Detection of complex structural variation from paired-end sequencing data

Joseph G. Arthur* Xi Chen* Bo Zhou† Alexander E. Urban†
Wing Hung Wong*‡

Abstract

Detecting structural variants (SVs) from sequencing data is a key problem in genome analysis, but the full diversity of SVs is not captured by most methods. We introduce the Automated Reconstruction of Complex Structural Variants (ARC-SV) method, which detects a broad class of structural variants from paired-end whole genome sequencing (WGS) data. Analysis of samples from NA12878 and HuRef suggests that complex SVs are often misclassified by traditional methods. We validated our results both experimentally and by comparison to whole genome assembly and PacBio data; ARC-SV compares favorably to existing algorithms in general and gives state-of-the-art results on complex SV detection. By expanding the range of detectable SVs compared to commonly-used algorithms, ARC-SV allows additional information to be extracted from existing WGS data.

Since the observation of microscopically visible aneuploidies and gene duplications, large-scale genomic alterations have been of interest in studies of disease, comparative genomics, and population genetics [1–4].

Microarrays and sequencing technologies have since enabled the detection of submicroscopic variants, and structural variants are defined as mutations affecting at least 50 bp of sequence. While sequencing technologies may eventually yield a complete description of genomic variation, there are currently significant experimental and computational challenges.

The vast majority of sequencing-based SV callers detect deletions, duplications, inversions, insertions, and/or translocations [5]. Aside from insertions, any of these "simple" SVs is defined by a single novel adjacency in the sample genome, making detection theoretically straightforward given accurate read mapping and high coverage. Misaligned reads make SV detection difficult in practice, and investigators typically apply multiple methods together with heuristic filters to achieve high accuracy [2, 4, 6, 7]. Structural variants with complexity beyond the scope of most detection algorithms have also been observed in a variety of phenotypic contexts [2, 8–10]. As concerns this work, our definition of a complex SV (cxSV) is any rearrangement not reducible to non-overlapping deletions, tandem duplications, novel

*Department of Statistics, Stanford University
†Department of Psychiatry and Behavioral Sciences and Department of Genetics, Stanford University School of Medicine
‡Department of Biomedical Data Science, Stanford University

insertions, and inversions; we focus here on the localized cxSVs commonly observed in the germline.

Identification of cxSVs requires both detection and phasing of breakpoints, but methods targeting simple SVs generally equate individual breakpoints with SV detections (Figure 1D), which can significantly affect interpretation. While several previous methods are capable of cxSV detection with varying degrees of generality [9, 11–13], cxSVs have primarily been identified through manual inspection of paired-end mapping patterns, local assemblies, and long sequencing reads [2, 8, 10]. Thus, we have developed ARC-SV, a tool for detecting a broad class of simple and complex SVs from standard WGS data.

There are numerous possible cxSV structures, each producing a signature of read depth, mapped read orientations, insert sizes, and split alignments. The observed signature will depend on both sequencing library characteristics (especially insert size) as well as the SV. Some previous authors [9, 12, 13] take the approach of classifying cxSV structures using predefined mapping signatures. To achieve more general detection of cxSV structures, ARC-SV proposes candidate diploid rearrangements and scores each one using a probabilistic model for the data that incorporates all available reads, as illustrated in Figure 1 and in our Methods. Notably, ARC-SV simultaneously detects and genotypes SVs using all relevant reads, instead of relying only on discordant reads. Previous work most similar to ours includes SVelter [11], which detects complex SVs using another "propose and score" approach, and SWAN [14], which uses a detailed probabilistic model for insertion and deletion detection.

Comprehensive gold sets of complex SVs are not available, and it is not *a priori* clear how to compare general rearrangements. To comprehensively evaluate SV calling accuracy, we aligned each variant's rearranged sequence against a published assembly or long read sequencing. Validation of an SV requires that the best such alignment spans the entire rearrangement aligns each rearranged segment with high sequence identity, and aligns across simple deletion breakpoints. Additionally, we require that the same validation criterion is not met by alignment to the original reference (Methods). This validation procedure is similar to a previous method [11], though we additionally incorporate the SV structure into our scoring function.

We applied ARC-SV and 5 other SV callers to WGS data derived from two healthy human samples — Venter and NA12878 — chosen so we could validate predictions against a diploid assembly [15] and Pacbio data [16], respectively (Methods). In what follows, we have excluded variants with 85% overlap to tandem repeat regions, removed near-duplicate SV calls within each call set, and removed NA12878 variants with very low PacBio read coverage (Methods).

We first compare validation rates between our alignment-based method and the usual 50% reciprocal overlap criterion, using sets of known NA12878 and Venter deletions [6, 7]. In terms of the relative sensitivity and specificity of different callers, our alignment-based validation gives nearly identical results to those based on the gold sets (Supp. Figures 4 and 5). Across all size ranges below 10 kb, we observed high concordance between the sets of true positives from each validation method; concordance among false positives was somewhat lower but may reflect incompleteness in the gold sets (Supp. Figure 6).

In order to evaluate the performance of SV callers, we compared the accuracy of the initial call sets (Supp. Table 1) as well as calls after filtering for quality (Figure 2A; stratified by SV size in Supp. Figures 7, 8, 9). Each call set was filtered by a random forest classifier

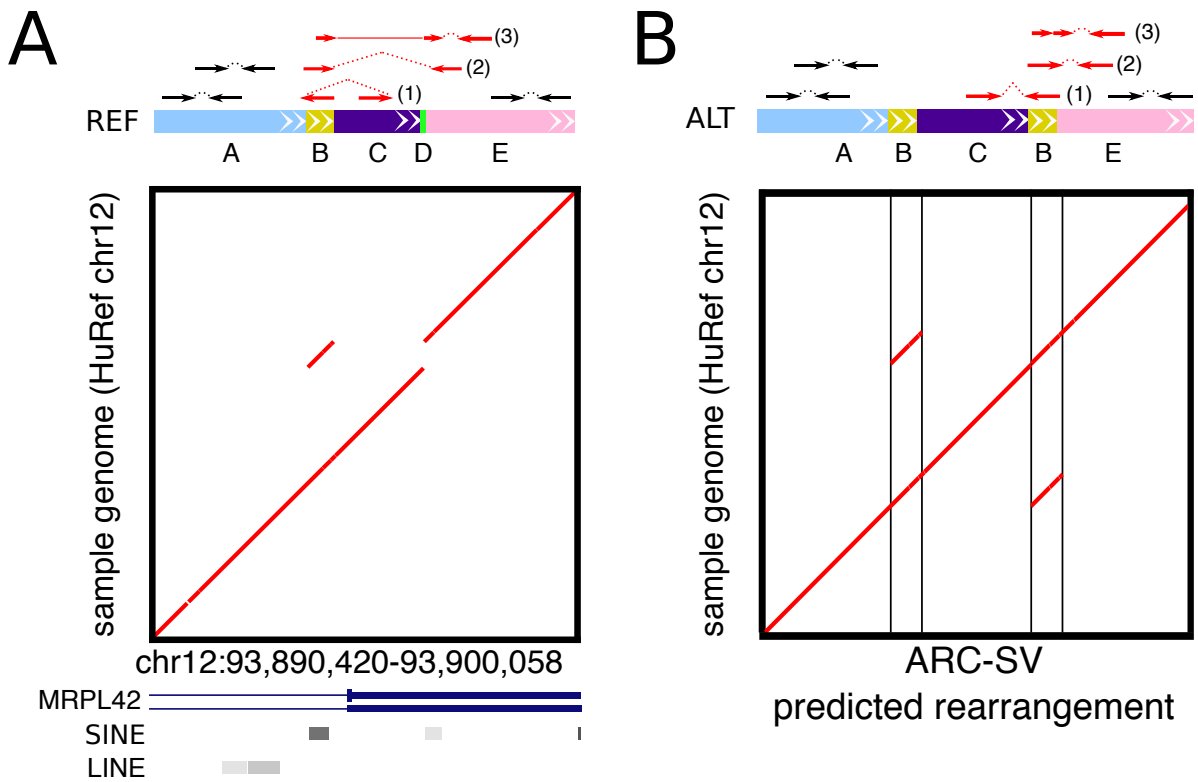
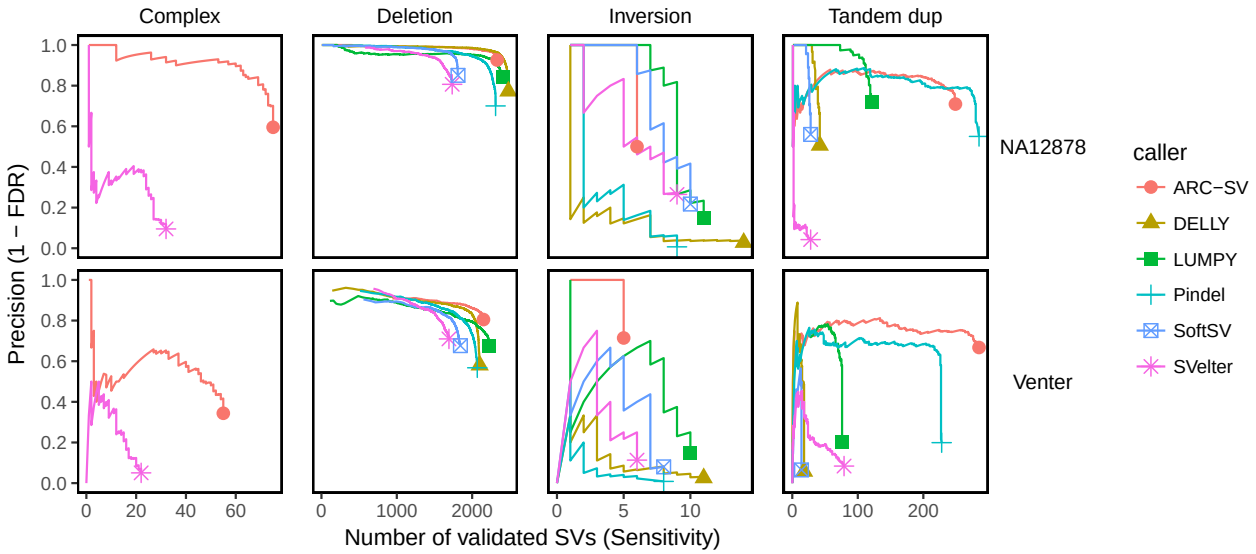


Figure 1: A: Sequence comparison between the reference genome and a corresponding region in the sample (ground truth sequence from HuRef assembly). Discordant alignments are consistent with both (1) a tandem duplication of BC; and (2, 3) a deletion of C or CD. B: Comparison using ARC-SV's predicted rearrangement instead of the reference. The reads producing discordant alignments are more likely given this alternative structure.

trained on validation results from the opposite sample, using the relevant features from each algorithm's output (Supp. Table 5; Methods).

Before random forest filtering, ARC-SV achieves the highest precision in both samples and across all variant types except Tandem Duplications in NA12878, where LUMPY is more accurate (FDR of 28% vs 29% in ARC-SV) but much less sensitive. The sensitivity-precision curves suggest that, after filtering, ARC-SV and DELLY are most competitive for deletions, ARC-SV and Pindel for Tandem Duplications. There are too few validated simple inversions for a thorough comparison, though we note that ARC-SV's potential to resolve complex inversions probably contributes to its relatively high precision in simple inversion detection. ARC-SV sacrifices sensitivity in some variant classes. Notably, LUMPY detects 25% more deletions > 1 kb across both samples (1227; ARC-SV detects 984), though with a drop in precision compared to ARC-SV (63% vs 91%).

