Bioinformatics, 32(4), 2016, 505–510
doi: 10.1093/bioinformatics/btv621
Advance Access Publication Date: 27 October 2015
Original Paper

OXFORD

Sequence analysis

Detecting dispersed duplications in high-throughput sequencing data using a database-free approach

M. Kroon[1], E.W. Lameijer[1], N. Lakenberg[1], J.Y. Hehir-Kwa[2,3], D.T. Thung[2], P.E. Slagboom[1], J.N. Kok[1] and K. Ye[1,4,*]

[1]Department of Molecular Epidemiology, Leiden University Medical Center, Leiden, [2]Department of Human Genetics, Nijmegen Center for Molecular Life Sciences, Institute for Genetic and Metabolic Disease, Radboud University Nijmegen Medical Center, Nijmegen, [3]Donders Centre for Neuroscience, Nijmegen, The Netherlands and [4]The Genome Institute, Washington University, St Louis, MO 63108, USA

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on July 13, 2015; revised on October 16, 2015; accepted on October 20, 2015

Abstract

Motivation: Dispersed duplications (DDs) such as transposon element insertions and copy number variations are ubiquitous in the human genome. They have attracted the interest of biologists as well as medical researchers due to their role in both evolution and disease. The efforts of discovering DDs in high-throughput sequencing data are currently dominated by database-oriented approaches that require pre-existing knowledge of the DD elements to be detected.

Results: We present DD_DETECTION, a database-free approach to finding DD events in high-throughput sequencing data. DD_DETECTION is able to detect DDs purely from paired-end read alignments. We show in a comparative study that this method is able to compete with database-oriented approaches in recovering validated transposon insertion events. We also experimentally validate the predictions of DD_DETECTION on a human DNA sample, showing that it can find not only duplicated elements present in common databases but also DDs of novel type.

Availability and implementation: The software presented in this article is open source and available from https://bitbucket.org/mkroon/dd_detection

Contact: kye@genome.wustl.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The term dispersed duplication (DD) denotes any DNA sequence duplicated non-locally in a genome. DDs include copies of transposable elements such as members of the *Alu* and L1 families, which are ubiquitous in the human genome, but also less frequent duplications such as chromosomal translocations and copies of mitochondrial DNA sequences embedded in nuclear DNA. DDs are very common, as estimates show that known transposable elements comprise nearly 50% of the human genome (Lander *et al.*, 2001). An arbitrary human sample may contain upwards of 1000 DDs compared to the reference genome (Stewart *et al.*, 2011).

DDs have been found to be disruptive to the genome, altering gene expression and sometimes causing disease (Kazazian *et al.*, 1988; Miki *et al.*, 1996). This type of structural variation has often been considered in cohort studies aiming to link phenotypes to causal variants. Consequently, there has been an increased effort in developing methodologies to uncover genetic variation beyond the single nucleotide level.

The advent of high-throughput DNA sequencing provides a new information source for genetic variant discovery that is both fast and is steadily becoming less expensive. However, whole-genome sequencing output is typically bulky and non-trivial to analyze,

highlighting the need for robust and computationally efficient analysis methods. There is a variety of tools available to find structural variants in sequencing data [e.g. CNVnator (Abyzov *et al.*, 2011), GenomeSTRiP (Handsaker *et al.*, 2011), Pindel (Ye *et al.*, 2009) and BreakDancer (Chen *et al.*, 2009)]. Typically, these tools can be applied in a resequencing setting where a reference genome is available, and the DNA sample to be investigated is sequenced with low to moderate coverage. Short sequencing reads are produced of approximately 30–150 nucleotides long and subsequently aligned to the reference. Current sequencing technology allows the production of *paired-end* reads, where two ends of a larger fragment are sequenced, adding more information about the expected alignment of these reads.

Insertions of transposable elements are one important class of DDs, and most currently available methods detect these types of variation using a predefined database. Some of the known tools that cover this type of variation are RetroSeq (Keane *et al.*, 2013), TE-Locate (Platzer *et al.*, 2012), Tangram (Wu *et al.*, 2014) and Mobster (Thung *et al.*, 2014). The usual strategy for detecting DDs in sequencing data is to look for anomalous read alignments and try to realign the respective reads or read-parts to a database of known duplication sequences. A DD insertion is called when a certain number of aligned reads support the same duplication element type and show consensus on the insertion site.

