BIOINFORMATICS ORIGINAL PAPER

Vol. 30 no. 18 2014, pages 2559–2567
doi:10.1093/bioinformatics/btu360

Genome analysis

Advance Access publication May 29, 2014

Reference-free prediction of rearrangement breakpoint reads

Edward Wijaya[1,*], Kana Shimizu[2], Kiyoshi Asai[2,3] and Michiaki Hamada[2,4,*]

[1]Immunology Frontier Research Center, Osaka University, 3-1 Yamadaoka, Suita, Osaka 565-0871, [2]Computational Biology Research Center, National Institute of Advanced Industrial Science and Technology, 2-4-7 Aomi, Koto-ku, Tokyo 135-0064, [3]Graduate School of Frontier Sciences, University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa, Chiba 277-8562 and [4]Department of Electrical Engineering and Bioscience, Faculty of Science and Engineering, Waseda University, 55N–06–10, 3–4–1, Okubo Shinjuku-ku, Tokyo 169–8555, Japan

Associate Editor: John Hancock

ABSTRACT

Motivation: Chromosome rearrangement events are triggered by atypical breaking and rejoining of DNA molecules, which are observed in many cancer-related diseases. The detection of rearrangement is typically done by using short reads generated by next-generation sequencing (NGS) and combining the reads with knowledge of a reference genome. Because structural variations and genomes differ from one person to another, intermediate comparison via a reference genome may lead to loss of information.

Results: In this article, we propose a reference-free method for detecting clusters of breakpoints from the chromosomal rearrangements. This is done by directly comparing a set of NGS normal reads with another set that may be rearranged. Our method SlideSort-BPR (breakpoint reads) is based on a fast algorithm for all-against-all comparisons of short reads and theoretical analyses of the number of neighboring reads. When applied to a dataset with a sequencing depth of 100×, it finds ~88% of the breakpoints correctly with no false-positive reads. Moreover, evaluation on a real prostate cancer dataset shows that the proposed method predicts more fusion transcripts correctly than previous approaches, and yet produces fewer false-positive reads. To our knowledge, this is the first method to detect breakpoint reads without using a reference genome.

Availability and implementation: The source code of SlideSort-BPR can be freely downloaded from https://code.google.com/p/slidesort-bpr/.

Contact: ewijaya@ifrec.osaka-u.ac.jp or mhamada@waseda.jp

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

Received on November 5, 2013; revised on May 6, 2014; accepted on May 21, 2014

1 INTRODUCTION

It has been discovered that cancer cells include chromosomal rearrangements, which are large-scale modifications of the genome, and those modifications are thought to be responsible, at least in part, for the irregular growth of cancer cells. Although it is known that these chromosomal rearrangements often lead to translocations and fusions of genes (Albertson *et al.*, 2003; Yates and Campbell, 2012; Zhang *et al.*, 2009), it has not yet been determined how the somatic genome rearrangement works at the molecular level.

A unit of the mosaic structure of a rearrangement is called a *breakpoint*. A breakpoint can be represented as a pair of genomic locations that do not appear in the reference genome but occur in the rearranged genome. Typical rearrangement events generate single breakpoints, including breakage-fusion-bridge cycles, non-homologous end joining and homologous recombination-based repair (Stephens *et al.*, 2011), whereas complex rearrangement events produce multiple breakpoints (Liu *et al.*, 2012). These rearrangements have been suggested to cause little or no loss of genomic material and are likely to produce functional gene fusions (Berger *et al.*, 2011; Kannan *et al.*, 2011).

Next-generation sequencing (NGS) technologies have enabled the analysis of individual genomes at unprecedented depth with low cost (Metzker, 2010). However, these NGS technologies have created new difficulties in examining cancer genomes to determine the signatures of breakpoint junctions unambiguously at the sequence level (Onishi-Seebacher and Korbel, 2011; Robinson, 2010). To detect large-scale genomic variations, several tools have been developed for detecting chromosomal rearrangements from NGS sequences. These tools include deFuse (McPherson *et al.*, 2011), FusionMap (Ge *et al.*, 2011) and others (Wang *et al.*, 2012) for detecting gene fusions, and BreakDancer (Chen *et al.*, 2009), CREST (Wang *et al.*, 2011) and others (Medvedev *et al.*, 2009) for detecting structural variation (SV). The foundation of these methods is locating breakpoints by mapping the NGS reads to a *reference genome*.

The human reference genome, while extraordinarily useful, has inherent shortcomings that may hinder the task of characterizing human cancer because it is still incomplete and, as the 1000 Genomes Project and other studies have revealed, there are significant numbers of genome differences between individuals (Rosenfeld *et al.*, 2012). Complexity also emerges from SVs among individuals. SV accounts for a larger fraction of the diversity between persons than any other factor (Altshuler *et al.*, 2010; Li *et al.*, 2011). This complexity makes it difficult to map short reads to the reference genome, and the resulting mismapped or unmapped reads can lead to loss of important information (Chen *et al.*, 2011; Kidd *et al.*, 2010).

One possible approach for detecting breakpoints from an NGS dataset without knowledge of a reference genome is to use genome assembly first and then compare two assembled genomes to find breakpoints. Many studies, however, have

*To whom correspondence should be addressed.

suggested difficulties for genome assembly arising from the variability of genomes (Alkan *et al.*, 2011; Li *et al.*, 2010), and few attempts have been made to detect breakpoints after assembling short reads.

To overcome the aforementioned limitations, we present a new method, SlideSort-BPR (breakpoint reads), which detects chromosomal breakpoints by directly comparing two sets of NGS reads from different kinds of cells without mapping them to a reference genome and without assembling reads. SlideSort-BPR determines the reads associated to the breakpoints by finding the groups of reads that are 'unbalanced' between two sets of samples; this determination is based on a probabilistic model of the number of similar reads (neighborhood reads). One might think that finding neighborhood reads by pairwise comparison is computationally too expensive for a large number of NGS reads. However, we have overcome this difficulty by using an efficient algorithm (Shimizu and Tsuda, 2011) that can enumerate all similar pairs within a given edit distance in a computational time linear on the number of reads. As a result, SlideSort-BPR can efficiently detect the NGS reads associated with breakpoints between samples from normal cells and samples from cancer cells.

