Bioinformatics, 31(13), 2015, 2084–2090
doi: 10.1093/bioinformatics/btv086
Advance Access Publication Date: 16 February 2015
Original Paper

Sequence analysis

RAPTR-SV: a hybrid method for the detection of structural variants

Derek M. Bickhart[1,*], Jana L. Hutchison[1], Lingyang Xu[2], Robert D. Schnabel[3], Jeremy F. Taylor[3], James M. Reecy[4], Steven Schroeder[1], Curt P. Van Tassell[1], Tad S. Sonstegard[1] and George E. Liu[1]

[1]Animal Genomics and Improvement Laboratory, ARS, USDA, Beltsville, MD 20705, USA, [2]Department of Animal and Avian Sciences, University of Maryland, College Park, MD 20705, USA, [3]Division of Animal Sciences, University of Missouri, Columbia, MO 65211, USA and [4]Department of Animal Science, Iowa State University, Ames, IA 50011, USA

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on September 23, 2014; revised on January 27, 2015; accepted on February 9, 2015

Abstract

Motivation: Identification of structural variants (SVs) in sequence data results in a large number of false positive calls using existing software, which overburdens subsequent validation.

Results: Simulations using RAPTR-SV and other, similar algorithms for SV detection revealed that RAPTR-SV had superior sensitivity and precision, as it recovered 66.4% of simulated tandem duplications with a precision of 99.2%. When compared with calls made by Delly and LUMPY on available datasets from the 1000 genomes project, RAPTR-SV showed superior sensitivity for tandem duplications, as it identified 2-fold more duplications than Delly, while making ~85% fewer duplication predictions.

Availability and implementation: RAPTR-SV is written in Java and uses new features in the collections framework in the latest release of the Java version 8 language specifications. A compiled version of the software, instructions for usage and test results files are available on the GitHub repository page: https://github.com/njdbickhart/RAPTR-SV.

Contact: derek.bickhart@ars.usda.gov

1 Introduction

Among one of the larger classes of heritable genetic mutations, structural variants (SVs) are difficult to detect within data derived from current high throughput sequencing technologies. SVs have been implicated as the causative agents of several phenotypes in animal species such as color-sidedness in cattle (Durkin *et al.*, 2012) and peacomb in chickens (Wright *et al.*, 2009); however, their reliable detection from next-generation sequencing data requires cutting-edge computational algorithms and extensive molecular validation. Much of the need for validation stems from the high false discovery rates (FDRs) of several popular SV callers, several of which have been shown to have a FDR of 90% (Mills *et al.*, 2011). Additionally, the exact nucleotide breakpoints of SV events are difficult to detect from sequence data using existing methods. Many algorithms, such as read depth-based copy number variation (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 resolution 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 as was implemented in the program PEMER (Korbel *et al.*, 2009). One such algorithm, implemented in the program PINDEL, splits reads into smaller constituents prior to realignment to the reference genome to find the precise breakpoints of smaller events (Ye *et al.*, 2009). These two techniques were shown to contribute the highest quality SV predictions in the recent human 1000 genomes project (Mills *et al.*, 2011). Still, these methods are highly prone to false positive SV call predictions due to chimeric read fragments and repetitive sequence misalignment. We expand on these methods by combining their predictions to generate highly confident SV calls, which can be filtered at runtime for improved accuracy. Such a strategy can be considered a 'hybrid' of the split read and paired-end algorithms, and has previously been implemented in software such as Delly (Rausch *et al.*, 2012) and Lumpy-SV (Layer *et al.*, 2014). We have also designed our tool to be used on non-model organism reference assemblies by taking into account the uncertain nature of gap regions in our runtime filters. We call our method RAPTR-SV which embodies a combination of read pair (RP) and split-read (SR) methodologies.

2 System and methods

2.1 Simulated dataset

Test data were derived from simulated reads based on sequence from cattle chromosome 29, extracted from the UMD3.1 reference assembly (Zimin *et al.*, 2009). In order to create simulation data, 50 replicates of chromosome 29 were generated with simulated, homozygous, tandem duplications and deletions inserted into the chromosome at random. The maximum number of SVs per replicate was set to 50 non-overlapping events. Fifty sets of simulated reads were generated using wgsim (https://github.com/lh3/wgsim) for each simulated chromosome. Wgsim was run with the INDEL rate set to 0% and all other settings at the default (500 bp insert size with a 50 bp standard deviation in insert length). The equivalent of 10× average sequence coverage of the chromosome was generated using wgsim within each simulation. The average sizes of tandem duplications and deletions were ∼2305 and 2281 bp, respectively. The largest simulated deletion was 42 782 bp in size, whereas the largest tandem duplicate was 33 201 bp. All programs and analyses were run on a linux blade server with 24 threads and 100 GB of RAM.