Another previously published method, named Gustaf (Trappe *et al.*, 2014), does not realign to a database but focuses completely on split read alignments to identify DDs. This study aims to develop a method to find DDs without requiring realignment to a predefined database of known elements. Instead, detecting DDs based on the information provided by discordant alignment of read pairs and partially aligned reads allows our method to find elements that have not been included in available databases or have not even been discovered yet. The method described in this article can be seen as a complement to the existing methodology referenced earlier, as it is applicable to situations where providing a database of known duplication elements is not desirable.

We introduce DD_DETECTION as a method that can be used in a resequencing setting where short, paired-end reads are aligned to a reference genome. The underlying algorithm is implemented with adjustable parameters to allow users to have it perform according to their custom needs and wishes. We compared the sensitivity of DD_DETECTION with database-oriented methods using a human DNA sample for which extensive experimental validation data are available. In addition, we tested the specificity of our method using new human samples and performed experimental validation with Sanger sequencing to find out how well DD_DETECTION could identify both DDs of known structure and novel types.

2 Methods

The algorithm to find DDs uses output from a standard high-throughput paired-end read sequencing system such as the popular Illumina platform. The sequencing output is generated for a, typically large, number of DNA fragments whose *insert sizes* (i.e. fragment lengths) have a mean value of F^{IS}. Of all n fragments, both ends are sequenced, resulting in a set of *reads* $S = \{r_1, \ldots, r_{2n}\}$, where r denotes a DNA sequence of some fixed length F^{RL} with a unique associated name name(r). We define $r_j = \mathtt{mate}(r_i)$ to denote that reads r_i and r_j (where $i \neq j$) are sequenced from the same fragment. After sequencing, all reads are aligned to a reference genome. Our method is designed for use with the BWA alignment tool (Li and Durbin, 2009). In principle, the alignment information can be provided by any existing alignment tool that can handle paired-end reads, such as Bowtie (Langmead *et al.*, 2009) or NovoAlign (http://www.novocraft.com/products/novoalign/). Additional information provided by alignment includes $\mathtt{is\_aligned}(r)$, a Boolean value stating whether the read could be aligned to the reference genome, $\mathtt{chr}(r)$, the chromosome to which the read is aligned, $\mathtt{pos}(r)$, the left-most base position on the chromosome to which the read is aligned and $\mathtt{strand}(r)$, the strand to which the read is aligned.

Given the alignment information as input, the algorithm proceeds in three steps. First, all reads with an abnormal alignment are deemed potentially indicative of a DD insertion. These reads are collected and clustered by strand and position in the genome. Second, for each cluster, a potential breakpoint of a new DD insertion is estimated based on either information from local split reads or, in absence of a split read signal, from the distribution of alignment positions of the reads in the cluster. Finally, estimated breakpoints from opposite strands are combined into duplication events, which are then reported. The process is described below in further detail, a graphical outline is shown in Figure 1.

2.1 Detecting clusters of discordantly mapped reads

In the first step of the algorithm, all read alignments in S are examined. Reads for which the mates align to different chromosomes or for which the distance of the alignment positions exceeds a certain

