Proceedings, 10th World Congress of Genetics Applied to Livestock Production

Using the Whole Read: Structural Variant Detection Using NGS Data

D. M. Bickhart[1], J. B. Cole[1], J. L. Hutchison[1], L. Xu[2,3], G. E. Liu[2].
[1]AIPL USDA-ARS, Beltsville, MD, [2]BFGL USDA-ARS, Beltsville, MD, [3]University of MD, College Park, MD

ABSTRACT: Several classes of Structural Variants (SV) remain difficult to detect within sequenced genomes. Deletions and tandem duplications may affect a large proportion of variable genomic sequence space, yet their detection is still difficult to discern from false positive signals. Here, we present a method for detecting such variants from short-read sequence data using the orientation and distance of paired-end, and split-read mappings in addition to using read-depth as a filtering agent. We test our data using simulated SVs and find that our method is 27.5 times more precise than a competing detection program in detecting tandem duplications. Our method is also able to detect three times the number of deletions than a competing algorithm. This high degree of precision should enable better functional prediction of SVs from short-read sequence data.

Keywords: variants; sequencing; software

Introduction

Structural variants (SV) present substantial difficulties in the post-genome world. They have been implicated as the causative agents of several phenotypes such as color-sidedness in cattle (Durkin et al. (2012)) and peacomb in chickens (Wright et al. (2009)); however, their reliable detection requires cutting-edge computational algorithms and extensive molecular validation. Even when such variants are detected, interpretation of their real impact on genome structure is difficult. Much of the difficulty in interpretation is owed to the inexact nature of SV breakpoint detection, which is defined as the exact base pair coordinates where the SV differs from the reference genome assembly. Many algorithms, such as read depth-based CNV detection (Alkan et al. (2009)), attempt to improve SV detection precision by lowering the resolution of detection; however, this prevents reliable breakpoint estimation.

Higher SV breakpoint detection has recently been the subject of extensive research within the genomics community. Much work has been done to utilize short-read library construction techniques, such as paired-end read libraries, to infer the exact breakpoints of SVs in the genome (Korbel et al. (2009)). Additionally, algorithms have been developed to identify SV breakpoints by splitting reads into smaller constituents prior to realignment to the reference genome (Ye et al. (2009)). Such techniques were shown to contribute the highest quality SV predictions in the recent human 1000 genomes project (Mills et al. (2011)). Still, these methods carry the unfortunate side-effects of having high false positive rates due to improper interpretation of read chimeras that result from library creation. We expand on these methods by combining their predictions to generate highly confident SV calls which can then be filtered for improved accuracy. Additionally, we have designed our tool to be used on nearly all reference assemblies by taking into account the uncertain nature of gap regions in our runtime filters. We call our method RPSR as it is a combination of read pair (RP) and split-read (SR) methodologies.

Materials and Methods

Test dataset and discovery. Test data was derived from simulated reads derived from cattle chromosome 29. Fifty sets of simulated reads were generated using wgsim (https://github.com/lh3/wgsim) on a single cattle chromosome that had been modified with deletions and tandem duplications by custom Perl scripts. Wgsim was run with the INDEL rate set to 0% and all other settings at the default. The equivalent of 10X coverage of the genome was generated using wgsim with each simulation. The average sizes of tandem duplications and deletions were approximately 530 bp each, with a minimum size of 55 bp and a maximum of 850 bp. RPSR was implemented in the Java programming language version 1.7. A post-hoc filtering program was written in Perl v5.8.8. All programs and analysis were run on a linux blade server with 24 threads and 100 gigabytes of RAM.

Read alignment and pre-processing. The detection of SVs from paired-end read alignments benefits from the identification of all potential read pair alignment locations and orientations. In order to identify these locations, we used the MrsFAST short-read alignment tool version 2.0.5.4 (Hach, et al. (2010)). MrsFAST identifies all read alignment positions in the reference genome in a cache-oblivious fashion (Hach, et al. (2010)). This has the unintended side-effect of increasing alignment time and alignment file size if repeats are not properly masked in the reference genome, so we used RepeatMasker (Smit, et al. (2008)) on the UMD3.1 cattle reference assembly (Zimin, et al. (2009)) to mask highly repetitive sequence. Average read alignment lengths (A_{rp}) and alignment length standard deviations (σ_{rp}) for each sequencing library were estimated from the alignment of 100,000 sampled reads from that library prior to the alignment of all data. After the determination of A_{rp} and σ_{rp}, all reads were aligned to the reference genome using MrsFAST in single-end mode. We then used a custom Java program to sort read alignments and to attempt to pair reads. Pairs in which one read aligns to the genome and the other does not – also termed one-end anchors (OEA) -- were also saved for further analysis.

