Published online 22 November 2017 *Nucleic Acids Research, 2018, Vol. 46, No. 3* ***e18***
doi: 10.1093/nar/gkx1175

Jointly aligning a group of DNA reads improves accuracy of identifying large deletions

Anish M.S. Shrestha[1], Martin C. Frith[1,2,3], Kiyoshi Asai[1,2] and Hugues Richard[4,*]

[1]Department of Computational Biology and Medical Sciences, University of Tokyo, 5-1-5 Kashiwanoha, Kashiwa-shi, Chiba, Japan, [2]Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology (AIST), 2-3-26 Aomi, Koto-ku, Tokyo, Japan, [3]Computational Bio Big-Data Open Innovation Laboratory (CBBD-OIL), National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan and [4]Sorbonne Universités, UPMC Univ Paris 06, CNRS, IBPS, UMR 7238, Laboratoire de Biologie Computationnelle et Quantitative (LCQB), 4 place Jussieu, 75005 Paris, France

Received May 17, 2017; Revised September 06, 2017; Editorial Decision November 08, 2017; Accepted November 16, 2017

ABSTRACT

Performing sequence alignment to identify structural variants, such as large deletions, from genome sequencing data is a fundamental task, but current methods are far from perfect. The current practice is to *independently* align each DNA read to a reference genome. We show that the propensity of genomic rearrangements to accumulate in repeat-rich regions imposes severe ambiguities in these alignments, and consequently on the variant calls—with current read lengths, this affects more than one third of known large deletions in the C. Venter genome. We present a method to *jointly* align reads to a genome, whereby alignment ambiguity of one read can be disambiguated by other reads. We show this leads to a significant improvement in the accuracy of identifying large deletions (≥20 bases), while imposing minimal computational overhead and maintaining an overall running time that is at par with current tools. A software implementation is available as an open-source Python program called JRA at https://bitbucket.org/jointreadalignment/jra-src.

INTRODUCTION

The usual first step in analyzing high-throughput sequencing assays involves alignment of short DNA reads to a larger reference sequence (e.g. the reference genome, the genome of a matching somatic cell, or of a closely related species). This alignment step is crucial as it precedes important subsequent analyses such as the identification of sequence conservation between species (1), the estimation of allele fluctuation among a population of individuals (2), or the characterization of somatic and causal mutations in the genome of tumoral cells (3).

Standard alignment-based workflows independently align billions of reads, each a few hundred base-pairs long, onto a reference sequence in order to identify sequence variations such as single nucleotide variants, small indels, or structural variants (e.g. large deletions or translocations). *Split-alignments*, which are alignments where two different portions of a read align to disjoint genomic locations on the reference, are direct evidence of structural variants (SV), and is the focus of this work.

Accurately computing split-alignments remains a very challenging problem (4,5). The main difficulty stems from the highly repetitive nature of genomes and the propensity of genomic rearrangements to accumulate in the vicinity of repeats (6). Additionally, reads contain sequencing errors and library artefacts. These factors lead to ambiguities in pairwise alignments: *positional ambiguity* due to a portion of a read aligning to more than one region in the reference (Figure 1A, SI-S1.1), and *breakpoint ambiguity* due to microhomologies surrounding the SV locus (Figure 1B, SI-1.2). The former hinders the identification of true SV with certainty, while the latter is a major cause of redundancies in variant call sets (7). To estimate the severity of ambiguities, we examined the variants reported in the Venter genome (8). Our analysis reveals that, even under ideal conditions of no sequencing errors and very high coverage, 40% of deletions ≥32 bp cannot be identified with certainty by pairwise alignments of 100 bp-long reads (13% for paired-end reads, SI-S1.1). This alarming result shows that alignment ambiguity issues cannot be ignored.

These issues highlight two major shortcomings of current techniques based on split-alignments. First, all current split-aligners align reads *independently* of each other. They lack a principled way to effectively combine, the joint information contained in reads belonging to the same genomic region. Since those reads are in fact highly correlated, utilizing information from the group as a whole, can mitigate misalignment issues arising due to repeat-rich genomic

*To whom correspondence should be addressed. Tel: +33 1 4427 7325; Fax: +33 1 4427 7336; Email: hugues.richard@upmc.fr

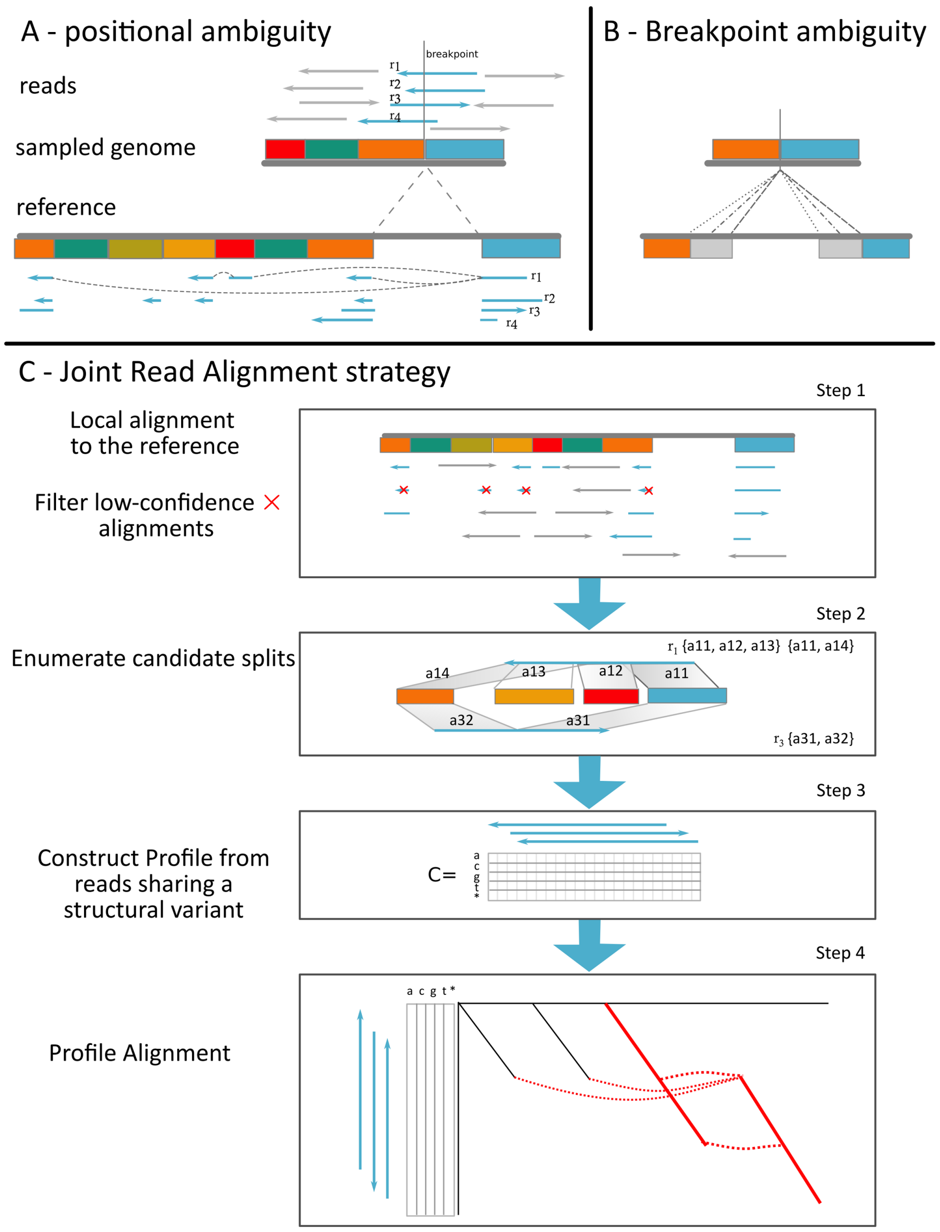
A - positional ambiguity
reads
sampled genome
reference
breakpoint
r1
r2
r3
r4
B - Breakpoint ambiguity
C - Joint Read Alignment strategy
Step 1
Local alignment to the reference
Filter low-confidence alignments
Step 2
Enumerate candidate splits
r1 {a11, a12, a13} {a11, a14}
a14
a13
a12
a11
a32
a31
r3 {a31, a32}
Step 3
Construct Profile from reads sharing a structural variant
C=
a c g t *
Step 4
Profile Alignment
a c g t *