Complex SVs make up 2.5% (130/5229) of all validated ARC-SV predictions: there are 55 validated complex events in Venter, 75 in NA12878, and 16 calls in NA12878 excluded for low Pacbio coverage. At 129/130 (99%) of these complex sites, one or more simple SVs was called by other methods on the same sample (with at least 50% overlap to the complex call). Many of the overlapping simple SVs were validated by our alignment method, but the



A

B

Complex SV type	ARC-SV	SVelter
Interspersed DUP [+ DEL]	35 (49%)	15 (12%)
Inverted DUP [+ DEL]	36 (59%)	23 (19%)
INV + DEL	32 (71%)	13 (23%)
Other	27 (25%)	3 (1%)

Figure 2: (A) Precision and sensitivity results based on raw SV calls (points) and cross-sample random forest-based filtering (curves). (B) Types of cxSVs discovered across NA12878 and Venter (combined results, validation rates in parentheses).

results from ARC-SV suggest that additional complexity was missed.

Most validated complex events fall into three categories: interspersed (non-tandem) duplications, inverted duplications (mostly non-tandem), and inversions flanked on one or both sides by deletions (Figure 2B, Supp. Fig. 11, Supp. Tables 3 and 4). A number of complex SV structures were also observed (Supp. Fig. 12, illustrating one advantage of a highly general cxSV caller.

Complex SVs containing duplicated segments make up a significant portion of all validated ARC-SV duplications (249 tandem vs 55 complex duplications in NA12878; 285 vs 38 in Venter). Out of 71 interspersed duplications correctly called by ARC-SV across our two samples (without any merging or deduplication of events), 51% involve an inversion of the duplicated segment, and 70% have an accompanying deletion at the insertion site. The median sizes of these variants were 187 bp duplicated, an insertion site 1155 bp away, and, among cases with deletions, 13.5 bp of deleted sequence (Supp. Figure 13).

Consistent with recent surveys of complex variation [2, 9], we find few validated simple inversions compared to complex events with inverted segments (ARC-SV: 6 simple vs 49 complex in NA12878; 5 vs 36 in Venter). The most common structures seen among complex inversions are inverted duplications and inversions with deletions at one or both flanks (Figure 2B, Supp. Table 3).

We used targeted sequencing to experimentally interrogate several Venter SV calls. The validation rate was 80% (4/5) for simple SVs and 80% (4/5) for complex SVs; genotype calls

were correct for 7 of the 8 validated SVs (Methods; Supp. Table 6). We also investigated SVs called from whole genome sequencing of HepG2 cells, whose abnormal karyotype does not match ARC-SV's modeling assumption of a diploid genome (Supp. Tables 7, 8). Sequencing across predicted breakpoints yielded validation rates of 93% (40/43) for deletions and 80% (16/20) for tandem duplications; testing 5 non-tandem duplications confirmed both breakpoints for 3 variants and a single breakpoint of another variant.

ARC-SV and SVelter each required approximately 110 CPU hours per sample, compared to LUMPY's 9 hours and DELLY's 18 hours. Future versions of the ARC-SV software will likely incorporate computational speedups, but for now, our detection of cxSV structures comes at an increased computational cost.

Limitations of our current methodology include a lack of novel insertion detection or non-reference sequence determination in general. Also, unambiguously resolving some SVs (e.g., large homozygous tandem duplications) requires phasing information not available in paired-end reads from a single sample.

The promises of genome sequence analysis may only be fulfilled by ubiquitous long read sequencing. However, there is a wealth of data from paired-end sequencing experiments, and new methodologies such as ARC-SV may bring new insights.

Methods

Breakpoint detection

Soft-clipped and split alignments. Soft-clipped and split alignments produced by BWA–MEM are both used for breakpoint detection. Split alignments are assigned a type (deletion, duplication, left/right side of inversion) and a breakpoint interval based on the orientation and mapped positions of the primary and secondary alignment. Soft-clipped reads without split alignments are also used. We require the soft-clipped portion is at least 5 bases long and has median base quality 15 or higher. For increased sensitivity on soft-clipped reads, we require a MAPQ score of 10 rather than the threshold of 20 used elsewhere in ARC-SV.

Soft-clipped reads within 5 bp and with the same clip orientations are merged; the breakpoint position whose supporting reads have the highest total MAPQ is used as a candidate breakpoint. Also, microhomology can lead to two soft-clip clusters at a single insertion or inversion breakpoint. We thus merge each pair of soft-clip clusters with overlapping aligned bases and opposite clip orientations whenever the two breakpoints are within 25 bp; the region between the breakpoints is then used as a breakpoint interval.

Discordant read pair extraction. Discordant read pairs (DRPs) are read pairs whose abnormal alignment suggests the presence of structural variation. Pairs with abnormal insert sizes suggest deletions and insertions, and pairs with abnormal mapping orientation suggest tandem duplications and inversions. We treat DRPs from the left and right sides of inversions separately.

Deletion and insertion read pairs are defined according to insert size cutoffs. Conventionally, discordant insert sizes are those outside of $(\mu - n\sigma, \mu + n\sigma)$, where μ and σ are the mean and standard deviation of the estimated insert size distribution f, and the multiplier n is commonly set to 3 [17, 18]. To allow for a variety of different insert size distributions, we instead use cutoffs based on a likelihood ratio. Because discordant reads are extracted during the first pass of the data, the insert size distribution is estimated beforehand using a random subsample of reads (1 million by default). To avoid including read pairs that clearly span large deletions, we truncate the insert size distribution at 3 times the median insert size. We define deletion pairs as those with insert sizes exceeding z^*, where z^* is the smallest value such that, for all $z > z^*$,

$$\frac{\max_{S \geq 0} f(z - S)}{f(z)} > C.$$

The ratio above involves the probability of observing an insert size z under the reference genome versus a genome with a homozygous deletion of maximal likelihood. We set $C = \exp(3^2/2)$ so that, if f is a normal distribution, the cutoff z^* is the commonly-used value $\mu + 3\sigma$. Insertion read pairs are defined by insert sizes smaller than z_*, where z_* is defined analogously to z^* but with maximization over $\{S \leq 0\}$.

Discordant read pair clustering. For discordant pair i, we denote the mapped insert size as z_i and the 3′-most mapped read positions as a_i (leftmost read in the reference) and b_i (rightmost). After extracting DRPs, we construct a graph with nodes corresponding

to DRPs. Two DRPs i and j are connected if they are of the same type, if their a_i and b_i positions are no farther apart than $\mu + 3\sigma$, and if $\max\{a_i, a_j\} < \min\{b_i, b_j\}$. (For insertions, we adjust this maximum length $\mu + 3\sigma$ downward by a conservative estimate of the insertion size.) The third condition guarantees some possible SV can be defined spanned by the two read pairs. (For insertions, we require $\max\{a_i, a_j\} - h < \min\{b_i, b_j\}$ to allow for up to $h = 20$ base pairs of microhomology at the insertion site.) The parameters μ and σ are estimated from the insert size distribution using the median and 1.35 times the interquartile range, respectively. After constructing the graph of compatible read pairs, we cluster the read pairs by taking the largest clique(s) in each connected component (ignoring singleton components).

Discordant read pair FDR. We select "significant" clusters using a procedure that adapts to the coverage of the data and only requires a target false discovery rate (FDR). For a given DRP type with M total clusters, we assign each cluster j a heuristic score s_j indicating the strength of evidence. For inversions and tandem duplications, s_j is the cluster size. For insertions and deletions, s_j is a log-likelihood ratio statistic. For deletion clusters, we first estimate the deletion size $\widehat{D}_j$:

$$D_j = \min\left\{\frac{1}{|\mathcal{C}_j|}\sum_{i\in\mathcal{C}_j} z_i - \mu, \min_{i\in\mathcal{C}_j} b_i - \max_{i\in\mathcal{C}_j} a_i\right\},$$

where $\mathcal{C}_j$ is the set of read pairs within the cluster. The cluster score is

$$s_j = \prod_{i\in\mathcal{C}_j} \frac{\phi(z_i; \mu + D_j, \sigma)}{\phi(z_i; \mu, \sigma)},$$

where $\phi(\cdot; \mu, \sigma)$ is the density function for the normal distribution. The case for insertions is completely analogous.

To find an appropriate cutoff for cluster scores s_j, the observed discordant read pairs are shuffled throughout the interval $[0, L)$, where L is the length of the reference sequence with any assembly gaps removed. Clustering these permuted DRPs yields M_0 clusters with scores t_j, $j = 1, \ldots, M_0$. The distribution of these scores is used to estimate a null distribution of cluster scores: $\widehat{F}_0(t) \approx P(\text{null score} \leq t)$. We use $\widehat{\pi}_0 = \min\{1, M_0/M\}$ to estimate the proportion of null clusters in the unshuffled sample. If FDR control at level α is desired, we use a cluster score cutoff

$$s^* = \min_j \left\{ s_j : \frac{M\widehat{\pi}_0(1 - \widehat{F}_0(s_j))}{\#\{k : s_k \geq s_j\}} \leq \alpha \right\}.$$

Then all clusters i of the appropriate type with scores $s_i \geq s^*$ are kept as significant.

Breakpoint intervals. Discordant read clusters alone do not give precise breakpoint locations, so we provide an estimated breakpoint location along with a confidence interval. A natural estimate for the breakpoint position uses the locations of the 3′-most aligned bases among all reads in the cluster. For example, a deletion cluster with 3′ positions $\{a_i\}$ and $\{b_i\}$ yields breakpoint estimates $\max_i a_i$ and $\min_i b_i$. The true distribution of 3′ positions around

each breakpoint will depend on the insert size distribution. For simplicity, we approximate using a uniform distribution on an unknown but fixed genomic interval $[\underline{\theta}, \overline{\theta})$ and use a 95% confidence interval for the breakpoint position (see Supplementary Note for a derivation).

Breakpoint merging. After the above steps, we have breakpoint locations derived from soft-clipped reads, split reads, and discordant read pairs. We merge all overlapping breakpoint intervals and exact breakpoints (represented as intervals of length 1); we keep merged intervals with a supporting DRP cluster or at least 2 supporting split/clipped reads. Given merged breakpoint intervals $\{I_j = [a_j, b_j)\}_{j=1}^{n_{\text{bp}}}$, along with $I_0 = [0, 1)$ and $I_{n_{\text{bp}}+1} = [L, L+1)$, where L is the length of the chromosome, we define blocks $B_j = [b_{j-1} - 1, a_j)$ for $j = 1, \ldots, n_{\text{bp}} + 1$. (Note that if there is breakpoint uncertainty, i.e., if $b_j - a_j > 1$ for some j, then there will be gaps between some adjacent blocks.) These blocks are the genomic segments that will be rearranged during SV detection.

Insertion breakpoint detection. In order to mitigate some false positives, ARC-SV internally detects novel insertions (though the current version does not output them). Before the main inference procedure, at breakpoints derived from an insertion-type read cluster or from both orientations of soft-clipped reads, we compute the likelihood (model described below) of heterozygous and homozygous insertions over a range of lengths. If the likelihood surpasses that of no insertion then we add an insertion block S_i of the inferred length.

Adjacency graph

Graph creation. The adjacency graph is an undirected graph with two nodes for each block, B_j^{in} and B_j^{out}, corresponding to the start and end relative to the reference coordinates; insertion blocks S_i are treated likewise. "Block edges" are added between each block's pair of nodes. Other edges in the graph represent adjacencies between genomic segments and are called "adjacency edges." We do not include any adjacency edges across gaps in the reference.

Adjacency edges $B_j^{\text{out}} - B_{j+1}^{\text{in}}$ implied by the reference sequence are added automatically, as are edges connecting candidate insertions S_i to their flanking blocks. Other adjacency edges must be supported by the data. For example, a read pair aligned to the forward strand of B_i and to the reverse strand of B_k is considered to support the edge $B_j^{\text{out}} - B_k^{\text{in}}$ if the implied insert size under that adjacency falls within the expected range (middle 95% of the insert distribution). An exception is made for adjacencies $B_i^{\text{out}} - B_i^{\text{in}}$, i.e., those supporting a duplicated block. In this case paired-end support requires that the implied insert size under no tandem duplication falls outside the expected range. Split reads whose splits occur at candidate breakpoints automatically add support to the implied novel adjacencies. The final adjacency graph is formed by removing edges with only 1 supporting read, as well as edges spanning more than 2 Mb. This distance is longer than, for example, the largest deletion in the Venter gold set [19].

