Vaquita: Fast and Accurate Identification of Structural Variation Using Combined Evidence

Jongkyu Kim[1] and Knut Reinert[2]

1 **Department of Mathematics and Computer Science, Freie Universität Berlin, Berlin, Germany; and**
Max Planck Institute for Molecular Genetics, Berlin, Germany
j.kim@fu-berlin.de

2 **Department of Mathematics and Computer Science, Freie Universität Berlin, Berlin, Germany; and**
Max Planck Institute for Molecular Genetics, Berlin, Germany
knut.reinert@fu-berlin.de

Abstract

Motivation: Comprehensive identification of structural variations (SVs) is a crucial task for studying genetic diversity and diseases. However, it remains challenging. There is only a marginal consensus between different methods, and our understanding of SVs is substantially limited. In general, integration of multiple pieces of evidence including split-read, read-pair, soft-clip, and read-depth yields the best result regarding accuracy. However, doing this step by step is usually cumbersome and computationally expensive.

Result: We present Vaquita, an accurate and fast tool for the identification of structural variations, which leverages all four types of evidence in a single program. After merging SVs from split-reads and discordant read-pairs, Vaquita realigns the soft-clipped reads to the selected regions using a fast bit-vector algorithm. Furthermore, it also considers the discrepancy of depth distribution around breakpoints using Kullback-Leibler divergence. Finally, Vaquita provides an additional metric for candidate selection based on voting, and also provides robust prioritization based on rank aggregation. We show that Vaquita is robust in terms of sequencing coverage, insertion size of the library, and read length, and is comparable or even better for the identification of deletions, inversions, duplications, and translocations than state-of-the-art tools, using both simulated and real datasets. In addition, Vaquita is more than eight times faster than any other tools in comparison.

Availability: Vaquita is implemented in C++ using the SeqAn library. The source code is distributed under the BSD license and can be downloaded at http://github.com/seqan/vaquita.

1998 ACM Subject Classification J.3 Life and Medical Sciences

Keywords and phrases Structural variation

Digital Object Identifier 10.4230/LIPIcs.WABI.2017.13

1 Introduction

Next generation sequencing (NGS) provides us remarkable opportunity to find genetic variants that are directly linked to diseases such as cancer [13] and rare genetic disorders [2]. Therefore, there has been a growing attention in identifying such variants. The size of genetic variations ranges from a single base pair to megabases [15]. Among them, structural variations (SVs), i.e. variations that are usually larger than 50 nucleotides in size, play a major role in many phenotypic differences. In contrast to single-nucleotide polymorphisms

(SNPs) or small indels, SVs are much more diverse in type and size and are often harder to find confidently [1]. Consequently, it is not surprising that there is only a marginal consensus between different variant callers [1]. It is fair to say that the current understanding of SVs is substantially limited and large-scale studies often rely on multiple variant callers that use different methods to obtain the most comprehensive list of SVs. However, the integration of multiple outputs is often cumbersome due to the required prior knowledge of different algorithms and their parameters and also suffers from limited computational resources. Therefore, there is an urgent need for a better method that can detect SVs more accurately and efficiently.

The algorithms for SV identification can be categorized into four types [1]. First, we can use split-read evidence. The reads spanning a breakpoint have to be split to be able to map multiple loci. For example, Pindel [24] splits discordant reads and tries to find breakpoints by mapping them to different positions. However, it is difficult to find SVs in some part of a genome such as repeat-rich regions using just the spilt-read information.

Additionally, read-pair information can be used to identify SVs. With the prior knowledge of the proper orientations and distribution of insertion sizes in paired-end sequencing libraries, read-pairs with improper orientation and/or insertion size can be identified and used to detect SVs. However, read-pair information alone does not provide base-pair resolution accuracy. Accordingly, a variant caller like Delly [19] considers read-pair information together with split-read information.

Many recently-developed short-read mappers [12, 10] provide local alignments. These mappers produce soft-clipped reads, meaning that only a part of a sequence is mappable to the reference genome. The unmapped sequences are relatively short and erroneous, which make them difficult to map to a unique position. To resolve this issue, CREST [23] assembles contigs around potential breakpoints and map them to a reference genome using Blat [9].

Lastly, read-depth information is also useful in finding copy number variations. However, the depth of coverage of sequencing data is usually non-uniform [14]. Thus, a significance testing such as event-wise testing [25] is required to distinguish the true signals from background noise. Moreover, read-depth information alone cannot provide base-pair resolution accuracy.

Often, integrating results from multiple approaches yield better performance regarding accuracy. In this aspect, LUMPY [11] uses a probabilistic framework to combine split-read and read-pair information by default, and MetaSV [16] focuses on connecting multiple external tools.

Our method, Vaquita, integrates split-read, read-pair, soft-clipped, and read-depth information in a single program to achieve maximum accuracy while also maintaining speed. Vaquita utilize all four types of information without contributions from external tools. The overall workflow of Vaquita is depicted in Figure 1.

2 Methods

2.1 Breakpoint and structural variation identification

The overall workflow of Vaquita is depicted in Figure 1(a). We define a breakpoint using the coordinate information of two genomic segments (intervals) and their orientation with respect to each other. We call the two intervals *left* and *right* intervals according to their genomic coordinates. We also define three types of orientation as shown in Figure 1(b), namely, normal, inverted and swapped. Reads and read-pairs with inverted and swapped orientations are considered to be discordant, and suggesting a breakpoint. These discordant

