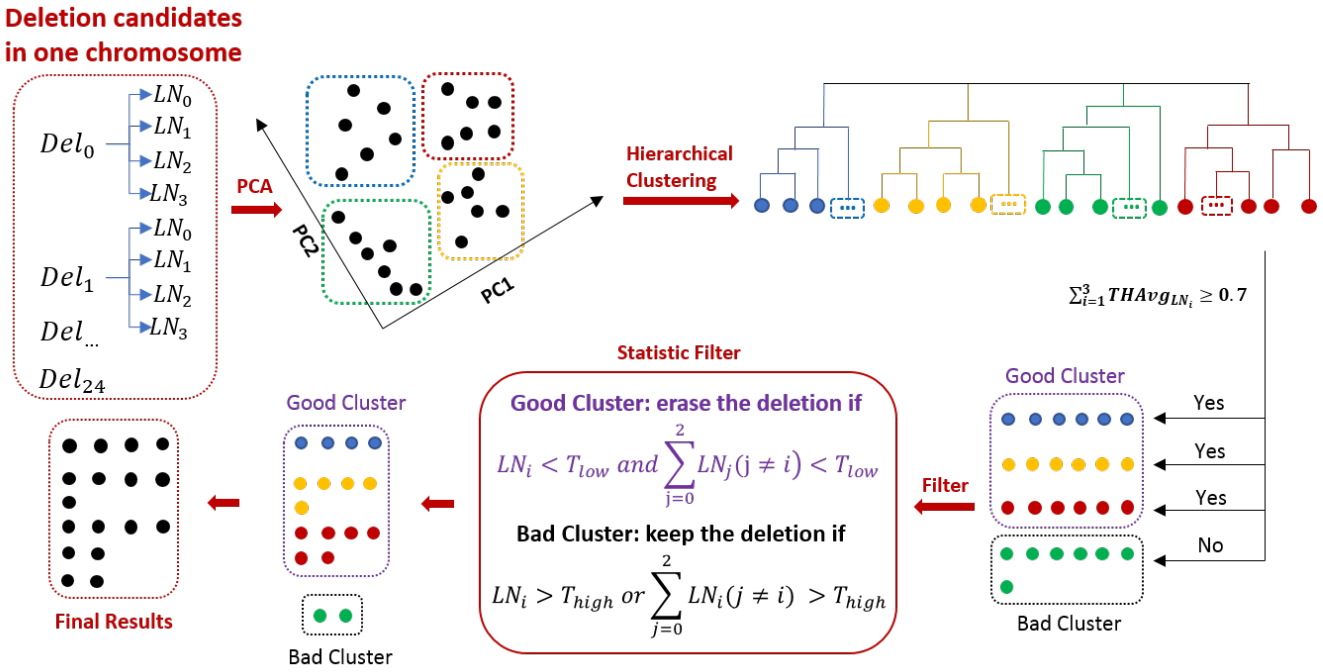

Figure 4. Detecting true deletions with unsupervised learning. 25 deletion candidates from Del_0 to Del_{24} are identified, and each of which contains multiple features. PCA is applied to all candidates, and the top two principle components are used to present each candidate. All candidates are classified into four clusters through hierarchical clustering, including blue (6), yellow (6), red (6) and green (7). After checking $\sum_{i=0}^{2} THAvg_{LN_i}$, three clusters are marked as good, including blue, yellow and red, while green is marked as bad. Then statistic filter is applied to find true deletions. For a good cluster, the deletions are discarded if $LN_i < T_{low}$ and $\sum_{j=0}^{2} LN_i(j \neq i) < T_{low}$. For a bad cluster, the deletions are kept if $LN_i > T_{high}$ or $\sum_{j=0}^{2} LN_i(j \neq i) > T_{high}$. Afterwards, 4, 5, 6, 2 deletions in blue, yellow, red and green groups are remained. These deletions are reported as true deletions.

