BMC Bioinformatics

METHODOLOGY Open Access

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

Xin Li[1,2*] and Yufeng Wu[3]

From 16th International Symposium on Bioinformatics Research and Applications
Virtual. 1-4 December 2020

*Correspondence:
xin.li4@nih.gov

[1] Division of Cancer Epidemiology and Genetics, National Cancer Institute, Bethesda, MD 20892, USA
[2] Cancer Genomics Research Laboratory, Frederick National Laboratory for Cancer Research, Leidos Biomedical Research Inc, Frederick, MD 21702, USA
[3] Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269, USA

Abstract

Background: 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.

Results: In this paper, we develop a new method called EigenDel for detecting the germline submicroscopic 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.

Conclusions: 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.

Keywords: Structure variation, Deletion, High-throughput sequencing, Genomics, Unsupervised learning

Background

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].

Traditionally, three types of sequence data based signals are used for deletion detection, including discordant reads pairs, reads depth and split reads [2]. Discordant read pairs are the reads pairs that the mapped positions and/or orientation of the two ends of the pairs are inconsistent with the reference genome. Read pairs that are mapped too far apart may be related to deletions [2]. Read-depth approaches assume a random distribution in mapping depth and investigate the divergence from this distribution to highlight duplications and deletions. Deleted regions may 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 [7]. The presence of the so-called SV breakpoint is used as the basis of a split sequence-read signature. A breakpoint breaks the alignment of a read into multiple segments on the reference. A split-read may indicate the presence of a deletion [2]. There are some limitations for those three signals. Discordant read pairs may uncover structural variants but only give inexact positions of breakpoints. Split-read methods have low time and memory efficiency, and can have both high false positive and false negative rates. Read depths are not able to identify smaller events and much poorer at localizing breakpoints [2]. Moreover, De novo assembly is another common method in bioinformatics [8], which has also been used for finding structure variations. It allows - at least in principle - for the detection of all the forms of structural variations. However, the application of this approach is still challenging due to the limited length of NGS (next-generation sequencing) reads [9]. Many methods have been developed for SV detection by using one or multiple signals mentioned above. Pindel [10] uses an algorithm called pattern growth to report deletions with micro-insertions. Delly [11] 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 [12] integrates multiple SV signals and uses different reads mappers for SV detection. SvABA [13] is a method for detecting structural variants in sequencing data using genome-wide local assembly. Manta [14] is developed and maintained by Illumina, which calls structural variants and indels from mapped paired-end sequencing reads. Machine learning is widely used in many research fields in recent decades. Some tools, such as forestSV [15], extract the features from alignment signals and apply supervised learning method to find SV. Although many approaches have been developed for SV detection, there is no single method that outperforms others, 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 unsupervised-learning-based method called EigenDel to detect germline deletions in submicroscopic SV from pair-end reads for diploid organisms. Since each potential deletion is presented by multiple principal components, which are extracted based on eigenvalue, we name our method as EigenDel. There are two major advantages of applying unsupervised-learning-based methods. First of all, since the BAM file may contain many reads mapping errors, such as repetitive ranges, it is hard to use a single threshold to separate potential deletions (homozygous/hemizygous) and normal (none-SV) ranges. Unsupervised learning can discover hidden signals within dataset, and these hidden signals are significant for calling true deletions from raw candidates. 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 that EigenDel outperforms these existing methods.

Method

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 either 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 Fig. 1.