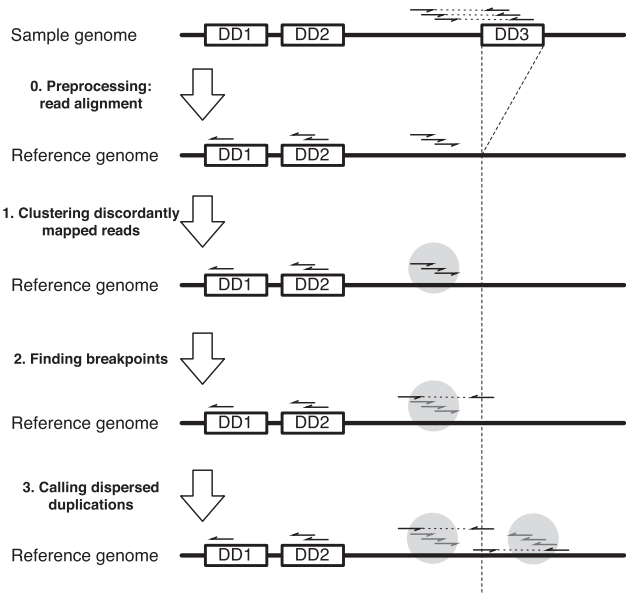


Fig. 1. Method outline for detecting DDs. The top of the figure shows a paired-end sequenced sample genome containing three identical elements *DD1*, *DD2* and *DD3*, where DD3 is not present in the reference genome. In pre-processing step 0, the paired-end reads are aligned to the reference genome. Reads that originate from the region flanking the DD3 element are aligned to the corresponding region in the reference. Reads originating from inside the DD3 element are aligned to DD1 and DD2 in the reference genome. Our method starts with step 1, where reads that are aligned to the region flanking the DD3 insertion site are clustered as a group of discordantly mapped reads (gray circle). In step 2, the method searches a read pair where one read aligns to the flanking region and its mate can be partially aligned adjacent to the assumed insertion site of DD3, giving an estimation of the breakpoint on the reference genome. In the final step, the breakpoint estimation is combined with a breakpoint estimated by a similar process on the opposite strand and the DD event is reported

threshold ($r_i = \text{mate}(r_j)$ and $|\texttt{pos}(r_i) - \texttt{pos}(r_j)| > \theta$) are marked *discordant* and are collected for further processing. The threshold parameter θ ensures that the algorithm focuses on dispersed events rather than small or local variations.

The collected discordantly aligned reads are subsequently clustered by strand and by alignment position. Two reads (r_i, r_j) are assigned to the same cluster $C \subseteq S$ if $\texttt{chr}(r_i) = \texttt{chr}(r_j)$ and $\text{strand}(r_i) = \texttt{strand}(r_j)$ and $|\texttt{pos}(r_i) - \texttt{pos}(r_j)| \leq \delta$. The value of δ determines how close two reads must be aligned to be in the same cluster. δ is manually configurable and allows control over the number of times reads are mistakenly clustered together or conversely assigned to the same cluster while not supporting the same DD event. The expected number of such mistakes also depends on read coverage and on the size characteristics of the paired-end read fragments (F^{IS}, F^{RL}). The algorithm also enforces a maximum on the size of the region that is spanned on the reference genome by its read alignments to avoid adding reads bearing signals from different duplication events to the same cluster. Formally, a cluster C is invalid if

$$\exists_{r_i \in C, r_j \in C} |\texttt{pos}(r_i) - \texttt{pos}(r_j)| > F^{\text{IS}} - F^{\text{RL}}.$$

While clustering, the reads are traversed in order of their alignment position on the genome, assigning them to clusters one at a time. New clusters are created for the first read and every read that cannot be assigned to an existing cluster without invalidating it by the definition described above. Afterward, a noise threshold α is applied such that all clusters of insufficient size $|C| < \alpha$ are discarded.

2.2 Finding breakpoints

Having found clusters of discordant reads, the algorithm estimates for each cluster the breakpoint location of the putative DD. Estimation of the breakpoint is done using a split read signal provided by applying the *pattern growth* algorithm locally. This algorithm was first described by Pei *et al.* (2001) and has successfully been used in finding patterns in biological sequence data (e.g. Ye *et al.*, 2007, 2009; Zhang *et al.*, 2012). In the context of finding DDs, pattern growth is used to find partial alignments of reads (i.e. split reads) overlapping the insertion breakpoint.