performance of unsupervised learning for chromosome 1. There are 149 deletion candidates detected in chromosome 1, and 67 of them are presented in the Phase3 callset (i.e., the presumed true deletions). X and Y axes in Figure 6 (C.1 and C.2) come from the top two principle components of PCA. Figure 6 (C.1) shows all deletion candidates found by EigenDel, and the cyan dots stand for the true deletions from the Phase3 callset. Figure 6 (C.2) shows the classification result of hierarchical clustering. Four clusters of deletions are generated, and they are marked in different colors. The majority of false deletions are classified in the blue cluster. The deletions in the same cluster share similar features. For example, there are 35 deletion candidates in green cluster, and the values of LN_0 for all those candidates are $\leq 81\%$. The yellow, green and red clusters are marked as good, while the blue cluster is marked as bad. After the statistic filter is applied for each cluster respectively, 130 deletions are left (19 false deletions are discarded) and 67 of them are presented in the Phase3 callset. This means 23.2% false positives are discarded while no true deletion is lost. This demonstrates that unsupervised learning can cluster deletions with similar features, which helps to filter false positives efficiently for both of the whole genome and most single chromosome of NA12878.

Comparison on five 1000 Genomes individuals

The low coverage BAM files from five 1000 Genomes individuals, including NA12777 (20130415), NA12776 (20130415), NA12878 (20121211), NA12775 (20130415) and NA12763 (20130502), are used in this comparison. The read depth of these five individuals are 9.08, 5.89, 5.26, 9.63 and 7.84 respectively. The BAM files of the four individuals, including NA12777, NA12776, NA12775 and NA12763, contain multiple sequence libraries. NA12777 contains single sequence library. There are 2032, 2115, 1982, 1988 and 2105 deletions in the Phase3 callset for these five individuals respectively.

Figure 5(B) shows that EigenDel has the highest F1 score for the whole genome of all five individuals, and all F1 scores from EigenDel are above 0.5. Others tools, such as Lumpy, have poor performance in some individuals, such as NA12776. Figure 6 shows the examples of clustering results of unsupervised learning from chromosomes 6, 10, 1, 4 and 13 of NA12777, NA12776, NA12878, NA12775 and NA12763 respectively. For chromosome 6 in NA12777 (Figure 6 A.1 and A.2), 140

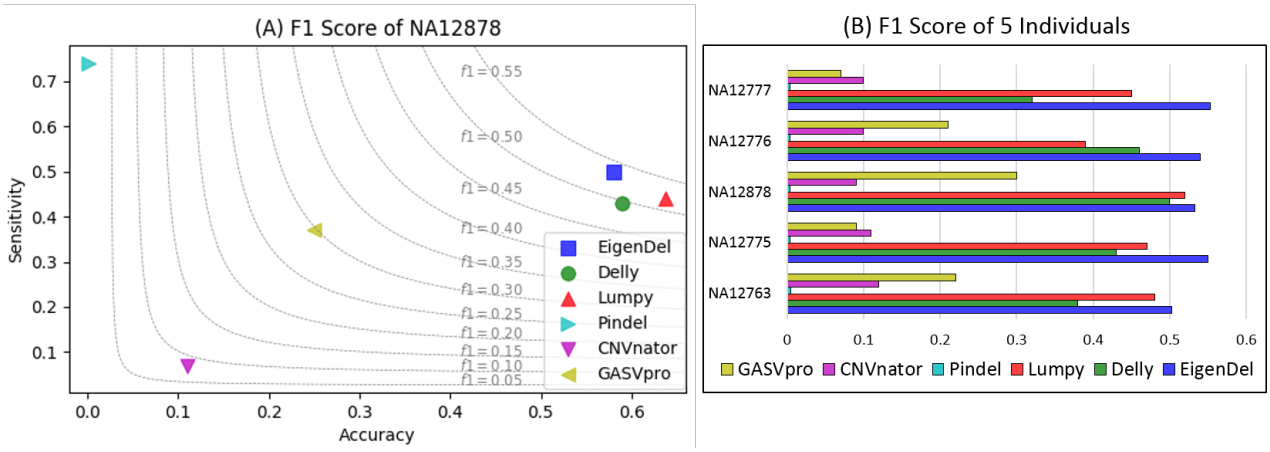

Figure 5. F1 scores. (A) F1 scores of all comparison tools on the whole genome of NA12878. (B) F1 scores of all comparison tools on five 1000 Genomes individuals: NA12777, NA12776, NA12878, NA12775 and NA12763.