Paired-end discordancy analysis. We base much of the core algorithm of our tool on the work of Hormozdiari et al. (2009) in their program VariationHunter-CR. Let F_l and F_r be the leftmost and rightmost mapping coordinates of the first read, respectively, and S_l, S_r be the mapping coordinates of the second read. The orientation of the read is based on the 5' to 3' directionality of the read compared to the reference genome, with a '+' indicating the same directionality and a '-' indicating reverse directionality. Let us define the orientation of the read pair as O, where O is comprised of the following set: {++, +-, -+, --}. The definition of a read pair (P) would therefore include information from all five points of data: P = {(F_l, F_r), (S_l, S_r), O}. The insert length (L) of read pair P, would be equivalent to the distance from the closest read coordinate of the first read and the closest read coordinate of the second read based on their orientation. Concordant reads are reads that do not deviate significantly in insert length (L) or default read orientation (+-) after alignment. Discordant read pairs are defined as the set of P that has one or more of the following characteristics:

1. $L \geq (A_{rp} + 3\sigma_{rp})$
2. $L \leq (A_{rp} - 3\sigma_{rp})$
3. O = (++ or --)
4. O = (-+)

Read pairs that fall within criteria 1 and 2 are indicative of deletions and insertions relative to the reference assembly, respectively. Pairs that have a ++ or – orientation as shown in criteria 3 indicate the edges of inversions of sequence relative to the reference. Finally, pairs with an "everted" orientation as in criteria 4 indicate regions where there may be a tandem duplication. Please see Figure 1 for examples of these criteria and their implications on variant detection. Deviations in insert length (criteria 1 and 2) can mix with abnormal read orientation (criteria 3 and 4) to generate complex variants such as inverted deletions as well. In order to identify variants with confidence, we group discordant read pairs with overlapping coordinates into sets (G). To avoid creating chimeric sets derived from variants present on different chromosomes in the diploid genome, we only collect discordant read pairs that have overlapping read alignment coordinates as shown in Figure 2.

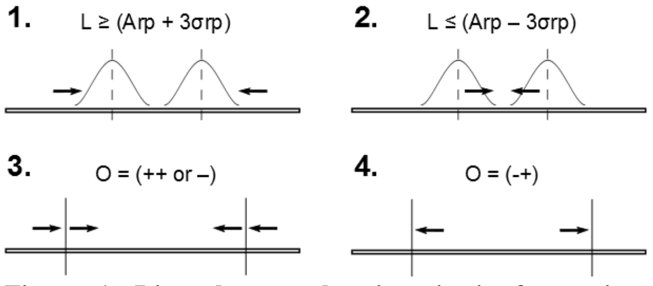

Figure 1. Discordant read pair criteria for variant discovery. Four criteria derived from read alignment orientation (arrows) and size are used to identify putative SVs. Read pair lengths that diverge from the average size distributions in criteria (1) and (2) correspond to deletions and insertions of sequence, respectively. When read pairs show the same alignment orientation (3), they indicate the ends of an inversion event in the genome (solid vertical bars). Conversely, read pairs that show different, abnormal orientations (4), indicate the breakpoints of a tandem duplication (solid vertical bars).

Split-read creation and alignment. Discordant read pairs in our dataset only provide approximate locations for insertion and deletion events based on the criteria we use for alignment. In order to refine our event detection and to find the actual breakpoints of our variants, we incorporate a method known as split-read analysis (Ye, et al. (2009)). Split-read analysis breaks apart reads into shorter fragments, then realigns the reads to identify the breakpoints of sequence variants through discordant mapping. This is, in principle, similar to the methods used to identify discordant read pairs described above. Performing split-read analysis on all reads present in a large dataset is a computationally prohibitive action since it effectively quadruples the amount of time dedicated to read alignment. To reduce the complexity of the analysis, we adopt a method pioneered by Karakoc et al. (2012) which uses only OEA read pairs to pre-select reads for targeted split-read analysis. Since the MrsFAST alignment tool does not perform gapped alignment, OEA reads likely originate from read pairs in which the unmapped read spans a sequence variant.