EigenDel

Collecting border-clipped reads and discordant read-pairs

Bam file that contains alignment information of read-pairs is required by EigenDel. EigenDel uses Picard [16] to get Avg_{IS} and STD_{IS} from BAM file. Samtools [17] 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. For discordant reads, since the general read insert size follows a normal distribution across the genome and the discordant reads come with abnormal

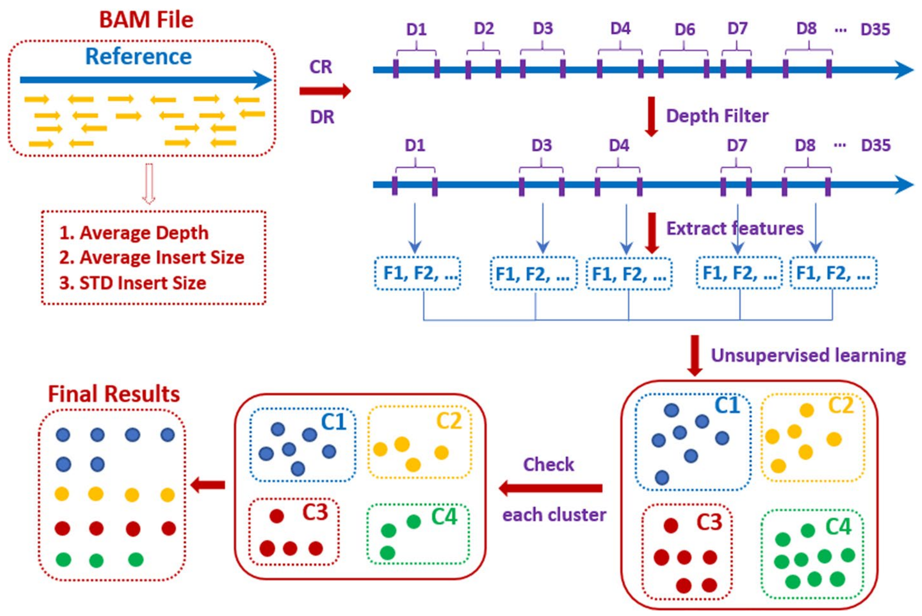

Fig. 1 High-level approach. EigenDel takes BAM file as input. Clipped reads (CR) and discordant reads (DR) are used to obtain deletion candidates (total 35 candidates in the figure, denoted as D1 to D35). Then, some candidates, such as D2 and D6, are discarded by the depth filter. EigenDel extracts features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

insert size, read-pairs that satisfy $Len_{IS} > Avg_{IS} + 3 * STD_{IS}$ are collected as the discordant reads 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. Since deletions are intrachromosomal events, a single deletion never spans different chromosomes. Therefore, we collect border-clipped reads and discordant read-pairs to 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 if the distance between their left mates is shorter than the length of read (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 three 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.

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

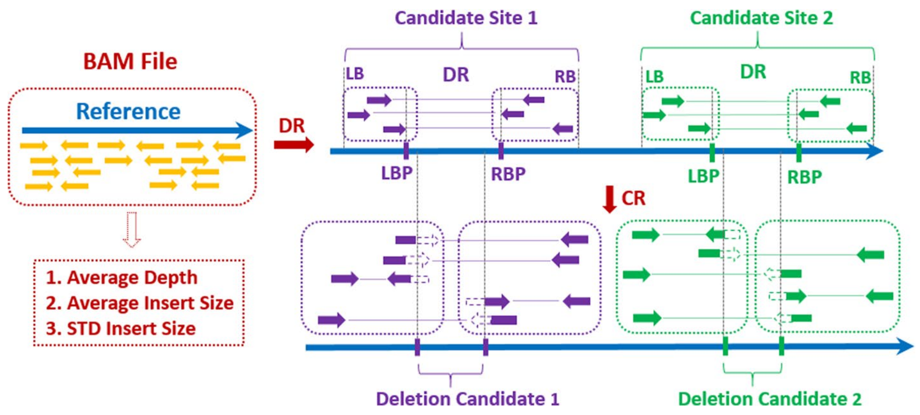

Fig. 2 Identifying deletion candidates. Discordant read-pairs (DR) and border-clipped reads (CR) are collected from BAM file. Two deletion candidate sites, including candidate site 1 (purple) and candidate site 2 (green), are identified by DRs. Each site contains 3 DRs. Left boundary (LB) and right boundary (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

designed for detecting germline deletions in diploid organism. 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 [18]. 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.