deletion candidates are detected and 75 of them are in the Phase3 callset. After the statistic filter is applied to each cluster, 23 false deletions are discarded and 71 true deletions are detected, which means EigenDel discards 35.4% false positives while only loses 5% true deletions. For chromosome 10 in NA12776 (Figure 6 B.1 and B.2), 76 deletion candidates are detected and 43 of them are recorded in the Phase3 callset. After the statistic is applied to for each cluster, 9 false deletions are discarded and 43 true deletions are detected, which means EigenDel discards 27.3% false positives while no true deletion is lost. The case of chromosome 1 in NA12878 has been introduced in previous comparison. For chromosome 4 in NA12775 (Figure 6 D.1 and D.2), 181 deletion candidates are detected and 103 of them are recorded in the Phase3 callset. After applying the statistic filter to each cluster, 32 false deletions are discarded and 97 true deletions are detected, which means EigenDel discards 41% false positives while only loses 5.8% true deletions. For chromosome 13 in NA12763 (Figure 6 E.1 and E.2), 126 deletion candidates are detected and 47 of them are recorded in the Phase3 callset. After applying the statistic filter to each cluster, 50 false deletions are discarded and 46 true deletions are detected, which means EigenDel discards 63.3% false positives while only loses 2% true deletions. All results demonstrate that PCA and hieratical clustering can cluster deletions with similar features together, which helps filter false positives efficiently for different individuals on real data.

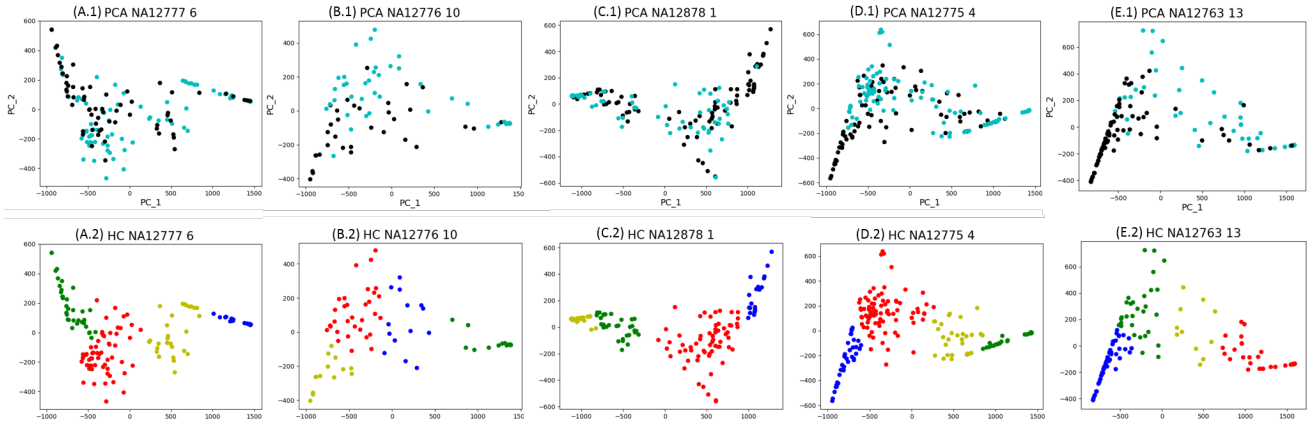

Figure 6. Clustering results with unsupervised learning. (A.1, B.1, C.1, D,1, E.1) The two axes are from the top two principle components of PCA. The dots represent all deletion candidates in chromosome 6, 10, 1, 4 and 13 of NA12777, NA12776, NA12878, NA12775 and NA12763 respectively. The cyan dots stand for the deletion candidates recorded in the 1000 Genomes Project Phase3 callset, which are viewed as true deletions. The black dots refer to the candidates that are not in Phase3 callset, which are viewed as false positives. (A.2, B.2, C.2, D.2, E.2) Classification results of hierarchical clustering on chromosome 6, 10, 1, 4 and 13 of NA12777, NA12776, NA12878, NA12775 and NA12763 respectively. In each scatter plot, four clusters of deletions are classified, which are marked in different colors.