Figure 1. (**A**) Example of a typical deletion locus. The left flank in the sampled genome possesses many similar regions in the reference (represented as blocks of similar colors), causing portions of read, for example r_1, to align to several positions, leading to ambiguity about the true deletion. (**B**) Ambiguity in identifying the exact breakpoint position. Here, the suffix of the left block matches the prefix of the right (colored in gray), making it impossible to pinpoint the exact deletion site. (**C**) The main steps of our Joint Read Alignment strategy, detailed in Section Workflow.

context of SVs or due to sequencing errors. Although local realignment and local assembly (9,10) combine information from multiple reads, they are limited to substitutions and small indels in a specific contiguous region, and cannot handle SVs that span large portions of the genome. The idea of borrowing information across reads has also been proposed in the context of resolving conflicting predictions from paired-end reads (11–13). However, these methods choose between candidate alignments, but themselves do not compute (split-) alignments. Also, as is the general case with non-split alignments of paired-end reads, they do not precisely identify the breakpoint of SVs or shed light on the sequence context at immediate locus of the SV.

The second shortcoming of current methods is that they do not properly account for alignment uncertainties that could be computed using probabilistic model of sequence alignments (14). Most aligners performing split-alignments, either report only a single best alignment or enumerate all high-scoring ones (15). To our knowledge, only LAST (16) reports confidence values (error probabilities) of split-alignments. The SV calling phase is also usually oblivious to alignment uncertainty, relying on *ad-hoc* thresholds such as the number of reads covering an SV. Some post-processors (e.g. CLEVER (12)) model alignment uncertainty, but they assume semi-global alignment of reads and cannot be directly applied to the case of split alignments.

Here we take a slight departure from the conventional align-and-call workflow, and propose a new framework of *jointly* aligning a group of reads identifying a common genomic structural variant. We define a statistically sound scoring scheme for joint alignment that models both evolutionary divergence and sequencing errors, and propose an alignment algorithm corresponding to this scheme. Our method also measures the uncertainty of each reported joint split-alignment, based on a probabilistic model of sequence alignment. This enables users to directly control the trade-off between sensitivity and specificity of variant prediction, rather than having to guess *ad-hoc* threshold values such as read support. Additionally, we show how to incorporate paired-end reads in our workflow by using pairing information to improve the confidence value of a prediction.

We demonstrate the advantages of our method over other split-aligners, by applying it to the problem of identifying medium to large-sized deletions ($\geq$20 bp) from typical human genome resequencing datasets, both simulated and real. We also contrast our technique of bolstering otherwise ambiguous split alignments by combining read group and paired-end information to the conventional method of detecting deletions through discordant alignments of paired-end reads. Quite surprisingly, we find that these conventional methods have lower sensitivity in practice, as they miss a majority of deletions in their attempt to account for variability in fragment size. We show that our method incurs only a small computational overhead and maintains an overall running time that is well within the range of contemporary methods.

MATERIALS AND METHODS

A joint alignment model for the detection of rearrangements

We begin by describing how our joint split-alignment problem can be formulated as a profile-to-sequence alignment problem.

Let us first consider the split-alignment problem in its basic form: given a reference genome $\mathcal{G}$ and a set $\mathcal{R}$ of reads originating from a region $\mathcal{H}$ in the sampled genome that contains an SV, we wish to find the split-alignment of $\mathcal{H}$ to $\mathcal{G}$ which determines the position of the SV as well as the sequence of $\mathcal{H}$ in the SV locus. Figure 1A shows a typical instance of this problem.

Note that this problem is different from the classical pairwise split-alignment in which an alignment is computed between each read and $\mathcal{G}$, rather than between $\mathcal{H}$ and $\mathcal{G}$.

If indeed we had a fairly long and accurate sequence of $\mathcal{H}$, we could score its pairwise alignment according to a scheme that accounts for the evolutionary divergence between $\mathcal{G}$ and $\mathcal{H}$. Such a score $S_{\text{genome}}(A; \mathcal{H}, \mathcal{G})$ is usually defined as the sum over contributions from individual aligned bases and indels that are penalised with affine cost functions. Other types of rearrangements, such as large indels or translocations are assigned specific penalties. SVs can be identified by finding an alignment that maximizes $S_{\text{genome}}(A; \mathcal{H}, \mathcal{G})$.