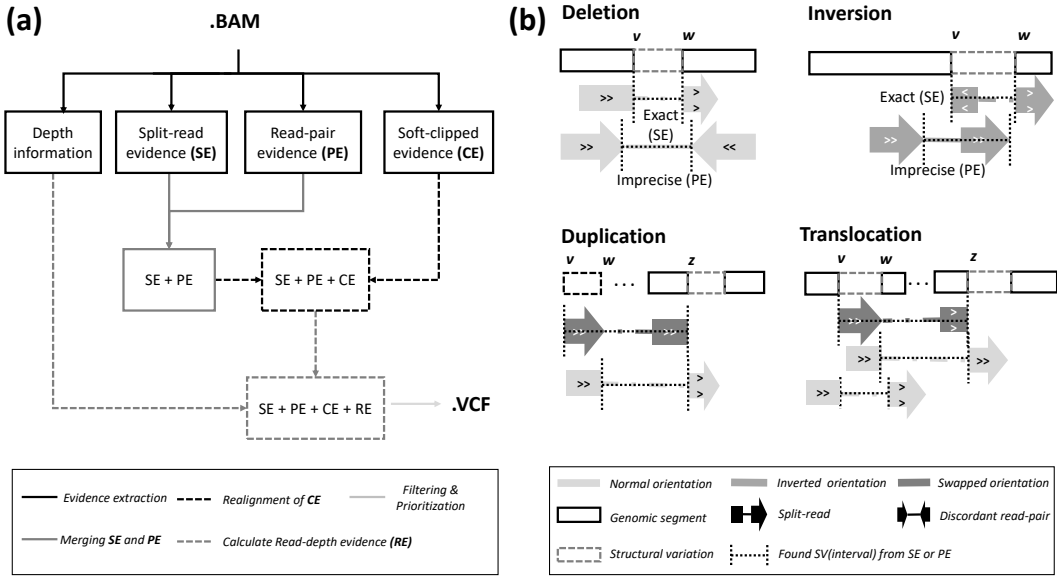

Figure 1 (a) The overall process of Vaquita. (b) The four types of structural variations. The dotted rectangles colored with gray denote the modification of the reference genome. The deletion and inversion in the figures show structural variations from v to w. The duplication and translocation in the figures illustrate copied or moved segments from v to w, and z is the target position.

reads also constitute of split-read evidence (SE) and read-pair evidence (PE), which are the number of reads and read-pairs that support a breakpoint, respectively. For read-pairs with normal orientation, we estimate m and s which denote the median and median absolute deviation of insertion size distribution in a sequencing dataset. Then we apply a cutoff that is $m + s \times 9$ by default. The adjustment of this cutoff affects the accuracy of the result. A weak cutoff yields a sensitive result, but at the cost of specificity. The value 9 is empirically decided after testing values from 5 to 10 (data not shown). Note that this default value is the same as Delly2 and was used in large scale studies such as 1000 Genome Project [21]. We define four SV types, namely *deletion, inversion, duplication*, and *translocation* that are illustrated in Figure 1(b). We identify deletions and inversions from breakpoints with normal and inverted orientation, respectively. The two types of breakpoints are required to find duplications and translocations. The definition is based on previous studies [22, 19]. Nevertheless, there are alternative definitions of SVs which are not always mutually exclusive. Other types of SVs can be identified by the user from the reported breakpoints.

2.2 Candidate merging: SE + PE

Two breakpoints with the same orientation can be merged if both the left and right intervals are adjacent or overlapping. A distance of 50 bases is set by default in assessing adjacency. When two breakpoints are merged, the minimum and maximum positions of each left and right intervals are selected to define the merged breakpoint. The original positions are kept in a list, and the median positions are reported as final positions in the last step. We merge all the breakpoints identified by SE or PE according to this principle. For efficiency, the reference genome is divided into equally sized regions that are 1000 bp by default. The *left* and *right* intervals of SVs belong to one or more regions according to their size and genomic coordinates. The entire merging process can be efficiently done by identifying breakpoints in

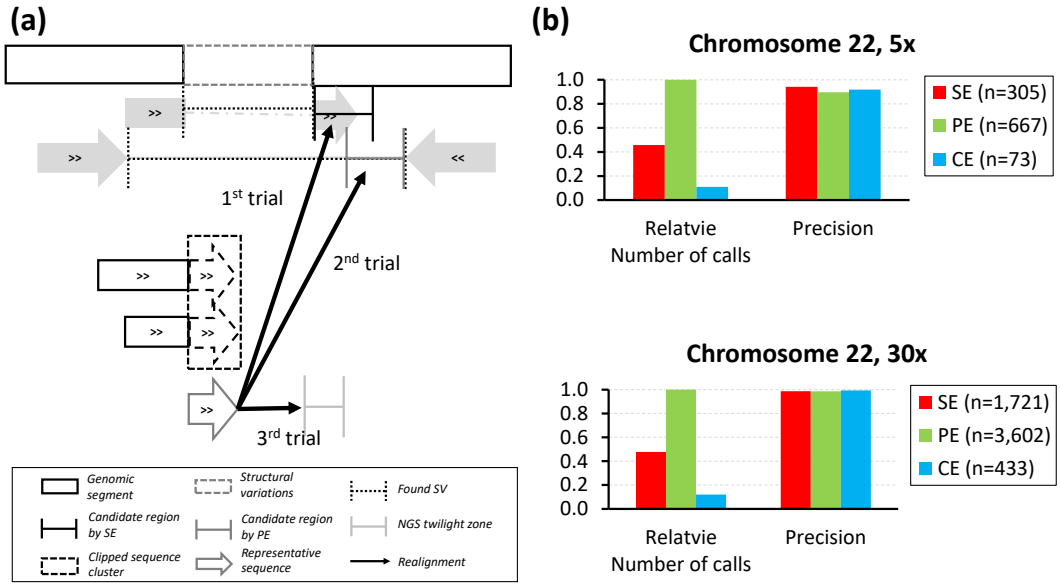

Figure 2 (a) The realignment process for soft-clipped evidence (CE). (b) The relative number of structural variation calls and precision of each evidence types in simulation datasets.

the same region. In the worst case, all the SVs are distinct and fall into the same region. In this scenario, the comparison step takes $O(n^2)$ where n is the number of SVs. However, in practice, the distribution of SVs are sparse, and only a small number of SVs are expected to co-exist in a single region. If all the SVs exist in unique regions, the time complexity is $O(n)$ since only one hashing is required. This process is conducted in parallel while decompressing a `.bam` file, which is usually an I/O bound process.