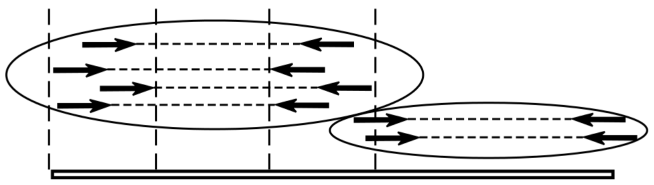

Figure 2. Construction of sets of discordant reads that predict the same SV. In order to differentiate between false positive calls and true positive calls, reads pairs (arrows connected by dotted lines) that support the same variant call are clustered into sets (ellipses). Read pairs must overlap by read fragment position coordinates (vertical dotted lines) on both sides of the alignment to be grouped into sets.

In order to perform the split-read analysis, we divide the unmapped read into two separate half-reads (V_1 and V_2) of equal size and we align the half-reads to the reference. The division point of the read (*I*) is one half of the original read length in all cases. Half-read alignments will fall into two categories based on the location of the variant relative to the location where the half-reads were split. The first category is a balanced (B) split read, where both V_1 and V_2 align to the reference genome. The second category consists of unbalanced (U) split-reads, where only V_1 or V_2 aligns to the reference genome. Balanced split reads indicate that the variant breakpoints exist right at the division point of the read (*I*), whereas unbalanced reads indicate that *I* did not span the exact variant breakpoints (Figure 3). Sets of split reads (H) that predict the same variant are created by grouping B and U reads that cover the same exact breakpoint location. These sets are then combined with prior discordant read sets (G) such that the coordinates of the breakpoints predicted in H are internal to the coordinates identified in each G. With the combination of H and G sets (defined as HG), we have enough information to generate our variant callset.

Weighted set cover and filtration. Because MrsFAST alignment identifies all potential alignments for read pairs, it is necessary to find the best, minimal set of read alignments that detect variants. In order to accomplish this, we use a modification of the set weight cover algorithm that was used by Hormozdiari et al. (2009). Given a collection of subsets of discordant read pairs and split-reads, U = {HG_1, HG_2 … HG_n}, the set weight cover algorithm attempts to find the set that uses the most uncovered elements in a greedy fashion at each iteration (Vazirani (2001)). As such, the HG set that has the highest count of uncovered discordant reads (G) and split-reads (H) is chosen as the first set. In order to account for the bias that may result from sets that derive from repetitive region alignments, we do not use the number of uncovered elements to prioritize set generation under the set weight cover. Instead, we use Hormozdiari et al.'s (2009) phred-based probability (PBP) estimate derived from the quality scores of the read alignments to determine if read alignments originate from alignment mismatches rather than their real mapping locations. Given a series of k mismatches, represented by the set MM = {n_1, n_2… n_k }, in a read, the PBP is determined from the following equation (Hormozdiari et al. (2009)):

$$PBP(MM) = \prod_i \left(\frac{1}{1000} + 10^{-\frac{phred(n_i)}{10}} - \frac{1}{1000} * 10^{-\frac{phred(n_i)}{10}}\right)$$

If a read alignment has no mismatches, PBP = 1. The sum of PBP estimates from uncovered discordant read pairs and split-read pairs is used to select the first set in each iteration of the set weight cover algorithm rather than the raw count of associated read subsets. Unbalanced split read PBPs are divided by 2 in order to account for the loss of a balancing alignment in the probability estimate. After grouping the reads into their appropriate sets, variants are called.

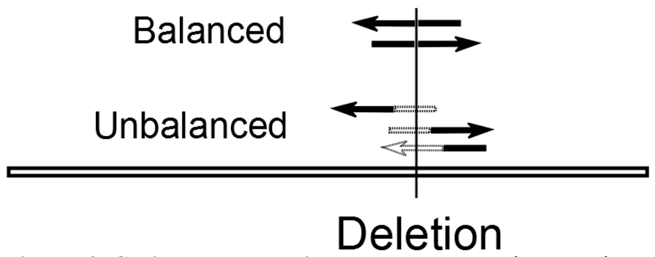

Figure 3. Split read detection strategy. In order to reduce the computational complexity of split read analysis, we split read fragments in half and realign them to the genome. If the variant site (vertical solid line) is positioned right at the center of the read fragment, both split reads align to the genome giving a "balanced" split read alignment (solid arrows and solid bars). If the variant site is off-center, then only one split read aligns, giving an "unbalanced" alignment (solid and empty arrows and bars). Actual variant breakpoints can then be inferred from balanced and unbalanced read positions.