2.2 Real datasets and benchmarking

Data were derived from two sources: (1) the NA12878 BAM files released by the 1000 genomes project and (2) fastq files from an Angus bull sequenced to 20× coverage. The first dataset (1) represents an excellent 'truth' dataset with which to assess the accuracy of the program on real, validated data provided by the research community. The second dataset (2) provides a contrast by demonstrating the application of our method to a non-model organism's raw data. Binary alignment map (BAM) files for NA12878 were downloaded from the NCBI mirror of the human 1000 genomes FTP site (ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/technical/working/20101201_cg_NA12878/NA12878.hiseq.wgs.bwa.raw.bam). This alignment file contained reads originally used in the first Genome Analysis Toolkit publication (DePristo *et al.*, 2011), so this dataset is both highly validated and serves as an excellent test case of data derived from an older Illumina chemistry and older read alignment algorithms. Summary statistical tests run on the BAM index files showed that the total fold coverage of the dataset was ∼90× distributed among eight separate read groups. True positive variant locations were downloaded from the 'golden standard SV sets' from the human 1000 genomes SV paper (Mills *et al.*, 2011; Supplementary Table S5). We note that the coordinates for these variants and the BAM file are on the hg18 human reference genome, and to avoid issues with coordinate liftover we did not convert the coordinates to the hg19 assembly. In order to provide a non-model organism test dataset, we used 20× coverage sequence data derived from an Angus bull. Reads that did not pass the chastity filter were removed from the dataset, and the 3′ terminal ends of each read were trimmed by up to two bases if the bases were flagged as poor quality (ascii character: '!'; phred value: 0). Dataset comparisons were performed by using custom Perl scripts and through the use of the BedTools software suite (Quinlan and Hall, 2010). Simulation results, results from the Angus bull dataset and the custom scripts used to process the data can be obtained from a subfolder of our GitHub repository: https://github.com/njdbickhart/RAPTR-SV/tree/master/test.

2.3 Read alignment and pre-processing

RAPTR-SV can be used to identify SVs within BWA-aligned, SAMPE BAM files with or without read groups. Two features of the BWA aligner that make it amenable for use with RAPTR-SV are that it lists unmapped reads in the alignment file, and the fact that it outputs soft-clipped alignments. Both of these types of data entries are used to populate potential split read sites for the subsequent clustering algorithm. The detection of SVs from paired-end and SR read alignments benefits from the identification of all potential RP alignment locations and orientations. To identify these locations, we wrap the MrsFAST short-read alignment tool version 2.0.5.4 (Hach *et al.*, 2010) in our 'preprocess' pipeline. 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 (http://www.repeatmasker.org/) on the UMD3.1 cattle reference assembly (Zimin *et al.*, 2009) and hg18 to mask highly repetitive sequence. We downloaded a version of the hg18, human reference genome from the UCSC genome browser (http://hgdownload.soe.ucsc.edu/downloads.html). Average read alignment lengths (A_{rp}) and alignment length standard deviations (σ_{rp}) for each read group ID were estimated from the alignment of 10 000 sampled reads from that ID using the RAPTR-SV 'preprocess' mode.

3 Algorithm

3.1 Paired-end discordancy analysis

Our software uses an expanded algorithm for paired end discordancy first used by Hormozdiari *et al.* in their publication of the software, VariationHunter-CR (Hormozdiari *et al.*, 2009). Let F_l and F_r be the leftmost and rightmost map coordinates of the first read, respectively, and let S_l and S_r be the map coordinates of the second read. The orientation of the read is based on the 5′ to 3′ directionality of the read compared with the reference genome, with a '+' indicating the same directionality and a '−' indicating reverse directionality. Now define the orientation of the sequential reads as O, where O is comprised of the following set: {++, +−, −+, −−}. A RP (P) would therefore include information from all five data points: $P = \{(F_l, F_r), (S_l, S_r), O\}$. The insert length (L) of RP (P), is equivalent to the distance from the closest read coordinate of the first read to 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 (+−)