2.3 Realignment of soft-clipped reads: SE + PE + CE

Mapping the clipped part of sequences is challenging because they are short and erroneous. One can assemble longer contigs to map them correctly and uniquely. However, the entire process is computationally expensive. Instead, Vaquita selects a representative sequence without assembly and reduces the search space by surveying only the pre-selected regions. Initially, Vaquita identifies clusters of soft-clipped sequences according to the genomic coordinates of their mapped parts. Subsequently, it locates the longest unmapped sequence in a cluster and uses it as a representative sequence. Then, it tries to map those representative sequences to candidate regions identified by SE or PE using a fast bit-vector algorithm for approximate string matching [17], using lenient criteria. The time complexity of the algorithm is $O(nr/k)$ where n and r are the sizes of the read and the reference, and k is the word size of the machine which is 64 in modern hardwares including ours. Often, relatively small deletions are difficult to find using read-pair information because the sizes of SVs fall within the variance of the insertion size. This region has been defined as the NGS twilight zone [22]. To address this, Vaquita also examines the genomic sequences around the clipped position for queries that failed the mapping. The size of the additional searching region is set to $m + s \times 9$ by default, where m and s are median and median absolute deviation of the insertion size distribution. The default value is based on criterion for identifying discordant read-pairs in Section 2.1. Only soft-clipped parts that are equal to or longer than

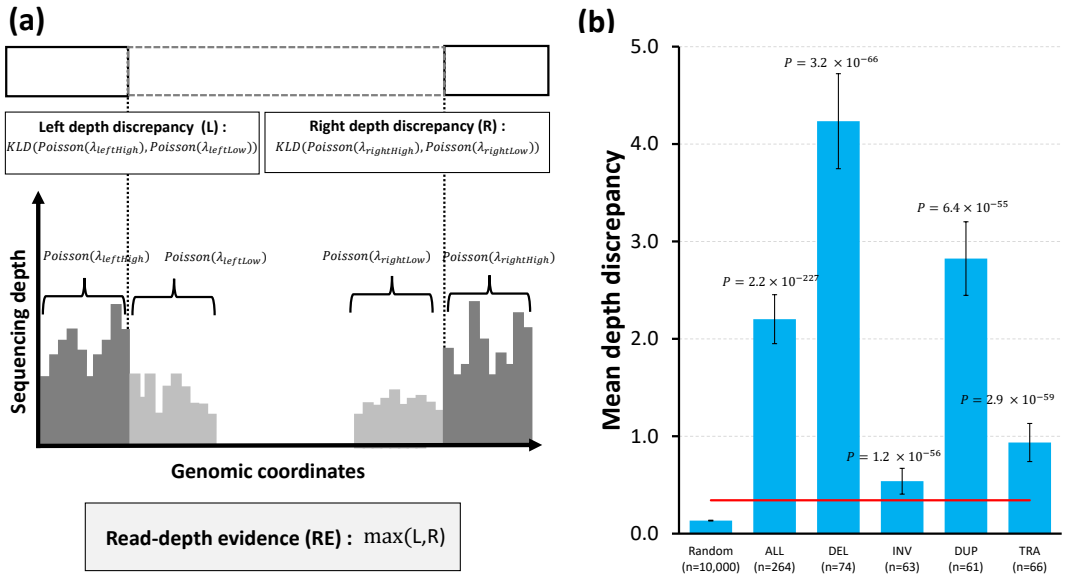

Figure 3 (a)The read-depth evidence. (b) The depth discrepancy distribution of random positions and four types of structural variations in the simulation dataset (Chromosome 22 and 30x coverage). P indicates p-values obtained by two-tailed *Kolmogorov-Smirnov test* using a random sample. The red line shows the third quartile plus the interquartile range of the random sample.

20 nucleotides undergo realignment, allowing edit distance of 10% of the sequence size by default. The overall process is described in Figure 3(a). In the two simulation datasets, the quantity of CE was more than 20% of SE, and 10% of PE. Furthermore, the precision of CE is about the same as SE and PE, as shown in Figure 2(b). Note that Mason [6] we used in the simulation selects random positions to introduce variations. This limitation usually makes simulation tests less challenging since SVs are not randomly distributed. For example, many of the SVs are found in repeat-rich regions in real datasets. The higher precisions reported in Figure 2(b) reflect this limitation.

2.4 Calculation of depth discrepancy

The depth distribution of a sequencing sample has been previously reported to be non-uniform. Hence, the distributions around two randomly picked positions that are not adjacent to each other are likely to be different. We use this observation to discriminate true breakpoints from false positives, especially for unbalanced structural variations. For two genomic intervals i_1 and i_2, we calculate λ_1 and λ_2 that are the mean depth of each interval. We then assume that the local distributions follow a *Poisson* distribution and calculate the *Kullback-Leibler divergence (KLD)* from *Poisson*(λ_1) to *Poisson*(λ_2) as follows:

$$KLD_{\lambda_1,\lambda_2} = \lambda_1 - \lambda_2 + \lambda_1 \log \frac{\lambda_2}{\lambda_1}. \quad (1)$$

We use this as a metric of depth discrepancy between i_1 and i_2 and calculate the read-depth evidence (RE) for each breakpoint as described in Figure 3(a). As a rule, we always calculate the divergence from the higher depth region to the lower depth region and select the larger value between the discrepancies of the left and right side of the breakpoint. We use the window size of 20 bases for calculating local coverages. To efficiently calculate RE for all

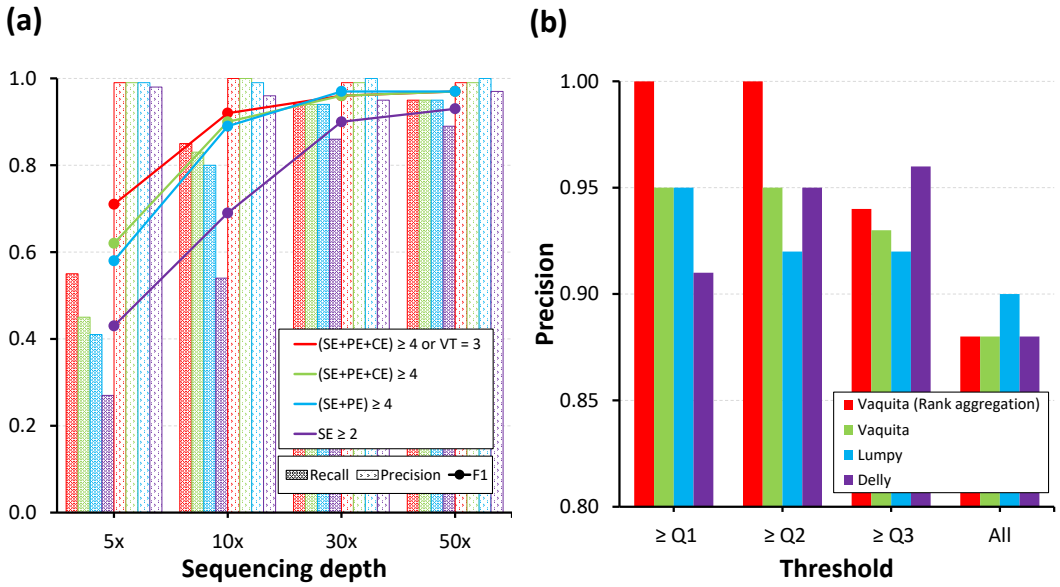

Figure 4 (a) The impact of evidence integration on breakpoint identification. The criterion VT=3 rescues SVs that are supported by all three evidence types as described in Section 2.5.2. (b) Prioritization performance of deletion calling in the 30x simulation dataset. Q1-Q3 indicates the first to the third quartile.

