Journal of
Bioinformatics and Comparative Genomics

Research Article Open Access

MapCaller – An Integrated and Efficient Tool for Short-Read Mapping and Variant Calling Using High-Throughput Sequenced Data

Hsin-Nan Lin and Wen-Lian Hsu*

Institute of Information Science, Academia Sinica, Taipei, Taiwan

***Corresponding author:** Wen-Lian Hsu, Institute of Information Science, Academia Sinica, Taipei 11529, Taiwan, Tel: +886-27883799, E-mail: hsu@iis.sinica.edu.tw

Received Date: April 18, 2021 **Accepted Date:** May 18, 2021 **Published Date:** May 20, 2021

Citation: Hsin-Nan Lin (2021) MapCaller – An Integrated and Efficient Tool for Short-Read Mapping and Variant Calling Using High-Throughput Sequenced Data. J Bioinfo Comp Genom 4: 1-11

Abstract

With the advance of next-generation sequencing (NGS) technologies, more and more medical and biological researches adopt NGS technologies to characterize the genetic variations between individuals. The identification of personal genome variants using NGS technology is a critical factor for the success of clinical genomics studies. It requires an accurate and consistent analysis procedure to distinguish functional or disease-associated variants from false discoveries due to sequencing errors or misalignments. In this study, we integrate the algorithms for read mapping and variant calling to develop an efficient and versatile NGS analysis tool, called MapCaller. It not only maps every short read onto a reference genome, but it also detects single nucleotide variants, indels, inversions and translocations at the same time. We evaluate the performance of MapCaller with existing variant calling pipelines using three simulated datasets and four real datasets. The result shows that MapCaller can identify variants accurately. Moreover, MapCaller runs much faster than existing methods. We demonstrate MapCaller is both time and space efficient for NGS data analysis. MapCaller is available at https://github.com/hsinnan75/MapCaller.

Keywords: Variant Calling; SNP; INDEL; Structural Variant

Background

With the advance of next-generation sequencing (NGS) technologies, it is becoming affordable to support various applications of precision medicine in the near future [1]. More and more medical and biological researches adopt NGS technologies to characterize the genetic variations between individuals[2,3]. Such genetic variations can be classified into three types: (1) single nucleotide variant (SNV, also referred to as SNP); (2) insertion and deletion (indel); and (3) structural variant (SV, including translocation, inversion, copy number variation and indels of size at least 50 bp).

The identification of genome variants is a critical factor for the success of clinical genomics studies [4]. It requires an accurate and consistent analysis procedure to distinguish true variants from false discoveries. This procedure often involves the steps of short read alignment, alignment rearrangement, and variant calling. In each step, one or more tools are applied to generate desired output. For example, BWA [5], Bowtie [6,7], GEM [8], Subread [9], HISAT/HISAT2 [10], and KART [11] are read aligners that can map NGS short reads onto a reference genome and generate their alignments. SAMtools [12] and Picard [13] provide various utilities for manipulating read alignments. For variant calling, the Genome Analysis Tool Kit (GATK) [14], Freebayes [15], Platypus [16], VarScan [17] and SAMtools are widely used. Different combinations of those tools produce various analysis pipelines. Different variant calling pipelines may generate substantial disagreements of variant calls. Several studies [4, 18] have been conducted to confirm the disagreements of variant calling among different pipelines. Besides, all existing variant calling pipelines are time and space consuming. They require read alignments are sorted and stored in desired format.

In this study, we present MapCaller, an efficient and versatile NGS analysis tool, by integrating the algorithms for read mapping and variant calling. For read mapping, we adopt a divide-and-conquer strategy to separate a read into regions with and without gapped alignment. With this strategy of read partitioning, SNVs, indels, and breakpoints can be identified efficiently. For variant calling, MapCaller maintains a position frequency matrix to keep track of every nucleobase's occurrence at each position of the reference genome while mapping the read sequences. Since MapCaller collects all information required for variant identification while reads are mapped onto the reference genome, variants can be called directly in the same process. Therefore, the conventional analysis pipeline can be simplified greatly. Most existing variant callers can only detect a few specific types of variants, however MapCaller can detect multiple types of variations, including SNVs, indels, inversions, and translocations. We demonstrate that MapCaller not only produces comparable performance on variant calling, but it also spends much less time compared to selected variant calling pipelines. MapCaller was developed under Linux 64-bit environment and implemented with standard C/C++. It takes read files (FASTA/FASTQ) as input and outputs all predicted variants in VCF format. The source codes of MapCaller and benchmark datasets are available at https://github.com/hsinnan75/MapCaller.