2 APPROACH

Given two sets of NGS reads (e.g. from normal and cancer cells), our method consists of two main steps. (See Fig. 1 for an illustration of the overall method.) In the first step, similar reads within a certain edit distance of each read are taken as the neighborhood reads. To avoid the time-consuming process of all-against-all read comparisons, we extended the SlideSort algorithm (Shimizu and Tsuda, 2011), which lists all similar pairs of reads within a given edit distance in a computational time linear on the number of reads. This efficiently finds a given read's neighborhood in terms of edit distance, both within and across

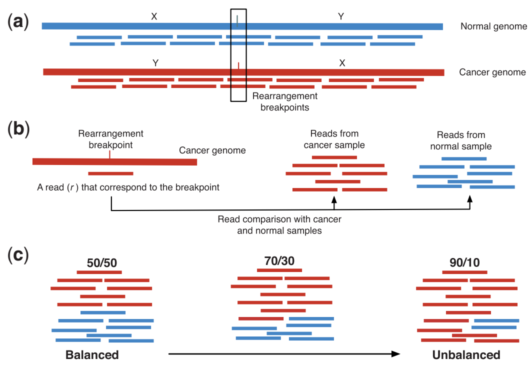

Fig. 1. (**a**) Cancer genomes (Y, X) consist of rearranged segments from normal genomes (X, Y). The breakpoints are the positions where the rearrangements occur (i.e. a place between X and Y). From these genomes we sample the reads. Those reads that correspond to a breakpoint are called breakpoint reads. (**b**) To detect a breakpoint read r, we compare it against all the reads from both normal and cancer samples. (**c**) The comparison will lead to several possible scenarios. Read r will have a neighborhood ratio that ranges from *balanced* to *unbalanced*. Unbalanced reads determine the breakpoints. Non-breakpoint reads are expected to be balanced because they are included in both genomes

the two sets of reads. In the second step, the numbers of neighborhood reads in the two sets of reads are compared. The neighborhood reads of each read have different compositions. If the ratio of the number of neighborhood reads between the two sets is close to zero, we call it an *unbalanced* (overrepresented) read. Each unbalanced read is expected to include a breakpoint.

In addition to the above, we use theoretical analysis of degree distribution to estimate false- and true-positive predictions, which is useful for choosing threshold parameters in our method and for designing experimental settings for sequencing.

3 METHODS

3.1 Neighborhood and degree of a read

In this section, we provide the formal definition of a read's neighborhood; this is the core concept of our approach.

For a read r and a set of reads S, we define the *neighborhood* of r and *degree* (the number of neighboring reads) of r by

$$\mathcal{N}_{S,d}(r) := \{t \in S \mid \textsc{EditDist}(r,t) \le d\}, \tag{1}$$

$$\deg_{S,d}(r) := |\mathcal{N}_{S,d}(r)|, \tag{2}$$

where EditDist gives the edit distance (also called the Levenshtein distance) (Levenshtein, 1966) between reads r and t, which is defined by the minimum number of single-character edits (insertion, deletion or substitution) required to transform r into t; d is a threshold for the distance; and $|\mathcal{N}_{S,d}(r)|$ denotes the number of reads in $\mathcal{N}_{S,d}(r)$. In this formulation, r is not necessarily in S.

3.2 Criteria for unbalanced reads

Given input from two read sets S and S', we can use the following three conditions to predict highly unbalanced reads in the set S. In the following, we will focus on a read $r \in S$ (the results also hold for $r \in S'$).

C1 For a parameter $M > 0$, $d_r \ge M$ where $d_r := \deg_{S,d}(r)$. This condition means that we discard a read with small degree because it would have many sequencing errors.

C2 For a parameter $\tau \in [0, 1]$

$$\alpha(r)\left(:= \frac{d'_r}{d_r}\right) \le \tau, \text{ where } d'_r := \max_{r' \in \mathcal{N}_{S',d}(r)} \deg_{S',d}(r')$$

This condition means that the degree of the target read r in S is larger than the degree of every read in S' similar to the read r. That is, the read r is unbalanced (overrepresented).

C3 For a parameter $\tau \in [0, 1]$

$$\beta(r) \le \tau, \text{ where } \beta(r) := \max_{t \in \mathcal{N}_{S,d}(r)} \alpha(t)$$

This condition means that not only the target read r but also every neighborhood read (in S) of r should be unbalanced.

The value M and μ in Criterion 1 is important to control true-positive and false-positive predictions. See Section 3.6. The values of α and β refer to the ratio of the degrees, thus the closer τ is to 0, the more unbalanced the reads. Note that C2 always holds if C3 holds.

3.3 An efficient algorithm for detecting unbalanced reads

In the criteria described in Section 3.2, computing the degree of each read is central to determining whether the neighborhood is unbalanced. To achieve this aim, we constructed an algorithm (Algorithm 1) based on SlideSort (Shimizu and Tsuda, 2011) to perform an exhaustive all-pair similarity search for every read occurring in both samples with a predetermined edit distance threshold. When the number of reads is huge, designing an efficient algorithm to satisfy our criteria is not trivial because naive pairwise similarity search is time-consuming. Typically, the running time complexity is $O(N^2)$, where N is the total number of reads. SlideSort efficiently reduces the time complexity to linear order by using sorting and a pattern growth algorithm (Shimizu and Tsuda, 2011).

Algorithm 1 illustrates our approach. In this pseudocode, the procedure FindUnbalancedReads finds all the unbalanced reads in S in the following manner. The function calcDegree (defined in Line 15 and used in Lines 5 and 6) computes the degrees of each read within the sets S and S'. Then, the function filterReads filters reads by using the thresholds M and τ without loss of prediction sensitivity. Subsequently, the functions computeAlpha (Line 9) and computeBeta (Line 10) compute the $\alpha(r)$ and $\beta(r)$ values in Criteria C2 and C3 for all $r \in S$.

These functions use SlideSort as their core to perform fast pairwise string comparison given a minimum edit distance (d). SlideSort(S, d) [resp. SlideSort(S, S', d)] exhaustively enumerates read pairs $(r, r') \in S \times S$ [resp. $(r, r') \in S \times S'$] with edit distance no more than d. We then process each pair of reads enumerated by SlideSort (Lines 17–20; Lines 29–33; Lines 41–45). When the number of input reads is huge, it is impossible to store all the information of pairs of reads within a given edit distance, so SlideSort is used four times (Lines 5, 6, 9 and 10) in Algorithm 1.