However, we only have at hand an approximate representation of $\mathcal{H}$ in the form of short and possibly erroneous reads, and our task is to determine the sequence of $\mathcal{H}$ and its alignment to $\mathcal{G}$. Towards this, let us assume first that the true multiple sequence alignment of $\mathcal{R}$ is known, and it spans ℓ columns. We can report at each position the number of letters from $\mathcal{D} = \{\texttt{a}, \texttt{c}, \texttt{g}, \texttt{t}, *\}$ ($*$ denotes the gaps occurring within reads) and store it in a $(5 \times \ell)$ matrix C. Let $\mathcal{H}$ be an ℓ-length string over $\mathcal{D}$, with $\mathcal{H}_i$ corresponding to column i in C. This string represents the reconstructed sequence of the sampled genome around the SV locus. We score $\mathcal{H}$ and an alignment A of the matrix C to $\mathcal{G}$ as:

$$S(\mathcal{H}, A; \mathcal{G}, \mathcal{R}) = S_{\text{genome}}(A; \mathcal{H}, \mathcal{G}) + \sum_{i=1}^{\ell} \sum_{x \in \mathcal{D}} C_{xi} \times S_{\text{sequencer}}(\mathcal{H}_i, x), \tag{1}$$

where S_{genome} accounts for the $\mathcal{H}$-to-$\mathcal{G}$ evolutionary divergence, and $S_{\text{sequencer}}$ accounts for the base-pair (dis-) similarities expected due to the sequencer.

The parameters used in the scoring schemes are shown in Figure 2A. S_{genome} consists of a substitution matrix, affine gap penalties for small indels, and a constant penalty for large splits, and $S_{\text{sequencer}}$ consists of a substitution matrix and linear gap penalties. The values in both scoring schemes can be adjusted to reflect the relative importance of the reference genome and of the read sequences. Our split penalty is similar to previous work for breakpoint detection in the context of pairwise alignment, for instance in (17). We show an application of our scoring model on a toy example in Figure 2B. Note that the choice of linear gap penalties allows us to express ((1)) equivalently as the score of C-to-$\mathcal{G}$ alignment, with each column of C treated independently. A formal description of the scoring scheme is provided in SI-S2, and an exact alignment algorithm in SI-S3.

We have made two non-trivial assumptions so far: that we can extract from the massive set of reads, those originating from the region $\mathcal{H}$, and that we can construct their multiple sequence alignment. In the next section, we provide approximate solutions to these by incorporating information from seed alignments, which are local pairwise alignments of the reads to $\mathcal{G}$.

Workflow

Our method progresses in several steps as depicted in Figure 1C. First, we perform local pairwise alignment of each read to $\mathcal{G}$ (Figure 1C Step 1). Next, we extract reads that are likely to have originated from SV sites by enumerating candidate SVs that can be inferred from their local alignments (Step 2). We group these reads according to the candidate variant site they point to. Each group along with $\mathcal{G}$ forms an instance of the joint split-alignment problem described in the previous section. We solve each instance by constructing a profile matrix of the reads (Step 3), and finding its maximum-scoring alignment to $\mathcal{G}$ (Step 4). Finally, we compute a confidence value for the joint split-alignments (Step 5). In the following, we describe these steps in more detail.

Step 1: Preliminary local pairwise alignment and filtering. We begin by identifying for each read, a set of its pairwise local alignments that are above a certain minimum score threshold. These alignments will provide primary information to detect candidate SV sites and to perform joint alignment in the following steps.

A

	$\mathcal{H}$-to-$\mathcal{G}$ (S_{genome})	Sequencer ($S_{\text{sequencer}}$)
substitution	σ	σ'
deletion open	δ_o	-
deletion extend	δ_e	δ'
insertion open	ϵ_o	-
insertion extend	ϵ_e	ϵ'
split	Δ	-

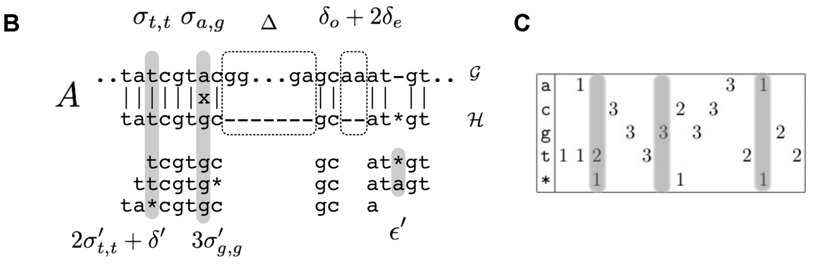
B
$\sigma_{t,t}$ $\sigma_{a,g}$ Δ $\delta_o + 2\delta_e$
A
..tatcgtacgg...gagcaaat-gt.. $\mathcal{G}$
||||||x| || || ||
tatcgtgc-------gc--at*gt $\mathcal{H}$
tcgtgc gc at*gt
ttcgtg* gc atagt
ta*cgtgc gc a
$2\sigma'_{t,t} + \delta'$ $3\sigma'_{g,g}$ ϵ'
C
a c g t *

Figure 2. Scoring a joint split-alignment. (**A**) Parameters of the scoring scheme. (**B**) Toy example with an alignment A of three reads identifying two deletions (dashed boxes). Computation of the score is indicated on some columns of the alignment—highlighted in gray, with contributions from S_{genome} and $S_{\text{sequencer}}$ at the top and at the bottom, respectively. (**C**) The matrix C for the example (a 15-bp count profile). The choice of linear gap penalties in $S_{\text{sequencer}}$ allows us to express the joint split-alignment as the alignment of C to $\mathcal{G}$, with columns of C treated independently. The gray columns match the ones in (**B**).

Due to the tendency of SVs to occur in repeat-rich regions, a large number of alignments will be reported for a single read (Figure 1A and SI-S1), most of which are uninformative about the true location of SVs. To keep only confident ones, we filter the local alignments based on their error probability (16) (Figure 1C step 1). The use of a probabilistic measure to filter alignments allows us to easily balance sensitivity and specificity. It also helps to reduce running time by avoiding unnecessary candidates. As opposed to the general practice among split-aligners of relying on ad-hoc thresholds such as the number of multi-mappings, our probability-based filter prioritizes propitious candidate split reads, an ingredient which is unique to our method.