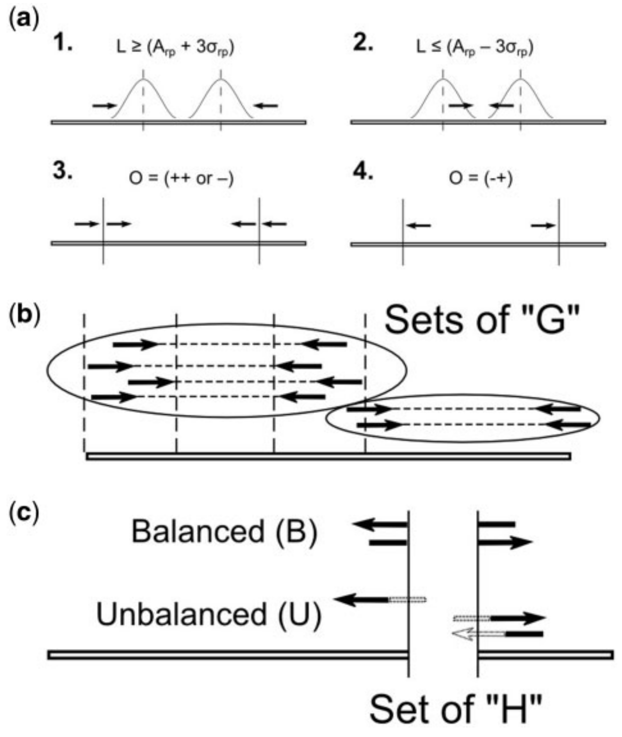

Fig. 1. Visual representation of the algorithm. Key criteria for discordant RP identification (**a**) consist primarily of read orientation and length deviations. Deletions (**a.1**) can be identified by RPs that align to the reference genome further apart than expected, whereas insertions (**a.2**) are pairs that align far closer than expected. Small inversions (**a.3**) are identified by RPs that have the same orientation for both mates and tandem duplications (**a.4**) are identified by RPs that have everted mate orientations. Evidence for a single event can be grouped into sets of 'G' (**b**) by stringently ensuring that overlapping discordant RPs with the same orientation (designated by dashed lines) are placed in the same set. Split reads (**c**) can be quickly assigned to a set of 'H' by looking for complete (balanced) and one-end unmapped (unbalanced) alignments of split pairs. Balanced events (B) denote the exact breakpoints of a deletion; however, unbalanced events (U) are the most likely outcome of our split read alignment strategy and can still be used to refine event breakpoints

after alignment. Discordant RPs is defined as the set of P that has one or more of the following characteristics (Fig. 1a):

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

RPs that fall under criteria 1 or 2 are indicative of deletions and insertions relative to the reference assembly, respectively. Pairs that possess a ++ or − orientation and meeting criterion three indicate the edges of inversions of sequence relative to the reference. Finally, pairs with an 'everted' orientation as in criterion four indicate regions where there may be a tandem duplication. Figure 1a demonstrates examples of these criteria and their implications for variant detection. Complex SVs can be detected as a result of deviations of several criteria by a single P (e.g. $L > A_{rp} + 3\sigma_{rp}$ and $O = ++$). To identify variants with confidence, we group discordant RPs with overlapping coordinates into sets (G). Sensitivity in variant identification is achieved by grouping RPs only when they directly overlap in map coordinates for their reads (F_l, F_r, S_l, S_r) and only when the union of the map coordinates of both P do not expand beyond the insert length (i.e. $P_1 \cup P_2 < A_{rp}$) as shown in Figure 1b.

3.2 SR creation and alignment

In addition to discordant read analysis, we incorporate a method known as 'SR' to identify SV break points (Ye *et al.*, 2009). SR analysis of 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 adopted a method used by Karakoc *et al.* which selects only one-end anchored (OEA) RPs to pre-select reads for targeted SR analysis (Karakoc *et al.*, 2012). We have expanded this algorithm by identifying targets from extensive soft-clipping of reads where one read still maps unambiguously to the reference genome. We define extensive soft-clipping as reads that have >25% of their bases identified as being soft-clipped near the 5′ or 3′ terminal ends of the read.

To perform the SR analysis, we divide the unmapped read into two separate half-reads (V_1 and V_2) of equal size and we use MrsFast alignment (Hach *et al.*, 2010) to place the half-reads on the reference genome assembly. The division point of the soft-clipped or unmapped read is one half of the original read length in all cases. SR alignments will fall into two categories based on the location of the variant relative to the location, where the SRs 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) SRs, where only one read (V_1 or V_2) map to the reference. Balanced SRs indicate that the variant breakpoints exist exactly at the division point of the read; whereas, unbalanced reads indicate that the SR breakpoint did not span the exact variant breakpoints (Fig. 1c). Sets of SRs (H) that predict the same variant are created by grouping B and U reads that cover the same breakpoint location. These sets are then combined with prior discordant read sets (G) such that the coordinates of the breakpoints predicted in H are consistent with the SV call made by the set of G. Additionally, any set of H that is not contained within a set of G is still retained by our algorithm as a valid set for analysis, and *vice versa* for any set of G that does not contain an internal set of H. This inclusive algorithm represents a contrast to other hybrid signal detection strategies, such as those found in Delly (Rausch *et al.*, 2012). The combination of H and G sets (HG) comprise candidate SV calls.

3.3 Weighted set cover and filtration

Because MrsFAST alignment identifies all potential alignments for each P, it is necessary to find the best, minimal set of read alignments that form a set of HG. Any P that is shared by more than one set of HG should, realistically, be found only in one of those sets. To accomplish this, we use a modification of the set weight cover algorithm. Any P that is not currently assigned to a single set of HG is considered to be 'uncovered'. Given a collection of sets of discordant RPs and SRs, $V = \{HG_1, HG_2 \ldots HG_n\}$, 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). To account for the bias that may result from the incorporation of sets that arise 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 phred-based probability (PBP) estimate, which gives the probability that read alignments originate from alignment mismatches rather than their true map 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):

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

If a read alignment has no mismatches, PBP = 1. The sum of PBP estimates from uncovered discordant RPs, and the anchors of SR pairs are used to select the first set in each iteration of the set weight cover algorithm. Unbalanced split read PBPs are currently not used in the weight estimation as such alignments have a high probability of being categorized as false positive calls without additional supporting reads. After grouping the reads into their appropriate sets, each set should constitute a unique variant call.