We incorporate four additional filters to reduce the likelihood of false positive calls. First, we remove any sets of HG that contain support only from unbalanced (U) split reads. Such calls likely result from chimeric read fragments or from alignments to repetitive regions of the genome. Additionally, we remove any HG sets that have a cumulative PBP of less than 3 or greater than 10 times the predicted sequence coverage estimate for the library. Calls that have fewer than three reads supporting them are indistinguishable from chimeric read fragments generated during sequencing library creation. Likewise, support from over 10 times the predicted coverage estimate indicates misalignment of a read to repetitive regions of the genome. We also include a filter that removes discordant read pairs that span gaps during the program runtime. Assembly gaps often coincide with highly repetitive regions of the genome that were difficult to resolve during reference genome assembly. While discordant reads that span such gaps could denote actual variants, the uncertainty of reference genome structure in these regions makes such ascertainments difficult. Finally, we use a sum total of aligned reads within the discordant region to filter variants according to expected read depth (RD) profiles. Depending on the variant type and the precision of breakpoint estimates, we expect higher RD values internal to insertions and tandem duplications and lower RD values within predicted deletions. We take the average of read counts per base pair within 100 bases outside of the variant region and internal to the variant region on both sides of the prediction and filter variants that do not have a greater than 75% change in RD profile.

Results and Discussion

Performance and runtime statistics. In order to take advantage of multiple core systems, reads originating from different chromosomes are processed on different threads. The average runtime for a 50 megabase (Mb) chromosome with 10 X coverage was approximately 5 minutes on a single thread. Memory usage was approximately 5 gigabytes on average per thread; however, the memory management model of the Java language often defers garbage collection to improve runtime performance, so it is likely that this memory usage estimate is inflated by deferred collection of dereferenced classes (www.oracle.com/technetwork/java/index-138747.html). Memory usage increases with the number of discordant and unmapped reads in the dataset, with each new discordant read pair occupying > 104 bytes and each pair of split reads occupying > 132 bytes of memory. Improvements to program structure could reduce the total memory footprint of these classes of reads. Additionally, a limit to the number of supporting discordant reads that are stored in each set could allow analysis of higher coverage data without extensive memory consumption. The runtime of the program should theoretically scale well with additional processor cores given sufficient memory overhead.

Simulations and comparison with existing tools. In order to gauge the predictive power of our method, we created simulations of the smallest cattle autosome (chromosome 29) with randomly distributed, artificial SVs. SV types were limited to deletions and tandem duplications so as to provide direct comparisons to the predictive power of the Delly SV detection suite version 0.0.9 (Rausch, et al. (2012)). Simulations were repeated 50 times, with an average of 24 SVs (12.29 deletions and 11.83 tandem duplications) per dataset. Reads were aligned with BWA version 0.6.2-r126 for Delly/Duppy detection and MrsFAST for RPSR. Delly and Duppy were run with default settings, and RPSR output was filtered using two types of progressive filters. The first RPSR filter, called the "P" filter, simply removes calls that have a cumulative PBP score of 3 or less, or over 10 times the expected coverage of the library. The last RPSR filter, or the "RD" filter, checks the read depth profile near the breakpoints of the variant call to see if a change in read depth accompanies the variant. If the RD profile does not show substantial changes, the variant is filtered. Given that the simulation tracks the locations of randomly generated SVs, we were able to estimate average precision and recall values for each method (Table 1).

Table 1. Average recall and precision of RPSR and Delly/Duppy on 50 simulated autosomes with structural variants

Program	TP	Calls	Precis	Recall
DELLY_DELS	0.979	95.813	0.009	0.083
DUPPY_TAND	6.625	366.375	0.018	0.571
RPSR_TAND	5.750	51.354	0.106	0.503
RPSR_DELS	3.083	303.375	0.010	0.245
RPSR_P_DELS	0.125	128.688	0.001	0.010
RPSR_P_TAND	5.438	11.250	0.495	0.491
RPSR_RD_DELS	2.625	16.708	0.245	0.209
RPSR_RD_TAND	2.833	6.479	0.415	0.237
RPSR_PRD_TAND	1.229	1.458	0.653	0.102
RPSR_PRD_DELS	0.083	4.854	0.046	0.007

All numbers are averages from 50 simulations. Simulations had an average of 12.3 deletions (DELS) and 11.8 tandem duplications (TAND). Precision (Precis) is defined as the count of true positives (TP) divided by the total number of calls by the program (Calls). Recall is defined as the number of TPs divided by the known number of positives (12.3 or 11.8 for DELS and TAND, respectively). RPSR variants were subjected to two different types of filters ("P" and "RD") in an alternating and combined ("PRD") fashion.