Step 2: Candidate SV enumeration and read grouping. In the set of remaining high-quality local alignments of a read, we search for (pairwise) split-alignments, i.e. a set of two or more local alignments of the read to disjoint regions of $\mathcal{G}$, indicating an SV. These split-alignments serve a dual purpose of reducing the search space for the joint alignment algorithm by providing candidate SV sites and also aiding the grouping of related reads.

Different subsets of local alignments of the same read could indicate different SVs – for example, in Figure 1C, step 2, alignment subsets $\{a_{11}, a_{12}, a_{13}\}$ and $\{a_{11}, a_{14}\}$ of read r_1 suggest different deletion events. We enumerate and record all such events implied by the local alignments of each read. For the case of large deletions, this enumeration problem can be solved by defining a simple binary relation on the set of local alignments and traversing the order induced by this relation. We provide the problem definition and a description of the algorithm in SI-S4.

Finally, we group together reads with split-alignments using the following rule: if two reads have alignments overlapping in $\mathcal{G}$, they are placed in the same group.

Step 3: Profile construction. At this point, we have several groups of reads, each group containing reads likely to have been sequenced from the same SV site. We represent each read group $\mathcal{R}$ by a profile matrix C of size $5 \times \ell$, where the rows correspond to the characters in $\mathcal{D} = \{\mathtt{a}, \mathtt{c}, \mathtt{g}, \mathtt{t}, *\}$, the columns correspond to the columns in a multiple alignment of the reads, and C_{xi} contains the number of times x appears in column i of the multiple alignment (Figure 1C step 3). This definition requires finding a multiple sequence alignment of the reads in $\mathcal{R}$, which is known to be computationally expensive. Instead, we utilize information from local pairwise alignments of the reads to construct C, using the following principle: all positions aligning to the same position in $\mathcal{G}$ contribute to the same column of C (see SI-S5 for more details).

Step 4: Joint alignment. Next we align C to $\mathcal{G}$ using the scoring scheme described in Equation (1). We devise a dynamic programming algorithm to find a maximum-scoring pair A, H, as a simple extension to the traditional pairwise semi-global alignment. Recurrence formulas are given in SI-S3.

The time complexity of our exact dynamic programming algorithm is $O(|\mathcal{G}|^2 \cdot \ell \cdot |\mathcal{D}|)$, which is not practical for large datasets. To reduce time consumption, we use a speed-up heuristic (see SI-S5) that restricts the dynamic programming only to regions surrounding the candidate SV sites enumerated in Step 2. In essence, the algorithm chooses among the candidate SVs provided by pairwise alignments, the one that best matches the profile C which contains information from all the reads in the group.

Step 5: Joint alignment error probability. Let A be the split-alignment of C to $\mathcal{G}$ reported by our joint alignment algorithm, and let $A_1, \ldots, A_k$ be parts of A that align blocks of C to disjoint regions $\mathcal{G}_1, \ldots, \mathcal{G}_k$ of $\mathcal{G}$. Associated with each A_i is a set L_i of local alignments from the (pairwise) split-alignments enumerated in Step 2, that align segments of reads in $\mathcal{R}$ to the same region $\mathcal{G}_i$. For each alignment in L_i, its uncertainty is reported by LAST (16). Our computation of the error probability of A combines these (un)certainties associated with separate pairwise alignments into a single confidence value for the joint alignment. We describe this method in SI-S6.

Incorporating paired-end information

When available, we can utilize information from paired-end reads to strengthen the confidence value of local alignments in Step 1. For each local alignment of a read, we substitute the error probability of an alignment by that of an alignment of its mate read if the latter is lower and if the following conditions are fulfilled: (i) the mate is aligned on its full length and no other read block is aligned in-between (ii) the mate is within the distance $[\mu + z_{0.1}\,\sigma; \mu + z_{0.9}\,\sigma]$, where μ and σ are respectively the mean and standard deviation of the fragment size distribution, and z_α is the quantile of level α for the normal distribution $\mathcal{N}(0, 1)$ and (iii) there is no other candidate alignment within the same range of $[\mu + z_{0.1}\,\sigma;\ \mu + z_{0.9}\,\sigma]$.

RESULTS

We evaluated our Joint Read Alignment (JRA) framework of computing split alignments by comparing its performance to three other pairwise split-alignment tools: Segemehl (18), Splazers (19), and LAST (16). Additionally we investigated the effect of incorporating paired-end information in the JRA workflow by comparing its performance with Lumpy (20) and Delly (21), which are commonly used structural variant detection tools that can combine information from discordant alignments of mate pairs as well as split alignments. Further, to compare our strategy to local assembly and local realignment techniques, we included Platypus (10) in the evaluation. Platypus also combines haplotype and multi-sample information to make variant predictions, but we did not explore this aspect of Platypus as it is out of the scope of this work. Parameter settings for all tools are detailed in SI-S8.

Datasets

We used both simulated and real datasets for evaluation.

Our simulated dataset is based on the set of high-confidence variants in the diploid genome of a single individual (J. Craig Venter) (8). Since this dataset is derived from Sanger sequencing, it avoids the ambiguity issues faced by short erroneous reads, making it ideal for benchmarking of split alignments. We reconstructed Chromosomes 1 and 2 of the diploid Craig Venter genome from hg19 using a workflow similar to (22). Single-end and paired-end error-free sequencer reads of 100bp were simulated by random draws from the reconstruction. We chose the read length to be 100 bp as it is still the most commonly used setup for genome resequencing – take for instance, the last published SV annotations within the 1k genome project (23). To reproduce the proportion of low quality reads of real sequencing experiments, errors were then introduced according to the per-base phred quality values observed in a real dataset (dataset SRR067577, see SI-S7).

We emphasize that this benchmarking scheme is much more fitted to the challenges of SV identification than simply placing variants randomly, accounting only for their length distribution, as the latter would grossly underestimate the proportion of ambiguous regions (Supplementary Figures S1 and S6). In addition, this setup allows us to sequentially assess the effect of two major problems arising in split alignment: ambiguities due to genomic repetitions and artifacts from sequencer errors. The datasets are available for download at the JRA project homepage.