Discussion

EigenDel is designed for detecting deletions based on low coverage data. Since low coverage data contain significant noise, it is challenging to find genomic deletions efficiently based on low coverage data. All low coverage reads used in comparison are from the 1000 Genomes Project. We use multiple individuals, including some widely studied samples, such as NA12878, for comparison. These comparisons are based on both of the whole genome of individuals and each chromosome of single individual. Some BAM files contain single sequence library while others contain multiple libraries. Our results show that EigenDel can handle both types of BAM files, perform better than others for the whole genome of all testing samples, and give higher F1 score for majority of chromosomes of each individual. In addition, unlike Lumpy, EigenDel uses BAM file directly without requiring sequence reads to do alignment, which can reduce the running time.

Some tools, such as Pindel, provide high sensitivity but have a lot of false positives, which leads to low accuracy. Some other tools give better accuracy but lower sensitivity. Thus, how to balance sensitivity and accuracy is a key point of evaluation. By taking advantage of PCA and hierarchical clustering, similar deletions candidates are classified together efficiently, which helps us apply different filters to identify the true deletions in each cluster. The results show that a large number of false positives are filtered while only lose a few true deletions from the clustering results. This gives the highest F1 score among all comparison methods.

EigenDel takes 20 mins to 50 mins on running each testing sample, and this is similar to CNVnator, Delly and GASVpro. Lumpy takes around 1.5 hours on running each single individual while Pindel costs about 5 hours. As a result, the running time of EigenDel is competitive. EigneDel is designed for germline samples of diploid species. It uses discordant read pairs to get raw deletion candidates. Therefore, in principle, all of deletions shorter than $3 * STD_{IS}$ are discarded. The benchmark dataset we use includes all types of deletions with the length from tens to tens of thousands bp. Based on the comparison results, EigenDel performs well even when short deletions in the benchmark dataset are included.

Conclusion

In this paper, we design a method named EigenDel for detecting submicroscopic structural variations deletions in germline samples of diploid organisms. EigenDel uses discordant read pairs to collect deletion candidates, and it uses clipped reads to update the boundary for each of them. The main idea of EigenDel is that it uses unsupervised learning to detect true deletions. For this, EigenDel first applies a read depth filter, and then it extracts four features for remaining candidates based on depth. Unsupervised learning is used to cluster similar deletions together: the top two principle components from PCA are used to present each deletion candidate. Hierarchical clustering is used to classify all candidates into four clusters. Then, EigenDel marks each cluster as either good or bad by using the statistic values calculated from the depth features of all candidates in the same cluster. A good cluster means the majority in the cluster are true deletions while a bad one means the majority candidates are false. EigenDel applies these different statistic filters to both good and bad clusters to extract true deletions.

The deletions from the 1000 Genomes Project Phase 3 callset are used as benchmark. The low coverage BAM files of five different 1000 Genomes individuals are used for comparison. Five existing deletion calling methods are compared with EigenDel, including Pindel, CNVnator, GASVpro, Delly and Lumpy. The results show that EigenDel gives the highest F1 score in the whole genome of all tested samples. For each individual, EigenDel performs better in the majority of chromosomes than other tools. Thus, EigenDel has the best performance in balancing accuracy and sensitivity with low bias.

EigenDel is developed by C++ and could be downloaded from https://github.com/lxwgcool/EigenDel.