Graph partitioning. Paths through the adjacency graph will correspond to candidate SVs. To limit computational costs, we consider a set of subgraphs formed as follows. A reference block B_j as "spanned" by the adjacency graph if there is some edge connecting B_i

to B_k, where $i < j < k$. Define a "back edge" from B_j^{out} as any edge connecting to a block B_i with $i \leq j$; similarly, a back edge from B_j^{in} connects B_j^{in} to some block B_k with $k \geq j$. We call j a cut point in case B_j is not spanned and there are no back edges connected to B_j^{in} or B_j^{out}. Additionally, the first and last reference blocks, as well as blocks adjacent to assembly gaps, are considered cut points. Thus, we have cut points $1 = j_1 < j_2 < \cdots < j_P = J$ within the graph. For each 2 consecutive cut points j_m and j_{m+1}, we will consider the subgraph formed by blocks $B_{j_m}, \ldots, B_{j_{m+1}}$ as well as any insertion blocks contained within that genomic region. Subgraphs consisting of only a pair of reference blocks are discarded. Finally, each subgraph is extended on either side to ensure that some minimal amount of sequence flanks any structural variants, and subgraphs that overlap as a result are merged.

Subgraph traversal. Given two nodes s and e, a valid traversal through the graph is a path from s to e that alternates between block edges and adjacency edges, beginning and ending with block edges. Any valid traversal is equivalent to some sequence of oriented blocks, i.e., a genomic rearrangement. For tractable computation, we limit the number of times (by default, 2) each node may be exited using any back edges.

Probabilistic model

Suppose a subgraph R consists of reference blocks $B_{\text{start}}, \ldots, B_{\text{end}}$ and generates a set of traversals Θ. Each traversal θ corresponds to some haplotype of the region R having genomic positions $\mathcal{R}_\theta$. Our goal is to maximize the likelihood of the observed paired-end alignments, $P(\text{data}; \theta_1, \theta_2)$, as a function of the sample's (diploid) genotype $(\theta_1, \theta_2) \in \Theta^2$.

Fragment and alignment model. The probability model has two levels. First, DNA fragments are assumed to be generated from the region R in the target genome with probabilities proportional to the insert size density. In other words, the fragment $[x, y)$ is sampled with probability

$$P(x, y; \theta) = \frac{f(y - x)}{\sum_{u,v \in \mathcal{R}_\theta, u<v} f(u - v)}.$$

The second level of the probability model deals with the observed alignments to the reference, as the true coordinates of the sampled molecules are unknown. A single DNA fragment derived from the region R can be described as a sequence of oriented blocks along with base pair offsets for the first and last blocks. Mathematically, we represent each observed alignment a as a tuple, $a = (s, o, m)$, where s is the sequence of oriented blocks contained in the read, o is the offset for the $5'$ end of the read, and m indicates whether the read was mapped. Read pairs are written as $\underline{a} = (a_1, a_2)$. Note that, since s is directly observed, we assume alignments are perfect, including split alignments. For simplicity, we do not explicitly model soft-clipping; clipped bases on the $3'$-end are ignored, and reads with clipped $5'$ ends have their offsets o adjusted so the fragment length is calculated correctly.

Hanging read model. The alignment status of a read is encoded by an indicator m:

$$m = \begin{cases} -1 & \text{mapped distantly from mate} \\ 0 & \text{unmapped} \\ 1 & \text{mapped to } R. \end{cases}$$

A normal paired-end alignment $\underline{a} = (a_1, a_2)$ has $m_1 = m_2 = 1$. If exactly one of m_1, m_2 is equal to 1, we say the non-mapped read is "hanging." The case $m_1 = m_2 = 0$ occurs when neither end is mapped to the region R, e.g., when both ends fall within a novel insertion; we will perform inference conditional on $(m_1, m_2) \neq (0, 0)$.

In our hanging read model, the distribution of (m_1, m_2) depends on the unobserved variable $\underline{I} = (I_1, I_2)$, where $I_j = 1$ if read j is derived from a novel insertion. (Specifically, $I_j = 0$ if at least 20 contiguous bases are aligned to the reference.) The distribution $P(m_1, m_2 | \underline{I} = (0, 0))$ is estimated using the observed frequencies; we make these estimates during our initial sampling of reads to determine insert size cutoffs. Note that, when two reads in a pair map distantly from one another, say to regions R_1 and R_2, that pair is counted twice since the reads will appear in the likelihood model once for each region. To handle the case in which one read falls in an insertion, we assume that the reference derived read is mapped with the same marginal probability as before, i.e., $P(m_1 | \underline{I} = (0, 1)) = P(m_1 | \underline{I} = (0, 0))$; conditional on the reference-derived read mapping, the insertion-derived read is either unmapped or mapped distantly with equal probability. If the one reference-derived read is unmapped, or if both reads derive from insertions, then necessarily $m_1 = m_2 = 0$.

Likelihood evaluation. We define $C_\theta(\underline{a})$ to be the set of all fragments compatible with the paired-end alignment $\underline{a} = (a_1, a_2)$. Specifically, a fragment is $\phi = (x, y, \xi) \in \mathcal{R}_\theta^2 \times \{1, 2\}$, where ξ indicates whether position x corresponds to read 1 or 2. Compatibility means that the oriented block sequences and offsets contained in a_1 and a_2 are identical to those obtained by sequencing the given read lengths from the interval $[x, y)$.

We assume that the read pair $\underline{a}$ is observed conditional on the mapped read/reads falling within the region R. Recall that $\mathcal{R}_\theta$ is the set of locations within the region R under a given rearrangement. Let $\eta_\theta(\phi) \in \{0, 1\}^2$ indicate whether read 1 and/or read 2 fall within insertions. Then the likelihood of $\underline{a}$ is

$$\begin{aligned} P_\theta(\underline{a} | \text{mapped ends in } \mathcal{R}_\theta) &= \frac{\displaystyle\sum_{(x,y,\xi) \in C_\theta(\underline{a})} f(y - x) P\left(\underline{m} | \underline{I} = \eta_\theta((x, y, \xi))\right)}{\displaystyle\sum_{x,y \in \mathcal{R}_\theta : x < y} \sum_{\xi \in \{1,2\}} f(y - x) P\left(\underline{m} \neq (0, 0) | \underline{I} = \eta_\theta((x, y, \xi))\right)} \\ &\equiv \frac{p_\theta(\underline{a})}{G_\theta}, \end{aligned}$$

where $\eta_\theta(\phi) \in \{0, 1\}^2$ indicate whether read 1 and/or read 2 fall within insertions. Independence is assumed between all reads, and the likelihood of the data is

$$P_\theta(\text{data}) = \frac{1}{G_\theta^n} \prod_{i=1}^{n} p_\theta(\underline{a}_i).$$

Note that $p_\theta(\underline{a}_i) = 0$ if $\underline{a}_i$ has no compatible fragments under θ, allowing a single read to make the full likelihood 0. This is expected to occur whenever, e.g., there is a spurious split alignment or incorrect mapping orientation. Thus, we use the following "robust" likelihood,

$$\tilde{P}_\theta(\text{data}) = \frac{1}{G_\theta^n} \prod_{i=1}^{n} (\pi + (1-\pi) p_\theta(\underline{a}_i)).$$

This modified likelihood gives a small probability to all possible mappings, eliminating the effect of extreme outliers. Similar techniques for classical statistical problems are discussed in [20] — see, e.g., "redescending M-estimators". For this work we used a fixed value of $\pi = 10^{-6}$, but future versions of ARC-SV will adjust this parameter based on the scale of the insert size distribution.

Diploid rearrangements. Since we are working with diploid genomes, ARC-SV attempts to maximize the likelihood over diploid rearrangements $(\theta_1, \theta_2) \in \Theta_R^2$:

$$\tilde{P}_{\theta_1,\theta_2}(\text{data}) = \frac{1}{(G_{\theta_1} + G_{\theta_2})^n} \prod_{i=1}^{n} (2\pi + (1-\pi)(p_{\theta_1}(\underline{a}_i) + p_{\theta_2}(\underline{a}_i)).$$

Specifically, we use the following procedure:

1. Enumerate rearrangements $\theta \in \Theta_R$ based on traversals through the current subgraph R. If the size of Θ_R is greater than 1000, skip this subgraph.

2. Rank each θ by its homozygous likelihood, $\tilde{P}_{\theta,\theta}(\text{data})$, and let $\overline{\Theta}_R \subseteq \Theta_R$ contain the best 50 rearrangements.

3. Maximize the diploid likelihood $\tilde{P}_{\theta_1,\theta_2}$ over $(\theta_1, \theta_2) \in \overline{\Theta}_R^2$ to produce the final SV call.

Implementation details. Recall that there may be gaps between some blocks due to breakpoint uncertainty. It is possible (though rare in practice) that an alignment falls entirely into one of these gaps; these reads are ignored. Also, when rearranging blocks, we take the median of any breakpoint gap to be the best guess for the breakpoint location. For example, a deletion of B from ABC will leave behind half of any gaps between A and B, B and C.

ARC-SV only uses mapped reads with a MAPQ score of at least 20, except when determining candidate breakpoints from soft-clipped reads. The unmapped or "distantly mapped" ends of hanging reads are not subject to a MAPQ constraint.

Split reads with breakpoint positions that do not agree with our merged, filtered breakpoints are ignored. (Breakpoint filtering requires two supporting reads, so isolated split reads do not contribute to the final breakpoints.)

Computational validation of SVs

Motivation. In the validation of many previous methods, a simple SV call is often labeled a correct detection if it has high overlap to a known SV [7, 11, 17, 21]. Complex SVs

do not generally correspond to unique sequences of operations on intervals, even in unique portions of the genome. For example, may obtain $B_1 B_3' B_4$ from the reference $B_1 B_2 B_3 B_4$ by deleting B_2 then inverting B_3, or by inverting $B_2 B_3$ and deleting B_2. Thus, we validate SV predictions by direct sequence comparison to a ground truth, either a high-quality assembly or long reads. The SV is considered validated if the predicted sequence matches the truth; additionally, we check that the predicted sequence is significantly different than the original reference when subjected to the same comparison.

Altered reference and ground truth sequences. We construct a predicted sequence for each SV call by rearranging the reference genome accordingly and including 1000 bp of flanking sequence on each side. This altered sequence is annotated with the locations and sizes of any deletions as well as the boundaries of the rearranged reference blocks. We say block B_k is deleted if B_k is absent in the rearrangement and the subsequence $B_{k-1} B_{k+1}$ is present in either orientation.

If an assembly is used as ground truth, SV predictions from each chromosome are aligned to the assembled versions, as well as all unplaced contigs. When validating against a set of long reads, each SV prediction is aligned to the set of long reads mapping discordantly (i.e., with an indel ≥ 50 bp, a split alignment, or at least 100 soft-clipped bases) to the appropriate chromosome.

Alignment and heuristics. For SV validation, the following options are used with BWA–MEM [22]:

```
bwa mem -a -D 0 -w 1000 -x intractg {ground_truth} {altered_seq}
```

These options are chosen to obtain a large, sensitive set of alignments that include large indels.

Sometimes BWA–MEM will produce multiple alignments to a single ground truth sequence. We find the single best alignment chain according to the same scoring function used by BWA–MEM. Formally, we have alignment segments with coordinates $[s_i^q, e_i^q)$ in the query (i.e., predicted SV) sequence and coordinates $[s_i^r, e_i^r)$ in the reference (ground truth). Segment j can follow i in the alignment chain if their orientations are the same and if j appears after i in both query and reference: $e_i^q \leq s_j^q$ and $e_i^r \leq s_j^r$ (for positive strand alignments). In order to properly chain together large segments with small amounts of overlap — e.g., a pair of segments with query coordinates $[0, 100)$ and $[95, 200)$ — we split each segment at all points of overlap (in reference or query coordinates) with other segments.