Additionally we used the short read dataset SRX652547 generated from the CHM1 cell line, which is derived from a haploid genome. It contains Illumina paired-end reads of length 101 bp sequenced at a 41-fold coverage. Interestingly, a list of variants was compiled for the same cell line by sequencing conjointly single-molecule long reads with Pacific Biosciences instrument (PacBio) at a 54-fold coverage (24). We used this list of variants to evaluate the quality of the predicted deletions. We would like to point out that although the long reads from PacBio instrument combined with the high coverage ensure a good confidence of the calls made, the variant calling methodology seems to have its own biases, and therefore there is no warranty that the annotation is complete (discussed later).

Evaluation of accuracy

For the simulated dataset, the set of positive examples consists of all deletions from the C. Venter genome located in Chromosomes 1 and 2 and spanning at least 20bp (2130 deletions). For the real dataset of CHM1, we compare the predicted deletions to the list of deletions derived from PacBio sequencing (11273 deletions). The length distribution of the deletions for the two genomes are compared in Figure 4A and B.

For most deletions, it is not possible to unambiguously pinpoint its start and end coordinates. We describe this issue in SI-S1.2, where we associate each deletion with breakpoint ambiguity intervals. We count a predicted deletion as true positive if its start and end coordinates are within the breakpoint ambiguity intervals of a true deletion. Breakpoint ambiguity also means that multiple predictions may in fact be pointing to the same deletion event, and we make sure to avoid such multiple-counting issues when reporting the number of true positives.

Sensitivity is defined as the proportion of deletions recovered, and the positive predictive value (PPV) as the proportion of correct deletions in the set of predicted ones.

Performance of deletion calling

We first assessed the performance of the four split-alignment tools at various coverage values for 100bp reads under two simulation scenarios: no sequencing errors and with an error profile derived from the real dataset SRR067577. For Splazers and Segemehl, we used the scripts provided by the authors to make deletion calls (SI-S8). For LAST, we use as threshold the maximum error probability over the local alignments constituting the split-alignment, and identify as redundant, deletion calls with matching start and end coordinates.

Overall, JRA shows the best performance over the whole range of simulations and parameter values (Figure 3). Splazers, due to its exhaustive search strategy, manages to detect, in some cases, slightly more deletions but at the cost of many more false calls. For instance, with coverage 10, and looking at all predicted deletions, a very slight increase in sensitivity with Splazers (4% or 5%), comes at the cost of a 4- to 5-fold increase in the amount of false detections (PPV of 67% with JRA compared to 29% with Splazers for the sequences with errors). Given the clinical implications of SV detection, it is important that those are made with high specificity.

Relying on the number of supporting reads to call a deletion can improve the specificity of calls but discards many *bona-fide* deletions. Indeed coverage alone is not able to discriminate between a deletion called by only a few reads which are split-aligned perfectly on both flanks and another

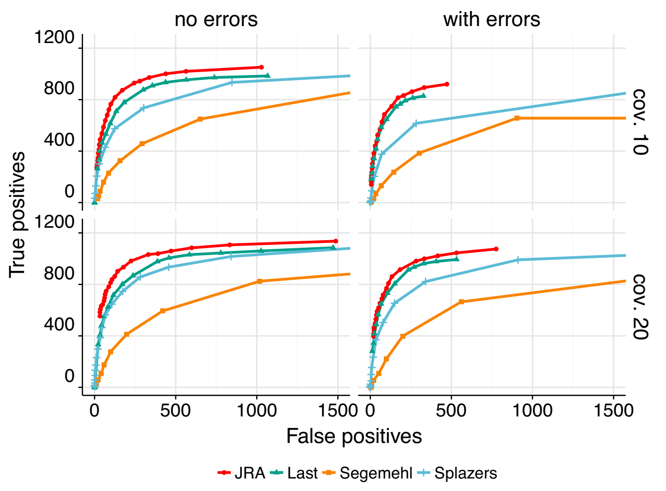

Figure 3. Number of true positive and false positive deletions for JRA (red circles), LAST (green triangles), Splazers (blue crosses) and Segemehl (orange squares) for different values of expected coverage (top: 10, bottom: 20) and different simulation scenarios (left: error-free reads, right: reads with errors).

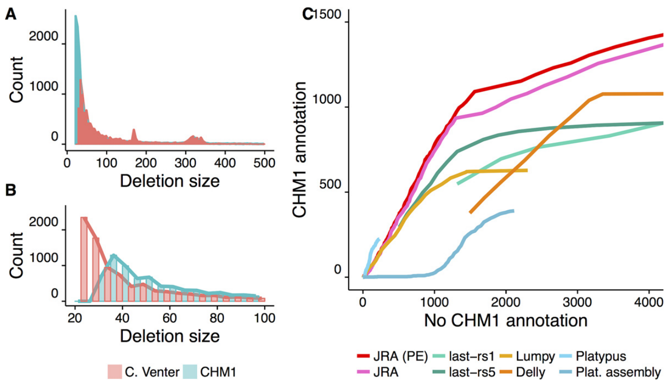

Figure 4. Left: Length distribution of the deletions annotated in the CHM1 (red) and the Craig Venter genome (blue) at two different scales (**A**: 20–500 bp, **B**: 20–100 bp). The two distributions strongly agree, except in the range from 20 to 36 bp, which are not reported in the CHM1 annotation, but consitute a large set in Venter. (C) Performance of the different tools JRA (paired-end mode: red, default mode: pink), LAST with a minimum read support of 1 (dark green) or 5 (light green), Lumpy (ocre), Delly (orange) and Platypus in default mode (light blue) or with the assembly option activated (dark blue).

one called by many reads but with sub-optimal alignments. This is well illustrated when comparing JRA and Splazers results. At the same level of false positive (FP) predictions (coverage 10, 250 FP, JRA error probability of 0.08), we observed that more than a third of the true deletions called by JRA are only supported by one read (60% by less than two reads) in the Splazers predictions.

Additionally, the error probability reported by JRA integrates the uncertainties of alignments of single reads at the SV site, enabling the user with a more accurate choice when deciding a threshold. Of the tools we compared, only LAST reports a similar probability, but it does not incorporate group information. We compared the sensitivity and the PPV between JRA and LAST for varying probabilities, and JRA had consistently better sensitivity and slightly lower PPV at the same probability threshold (Supplementary Figure S12)

We also evaluated the accuracy of the sequence of the deletion locus reconstructed by JRA from the dataset with sequencing errors. For each true positive prediction, we counted the number of mismatches when comparing the predicted sequence to the corresponding true sequence in the Venter genome. Averaged over a range of error-probability thresholds, we observed an average per-base mismatch rate of just 0.6 %. This number is remarkable given that the sequencing dataset has a per-base average error rate of around 10% (Supplementary Figure S11).