breakpoints, we maintain a lookup table regarding various λ_1 and λ_2. In the simulation datasets, the depth discrepancy distributions were significantly different from random samples as shown in Figure 3(b). As expected, RE is more effective in discriminating unbalanced structural variations like deletions and duplications than balanced structural variations such as inversions. By default, we do not use RE for inversions. However, one can turn on this option.

2.5 Combined evidence

2.5.1 SE + PE + CE

We assessed the impact of evidence integration in breakpoint identification starting from the SE only case. The result is shown in Figure 4(a). We combined the number of split-reads and read-pairs that supported a breakpoint or an SV and used it as a cutoff. We applied a cutoff of 4 as we explained the reason in Section 3.2. However, we had to apply a cutoff of 2 for SE only case since it was too stringent when using a single evidence type in low-coverage samples. Our experiment showed that more accurate results were achieved when additional types of information were considered. The impact is more dramatic in datasets with low-coverage. In the 5x dataset, we obtained the F1 score of 0.62 using the evidence from SE+PE+CE and only 0.43 using SE only. The effect is less pronounced in high-coverage datasets. For the 50x dataset we obtained 0.97 and 0.93, respectively.

2.5.2 Voting based metric for candidate selection

Often, variant callers such as Delly2 and LumpyExpress apply basic filtration using a sum of split-reads and read-pairs that support SVs. Instead of using a simple sum of signals from

different types of evidence, Vaquita provides an additional metric for candidate selection based on voting. In this scheme, each type of evidence for a breakpoint is checked by a relatively lenient cutoff, and then we calculate the number of evidence types that pass the criteria that we denote as VT. For example, a structural variation with $VT = 3$ is supported by three evidence types. By default, we used ≥ 1 for SE and PE since this is the most lenient condition. For RE, we used $\geq (Q3 + IQR \times 1.0)$ where $Q3$ and IQR denote the third quartile and the interquartile range of depth discrepancy score from random positions. The red line in Figure 3(b) shows this default RE cutoff in a simulation dataset. Note that we add CE to SE to treat them as a single evidence type. In Figure 4(a), we applied $VT = 3$ as an additional criterion to rescue SV candidates in low-coverage samples and obtained better recalls without reducing precision. Therefore, we used this option by default for later analyses. One can also use this metric to filter out false positives at repeat-rich regions, instead of excluding those regions from the analysis.

2.5.3 Prioritization by rank aggregation

In practice, prioritization of SVs is an important task for downstream analysis. We formulate this problem to find an aggregated rank from the ranks based on multiple evidence types. At first, we define the goodness of a rank based on *Spearman's footrule distance* [3]. It is given as follows:

$$F(\phi) = \sum_{e}^{\Phi} \sum_{s}^{S} |\phi_e(s) - \phi(s)| \tag{2}$$

where $\phi_x(i)$ is the rank of the element i by the evidence type x, $\Phi = \{SE, PE, RE\}$, and S is the set of all structural variation candidates. In this scheme, the optimal rank ϕ^* is the one that minimize F. We also define the median rank ϕ^M that is defined as follows:

$$\phi^M(s) = median(\phi_{SE}(s), \phi_{PE}(s), \phi_{RE}(s)) \,. \tag{3}$$

The cost of calculating ϕ^M is $O(|\Phi| \cdot |S| \log |S|)$ using a quick sort. Hence, it is applicable to large datasets. Furthermore, $\phi^M = \phi^*$ if there is no tie [4] and, in the presence of ties, ϕ^M is still a 3-approximate solution of ϕ^* [5]. Hence, we calculated ϕ^M instead of ϕ^* for rank aggregation. To prevent arbitrary breaking up of ties, we obtain the ϕ using two different criteria. At first, we order the candidates according to the strength of the evidence, for example, the number of split-reads for SE. Secondly, we use the depth around the breakpoints as a tie breaker. In this scheme, the candidate that is in the lower covered region receives the higher rank. The impact of rank aggregation is shown in Figure 4(b). All the parameters including the cutoff value was same as described in Section 3.2. In the figure, the top 50% ($\geq Q2$) of the structural variations detected by Vaquita turned out to be true positives after prioritization.

3 Result

3.1 Preliminaries

3.1.1 Datasets and variant callers

We generated a diploid that contains SVs based on chr22 of hg19/GRCh37 using Mason [6]. We set the size range from 30 to 5000 and used simulated rates of 4.0×10^{-6} for indel and 2.0×10^{-6} for inversion, duplication, and translocation, respectively. We also introduced

SNPs and small indels with simulated rates of 2.0×10^{-4} and 4.0×10^{-5} to mimic the natural variation, but these were not the focus of the evaluation. From the SVs introduced in chr22, we generated a simulation dataset using ART [7]. We selected Mason since it can simulate the types of SVs that are defined previously, and ART because it provides simulation profiles for the platforms such as Illumina HiSeq-2500. We simulated Illumina paired-end sequencing data with the HS25 option. The depth, read-length, the mean and standard deviation of insertion sizes are shown in Figure 5. We used the 50x sequencing samples of a trio from the Illumina Platinum Genome, NA12878, NA12891 and NA12892, the accession numbers being ERR194147, ERR194160, and ERR194161, respectively. We also used the validation set from the Genome In A Bottle (GIAB) consortium that was constructed using multiple sequencing platforms, including a long-read technology [18]. We compared the performance of Vaquita with five variant callers relying on different sets of evidence types. Delly2 [19], LumpyExpress [11] and Pindel [24] use split-read and read-pair information. We also considered CREST [23] that uses read-depth and soft-clipped reads information, and GASVPro [20] that uses paired-end and read-depth information. All the reads were aligned to hg19/GRCh37 using BWA-MEM [12] with default parameters. We also used BLAT for CREST and SAMBLASTER for LumpyExpress.