We incorporate four additional filters to reduce the likelihood of false positive calls. First, we remove any sets from 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 allow the user to define a threshold for the number of raw supporting reads that is necessary to call a set. Calls that have only one read supporting them are indistinguishable from chimeric read fragments generated during sequencing library creation, so the default setting of RAPTR-SV is to filter sets that have only one supporting read. We also include a filter that removes discordant RPs that span gaps in the assembly during the program runtime. Assembly gaps often coincide with highly repetitive regions of the genome that were difficult to resolve during reference genome assembly. Although discordant reads that span such gaps could denote actual variants, the uncertainty of reference genome structure in these regions confounds subsequent validation. This gap filtration utility is packaged as a separate application programmer interface (API) in our BedUtils library (https://github.com/njdbickhart/BedUtils). Finally, we also allow the user to set a threshold for the minimum summed PBP value for a retained set. This filter serves to eliminate many spurious alignments that result from repetitive region alignments as being reported as discordant read sets. RAPTR-SV defaults to a minimum of one summed PBP value for a set to be considered valid for further analysis; however, the user is encouraged to increase this threshold depending on the fold coverage of their input dataset.

4 Discussion

4.1 Performance and runtime statistics

To take advantage of multiple core systems, we have incorporated into RAPTR-SV several functional programming tools included in the latest jdk version 8 release. This enables us to use both ‘divide and conquer’ and ‘map reduce’ frameworks for some of the more computationally intensive sections of the algorithm. We have included a compatibility release of RAPTR-SV that will run on the jdk version 7; however, it does not currently possess the same threading potential as the main release. The average runtime for a 50 megabase (Mb) chromosome with 10× coverage was ~15 min on a single thread, excluding prior BWA alignment time. This time and resource estimate includes both the ‘preprocess’ and ‘cluster’ modes of the program. Memory usage was ~1 GB, on average, per thread. In order to profile resource usage for more common test case usages in ‘cluster’ mode, we sampled reads from a 1000 genomes reference dataset (NA12878; for further details see methods). Test conditions included combinations of a large chromosome (human chr1; 247 Mb), a small chromosome (human chr22; ~50 Mb), 90× coverage and 10× coverage. We found that CPU usage scaled based on the number of potential variant sets discovered during the ‘preprocessing’ step (see additional file: performance_statistics.xlsx). There was a linear trend when larger datasets were considered, suggesting that the initial input stage of the ‘cluster’ algorithm took a larger proportion of runtime for smaller datasets. Memory usage was difficult to predict as we suspect that the Java virtual machine’s (JVM’s) garbage collection algorithm acted responsively to our larger data tests and did not need to act to recover memory in our smaller tests. Our largest trial (247 Mb chromosome; single thread; 90× coverage) utilized 14 GB of RAM at peak usage. The smallest trial dataset (50 Mb chromosome; single thread; 10× coverage) used 5 GB at peak.

To limit the memory overhead incurred by analyzing supporting reads, RAPTR-SV has been designed to allow the user to specify how many supporting RPs for each set are held in memory prior to spilling to disk. Program runtime should theoretically scale well with additional processor cores given sufficient memory overhead and suitable raid-storage. We have included several command line options to allow the user to scale the program’s resource consumption against its potential performance profile. We present an example workflow in Figure 2 that shows the necessary prerequisite files and the two modes of operation of the main program. The ‘preprocess’ mode generates the necessary metadata files from input BAM files, whereas the ‘cluster’ mode generates discordant read sets and makes the final SV calls. There is the potential to use an alternative preprocessing step, such as Samblaster (https://github.com/GregoryFaust/samblaster), to generate preliminary data to be used in the RAPTR-SV ‘cluster’ mode; however, the RAPTR-SV ‘preprocess’ mode is not dependent on BWA-MEM alignment annotation and can be used on a wide variety of ‘legacy’ BWA alignment annotations. In tests using the NA12878 dataset, we found that Samblaster recovered 14-fold fewer discordant RPs than did the RAPTR-SV ‘preprocess’ mode, likely due to the former’s reliance on BWA-MEM annotations in the BAM files (test script: associateDiscordantReadsSamDivet.pl). We would like to reiterate that the NA12878 dataset was aligned with the BWA ALN algorithm, so such a discrepancy in detection between the two algorithms may not apply to datasets aligned with BWA-MEM.