Finding the best alignment chain from N segments costs $O(N^2)$ time with the standard dynamic programming solution (though speedups are possible [23]), so we incorporate several heuristics. We may ignore alignment gaps longer than the length L of the entire query sequence since, with our choice of penalties, such an alignment chain would be inferior to any single aligned segment. We also ignore segments smaller than 0.5% of the smallest non-flanking reference block's length.

It is possible that the predicted SV is correct, but (say) a long read that matches the prediction is too short to align across the entire flanking sequence. The best overall alignment may be to a read covering both flanking sequences in the query but not matching the predicted SV. Thus, when an alignment extends at last 200 bp out into the query's flanking

sequence but cannot continue because the reference sequence ends, we adjust the alignment score upward as if the alignment continued to the end of the query. This adjustment is only to select the best alignment chain and does not otherwise affect the validation score (described below) for that SV.

Finally, if the ground truth sequence is an oriented assembly contig, we ignore alignments to negative strand. We will fail to validate any variants contained within large inversions, but in general using negative strand alignments tends to validate many incorrect inversions, as in Figure 3.

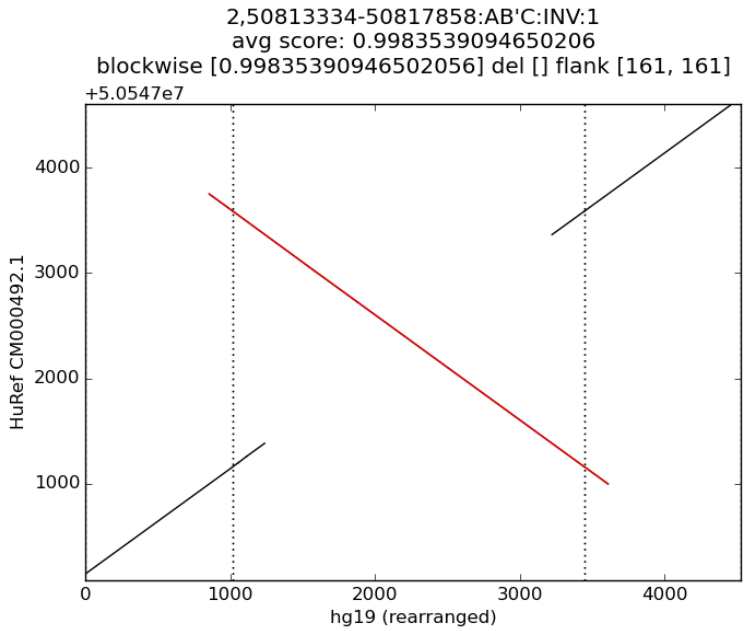


Figure 3: An inversion called by LUMPY in the HuRef sample and incorrectly validated by alignment to the reverse strand (best alignment chain in red).

Block-wise SV validation score. Using the best alignment, we compute a score for each simple deletion (i.e., a block not present but whose adjacent segments remain in standard orientation) and non-flanking block in the predicted SV. Validation requires that the minimum blockwise score is at least 0.5 and that at least 100 bp of flanking sequence is aligned on both sides of the SV region; validation of the complex SV $ABCBE$, for example, requires that the inner 3 blocks are aligned well, and that the alignment spans 100 bp into the flanking blocks. We additionally require the validation fails when comparing the SV to the original reference.

Our score for query blocks is

$$\frac{\#\text{aligned bases} - \#\text{assigned gapped bases}}{\text{block size}}.$$

The score for a deletion is 0 if not spanned at least 50 bp on each side by the alignment;

otherwise the score is

$$\frac{\text{deletion size} - \#\text{assigned gapped bases}}{\text{deletion size}}.$$

Alignment gaps may span multiple blocks or deletions, but are assigned conservatively to the smallest such spanned region. If segment j follows directly after i in the optimal alignment chain, we say there is a reference gap if $r_{ij} = s_r^j - e_r^i > 0$, and a query gap if $s_q^j - e_q^i > 0$. The reference gap's location in the query sequence is the interval $Q_{ij} = [e_q^i, s_q^j)$. We assign the gap error r_{ij} to the smallest non-flanking query block or predicted deletion within 1 bp of Q_{ij}. For this purpose, a length D deletion is represented by a window of total width $2\max\{50, D\}$. Query gaps are assigned in the same way, but only to deletions. An exception is made for query gaps within flanking sequence but within $\max\{50, \text{block size}\}$ of the outermost non-flanking block, in which case the gap may be assigned to that block.

Analysis of validation data. Our validation analysis excludes certain SV calls. As stated, variants with significant overlap to simple and tandem repeat regions were considered separately (Supp. Table 2); note these validation results are likely less reliable. We remove all variants affecting fewer than 50 bp, having predicted sequences exceeding 2 Mb, or overlapping gaps in the reference genome. We also excluded NA12878 calls for which the relevant region of the reference is covered by fewer than 5 long reads, in order to avoid bias against large inversions and duplications. To avoid overcounting both true and false positives, we remove duplicates (same SV type and 90% reciprocal overlap) within each call set, keeping the calls with higher validation scores; only simple SV calls are filtered in this way, primarily from Pindel, but also from LUMPY, DELLY, and SoftSV.

Finally, we did not consider breakend (BND) type events as reported by lumpy and delly, as they represent single breakpoint and not complex SV structures. Neither do we consider replacement (RPL) events from Pindel, as Pindel does not distinguish between inserted sequences that are *de novo* and those that derive from reference sequence.

Compound SV calls. Some calls from ARC-SV and SVelter consist of multiple non-overlapping simple SVs; these were termed compound events. Both the number of calls and the validation rate were slightly lower than for complex SVs (Supp. Fig. 10).

Random forest filtering. Random forest filtering was performed on all SV calls. We fit a separate random forest for each SV type, caller, and sample. The model predicts SV validation (a binary outcome) given features taken from the SV caller's output (see Supp. Table 5). We used the `randomforest`[1] package in R with 5000 trees per forest. Filtering was performed on each call set using the classifier trained on the opposite (independent) sample.

Experimental validation of SVs

Venter SVs. We experimentally tested 2 deletions, 3 tandem duplications, and 5 complex SV calls, as well as two control regions containing no ARC-SV calls (Supp. Table 6). First,

[1] https://cran.r-project.org/package=randomForest

we applied long range PCR with multiplexed sequencing on an Oxford Nanopore MinION device (flowcell version R7). Passing 2D reads were aligned to the reference with `bwa mem -w 1000 -x ont2d`, alignments with MAPQ ≥ 20 were divided into two groups based on the presence or absence of any large errors: split alignments, indels ≥ 50 bp, or soft-clipping ≥ 200 bp. We inspected a sample of 50 dotplots within each group to estimate the total number of reads supporting the reference, the variant call, some other structure, or else ambiguous. Sanger sequencing was also applied in cases where the long range PCR failed to amplify the alternative allele. For detailed experimental procedures, see Supplementary Note 3.

HepG2 SVs. For the HepG2 SV calls we conducted Sanger sequencing across the breakpoints of 43 deletions, 20 tandem duplications, and 5 non-tandem duplications (Supp. Tables 7, 8). We required breakpoints to be verified both by PCR and Sanger sequencing (with 70% identity to the expected sequence).

Reproducibility

Read alignment. Reads used for SV calling were aligned to the GRCh37 reference genome using `bwa mem` [22] version 0.7.12-r1044 with default parameters, and PCR duplicates were marked using Picard tools (http://broadinstitute.github.io/picard).

Other SV software. The following SV calling tools were used (with default parameters unless noted):

- DELLY [17]; version 0.7.3, using the default list of excluded regions
- LUMPY [18]; version 0.2.13, using the `lumpyexpress` command
- Pindel [24]; version 0.2.5b8, using the options `-x 4 -M 3 -B 0`
- SoftSV [25] version 1.4.2; we require both paired end and split read support
- SVelter [11]; Git commit `deb24b5` (July 8 2016); we remove calls with score ≤ 0

Data availability. The Venter/HuRef was sequenced to approximately 40× coverage by 2×100 bp paired-end reads (SRA accession SRX1016818) [7]. The NA12878 paired-end sequences were obtained from the Illumina Platinum Genomes Project (ENA accession ERR194147) and amount to 50× coverage by 2×100 bp reads [26].

For validation we have used the HuRef diploid assembly of Levy et al.:

`https://www.ncbi.nlm.nih.gov/assembly/GCA_000252825.1/`

and error-corrected PacBio reads derived from NA12878 by Mt. Sinai School of Medicine fror the Genome in a Bottle Consortium [16]:

`ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/NA12878/NA12878_PacBio_MtSinai/`

We also used two sets of high-quality deletions from previous literature: the Venter gold set from [7] and the NA12878 deletion call set from [6]. The NA12878 set was created with preliminary 1000 Genomes Project results, so we augment the calls with NA12878 deletions from Phase 3 of the Project [2] (excluding deletions overlapping $\geq$50% reciprocally with the original call set).

Software availability

The ARC-SV software will be available soon as a Python3 package.

References

[1] Alkan, C., Coe, B. P., and Eichler, E. E. Genome structural variation discovery and genotyping. *Nature reviews. Genetics*, 12(5):363–76, 2011. ISSN 1471-0064. doi:10.1038/nrg2958.

[2] Sudmant, P. H., et al. An integrated map of structural variation in 2,504 human genomes. *Nature*, 526(7571):75–81, 2015. ISSN 0028-0836. doi:10.1038/nature15394.

[3] Escaramís, G., Docampo, E., and Rabionet, R. A decade of structural variants: Description, history and methods to detect structural variation. *Briefings in Functional Genomics*, 14(5):305–314, 2015. ISSN 20412657. doi:10.1093/bfgp/elv014.

[4] Hehir-Kwa, J. Y., et al. A high-quality human reference panel reveals the complexity and distribution of genomic structural variants. *Nature Communications*, 7:12989, 2016. ISSN 2041-1723. doi:10.1038/ncomms12989.

[5] Guan, P. and Sung, W.-K. Structural variation detection using next-generation sequencing data: A comparative technical review. *Methods*, 102:–, 2016. ISSN 1046-2023. doi:http://dx.doi.org/10.1016/j.ymeth.2016.01.020.

[6] Parikh, H., et al. Svclassify: a Method To Establish Benchmark Structural Variant Calls. *BMC genomics*, 17(1):64, 2016. ISSN 1471-2164. doi:10.1186/s12864-016-2366-2.

[7] Mu, J. C., et al. Leveraging long read sequencing from a single individual to provide a comprehensive resource for benchmarking variant calling methods. *Scientific reports*, 5(August):14493, 2015. ISSN 2045-2322. doi:10.1038/srep14493.

[8] Quinlan, A. R. and Hall, I. M. Characterizing complex structural variation in germline and_nihms339318. *Trends in Genetics*, pages 1–19, 2012.

[9] Collins, R. L., et al. Defining the diverse spectrum of inversions, complex structural variation, and chromothripsis in the morbid human genome. *Genome biology*, pages 1–21, 2017. ISSN 1474-760X. doi:10.1186/s13059-017-1158-6.

[10] Yalcin, B., et al. The fine-scale architecture of structural variants in 17 mouse genomes. *Genome Biol*, 13(3):R18, 2012. ISSN 1465-6914. doi:10.1186/gb-2012-13-3-r18.

[11] Zhao, X., et al. Resolving complex structural genomic rearrangements using a randomized approach. *Genome Biology*, 17(1):126, 2016. ISSN 1474-760X. doi:10.1186/s13059-016-0993-1.

[12] Yang, L., et al. Diverse mechanisms of somatic structural variations in human cancer genomes. *Cell*, 153(4):919–929, 2013. ISSN 00928674. doi:10.1016/j.cell.2013.04.010.