Results and Discussion

Experiment design

We develop a simulator to generate genome variations using the E.Coli K-12 strain, human chromosome 1, and whole human reference genome (GRCh38). The simulator (please refer to Supplementary material *S9*) randomly generates sequence variations with occurrences of 2700 substitutions, 180 small indels (1~10 bp), 45 large indels (11~50 bp), 1 translocation (TNL, size ranges 1000~2000bp), 1 inversion (INV, size ranges 1000~2000bp) and 1 copy number variation (CNV, size ranges 300~1300bp) for every 1,000,000 base pairs. We use WGSIM (https://github.com/lh3/wgsim) to generate simulated short read sequences for each mutant genome. The simulated read coverage is 30X, sequencing error rate is 0.02, and read length is 100bp. The synthetic datasets are referred to as Sim_Ecoli, Sim_Chr1, and Sim_HG respectively. We also download four real NGS datasets from SRA web site, two from sample of HG001-NA12878 (RUN: SRR6062143 and SRR7781445) and two from sample of HG002-NA24385 (RUN: SRR3440404 and SRR6691661), where the dataset of SRR3440404 includes short reads from RUNs of SRR3440404 to SRR3440422 in order to have enough read depth. Note SRR6062143 and SRR7781445 are two separate runs of Illumina sequencing data of NA12878. SRR3440404 and SRR6691661 are also two separate runs of NA24385. GIAB (The Genome in a Bottle Consortium) provides high-confidence SNP, small indel calls for the two sample genomes. They can be found at ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/release/. Those variant calls were made by GATK, Freebayes and Sentieon. Table 1 shows the number of short reads of each dataset as well as the number of each type of variants. It is noteworthy that GIAB does not provide structural variant annotation for samples NA12878 and NA24385.

A conventional analysis pipeline includes a read mapper, SAM/BAM file processing, and a variant caller. In study, we compare MapCaller to different combinations of those read mappers and variant callers. For read mapping, we selecte KART, BWA-MEM (BWA for short), Bowtie2, and GEM. For the variant calling, we select GATK HaplotypeCaller (GATK for short), Freebayes and SAMtools mpileup (Mpileup for short). For SAM/BAM file processing, we use SAMtools view/sort to perform file format converting and alignment sorting. We also compare the performance of structural variant calling with existing methods, including DELLY [19], LUMPY [20], and SVDetect [21]. The commands as well as the argument setting used for each pipeline are shown in Supplementary material (S3). handles the whole procedure of analysis pipeline alone. The run time is estimated from the read mapping to variant calling. We estimate the precision and recall for each dataset. We define a true positive case (TP) as a true variant call; a false positive case (FP) as a false variant call; and a false negative case (FN) as a true variant that is not called. A predicted SNV event is considered an TP if its genomic coordinate is correct without any tolerance, otherwise it is a FP. A predicted indel event is considered an TP if the genomic coordinate is within 10 bp, otherwise it is a FP.

Performance comparison on synthetic datasets

Table 2 summarizes the comparison result on the three synthetic datasets. We test every combination of read mapper and variant caller. We find that BWA-MEM combined with any variant caller generally performs better than any other selected read mapper; therefore, we only show the performance of pipelines involved with BWA-MEM here to compare to MapCaller. The complete experiment result can be found in Supplementary material (S4).

Table 1: The benchmark datasets. Sim_Ecoli, Sim_Chr1, and Sim_HG are synthetic datasets. SRR6062143 and SRR7781445 are generated from NA12878 (HG001); SRR3440404 and SRR6691661 are generated from NA24385 (HG002). TNL: translocation; INV: inversion

Dataset	# of short reads	read length	SNV	indels	TNL	INV
Sim_Ecoli	1,472,676	100	12,822	1,035	6	3
Sim_Chr1	78,859,696	100	627,576	52,067	261	211
Sim_HG	978,158,892	100	8,003,103	666,213	2,650	2,559
SRR6062143	1,558,904,052	101	3,084,732	534,739	NA	NA
SRR7781445	609,362,230	151	3,084,732	534,739	NA	NA
SRR3440404	487,042,582	250	3,076,552	519,569	NA	NA
SRR6691661	638,722,972	151	3,076,552	519,569	NA	NA

Table 2: The performance comparison on synthetic datasets. GATK producing low recall is due to high sequencing error rate (2%) and low sequencing coverage (30X). We investigate the effect of sequencing error rate and coverage in the Supplementary material

Pipeline	SNV		INDEL		Runtime
	Precision	Recall	Precision	Recall	(minutes)
Sim_Ecoli					
MapCaller	100	99.3	99.9	98.9	0.1
BWA + Freebayes	99.6	99.5	100	98.7	9.6
BWA + Mpileup	100	99.2	100	98.7	3.5
BWA + GATK	99.3	76.4	100	15.9	66.4
Sim_Chr1					
MapCaller	99.9	98.5	99.8	98.0	8.3
BWA + Freebayes	99.3	99.0	99.9	98.3	611.5
BWA + Mpileup	100	98.6	99.9	97.9	277.7
BWA + GATK	99.5	78.4	99.6	17.1	3264.2
Sim_HG					
MapCaller	99.2	96.9	99.1	96.3	146.1
BWA + Freebayes	99.2	92.3	99.9	91.6	6954.7
BWA + Mpileup	44.7	91.8	99.9	91.0	3414.1
BWA + GATK	99.4	71.7	99.8	14.9	36812.3

It can be observed that MapCaller and Freebayes perform comparably on the three synthetic datasets; however, MapCaller produces better performance on Sim_HG with respect to recalls. The SNV and indel recalls of MapCaller on Sim_HG are 96.9% and 96.3% respectively. When further analyze some of the false negative cases, we find that most of false negatives occur at repetitive or highly similar regions. Freebayes and mpileup produces higher precisions on indel detection on Sim_HG; however, they compromise the recalls. The precisions and recalls of Freebayes are 99.9% and 91.6%, and those of mpileup are 99.9% and 91.0%. MapCaller produces balanced result on indel detection. Its precision and recall are 99.1% and 96.3% respectively.