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.

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 and repetitive ranges), which may change the real depth of candidates. Moreover, inaccurate breakpoints may bring the normal range into deletion candidates. These may shrink the depth difference among homozygous deletion, hemizygous deletion and normal range. Therefore, using simple thresholds alone is not able to filter many false positives.

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 together due to share features. That is, the same types of true (homozygous or

Table 1 Coverage ranges for feature collection

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	$(Ceil(Depth_{avg} * 0.5), Ceil(Depth_{avg})]$

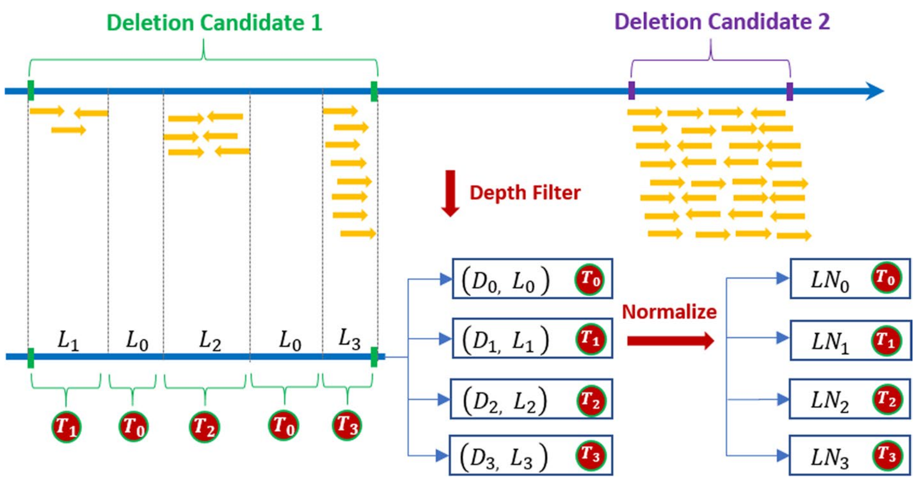

Fig. 3 Feature extractions from deletion candidates. Two deletion candidates are identified by discordant reads. "Deletion Candidate 2" is discarded after depth filter because its depth is larger than $Depth_{avg}$. For "Deletion Candidate 1", 5 ranges are identified by T_i. L_i and D_i are the total 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 $LN_i (i = 0, 1, 2, 3)$

hemizygous) deletions tend to be similar in features (e.g., depth profile within the deletions). Similarly, the 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. In each cluster, since the majority candidates share the similar features, it is more easy and accurate to find true deletions by applying statistical threshold. 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 [19]. 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 4 cases, including perfect homozygous deletions, homozygous deletions with error tolerance, hemizygous deletions with error tolerance, and the mix of heterozygous deletions and normal range. 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 [20]. We use an 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 candidates in a cluster. Third, it is not sensitive to the shape of the cluster (e.g. k-means prefers spherical clusters), which makes it adaptable for different dataset. The Euclidean metric and ward (sum of squares of deviations) are used for implementation.

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 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 ($THAvg_{LN_i}$). The cluster is defined as good if $\sum_{i=0}^{2} THAvg_{LN_i} \geq 0.7$, 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_{high} and T_{low}, are defined by $Avg_{LN_i} \pm STD_{LN_i}$ respectively. 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 Fig. 4.

Results

We use 1000 Genome Project [21] Phase3 dataset as the benchmark, and only the deletions recorded inside are viewed as true deletions. Seven existing tools are used for comparison, including Pindel, CNVnator [22], GASVpro [23], SvABA, Manta, Delly and Lumpy. We directly use low coverage BAM files provided by 1000 Genome Project as input. For some tools that require separate reads files, such as Lumpy, we dump reads from BAM file.