3.1.2 Validation process

We only considered breakpoints and SVs that are ≥ 50 nucleotides in size. The identified breakpoints were considered as true positive if we could find a match in the validation set that had more than 80% of reciprocal overlap. For variant callers that report intervals rather than exact positions like GASVPro, we considered the identified variations as valid if there was a match in the validation set that had both ends within the identified intervals. We used in-house scripts to interpret each of `.vcf` files from different variant callers according to our definition of SVs in Figure 1(b). We also used default parameters for each variant callers and noted for when otherwise.

3.2 Performance comparison using simulation data

The comparison with other variant callers using the simulation datasets is shown in Figure 5. We combined the number of split-reads with read-pairs and applied 4 as the minimum cutoff for all variant callers. We used this single cutoff throughout all the comparison in the manuscript to see the performance in overall and noted for when otherwise. Note that there can be best parameters for each variant callers for each test condition, and sometimes 4 is not always the default value. For example, Lumpy uses 4 while Delly2 uses 3 by default. We also applied a mapping quality cutoff of 20 for when a variant caller supported such functionality. Vaquita included voting based candidates with the default parameters explained in the method section. All the variant callers yielded better accuracy as the depth increases. Notably, Vaquita, LumpyExpress, and Delly2 constantly ranked as the top three. Vaquita clearly outperformed the top three in the 5x and 10x datasets, mainly because of voting based rescue. However, the difference in performance decreased for the 30x and 50x datasets. Although we could not observe significant differences in high-coverage samples, the result of Figure 4(b) suggests that the prioritization performance of Vaquita is better than the others. Note that Delly2 showed the highest precision when using Q3 as the threshold in Figure 4(b), However, the difference between Q1 and Q3 cases was only 0.05. This result can be explained by the limitation of simulation based testing that we mentioned in Section 2.3. However, this pattern is not observed when using real datasets as shown in Figure 6(a). We could

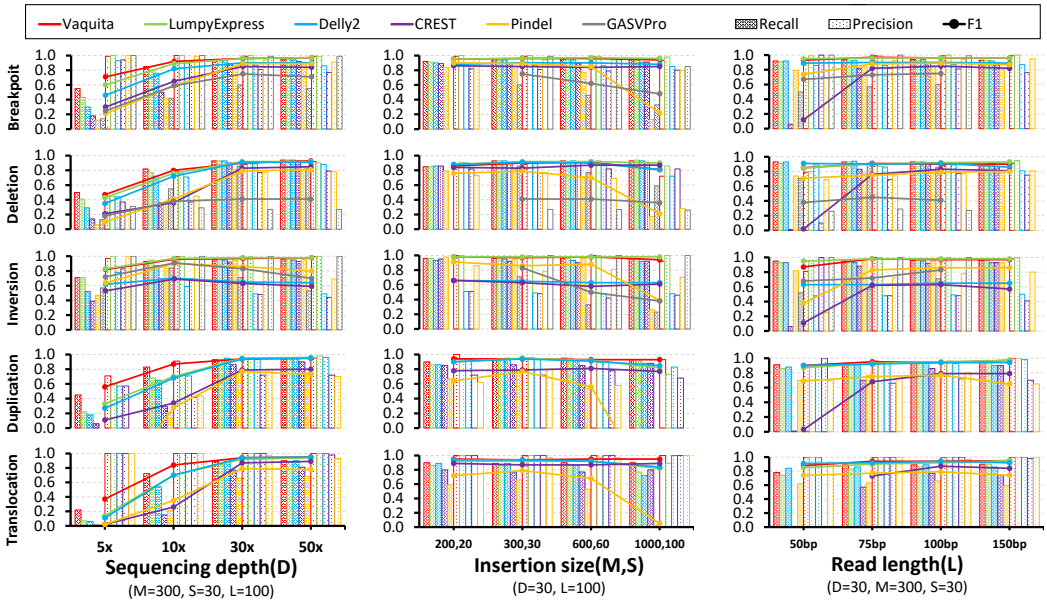

Figure 5 The assessment of accuracy using simulated datasets. M and S indicate the mean and standard deviation of the insertion size.

not observe notable differences from Vaquita, LumpyExpress, Delly2 and CREST from the insert size variations. However, Pindel and GASVPro's accuracy were dramatically reduced in larger insert size samples. Vaquita, LumpyExpress, and Delly2 showed robust accuracy across the read length. However, CREST and Pindel that undergo split-read alignments internally showed poor performance in the samples with read lengths of 50bp. GASVPro was unable to find duplications and translocations (defined according to our criteria), reported alternative SV types, and was unable to run on smaller insert size (200bp) or longer read length (150bp) samples.

From such results, we selected Vaquita, LumpyExpress, and Delly2 as the top three variant callers and their performance was further analyzed using real datasets.

3.3 Performance comparison using real datasets

3.3.1 Overlap between variant callers

The matched fraction of deletions calls compared to GIAB were 0.55, 0.57, and 0.48 for Vaquita, LumpyExpress, and Delly2, respectively. We also investigated the prioritization performance by selecting top 25%, 50%, and 75% of SVs using the rank aggregation for Vaquita, and the evidence summation for LumpyExpress and Delly2 in Figure 6(a). The difference between the overall matching fraction and that of top 25% ($\geq Q1$) were 0.30, 0.26 and 0.19 for Vaquita, LumpyExpress, and Delly2, respectively. This result suggests that the rank aggregation based on multiple evidence types is still effective in this relatively high-coverage samples. In Figure 6(b), Vaquita and Delly2 contain about 28% and 25% of unique breakpoints, while LumpyExpress only has 10%. However, the total number of breakpoints calls were 6,681 and 6,658 for Vaquita and Delly2, while only 5,420 for LumpyExpress. This result suggests that the cutoff used in the comparison are more stringent for LumpyExpress. Notably, LumpyExpress only identified 154 inversions while Vaquita and Delly2 identified 816 and 602 inversions, respectively. One possible explanation is that Vaquita and Delly2