Surprisingly GATK produces poor recalls on SNV and indel detection on all of the three datasets. The recalls on SNV detections are all below 80%, and those on indel detection are all below 20%. In this experiment, we feed all the three variant callers with the same read alignments; however, the fact that Freebayes and Mpileup produce much higher recalls on the three datasets suggests BWA can process those indel events correctly in the alignments. We try to diagnose the reason by configuring the pipeline but it did not help. We then increase the read depth from 30X to 100X (Sim_Ecoli) and we find both of the recalls improve from 76.4% to 95.7% (SNV) and from 15.9% to 65.7% (indel). If we further increase the read depth to 200X (Sim_Ecoli), the two recalls achieve 96.7% and 87.2% respectively. It suggests that GATK performs much better on NGS data with deeper coverage. The performance regarding different read depths is shown in Supplementary material (S5). We also find that if the NGS data is error-free, then the performance of GATK will be very good. We simulate an NGS data without any sequencing errors (read depth: 30X, Sim_Ecoli) and test with GATK combined with BWA, the precision and recall on SNV detection are 100% and 99.4% respectively, and those on indel detection are 99.5% and 99.0% respectively. It implies that GATK is more sensitive to sequencing errors.

Another noteworthy observation is that Mpileup performs much worse on Sim_HG. The precision on SNV detection is only 44.7%. When we further analyze the output, we find that Mpileup generates many false positive SNVs at loci with very shallow depth. Fortunately, we can filter out some of the false calls according to their confidence scores.

In terms of run time, it can be observed that MapCaller is much more efficient than the selected pipelines. For example, MapCaller only spends 148 minutes to handle Sim_HG. However, the other three pipelines spend 3414, 6954, and 36812 minutes respectively on the dataset. Since the variant analysis pipeline consists of multiple steps, we can further decompose the runtime into multiple parts. Take the pipeline of BWA+GATK for example, it spends 1444 minutes for read mapping, 249 minutes for SAM/BAM file processing, and then another 35119 minutes for variant calling. If we only consider the run time for variant calling, MapCaller only spends 2.6 minutes (155 seconds). MapCaller is much more efficient because it not only adopts a very efficient read mapping algorithm, but it also saves run time by skipping processing SAM/BAM files and collects variant information directly during read mapping.

Performance comparison on real datasets

We download four real NGS datasets sequenced for genomes of NA12878 (HG001) and NA24385 (HG002). However, the two genomes do not have ground truth variant annotation. GIAB provides reference calls that were made by GATK, Freebayes and Sentieon for the sample genomes. In this study, we use the reference calls as the gold standard to estimate the performance of each method. We download two separate datasets from each sample genome because we would like to demonstrate variant callers produce difference results on different data sets even if the sample genome is the same.

Table 3 summarizes the comparison result. It can be observed that MapCaller, Freebayes and GATK perform comparably. It is not obvious that which method performs the best. Each method has their strength and weakness. For example, MapCaller and GATK generally produce higher precisions on SNV detections than the other two methods. Freebayes and GATK generally produced higher recalls on indel detections. However, it seems like Mpileup produces relatively lower precisions on SNV detection and lower recalls on indel detection. Mpileup still generates many false positive cases on SNV detection. Its precisions are all below 20%. We also find that some of false calls made by MapCaller are due to ambiguous alignments of indels. If we relax the tolerance of locus difference for indel events to 50bp, the precisions and recalls of MapCaller on indel detection will be increased by around 10% and 4% respectively.

Moreover, some of false negative cases are due to poor alignment quality and their read alignments are discarded by MapCaller. For example, the reference calls report three adjacent variants, which are chr1:20761541 (an insertion: AGAG), chr1:20761544 (SNV: C), and chr1:20761545 (a deletion: AT). Since there are more than three state transitions in the corresponding alignment, MapCaller generate the corresponding alignments

Table 3: The performance comparison on real datasets.

Pipeline	SNV		INDEL		Runtime
	Precision	Recall	Precision	Recall	(hours)
SRR6062143					
MapCaller	76.2	98.8	66.4	89.2	2.0
BWA + Freebayes	70.8	96.2	61.3	91.8	122.7
BWA + Mpileup	22.3	97.4	70.1	73.5	70.2
BWA + GATK	77.1	97.3	67.1	95.5	203.8
SRR7781445					
MapCaller	78.6	97.3	66.2	88.3	1.5
BWA + Freebayes	72.1	96.1	58.3	95.0	122.3
BWA + Mpileup	17.8	97.4	68.0	71.4	71.8
BWA + GATK	77.3	97.2	63.9	96.0	161.8
SRR3440404					
MapCaller	76.4	98.3	66.0	89.8	2.0
BWA + Freebayes	67.3	95.8	60.1	95.5	116.4
BWA + Mpileup	8.5	99.7	66.8	73.1	60.9
BWA + GATK	77.1	99.7	68.7	95.7	227.9
SRR6691661					
MapCaller	77.5	97.9	60.7	91.3	1.5
BWA + Freebayes	70.4	95.8	58.1	94.8	96.5
BWA + Mpileup	16.8	99.4	66.6	76.2	49.7
BWA + GATK	78.2	99.8	63.2	99.0	157.1

and then they are considered poor alignment and discarded. If the adjacent variants appear in two separated alignments, they still can be called by MapCaller. It is estimated NA12878 contains 11,018 indel events (2.1%) that are adjacent to one another within five nucleotides, and NA24385 contains 10,016 variants (2.0%). By contrast, MapCaller produces around 0.7% of indels that are adjacent within five nucleotides in average.