We evaluate the performance of balancing accuracy and sensitivity by using F1 score among these methods. 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. Therefore, 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 these samples and majority of chromosomes in one sample. Our results show that EigenDel performs better than others in all testing cases.

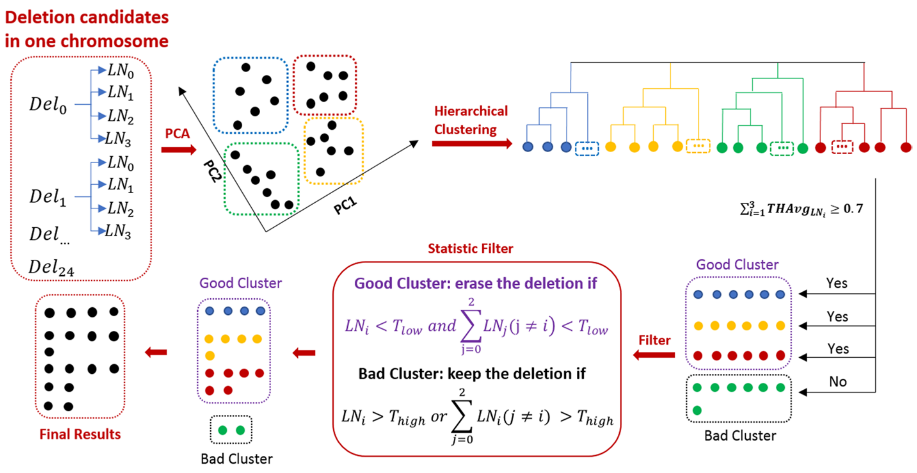
Deletion candidates in one chromosome
Del_0
LN_0
LN_1
LN_2
LN_3
Del_1
$Del_{...}$
Del_{24}
PCA
PC2
PC1
Hierarchical Clustering
$\sum_{i=1}^{3} THAvg_{LN_i} \geq 0.7$
Yes
Yes
Yes
No
Good Cluster
Bad Cluster
Filter
Statistic Filter
Good Cluster: erase the deletion if
$LN_i < T_{low}$ and $\sum_{j=0}^{2} LN_j(j \neq i) < T_{low}$
Bad Cluster: keep the deletion if
$LN_i > T_{high}$ or $\sum_{j=0}^{2} LN_i(j \neq i) > T_{high}$
Good Cluster
Bad Cluster
Final Results

Fig. 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

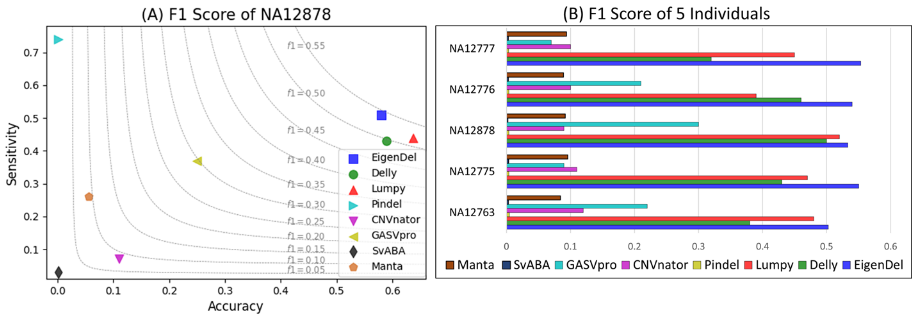

Fig. 5 F1 scores. **A** F1 scores of all comparison tools on the whole genome of NA12878. **B** F1 scores of all comparison tools on five 1000 Genomes individuals: NA12777, NA12776, NA12878, NA12775 and NA12763

NA12878

The individual NA12878 in 1000 Genomes Project has been studied by many researchers. We use the low coverage BAM file (20121211) of NA12878 from the 1000 Genomes Project for comparison. The average depth of this BAM file is 5.26. It contains the aligned result of SRR622461 (92,459,459 pair-end reads). The reads length in this sequence library is 101 bps. There are 1982 deletions from 23 different chromosomes of NA12878 are reported in benchmark.

