*Bioinformatics*, 32(12), 2016, 1788–1796
doi: 10.1093/bioinformatics/btw053
Advance Access Publication Date: 1 February 2016
Original Paper

OXFORD

Sequence analysis

# Sprites: detection of deletions from sequencing data by re-aligning split reads

**Zhen Zhang[1,2], Jianxin Wang[1,*], Junwei Luo[1], Xiaojun Ding[1], Jiancheng Zhong[1], Jun Wang[3], Fang-Xiang Wu[4] and Yi Pan[5]**

[1]School of Information Science and Engineering, Central South University, Changsha, 410083, China, [2]College of Information and Communication Engineering, Hunan Institute of Science and Technology, Yueyang, 414006, China, [3]Department of Molecular Physiology & Biophysics, Baylor College of Medicine, Houston, TX 77030, USA, [4]Department of Mechanical Engineering and Division of Biomedical Engineering, University of Saskatchewan, Saskatoon, SK S7N 5A9, Canada and [5]Department of Computer Science, Georgia State University, Atlanta, GA 30302-4110, USA

*To whom correspondence should be addressed.

Associate Editor: Inanc Birol

Received on September 5, 2015; revised on January 25, 2016; accepted on January 25, 2016

## Abstract

**Motivation**: Advances of next generation sequencing technologies and availability of short read data enable the detection of structural variations (SVs). Deletions, an important type of SVs, have been suggested in association with genetic diseases. There are three types of deletions: blunt deletions, deletions with microhomologies and deletions with microsinsertions. The last two types are very common in the human genome, but they pose difficulty for the detection. Furthermore, finding deletions from sequencing data remains challenging. It is highly appealing to develop sensitive and accurate methods to detect deletions from sequencing data, especially deletions with microhomology and deletions with microinsertion.

**Results**: We present a novel method called Sprites (SPlit Read re-allgnment To dEtect Structural variants) which finds deletions from sequencing data. It aligns a whole soft-clipping read rather than its clipped part to the target sequence, a segment of the reference which is determined by spanning reads, in order to find the longest prefix or suffix of the read that has a match in the target sequence. This alignment aims to solve the problem of deletions with microhomologies and deletions with microinsertions. Using both simulated and real data we show that Sprites performs better on detecting deletions compared with other current methods in terms of F-score.

**Availability and implementation**: Sprites is open source software and freely available at https://github.com/zhangzhen/sprites.

**Contact**: jxwang@mail.csu.edu.cn

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

## 1 Introduction

Structural variation (SV) was originally defined as insertions, deletions and inversions larger than 1k bp in size (Feuk *et al.*, 2006), and now has been extended to include much smaller variants (e.g. those >50 bp in length) (Alkan *et al.*, 2011) and more types of variants, such as translocation and tandem duplication. These variants are prevalent in human populations and are associated with human diseases, complex traits and evolution (Baker, 2012). Thus, finding SVs is an important task. Recent advances in high throughput sequencing make it possible to reveal more variants than ever before. Many efforts have been made to detect variants from high throughput sequencing data. For example, the 1000 Genomes Project

Consortium has released SV data of 1092 individuals from 14 populations (Consortium *et al.*, 2012). Some methods are specially designed for detecting a specific type of SVs: SVSeq (Zhang and Wu, 2011) for deletion and MindTheGap (Rizk *et al.*, 2014) for insertion. A deletion indicates a DNA segment missing in an individual genome, also known as a donor/sample genome, compared with the reference genome. Eighty percent of genetic disorders in the disease database, Database Chromosomal Imbalance and Phenotype in Humans using Ensembl Resources (DECIPHER), are caused by deletions (Weischenfeldt *et al.*, 2013). Deletions are such an important type of SVs that almost every SV discovery tool has developed a module to find deletions. We focus on the discovery of deletions in this article.

Read pairs are the most common form of current sequencing data. DNA libraries are generally constructed by shearing a genome into fragments, cloning and size-selecting the fragments. A library is a collection of fragments with a roughly equal size. The length of a fragment excluding adapters at two ends is commonly referred to as the insert size. The insert size varies from fragment-to-fragment. The exact value of insert size for each fragment cannot be determined but its approximate value can be estimated by sampling. The normal range of insert sizes is specified through the library mean and standard deviation (Luo *et al.*, 2015a, b). Two reads of a read pair are generated by sequencing two ends of a fragment. Before calling variants are performed, these read pairs need to be mapped to a reference genome using read mappers such as BWA (Li and Durbin, 2009) and Bowtie2 (Langmead and Salzberg, 2012). If two reads of a read pair are successfully mapped, its insert size is then given as the distance between two corresponding locations on the reference genome. An anomalous insert size indicates a value beyond the normal range. The corresponding read pairs are called discordant read pairs.

Analyzing discordant read pairs to reveal variants, such as read pair method, is one of the most common approaches. Many tools adopt such approach, such as BreakDancer (Chen *et al.*, 2009), PEMer (Korbel *et al.*, 2009), VariationHunter (Hormozdiari *et al.*, 2009, 2010), and GASV (Sindi *et al.*, 2009). Although read pair methods can improve the resolution of calling with high-coverage data, they uncover variants by giving only inexact positions of breakpoints. The read depth method is another approach that gives approximate breakpoints. Read depth refers to the number of reads mapped to a particular part of the genome and can indicate how many copies of a region are present, but it cannot indicate where the copies occur (Baker, 2012). SegSeq (Chiang *et al.*, 2009), EWT (Yoon *et al.*, 2009) and CNVnator (Abyzov *et al.*, 2011) are some examples of algorithms that apply this approach.

Assembly and split read methods are two types of approaches that are able to detect variants with base-pair breakpoint resolution. Assembly methods exploit aberrations from the reference genome to identify locations where variants might be, and then assemble reads just for that area (Baker, 2012). Comparing the assembled contigs to the area on the reference genome can detect variants with exact breakpoints. However, assembly methods have limitations. Although only local assembly is performed, all reads of the library are processed in order to construct the $k$-mer spectrum that is required for assembly. The step requires a large amount of time and memory to run. It also tends not to deal well with heterozygous variants, which occur on only one of a pair of homologous chromosomes (Baker, 2012).