Performance on paired-end data

Paired-end libraries are often generated with the goal of disentangling ambiguities due to repetitive regions and to increase sensitivity of SV calls. Thus, multiple strategies for SV detection rely on the detection of *discordant* mate pairs, i.e. pairs where the alignment on the reference disagree with the expected fragment size properties. These methods have a sensitivity directly related to the variability of the fragment lengths. We chose Lumpy (20) and Delly (21), as they both combine discordant paired-end alignments with split alignments, and compared it with the results of JRA, and of a modified JRA where we integrate evidence from mate pair alignment. We also compared it to Platypus, which additionally integrates information from local realignment and local assembly. The results for different sizes of deletions are given in Table 1.

Deletions with size between 20 and 250 bp account for a large majority (66.67%) of all Venter deletions. In this range, JRA clearly outperforms Delly and Lumpy, detecting at least 2.5 times more true deletions. It is well known that discordant alignments are not very informative in this range, but it is surprising that they can be so detrimental, as witnessed by the drastic loss in sensitivity. This could be attributed to the fact that Delly, for instance, uses its paired-end module and split alignment module in a cascading fashion, such that split alignment is performed only on the regions that is detected by the paired-end module to likely contain SVs.

Platypus, in its default mode, detects short deletions (<50 bp) with a very good accuracy, on par with JRA, but with half of its sensitivity. However, in this mode, it does not return any longer deletion. Adding the use of local assembly allows us to get a few longer predictions but with a sensitivity that is around half of the other methods. Additionally, calls between 50 and 250 bp are very poor, maybe owing to the specifics of the local assembly construction.

For deletions larger than 250 bp, Delly and Lumpy show results comparable with paired-end mode of JRA. This can be expected since the signal coming from discordant pairs needs to account for fragment length variability and therefore becomes effective only for longer deletions. However, it is surprising that almost no additional deletions are detected by Delly or Lumpy with respect to JRA.

This result also suggests that the gain in using paired-end information declines for larger deletions. It turns out that in the Venter deletion set, the proportion of deletions that

Table 1. No. of true positives (TP), and positive predictive value (PPV) in percentage for different categories of deletion lengths, when combining paired-end alignment information in JRA, and comparison with other tools that use multiple sources of information for SV calling. JRA results are for maximum allowed error probability value set to 0.01. Total number of deletions is 2130

	Coverage 10×				Coverage 20×			
	No errors		w. errors		No errors		w. errors	
	TP	PPV	TP	PPV	TP	PPV	TP	PPV
Length range 20–50 bp (1419 deletions)								
JRA (PE)	733	87	654	88	813	87	744	87
JRA	635	88	577	89	739	89	680	89
Platypus	421	91	330	91	577	90	505	91
Platypus (assembly)	438	85	352	83	583	85	520	85
Delly	120	86	100	89	295	86	281	86
Lumpy	13	92	9	90	37	82	27	84
Length range 51–250 bp (438 deletions)								
JRA (PE)	90	73	69	72	107	67	89	68
JRA	67	74	55	73	90	70	76	71
Platypus	-	-	-	-	-	-	-	-
Platypus (assembly)	16	18	12	15	19	18	13	13
Delly	36	50	26	55	62	40	52	44
Lumpy	34	50	25	50	47	47	43	50
Length range 251–500 bp (174 deletions)								
JRA (PE)	101	90	99	95	108	90	105	90
JRA	99	91	98	96	102	88	102	91
Platypus	-	-	-	-	-	-	-	-
Platypus (assembly)	38	58	32	66	46	63	42	65
Delly	89	68	78	63	114	62	106	63
Lumpy	101	84	96	83	103	79	101	80
Length range > 500 (99 deletions)								
JRA (PE)	36	42	36	48	38	31	38	39
JRA	34	44	34	50	38	33	38	40
Platypus	-	-	-	-	-	-	-	-
Platypus (assembly)	13	41	11	42	15	42	14	35
Delly	35	26	34	30	38	20	37	24
Lumpy	35	63	35	66	35	44	35	53

are located in regions that would result in ambiguous alignments of reads, is much higher for longer deletions than shorter ones. This is the case even if we assume hypothetical reads of length 275 bp, which we can think of as modeling the best case scenario for paired reads from similarly long fragments (SI-S1.1.2 and Figure S3).

In summary, over the whole range of deletions sizes, our joint split-alignment method performs better than conventional align-and-call techniques and local realignment and local assembly techniques.

Evaluation on real sequencing data

For evaluation on the CHM1 dataset, we compared the predictions of JRA with Last and the tools used for variant detection: Delly, Lumpy and Platypus, which were provided with the alignments computed by BWA. We omitted the other aligners Splazers and Segemehl because the former failed on larger datasets and the latter required too much running time (see Computational costs).

First, comparing the length distribution between the deletions annotated for CHM1 (24) and for the C. Venter genome (8) (Figure 4A and B), we can observe that globally, both annotations strongly agree. However, strikingly the shorter deletions in the range 20–35 bp are not reported at all in CHM1. We therefore limited ourselves to predicted deletions which are between 36 and 20 000 bp for all tools. The results are shown in Figure 4C.

The plots show that JRA is in general more accurate than all of the other tools reaching up to 30% more sensitivity in detecting *bona fide* deletions. Interestingly, Platypus manages in default mode to give more specific results on the first few hundreds deletions, but those are limited to medium size deletions as we previously noticed on simulated data. Also in line with simulated data, most of the gain in accuracy for JRA comes from deletion that are shorter than 250 bp, while the results become more comparable with Lumpy (and to a lower extent Delly) for longer deletions (Supplementary Table S1).

However, simulated and real data disagree regarding the proportion of calls identified as correct. For all tools, the proportion of calls with a CHM1 annotation is drastically lower than the positive predictive value on simulated data (Figure 3 and Table 1). This observation, along with the discrepancy in the deletion length distributions suggest that the CHM1 annotation might be missing many deletions.