To find a breakpoint for a specific cluster C with reads aligned to the forward strand, the left-most alignment position $m = \min_{r \in C} \texttt{pos}(r)$ is determined. Given m, the algorithm finds reads $r_* \in S$ for which $\texttt{mate}(r_*)$ is not aligned or only partially aligned to the reference and $\texttt{pos}(r_*) + F^{\text{RL}}$ is within the range $[m - F^{\text{IS}}, m + 2F^{\text{IS}}]$ (i.e. a region around the far end of the site spanned by the cluster on the reference genome). For every r_* the pattern growth algorithm is applied to $\texttt{mate}(r_*)$ in the local area on the reference genome to find a partial alignment. If such a partial alignment is found, a putative breakpoint is announced at the last aligned position before the read starts to diverge from the reference. Any announced breakpoint is discarded if it is not at least supported by α partial alignments or if the previously unaligned parts of the split reads can be aligned locally in the genome (in a $\pm\theta$ region around the breakpoint). For clusters with reads aligned to the reverse strand of the reference genome, a similar approach is used, the right-most aligned position being found by $m = \max_{r \in C} \texttt{pos}(r) + F^{\text{RL}}$ and using $[m - 2F^{\text{IS}}, m + F^{\text{IS}}]$ as the range that must overlap with $\texttt{pos}(r_*) + F^{\text{RL}}$.

If no breakpoint is found using split reads, it will be estimated directly from the alignments of the reads in the cluster. Given a cluster C with reads $(r_1, \ldots, r_{|C|})$ aligned to the forward strand ordered by alignment position, a breakpoint is estimated by

$$\texttt{pos}(r_{|C|}) + F^{\text{RL}} + \frac{1}{|C| - 1} \sum_{i=1}^{|C|-1} \Big(\texttt{pos}(r_{i+1}) - \texttt{pos}(r_i) \Big).$$

For clusters with reads aligning to the reverse strand, the breakpoint is calculated in similar fashion by subtracting the average distance between reads from the position of the left-most read.

2.3 Calling DD insertions

In the final step of the algorithm, DD events are called if they are supported by breakpoints from two clusters, one with reads aligned to the forward strand and one with reads aligned to the reverse strand of the reference genome. Two such breakpoints are assumed to support the same event if they are not further apart than λ base-pairs on the same chromosome. Specifically, the duplication events are discovered by combining consecutive breakpoints from an array of breakpoints sorted by their position on the genome.

The resulting DD events are reported with the position on the genome estimated from read clusters on both strands, the supporting discordant reads (i.e. those that map inside the element), the supporting split reads and the alignment positions of the discordant reads to the reference genome as provided in the input (i.e. positions indicating alternative copies of the DD elements).

3 Results

We implemented the method presented in the previous section and named it DD_DETECTION. The program is easy to use, configurable and capable of processing whole-genome sequencing input on a modern desktop computer system. For example, on a regular system (4-core 2.4 GHz processor, 8 GB memory) with human whole-genome sequencing data (40× coverage) as input, running DD_DETECTION with four threads and otherwise default parameters takes 127 min and consumes approximately 2.5 GB peak physical (9.3 GB peak virtual) memory. The C++ source of the program together with installation instructions can be found online at https://bitbucket.org/mkroon/dd_detection.

We have used DD_DETECTION to investigate the performance of our method for both medium and high-coverage human DNA alignment data. DD events called on publicly available data from the 1000 Genomes project (The 1000 Genomes Project Consortium, 2012) are used to compare the performance of DD_DETECTION to that of alternative methods. We also applied DD_DETECTION to a two-sample human dataset and validated in corresponding DNA samples a number of calls for both known transposon elements and elements not present in common databases.

3.1 Comparing methods in a human sample

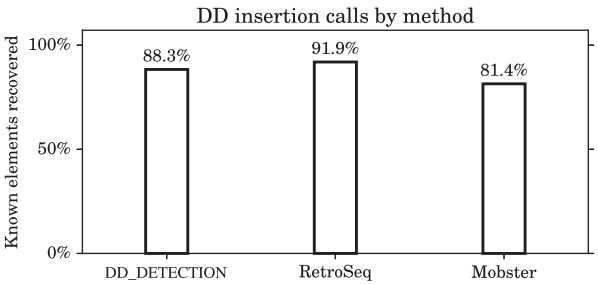
We used a well-studied human sample from the 1000 Genomes project (The 1000 Genomes Project Consortium, 2012) to compare our method against recent methods for finding transposon insertions. The particular public dataset used for this sample (NA12878, ftp://ftp.1000genomes.ebi.ac.uk/Vol03204/ftp/technical/working/20101201_cg_NA12878) contains paired-end reads from the Illumina HiSeq platform for the whole genome with approximately 200× coverage. The read data were re-aligned to the human reference genome GRCh37 using BWA (Li and Durbin, 2009). DD_DETECTION was run with $\theta = 8000$ bp, $\alpha = 10$, $\delta = 250$ bp and compared to two alternative methods: RetroSeq (Keane *et*