Split reads refer to those that cover breakpoints of variants whether they are single-end or paired-end. Split read methods, as their name implies, derive variants from these split reads. Read aligners can help identify split reads. Given a pair of reads $(r_i, r_i')$, if $r_i$ is mapped and $r_i'$ is either unmapped or soft-clipped at the 5′- or 3′-end, $r_i'$ may be a split read. In some cases it may not be a split read due to either sequencing error or mapping error. There are two ways to use split reads to detect variants: via split read mapping and via soft-clipped mapping. Split read mapping focuses on unmapped reads. An unmapped read was first broken up into two parts. Then, these two parts are respectively mapped to the reference sequence, which results in the breakpoint of the corresponding variant being pinpointed. Examples of split read mapping-based methods include Pindel (Ye *et al.*, 2009), AGE (Abyzov and Gerstein, 2011), SVSeq (Zhang and Wu, 2011), PRISM (Jiang *et al.*, 2012) and DELLY (Rausch *et al.*, 2012). Soft-clipped mapping focuses on reads with the 5′- or 3′-end soft-clipped. These reads are also called soft-clipping reads. One breakpoint of the variant is specified by the mapping location where soft-clipping occurs. The other breakpoint is determined by aligning the soft-clipped segment of the read to the reference sequence. ClipCrop (Suzuki *et al.*, 2011), CREST (Wang *et al.*, 2011), SVSeq2 (Zhang *et al.*, 2012) and Socrates (Schröder *et al.*, 2014) are representatives of soft-clipped mapped-based methods. Split read methods have a few disadvantages, such as time and memory inefficiency, and both high false positive and false negative rates. Some of them do not perform well on low-coverage data.

Three deletion types are observed in the human genome: (1) blunt deletions: nothing special happened at the breakpoints, (2) deletions with microhomologies: two small identical sequences at deletion breakpoints, and (3) deletions with microinsertions: deletion breakpoints having a small untemplated sequence inserted. Conrad *et al.* (2010) studied the breakpoints of 315 deletions and found that 70% of breakpoints have 1–30 bp of microhomology, 33% of breakpoints contain 1–369 bp of inserted sequence, and 10% of breakpoints have both simultaneously. Only a few breakpoints (∼7%) have blunt ends. The presence of microhomology and microinsertion creates problems for re-aligning the clipped part. Microhomology in a soft-clipping read causes the clipped part to be too short for the alignment. The alignment algorithm returns multiple hits for the clipped part. Finding the correct one among these hits is challenging. Microinsertion in the clipped part causes the alignment to fail because inserted sequence cannot match the reference. However, split read mapping can deal with microhomology and microinsertion. Pindel uses the pattern growth approach to report deletions with microinsertions. AGE aligns the 5′ and 3′ ends of two given sequences simultaneously and creates a jumping gap to address their presence. Delly follows the AGE approach and makes changes to AGE. Despite the availability of these tools, methods with high accuracy are required for the detection of deletions with microhomologies and microinsertions.

In this article, we present a new method called Sprites (SPlit Read re-alIgnment To dEtect Structural variants) for detecting deletions from sequencing data. Sprites can solve the problem that microhomologies and microinsertions cause. It re-aligns the whole read rather than the clipped part to the target sequence, a segment of the reference, in order to find the longest prefix or suffix of the read that has a match in the target sequence. In the case of microhomology, the length of the sequence to be matched is extended to the length of clipped part plus the length of microhomology. The longest mapped prefix or suffix of the read can usually cover microhomology. Thus, the deletion call is easy to determine. In the case of microinsertion, the longest matched prefix or suffix of the read can avoid the impact of microinsertion on the detection. The comparison of the re-alignments of the soft-clipped segment and the whole read is illustrated in Figure 1. Sprites uses alignments produced by

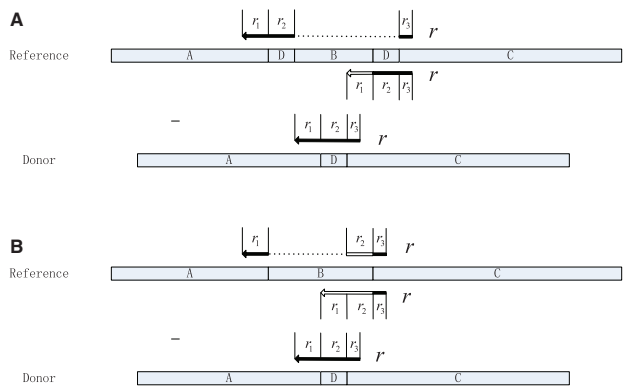


**Fig. 1.** Comparison between the re-alignments of the soft-clipped segment and the whole read in the detection of deletions with microhomology and microinsertion. The read $r$ consists of three segments: $r_1$, $r_2$ and $r_3$. The re-alignment of the soft-clipped segment is shown above the reference. The re-alignment of the whole read is shown below the reference. The segments A and C are common in both the reference and the donor. The segment B represents the deletion in the donor. (**A**) Microhomology at the breakpoints. The two Ds refer to microhomologies. They are two identical sequences of small size. The read includes one D, expressed as $r_2$. $r_3$ represents the soft-clipped segment. (**B**) Microinsertion at the breakpoints. D is a microinsertion. $r_2$ refers to the microinsertion

BWA, while it can also use alignments produced by other read aligners that support 5′- or 3′-end soft-clipping, like Bowtie2 (Langmead and Salzberg, 2012).

Re-alignment is one of most time-consuming tasks in the detection. A target sequence is a segment of the reference. For a soft-clipping read, Sprites relies on its spanning read pairs to determine the size and location of target sequences. Given that most of these target sequences have a length of only hundreds of base pairs, re-aligning soft-clipping reads to them saves a large amount of time. The input file has the size on the order of Gigabytes. Sprites transverses it from start to end only once and only stores information about soft-clipping reads that are useful for deletion detection, which reduces Sprites' memory footprint. Besides its great performance on low-coverage data, Sprites can also be used for the analysis of high-coverage data. We tested it extensively on the simulated data and real sequencing data and compared it with four other detection tools including SVSeq2, LUMPY, Delly and Pindel. The results show that among these tools Sprites is highly sensitive at the relatively low false discovery rates and thus has the greatest *F*-scores in many cases.

The major contributions of this article include: (1) our method can find the longest prefix or suffix of a soft-clipping read that has a match in the target sequence by performing the re-alignment; (2) our method solves the problem of deletions with microhomology and deletions with microinsertion, which are very challenging to be found from the sequencing data; (3) our method limits the alignment length so that time and memory usage are dramatically reduced; (4) a piece of open source software is implemented based on our method and can be freely available.