To further validate the behavior of our framework on other real genomic data, we used reads sampled from sequencing dataset (SRX1567556) of the *Ler-0* strain of the plant *A. thaliana*, and performed deletion calling with respect to the *A. thaliana* reference genome. We monitored the number of deletions called as a function of the confidence of the call, and observed similar behavior as in the simulated case (Supplementary Figure S14).

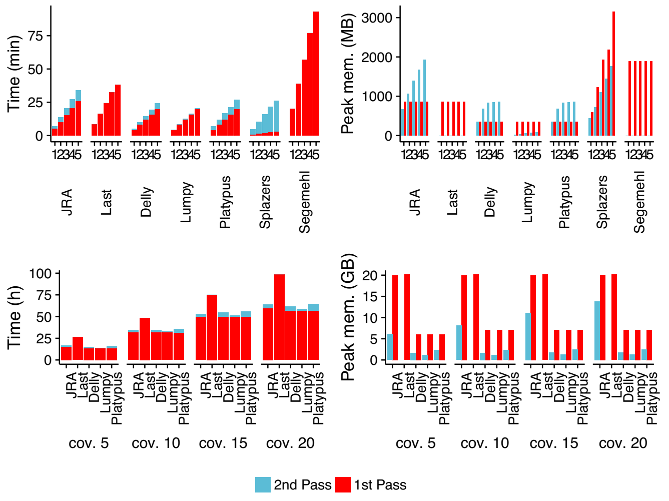

Figure 5. Growth of running time and memory usage of different methods for down-sampled *A. thaliana* dataset containing 1 through 5 million reads (top) and human whole genome dataset containing reads corresponding to coverage 5 through 20 (bottom). For JRA, first pass refers to LAST gapless alignments, 2nd pass refers to our joint alignment phase. For LAST, first pass refers to gapped alignment. For Delly, Lumpy and Platypus, 1st pass refers to BWA alignment, and 2nd pass refers to variant calling including tasks such as sorting of BWA alignments, discordant alignment extraction. For Splazers, the first pass alignment is performed by Razers, which gives reads that it cannot fully align to Splazers for the second alignment phase. We did not run Splazers and Segemehl on the whole genome dataset because of insufficient memory for the former and extremely long running time for the latter.

Computational costs

We examined running time and memory usage behavior of the four tools using reads from the SRX1567556 *A. thaliana* dataset and the ERR037900 human whole-genome dataset. From the former, we constructed small datasets by sampling 1 through 5 million reads, in order to understand the growth rate of running time and memory usage. From the latter, we sampled reads to construct datasets of coverages 5, 10, 15 and 20, in order to investigate the performance of the tools on larger-scale input data. All experiments were conducted on a machine with Intel(R) Core(TM) i7-4710MQ CPU and 32GB main memory running Ubuntu 14.04, and time measurements are for single-threaded operations.

Results are shown in Figure 5. Overall running time for JRA consists of time taken by LAST for performing preliminary gapless alignments and time taken for joint alignment. We can observe that it grows linearly with input size and that it compares well against other tools. It also worth noting that the joint alignment phase imposes only a small overhead.

Memory usage of the joint alignment phase depends on the number of reads involved in candidate SV sites, and we can see that it grows linearly with the number of reads. For large-size genomes, we observe that peak memory usage occurs during the preliminary alignment phase. Overall peak memory usage compares well against other tools.

Together with the results from performance evaluation, reasonable computational costs further justify the use of JRA as tool for detecting large-sized deletions, even from large datasets.

DISCUSSION

We have provided a general framework to incorporate information from a set of error-prone short reads covering an SV site, in order to compute an alignment of this SV region to the reference, while simultaneously reconstructing the SV sequence. We have demonstrated a proof of concept of our joint alignment strategy by applying it to the problem of identifying large deletions and showing its advantage over independent pairwise alignments and over calling tools that rely on such alignments. Given that almost all sequence datasets today are of this nature, it is surprising that current methods rely on pairwise alignments, and that no grouped alignment approach has been proposed before. We hope to have initiated a step in this direction.

It is worth noting that in line with our preliminary observation on SV ambiguity (SI-S1), none of the aligners managed to recover more than half of the deletions annotated in the Venter genome, even when considering flawless sequencing. This points out the challenges posed by genomic repetitions and the importance of accounting for them during alignment. We use a sound probabilistic framework to incorporate ambiguity information at two key steps: for filtering initial SV candidates and when assessing the confidence of a joint-split alignment. We believe these unique features of JRA contribute to its overall better performance over other existing tools, and makes it a practical choice for split-alignment based SV prediction.

JRA is an alignment tool, and as such does not make any assumption on the number of reads needed to support a split alignment. JRA can thus detect heterozygous deletions as well. And indeed, a sound percentage of the calls are true heterozygous deletions (5% rate predicted in JRA against a 9% rate expected in Venter). This suggests that JRA could also be successfully applied to the annotation of somatic or tumor deletions, provided there is enough coverage.

Paired-end data are often recommended for SV detection as leading to higher sensitivity. However, most of the tools exploiting this information are limited to using discordant mate pairs. Even when allowing for modest variability in fragment size, JRA outperforms the most common paired end/split alignment framework for deletions of medium lengths (20–250 bp), while having the same accuracy for longer lengths. We also observed that the benefit from paired information declines for longer deletions, as the proportion of deletions which are located in regions where even paired-end reads cannot resolve alignment ambiguity is much higher for longer deletions.

The generality of our framework provides many potential avenues for enlarging the scope of our current work. An immediate extension would be to apply it to more complex SVs involving multiple chromosomes or containing more complicated events such as inversions and duplications. We started out by targeting deletions because this restriction implies that the joint alignments we seek are co-linear (i.e. left-to-right on both query and reference), which simplifies several steps. It allows enumerating candidate variant sites (Workflow Step 2) by defining a simple binary relation on the set of local alignments and traversing the order induced by this relation (SI-S4). It also simplifies the dynamic programming algorithm, which takes a form similar to the familiar classical semi-global alignment dynamic programming (SI-S3). These are key steps in the current JRA, and to allow complicated evolutionary transformations such as inversions and duplications would require generalizing these steps (See SI-S4 and SI-S3 for further discussions). These are interesting challenges. Another area of improvement is the profile construction step, which currently does not tolerate a high indel error rate. By borrowing ideas from the rich literature in multiple sequence alignment, the scope of application can be expanded to reads with high indel error rates.