4.2 Simulations and comparison to existing tools

To finely tune the detection of SVs, we created simulated SVs using the current cattle reference assembly sequence from the smallest cattle chromosome, chromosome 29. Simulated SVs were limited to deletions and tandem duplications to provide direct comparisons to predictions made by the Delly/Duppy SV detection suite version 0.0.9 (Rausch *et al.*, 2012) and Lumpy-SV version 0.2.6 (Layer *et al.*, 2014). Simulations were repeated 50 times, with an average of 42 SVs (21.23 deletions and 20.84 tandem duplications) per simulated dataset. Reads were aligned with BWA version 0.6.2-r126 prior to being used by all three detection programs. Delly and Duppy were run with default settings, Lumpy-SV was run with the recommended settings (https://github.com/arq5x/lumpy-sv/blob/master/README.rst), and the RAPTR-SV preprocess mode was run on 15 threads with an initial read sample size limit of 100 000 reads. Delly/Duppy can improve the accuracy of SV breakpoint detection by utilizing a reference genome fasta file to align split reads within identified SV candidates. We provided the original reference genome sequence for chromosome 29 to Delly/Duppy to improve their predictions. Similarly, Lumpy-SV can utilize SR mappings from other software to improve prediction, so we used the split_unmapped_to_fasta.pl Perl script included in the Lumpy-SV package to select

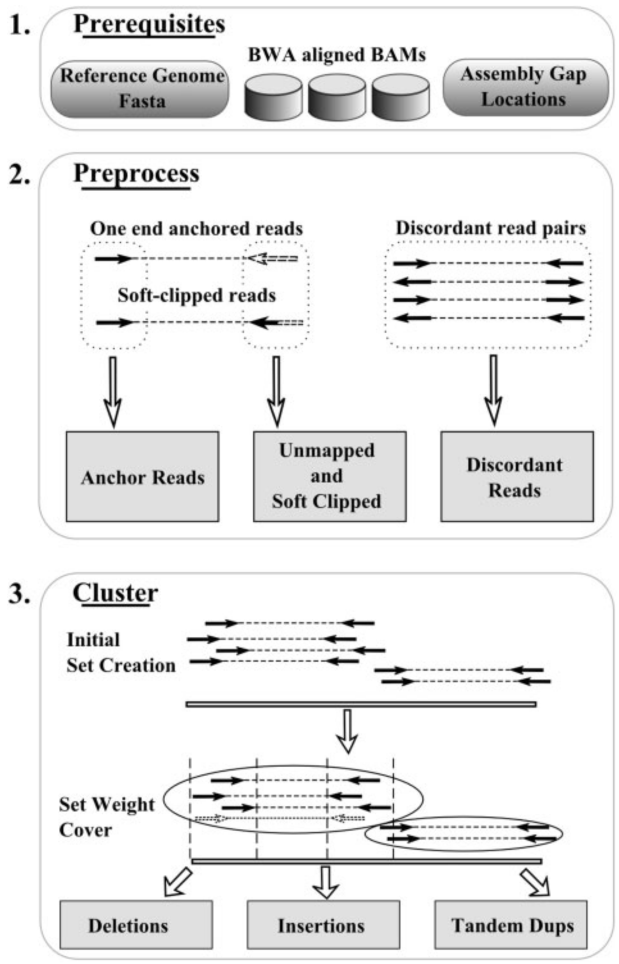

Fig. 2. RAPTR-SV workflow and prerequisites. To generate the necessary metadata, RAPTR-SV requires the user to provide a repeat-masked reference genome fasta file, the location of all assembly gaps (in bed file format) and a BAM file aligned with BWA. The first step of the workflow is the 'preprocess' stage, where RAPTR-SV scans the alignments for one-end anchors, soft-clipped reads and discordant read pairs. These variant reads are stored in three metadata files containing the read anchors, split reads and discordant reads, respectively. The user then uses the 'cluster' mode of RAPTR-SV to process these metadata. First, all metadata are read and assigned to sets. The sets are then refined using a set weight cover algorithm, with the remaining set data sorted into deletion, insertion and tandem duplication calls

putative split read candidates, and realigned these reads using BWA-SW to generate SR alignments. We tested several minimum weight (mw) settings from Lumpy-SV, and found that the recommended mw value of 4 provided good sensitivity without sacrificing precision (data not shown). Output from the RAPTR-SV preprocess mode was used in the RAPTR-SV cluster mode with a raw read filter of 2 and a Probability-based Phred filter of 4.0. These filters effectively eliminated all call sets that had fewer than two supporting reads and poor read mapping scores (due to ambiguous read alignments). RAPTR-SV identified insertions were first merged into non-redundant sets and then combined with RAPTR-SV tandem duplication calls prior to comparisons with Delly/Duppy. Several soft-clipped bases from BWA alignments caused the strict set generation criteria of RAPTR-SV to identify the same smaller insertions multiple times, which necessitated this merger. Given that the simulation tracks the locations of the randomly generated SVs, we were able to estimate average precision and sensitivity values for each method (Table 1). Precision was calculated by dividing the average number of true positive calls by the average number of calls made by the program. Sensitivity was estimated by dividing the average number of true positive calls by the average number of simulated SVs in the dataset.