## 2 Methods

Sprites takes as input a BAM file that stores paired-end reads along with alignment information. The BAM file needs to be generated by any read aligner that is able to perform soft-clipping alignment such as BWA, Bowtie2. We mainly focused on soft-clipping reads, at the 5′-end of which more than a specified number of base pairs are soft-clipped. These soft-clipping reads then go through the following processing steps. Eventually deletions that these soft-clipping reads indicate will be discovered.

### 2.1 Preprocessing

Preprocessing as the first step of our method is aimed at retrieving some reads from the alignment file. Most of these reads correspond to deletions in the input human genome sample. These reads follow the same pattern: they have soft-clipping signature at either end. Information about all these reads including location, soft-clipping and their mate is stored into a read set when the preprocessing is done.

To define the 5′-end soft-clipping read, we first list two cases for a read pair $P(r_i, \overline{r}_i)$:

*Case 1*: The first $k_{r_i}$ base pairs of the read $r_i$ is forwardly mapped, the mate $\overline{r}_i$ is reversely mapped and the read $r_i$ appears upstream of the mate $\overline{r}_i$ on the reference.

*Case 2*: The last $k_{r_i}$ base pairs of the read $r_i$ is reversely mapped, the mate $\overline{r}_i$ is forwardly mapped and the read $r_i$ appears downstream of the mate $\overline{r}_i$ on the reference.

Read $r_i$ in read pair $P(r_i, \overline{r}_i)$ is called 5′-end soft-clipping if Case 1 or Case 2 is satisfied and the distance from the read $r_i$ to the mate $\overline{r}_i$ on the reference is in the range of the mean size within four standard deviations (Li, 2013), which is called as the normal range in this article. Note that the distance includes the length of two reads in this article. A soft-clipping read $r_i$ is represented by a 4-tuple $(seq_{r_i}, k_{r_i}, loc_{r_i}, case_{r_i})$, where $\text{seq}_{r_i}$ denotes the sequence of the read, $k_{r_i}$ is the length of mapped prefix or suffix and $loc_{r_i}$ is the location of soft-clipping. If the read is satisfied with Case 1, $case_{r_i}$ is assigned to 1, otherwise to 2.

Assume that the insert size of read pairs follows a normal distribution with the mean equal to the library average $\mu$ and the standard deviation equal to the library standard deviation $\sigma$. $\mu$ and $\sigma$ can be estimated by the sample average $\overline{X}$ and the sample standard deviation $s_X$, respectively. If the insert size and its standard deviation are not given at the beginning, we can use a sampling method presented in Zhang *et al.* (2012) to estimate them.

In the preprocessing, we will not use the reads for the deletion detection if they have ambiguous alignment. Reads that are from repetitive regions usually have multiple hits, so they are not used for the detection. Soft-clipping reads are selected from the input BAM file and added to a read set $\mathcal{R}$ based on the FLAG field and the CIGAR string of each read. Not only does the FLAG field show the mapping orientations of both the read and the mate, but it also indicates whether the distance between them is in the normal range. From the CIGAR string, we can determine the presence of soft-clipping and the value of $k_i$. Suppose that a read $r_i$ has a CIGAR string '50S51M', where 'S' represents soft-clipping, 'M' represents matching and the number before them indicates how many base pairs are involved (Li and Durbin, 2009). We know that the first 50 base pairs are soft-clipped and the remaining 51 base pairs are mapped, i.e. $k_{r_i}$=51. One thing to note is that only reads with the length of the soft-clipped segment $l_{r_i} - k_{r_i}$ larger than or equal to $M_{sc}$ should be kept, where $l_{r_i}$ denotes the read length and $M_{sc}$ denotes the minimum alignment length (12 by default).

To improve the efficiency of running time, we try to reduce the number of reads in the set $\mathcal{R}$. If multiple reads in $\mathcal{R}$ with the same Case are soft-clipped at the same location, i.e. the values of *loc* and *case* in their 4-tuples are the same, they will be grouped together. Reads in a group are sorted in ascending order of the length of soft-

clipped prefix or suffix. Then only the middle read in the group is kept and other reads in the group are removed from $\mathcal{R}$.

## 2.2 Determining the target sequences

In this step, we use a reversely mapped read $r_i$ with the soft-clipping location $loc_{r_i}$ in $\mathcal{R}$, i.e. $r_i$ is satisfied with Case 2, to illustrate the process of determining the target sequences. The process is illustrated in Figure 2. A read pair $P(r_k, \bar{r}_k)$ with $r_k$ forwardly mapped and $\bar{r}_k$ reversely mapped is called a spanning pair $SP_{r_i}(r_k, \bar{r}_k)$ on the read $r_i$ when $r_k$ occurs in the range from $loc_{r_i} - \mu - 3\sigma + l_{r_k}$ to $loc_{r_i}$ and $\bar{r}_k$ must be on the right of $loc_{r_i}$. This range was derived according to Lemma 1 in Zhang *et al.* (2012), which guarantees that almost all read pairs from this range really span the soft-clipping location $loc_{r_i}$, i.e. one breakpoint of $r_i$. All possible spanning pairs on the read $r_i$ were selected and added to a set $\mathcal{SP}_{r_i}$.

Given a spanning pair $SP_{r_i}(r_k, \bar{r}_k)$, a fragment of length $\mu + 3\sigma$ ending at the location $loc_{\bar{r}_k} + l_{\bar{r}_k}$, which we call a target sequence $s^k_{r_i}$ of the read $r_i$, was extracted from the reference according to Lemma 2 in Zhang *et al.* (2012). This lemma ensures that the target sequence Sprites obtains is long enough to completely cover the soft-clipped segment of $r_i$ in almost every case. For each spanning pair of the set $\mathcal{SP}_{r_i}$, a set $\mathcal{S}_{r_i}$ including a target sequence $s^k_{r_i}$ along with its ending location $end(s^k_{r_i})$ was created. All sequences in the set of target sequences $\mathcal{S}_{r_i}$ were then sorted by their ending location. If two adjacent sequences overlap, they will be merged. The larger one of the two ending locations acts as the ending location of the merged sequence. Merging continues until there is nothing left to merge. Actually, a soft-clipping read often corresponds to more than one target sequences. Eventually, Sprites obtained a set of target sequences for each reversely mapped read in the read set $\mathcal{R}$. The set of target sequences $\mathcal{S}_{r_i}$ will be used to call a deletion for the read $r_i$.