Since the reference calls were made by integrating three different callers, we analyze the performance of overlapping calls by integrating MapCaller, Freebayes, Mpileup and GATK on the dataset of SRR6062143. If an SNV is called by all the four methods, the precision is 83.6%. If an indel is called by all the four methods, the precision was 79.2%. It suggests that the overlaps of variant calls can increase calling accuracy significantly. Moreover, we also find the union of the four callers can cover 99.9% of reference SNVs and 98.6% of reference indels. The performance with respective to overlaps among the four callers for each dataset is shown in Supplementary material (S6).

In terms of run time, MapCaller is still much faster than any other pipeline. It is around 100 times faster than BWA+GATK. It spends around one or two hours to handle a human genome data with around 30X of read depth. If we only consider the run time for variant calling, MapCaller is much faster than GATK. For example, MapCaller spends three minutes on variant calling for SRR6062143, while GATK spends 203.8 hours for the same dataset. Though MapCaller runs very fast, it produces comparable result as Freebayes and GATK do in this analysis. Thus, we demonstrate MapCaller is a highly efficient variant calling method.

Performance comparison on structural variant detection

MapCaller is capable of identifying structural variants (referred to as SVs in the following description) simultaneously. We do not identify CNVs in this study. CNV calling will be included in MapCaller in the future version. Table 4 shows the performance comparison on SV detection for MapCaller and other selected callers. We estimate the precision and recall of each caller. It can be observed that MapCaller produces high precisions and recalls on all the three simulated datasets. In particularly, MapCaller produces the highest precisions among these callers. LUMPY produces high precisions and recalls on inversion detections. However, it produces relatively lower precisions and recalls on translocation detections. SVDetect loses precisions on translocation and inversion detection, but it generates much higher recalls on the two types of structural variants. Similar to LUMPY, DELLY generally produces high precisions and recalls on inversion detections, but it cannot detect any translocation events on the three datasets. In summary, MapCaller is the only method that handles both translocation and inversion detection at high precision and recall.

Variant filtering

We estimate the performance of MapCaller and each selected pipeline using the raw result. However, most variant callers provide specific filters to remove unlikely variants based on their algorithm design and statistic models; it may complicate the comparison if we try to optimize the filtering strategy for each caller. According to the performance comparison result using synthetic datasets, we find that MapCaller and the other selected methods produce high precisions on SNV and indel detection, except Mpileup loses its SNV accuracy on the dataset of Sim_HG. It suggests that MapCaller, Freebayes and GATK can produce reliable variant calls without any specific filters. However, Mpileup tends to produce more SNV calls. It is necessary to filter out false calls to improve the accuracy. Since all variant callers give an estimate on how likely a variant call is true with a quality score (the QUAL column), we analyze the SNV accuracy regarding Mpileup on Sim_HG, SRR6062143, SRR7781445, SRR3440404 and SRR6691661 to investigate the relationship between precision/recall and QUAL values. The analysis result is shown in Supplementary material (S7).

Table 4: The performance comparison on real datasets.

Data set	Method	Translocation		Inversion	
		Precision	Recall	Precision	Recall
Sim_Ecoli	MapCaller	100	100	100	100
	LUMPY	100	66.7	100	100
	SVDetect	100	100	100	100
	DELLY	NA	NA	100	100
Sim_Chr1	MapCaller	100	94.3	100	87.7
	LUMPY	84.2	33.3	99.5	92.9
	SVDetect	27.2	97.7	64.8	93.8
	DELLY	NA	NA	96.9	95.7
Sim_HG	MapCaller	99.9	94.8	100	87.8
	LUMPY	87.9	35.3	100	93.7
	SVDetect	28.9	95.4	62.3	93.1
	DELLY	NA	NA	99.1	95.2

Comparison of data storage to existing methods

All current variant calling pipelines require SAM/BAM files as inputs and take multiple steps to rearrange the alignments for variant calling. It is not only time consuming, but also requires a huge amount of disk space. MapCaller identifies variants directly from the NGS short reads. Thus, it is not necessary to output SAM/BAM files and it saves a lot of disk space. Pipelines involved with GATK require even more disk space for the additional preprocess of alignments. We compare the data storage space of each pipeline in the Supplementary material (S8).

Conclusion

In this manuscript, we present MapCaller, an integrated system for read mapping and variant calling. MapCaller collects read alignment information during read mapping and maintains a position frequency matrix to keep track of alignments at each position of the reference genome. We evaluate the performance of MapCaller and the selected variant calling pipelines using three synthetic datasets and four real data sets from human genomes. The comparison results show that MapCaller not only identifies highly accurate variants, but it also spends the least amount of time. MapCaller is also versatile. It is capable of identifying SNVs, INDELs, and structural variants simultaneously.

Since more and more medical and biological researches adopt NGS technologies to characterize the genetic variations between individuals, we believe MapCaller is able to provide accurate and reliable variant calls much faster than existing methods.

Methods

MapCaller aligns every short read onto a reference genome and collects the alignment information during read mapping to identify sequence variants. MapCaller uses a modified algorithm of KART to perform read mapping. It maintains a position frequency matrix to keep track of every nucleobase's frequency at each position in the reference genome and collects all insertion and deletion events that are found during read mapping. Furthermore, It gathers all possible breakpoints from discordant or partial read alignments. Finally, MapCaller finds sequence variants based on all of the above-mentioned information. The novelty of our algorithm derives from the integration of read mapping and variation information gathering into a coherent system for variant calling.