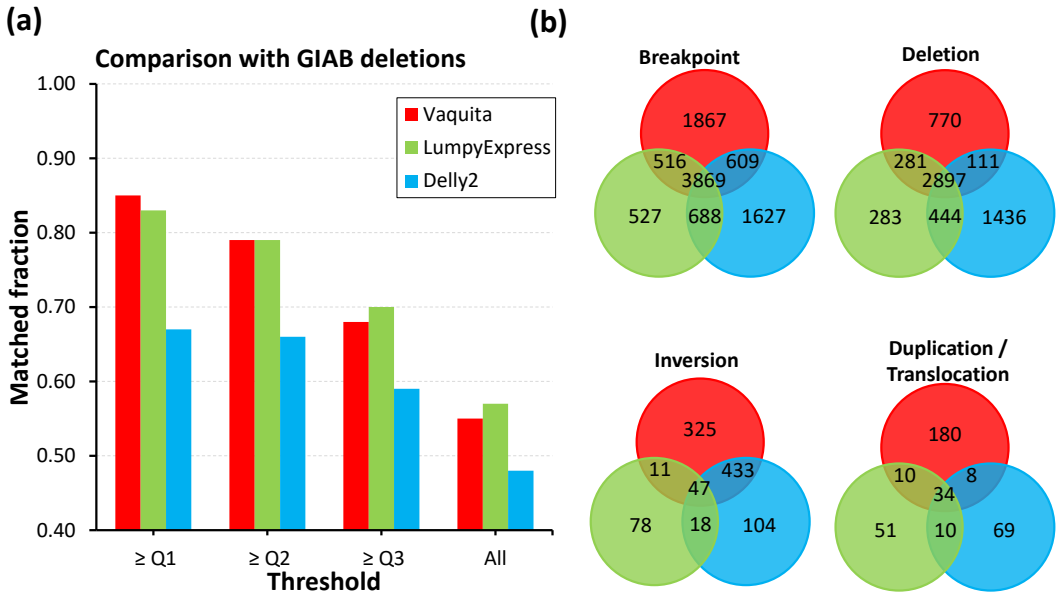

Figure 6 The comparion between variant callers using the NA12878 dataset. (a) Comparison with deletion calls from Genome in a Bottle consortium. (b) The result overlaps between tools.

have an internal realignment process while LumpyExpress is not. However, this explanation is not rigorously tested.

3.3.2 Trio analysis

In Figure 7, the Mendelian errors of breakpoints were 0.23, 0.11, and 0.18 for Vaquita, LumpyExpress, and Delly2, respectively. The error count of LumpyExpress being the lowest can be explained by the high fraction of structural variations that were overlapping in both parents. For LumpyExpress, these fractions were substantially higher than the others, especially in deletions. We additionally investigated the rate of overlaps between two parents and found that LumpyExpress had 0.42 of overlaps in breakpoints while Vaquita and Delly had 0.37 and 0.38, respectively. Note that a recent study using hydatidiform moles and long-reads sequencing technology suggested that 32% of overlaps in deletions and insertions between two genetically unrelated individuals [8]. Therefore, 42% of overlaps between two parents were higher than expected although the reason is not clear. The main difference of the Mendelian errors between Vaquita and Delly2 were due to inverisons. For inverisons, the Mendelian errors were 0.30 for Vaquita, and 0.13 for Delly2. This much of difference is not consistent with the previous analysis using simulation datasets. Therefore, we suspect that several types of inversion couldn't be simulated by Mason properly. However, Vaquita obtained the Mendelian errors of 0.19 and 0.46 for deletions and duplications while Delly2 obtained 0.20 and 0.48, respectively.

3.4 Runtime performance

We compared the CPU time reported by the `time` command in Debian Linux. In the comparison, Vaquita was significantly faster than the other tools for analyzing the human WGS sample with 50x coverage (NA12878). We found that Vaquita is 8.2 times faster than LumpyExpress and 9.6 times faster than Delly2 which only finds one variant type in

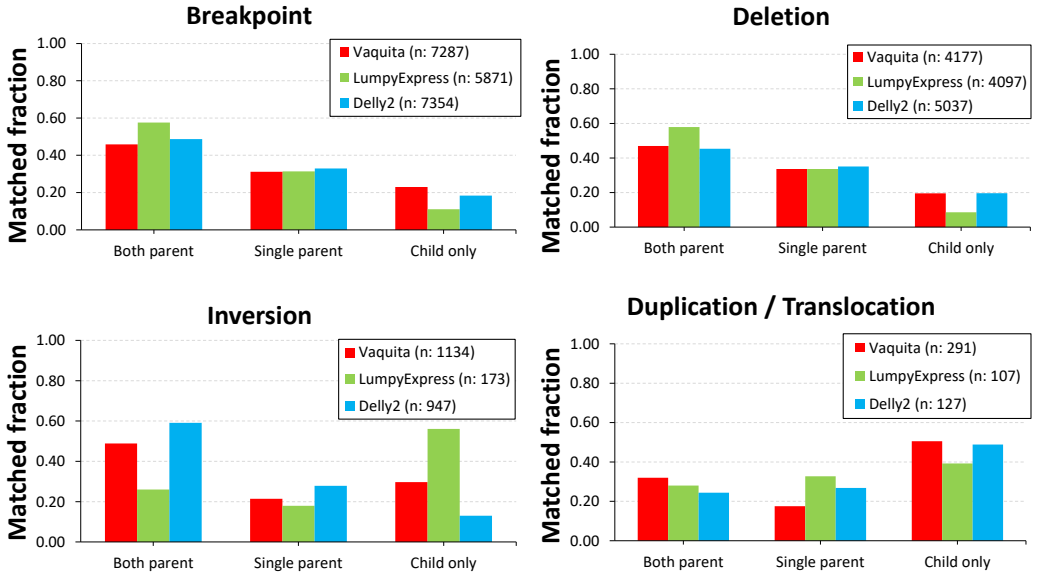

Figure 7 The SVs identified from the child dataset(NA12878) were compared to the SVs from the parent's datasets (NA12891 and NA12892).

a single run. Vaquita only required less than 40 minutes in our test environment. We can explain this speed-up based on three observations. First, the merging step of Vaquita that we explained in Section 2.2 is very fast in practice since structural variations are sparsely distributed across the human genome. Second, the realignment step is also very fast and took less than 5 minutes in total for the test case. Third, the `.bam` file processing of SeqAn library is also faster than other implementation since it internally separates several threads for decompression of `bgzf` stream. Regarding the last reason, we modified the original source code of the library so that the library was fixed to a single thread for the decompression process. Although it is still a separated thread, we used CPU time for the comparison. Note that LumpyExpress calls an external tool for `.bam` file parsing, and Delly2 should be ran multiple times to find all variant types. Therefore, we concluded that the comparison is not specifically biased to Vaquita. The peak memory consumption were 12.5G for Vaquita, 6.4G for LumpyExpress, and 320M for Delly2. This relatively large memory consumption was due to inefficient implementation for storing positions and can be improved in the future version.