The results are illustrated in Fig. 5A, Additional file 1: Table S1 and Fig. 6C.1, C.2. Figure 5A shows that EigenDel has the highest F1 score for NA12878. Additional file 1: Table S1 show that EigenDel has higher F1 score than others in the majority of chromosomes. Figure 6C.1, C.2 shows an example of the performance of unsupervised learning for chromosome 1. There are 149 deletion candidates detected in

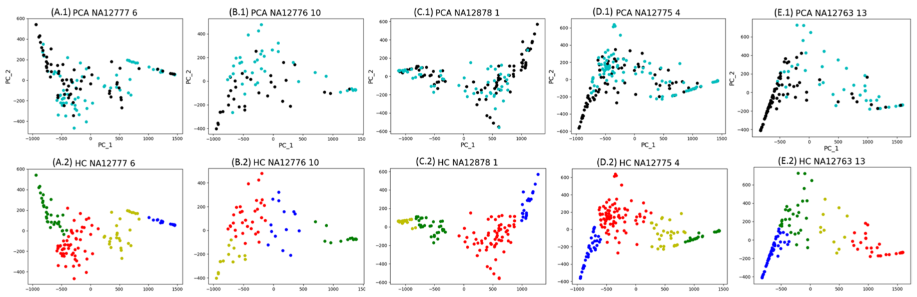

Fig. 6 Clustering results with unsupervised learning. **A.1**, **B.1**, **C.1**, **D.1**, **E.1** The two axes are from the top two principle components of PCA. The dots represent all deletion candidates in chromosome 6, 10, 1, 4 and 13 of NA12777, NA12776, NA12878, NA12775 and NA12763 respectively. 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

chromosome 1, and 67 of them are presented in benchmark (i.e., the presumed true deletions). X and Y axes in Fig. 6C.1, C.2 come from the top two principle components of PCA. Figure 6C.1 shows all deletion candidates found by EigenDel, and the cyan dots stand for the true deletions from Phase3 callset. Figure 6C.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 benchmark. 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.

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. Their reads depths are 9.08, 5.89, 5.26, 9.63 and 7.84 respectively. There are 2032, 2115, 1982, 1988 and 2105 deletions in benchmark for these five individuals respectively.

Figure 5B shows that EigenDel has the highest F1 score for all five individuals. 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 (Fig. 6A.1, A.2), 140 deletion candidates are detected and 75 of them are in benchmark. After the statistic filter be applied, 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 (Fig. 6B.1, B.2), 76 deletion candidates are detected and 43 of them are recorded in benchmark. After the statistic be applied, 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. For chromosome 4 in NA12775 (Fig. 6 D.1 and D.2), 181 deletion candidates are detected and 103 of them are recorded in benchmark. After the statistic filter be applied, 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 (Fig. 6E.1, E.2), 126 deletion candidates are detected and 47 of them are recorded in benchmark. After the statistic filter be applied, 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.

Case study: specific deletions analysis

Due to the unexpected mapping results and the complexity of genome sequence (e.g. repetitive regions), finding potential deletions is not always a straightforward job. In this section, we use IGV [25] to check the alignment results and list two typical deletions in Fig. 7 to show the advantage of EigenDel. These two deletions in Fig. 7 come from chromosome 1 in sample NA12777, and both of them are recorded in the 1000 genome trueset. In Fig. 7A, the deletion starts from 21786418 and ends at 21786695. The IGV shows that there are multiple clipped reads and discordant reads that clearly support both boundaries of this deletion. That's why all 8 tools used for comparison in our research work can find this deletion successfully. However, for the deletion shown in Fig. 7B, where the variant is from 63151819 to 63152158, the alignment results are much more complex. The clipped reads are not aligned together, and the discordant reads are distributed here and there. In addition, there are also some alignment results dropped into the deletion area. Therefore, due to these complex mapping results, it is hard to use the normal criteria to identify if it is a real deletion and where are its boundaries. With the help of unsupervised learning, EigenDel checks this event by tracking the relationship among the deletions with similar alignment situations, and it is the only algorithm among these 8 different methods that detects this variant successfully.