Read mapping and alignment profiles

The details of read mapping method, KART can be found in our previous study [11]. Here we focus on the high-level methodology description. KART adopts a divide-and-conquer strategy to handle matches and mismatches separately between read sequence and reference genome. KART identifies all locally maximal exact matches (LMEMs). We then cluster them according to their coordinates and fill gaps between LMEMs to create final alignments. KART divides a read alignment into two groups: simple region pairs (abbreviated as *simple pairs*) and normal region pairs (*normal pairs*), where all simple pairs are LMEMs and normal pairs are gaps between simple pairs and might require gapped alignment (due to mismatches or indels).

Given a read sequence R, the reference genome G, Let R_i be the i-th residue of R and $R[i1, i2]$ be the substring between R_{i1} and R_{i2}. Likewise, let G_j be the j-th nucleotide of G and $G[j1, j2]$ be the substring between G_{j1} and G_{j2}. A simple pair (or a normal pair) consists of a read's substring and its counterpart of reference's substring. They can be represented as $(R[i1, i2], G[j1, j2])$. One or more simple/normal pair forms a candidate alignment. We perform pairwise alignment for each normal pair in a candidate alignment.

We check the alignment quality of a normal pair at either end of the read sequence to determine whether they should be discarded. The quality evaluation is as follows. Given an alignment of a normal pair at either end, MapCaller counts the number of mismatches and state transitions of the alignment. A state transition is an alignment state change of base alignment, insertion and deletion. If there are more than three state transitions or the number of mismatches is more than a threshold in the normal pair, it will be discarded from the candidate alignment. Such normal pair may appear due to false mapping or structural variants. MapCaller infers breakpoints based on such normal pairs.

Fig. 1 illustrates two cases of read alignments that reveal a translocation event and an inversion event respectively. In Fig. 1(A), paired-end reads Read1 and Read2 are mapped far away from each other due to a translocation event in the sample genome. The two alignments are considered discordant. Read1 and Read2 are clipped a few bases since they cover a breakpoint. Likewise, in Fig. 1(B), paired-end reads Read3 and Read4 are

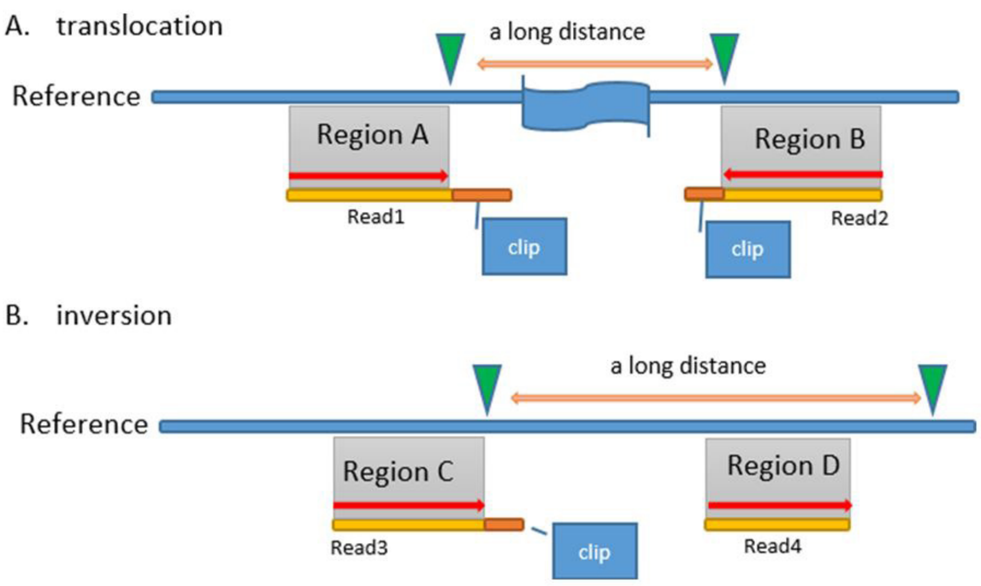
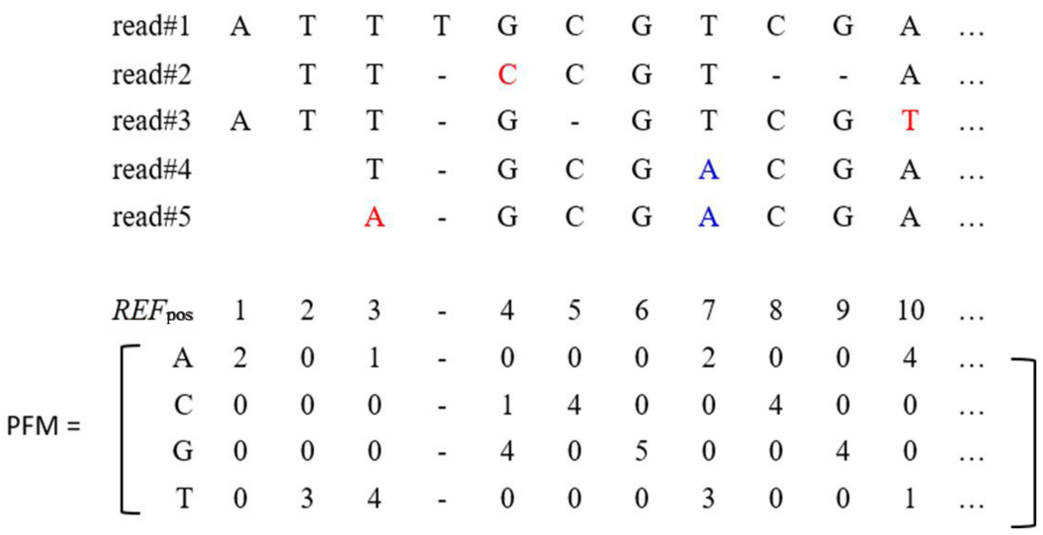