After applying the criteria in Section 3.2, Algorithm 1 outputs a list of unbalanced reads (R in Line 11) that are expected to correspond to breakpoints; in this list, there might be several reads corresponding to the same break point (Line 11). A post-processing procedure (the function connectedComponent in Line 12) summarizes those reads by clustering them, so that each cluster corresponds to one specific breakpoint. This is done by identifying the shared neighboring reads and forming the *connected component*.

Note that the algorithm is asymmetric (for S and S'). Because, in most cases, we would like to predict unbalanced reads for given S and S', both SlideSort-BPR(S, S', d, M, τ) and SlideSort-BPR(S', S, d, M, τ) should be executed.

Algorithm 1 SlideSort-BPR(S, S', d, M, τ)

```
1: $S$: target read set; $S'$: control read set;
2: $d$: edit distance                                  ▷ Define neighborhood
3: $M$: threshold in C1; $\tau$: threshold in C2/C3
4: procedure findUnbalancedReads($S$, $S'$, $d$, $M$, $\tau$)
5:     $\deg_S \leftarrow$ calcDegree($S$, $d$)
6:     $\deg_{S'} \leftarrow$ calcDegree($S'$, $d$)
7:     $\tilde{S} \leftarrow$ filterReads($S$, $\deg_S$, $M$)
8:     $\tilde{S}' \leftarrow$ filterReads($S'$, $\deg_{S'}$, $\tau M$)
9:     $\alpha \leftarrow$ computeAlpha($\tilde{S}$, $\tilde{S}'$, $\deg_S$, $\deg_{S'}$, $d$)
10:    $\beta \leftarrow$ computeBeta($\tilde{S}$, $\tilde{S}'$, $\alpha$, $d$)
11:    $R \leftarrow \{r \in \tilde{S} | \beta(r) \leq \tau\}$
12:    $\mathcal{C} \leftarrow$ connectedComponent($R$, $d$)
13:    return $\mathcal{C}$
14: end procedure
15: function calcDegree($S$, $d$)
16:    $\deg_S(r) \leftarrow 0$ for all $r \in S$
17:    while $((r, r') \leftarrow$ SlideSort($S$, $d$)) do
18:        $\deg_S(r) \leftarrow \deg_S(r) + 1$
19:        $\deg_S(r') \leftarrow \deg_S(r') + 1$
20:    end while
21:    return $\deg_S$
22: end function
23: function FilterReads($S$, $\deg_S$, $\gamma$)
24:    $\tilde{S} \leftarrow \{r \in S | \deg_S(r) \geq \gamma\}$
25:    return $\tilde{S}$
26: end function
27: function computeAlpha($S$, $S'$, $\deg_S$, $\deg_{S'}$, $d$)
28:    $\alpha(r) \leftarrow 0$ for all $r \in S$
29:    while $((r, r') \leftarrow$ SlideSort($S$, $S'$, $d$)) do
30:        if $\alpha(r) < \deg_{S'}(r')$ then
31:            $\alpha(r) \leftarrow \deg_{S'}(r')$
32:        end if
33:    end while
34:    for $r \in S$ do
35:        $\alpha(r) \leftarrow \alpha(r)/\deg_S(r)$
36:    end for
37:    return $\alpha$
38: end function
39: function computeBeta ($S$, $S'$, $\alpha$, $d$)
40:    $\beta(r) \leftarrow \alpha(r)$ for all $r \in S$
41:    while $((r, r') \leftarrow$ SlideSort($S$, $d$)) do
42:        if $\beta(r) < \alpha(r')$ then
43:            $\beta(r) \leftarrow \alpha(r')$
44:        end if
45:    end while
46:    return $\beta$
47: end function
48: function SlideSort($S$, $d$)
49: return $(r, r') \in S \times S$ where EditDistr($r, r' \leq d$) Return all pairs one-
    by-one
50: end function
51: function SlideSort($S$, $S'$, $d$)
52: return $(r, r') \in S \times S'$ where EditDistr, ($r, r' \leq d$) ▷ Return all pairs
    one-by-one
53: end function
54: function ConnectedComponent($R$, $d$)
55:    return a set of connected components in $R$
56: end function
```

3.4 Splitting long reads into several subreads

It is difficult to find unbalanced reads for relatively long reads because the degrees of long reads are smaller than those of short reads (cf. Section 3.5). By splitting one read into several (disjoint) reads, however, our method is also applicable to longer reads (e.g. a read of 120 bp). In this case, to use the information of longer reads effectively, we retain the information about the origin of each split read because that information is used in the clustering procedure. Specifically, every pair of subreads that come from the identical read is connected with the edge before we find connected component in ConnectedComponent (the Line 12 in Algorithm 1), which is expected to provide more reliable clusters of breakpoint reads. Note that, for example, the case in which every 120 bp read is split into four subreads (30 bp×4) is expected to achieve a similar performance to the case of 30 bp reads with the same coverage depth. This functionality is already implemented in our software, SlideSort-BPR, and is used in our computational experiments (cf. Section 4.1).

3.5 Theoretical distribution of degrees

Here we describe probabilistic models for the distribution of a read's neighborhood given an edit distance threshold (d), a level of sequencing error (e), coverage depth of reads (c) and length of reads (l); these probabilistic models will be useful for experimental design or determination of thresholds in our algorithm, as described later (cf. Section 3.6).

Suppose we have N reads of length l with uniform error rate e randomly sampled from a reference sequence of length L. Let c denote the coverage depth of the reads, so $N \cdot l = c \cdot L$ holds. The neighborhood degree of each read is determined by the edit distance threshold d [see Equation (1)]. With the assumptions that the reads are uniformly sampled and that the reference sequence includes no repeats, the probability $p(n)$ of a degree-n ($n = 0,1,\cdots$) read can be computed as follows.

If the error rate of each read is equal to 0 ($e = 0$), then it is easily seen that the probability of degree n is

$$p(n;\, d \geq 0,\, e = 0) = \binom{N}{n}\left(\frac{1+2\lfloor d/2\rfloor}{L}\right)^{n}\left(1-\frac{1+2\lfloor d/2\rfloor}{L}\right)^{N-n} \tag{3}$$

where $\binom{N}{n}$ is the binomial coefficient. That is, the degree has a binomial distribution with parameters $\alpha = \frac{1+2\lfloor d/2\rfloor}{L}$ and N: degree $\sim B(N,\alpha)$.