References

1. Feuk, L., Carson, A. R. & Scherer, S. W. Structural variation in the human genome. *Nat. Rev. Genet.* **7**, 85 (2006).
2. Alkan, C., Coe, B. P. & Eichler, E. E. Genome structural variation discovery and genotyping. *Nat. Rev. Genet.* **12**, 363 (2011).
3. Lewis, R. *Human Genetics: Concepts and Applications* (McGraw-Hill Higher Education, 2007).
4. Mullaney, J. M., Mills, R. E., Pittard, W. S. & Devine, S. E. Small insertions and deletions (INDELs) in human genomes. *Hum. molecular genetics* **19**, R131–R136 (2010).
5. Firth, H. V. *et al.* DECIPHER: database of chromosomal imbalance and phenotype in humans using ensembl resources. *The Am. J. Hum. Genet.* **84**, 524–533 (2009).
6. Weischenfeldt, J., Symmons, O., Spitz, F. & Korbel, J. O. Phenotypic impact of genomic structural variation: insights from and for human disease. *Nat. Rev. Genet.* **14**, 125 (2013).
7. Griffiths, A., Miller, J., Suzuki, D., Lewontin, R. & Gelbart, W. *An Introduction to Genetic Analysis. 7th edition.* (W H Freeman & Company, 2000).

8. Abel, H. J. & Duncavage, E. J. Detection of structural dna variation from next generation sequencing data: a review of informatic approaches. *Cancer genetics* **206**, 432–440 (2013).
9. Chu, C., Li, X. & Wu, Y. Gappadder: A sensitive approach for closing gaps on draft genomes with short sequence reads. In *2017 IEEE 7th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)*, 1–1 (IEEE, 2017).
10. Tattini, L., D'Aurizio, R. & Magi, A. Detection of genomic structural variants from next-generation sequencing data. *Front. bioengineering biotechnology* **3**, 92 (2015).
11. Abyzov, A., Urban, A. E., Snyder, M. & Gerstein, M. CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome research* **21**, 974–984 (2011).
12. Ye, K., Schulz, M. H., Long, Q., Apweiler, R. & Ning, Z. Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics* **25**, 2865–2871 (2009).
13. Rausch, T. *et al.* DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics* **28**, i333–i339 (2012).
14. Layer, R. M., Chiang, C., Quinlan, A. R. & Hall, I. M. LUMPY: a probabilistic framework for structural variant discovery. *Genome biology* **15**, R84 (2014).
15. Chu, C., Li, X. & Wu, Y. Splicejumper: a classification-based approach for calling splicing junctions from rna-seq data. *BMC bioinformatics* **16**, S10 (2015).
16. Michaelson, J. J. & Sebat, J. ForestSV: structural variant discovery through statistical learning. *Nat. methods* **9**, 819 (2012).
17. Picard toolkit. http://broadinstitute.github.io/picard/ (2019).
18. Li, H. *et al.* The sequence alignment/map format and samtools. *Bioinformatics* **25**, 2078–2079 (2009).
19. Liu, W. *et al.* Homozygous deletions and recurrent amplifications implicate new genes involved in prostate cancer. *Neoplasia* **10**, 897–IN37 (2008).
20. Johnson, S. C. Hierarchical clustering schemes. *Psychometrika* **32**, 241–254 (1967).
21. Pedregosa, F. *et al.* Scikit-learn: Machine learning in python. *J. machine learning research* **12**, 2825–2830 (2011).
22. Siva, N. 1000 genomes project (2008).
23. Sindi, S. S., Önal, S., Peng, L. C., Wu, H.-T. & Raphael, B. J. An integrative probabilistic model for identification of structural variation in sequencing data. *Genome biology* **13**, R22 (2012).
24. Li, X. & Wu, Y. Detecting circular RNA from high-throughput sequence data with de Bruijn graph. *bioRxiv* 509422 (2019).

Author Contributions

X.L. designed algorithms, developed software, performed analysis and experiments, and wrote the paper. Y.W. designed the algorithms, wrote the paper and supervised the project. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Data availability

The software is available under the GPLv3 licence at https://github.com/lxwgcool/EigenDel
The datasets used in this paper are available at 1000 Genomes Project (http://www.internationalgenome.org/data) with identification numbers NA12777, NA12776, NA12878, NA12775 and NA12763.

Additional information

Supplementary information
EigenDel_supplemental_materials.pdf contains two types of information, including F1 scores of each chromosome on NA12878 and the scripts used to run all comparison tools in high performance cluster (HPC).

Funding
This work is supported in part by grants IIS-1526415 and CCF-1718093 from US National Science Foundation to YW.