[13] Iakovishina, D., et al. SV-Bay: Structural variant detection in cancer genomes using a Bayesian approach with correction for GC-content and read mappability. *Bioinformatics*, 32(7):984–992, 2016. ISSN 14602059. doi:10.1093/bioinformatics/btv751.

[14] Xia, L. C., et al. A genome-wide approach for detecting novel insertion-deletion variants of mid-range size. *Nucleic acids research*, 44(15):e126–e126, 2016.

[15] Levy, S., et al. The diploid genome sequence of an individual human. *PLoS Biology*, 5(10):2113–2144, 2007. ISSN 15449173. doi:10.1371/journal.pbio.0050254.

[16] Zook, J. M., et al. Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls. *Nature Biotechnology*, 32(3):246–251, 2014. ISSN 1087-0156. doi:10.1038/nbt.2835.

[17] Rausch, T., et al. DELLY: Structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, 28(18):333–339, 2012. ISSN 13674803. doi:10.1093/bioinformatics/bts378.

[18] Layer, R. M., Chiang, C., Quinlan, A. R., and Hall, I. M. LUMPY: a probabilistic framework for structural variant discovery. *Genome Biology*, 15(6):R84, 2014. ISSN 1465-6914. doi:10.1186/gb-2014-15-6-r84.

[19] Mu, J. C., et al. Leveraging long read sequencing from a single individual to provide a comprehensive resource for benchmarking variant calling methods. *Scientific reports*, 5(August):14493, 2015. ISSN 2045-2322. doi:10.1038/srep14493.

[20] Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. *Robust statistics: the approach based on influence functions*. John Wiley & Sons, 2011.

[21] Hormozdiari, F., Alkan, C., Eichler, E. E., and Sahinalp, S. C. Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Research*, 19(7):1270–1278, 2009. ISSN 10889051. doi:10.1101/gr.088633.108.

[22] Li, H. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[23] Zhang, Z., Raghavachari, B., Hardison, R. C., and Miller, W. Chaining multiple-alignment blocks. *Journal of Computational Biology*, 1(3):217–226, 1994.

[24] Ye, K., et al. Pindel: A pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, 25(21):2865–2871, 2009. ISSN 13674803. doi:10.1093/bioinformatics/btp394.

[25] Bartenhagen, C. and Dugas, M. Robust and exact structural variation detection with paired-end and soft-clipped alignments: SoftSV compared with eight algorithms. *Briefings in Bioinformatics*, 17(1):51–62, 2016. ISSN 14774054. doi:10.1093/bib/bbv028.

[26] Eberle, M. A., et al. A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree. *Genome Research*, 27(1):157–164, 2017.

[27] DasGupta, A. Finite sample theory of order statistics and extremes. In *Probability for Statistics and Machine Learning*, pages 221–248. Springer, 2011.

Supplementary Figures

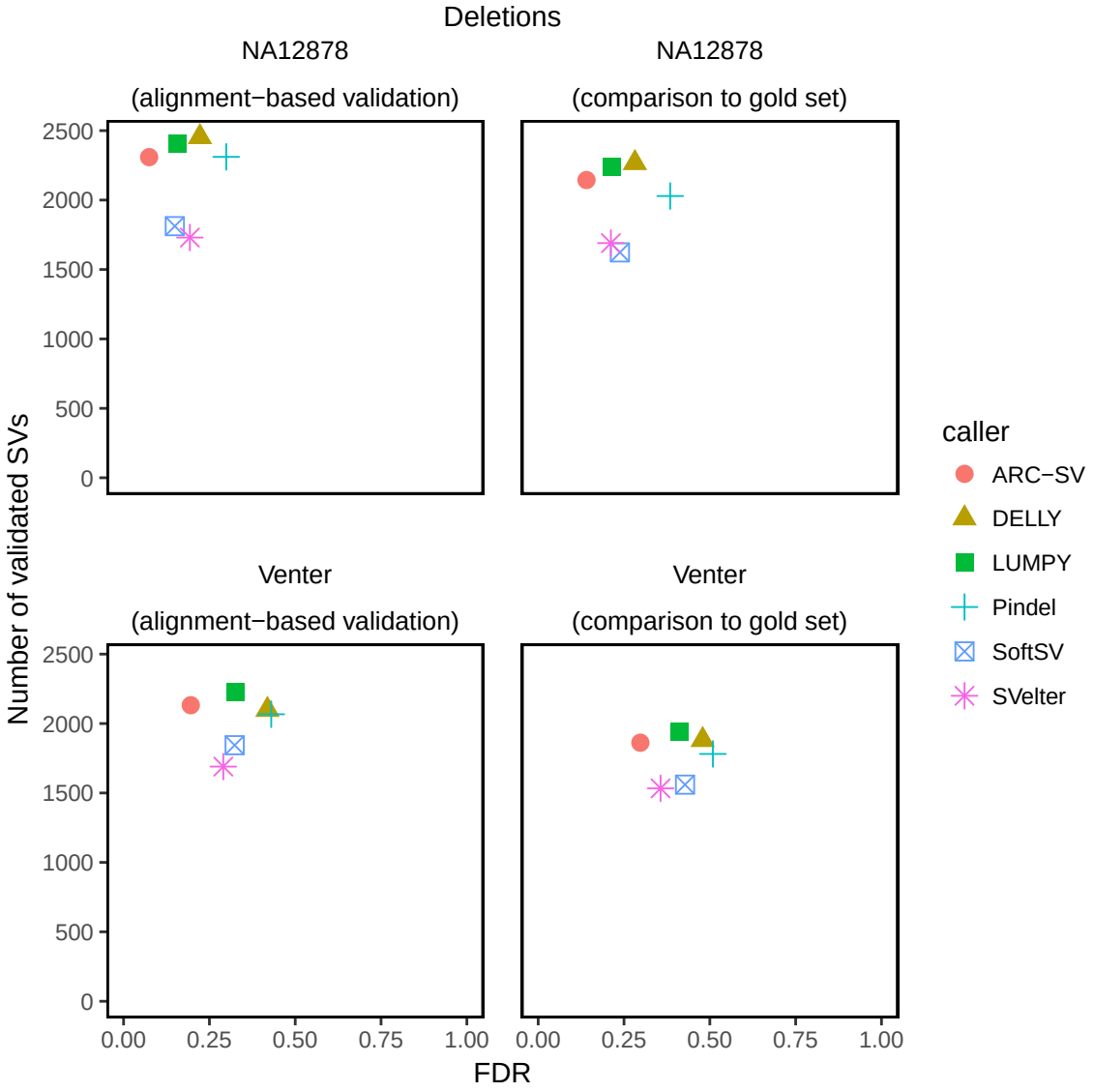


Figure 4: False discovery rate and sensitivity of deletion detections in Venter and NA12878, as evaluated by both validation methods. Results are prior to filtering by random forests.

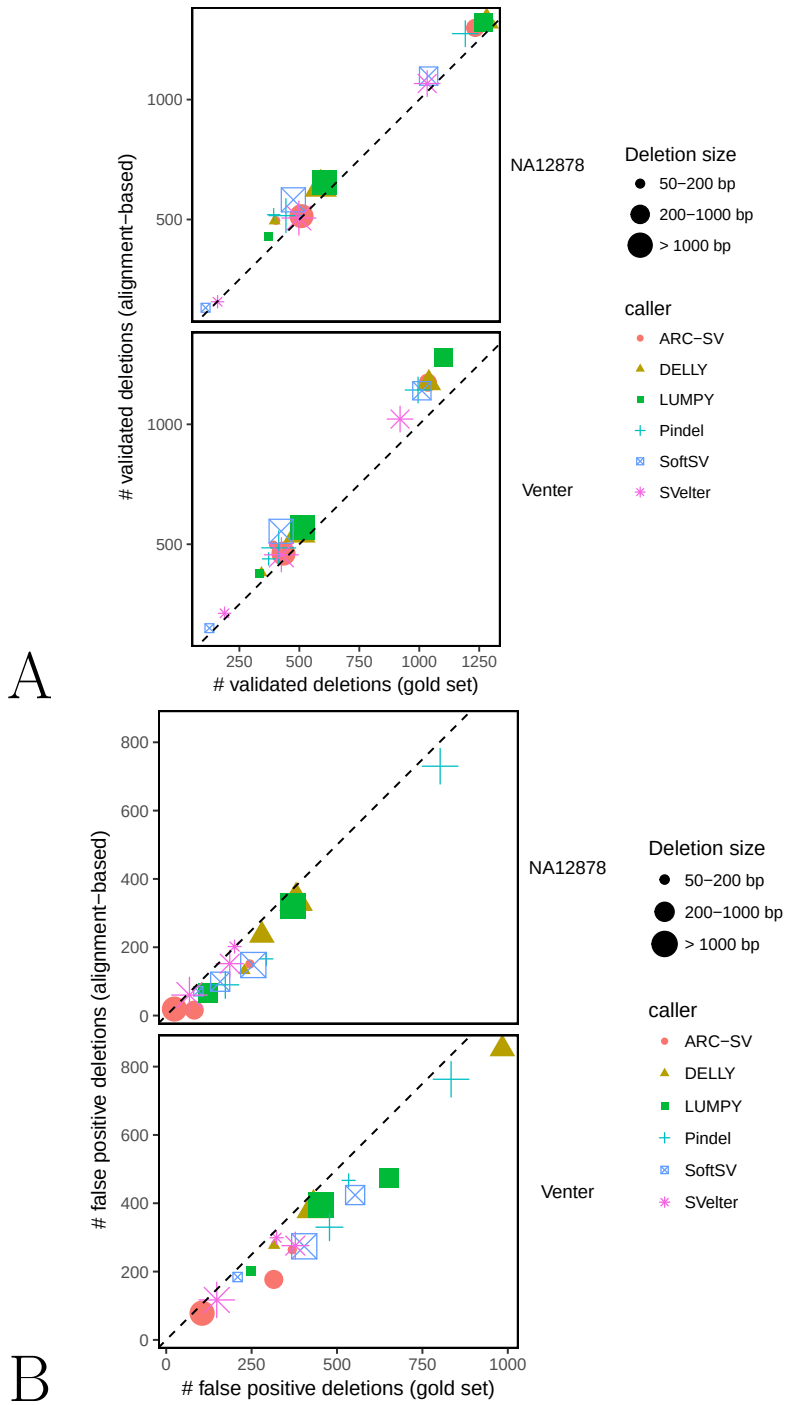


Figure 5: Comparison of (A) true positive and (B) false positive deletion counts based on our alignment-based validation and gold set overlap. Results are stratified by SV caller and SV size range. Results are prior to filtering by random forests.