al., 2013) and Mobster (Thung *et al*., 2014). Gustaf (Trappe *et al*., 2014) was omitted from the comparison, due to its large memory requirement for whole-genome sequencing data, it could not be applied to this dataset of our system.

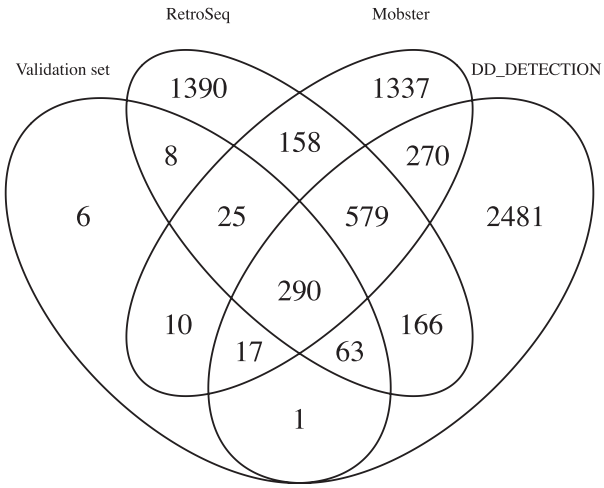
RetroSeq, which is a database-oriented approach that uses detection of discordantly mapped reads and read realignment to find new element insertions, was run with default settings while employing the Repbase Update (Jurka *et al*., 2005) database with selected transposon elements for human and human ancestors. Furthermore, duplicate calls arising from homologous elements in the database were removed from the RetroSeq output.

Mobster also has a database-oriented approach and uses realignment of discordant and split reads to find new element insertions. Mobster was run with default parameters and the transposon database that came supplied with the software, which contained sequences for the L1, *Alu* and SVA families.

Calls made by each method were compared to previously validated transposon insertion events taken from Stewart *et al*. (2011). The validation set contains 420 insertion events of *Alu*, L1 and SVA elements. Figure 2a shows the percentage of validated events recovered by each detection method, as computed by matching calls with validated events within a 250 bp range. All three methods recover a decent number of events in the validation set, with RetroSeq performing best with 91.9% and Mobster performing worst with 81.4%.



(a) Recovery per detection method of known elements in the validation set.



(b) Overlap of calls made by each detection method and validation set.

Fig. 2. Comparison of DD insertion calls made by DD_DETECTION, RetroSeq and Mobster on a deep-sequenced human sample. (**a**) For each detection method, the percentage of known elements recovered from the validation set of 420 *Alu*, L1 and SVA elements provided by (Stewart *et al*., 2011). (**b**) The overlap in calls made by each detection method

Figure 2b shows the overlap within a 250 bp range for the calls made by the three detection methods and the validation set. The majority of validated events overlap with the call sets of all methods. RetroSeq and Mobster show a similar number of unique calls in the 1300–1400 range, while DD_DETECTION has nearly 2500 unique calls. With 10 calls, Mobster has the highest number of calls overlapping with the validation set that are unique to that particular method, RetroSeq and DD_DETECTION follow with 8 and 1, respectively.

Estimated DD insertion locations of the three detection methods were compared with the actual locations in the validation set. For DD_DETECTION, using the mean of the estimations based on both strands, we find an average distance of 37.4 bp (standard deviation of 21.7) between the estimated insertion location and the validated insertion location. When looking solely at the split read signal of the DD_DETECTION calls, the average distance drops to 9.4 bp (standard deviation of 6.9). RetroSeq shows an average distance between estimated and validated locations of 16.4 (standard deviation of 16.4). Mobster shows a distance of 10.6 bp (standard deviation of 22.3) for the same measurement.