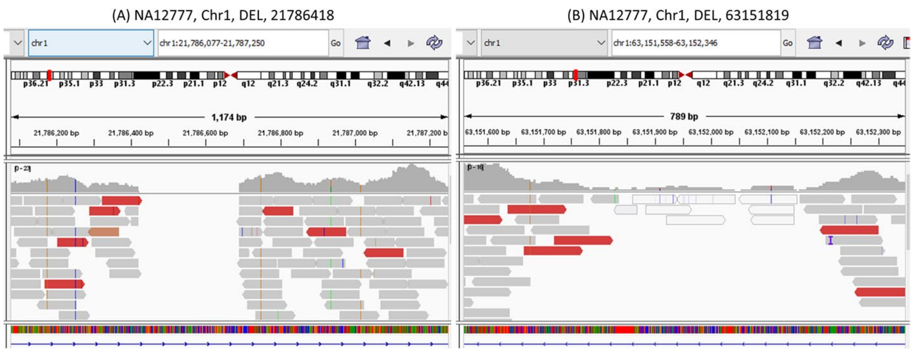

Fig. 7 Two specific deletions. **A** Deletion from 21786418 to 21786695, chromosome 1, sample NA12777. Detected by all 8 different methods. **B** Deletion from 63151819 to 63152158, chromosome 1, sample NA12777. Detected by EigenDel only

Discussion

Structural variants can be divided into two categories in terms of length, including the microscopic variants (large variant, > 3 Mb) and submicroscopic variants (small variant, 50 bp to 3 Mb). Microscopic variants have a relatively long history, since they are very long and easy to be found (e.g. visible to the unassisted eye). With the development of molecular biology and DNA sequencing technology, smaller and more abundant variants were observed. We call these smaller variants submicroscopic variants. Submicroscopic variants occur at a much higher frequency and are not easy to detect correctly. Our research work focuses on detecting these smaller events. In addition, the motivation of this paper is designing an efficient algorithm that can call as much as deletions recorded in SV trueset (sensitivity) while not introducing so many unrecorded deletions (accuracy). In order to balance sensitivity and accuracy, we use F1 score as the principal metrics for comparison. Proving whether or not the newly discovered deletions are real is not our research question.

EigenDel is designed for calling deletions based on illumina pair-end reads. All comparisons conducted in this paper are based on the low coverage BAM files from 1000 genome project phase 3 datasets. Since a low coverage data set does not contain enough high-quality reads, gaining best performance for balancing sensitivity and accuracy based on low coverage dataset is much more challenging than high coverage dataset, and this is one of the major motivations of EigenDel. We use multiple individuals, including some widely studied samples, such as NA12878, for comparison. Some BAM files contain single sequence library while others contain multiple libraries. When comparing the breakpoints of each deletion, we allow up to 15 bp tolerance.

For the performance, 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 20–50 mins on running each testing sample, and this is similar to CNVnator, Delly, GASVpro, and SvABA. Lumpy and Manta take around 1.5 h and 2.5 h on running each single individual respectively while Pindel costs about 5 h. As a result, the running time of EigenDel is competitive. EigneDel is designed for germline mutations of diploid organism. 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 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.

EigenDel is implemented by C++, and bamtools [26] is a very popular C++ toolkit to parse BAM files. Similar to samtools, bamtools can report many features of each alignment reads, such as PCR duplicate, reads QC, mate index, primary alignment, etc. EigenDel uses bamtools directly while implementing.