We found that RPSR has better predictive power than Delly and Duppy in our simulation dataset. Unfiltered RPSR deletion calls show an improved recall (24.5%) rate when compared to Delly deletion calls (0.08%). Duppy shows a slight advantage in terms of recall rate (57.1%) for tandem duplications compared to our unfiltered RPSR tandem duplication calls (50.2%); however, the precision of our method (10.5%) is greater than that of the Duppy method (0.02%). Post-hoc filters showed a significant improvement in the precision of our datasets without a substantial loss in recall rate. Based on our simulation, the simple P filter provided the best precision and recall for the RPSR tandem duplication algorithm with 49.5% precision and 49.1% recall. By contrast, the RD filter seemed to provide the best return for the RPSR deletion algorithm with a 24.5% precision and 20.9% recall. Since the deletions that were simulated in our method were the equivalent of homozygous deletions, this may be the primary reason why the RD filter served as the best setting for the deletion detection algorithm.

Several key differences in algorithms contribute to the substantial variation in precision and recall between Delly/Duppy and RPSR. The first, and perhaps foremost, difference is in the alignment stage. BWA provides a "best hit" alignment that does not consider all possible read mapping locations. While this method works very well at reducing the number of potential discordant reads to analyze, it unfortunately discards true positive discordant reads that map in multiple locations in the genome. MrsFAST, by contrast, outputs all discordant read locations, thereby allowing the RPSR algorithm the liberty of clustering read pairs into predicted SVs. Another key difference between algorithms is how split-reads are prepared and used. Delly reduces the complexity of split-read analysis by searching for OEAs within SV intervals predicted by discordant paired end reads. While this is a useful strategy for breakpoint detection, there is the potential for Delly to miss SVs that only have evidence from split-reads. RPSR calculates split-read estimates separately from discordant reads, thereby identifying SVs that only have such information.

Future Directions. While RPSR shows substantial improvements in SV detection over a similar competitor, improvements to runtime performance and a reduction in generated meta-data are future goals for the program's development. Additionally, we plan on adding larger scale SV detection features in the near future. The set weight cover algorithm should be suitable to cluster additional SV events such as trans-chromosomal and inter-chromosomal translocation. Finally, the post-hoc filters will be incorporated into the runtime component of the algorithm to improve detection specificity.

Conclusion

We present the RPSR program which implements a combined SV detection algorithm that uses discordant read pairs, split-reads and read depth to identify moderate size insertions, deletions and tandem duplications within paired-end sequence data. RPSR runs efficiently on multiple-core systems and it makes use of several cutting-edge techniques to reduce the computational complexity of split-read analysis. We have shown that RPSR outperforms a similar program in simulations, with an increased precision and recall after post-hoc filtration. We plan to release RPSR as an open-source software package for use on the command line for Windows and Unix-based systems.

Literature Cited

Alkan, C., Kidd, J. M., Eichler, E. E., et al. (2009). Nature Genetics. Aug 30. 41 1061- 1067.

Durkin, K., Coppierers, W., Charlier, C., et al. (2012). Nature. Feb 1;482(7383):81-84.

Hach, F., Hormozdiari, F., Alkan, C., et al. (2010). Nat Methods. Aug;7(8):576-577.

Hormozdiari, F., Alkan, C., Eichler, E. E., et al. (2009). Genome Res. 19:1270-1278.

Karakoc, E., Alkan, C., O'Roak, B.J., et al. (2011). Nat Methods. Dec 18;9(2):176-178.

Korbel, J. O., Abyzov, A., Gerstein, M. B., et al. (2009). Genome Biol. Feb 23. 10:R23.

Mills, R. E., Walter, K., 1000 Genomes Project, et al. (2011). Nature. Feb 3;470(7332):59-65.

Rausch, T., Zichner, T., Schlattl, A., et al. (2012). Bioinformatics. Sep 15;28(18):i333-i339.

Smit, A.F.A., Hubley, R., (2008) RepeatMasker Open-3.0. www.repeatmasker.org.

Vazirani, V. V. (2001). Approximation Algorithms. Springer.

Wright, D., Boije, H., Andersson, L., et al. (2009). PLoS Genet. Jun;5(6).

Ye, K., Shulz, M. H., Long, Q., et al. (2009). Bioinformatics. Jun 26. 25;21:2865-2871.

Zimin, A. V., Delcher, A. L., Salzberg, S. L., et al. (2009). Genome Biol. 10(4):R42.