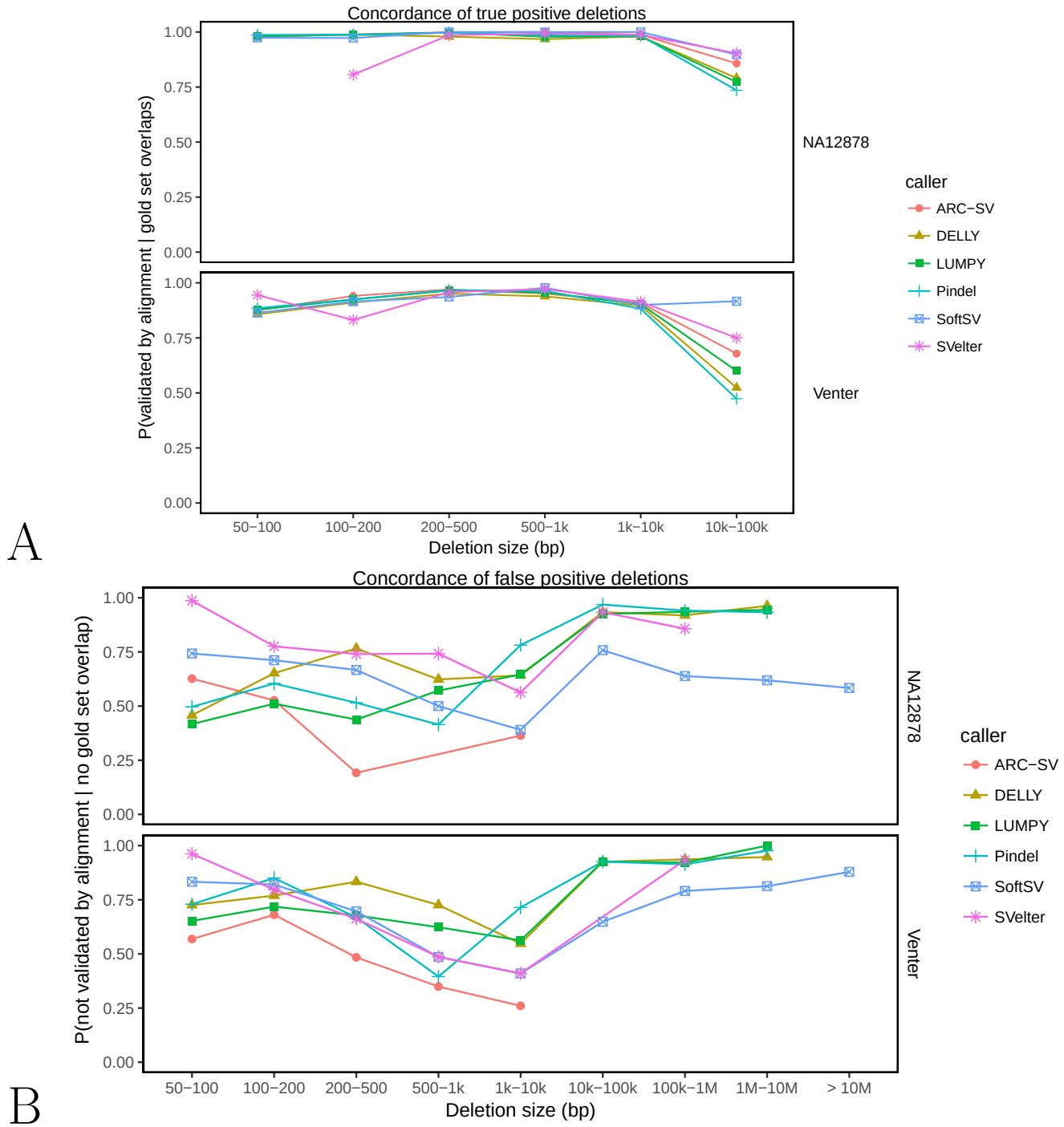


Figure 6: Concordance of (A) true positive and (B) false positive deletions between our alignment-based validation and gold set overlap (higher is better). Results are stratified by caller and SV size range, and only bins with more than 10 deletions are displayed. Results are prior to filtering by random forests.

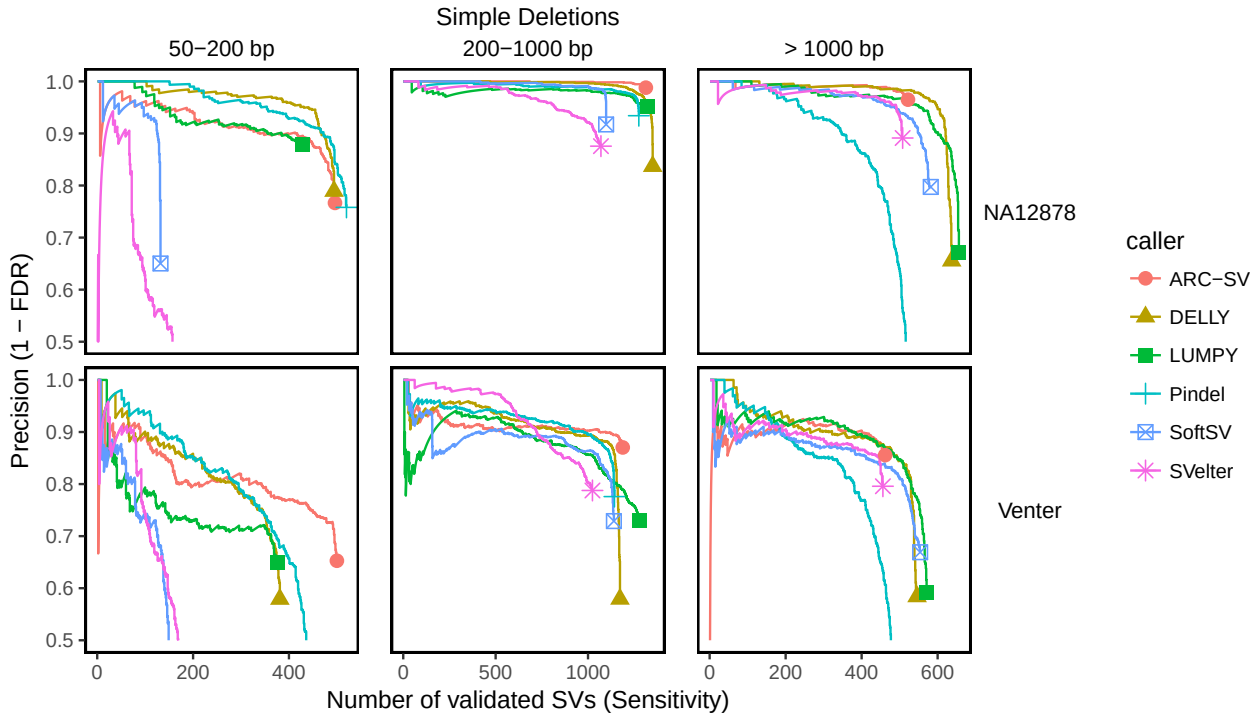


Figure 7: Sensitivity and precision of deletion calls across each size range. Curves are constructed from random forest filtering.

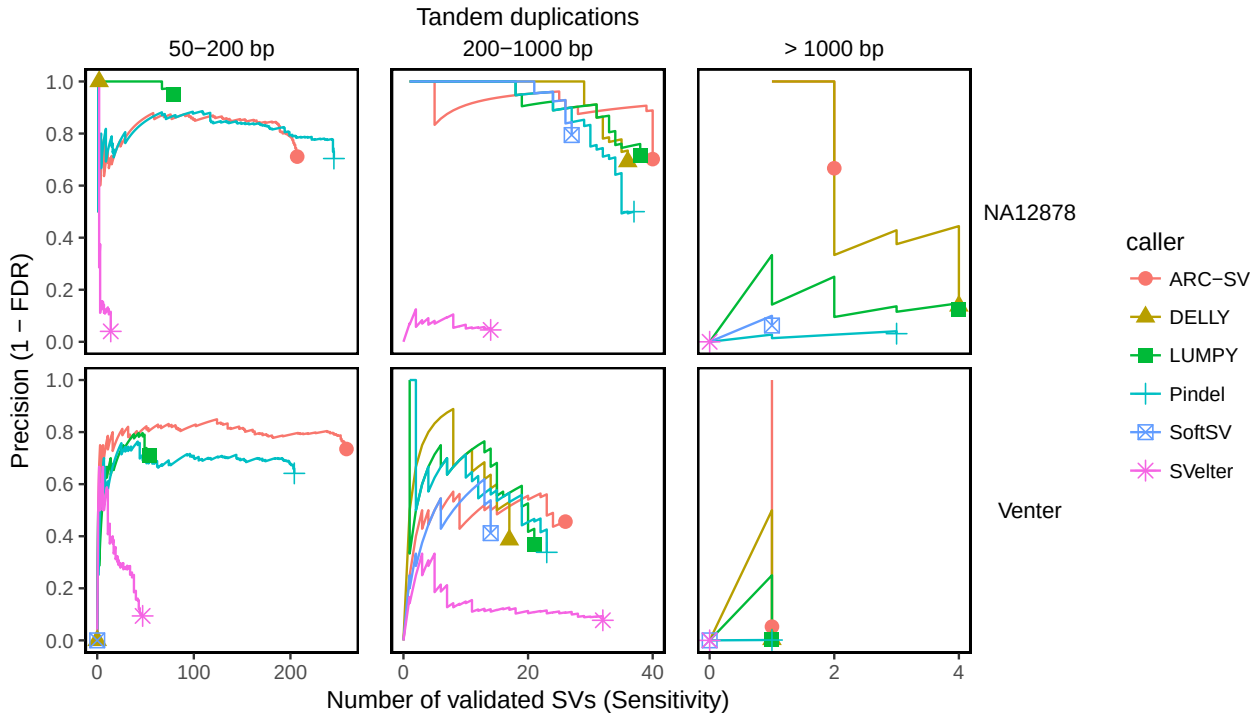


Figure 8: Sensitivity and precision of tandem duplication calls across each size range. Curves are constructed from random forest filtering. (Note the lack of data in the "> 1000 bp" category — the most sensitive callers achieve only 5 true detections across both samples.)

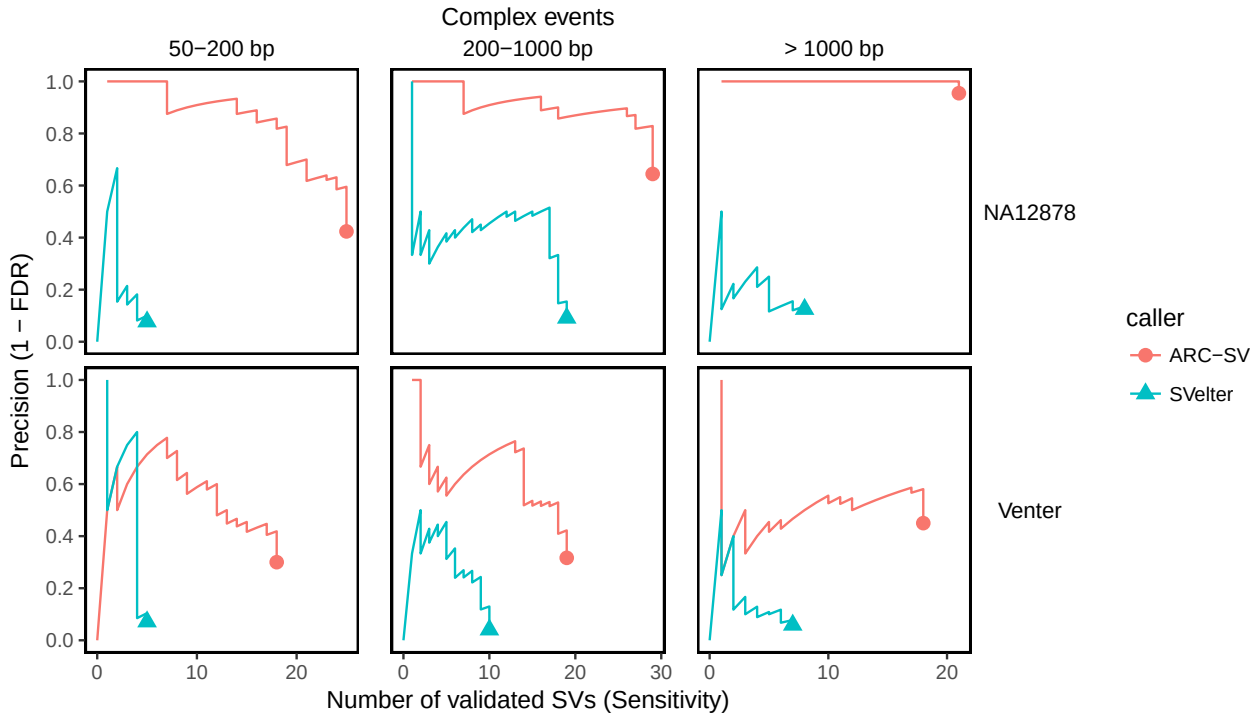


Figure 9: Sensitivity and precision of complex SV calls across each size range. Curves are constructed from random forest filtering.