Because a binomial distribution $B(N,\alpha)$ has mean and variance $N\alpha$ and $N\alpha(1-\alpha)$, respectively, the mean and variance of this distribution can be analytically computed as

$$\begin{aligned}\mu &= \frac{N}{L}\left(2\left\lfloor\frac{d}{2}\right\rfloor+1\right) = \frac{c}{l}\left(2\left\lfloor\frac{d}{2}\right\rfloor+1\right),\\ \sigma^2 &= \frac{c}{l}\left(2\left\lfloor\frac{d}{2}\right\rfloor+1\right)\left(1-\frac{1+2\lfloor d/2\rfloor}{L}\right) \approx \frac{c}{l}\left(2\left\lfloor\frac{d}{2}\right\rfloor+1\right).\end{aligned} \tag{4}$$

Moreover, when $N\alpha$ is large, the degree follows a normal distribution with mean μ and variance σ^2: degree $\sim N(\mu,\sigma^2)$.

If the error rate is not equal to 0 and $d = 0$, the distribution can be represented as a *mixed* distribution with respect to the number of errors in the read:

$$p(n;\, d=0,\, e \geq 0) = \sum_{k\geq 0}\left[p_k \cdot p^{(k)}(n;\, d=0,\, e\geq 0)\right]. \tag{5}$$

Here, p_k is the probability that a read with length l contains k erroneous nucleotides

$$p_k = p_k(l, e) = \binom{l}{k}(1-e)^{l-k}e^k \tag{6}$$

and $p^{(k)}(\cdot)$ is the conditional probability of observing degree n in a read r that has k errors:

$$p^{(k)}(n;\, d=0,\, e\geq 0) = \sum_{m\geq 0} p(n+m;\, d=0,\, e=0)\binom{n+m}{m} r_k^n \cdot (1-r_k)^m, \tag{7}$$

where $r_k = r_k(l, e) = (1-e)^{l-k}(e/3)^k$ (i.e. the probability that k nucleotides are substituted with the same nucleotides as r because of the error; see Supplementary Fig. S1a). The probability $p(n+m;\, d = 0,\, e = 0)$ is given by Equation (3).

For more general cases, refer to Supplementary Section S1.

In these formulations, we assumed that there are no repeated regions in the reference genome from which reads are sampled. The empirical results are generated by first performing single nucleotide shuffling on the *Beta Vulgaris* BAC sequences followed by simulating reads with sequencing error. Figure 2 illustrates the consistency between the theoretical estimation and the empirical results. We give graphs for coverages ranging from 10× to 200×, error rates from 0 to 4% and edit distances from 0 to 4 in Supplementary Figures S2 and S3.

3.6 Estimations of false- and true-positive rates with respect to reads

For a positive integer m, we introduce

$$F_m := \sum_{k\geq 0}\left\{p_k \cdot p^{(k)}(0) \times \left(\sum_{n\geq m} p_k \cdot p^{(k)}(n)\right)\right\} \tag{8}$$

$$T_m := \sum_{k\geq 0}\sum_{n\geq m} p_k \cdot p^{(k)}(n), \tag{9}$$

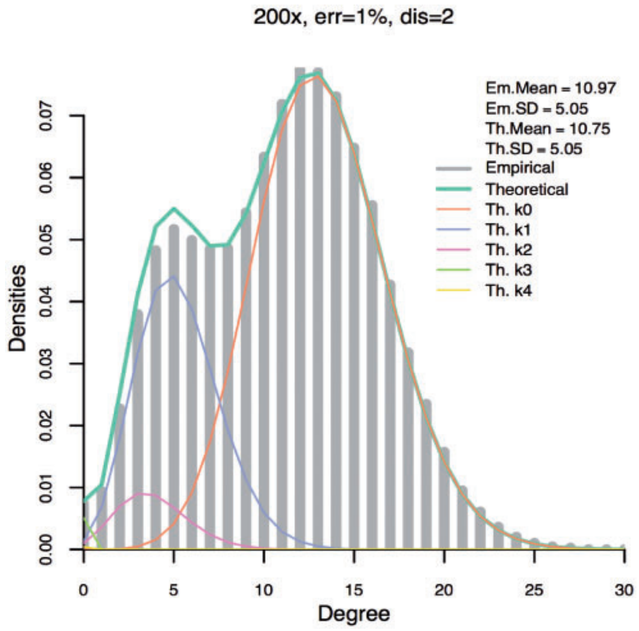

Fig. 2. Correspondence between degree distributions derived from theoretical estimation using $p(n;\, d \geq 0,\, e \geq 0)$ [Supplementary Equation (S10)] and from empirical estimation. The thin lines (orange, blue, purple, light green and yellow) refer to theoretical distributions with a given number ($k = 0,1,2,3,4$) of erroneous nucleotides [see Supplementary Equation (S11)]; gray bars show the empirical distribution. The total of all the lines for different values of k add up to the bold green line, which shows the overall theoretical distribution. In this figure, we set $l = 36$. See Supplementary Figures S2 and S3 for other conditions

where p_k is defined in Equation (6), and $p^{(k)}(n)$ is defined in Equations (7) and Supplementary Equations (S11) and (S12). These two values are interpreted in the following way.

Suppose two read sets S and S' are independently sampled from the *same* reference genome, so ideally there are no unbalanced reads between two datasets. Then, F_m is the joint probability that a (target) read $r \in S$ satisfies Criterion C1 with $M = m$, and there is no neighborhood of r in the other set S'. Those reads are likely to be false-positive predictions even if we set very low thresholds (say 0) for τ in Criteria C2 and C3, and so F_m is a rough estimation of the false-positive rate, where false positive indicates the predicted unbalanced read that does not include any breakpoint. (Note that $p_k \cdot p^{(k)}(0)$ is the probability that the degree of r in S is zero, and $\sum_{n\geq m} p_k \cdot p^{(k)}(n)$ is the probability that the degree of r in S' is larger than m, leading that r is predicted as an unbalanced read, i.e. false positive.) The actual false-positive rates (false-positive rate with respect to breakpoints) should be lower than F_m, because our algorithm incorporates a heuristic method in which we use the information from neighborhood reads to reduce false-positive predictions (see Section 4).

T_m represents the probability that a read r satisfies Criterion C1, which is considered to be a rough estimation of the true-positive rate with respect to reads (because if r corresponds to a breakpoint, r should satisfy Criteria C2 and C3). Because several reads will overlap with one breakpoint, the true-positive rates with respect to each breakpoint should be larger than T_m (see Section 4).