Another application of our idea of joint alignment is to long read datasets obtained from PacBio or Oxford Nanopore technologies. Long reads allow capturing large variants more accurately, and it also has the added benefit that a single read may span multiple such variants. However, they are known to have high error rates, and therefore sharing information among reads may lead to more accurate alignments and consequently accurate variant calls. However, the data structures and profile construction methods currently employed in JRA are suited only for variant site loci of a few hundred base pairs, and it is a stimulating challenge to make them computationally viable for longer reads that can span thousands of base pairs.

AVAILABILITY

A software implementation is available as an open-source Python program called JRA at https://bitbucket.org/jointreadalignment/jra-src.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR online.

ACKNOWLEDGEMENTS

We thank Dr Elodie Laine for her careful reading of the manuscript. We also thank the three reviewers whose comments have substantially improved the manuscript.

FUNDING

Japanese Ministry of Education, Sport, Science and Technology [MEXT KAKENHI 16H06279]; Japan Society for the Promotion of Sciences [PE11014 to H.R.]. Funding for open access charge: University funding.
Conflict of interest statement. None declared.

REFERENCES

1. Kuzniar,A., van Ham,R.C.H.J., Pongor,S. and Leunissen,J.A.M. (2008) The quest for orthologs: finding the corresponding gene across genomes. *Trends Genet.*, **24**, 539–551.
2. 1000 Genomes Project Consortium, Abecasis,G.R., Altshuler,D., Auton,A., Brooks,L.D., Durbin,R.M., Gibbs,R.A., Hurles,M.E. and McVean,G.A. (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–1073.
3. Stephens,P.J., McBride,D.J., Lin,M.-L., Varela,I., Pleasance,E.D., Simpson,J.T., Stebbings,L.A., Leroy,C., Edkins,S., Mudie,L.J. *et al.* (2009) Complex landscapes of somatic rearrangement in human breast cancer genomes. *Nature*, **462**, 1005–1010.
4. Alkan,C., Coe,B.P. and Eichler,E.E. (2011) Genome structural variation discovery and genotyping. *Nat. Rev. Genet.*, **12**, 363–376.
5. Medvedev,P., Stanciu,M. and Brudno,M. (2009) Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods*, **6**(Suppl), S13–S20.
6. Treangen,T.J. and Salzberg,S.L. (2012) Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nat. Rev. Genet.*, **13**, 36–46.
7. Wittler,R., Marschall,T., Schönhuth,A. and Mäkinen,V. (2015) Repeat- and error-aware comparison of deletions. *Bioinformatics*, **31**, 2947–2954.
8. Levy,S., Sutton,G., Ng,P.C., Feuk,L., Halpern,A.L., Walenz,B.P., Axelrod,N., Huang,J., Kirkness,E.F., Denisov,G. *et al.* (2007) The diploid genome sequence of an individual human. *PLoS Biol.*, **5**, e254.
9. Homer,N. and Nelson,S.F. (2010) Improved variant discovery through local re-alignment of short-read next-generation sequencing data using SRMA. *Genome Biol.*, **11**, R99.
10. Rimmer,A., Phan,H., Mathieson,I., Iqbal,Z., Twigg,S.R., Wilkie,A.O., McVean,G. and Lunter,G. (2014) Integrating mapping-, assembly- and haplotype-based approaches for calling variants in clinical sequencing applications. *Nat. Genet.*, **46**, 912–918.
11. Hormozdiari,F., Alkan,C., Eichler,E.E. and Sahinalp,S.C. (2009) Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Res.*, **19**, 1270–1278.
12. Marschall,T., Costa,I.G., Canzar,S., Bauer,M., Klau,G.W., Schliep,A. and Schönhuth,A. (2012) CLEVER: clique-enumerating variant finder. *Bioinformatics*, **28**, 2875–2882.
13. Canzar,S., Elbassioni,K., Jones,M. and Mestre,J. (2016) Resolving conflicting predictions from multimapping reads. *J. Comput. Biol.*, **23**, 203–217.
14. Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press.
15. Nielsen,R., Paul,J.S., Albrechtsen,A. and Song,Y.S. (2011) Genotype and SNP calling from next-generation sequencing data. *Nat. Rev. Genet.*, **12**, 443–451.
16. Frith,M.C. and Kawaguchi,R. (2015) Split-alignment of genomes finds orthologies more accurately. *Genome Biol.*, **16**, 106.
17. 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.
18. Hoffmann,S., Otto,C., Kurtz,S., Sharma,C., Khaitovich,P., Vogel,J., Stadler,P. and Hackermueller,J. (2009) Fast mapping of short sequences with mismatches, insertions and deletions using index structures. *PLoS Comput. Biol.*, **5**, e100502.
19. Emde,A.-K., Schulz,M.H., Weese,D., Sun,R., Vingron,M., Kalscheuer,V.M., Haas,S.A. and Reinert,K. (2012) Detecting genomic indel variants with exact breakpoints in single- and paired-end sequencing data using SplazerS. *Bioinformatics*, **28**, 619–627.
20. Layer,R.M., Chiang,C., Quinlan,A.R. and Hall,I.M. (2014) LUMPY: a probabilistic framework for structural variant discovery. *Genome Biol.*, **15**, R84.
21. Rausch,T., Zichner,T., Schlattl,A., Stütz,A.M., Benes,V. and Korbel,J.O. (2012) DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, **28**, i333.
22. Talwalkar,A., Liptrap,J., Newcomb,J., Hartl,C., Terhorst,J., Curtis,K., Bresler,M., Song,Y.S., Jordan,M.I. and Patterson,D. (2014) SMaSH: a benchmarking toolkit for human genome variant calling. *Bioinformatics*, **30**, 2787–2795.
23. Sudmant,P.H., Rausch,T., Gardner,E.J., Handsaker,R.E., Abyzov,A., Huddleston,J., Zhang,Y., Ye,K., Jun,G., Fritz,M. H.-Y. *et al.* (2015) An integrated map of structural variation in 2,504 human genomes. *Nature*, **526**, 75–81.
24. Chaisson,M.J.P., Huddleston,J., Dennis,M.Y., Sudmant,P.H., Malig,M., Hormozdiari,F., Antonacci,F., Surti,U., Sandstrom,R., Boitano,M. *et al.* (2015) Resolving the complexity of the human genome using single-molecule sequencing. *Nature*, **517**, 608–611.