Table 1. Performance of Delly/Duppy and RAPTR-SV on fifty simulated SV datasets

Program	True positives[i]	Total calls[i]	Precis (%)[t]	Sens (%)[b]
DELLY	7.41 (2.2)	31.29 (42.1)	23.7	34.9
DUPPY	13.08 (4.1)	18.92 (40.3)	69.1	62.8
LUMPY DELS	6.60 (2.6)	**14.72 (5.3)**	**44.8**	31.1
LUMPY DUPS	11.70 (3.5)	**11.70 (3.5)**	**100**	56.1
RAPTR-SV DELS	**7.88 (2.4)**	26.77 (5.1)	29.4	**37.1**
RAPTR-SV TAND	**13.84 (3.8)**	13.94 (3.7)	99.3	**66.4**

[i]Numbers indicate the average count of calls from 50 simulations. Standard deviation in parenthesis. Bold highlighting indicates the best performance in the category. [t]Precision (Precis) is defined as the average number of true positives divided by the average number of total calls made by the program. [b]Sensitivity (Sens) is defined as the average number of true positives divided by the average number of actual variants in the simulations. The average number of deletions per simulation was 21.23 and the average number of duplications was 20.84.

We found that RAPTR-SV had better predictive power than Delly, Duppy and Lumpy-SV in our simulation dataset. RAPTR-SV had an improved sensitivity for detecting deletions relative to Delly (37.1 and 34.9% sensitivity, respectively) and Lumpy-SV (31.1% sensitivity). Additionally, RAPTR-SV identified the same deletion events with an improved precision compared with Delly (29.4 and 23.7% precision, respectively), mostly as a property of making fewer total calls than Delly (RAPTR-SV average calls per simulation: 26.77; Delly: 31.3). When the standard deviation of call sets was taken into account, Delly had a wider distribution of total calls per simulation ($\sigma = 42.1$) than did RAPTR-SV ($\sigma = 5.1$), which indicates that the total number of calls made by Delly varied greatly per dataset. In contrast, Lumpy-SV had nearly double the precision than both Delly and RAPTR-SV in our deletion call simulation, suggesting that it was far more conservative than the other SV callers. Comparisons of tandem duplications and insertions revealed that RAPTR-SV had an improved sensitivity (66.4%) and precision (99.3%) when compared with Duppy (sensitivity: 62.8%; precision: 69.1%). Lumpy-SV showed a slightly higher precision (100%); however, it was less sensitive than RAPTR-SV (56.1% versus 66.4%, respectively). Although RAPTR-SV missed an average of 44.6% of the simulated tandem duplications, nearly every insertion and tandem duplication call made by the program was a true positive event. We applied a *post-hoc* filter to Delly/Duppy SV calls to see if there was an improvement to algorithm performance similar to that achieved by our use of runtime filters in RAPTR-SV. To mirror the runtime filters of RAPTR-SV, we removed all Delly/Duppy calls that had fewer than 2 supporting reads and had a cumulative PBP score of less than 4.0. Although this *post-hoc* filter greatly increased the precision of Delly and Duppy (100 and 99.3%, respectively), it also decreased the sensitivity of both programs (19.6 and 58.3%, respectively). This filter was not applied to Lumpy-SV, as its duplication calls already had a near perfect precision.

Several key differences between the algorithms contribute to the variation in precision and sensitivity among Delly/Duppy, Lumpy-SV and RAPTR-SV. The first, and perhaps foremost, difference is in the alignment stage of split reads. BWA provides a 'best hit' alignment that does not report all possible read mapping locations.

Although this method works very well at reducing the number of potential discordant reads to be analyzed, it unfortunately discards true positive discordant reads that may map to multiple locations in the genome. MrsFAST, in contrast, outputs all discordant read locations, thereby allowing the RAPTR-SV algorithm the liberty of clustering SRs into predicted SVs. The subsequent set weight cover algorithm then selects for SV calls those that have the best support and removes alternative mappings for reads that have already been used to construct previous SV calls. Another key difference between the algorithms is how SRs are prepared and used. Delly reduces the complexity of SR analysis by searching for OEAs near SV intervals predicted by discordant paired-end reads. Although this is an efficient strategy for breakpoint detection, there is the potential for Delly to miss SVs for which the only evidence is from SRs. RAPTR-SV calculates SR estimates separately from discordant reads, thereby allowing the program to identify SVs, which do not have evidence from discordant RPs. Lumpy-SV does not include a split read alignment algorithm and instead relies on other software to create split read mappings for SV detection. The method used to generate Lumpy-SV split read alignments in this simulation was similar to the general algorithm used by RATPR-SV; however, each split read was assigned a single mapping location by BWA-SW and is therefore subject to the same limitations that are discussed earlier. We would like to note that it is also possible to submit RATPR-SV calls to Lumpy-SV for use in multiple-calling comparisons with other SV detection tools.