It should be emphasized that computing the exact false- and true-positive rates with respect to read and breakpoint is intractable, which is why we have introduced the above rough estimations. Note also that the evaluation of breakpoint detection is usually conducted by using false- and true-positive rates with respect to breakpoints, which we also use in our experiments (cf. Section 3.8).

In Supplementary Tables S1 and S2, we summarize F_m and T_m for every combination of conditions: $l = 36, 30$, $e = 1, 2, 4$, $c = 50, 100, 150, 200$ and $d = 0, 2, 4$. This will be useful for experimental design or setting the parameter M in the algorithm (see Section 4).

3.7 Datasets

3.7.1 Dataset 1: simulated reads and breakpoints The datasets are created in the following way. First, we obtained chr22 (50 MB) human genome (hg19; as a normal genome) data from the University of California Santa Cruz genome browser (Meyer *et al.*, 2013). Subsequently, we created another rearranged genome by breaking the normal genome into 118 equal-sized regions and rearranging these regions randomly. Both genomes (original and rearranged) are preserved, as they are without repeat masking. In total, there are 236 (118×2) breakpoints to be detected from both normal and rearranged genomes. From both normal and rearranged genomes, we simulated short reads of lengths 30, 60, 120 and 300 bp using the *dwgsim* tool (http://sourceforge.net/projects/dnaa/). The coverage of these datasets ranged from 50× to 300×, with uniform error rates between 0 and 4%. In this dataset, all the differences between the two genomes are derived from the breakpoints only.

3.7.2 Dataset 2: curated rearrangement breakpoints We used dbCRID, which is a database of actual chromosomal rearrangements together with their related diseases (Kong *et al.*, 2011). From 210 inversion events in the database, we obtained only 34 events where the accuracy of the positioning of the chromosome rearrangement was at the nucleotide level (class A); the remaining events, with accuracy at the chromosomal (Class B) or gene level (Class C), were discarded.

The reads were simulated in STAMPY (Lunter and Goodson, 2011) from the real quality score in the SRR987699 dataset, which can be obtained from the National Center for Biotechnology Information Sequence Read Archive (NCBI-SRA) database. For every event, we generated reads of length 120 bp with 100× coverage for the rearranged genomes. The mean error rate of this dataset is 1.83×10^{-4} (computed from the quality scores).

3.7.3 Dataset 3: real dataset We also obtained real datasets that have been previously published in Kannan *et al.* (2011). These paired-end RNA-Seq datasets consist of samples from 20 human prostate cancer cases and 10 matched benign/normal cases. The paper also reported 32 gene fusions that are responsible for the cancer. The datasets can be obtained from the NCBI-SRA database with accession number SRP002628. In total, there are 30 libraries with a total size of 17 GB. From these libraries, we chose 10 libraries that contain both normal and cancer samples. The reads were generated using Illumina GAII. Each dataset has a 36 bp read length. In total, there were 344 749 894 reads.

3.8 Evaluation methods

In the evaluation of our procedure for simulated datasets (Dataset 1 and Dataset 2), a read is deemed to be true (correct) if it overlaps with the known (correct) breakpoints. A cluster (cf. $\mathcal{C}$ in Algorithm 1) is called a true cluster if at least 90% of the reads in the cluster are true.

For Dataset 1, we measured the predictive power of our method by looking at precision and recall. Precision is the fraction of predicted true clusters over all predictions, and recall corresponds to the fraction of breakpoints correctly predicted over all known breakpoints.

For the evaluation of curated rearrangement breakpoints obtained from the dbCRID database (i.e. Dataset 2), we also looked at the overlap between the predicted reads and the annotated breakpoints for each event to validate our prediction. However, as the number of clusters we predicted and the known breakpoints were few, we found that evaluation with precision and recall could be misleading. Hence, we present the absolute numbers of predicted clusters and breakpoints instead. Note that in experiments for Datasets 1 and 2, the genomic positions for the breakpoints are already known from the annotations. They are not inferred from our method. By using read simulators, dwgsim and Stampy, we can preserve the genomic positions in the simulated reads. Based on this information, we evaluate the precision and recall. The evaluation is then straightforward. When the predicted reads contain the breakpoint-annotated simulated reads, they are considered as correct reads.

For Dataset 3, the junction sequences of the 32 fusion genes responsible for the prostate cancer have been reported (Kannan *et al.*, 2011). For this dataset, we evaluated the performance by verifying whether at least 14 bp of junction reads we predicted intersected with the reported junction sequences.

4 RESULTS AND DISCUSSION

4.1 Results for a simulated dataset (Dataset 1)

Figure 3 shows the results for the simulated dataset, which indicate that for stringent criteria ($\tau \le 0.1$) on a 50× coverage dataset of 120 bp reads with a 4% error rate, the precision and recall were 89 and 57%, respectively. Lower error rates increase the precision. We found that an increase of coverage produced better results. This is because the criteria are more discriminating. When we applied our method to a 150× coverage dataset with 4% error, we obtained values of ~100 and 88% for precision and

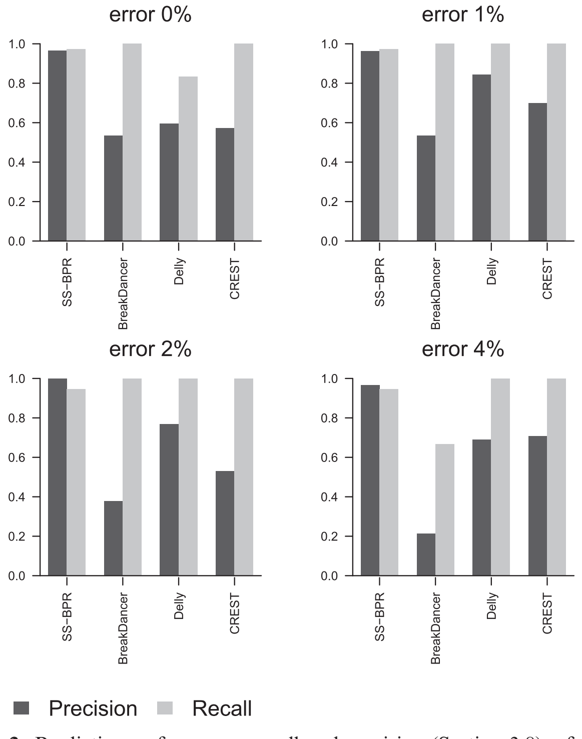

Fig. 3. Predictive performance, recall and precision (Section 3.8), of predicted breakpoints in Dataset 1 (which consists of simulated reads of 120 bp length and simulated breakpoints; Section 3.7.1). The horizontal axis indicates the τ threshold in the criteria (Section 3.2); the vertical axis shows the precision and recall rate. In these experiments, each read is split into four 30 bp subreads (Section 3.4). See Supplementary Figure S5 for the results with read lengths 30 and 60 bp

recall, respectively. For low- and high-coverage cases, recall is not affected by the value of τ, so our method performed consistently well. We also emphasize that *repeats* have little impact on our method because, in this experiments, we used chr22 as a reference genome without masking repeats (Section 3.7.1). Moreover, as we expected (see Section 3.4), similar results were obtained for 30 and 60 bp reads with identical coverage, in which every 60 bp read was also split into 30 bp subreads (Supplementary Fig. S11). This result suggests the effectiveness of our method for long reads.