Figure 1: (A) Paired-end Read1 and Read2 are mapped distantly due to a translocation event in the sample genome. (B) Paired-end Read3 and Read4 are mapped with same orientation due to an inversion event in the sample genome



```
read#1   A   T   T   T   G   C   G   T   C   G   A   …
read#2       T   T   -   C   C   G   T   -   -   A   …
read#3   A   T   T   -   G   -   G   T   C   G   T   …
read#4           T   -   G   C   G   A   C   G   A   …
read#5           A   -   G   C   G   A   C   G   A   …
```

$$\begin{array}{c|c|cccccccccccc} & REF_{pos} & 1 & 2 & 3 & - & 4 & 5 & 6 & 7 & 8 & 9 & 10 & \dots \\ \hline & A & 2 & 0 & 1 & - & 0 & 0 & 0 & 2 & 0 & 0 & 4 & \dots \\ \text{PFM} = & C & 0 & 0 & 0 & - & 1 & 4 & 0 & 0 & 4 & 0 & 0 & \dots \\ & G & 0 & 0 & 0 & - & 4 & 0 & 5 & 0 & 0 & 4 & 0 & \dots \\ & T & 0 & 3 & 4 & - & 0 & 0 & 0 & 3 & 0 & 0 & 1 & \dots \end{array}$$

Figure 2: An example of position frequency matrix (PFM). Five read alignments are used to count the frequency of A, C, G and T at each position. Nucleobases in red are sequencing errors, and those in blue are SNPs

mapped with same orientation due to an inversion event in the sample genome. The two alignments are also considered discordant. MapCaller infers breakpoints both from the clipped and discordant alignments. On the other hand, if two normal pairs at both ends are removed after quality evaluation, then we will discard the whole candidate alignment since it is very likely the candidate alignment is a false alignment. Finally, each candidate alignment is evaluated by their numbers of exact matches and mismatches. For each short read, we only keep the alignment with the highest alignment score.

MapCaller creates a position frequency matrix (PFM) to count the nucleobase occurrences at each position. PFM is a matrix of $4 \times L$, where L is the reference genome size. Therefore, each column of PFM represents the occurrences of nucleobases A, C, G, and T at that position of the reference genome. MapCaller updates the occurrence at each column according to read alignments. Fig. 2 shows an example to illustrate how PFM works in this study. Five reads are mapped onto the reference genome. MapCaller counts the frequencies of each involved column. For example, PFM[3] = (1, 0, 0, 4) indicates there are one 'A' and four 'T's aligned at the third position of the reference genome. Any insertion and deletion events are kept otherwise. PFM can be further extended by increasing the column size if we would like to keep more mapping information.

We use a 3-tuple, *Ins*(*Gpos, R*[*i,j*], *k*) and *Del*(*Gpos*, *G*[*m,n*], *k*) to represent an insertion and deletion event, where *Gpos* indicates the occurrence location, *R*[*i,j*] and *G*[*m,n*] are the indel strings, and *k* is the number of occurrences. MapCaller would create the following three 3-tuples: *Ins*(3, T, 1), *Del*(5, C, 1), and *Del*(8, CG, 1) based on the example cases in Fig 2. We describe how the SNVs, indels, inversions, and translocations are identified based on the PFM and by MapCaller below. It is noteworthy that the quality measurement of each type of variant is described in the Supplementary (S1).

SNV detection

MapCaller uses PFM to keep track of the occurrences of the four nucleobases at each position in the reference genome. The depth of position *p*, denoted as *Depth*(*p*), where *Depth*(*p*) = PFM[p, A]+ PFM[p, C]+ PFM[p, G] + PFM[p, T]. We partition the reference genome sequence into blocks of 100 nucleobases. For each block *i*, MapCaller determines a threshold, denoted as *depthr*(*i*), which is the half of the average depth for all the nucleobases within the block.

A nucleobase at position *p* is considered an alternative allele if its occurrence is above *OccurrenceThr*(*p*)= *Depth(p)* × *MinAlleleFrequency*, where *MinAlleleFrequency* is a user defined threshold. Thus, a nucleobase *b* at position *p* is considered an SNV if the following two conditions are satisfied: (1) *Depth*(*p*) ≥ *depthr*(*i*); (2) PFM[p, b] ≥ *OccurrenceThr*(*p*). Since the sample genome may carry multiple sets of chromosomes, the genotype at the same position can be homozygous or heterozygous. MapCaller considers both haploid and diploid scenarios. If only one nucleobase is called and its frequency is less than 1- *MinAlleleFrequency*, then the locus is considered heterozygous, otherwise it is homozygous.