The procedure in determining target sequences for a forwardly mapped read is the same except for the following subtleties. In choosing spanning pairs, the range used for selecting $\bar{r}_k$ is from $loc_{r_i}$ to $loc_{r_i} + \mu + 3\sigma - l_{\bar{r}_k}$, and $r_k$ must be on the left of $loc_{r_i}$. In extracting a target sequence, Sprites takes the fragment of length $\mu + 3\sigma$ starting at $loc_{r_k}$ on the reference. When adding a target sequence to $\mathcal{S}_{r_i}$, its starting location $start(s^k_{r_i})$ is added along with it. The smaller one of the two starting locations acts as the starting location of the merged sequence.

## 2.3 Finding deletion calls

The aim of calling deletion is, given a soft-clipping read $r_i$ and the set of target sequences $\mathcal{S}_{r_i}$, to pinpoint the left breakpoint interval $I^l_{r_i}$

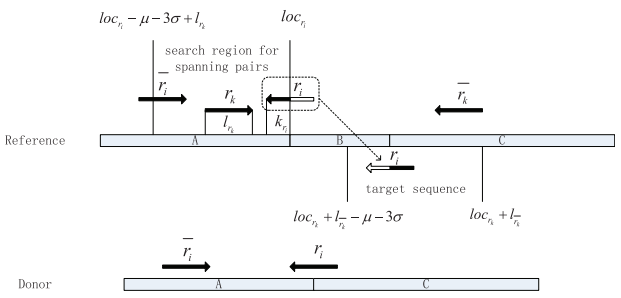


**Fig. 2.** Identification of spanning pairs and re-alignment of 5′-end soft-clipping reads in Case 2. The segment A and C are adjacent in the donor, but they are not adjacent in the reference. The segment B of the reference refers to a deletion that occurs on the donor sequence. The arrow $r_i$ is a 5′-end soft-clipping read. The solid part represents the mapped part of the read, while the hollow part represents the soft-clipped part of the read. The arrow pair $(r_k, \bar{r}_k)$ is a spanning pair for $r_i$

and the right breakpoint interval $I^r_{r_i}$ of the deletion that corresponds to $r_i$, respectively. The result obtained at the end of this step is a set of breakpoint interval pairs, which will be merged at the next step. Like the previous step, we used a reversely mapped read to illustrate the process of calling deletions.

Target sequences in $\mathcal{S}_{r_i}$ were arranged in right-to-left order according to their ending location. As a result, the rightmost target sequence was in first place and the leftmost target sequence was in last place. $r_i$ was then re-aligned to each target sequence in $\mathcal{S}_{r_i}$ one by one until Sprites found an alignment between a suffix of $r_i$ and a substring of a target sequence of this read, which is satisfied with the following conditions: (1) its length should not be less than $M_{sc}$; (2) its percentage identity should not be less than $M_{pi}$ which denotes the minimum percentage identity (96% by default); (3) it is the longest one among all alignments satisfied with the conditions (1) and (2). Both mismatches and indels were taken into account in the alignment.

Suppose that when aligning $r_i$ to the target sequences in $\mathcal{S}_{r_i}$, an alignment $a_i$ between the suffix of $r_i$ and a substring of the target sequence $s^k_{r_i}$ was found. The initial location of the right breakpoint was computed using $end(s^k_{r_i}) - l_{a_i}$, where $l_{a_i}$ denotes the length of $a_i$. If $l_{a_i}$ is larger than the length of soft-clipped segment $l_{r_i} - k_{r_i}$, the deletion is mediated by microhomology with high probability, especially when neither sequencing error nor mapping error happens. So an interval with a length of $|l_{a_i} - l_{r_i} + k_{r_i}|$ starting at the location $end(s^k_{r_i}) - l_{a_i}$ can be used to represent the right breakpoint. $loc_{r_i}$ denotes the initial location of the left breakpoint. Similarly, an interval with a length of $|l_{a_i} - l_{r_i} + k_{r_i}|$ ending at $loc_{r_i}$ is used to represent the left breakpoint. The two intervals of left and right breakpoints which represent a deletion call of $r_i$ are added to a call set $\mathcal{D}$. If $l_{a_i}$ is less than $l_{r_i} - k_{r_i}$, a small non-template insertion may occur with high probability, especially when neither sequencing error nor mapping error happens. In this case, an interval with a length of $|l_{a_i} - l_{r_i} + k_{r_i}|$ ending at the location $end(s^k_{r_i}) - l_{a_i}$ is used to represent the right breakpoint. The left breakpoint is represented by another interval starting at $loc_{r_i}$ with the same length. Similarly, the two intervals of left and right breakpoints are added to the call set $\mathcal{D}$.

Each deletion call was represented by a pair of intervals which represent the left breakpoint and the right breakpoint, respectively, i.e. the left interval and the right interval. It is clear that each read in $\mathcal{R}$ has one deletion call at most. The length of the deletion call was determined by subtracting the starting location of the left interval from the starting location of the right interval.

When dealing with a forwardly mapped read with soft-clipping, there were small differences in calling deletions. Target sequences of $\mathcal{S}_{r_i}$ were arranged in left-to-right order instead. Sprites searched for an alignment between the longest prefix of $r_i$ and a substring of a target sequence. The initial location of the left breakpoint was computed using $start(s^k_{r_i}) + l_{a_i}$. The initial location of the right breakpoint was denoted by $loc_{r_i}$. When $l_{a_i} > l_{r_i} - k_{r_i}$, the interval with a length of $|l_{a_i} - l_{r_i} + k_{r_i}|$ ending at $start(s^k_{r_i}) + l_{a_i}$ was used to represent the left breakpoint, and the interval with the same length starting at $loc_{r_i}$ was used to represent the right breakpoint. When $l_{a_i} < l_{r_i} - k_{r_i}$, the interval with a length of $|l_{a_i} - l_{r_i} + k_{r_i}|$ starting at $start(s^k_{r_i}) + l_{a_i}$ was used to represent the left breakpoint, and the interval with the same length ending at $loc_{r_i}$ was used to represent the right breakpoint.

A deletion call is represented by a pair of intervals $(I^l_i, I^r_i)$, where $I^l_i$ and $I^r_i$ denote the left interval and the right interval, respectively. Two calls are said to overlap if their left intervals overlap and their right intervals overlap at the same time. These calls were lexically sorted by the ordered vector consisting of the start of $I^l_i$, the end of

$I_i^l$, the start of $I_i^r$ and the end of $I_i^r$. The first call is kept in $\mathcal{D}$. If any subsequent calls overlap it, they are removed from $\mathcal{D}$. The second call in $\mathcal{D}$ is dealt with in the same way. This process continues until the last call is done. The resulting call set $\mathcal{D}$ is output to a file in the BEDPE format, which is used to specify genomic regions, at the end.