In this experiment, we systematically set $M = \max\{\mu/2, 4\}$, where μ is the theoretical mean of the dataset (i.e. the mean value of degrees is computed from the theoretical distribution in Section 3.5 with a given coverage depth and error rate), because the thresholds consistently achieved reasonable F_m and T_m values; see the fourth and fifth columns in Supplementary Table S2.

Moreover, we have compared SlideSort-BPR with other methods including BreakDancer (Chen *et al.*, 2009), Delly (Rausch *et al.*, 2012) and CREST (Wang *et al.*, 2011) with the simulated dataset. See Supplementary Sections S2 and S3 for the details of the evaluation methods of each tool. In Figure 4, we exhibit comparison on 100× dataset and Supplementary Figure S10 on 50× dataset. In addition, we tried Velvet (Zerbino and Birney, 2008), one of the well-developed *de novo* assemblers, for detecting breakpoints, and evaluated the recall (Supplementary Fig. S11). These results indicate that on higher-coverage dataset (e.g. 100×), SlideSort-BPR performs better in precision and recall than other tools in detection of breakpoints. And on lower coverage (e.g. 50×), although our method performs worse in recall, still it maintains good precision.

Because our method does not predict the SV events *per se*, the comparison with the above SV prediction tools is done at the breakpoint reads (split reads) level. These reads are the intermediate results of these tools. Hence, this evaluation does not reflect the final performance of SV predictions. For predicting SVs, these tools use further steps (e.g. assembly and realignment of unmapped reads followed by minimum support counting). Notice that these methods aim to maximize sensitivity instead of precision.

The typical input for the SV prediction tools is the alignment file (SAM/BAM). In our experiment, the alignments are done using BWA as a front-end aligner with the default parameters. BreakDancer does not produce split reads, only SV region coordinates. We use Pindel (Ye *et al.*, 2009) to process BreakDancer output to obtain the split reads. When evaluating Velvet, we only considered recall as performance measure because it is difficult to evaluate precision; because Velvet is a general purpose assembly tool, it is not specifically designed for detecting breakpoints. There are no criteria applicable to remove false-positive findings in its result. Finding such criteria falls beyond the scope of this study. If the substring of a breakpoint read spans within 75 bp of the breakpoint position is contained in a contig, we deemed it to be a correct hit.

4.2 Results for curated rearrangement breakpoints (Dataset 2)

By our experiment using Dataset 2, we confirmed that SlideSort-BPR predicted few clusters (from one to eight) as candidates for breakpoints, but it still has good recall, predicting almost all the known breakpoints. See Supplementary Table S3 for details. This table shows the predictions of rearrangement breakpoints from SlideSort-BPR. They are ordered by their respective events and their related diseases. The rearrangement events occur in a specific chromosome as denoted by the karyotype. The annotation in dbCRID indicates the number of breakpoints and their positions in the genome for each event.

In this experiment, we split every read into 30 bp reads (×4) (Section 3.4) and used $d = 4$ and $\tau = 0.1$ because that threshold gave better results in our experiments on Dataset 1. We also set $M = 10$ because that threshold provides reasonable F_m and T_m values [Equations (8) and (9)]: $F_{10} = 5.92 \times 10^{-8}$ (related to false-positive findings) and $T_{10} = 0.982$ (related to true-positive findings) for the conditions in this experiment ($l = 30$, $c = 100$, $e = 1.83 \times 10^{-4}$ and $d = 4$).

4.3 Results for a real dataset (Dataset 3)

In applying our method to Dataset 3, we use data for only 10 prostate cancer cases and 10 corresponding matched cases. In

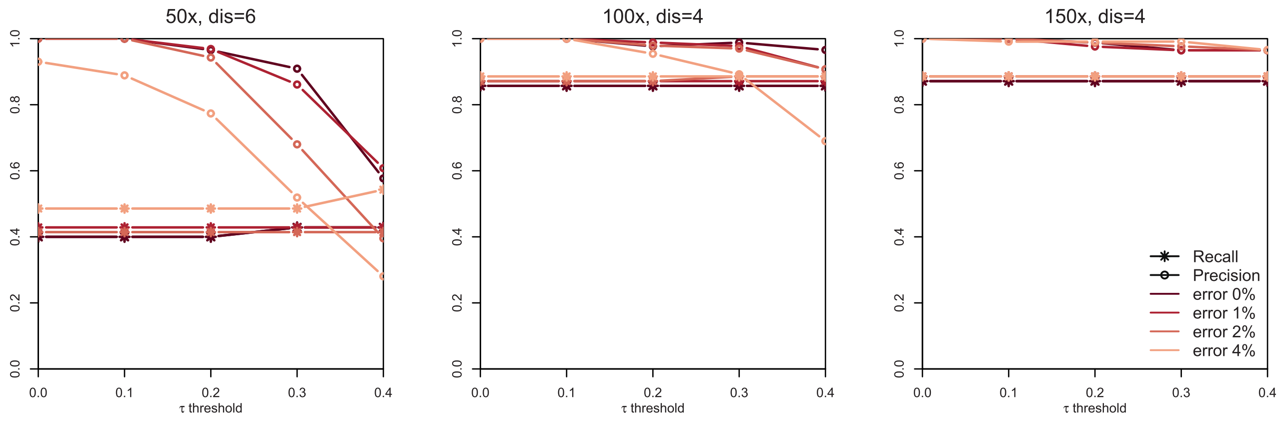

Fig. 4. Performance comparison of SlideSort-BPR with other tools for Dataset 1, read length 300 bp and 100× coverage. For SlideSort-BPR, we use the $\tau = 0.01$ as threshold and split reads of size 30 bp. The parameters for other tools are set to default. The comparison is done only for the breakpoint reads (split reads)

this experiment, all the neighborhoods are determined using an edit distance of $d = 4$. For Criterion C1, we use $M = 10$ because our theoretical analysis indicates that $M = 10$ provides fewer false-positive findings (see Supplementary Table S1), and for C2 and C3, we use $\tau = 0.1$ as the threshold.