3.2 Validation of DDs in medium coverage data

To test the ability of DD_DETECTION to find DDs of known type and DDs with non-database elements, we obtained a two-sample dataset originating from a monozygous twin pair (Ye *et al*., 2013). The two samples named *A* and *C* were sequenced with paired-end reads on an Illumina HiSeq platform with medium coverage (40×). The resulting reads are 100 bp in length and have a mean insert size of approximately 300 bp. The reads were aligned to human reference genome GRCh37 using the alignment tool BWA with default settings. DD_DETECTION was run for each sample separately with the following parameters: $\theta = 8000$ bp, $\alpha = 5$, $\delta = 100$ bp, $\lambda = 100$ bp.

RepeatMasker (Smit *et al*., 1996) was used to classify events by masking the supporting read sequences according to the Repbase Update (Jurka *et al*., 2005) transposon element database. We noticed some events to occur in low-complexity regions of the genome. Therefore, we divided the DD events into three disjoint groups: first, events of *known type*, for which >50% of their supporting reads could be aligned to the consensus sequence of an element in the database. Second, *low complexity* events, for which >50% of the reads were identified as low complexity sequences (including satellite repeats). Last, events classified *non-database* are those that do not fit the previous two descriptions. For these three groups, we found 1825 (74.9%), 433 (17.8%) and 177 (7.27%) events, respectively.

3.2.1 Validation of DDs with common transposon elements

For both samples combined, 1825 of the DD events called were classified as a known type in the RepBase update database. Of those, 1058 (58.0%) were detected in both samples, 329 (18.0%) were uniquely detected in sample A and 438 (24.0%) were uniquely detected in sample C. Please note that the discrepancy in calls between the two twin samples depends on the data coverage {In our experiment with medium coverage data for a human twin pair, the calls made by DD_DETECTION were not identical for the two samples as one might expect. We observed that approximately 42% of the calls were unique to one of the samples, however, of those calls, more than 75% were found in the other sample with support just below the detection threshold $\alpha = 5$ (i.e. the events would have been detected in the other sample with an α value of 3 or 4). These findings suggest that the call sets for the twin samples are more similar than they appear in the listed results.}. Of all 1825 events, we considered

210 (11.5%) to be novel, as they were not reported in previous literature (using Hormozdiari *et al.*, 2011; Lee *et al.*, 2012; Stewart *et al.*, 2011; Wang *et al.*, 2006 as reference). Furthermore, 436 (23.9%) of these events were called with split read support.

DD calls from both the group of novel events and known events were selected randomly for validation. For each selected event, polymerase chain reaction (PCR) was applied to amplify the region of the DNA containing the putative insertion. As the breakpoints may be imprecise, the primers were designed to be at least 150 bp away from the nearest breakpoint estimation. Before validation, the predicted length of the elements to be extracted was estimated from the alignment of the discordant read pairs supporting the corresponding event and from information provided by the Repbase Update transposon database if the particular event could be classified as a known transposon type. The actual length of the inserted elements was determined via gel electrophoresis on the PCR products. If the measured length of a product concurred with the estimated element length, the product was isolated and sequenced inwards from both directions using Sanger sequencing. The resulting sequences were aligned to the human reference genome using BLAT (Kent, 2002). We state that an event is validated when BLAT returns an incomplete alignment near the putative breakpoint with the remaining part of the sequence aligning to the consensus sequence of the element type as identified by RepeatMasker for the respective event or when the rest of the sequence corresponds to the supporting reads of the event and their alignment to the reference genome.

Validation results for nine events with elements from the Repbase Update database are listed in Table 1. Seven of the selected events were validated successfully. Detailed information on the selected events and validation experiments can be found in (Supplementary Table S1). Pictures of electrophoresis on the PCR products are shown in Supplementary Figs S1–S4.

3.2.2 Validation of DD events with non-database elements

DD_DETECTION found 177 events that were classified as neither present in the transposon element database nor arising from regions of low complexity. Fifty-five events (31.1%) were detected in both samples, while 57 (32.2%) events were unique for sample A and 65 (36.7%) events were unique for sample C. Twenty-two (12%) of these DDs were supported by split reads.