4.3 Identification of variants within a sequenced individual from the 1000 genomes project

Sequence data derived from PCR-based library creation methods used in Illumina technologies can often contain several contaminants that confound the accurate detection of SVs. To test our software against such a dataset, we downloaded the aligned reads from one of the CEU trio individuals (NA12878) that served as a gold standard for variant discovery. We used the hg18 reference assembly for all subsequent alignments including the Delly/Duppy split read prediction, and RAPTR-SV and Lumpy-SV split read alignment stages. As with the simulation, Delly/Duppy and Lumpy-SV were run with default/recommended settings. Specifically, Lumpy-SV was run with an mw value of 4 and the split read alignments were generated by the split_unmapped_to_fasta.pl script with the BWA-SW pipeline. RAPTR-SV was run with a raw read filter of 5 and a Prob-based Phred filter of 6.0 for deletions and 2.0 for tandem duplications and insertions. Delly/Duppy did not appear to distinguish between the 16 read groups of the BAM; however, RAPTR-SV calculated separate average and standard deviation values for RP insert lengths for each library. The BAM alignments were derived from two separate libraries (Solexa-18483 and Solexa-18484), which RAPTR-SV predicted had average read insert sizes of 380 and 407 bp with standard deviations of 40 and 43 bp, respectively.

Delly and Duppy predicted several large deletions and duplications that extended over 10 Mb in size. The largest of these variants was 246 Mb in size, which accounted for nearly all of the bases on human chromosome one. The largest events were predicted on the first human chromosome by both programs, with nearly the entirety of the chromosome predicted to be both duplicated and deleted by Duppy and Delly, respectively. Interestingly, the largest duplication (246 735 634 bp) and deletion (241 158 955 bp) predictions on chromosome one had breakpoints that were not near any annotated repetitive elements or assembly gaps, which indicates that large chimeric reads rather than misaligned RPs contributed to these false positive calls. Similarly, Lumpy-SV predicted a large deletion (150 047 106 bp)

Table 2. NA12878 SV call benchmarks

Program	Filtered calls[i]	Sensitivity < median[t] (%)	Sensitivity > median[t] (%)	Sensitivity total (%)
DELLY	1 545 981	59.5	62.3	60.90
DUPPY	155 638	2.0	13.3	7.64
LUMPY DELS	958 275	**65.4**	60.7	**63.0**
LUMPY DUPS	**2132**	**7.3**	14.0	10.6
RAPTR-SV DELS	**477 316**	46.1	**66.7**	55.0
RAPTR-SV TAND	22 546	2.6	**26.0**	**14.2**

[i]All SV calls >2 Mb in length were filtered from each respective dataset. Bold highlighting indicates the best performance for the 'Deletion' and 'Duplication' categories. [t]The median size of deletions and duplications in the NA12878 truth set were determined to be 3441 and 5889 bp, respectively. Sensitivity percentages indicate the number of truth set calls that had a 50% reciprocal overlap with SV calls from the program.

and a large duplication (239 646 682 bp) on chromosome one that did not intersect with known gap or repeat regions. RAPTR-SV did not detect such large variants primarily due to the fact that such events often cross assembly gaps and are filtered prior to SV calling. RAPTR-SV also uses an event size filter to ensure that extremely large events can be appropriately removed from final SV datasets.

After removing events larger than 2 Mb from all three datasets, we estimated the sensitivity rate of each program for known SVs in NA12878 by requiring a reciprocal 50% overlap of the variant call with the boundaries of the known variant site. Sensitivity was then calculated as stated in the simulated dataset. Known variant sites were obtained from the golden standard variant set for NA12878 provided by the Human 1000 genomes SV paper (Mills *et al.*, 2011). All three programs recovered the majority of the validated deletions in NA12878 (Table 2). Delly and Lumpy-SV correctly predicted 44 and 58 more true positive, small deletions than RAPTR-SV, respectively. A closer examination of the data revealed that RAPTR-SV had identified 14 of these sites; however, the breakpoints of these events could not be resolved due to noise in SR alignment in these regions. If the stringency of the overlap conditions were reduced to a reciprocal 25% overlap between the SV call and the known site, RAPTR-SV achieved a total sensitivity rate of 66.7% (data not shown), matching Delly and Lumpy-SV. We noticed a discrepancy in the sizes of known events and their sensitivity rate for each respective program. Known deletion sites that were larger than the median of the truth dataset (3441 bp) were predicted more frequently by RAPTR-SV than by Delly and Lumpy-SV. All of the Delly and Lumpy-SV exclusive deletion calls were below the median size, which indicated that both programs have improved breakpoint precision at this lower size range than does RAPTR-SV. With respect to duplications and insertions, RAPTR-SV correctly predicted twice the number of true positive duplications than did Duppy, and ~30% more true positive duplications than Lumpy-SV. RAPTR-SV was superior to Duppy and Lumpy-SV at predicting these events regardless of SV size, though all three programs struggled to predict smaller insertions and duplications. With sequencing datasets that have high read insert variance and numerous potential chimeric fragments, we therefore recommend using alignment-based INDEL discovery methods for the identification of insertions of sequence <50 bp in length. We have included a list of RAPTR-SV exclusive SVs detected from this dataset that were previously validated as part of the Mills *et al.* golden dataset. In all, 37 previously identified events were identified exclusively by RAPTR-SV (see additional file: RAPTR-SV.exclusive.NA12878.predictions.bed). We have also included some initial comparisons from our use of RAPTR-SV and