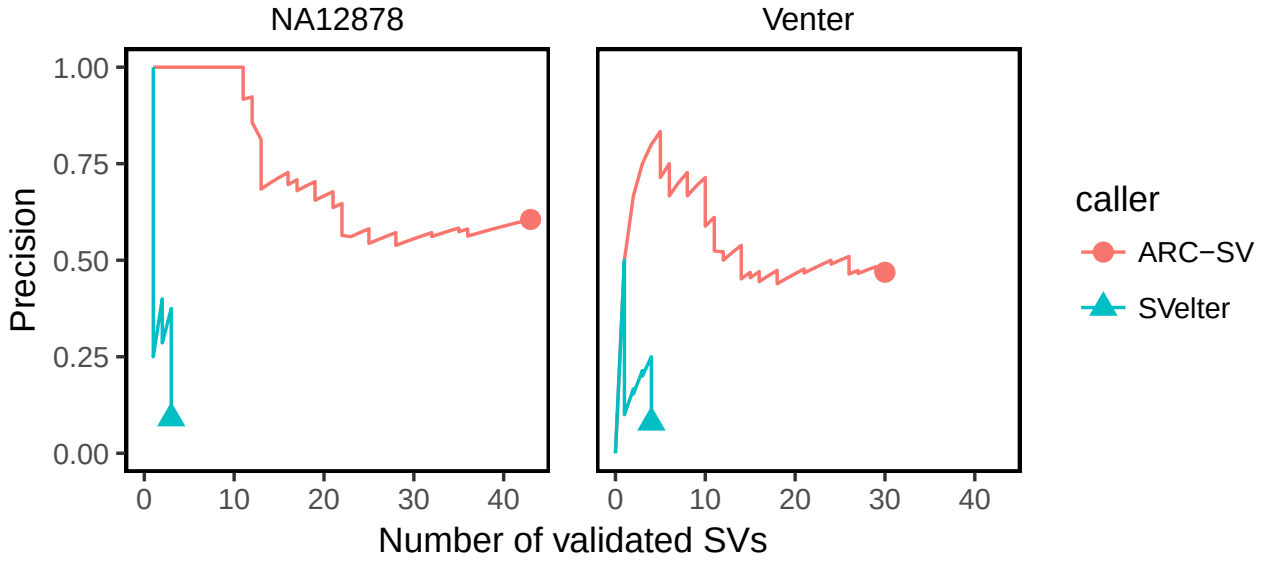


Figure 10: Sensitivity and precision of compound SV calls (see Methods). Curves are constructed from random forest filtering.

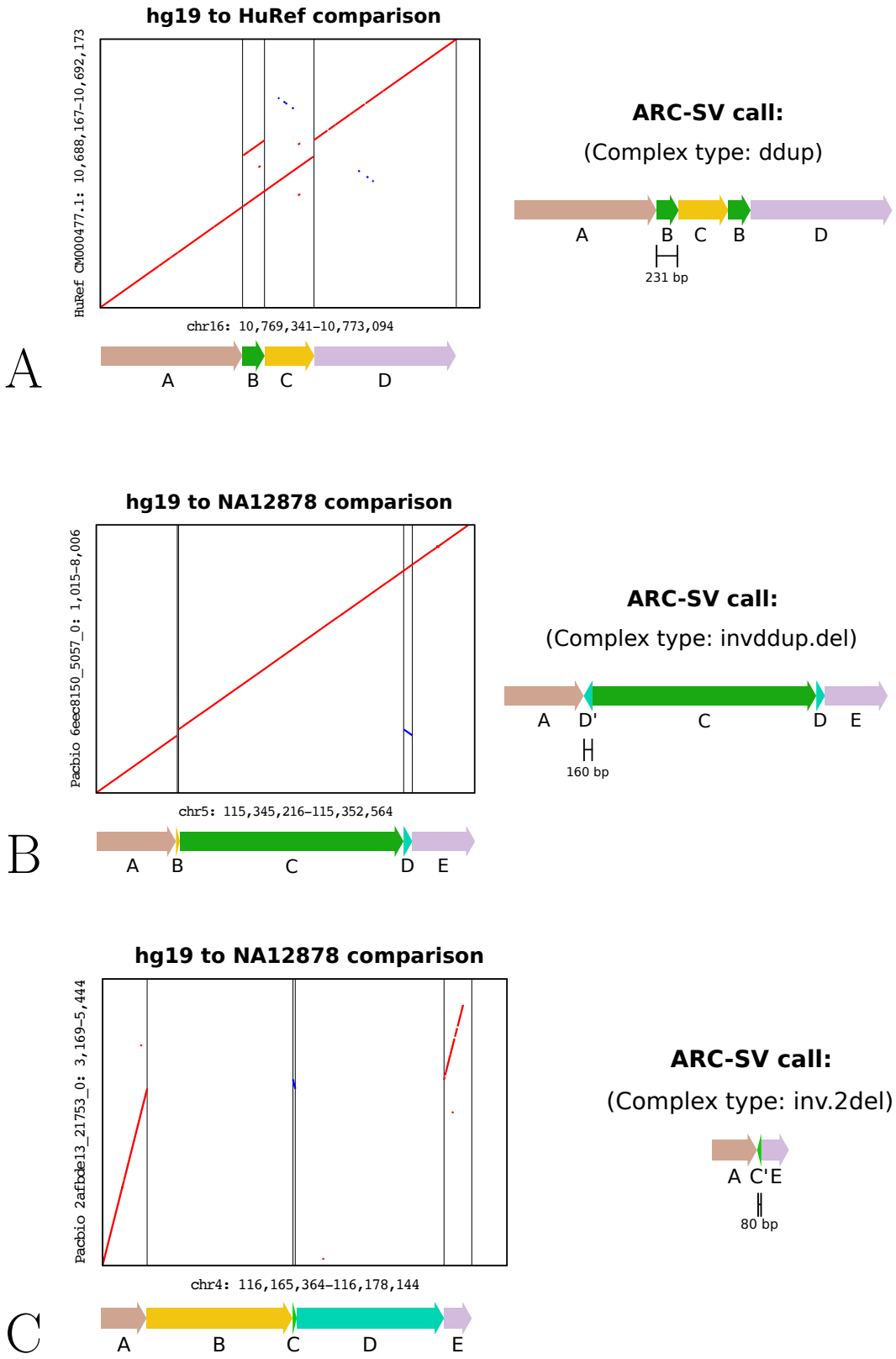


Figure 11: Validated complex ARC-SV calls: (A) interspersed duplication; (B) inverted interspersed duplication with deletion; (C) inversion with two flanking deletions

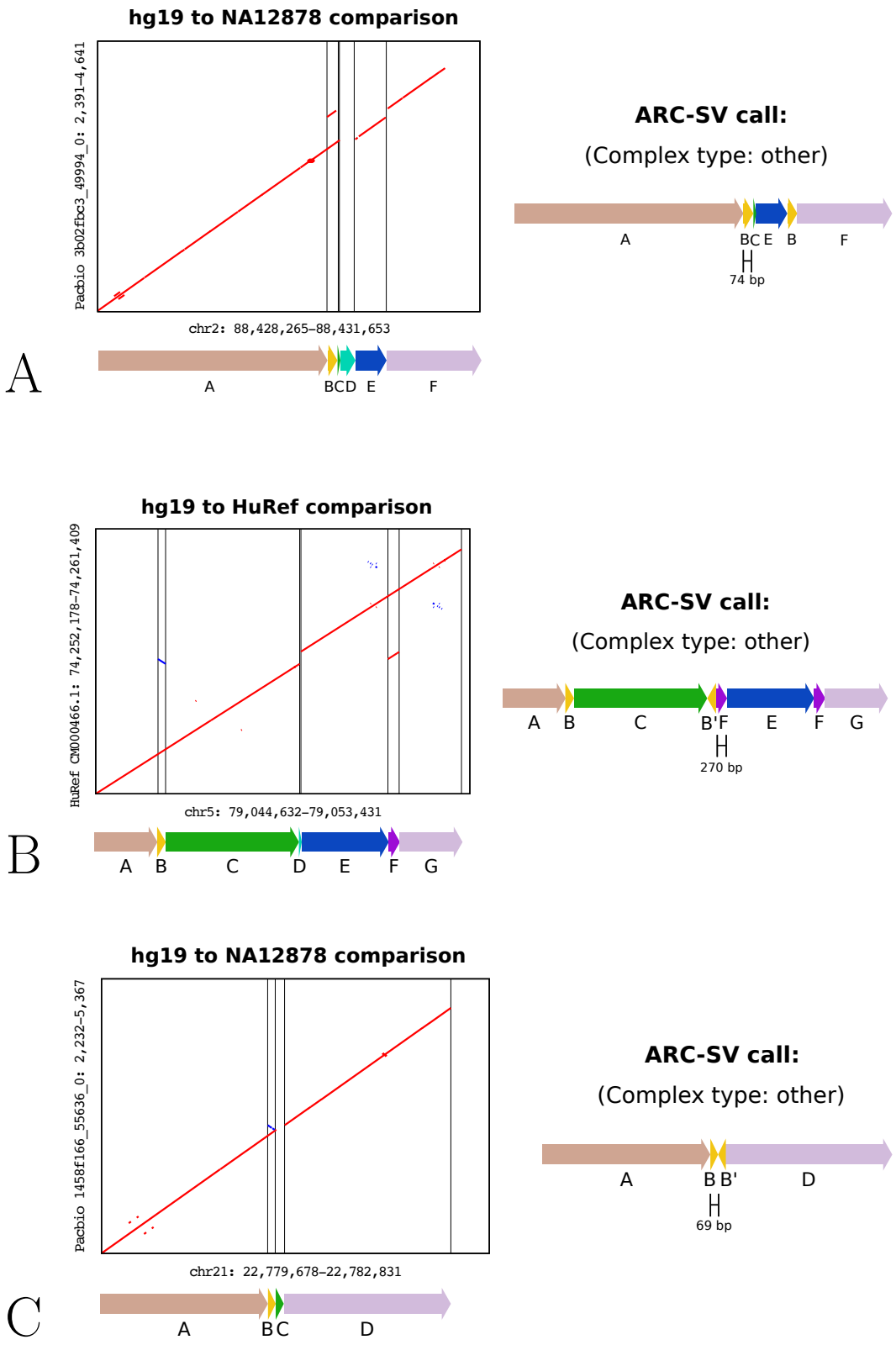


Figure 12: Examples of validated complex ARC-SV calls not fitting into our named categories.