## 3 Results

We compared Sprites with the four most commonly used SV detection tools, i.e. Pindel, SVSeq2, Delly and LUMPY (Layer *et al.*, 2014). Pindel is the first tool that relies on the concept of splitting reads to detect variants. Besides deletions, it is able to call other types of variants, such as insertions and inversions. SVSeq2 is a tool that specializes in finding deletion calls. Deletions are called by re-aligning the soft-clipped sequence of reads, which is similar to our tool. It focuses on analyzing low-coverage data. The latest version of SVSeq2 can only process one chromosome at a time. When working on whole genome data, we first ran SVSeq2 for each chromosome (or contig) of the human reference genome and then concatenated the results of individual chromosomes to obtain the final results. LUMPY is a probabilistic-based approach for SV discovery, which integrates multiple SV detection signals, such as read pairs, split reads, thereby achieving a substantial improvement in detection as compared with other popular SV tools such as BreakDancer, GASVPro (Sindi *et al.*, 2012).

Homozygous variants are commonly used for the detection evaluation. However, heterozygous variants are prevalent variants which are often less deleterious but more frequent among genetic disorders compared with homozygous variants. Moreover, the detection of heterozygous variants plays a substantial role in the tumor study because that real samples tend to be a mixture of abnormal and normal genomes and tumor samples usually to have more heterogeneous variants than homozygous ones (Weischenfeldt *et al.*, 2013). However, detecting heterozygous variants is problematic. So we use heterozygous variants for the evaluation besides homozygous variants.

Ryan Layer, the author of Lumpy, provided us with two artificial genomes: one genome with 2500 randomly generated deletions of size 100 bp to 10 kbp, the other genome with 5516 non-randomly generated deletions. The 5516 deletions were publicly released by the 1000 Genomes Project. SVSim (https://github.com/GregoryFaust/SVsim), an SV simulator, was used to generate these genomes by introducing these deletions into the b37 version of human reference, also known as GRCh37 version. The FASTA file of the b37 version can be found at ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/reference/. The location of these simulated deletions in these genomes was recorded in two BEDPE files. We used the first genome for homozygous deletion detection and the second genome for heterozygous deletion detection.

The first genome was intended for homozygous deletion detection. We used wgsim (https://github.com/lh3/wgsim), a read simulator, to sequence this genome with 2×, 5×, 10×, 20× and 50× haploid coverage, respectively, to generate paired-end reads of length 150 bp. The insert size of paired-end reads was centered at 500 bp with the standard deviation equal to 50 bp. Generated reads contained sequencing errors (the overall error rate of 0.5%).

The second genome was intended for heterozygous deletion detection. In order to generate heterozygous deletions, a normal genome and an abnormal genome were required. We used the b37 version of the human reference genome as the normal genome and the genome with 5516 deletions as the abnormal genome. Paired-end reads were generated by using wgsim to sequence the two genomes with 0.05, 0.1, 0.2 and 0.5 SV allele frequencies at 10×, 20× and 40×, respectively. For example, we generated reads with 0.05 SV allele frequency at 10× coverage like this: we used wgsim to sequence the normal genome at 9.5× haploid coverage and sequence the abnormal genome at 0.5× haploid coverage, then the two sets of reads were combined to form pair-end reads with 0.05 SV allele frequency at 10× coverage. Reads for heterozygous deletions have the same properties with reads for homozygous deletions.

Whether reads are for homozygous deletions or for heterozygous deletions, they need to be mapped to the reference for use with detection tools. BWA ALN was used to map reads. Then, alignment files were sorted and indexed by SAMtools (Li *et al.*, 2009). LUMPY required as input two BAM files: a file that was the original BAM file and a file that represented split read alignment. The split read alignment file was generated as follows: split reads were first extracted from the original BAM file using a custom script provided along with the Lumpy program; these split reads were then re-aligned by YAHA (Faust and Hall, 2012).

Furthermore, we used the data of the NA12878 individual released by the Illumina Platinum Genomes project and the data of the five other individuals (NA19311, NA19312, NA19313, NA19316 and NA19317) provided by the 1000 genomes project, as the real data for the evaluation.

Sprites uses two micro-intervals to represent the two breakpoints of a deletion. Micro-intervals reflect the fact that microhomologies and microinsertions occur at deletion breakpoints. LUMPY also uses two intervals to represent breakpoints. However, for each deletion SVseq2 and Pindel predicted, two breakpoints were converted to two breakpoint intervals with a length 100 bp. For each known deletion, the same conversion was performed. Since Delly provides the confidence interval of deletion breakpoints, we used them as breakpoint intervals and no conversion was performed.

A deletion call is represented by two breakpoint intervals $A$ and $B$. A known deletion is represented by two breakpoint intervals $A'$ and $B'$. The deletion calls overlaps the known deletion if and only if $A$ overlaps $A'$ and $B$ overlaps $B'$. BEDTools (Quinlan and Hall, 2010) was used for checking such overlaps. A call is a true positive ($TP$) if the call overlaps a known deletion, otherwise it is a false positive ($FP$). False negatives ($FN$) refer to the known deletions that SV detection tools failed to report. A comprehensive measure, called $F$-score, is mainly used to evaluate the methods. The $F$-score is defined as the harmonic mean of the sensitivity and the precision ($1 - FDR$). The sensitivity is defined as $\frac{TP}{TP+FN}$ while the false discovery rate ($FDR$) is defined as $\frac{FP}{TP+FP}$.

### 3.1 Evaluation on the strategy of re-aligning the whole read

We designed a dedicated experiment to show Sprites's improvement in the detection of deletions with microhomology and microinsertion. One hundred thirty-two validated deletions, which consists of 10 blunt deletions (7.6%), 96 deletions with microhomology (72.7%) and 26 deletions with microinsertion (19.7%), were introduced to the chromosome 15 of the Hg18 reference. These 132 deletions reported by Mills *et al.* (2011) are from 45 individuals of the CEU population on the chromosome 15, and the frequencies of lengths of these deletions share the same trend with the frequencies of lengths of deletions found by the 1000 Genomes Project. Furthermore, the frequencies of three deletion types roughly match the frequencies of three deletion types (Conrad *et al.*, 2010) found. Simulated paired-end reads ($101 \times 2$ bp) were then generated by wgsim at the error rate 0.005 and at the coverage of 5. The mean

and the standard deviation of insert size are 500 and 50, respectively. The results in Table 1 show that Sprites achieves the highest detection accuracy and is the most sensitive in finding deletions with microhomology and microinsertion. SVseq2 that re-aligns the clipped part is much less sensitive in finding deletions with microinsertion. Delly that uses split read mapping and Pindel that applies the pattern growth approach are less sensitive than Sprites that re-aligns the whole read.