Delly on 20× coverage sequence data derived from an Angus bull (see additional files: angus_venn.pdf and angus_test_table.xlsx). This initial comparison showed high overlap of RAPTR-SV calls with Delly (46.1% bp overlap of deletions) and Duppy (73.4% bp overlap of duplications); however, the size of the Delly/Duppy prediction dataset eclipsed that of RAPTR-SV (128 Mb versus 5.56 Mb, respectively). Without experimental validation, it is impossible to predict how many of these discovered variants are true positives; however, the Delly/Duppy dataset predicted that 5% of the cattle genome was variable, which is at least twice the size of previously determined estimates of 1.07–2.45% (Bickhart *et al.*, 2012; Liu *et al.*, 2010; Zhang *et al.*, 2014).

5 Conclusions

We present RAPTR-SV as a precise, tunable and scalable method for the detection of SVs using paired-end sequence data. Groups that are sequencing non-model organisms will likely find the most benefit from our algorithm, though we hope that the entire research community will adopt the tool. Future development goals will be to reduce the memory overhead of the program and to incorporate an algorithm to detect both chromosome translocations as well as read depth signal.

Acknowledgements

We would like to thank Tabatha Cooper for suggesting the name for this algorithm. Mention of trade names or commercial products in this article is solely for the purpose of providing specific information and does not imply recommendation or endorsement by the US Department of Agriculture. The USDA is an equal opportunity provider and employer.

Funding

D.M.B., G.E.L., T.S.S., C.V.T. and S.S. were supported by USDA CRIS project numbers 1265-31000-096-00D and 1265-31000-104-00D. G.E.L. was partially supported by AFRI grants no. 2011-67015-30183 from the USDA NIFA. J.F.T. and R.D.S. were supported by the Agriculture and Food Research Initiative competitive grant nos. 2011-68004-30214, 2011-68004-30367 and 2013-68004-20364 from the USDA National Institute of Food and Agriculture Animal Genome Program.

Conflict of Interest: none declared.

References

Alkan,C. *et al.* (2009) Personalized copy number and segmental duplication maps using next-generation sequencing. *Nat. Genet.*, **41**, 1061–1067.

Bickhart,D.M. *et al.* (2012) Copy number variation of individual cattle genomes using next-generation sequencing. *Genome Res.*, **22**, 778–790.

DePristo,M.A. *et al.* (2011) A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genet.*, **43**, 491–498.

Durkin,K. *et al.* (2012) Serial translocation by means of circular intermediates underlies colour sidedness in cattle. *Nature*, **482**, 81–84.

Hach,F. *et al.* (2010) mrsFAST: a cache-oblivious algorithm for short-read mapping. *Nat. Methods*, **7**, 576–577.

Hormozdiari,F. *et al.* (2009) Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Res.*, **19**, 1270–1278.

Karakoc,E. *et al.* (2012) Detection of structural variants and indels within exome data. *Nat. Methods*, **9**, 176–178.

Korbel,J. *et al.* (2009) PEMer: a computational framework with simulation-based error models for inferring genomic structural variants from massive paired-end sequencing data. *Genome Biol.*, **10**, R23.

Layer,R.M. *et al.* (2014) LUMPY: a probabilistic framework for structural variant discovery. *Genome Biol.*, **15**, R84.

Liu,G.E. *et al.* (2010) Analysis of copy number variations among diverse cattle breeds. *Genome Res.*, **20**, 693–703.

Mills,R.E. *et al.* (2011) Mapping copy number variation by population-scale genome sequencing. *Nature*, **470**, 59–65.

Quinlan,A.R. and Hall,I.M. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinforma. Oxf. Engl.*, **26**, 841–842.

Rausch,T. *et al.* (2012) DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, **28**, i333–i339.

Vazirani,V. (2001) *Approximation Algorithms*. Springer-Verlag, Berlin, Germany.

Wright,D. *et al.* (2009) Copy number variation in intron 1 of SOX5 causes the pea-comb phenotype in chickens. *PLoS Genet.*, **5**, e1000512.

Ye,K. *et al.* (2009) PINDEL: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, **25**, 2865–2871.

Zhang,L. *et al.* (2014) Detection of copy number variations and their effects in Chinese bulls. *BMC Genomics*, **15**, 480.

Zimin,A.V. *et al.* (2009) A whole-genome assembly of the domestic cow, Bos taurus. *Genome Biol.*, **10**, R42.