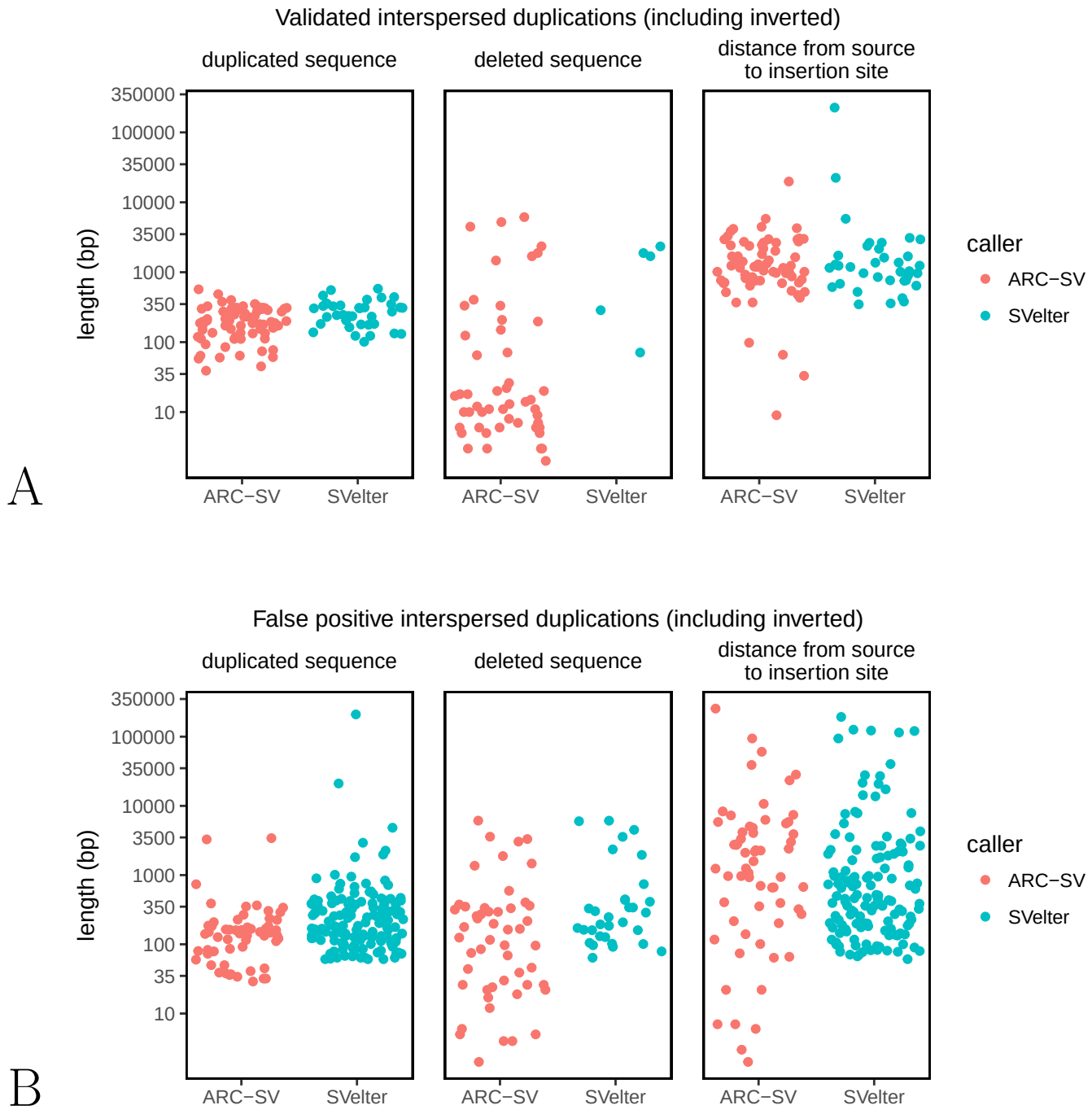


Figure 13: Sizes of interspersed (non-tandem) duplications. Only nonzero deletion sizes are shown.

		ARC-SV		SVelter		DELLY		LUMPY		Pindel		SoftSV	
		TP	Prec										
	Complex	**75**	**59%**	32	9%								
	Compound	**43**	**60%**	3	9%								
NA12878	Deletion	2331	**92%**	1735	81%	**2481**	77%	2407	84%	2311	70%	1812	85%
	Inversion	6	**50%**	9	26%	**14**	3%	11	15%	9	1%	10	22%
	Tandem dup	249	71%	28	4%	42	51%	121	**72%**	**285**	55%	28	56%
	Complex	**55**	**34%**	22	5%								
	Compound	**30**	**46%**	4	8%								
Venter	Deletion	2150	**80%**	1692	71%	2100	58%	**2226**	67%	2068	57%	1844	68%
	Inversion	5	**71%**	6	11%	**11**	3%	10	15%	8	1%	8	8%
	Tandem dup	**285**	**66%**	79	8%	18	6%	76	20%	228	20%	14	6%

Table 1: Overall SV calling results. TP = # validated SVs; Prec = precision = 1 - FDR. Results are before random forest filtering, and excluding simple and tandem repeat regions. Blank cells indicate that no calls were made.

		ARC-SV		SVelter		DELLY		LUMPY		Pindel		SoftSV	
		TP	Prec										
	Complex	**22**	**13%**	13	5%								
	Compound	**9**	**15%**	3	9%								
NA12878	Deletion	741	**61%**	488	54%	529	46%	407	42%	**1822**	29%	242	37%
	Inversion	0	0%	0	0%	0	0%	**4**	**13%**	4	2%	0	0%
	Tandem dup	246	**40%**	116	22%	105	29%	112	19%	**915**	17%	53	24%
	Complex	21	**17%**	**33**	9%								
	Compound	**28**	**20%**	5	8%								
Venter	Deletion	944	**57%**	637	54%	548	40%	721	48%	**1834**	34%	347	33%
	Inversion	1	**50%**	0	0%	2	3%	3	7%	3	1%	**5**	5%
	Tandem dup	304	**40%**	288	30%	29	7%	53	7%	**671**	14%	87	11%

Table 2: Results for SV calls within the excluded tandem and simple repeat regions.

		ARC-SV			SVelter		
		TP	N	Prec	TP	N	Prec
NA12878	ddup	**8**	9	**88%**	**8**	50	16%
	ddup.del	**14**	23	**60%**	1	11	9%
	dup.del	**3**	7	**42%**	0	14	0%
	dup.dup				0	32	0%
	inv.2del	**14**	14	**100%**	1	5	20%
	inv.del	3	6	**50%**	**7**	21	33%
	invddup	3	4	**75%**	**12**	23	52%
	invddup.del	**18**	23	**78%**	1	4	25%
	invddup.del.inv	**4**	4	**100%**	0	1	0%
	invddup.inv						
	invdup				**1**	39	**2%**
	other	**8**	36	**22%**	1	130	1%
	trans				0	6	0%
	trans.inv				0	1	0%
Venter	ddup	**6**	10	**60%**	4	48	8%
	ddup.del	**7**	29	**24%**	2	14	14%
	dup.del	**1**	5	**20%**	0	28	0%
	dup.dup	**1**	1	**100%**	0	56	0%
	inv.2del	**7**	14	**50%**	0	1	0%
	inv.del	**8**	11	**72%**	5	28	18%
	invddup	4	8	**50%**	**8**	19	42%
	invddup.del	**11**	26	**42%**	1	8	12%
	invddup.del.inv	0	6	0%	0	2	0%
	invddup.inv	0	3	0%	0	1	0%
	invdup				0	27	0%
	other	**10**	47	**21%**	2	190	1%
	trans				0	11	0%
	trans.inv				0	3	0%

Table 3: Complex SV calling results. TP = # validated SVs; N = # calls; Prec = precision = 1 - FDR. Results are before random forest filtering, and excluding tandem repeat regions. Blank rows indicate that no SV calls were made. See Table 4 for examples of each complex SV type.

SV type	**abbreviation**	**structure**
reference		$ABCDE$
deletion		$ACDE$
tandem duplication		$ABBCDE$
INV		$AB'CDE$
interspersed duplication	(ddup)	$ABCBDE$
interspersed duplication + deletion	(ddup.del)	$ABCBE$
tandem duplication + deletion	(dup.del)	$ABBDE$
double tandem duplication	(dup.dup)	$ABBCCDE$
inversion + 2 flanking deletion	(inv.2del)	$AC'E$
inversion + flanking deletion	(inv.del)	$AC'DE$
interspersed inverted duplication	(invddup)	$ABCB'DE$
invddup + intermediate inversion	(invddup.inv)†	$ABC'B'DE$
invddup + deletion	(invddup.del)	$ABCB'E$
invddup + deletion + intermediate inversion	(invddup.del.inv)	$ABC'B'E$
inverted tandem duplication	(invdup)	$ABB'CDE$
translocation	(trans)†	$ACBDE$
inverted translocation	(trans.inv)†	$ACB'DE$
other complex SV		$AEBDE$
		$AD'B'E$
		$ABC'BCD$
		$\vdots$
compound SV		$ACDDE$
		$ABBCD'E$
		$ACDE$
		$\vdots$

Table 4: Example genomic structures for different simple, complex, and compound SVs.
† No validated calls of this SV type.

ARC-SV (simple SV)	SV size; genotype; SR support; PE support; SV score; runner-up score; difference between best and runner-up scores; # paths (haplotypes) considered; # candidate breakpoints not used in call†; total breakpoint uncertainty
ARC-SV (complex SV)	in addition to the above: # affected base pairs; size of largest affected block; lowest split support across breakpoints; lowest paired-end support across breakpoints; number of breakpoints in SV
SVelter (simple SV)	SV size; genotype; # candidate breakpoints not used in call†; SV score
SVelter (complex SV)	in addition to the above: # affected base pairs; size of largest affected block; # breakpoints in SV
LUMPY	SV size; SR support; PE support; start/end position 95% confidence interval lengths; VCF tags INV_PLUS, INV_MINUS
DELLY	SV size; SR support; PE support; genotype; start/end position confidence interval lengths; VCF tags MAPQ, SRQ, CE, CT, FT, GQ, RC, RCR, RCL, CN, DR, DV, RR, RV
Pindel	SV size; SR support; genotype; length of microhomology (HOMLEN); Number of bases inserted in place of deleted code (NTLEN)
SoftSV	SV size; SR support; PE support

Table 5: Features used in random forest classification.
† For example, a deletion of B reported as $ACDE$ has 2 unused candidate breakpoints: between C and D; and between D and E

Supplementary Note: Uniform distribution inference

Let $X_1, \ldots, X_n$ be drawn iid from a Uniform distribution on $[a, b) \subset \mathbb{R}$. We construct a confidence interval for a having the form

$$[X_{(1)} - c(X_{(n)} - X_{(1)}), X_{(1)}],$$

where $c > 0$ depends on n and the confidence level $1 - \alpha$.

To solve for c we reduce the problem to one based on Uniform$(0, 1)$ order statistics $U_{(1)}, \ldots, U_{(n)}$. Defining

$$Z_j = \frac{\sum_{i=1}^{j} E_i}{\sum_{i=1}^{n+1} E_i},$$

where E_i are independent Exponential(1) random variables, it is known that $(U_{(1)}, \ldots, U_{(n)})$ and $(Z_1, \ldots, Z_n)$ have the same joint distribution [27]. Now we have

$$\begin{aligned}
1 - \alpha =& P\left(a \geq X_{(1)} - c(X_{(n)} - X_{(1)})\right) \\
=& P\left(\frac{a - X_{(1)}}{X_{(n)} - X_{(1)}} \geq c\right) \\
=& P\left(\frac{-U_{(1)}}{U_{(n)} - U_{(1)}} \geq c\right) \\
=& P\left(\frac{-E_1}{\sum_{i=2}^{n} E_i} \geq c\right).
\end{aligned}$$

The final term concerns Exp(1) divided by an independent Gamma$(n - 1,\ 1)$, which by definition follows a scaled F-distribution: $\frac{2}{2n-2} F_{2,2n-2}$. The confidence interval for b is given by the same argument.

Supplementary Note: Affected sequence calculation

A simple SV is defined by a single position and length in the reference genome. Complex SVs, however, may span large regions but make only small modifications to the sequence. In our analysis of complex SV calls, we keep the usual requirement that SVs must affect at least 50 bp of sequence. We propose a simple method that defines which reference segments are "affected" by an SV.

Suppose the reference region (separated into segments by the breakpoints) is given by $B_1^0 B_2^0 \dots B_n^0$, and the rearranged version is $B_{r_1}^{o_1} B_{r_2}^{o_2} \dots B_{r_m}^{o_m}$, where o_i is 1 if the segment is in reverse orientation. We find a pairwise alignment between these two strings of genomic segments (not the nucleotide sequences), treating blocks $B_i^{o_i}$ and $B_j^{o_j}$ as equal if $i = j$ and $o_i = o_j$. Alignments were computed using a very large score (1000) for matches and equally small penalties (-1) for mismatches and gap extensions. After finding an optimal alignment, blocks that are mismatched or opposite gaps in either sequence are considered "affected." The affected length is then computed as the sum of affected reference block lengths. Note that used the `Biostrings::pairwiseAlignment` function in R [1], which returns only a single optimal alignment. To partially mitigate the problem of non-unique optima, we computed a second alignment by switching the subject and query sequences and adding any additional affected blocks.

References

1. Pags H, Aboyoun P, Gentleman R and DebRoy S (2017). Biostrings: String objects representing biological sequences, and matching algorithms. R package version 2.44.2.