Indel detection

MapCaller keeps track of all indel events using the 3-tuples, thus indel events can be deduced from those 3-tuples. We use *InsOccurrence*(*p*) and *DelOccurrence*(*p*) to represent the occurrences of insertion and deletion events at position *p*. However, the alignments involved with indels could be ambiguous. For example, the two following alignments produce identical alignment scores:

```
AGCATGCATTG       AGCATGCATTG

AGCAT----TG  and  AG----CATTG
```

It can be observed that the two alignments will lead to different indel events, which are GCAT and CATG at neighboring positions. To avoid the ambiguity, MapCaller finds the indel with the maximal occurrences within the range of (p -5, p + 5). MapCaller also counts the total occurrences of insertions and deletions (denoted as *InsOccurrence*(*p*) and *DelOccurrence*(*p*)) within the range. MapCaller reports an insertion event at position *p* only if *InsOccurrence*(*p*) ≥ *depthr*(*i*) × 0.25. Likewise, it reports a deletion event at position *p* only if *DelOccurrence*(*p*) ≥ *depthr*(*i*) × 0.35. Since *depthr(i)* is normally lower than the neighboring depths when there are deletion events, we use 0.35 to determine the threshold for deletion detection.

Translocation and inversion detection

MapCaller estimates the average distance and fragment size of the paired-end reads periodically during read mapping. They are denoted as *AvgDist* and *AvgFragmentSize*. The definition of the distance and fragment size is described in Supplementary material (S2). We use *AvgDist* to distinguish concordant pairs from discordant pairs. In Illumina sequencing protocol, two paired-end reads are supposed to be mapped to different orientations of the same chromosome within an expected distance. Concordant pairs match paired-end expectations, whereas discordant pairs do not. Figure. 3(A) shows examples of regular paired-end reads that are mapped

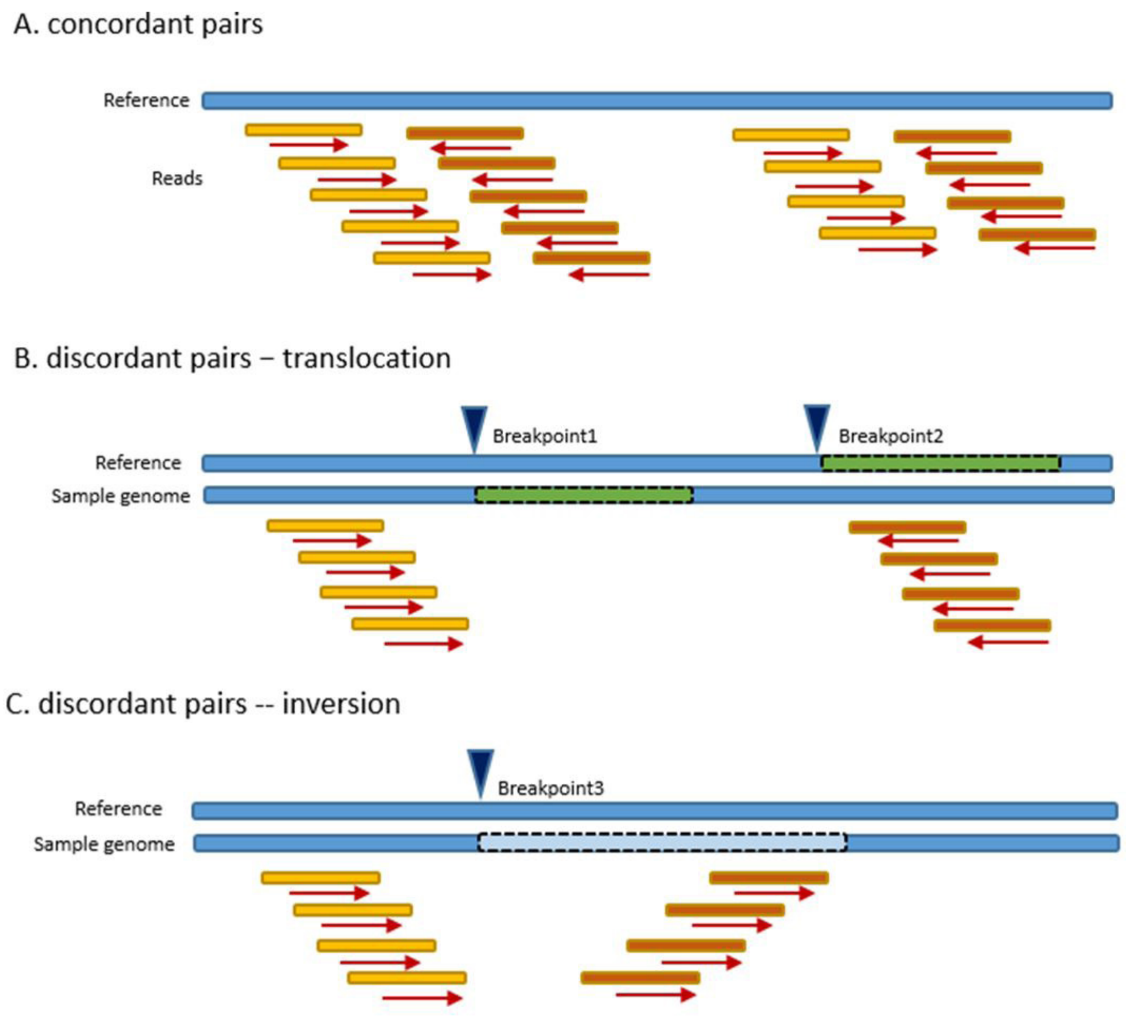


Figure 3: (A) Concordant pairs (B) discordant pairs –translocation (C) discordant pairs – inversion

concordantly since the sample genome do not have any structural variants at the corresponding region. Figure. 3(B) shows a translocation event. The green dotted- rectangle is translocated from Breakpoint2 to Breakpoint1. Thus, the corresponding paired-end reads will be mapped discordantly. Likewise, Figure 3(C) shows an example of an inversion event at BreakPoint3. The light-blue dotted-rectangle is the reversal of itself. Therefore, the corresponding paired-end reads will be mapped to the same orientation.