We compared our method with other classes of tools for detecting genome rearrangements. The first class is the gene fusion prediction method, which is used by deFuse (McPherson *et al.*, 2011), TopHat-Fusion (Kim and Salzberg, 2011) and FusionMap (Ge *et al.*, 2011). The second class is a general-purpose SV prediction approach, which is used by CREST (Wang *et al.*, 2011) and BreakDancer. We apply the default parameters for all methods except for TopHat-Fusion, where we used the `–very-sensitive` option. BreakDancer and CREST used BWA (Li and Durbin, 2009) as the front-end aligners and were executed with the default parameters.

Table 1 lists the fusion genes predicted by our method and three other fusion-gene prediction methods (BreakDancer, DeFuse and TopHat-Fusion). On this dataset, both FusionMap and CREST produced no predictions. In addition, we also report the total number of predictions given by these methods. The table shows that our method predicted six fusion breakpoints, whereas BreakDancer, DeFuse and TopHat-Fusion predicted two, three and five breakpoints, respectively (Supplementary Fig. S6 shows the predictions common to these methods). Furthermore, our method gave fewer predictions than DeFuse and TopHat-Fusion and, although it gave more predictions than BreakDancer, our method discovers three times as many fusion genes. These results suggest that SlideSort-BPR gives a better trade-off between sensitivity and precision compared with these other methods, even though SlideSort-BPR does not use any knowledge of reference genomes.

It should be emphasized that the predictions of SlideSort-BPR might include other differences [e.g. copy number variation (CNV), insertion of novel sequences, some large segmental deletions or insertions in the genome] because our method considers the reads from these regions as having actual differences (i.e. unbalanced reads), whereas the other methods focus on finding only breakpoints.

One of the disadvantages of mapping-based methods for fusion detection is the difficulty of mapping (e.g. split-read mapping) when the reference genome and the genome sampled from the read differ significantly. FusionMap and CREST are based on mapping, and this might be one of the reasons that they failed to detect the reported fusion genes.

Strictly speaking, the assumption of uniform coverage is not generally satisfied for RNA-seq datasets. Our results, however, suggest that our method is applicable to those datasets from a practical viewpoint.

4.4 Computational time

Supplementary Figure S7 shows running times for SlideSort-BPR, DeFuse, CREST, TopHat-Fusion, FusionMap and BreakDancer. In this comparison, we used the dataset with sample ID SRR057649, which is a subset of all the datasets from accession SRP002628 and contains 8 029 570 reads. Our method took ~140 min. The three fusion prediction tools (deFuse, TopHat-Fusion and FusionMap) are based on mapping algorithms, and the times shown exclude the indexing step for their mapping tools. Note that our method does not have an indexing procedure.

4.5 Several remarks when using SlideSort-BPR

In this study, we assume that every read is uniformly sampled from genomes. However, we carefully handle the stochastic effect in sampling reads. This is addressed in the theoretical part of our study (cf. Section 3.5). In actual cases, however, there exists sequencing depth bias, which might affect the accuracy of our method. We emphasize that biases commonly observed in both cancer and normal read sets is expected to be canceled in the algorithm of SlideSort-BPR. For example, the depth bias by GC contents of genomes and/or sequence specific errors in Illumina sequencers (Nakamura *et al.*, 2011) are observed commonly in the both genomes, which will less affect to our method. Moreover, we experimentally confirmed that slight depth biases between normal and cancer genomes do not affect the results (Supplementary Fig. S9).

Table 1. List of fusion genes [experimentally validated by Kannan *et al.* (2011)] predicted by SlideSort-BPR (this work), DeFuse (McPherson *et al.*, 2011), TopHat-Fusion (Kim and Salzberg, 2011) and BreakDancer (Chen *et al.*, 2009)

SlideSort-BPR	DeFuse	TopHat Fusion	BreakDancer
Predicted fusion genes			
C12orf76-ANKRD13A	TMEM165-CLOCK	ACTR8-IL17RB	MTG1-LOC619207
KRTCAP3-IFT172	TMEM79-SMG5	NARG1-NDUFC1	TMEM165-CLOCK
NARG1-NDUFC1	SLC44A4-EHMT2	SLC44A4-EHMT2	
NCAPD3-JAM3		SLC16A8-BAIAP2L2	
TMEM165-CLOCK		ZNF606-C19orf18	
TMPRSS2-ERG			
Number of predictions			
16 K	71 K	149 K	1.3 K

Note: In DeFuse and BreakDancer, default parameters were used; in TopHat-Fusion, we used the `–very-sensitive` option. BreakDancer and CREST were run with the default parameters, and BWA is used as the front-end aligner.

Our method detects the breakpoints that appear in the sequencing reads. Nonetheless, it does not report the genomic position for a breakpoint, as our method does not use reference genome. Because of this characteristic, our method cannot directly predict SVs.

Because of the recent advent of new-generation technologies, such as PacBio RS (http://www.pacificbiosciences.com/), the length of reads is becoming longer, which can address the issue of repetitive region (Kinsella *et al*., 2011; Sboner *et al*., 2010). Our method can deal with long reads by splitting each read into several subreads (cf. Section 3.4). In our computational experiments, we confirmed that our method achieved enough performance for reads whose length is ~300, showing that SlideSort-BPR is more suitable to be applied for reads produced by, for example, Illumina sequencers rather than PacBio RS.

4.6 Potential applications and further directions

Although we focused on detecting breakpoint reads in this study, the approach proposed in this article will be applicable to other types of difference, such as CNVs (Riggs *et al*., 2012) and the insertion of novel virus genomes into human genomes. Moreover, potential applications of our method are not limited to screening the reads from personal genome data. It could also be used in the area of metagenomic analysis to compare several samples (reads) directly. In this case, our method will be useful because only a limited number of reference genomes are available and assembly for metagenomic data is more difficult than usual genome assembly (Mende *et al*., 2012).

To address the issue of longer reads described in Section 4.5, more sophisticated post-processing procedure with using the information of long read would be desirable. In this study, we proposed a simple clustering procedure for reads with the information of long reads. Another possible approach for effectively handling the information of longer read is that we use *approximate* similarity search to find neighborhood reads, instead of the exact similarity search algorithm used in this study.