Eight events with non-database elements were selected for validation. We preferred events that were well-supported by discordant reads and for which the surrounding region on the genome was suitable for primer design. The followed validation procedure was as described in Section 3.2.1. Table 2 lists the validation result, three of the events were validated successfully. Details on the selected events and validation can be found in Supplementary Table S1 and Figs S1–S4.

Table 1. Validation results for nine DDs with common transposon elements

Chr	Position	Family	Novel	Validated
1	93167510	*Alu*	No	**Yes**
3	103171339	*Alu*	No	No
1	83201797	L1	Yes	**Yes**
1	60470601	*Alu*	No	**Yes**
4	80883489	*Alu*	Yes	**Yes**
1	241908648	*Alu*	Yes	**Yes**
2	30670002	*Alu*	Yes	**Yes**
5	21207706	L1	Yes	**Yes**
5	155290632	*Alu*	Yes	No

Seven events could be validated, of which five were considered novel events not found in previous literature. From left to right, the table columns describe the chromosome and genome position of the event, as averaged by the estimations based on forward and reverse strand. The last three columns contain the family to which the inserted transposon element belongs, whether it is a novel event and whether it was validated successfully.

4 Discussion and conclusion

We presented DD_DETECTION, a new method to detect dispersed duplicated DNA, an important but relatively neglected type of structural variation. The method uses the alignment information of paired-end reads to find non-reference duplications, thereby not relying on a predefined database of duplicated elements. As a software package, DD_DETECTION is publicly available and has few requirements for installation.

In a comparative study on public high-coverage data, DD_DETECTION was able to compete with database-oriented methods in finding new transposon insertions. Analysis of DDs called in medium coverage data shows that DD_DETECTION is able to find both new transposon insertions of known type as well as other DD insertion events with elements not present in common databases. A number of novel DD events and DD events with non-database elements were successfully validated using PCR and Sanger sequencing.

Table 2. Validation results for DD events with non-database elements

Chr	Position	Size	Val.	Alternative location	
				Chr	Position
4	88847037	NA	No	6	131648736
				X	586448
				...	...
10	98175504	NA	No	Y	1209696
				Un_g1000220	127792
				...	...
4	29441246	166	No	M	12836
				5	134261489
11	38812664	1345	**Yes**	8	52431261
				16	33940292
				...	...
8	15289364	215	No	13	74313858
12	108203260	609	**Yes**	7	111053153
X	136092966	4147	**Yes**	X	41649400
				1	119669113
				...	...
19	53602422	431	No	19	53632049
				19	53019294
				...	...

Three out of eight events were validated successfully. From left to right, the table columns describe the chromosome and position of the DD, as averaged by the estimations based on forward and reverse strand. The size of the inserted element, as estimated by the alternative alignment positions of the reads aligned inside the element. Whether the event was successfully validated (Val.). The final two columns show a selection of alternative alignment positions for reads mapping inside the element. The alternative alignments were found with BLAT having a score 80 or higher. All events have one or more such alternative alignments, all human chromosomes contain alternative alignment positions for the above events. There are also alternative alignments to mitochondrial DNA (M) and unordered sequences (e.g. Un_gl000220). A full list of alternative positions for the events in this table is given in the Supplementary Data.

Predictions made by DD_DETECTION for a human twin pair showed a discrepancy between both samples that is larger than one may expect for near identical genomes. Additionally, we noticed that low-complexity regions in the genome caused false-positive calls. We think these problems are part of the trade-off that comes with avoiding an external database of duplication elements. While it allows our method to call events with non-database elements, it cannot use information from a database to discard false calls. We showed a simple post-processing step with the RepeatMasker tool (Smit *et al.*, 1996) that we used to filter some of these false positives.

In the future, we want to focus on decreasing the false-positive rate, either by extending the pipeline with a filtering step for low-complexity DNA or by incorporating such filtering into the method. Another goal is to have the algorithm find optimal parameter values automatically given the input data. Currently, as a general guide line, the value of α should not be lower than 3, to keep control over the false-positive rate. In case of low coverage data, δ and λ can be increased to gain sensitivity. A more detailed explanation of the effects of parameter settings is given in the software documentation.