MapCaller collects all discordant pairs and classifies them into translocation and inversion candidate groups. If paired-end reads are mapped to different orientation and their distance is greater than ($AvgDist \times 1.5$), they are put into translocation group, denoted as *TnlGroup*. If paired-end reads are mapped to the same orientation, they are put into inversion group, denoted as *InvGroup*. Since MapCaller collects all breakpoints from read alignments with a single-end clipping, we set a window to identify breakpoints as follows. For each breakpoint p, we set a screening region of *AvgFragmentSize* width at both sides of p. To identify breakpoints of translocation events, we count the number of reads in *TnlGroup* that are mapped at the screening region. The number of reads at left side is denoted as $LRnum(p)$ and the right side is denoted as $RRnum(p)$. We measure the average read depth at both sides, denoted as $Ldepth(p)$ and $Rdepth(p)$. If p is a true breakpoint for a translocation event, we should be able to observe a certain number of discordant read alignments in the *TnlGroup* that are separated by the breakpoint. Thus, we decide p is a breakpoint if the following conditions are satisfied: (1) $LRnum(p) \geq depthr(i)$; (2) $LRnum(p) \geq Ldepth(p) \times 0.5$; (3) $RRnum(p) \geq depthr(i)$; and (4) $RRnum(p) \geq Rdepth(p) \times 0.5$. To identify breakpoints of inversion events, we adopt similar process to verify the positivity of every breakpoint by checking the discordant pairs in the *InvGroup*.

Data Availability

MapCaller was developed under Linux 64-bit environment and implemented with standard C/C++. It takes read files (FASTA/FASTQ) as input and outputs all predicted variants in VCF format. The source codes of MapCaller and benchmark datasets are available at https://github.com/hsinnan75/MapCaller.

Acknowledgement

We would like to appreciate Prof. Torsten Seemann (Microbiological Diagnostic Unit) and Dr. Devon Ryan (Max Planck Institute of Immunobiology and Epigenetics) for their help with the software.

Funding

This work was supported by Bioinformatics Core Facility for Translational Medicine and Biotechnology Development/Ministry of Science and Technology (Taiwan) 105-2319-B-400-002.

Author contributions

H.N.L. conceived and designed the study; H.N.L. collected and pre-processed the datasets; H.N.L. developed the software; H.N.L. and W.L.H. wrote the manuscript. All authors read and approved the final manuscript.

Additional Information

Competing Interests: The authors declare no competing interests.

References

1. Ku CS, Roukos DH (2013) From next-generation sequencing to nanopore sequencing technology: paving the way to personalized genomic medicine. Expert Rev Med Devic 10: 1-6.

2. Kulkarni P, Frommolt P (2017) Challenges in the Setup of Large-scale Next-Generation Sequencing Analysis Workflows. Comput Struct Biotec 15: 471-7.

3. Dong L (2015) Clinical Next Generation Sequencing for Precision Medicine in Cancer. Curr Genomics 16: 253-63.

4. Hwang S, Kim E, Lee I, Marcotte EM (2015) Systematic comparison of variant calling pipelines using gold standard personal exome variants. Sci Rep-Uk.

5. Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 25: 1754-60.

6. Langmead B, Trapnell C, Pop M, Salzberg SL (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome biology 10, R25.

7. Langmead B, Salzberg SL (2012) Fast gapped-read alignment with Bowtie 2. Nature methods 9: 357-9.

8. Marco-Sola S, Sammeth M, Guigo R, Ribeca P (2012) The GEM mapper: fast, accurate and versatile alignment by filtration. Nature methods 9: 1185-U1176.

9. Liao Y, Smyth GK, Shi W (2013) The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Res 41.

10. Kim D, Landmead B, Salzberg SL (2015) HISAT: a fast spliced aligner with low memory requirements. Nature methods 12: 357-U121.

11. Lin HN, Hsu WL (2017) Kart: a divide-and-conquer algorithm for NGS read alignment. Bioinformatics 33: 2281-7.

12. Li H (2009) The Sequence Alignment/Map format and SAMtools. Bioinformatics 25: 2078-9.

13. Institute B (2019) Picard toolkit.GitHub Repository.

14. DePristo MA (2011) A framework for variation discovery and genotyping using next- generation DNA sequencing data. Nat Genet 43: 491.

15. Erik Garrison GM (2012) Haplotype-based variant detection from short-read sequencing. arXiv preprint, q-bio.GN.

16. Rimmer A (2014) Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications. Nat Genet 46: 912-8.

17. Koboldt DC (2009) VarScan: variant detection in massively parallel sequencing of individual and pooled samples. Bioinformatics 25: 2283-5.

18. Yen JL (2017) A variant by any name: quantifying annotation discordance across tools and clinical databases. Genome Med 9.

19. Rausch T (2012) DELLY: structural variant discovery by integrated paired-end and split-read analysis. Bioinformatics 28: I333-9.

20. Layer RM, Chiang C, Quinlan AR, Hall IM (2014) LUMPY: a probabilistic framework for structural variant discovery. Genome biology 15.

21. Zeitouni B (2010) SVDetect: a tool to identify genomic structural variations from paired-end and mate-pair sequencing data. Bioinformatics 26: 1895-6.