5 CONCLUSION

In this work, we have presented SlideSort-BPR, a tool for performing direct comparisons of reads from different samples without using a reference genome. To our knowledge, this is the first method for detecting breakpoint reads without using a reference genome, although a paper was recently published that tries to detect mutations by direct comparison of NGS reads (Nordstrom *et al*., 2013). Computational experiments indicated that our approach is robust to the presence of repeats in the genome and to a wide range of sequencing errors (up to 4%). In extensive evaluations on simulated and real datasets, we showed that SlideSort-BPR can predict breakpoint reads with high accuracy.

ACKNOWLEDGEMENTS

We are grateful to Tatsuo Kitahashi (INTEC Inc.) for providing technical support for this project. We also thank the three anonymous reviewers whose comments helped improve and clarify this article. A part of this work was conducted while M.H. belonged to the University of Tokyo, who has joined to Waseda University since April 1, 2014.

Funding: This work was supported by a Grant-in-Aid for Scientific Research on Innovative Areas (221S002) from the Japanese Ministry of Education, Culture, Sports, Science and Technology.

Conflict of Interest: none declared.

REFERENCES

Albertson,D.G. *et al*. (2003) Chromosome aberrations in solid tumors. *Nat. Genet*., **34**, 369–376.
Alkan,C. *et al*. (2011) Limitations of next-generation genome sequence assembly. *Nat. Methods*, **8**, 61–65.
Altshuler,D. *et al*. (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–1073.
Berger,M. *et al*. (2011) The genomic complexity of primary human prostate cancer. *Nature*, **470**, 214–220.
Chen,G. *et al*. (2011) Revealing the missing expressed genes beyond the human reference genome by RNA-Seq. *BMC Genomics*, **12**, 590.
Chen,K. *et al*. (2009) BreakDancer: an algorithm for high-resolution mapping of genomic structural variation. *Nat. Methods*, **6**, 677–681.
Ge,H. *et al*. (2011) FusionMap: detecting fusion genes from next-generation sequencing data at base-pair resolution. *Bioinformatics*, **27**, 1922–1928.
Kannan,K. *et al*. (2011) Recurrent chimeric RNA enriched in human prostate cancer identified by deep sequencing. *Proc. Natl Acad. Sci. USA*, **108**, 9172–9177.
Kidd,J. *et al*. (2010) Characterization of missing human genome sequences and copy-number polymorphic insertions. *Nat. Methods*, **7**, 365–371.
Kim,D. and Salzberg,S. (2011) TopHat-Fusion: an algorithm for discovery of novel fusion transcripts. *Genome Biol*., **12**, 72.
Kinsella,M. *et al*. (2011) Sensitive gene fusion detection using ambiguously mapping RNA-Seq read pairs. *Bioinformatics*, **27**, 1068–1075.
Kong,F. *et al*. (2011) dbCRID: a database of chromosomal rearrangements in human diseases. *Nucleic Acids Res*., **29**, D895–D900.
Levenshtein,V.I. (1966) Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl*., **10**, 707–710.
Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.
Li,Y. *et al*. (2010) State of the art de novo assembly of human genomes from massively parallel sequencing data. *Hum. Genomics*, **4**, 271–277.
Li,Y. *et al*. (2011) Structural variation in two human genomes mapped at single-nucleotide resolution by whole genome de novo assembly. *Nat. Biotechnol*., **29**, 723–730.
Liu,P. *et al*. (2012) Mechanisms for recurrent and complex human genomic rearrangements. *Curr. Opin. Genet. Dev*., **22**, 211–220.
Lunter,G. and Goodson,M. (2011) Stampy: a statistical algorithm for sensitive and fast mapping of Illumina sequence reads. *Genome Res*., **21**, 936–939.
McPherson,A. *et al*. (2011) deFuse: an algorithm for gene fusion discovery in tumor RNA-Seq data. *PLoS Comput. Biol*., **7**, e1001138.
Medvedev,P. *et al*. (2009) Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods*, **6** (**Suppl. 11**), S13–S20.
Mende,D.R. *et al*. (2012) Assessment of metagenomic assembly using simulated next generation sequencing data. *PLoS One*, **7**, e31386.
Metzker,M. (2010) Sequencing technologies - the next generation. *Nat. Rev. Genet*., **11**, 31–46.
Meyer,L.R. *et al*. (2013) The UCSC Genome Browser database: extensions and updates 2013. *Nucleic Acids Res*., **41**, D64–D69.
Nakamura,K. *et al*. (2011) Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Res*., **39**, e90.
Nordstrom,K.J. *et al*. (2013) Mutation identification by direct comparison of whole-genome sequencing data from mutant and wild-type individuals using k-mers. *Nat. Biotechnol*., **31**, 325–330.

Onishi-Seebacher,M. and Korbel,J. (2011) Challenges in studying genomic structural variant formation mechanisms: the short-read dilemma and beyond. *Bioessays*, **33**, 840–850.

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

Riggs,E.R. *et al.* (2012) Towards an evidence-based process for the clinical interpretation of copy number variation. *Clin. Genet.*, **81**, 403–412.

Robinson,K. (2010) Application of second-generation sequencing to cancer genomics. *Brief Bioinform.*, **11**, 524–534.

Rosenfeld,J. *et al.* (2012) Limitations of the human reference genome for personalized genomics. *PloS One*, **7**, e40294.

Sboner,A. *et al.* (2010) FusionSeq: a modular framework for finding gene fusions by analyzing paired-end RNA-sequencing data. *Genome Biol.*, **11**, R104.

Shimizu,K. and Tsuda,K. (2011) SlideSort: all pairs similarity search for short reads. *Bioinformatics*, **27**, 464–470.

Stephens,P. *et al.* (2011) Massive genomic rearrangement acquired in a single catastrophic event during cancer development. *Cell*, **144**, 27–40.

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

Wang,Q. *et al.* (2012) Application of next generation sequencing to human gene fusion detection: computational tools, features and perspectives. *Brief. Bioinform.*, **14**, 506–519.

Yates,L. and Campbell,P. (2012) Evolution of the cancer genome. *Nat. Rev. Genet.*, **13**, 795–806.

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.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.

Zhang,F. *et al.* (2009) Complex human chromosomal and genomic rearrangements. *Trends Genet.*, **25**, 298–237.