### 3.2 Results on simulated homozygous deletions

Sensitivity and FDR of homozygous deletion detection on five tools are shown in Supplementary Table S1. Sprites consistently achieves higher sensitivity than other tools across almost all coverage levels. Pindel is negligibly more sensitive than Sprites at 20× and 50× coverage (99.4 versus 98.5% and 99.5 versus 98.5%), but has much higher FDR than Sprites (4.2 versus 0.4% and 18.8 versus 0.9%) at the same coverages. In terms of sensitivity, Sprites performs extraordinarily well at the low coverage (2× and 5×). It is more sensitive than the second best (SVseq2 and LUMPY) at 2× and 5× coverage (59.3 versus 44.8% and 92.1 versus 86.2%). The FDR of Sprites rises as the coverage increased from 2× to 50×, but the maximal FDR that appeared at 50× coverage is very small, just 0.9%. Although SVSeq2 has the lowest FDR over all coverage levels, it is not so sensitive compared to others in general.

To comprehensively compare all five methods, Figure 3 shows that *F*-score of homozygous deletion detection on five tools. Sprites gets the best *F*-score in all cases. In terms of *F*-score, Sprites performs even better in low-coverage data than other tools. The results show the evidence that our method is effective in solving the problems caused by deletions with microhomologies and microinsertions.

**Table 1.** The comparison of the detection of deletions with microhomology and microinsertion

| Tool | Total FDR | Sensitivity of blunt deletions | Sensitivity of deletions with microhomology | Sensitivity of deletions with microinsertion |
|---|---|---|---|---|
| Sprites | 0 | 60 | 76 | 76.9 |
| SVseq2 | 0 | 20 | 22.9 | 15.4 |
| Lumpy | 4.6 | 60 | 62.5 | 65.4 |
| Pindel | 2.5 | 60 | 59.4 | 53.8 |
| Delly | 9.4 | 60 | 65.6 | 69.2 |

FDR, false discovery rate. Sensitivity and FDR are expressed as percentages.

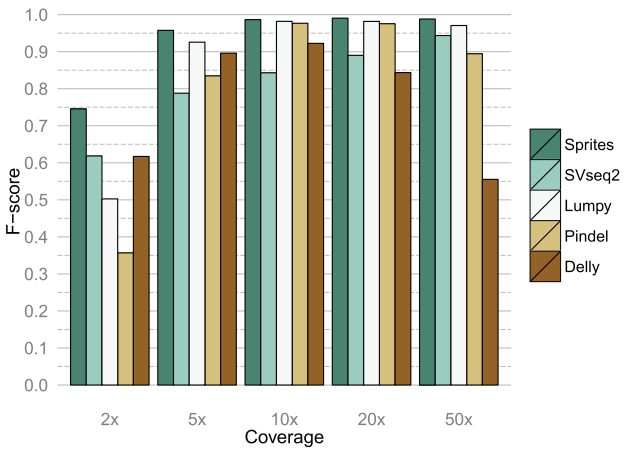


**Fig. 3.** Performance comparison of homozygous deletions. Two thousand five hundred deletions are randomly generated in the simulation (Color version of this figure is available at *Bioinformatics* online.)

### 3.3 Results on simulated heterozygous deletions

Supplementary Tables S2 and S3 show the sensitivity and FDR of heterozygous deletion detection on five tools, respectively. Sprites is more sensitive than others where either SV allele frequency or the coverage or both are low, i.e. the abnormal genome is sequenced at lower coverages. As a major competitor to Sprites for SV detection from low-coverage data, SVSeq2 performs well compared with LUMPY and Pindel, but is less sensitive than Sprites. For example, Sprites and SVSeq2 correctly detect 50.9 and 39.2% of deletions, respectively, in the case of 0.1 SV allele frequency and 10× coverage. As seen from Supplementary Table S3, SVSeq2 achieves the lowest FDR in nearly all cases. The FDRs of Sprites and SVSeq2 are generally very close. The highest difference (0.6%) between the FDRs of Sprites and SVSeq2 appears in the case of 0.5 SV allele frequency and 40× coverage. Sprites achieves a lower FDR than SVSeq2 in the case of 0.05 SV allele frequency and 50x coverage.

Figure 4 shows that *F*-score of heterozygous deletion detection on five tools. At 10× and 20× coverage, the *F*-score of Sprites is the best with 0.05, 0.1 and 0.2 SV allele frequencies. With 0.5 SV allele frequency, the *F*-score Sprites is not the best. At 40× coverage, the *F*-score of Sprites is the best only with 0.05 and 0.1 SV allele frequencies while it is not better in other cases compared with Lumpy and Pindel. The reason may be that both Lumpy and Pindel directly use paired-end information while Sprites does not use this information directly. Actually high-coverage data contain much more paired-end information which can help improve the detection if directly used.

### 3.4 Results on the subsample of NA12878

Calls made from the real data are difficult to evaluate in terms of sensitivity and FDR because there are still variants that remain unknown for each individual. The number of known variants varies from individual to individual. However, performance comparison based on the real data is indispensable because not all sequencing artifacts can be simulated by any read simulator. Besides, only if an SV detection tool performs well and precisely on the real data, can biologists actually benefit from this. Sequencing data of NA12878 is most commonly used in the evaluation of SV tools. The golden standard SV list we used is recently released in (Abyzov *et al.*, 2015). This list contains 8943 deletions.

Two FASTQ files storing paired-end reads of the NA12878 sample were downloaded under the run accession ERR194147 from the European Nucleotide Archive (http://www.ebi.ac.uk/ena). We subsampled 10% paired-end reads from these files by setting each paired-end read to have a probability of 0.1 to be selected. Since the original files have approximately 50× coverage, so the resulting subsampling files had roughly 5× coverage. They were then mapped with BWA ALN and sorted with SAMtools.

We can see from Table 2 that in terms of sensitivity, Sprites obtained the best value (14.12%); in terms of FDR, although Sprites got the second better value (42.7%), the difference between the value and the best one that SVSeq2 got (38.03%), is not that large. Sprites performs the best among all five methods in terms of the comprehensive *F*-score.