All the tests were done on a Debian GNU/Linux 8 machine with two Intel Xeon E5-2667V2 Octa core CPUs at 3.3 GHz, 387 GB of RAM, and 2 TB of SATA SSDs on RAID5 configuration. We did not attempt to use multi-threading for each variant caller.

4 Discussion and conclusion

Vaquita was developed to integrate split-read, read-pair, soft-clipped, and read-depth information and provides effective evidence combination strategy based on voting and rank aggregation. In the benchmark using the simulation datasets, Vaquita showed relatively robust performance across different sequencing depths, insert sizes and read lengths. In the comparison with GIAB deletions, about 55% of deletions found by Vaquita were matched and showed better prioritization results compared to LumpyExpress and Delly2. Vaquita also identified more breakpoints than the others and about 28 percent of them were unique. In

the trio analysis, Vaquita showed similar number of Mendelian errors compared to Delly2 and higher number of errors compared to LumpyExpress. The difference between these errors can be explained by the prevalence of overlapping variations in both parents (LumpyExpress), or by errors in inversions (Delly2). The runtime of Vaquita was significantly faster than those of LumpyExpress and Delly2 by a factor of more than 8 times. As a future goal, we will provide an improved functionality to confidently integrate other orthogonal datasets, including long-read datasets.

Acknowledgements. The authors thank Dr. Birte Kehr for her suggestions regarding validation process and many valuable comments on the manuscript. The authors also thank Dr. Bernhard Renard and Kathrin Trappe for their helpful discussion. J.K was supported by the International Max Planck Research School for Computational Biology and Scientific Computing and the Efficient Algorithms for Omics Data group at the MPI for Molecular Genetics.

References

1 Can Alkan, Bradley P. Coe, and Evan E. Eichler. Genome structural variation discovery and genotyping. *Nature reviews. Genetics*, 12(5):363–376, 2011. arXiv:NIHMS150003, doi:10.1038/nrg2958.

2 Kym M. Boycott, Megan R. Vanstone, Dennis E. Bulman, and Alex E. MacKenzie. Rare-disease genetics in the era of next-generation sequencing: discovery to translation. *Nature reviews. Genetics*, 14(10):681–91, 2013. doi:10.1038/nrg3555.

3 Persi Diaconis. Group representations in probability and statistics. *Lecture Notes-Monograph Series*, 11:i–192, 1988.

4 Cynthia Dwork, Ravi Kumar, Moni Naor, and D Sivakumar. Rank aggregation methods for the Web. *Proceedings of the 10th international conference on World Wide Web*, pages 613–622, 2001. doi:10.1145/371920.372165.

5 Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. Comparing and aggregating rankings with ties. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 47–58, 2004. doi:10.1145/1055558.1055568.

6 Manuel Holtgrewe. Mason – A Read Simulator for Second Generation Sequencing Data. Technical report, Freie Universität Berlin, 2010.

7 Weichun Huang, Leping Li, Jason R. Myers, and Gabor T. Marth. ART: A next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2012. doi:10.1093/bioinformatics/btr708.

8 John Huddleston, Mark Jp Chaisson, Karyn Meltz Steinberg, Wes Warren, Kendra Hoekzema, David S Gordon, Tina A Graves-Lindsay, Katherine M Munson, Zev N Kronenberg, Laura Vives, Paul Peluso, Matthew Boitano, Chen-Shin Chin, Jonas Korlach, Richard K Wilson, and Evan E Eichler. Discovery and genotyping of structural variation from long-read haploid genome sequence data. *Genome research*, page gr.214007.116, 2016. URL: http://www.ncbi.nlm.nih.gov/pubmed/27895111, doi:10.1101/gr.214007.116.

9 W James Kent. BLAT – The BLAST-Like Alignment Tool. *Genome Research*, 12:656–664, 2002. doi:10.1101/gr.229202.

10 Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with Bowtie 2. *Nat Methods*, 9(4):357–359, 2012. arXiv:{\#}14603, doi:10.1038/nmeth.1923.

11 Ryan M. Layer, Colby Chiang, Aaron R. Quinlan, and Ira M. Hall. LUMPY: a probabilistic framework for structural variant discovery. *Genome biology*, 15(6):R84, 2014. arXiv:1210.2342, doi:10.1186/gb-2014-15-6-r84.

12 Heng Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv*, 00(00):3, 2013. URL: http://arxiv.org/abs/1303.3997.

13 Matthew Meyerson, Stacey Gabriel, and Gad Getz. Advances in understanding cancer genomes through second-generation sequencing. *Nature reviews. Genetics*, 11(10):685–96, 2010. doi:10.1038/nrg2841.

14 Alison M Meynert, Morad Ansari, David R FitzPatrick, and Martin S Taylor. Variant detection sensitivity and biases in whole genome and exome sequencing. *BMC bioinformatics*, 15:247, 2014. doi:10.1186/1471-2105-15-247.