For future works, we plan to improve EigenDel in 4 aspects. First of all, Since very large deletions will induce read pairs with large insert sizes, average insert size may be more susceptible to outliers than median insert size. Therefore, instead of average insert size and insert size standard deviation, it is a good idea to use median insert size and median absolute deviations to collect discordant reads. Secondly, in order to reduce false positives, we can consider increasing the threshold by using a value larger than "a standard deviation coefficient of 3" to collect discordant reads. However, this change may cause the decrease of sensitivity as well. Therefore, it is a good idea to develop a statistical model to evaluate which value can provide us the best tradeoffs between sensitivity and accuracy. Thirdly, our current merging strategy may put two separated but overlapped deletions together, which may cause the incorrect variant calling result. We need to find a way to prevent these events. Finally, although it is good to gain better performance in low coverage dataset, making EigenDel fully support high coverage data is also necessary. This is because high coverage datasets contain more number of high-quality reads, which may improve the confidence of calling results. However, since the data scale in high coverage datasets is much larger than low coverage datasets, it is necessary to optimize some part of the current algorithm, such as merging discordant reads and finding potential deletion candidates, to improve the computational efficiency. The "learning method" may also need to be adjusted based on the running results from high coverage datasets.

Conclusion

In this paper, we design a method named EigenDel for detecting submicroscopic structural variations deletions in germline mutation of diploid organism. 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. The results show that EigenDel gives the highest F1 score in all experiments. For each individual, EigenDel performs better than other methods in the majority of chromosomes. 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.

Abbreviations

SV	Structure variation
DNA	Deoxyribonucleic acid
Avg_{IS}	The average insert size
STD_{IS}	The standard deviation of insert size
$Depth_{avg}$	Average depth
PCR	Polymerase chain reaction
PCA	Principle component analysis

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-023-05139-w.

Additional file 1. Supplemental materials for "Calling genomic deletions from sequence data using unsupervised learning". It contains the additional results presented in this paper. "**A** F1 score of NA12878 in each chromosome" contains the results of F1 score for each chromosome in NA12878 respectively. "**B** Command lines used for genomic deletions detection" is about the command lines used by each tool for deletion detection.

Acknowledgements
We thank doctor Huiting Qi for her help of revising the paper.

About this supplement
This article has been published as part of BMC Bioinformatics Volume 23 Supplement 8, 2022: Selected articles from the 16th International Symposium on Bioinformatics Research and Applications (ISBRA-20): bioinformatics. The full contents of the supplement are available online at https://bmcbioinformatics.biomedcentral.com/articles/supplements/volume-23-supplement-8.

Author Contributions
XL designed algorithms, developed software, performed analysis and experiments, and wrote the paper. YW designed the algorithms, revised the paper, and supervised the project. All authors read and approved the final manuscript.

Funding
This work is supported in part by Grants IIS-1526415 and CCF-1718093 from US National Science Foundation to YW. Funding agency has no role in design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

Availability of data and materials
The software is available under the GPLv3 licence at https://github.com/lxwgcool/EigenDel The sample datasets used in this paper are available at 1000 genome phase3 (ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data) with identification numbers NA12878, NA12777, NA12776, NA12878, NA12775 and NA12763. The structure variation ground truth dataset used in this paper is available at 1000 genome phase3 (ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/integrated_sv_map). The reference data and annotation files used in this paper are available at ensembl (http://useast.ensembl.org/index.html).

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 20 December 2022 Accepted: 5 January 2023
Published online: 27 January 2023

References

1. Feuk L, Carson AR, Scherer SW. Structural variation in the human genome. Nat Rev Genet. 2006;7(2):85.
2. Alkan C, Coe BP, Eichler EE. Genome structural variation discovery and genotyping. Nat Rev Genet. 2011;12(5):363.
3. Lewis R. Human genetics: concepts and applications. McGraw-Hill Higher Education, 2007. https://books.google.com/books?id=TKpyPwAACAAJ.
4. Mullaney JM, Mills RE, Pittard WS, Devine SE. Small insertions and deletions (INDELs) in human genomes. Hum Mol Genet. 2010;19(R2):131–6.
5. Firth HV, Richards SM, Bevan AP, Clayton S, Corpas M, Rajan D, Van Vooren S, Moreau Y, Pettett RM, Carter NP. DECIPHER: database of chromosomal imbalance and phenotype in humans using ensemble resources. Am J Hum Genet. 2009;84(4):524–33.