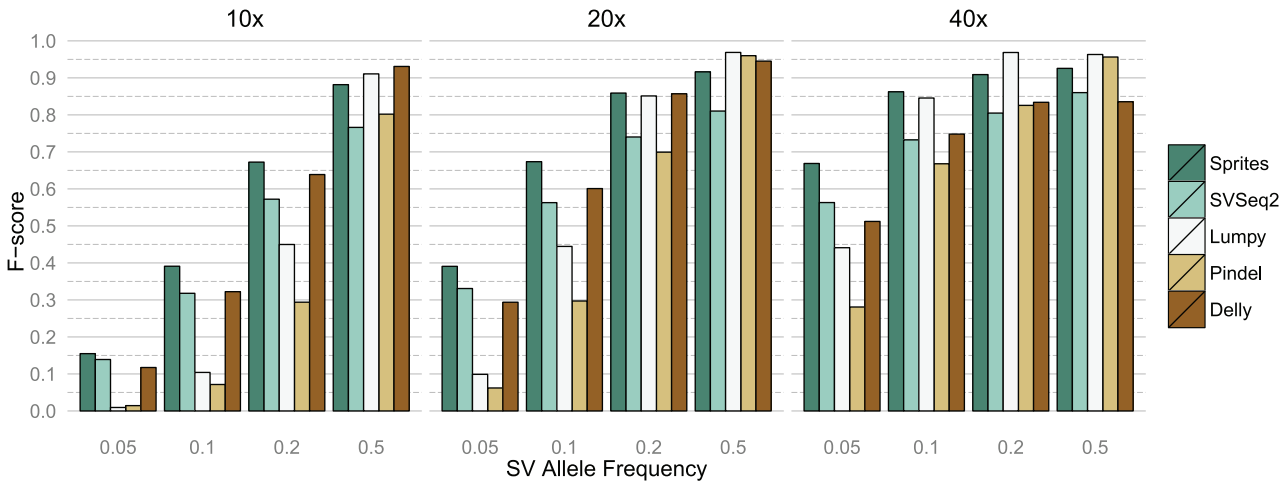


**Fig. 4.** Performance comparison of deletion detection in the heterozygous simulation. This simulation introduced 5516 non-overlapping deletions identified by the 1000 Genomes Project (Color version of this figure is available at *Bioinformatics* online.)

**Table 2.** Performance comparison of deletion detection in a 5× subsample of Illumina sequencing data of NA12878

| Tool | Sensitivity | FDR | *F*-score |
|---|---|---|---|
| Sprites | 14.1 | 42.7 | 0.227 |
| SVseq2 | 4.7 | 38 | 0.088 |
| Lumpy | 9.1 | 77.7 | 0.129 |
| Pindel | 8 | 61.7 | 0.132 |
| Delly | 13.8 | 87.7 | 0.13 |

FDR is short for false discovery rate. Sensitivity and FDR are expressed as percentages.

### 3.5 Results on the illumina data of five individuals

We used five real datasets from the 1000 Genomes Project to compare Sprites with SVSeq2, LUMPY, Pindel and Delly. Alignment files for chromosome 20 of five individuals that were also used for the evaluation in the SVSeq2 paper (NA19311, NA19312, NA19313, NA19316 and NA19317), were able to be downloaded from ftp://ftp.ncbi.nlm.nih.gov/1000genomes/ftp/phase1/data/. These files have been generated by mapping paired-end reads with BWA ALN. Each file involves only one fragment library and the insert size of each library is specified in the header section of each BAM file. Since the standard deviation of reads is not given, it is computed by detection tools themselves. Reads of NA19311, NA19312 and NA19313 have the insert size of 400, and reads of the other two individuals have that of 213. The coverages of these five samples NA19311, NA19312, NA19313, NA19316 and NA19317 are 4.6, 5.4, 5.5, 6.2 and 6.2, respectively.

Lumpy, Pindel SVseq2 and Delly are run with default parameters. For Sprites, we set the mismatching rate and the minimum overlap length to 0.1 and 15, respectively; the same mean and the same standard deviation of the insert size are used as SVSeq2. Table 3 shows the sensitivity, FDR and *F*-score of each tool for each individual.

The *F*-scores of Sprites and SVseq2 are very close and better than others for NA19311 and NA19312. For NA19313, Sprites gets the highest *F*-score. For NA19316, the *F*-score of Sprites is slightly lower than Pindel and higher than others. For NA19317, the *F*-score of Sprites is ranked third.

### 3.6 Speed and memory usage

The dataset for the chromosome 20 of NA19312 at the coverage of about 5.4 to benchmark the speed and memory usage of these five tools. The chromosome has 63, 025, 520 bp. All tools were run as a single thread with default arguments on a 2-GHz 12-core Intel Xeon server with 512-Gb RAM. The time of extracting splitters and discordants for Lumpy is not included in the comparison. From Table 4, it can be clearly seen that Sprites runs much faster than the other tools with the least memory footprint. Sprites deals with only soft-clipping reads one by one without saving them in memory. The alignment is one of performance bottlenecks in the detection. Sprites uses Lemmas in the SVseq2 paper to determine target sequences. These sequences have the length of hundreds of base pairs on average. This keeps the alignment from requiring much time and memory to run.

## 4 Discussion

Sprites is a new deletion detection method based upon re-aligning soft-clipping reads. Results of tests on both simulated and real data show that Sprites is more sensitive in low-coverage data. The false discovery rate of Sprites is also low. As a result, Sprites performs the best overall in terms of the *F*-score in low-coverage data. Furthermore, there is also evidence that realigning soft-clipping reads is more effective than realigning their soft-clipped segments. Tests on simulated data show that Sprites is able to detect deletions with microhomologies and microinsertions at breakpoints in addition to blunt deletions. We have only used reads with soft-clipping at the 5′-end because 5′-end has generally higher quality than 3′-end.

The reason why Sprites does not work well in high-coverage data may be that Sprites does not directly use paired-end information for the detection. Since there is plenty of paired-end information in high-coverage data, such information could play a more important role in the detection in high-coverage data than in low-coverage data. Utilizing paired-end information in order to improve its performance in high-coverage data is one of directions of our future work.