Acknowledgements

We thank Szymon Kiełbasa for stimulating discussions and his support. This study makes use of data produced by the 1000 Genomes project.

Funding

This research was supported by BBMRI-NL, a Research Infrastructure financed by the Dutch government (NWO 184.021.007). This publication was supported by the Dutch national program COMMIT.

Conflict of Interest: none declared.

References

Abyzov,A. *et al.* (2011) CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome Res.*, **21**, 974–984.

Chen,K. *et al.* (2009) BreakDancer: an algorithm for high-resolution mapping of genomic structural variation. *Nat. Methods*, **6**, 677–681.

Handsaker,R.E. *et al.* (2011) Discovery and genotyping of genome structural polymorphism by sequencing on a population scale. *Nat. Genet.*, **43**, 269–276.

Hormozdiari,F. *et al.* (2011) Alu repeat discovery and characterization within human genomes. *Genome Res.*, **21**, 840–849.

Jurka,J. *et al.* (2005) Repbase update, a database of eukaryotic repetitive elements. *Cytogenet. Genome Res.*, **110**, 462–467.

Kazazian,H.H. *et al.* (1988) Haemophilia A resulting from de novo insertion of L1 sequences represents a novel mechanism for mutation in man. *Nature*, **332**.

Keane,T.M. *et al.* (2013) RetroSeq: transposable element discovery from next-generation sequencing data. *Bioinformatics*, **29**, 389–390.

Kent,W.J. (2002) BLAT-the BLAST-like alignment tool. *Genome Res.*, **12**, 656–664.

Lander,E.S. *et al.* (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

Langmead,B. *et al.* (2009) Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol.*, **10**, R25.

Lee,E. *et al.* (2012) Landscape of somatic retrotransposition in human cancers. *Science*, **337**, 967–971.

Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, **25**, 1754–1760.

Miki,Y. *et al.* (1996) Mutation analysis in the BRCA2 gene in primary breast cancers. *Nature Genet.*, **13**, 245–247.

Pei,J. *et al.* (2001) PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: *Proceedings of the 17th International Conference on Data Engineering*, IEEE, pp. 215–224.

Platzer,A. *et al.* (2012) TE-Locate: a tool to locate and group transposable element occurrences using paired-end next-generation sequencing data. *Biology*, **1**, 395–410.

Smit,A.F. *et al.* (1996) RepeatMasker Open-3.0, http://www.repeatmasker.org/faq.html.

Stewart,C. *et al.* (2011) A comprehensive map of mobile element insertion polymorphisms in humans. *PLoS Genet.*, **7**, e1002236.

The 1000 Genomes Project Consortium (2012) An integrated map of genetic variation from 1 092 human genomes. *Nature*, **491**, 56–65.

Thung,D.T. *et al.* (2014) Mobster: accurate detection of mobile element insertions in next generation sequencing data. *Genome Biol.*, **15**, 488.

Trappe,K. *et al.* (2014) Gustaf: detecting and correctly classifying SVs in the NGS twilight zone. *Bioinformatics*, **30**, 3484–3490.

Wang,J. *et al.* (2006) dbRIP: a highly integrated database of retrotransposon insertion polymorphisms in humans. *Hum. Mutat.*, **27**, 323–329.

Wu,J. *et al.* (2014) Tangram: a comprehensive toolbox for mobile element insertion detection. *BMC Genomics*, **15**, 795.

Ye,K. *et al.* (2007) An efficient, versatile and scalable pattern growth approach to mine frequent patterns in unaligned protein sequences. *Bioinformatics*, **23**, 687–693.

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.

Ye,K. *et al.* (2013) Aging as accelerated accumulation of somatic variants: whole-genome sequencing of centenarian and middle-aged monozygotic twin pairs. *Twin Res. Hum. Genet.*, **16**, 1026–1032.

Zhang,Y. *et al.* (2012) PASSion: a pattern growth algorithm-based pipeline for splice junction detection in paired-end RNA-Seq data. *Bioinformatics*, **28**, 479–486.