6. Weischenfeldt J, Symmons O, Spitz F, Korbel JO. Phenotypic impact of genomic structural variation: insights from and for human disease. Nat Rev Genet. 2013;14(2):125.
7. Abel HJ, Duncavage EJ. Detection of structural DNA variation from next generation sequencing data: a review of informatic approaches. Cancer Genet. 2013;206(12):432–40.
8. 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); 2017, p. 1. IEEE
9. Tattini L, D'Aurizio R, Magi A. Detection of genomic structural variants from next-generation sequencing data. Front Bioeng Biotechnol. 2015;3:92.
10. Ye K, Schulz MH, 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. 2009;25(21):2865–71.
11. Rausch T, Zichner T, Schlattl A, Stütz AM, Benes V, Korbel JO. DELLY: structural variant discovery by integrated paired-end and split-read analysis. Bioinformatics. 2012;28(18):333–9.
12. Layer RM, Chiang C, Quinlan AR, Hall IM. LUMPY: a probabilistic framework for structural variant discovery. Genome Biol. 2014;15(6):84.
13. Wala JA, Bandopadhayay P, Greenwald NF, O'Rourke R, Sharpe T, Stewart C, Schumacher S, Li Y, Weischenfeldt J, Yao X, et al. Svaba: genome-wide detection of structural variants and indels by local assembly. Genome Res. 2018;28(4):581–91.
14. Chen X, Schulz-Trieglaff O, Shaw R, Barnes B, Schlesinger F, Källberg M, Cox AJ, Kruglyak S, Saunders CT. Manta: rapid detection of structural variants and indels for germline and cancer sequencing applications. Bioinformatics. 2016;32(8):1220–2.
15. Michaelson JJ, Sebat J. ForestSV: structural variant discovery through statistical learning. Nat Methods. 2012;9(8):819.
16. Picard toolkit. Broad Institute 2019.
17. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R. The sequence alignment/map format and samtools. Bioinformatics. 2009;25(16):2078–9.
18. Liu W, Xie CC, Zhu Y, Li T, Sun J, Cheng Y, Ewing CM, Dalrymple S, Turner AR, Sun J, et al. Homozygous deletions and recurrent amplifications implicate new genes involved in prostate cancer. Neoplasia. 2008;10(8):897–37.
19. Johnson SC. Hierarchical clustering schemes. Psychometrika. 1967;32(3):241–54.
20. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, et al. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12(Oct):2825–30.
21. Siva N. 1000 Genomes project. Nature Publishing Group; 2008.
22. Abyzov A, Urban AE, Snyder M, Gerstein M. CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. Genome Res. 2011;21(6):974–84.
23. Sindi SS, Önal S, Peng LC, Wu H-T, Raphael BJ. An integrative probabilistic model for identification of structural variation in sequencing data. Genome Biol. 2012;13(3):22.
24. Li X, Wu Y. Detecting circular RNA from high-throughput sequence data with de Bruijn graph. bioRxiv 2019;509422.
25. Robinson JT, Thorvaldsdóttir H, Winckler W, Guttman M, Lander ES, Getz G, Mesirov JP. Integrative genomics viewer. Nat Biotechnol. 2011;29(1):24–6.
26. Barnett DW, Garrison EK, Quinlan AR, Strömberg MP, Marth GT. Bamtools: a C++ API and toolkit for analyzing and managing BAM files. Bioinformatics. 2011;27(12):1691–2.

Publisher's Note