The development of next generation sequencing technologies has increased read length from 36 to 100–200 bp. As a result, SV detection methods have shifted from pure PE methods to SR and to hybrid methods such PE + SR methods. As the trend continues, we believe that re-aligning split reads-based methodology will play an important role in SV detection in population-scale and cancer

**Table 3.** Comparison of results on five individuals

| Q1 | NA19311 | | | NA19312 | | | NA19313 | | | NA19316 | | | NA19317 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *S* | FDR | *F*-score | *S* | FDR | *F*-score | *S* | FDR | *F*-score | *S* | FDR | *F*-score | *S* | FDR | *F*-score |
| Sprites | 19.9 | 39.7 | 29.9 | 11.6 | 58.8 | 18.1 | 9.9 | 67.3 | 15.2 | 6.1 | 31.2 | 11.2 | 5.5 | 58.3 | 9.7 |
| SVseq2 | 19.9 | 27.1 | 31.3 | 11.6 | 56.2 | 18.3 | 8.8 | 77.1 | 12.7 | 6.1 | 35.3 | 11.1 | 6.1 | 64.3 | 10.4 |
| Lumpy | 8.8 | 59.0 | 14.5 | 3.3 | 88.3 | 5.1 | 2.2 | 94.0 | 3.2 | 0 | 90 | 0 | 0.6 | 84.6 | 1.2 |
| Pindel | 5.5 | 23.1 | 10.3 | 5.5 | 33.3 | 10.2 | 2.8 | 70.6 | 5.1 | 6.6 | 25.0 | 12.1 | 5.5 | 44.4 | 10.0 |
| Delly | 20.4 | 82.4 | 18.9 | 14.9 | 84.7 | 15.1 | 16.6 | 87.8 | 14.1 | 3.9 | 95.4 | 4.2 | 3.9 | 93.5 | 4.9 |

S is sensitivity and FDR is false discovery rate. We estimated that the coverages of NA19311, NA19312, NA19313, NA19316 and NA1917 are 4.6×, 5.4×, 5.5×, 6.2× and 6.2×, respectively.

**Table 4.** Comparison of five tools on the speed and memory usage

| Tool | Run time | Peak memory usage (kb) |
|---|---|---|
| Sprites | 45.2 s | 23552 |
| SVseq2 | 1 min 33 s | 12138640 |
| Lumpy | 1 min 8 s | 290800 |
| Pindel | 38 min 18 s | 15188048 |
| Delly | 1 min 25 s | 1158384 |

genome studies, because such methods are applicable to these types of data.

## Acknowledgements

We are grateful to Ryan Layer for his generously providing simulated data and giving advice. We would also like to thank Jeffrey Duerr for taking the time and effort to help us revise the paper.

## Funding

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61232001, 61379108 and 61370172.

*Conflict of Interest*: none declared.

## References

Abyzov,A. and Gerstein,M. (2011) Age: defining breakpoints of genomic structural variants at single-nucleotide resolution, through optimal alignments with gap excision. *Bioinformatics*, **27**, 595–603.

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.

Abyzov,A. *et al.* (2015) Analysis of deletion breakpoints from 1,092 humans reveals details of mutation mechanisms. *Nat. Commun.*, **6**, 7256.

Alkan,C. *et al.* (2011) Genome structural variation discovery and genotyping. *Nat. Rev. Genet.*, **12**, 363–376.

Baker,M. (2012) Structural variation: the genome's hidden architecture. *Nat. Methods*, **9**, 133–137.

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

Chiang,D.Y. *et al.* (2009) High-resolution mapping of copy-number alterations with massively parallel sequencing. *Nat. Methods*, **6**, 99–103.

Conrad,D.F. *et al.* (2010) Mutation spectrum revealed by breakpoint sequencing of human germline cnvs. *Nat. Genet.*, **42**, 385–391.

Consortium,G.P. *et al.* (2012) An integrated map of genetic variation from 1,092 human genomes. *Nature*, **491**, 56–65.

Faust,G.G. and Hall,I.M. (2012) Yaha: fast and flexible long-read alignment with optimal breakpoint detection. *Bioinformatics*, **28**, 2417–2424.

Feuk,L. *et al.* (2006) Structural variation in the human genome. *Nat. Rev. Genet.*, **7**, 85–97.

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

Hormozdiari,F. *et al.* (2010) Next-generation variationhunter: combinatorial algorithms for transposon insertion discovery. *Bioinformatics*, **26**, i350–i357.

Jiang,Y. *et al.* (2012) Prism: pair-read informed split-read mapping for base-pair level detection of insertion, deletion and structural variants. *Bioinformatics*, **28**, 2576–2583.

Korbel,J.O. *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.

Langmead,B. and Salzberg,S.L. (2012) Fast gapped-read alignment with bowtie 2. *Nat. Methods*, **9**, 357–359.

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

Li,H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *ArXiv e-Prints*

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

Li,H. *et al.* (2009) The sequence alignment/map format and samtools. *Bioinformatics*, **25**, 2078–2079.

Luo,J. *et al.* (2015a) Epga: de novo assembly using the distributions of reads and insert size. *Bioinformatics*, **31**, 825–833.

Luo,J. *et al.* (2015b) Epga2: memory-efficient de novo assembler. *Bioinformatics*, **31**, 3988–3990.

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. *Bioinformatics*, **26**, 841–842.

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

Rizk,G. *et al.* (2014) Mindthegap: integrated detection and assembly of short and long insertions. *Bioinformatics*, **30**, 3451–3457.

Schröder, J. *et al.* (2014) Socrates: identification of genomic rearrangements in tumour genomes by re-aligning soft clipped reads. *Bioinformatics* **30**, 1064–1072.

Sindi,S. *et al.* (2009) A geometric approach for classification and comparison of structural variants. *Bioinformatics*, **25**, i222–i230.

Sindi,S.S. *et al.* (2012) An integrative probabilistic model for identification of structural variation in sequencing data. *Genome Biol.*, **13**, R22.

Suzuki,S. *et al.* (2011) Clipcrop: a tool for detecting structural variations with single-base resolution using soft-clipping information. *BMC Bioinformatics*, **12**(**Suppl 14**), S7.

Wang,J. *et al*. (2011) Crest maps somatic structural variation in cancer genomes with base-pair resolution. *Nat. Methods*, **8**, 652–654.

Weischenfeldt,J. *et al*. (2013) Phenotypic impact of genomic structural variation: insights from and for human disease. *Nat. Rev. Genet*., **14**, 125–138.

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.

Yoon,S. *et al*. (2009) Sensitive and accurate detection of copy number variants using read depth of coverage. *Genome Res*., **19**, 1586–1592.

Zhang,J. *et al*. (2012) An improved approach for accurate and efficient calling of structural variations with low-coverage sequence data. *BMC Bioinformatics*, **13(Suppl 6)**, S6.

Zhang,J. and Wu,Y. (2011) Svseq: an approach for detecting exact breakpoints of deletions with low-coverage sequence data. *Bioinformatics*, **27**, 3228–3234.