15 Ryan E. Mills, Klaudia Walter, Chip Stewart, Robert E. Handsaker, Ken Chen, Can Alkan, Alexej Abyzov, Seungtai Chris Yoon, Kai Ye, R. Keira Cheetham, Asif Chinwalla, Donald F. Conrad, Yutao Fu, Fabian Grubert, Iman Hajirasouliha, Fereydoun Hormozdiari, Lilia M. Iakoucheva, Zamin Iqbal, Shuli Kang, Jeffrey M. Kidd, Miriam K. Konkel, Joshua Korn, Ekta Khurana, Deniz Kural, Hugo Y. K. Lam, Jing Leng, Ruiqiang Li, Yingrui Li, Chang-Yun Lin, Ruibang Luo, Xinmeng Jasmine Mu, James Nemesh, Heather E. Peckham, Tobias Rausch, Aylwyn Scally, Xinghua Shi, Michael P. Stromberg, Adrian M. Stütz, Alexander Eckehart Urban, Jerilyn A. Walker, Jiantao Wu, Yujun Zhang, Zhengdong D. Zhang, Mark A. Batzer, Li Ding, Gabor T. Marth, Gil McVean, Jonathan Sebat, Michael Snyder, Jun Wang, Kenny Ye, Evan E. Eichler, Mark B. Gerstein, Matthew E. Hurles, Charles Lee, Steven A. McCarroll, and Jan O. Korbel. Mapping copy number variation by population-scale genome sequencing. *Nature*, 470(7332):59–65, feb 2011. doi:10.1038/nature09708.

16 Marghoob Mohiyuddin, John C. Mu, Jian Li, Narges Bani Asadi, Mark B. Gerstein, Alexej Abyzov, Wing H. Wong, and Hugo Y K Lam. MetaSV: An accurate and integrative structural-variant caller for next generation sequencing. *Bioinformatics*, 31(16):2741–2744, 2015. doi:10.1093/bioinformatics/btv204.

17 Gene Myers. A fast bit-vector algorithm for approximate string matching based on dynamic programming. *Journal of the ACM*, 46(3):395–415, 1999. doi:10.1145/316542.316550.

18 Hemang Parikh, Marghoob Mohiyuddin, Hugo Y K Lam, Hariharan Iyer, Desu Chen, Mark Pratt, Gabor Bartha, Noah Spies, Wolfgang Losert, Justin M Zook, and Marc Salit. Svclassify: a Method To Establish Benchmark Structural Variant Calls. *BMC genomics*, 17(1):64, 2016. doi:10.1186/s12864-016-2366-2.

19 T. Rausch, T. Zichner, A. Schlattl, A. M. Stutz, V. Benes, and J. O. Korbel. DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, 28(18):i333–i339, 2012. doi:10.1093/bioinformatics/bts378.

20 Suzanne S. Sindi, Selim Onal, Luke Peng, Hsin-Ta Wu, and Benjamin J. Raphael. An integrative probabilistic model for identification of structural variation in sequencing data. *Genome biology*, 13(3):R22, 2012. URL: http://www.ncbi.nlm.nih.gov/pubmed/22452995, doi:10.1186/gb-2012-13-3-r22.

21 Peter H. Sudmant, Tobias Rausch, Eugene J. Gardner, Robert E. Handsaker, Alexej Abyzov, John Huddleston, Yan Zhang, Kai Ye, Goo Jun, Markus Hsi-Yang Fritz, Miriam K. Konkel, Ankit Malhotra, Adrian M. Stütz, Xinghua Shi, Francesco Paolo Casale, Jieming Chen, Fereydoun Hormozdiari, Gargi Dayama, Ken Chen, Maika Malig, Mark J. P. Chaisson, Klaudia Walter, Sascha Meiers, Seva Kashin, Erik Garrison, Adam Auton, Hugo Y. K. Lam, Xinmeng Jasmine Mu, Can Alkan, Danny Antaki, Taejeong Bae, Eliza Cerveira, Peter Chines, Zechen Chong, Laura Clarke, Elif Dal, Li Ding, Sarah Emery, Xian Fan, Madhusudan Gujral, Fatma Kahveci, Jeffrey M. Kidd, Yu Kong, Eric-Wubbo Lameijer, Shane McCarthy, Paul Flicek, Richard A. Gibbs, Gabor Marth, Christopher E. Mason, Androniki Menelaou, Donna M. Muzny, Bradley J. Nelson, Amina Noor, Nicholas F. Parrish, Matthew Pendleton, Andrew Quitadamo, Benjamin Raeder, Eric E. Schadt, Mallory Romanovitch, Andreas Schlattl, Robert Sebra, Andrey A. Shabalin, Andreas Untergasser, Jerilyn A. Walker, Min Wang, Fuli Yu, Chengsheng Zhang, Jing Zhang, Xiangqun Zheng-

Bradley, Wanding Zhou, Thomas Zichner, Jonathan Sebat, Mark A. Batzer, Steven A. McCarroll, Ryan E. Mills, Mark B. Gerstein, Ali Bashir, Oliver Stegle, Scott E. Devine, Charles Lee, Evan E. Eichler, and Jan O. Korbel. An integrated map of structural variation in 2,504 human genomes. *Nature*, 526(7571):75–81, 2015. doi:10.1038/nature15394.

22 Kathrin Trappe, Anne-Katrin Katrin Emde, Hans-Christian Christian Ehrlich, and Knut Reinert. Gustaf: Detecting and correctly classifying SVs in the NGS twilight zone. *Bioinformatics (Oxford, England)*, 30(24):1–8, 2014. doi:10.1093/bioinformatics/btu431.

23 Jianmin Wang, Charles G Mullighan, John Easton, Stefan Roberts, Sue L Heatley, Jing Ma, Michael C Rusch, Ken Chen, Christopher C Harris, Li Ding, Linda Holmfeldt, Debbie Payne-Turner, Xian Fan, Lei Wei, David Zhao, John C Obenauer, Clayton Naeve, Elaine R Mardis, Richard K Wilson, James R Downing, and Jinghui Zhang. CREST maps somatic structural variation in cancer genomes with base-pair resolution. *Nature methods*, 8(8):652–4, 2011. arXiv:NIHMS150003, doi:10.1038/nmeth.1628.

24 Kai Ye, Marcel H. Schulz, Quan Long, Rolf Apweiler, and Zemin Ning. Pindel: A pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, 25(21):2865–2871, 2009. arXiv:NIHMS150003, doi:10.1093/bioinformatics/btp394.

25 Seungtai Yoon, Zhenyu Xuan, Vladimir Makarov, Kenny Ye, and Jonathan Sebat. Sensitive and accurate detection of copy number variants using read depth of coverage. *Genome Research*, 19(9):1586–1592, 2009. doi:10.1101/gr.092981.109.