nature methods

Article

https://doi.org/10.1038/s41592-022-01674-1

SVDSS: structural variation discovery in hard-to-call genomic regions using sample-specific strings from accurate long reads

Received: 23 February 2022

Accepted: 8 October 2022

Published online: 22 December 2022

Check for updates

Luca Denti[1,6], Parsoa Khorsand[2,6], Paola Bonizzoni[3,7] ✉, Fereydoun Hormozdiari[2,4,5,7] ✉ & Rayan Chikhi[1,7] ✉

Structural variants (SVs) account for a large amount of sequence variability across genomes and play an important role in human genomics and precision medicine. Despite intense efforts over the years, the discovery of SVs in individuals remains challenging due to the diploid and highly repetitive structure of the human genome, and by the presence of SVs that vastly exceed sequencing read lengths. However, the recent introduction of low-error long-read sequencing technologies such as PacBio HiFi may finally enable these barriers to be overcome. Here we present SV discovery with sample-specific strings (SVDSS)—a method for discovery of SVs from long-read sequencing technologies (for example, PacBio HiFi) that combines and effectively leverages mapping-free, mapping-based and assembly-based methodologies for overall superior SV discovery performance. Our experiments on several human samples show that SVDSS outperforms state-of-the-art mapping-based methods for discovery of insertion and deletion SVs in PacBio HiFi reads and achieves notable improvements in calling SVs in repetitive regions of the genome.

SVs are defined as medium to large-size genomic rearrangements[1,2]. SVs can range from tens of basepairs to over megabases of sequence. Different types of SVs include balanced SVs, such as inversions and translocations, and unbalanced SVs, such as insertions and deletions[3]. The study and characterization of SVs has been driven by constant improvements in the technologies available to assay variants. Although SVs are not the most ubiquitous type of genetic variants, the total volume of basepairs impacted by SVs is far more than any other type of genetic variant, including single nucleotide variants (SNVs)[4,5]. Furthermore, recent studies of SVs using orthogonal technologies have shown that SVs are the least well-characterized type of genetic variant, with many basic questions, such as the average number of SVs per sample or sequence biases contributing to their formation, still not completely resolved[6–9]. In addition, the homology-driven mechanisms behind SV formation (for example, nonallelic homologous recombination) have contributed to the complexity of their systematic study[10]. It is believed that a large fraction of polymorphic SVs are still not fully characterized[11,12].

As our current understanding of SVs evolves, it is becoming clear that SVs are a main contributing factor to human diseases[13–15], population genomics[5,16] and evolution[17]. The comparative study of SVs in

[1]Sequence Bioinformatics, Department of Computational Biology, Institut Pasteur, Paris, France. [2]Genome Center, UC Davis, Davis, CA, USA. [3]Department of Informatics, Systems and Communication, University of Milano-Bicocca, Milan, Italy. [4]UC Davis MIND Institute, Sacramento, CA, USA. [5]Department of Biochemistry and Molecular Medicine, Sacramento, UC Davis, Sacramento, CA, USA. [6]These authors contributed equally: Luca Denti, Parsoa Khorsand. [7]These authors jointly supervised this work: Paola Bonizzoni, Fereydoun Hormozdiari, Rayan Chikhi. ✉e-mail: paola.bonizzoni@unimib.it; fhormozd@ucdavis.edu; rayan.chikhi@pasteur.fr

several closely related species (for example, great apes) has shown the considerable contribution of SVs to evolution (for example, through gene duplication or deletion[18,19]). Furthermore, the study of rare and de novo SVs in disease such as autism and epilepsy has proven the notable contribution of these variants in such conditions[15,20–23]. It is also known that somatic SVs are one of the main causative variants in different types of cancer[24–27].

With the advent of short- and long-read high-throughput sequencing technologies in the past decade, noteworthy progress has been made in our understanding of the abundance, complexity and importance of SVs[28–31]. Many methods have been developed for prediction of SVs using whole-genome sequencing (WGS) data produced from different sequencing technologies[32–42]. Most of these methods try to predict variants by detecting certain SV signatures (that is, read-depth, read-pair or split-read) in mappings of the reads to the reference genome[6,43,44] and are hence known as 'mapping-based' methods. Mapping-based methods have contributed to our understanding of the abundance of SVs in the general population and their role in disease[11,45,46]. Mapping-free methods are a more recent group of approaches that try to predict SVs without mapping the reads to the reference genome and instead by comparing sequence data between different genomes[34,47]. Finally, assembly-based approaches first assemble the sequenced reads into longer contigs and use the assembled contigs to predict variants[48–50]. Assembly-based methods have recently been shown to provide superior performance to mapping-based tools[51].

There are limiting factors for predicting SVs using each of these frameworks. Since most SV prediction tools use mappings of the reads to the reference genome for making SV calls, predicting SVs in highly repeated regions of the genome (for example, segmental duplications) where mappings can be inaccurate would be prone to false discovery. Reference genome gaps and misassemblies further complicate the prediction of SVs in these regions and result in decreased accuracy and increased variability across tools[8]. The mapping-free approaches, on the other hand, suffer from not being able to provide the loci of the event. Furthermore, fixed-length (*k*-mer) sequence comparisons performed in mapping-free tools can result in collapse of repeats and lower sensitivity/accuracy. Finally, assembly-based approaches are very computationally resource intensive and often require integration of data from multiple different technologies (that is, long reads, short reads and Hi-C)[51,52], higher sequencing depths (35× was reported[51]), and extensive polishing and postprocessing to yield a high-quality de novo assembly suitable for variant prediction, thus making them impractical for SV discovery across large populations.

Here, we propose a method called SVDSS that combines advantages of all three mapping-based, mapping-free and assembly-based approaches for predicting SVs. Our method uses mapping-free sample-specific signatures[53] along with mapping information to cluster reads potentially including SVs, and then performs local assembly and alignment of the clusters for SV prediction. With the combination of different analysis methods, our algorithm is able to improve SV calling performance particularly in repetitive areas of the genome compared with other contemporary approaches.

Results

Overview of SVDSS

We present SVDSS—a method for the discovery of SVs from accurate long reads (for example, PacBio HiFi). SVDSS takes as input a reference genome and a mapped BAM file and produces SV calls in VCF format along with assembled contigs for SV sites in SAM format. We use the concept of sample-specific strings (SFS), which we introduced previously as all the shortest substrings unique to one string set with regards to another string set[53]. We employ SFS here to pinpoint differences between reads and a reference genome[53]. Our method computes SFS for coarse-grained identification of potential SV sites. It assembles clusters of SFS from such sites to produce contigs that are then locally aligned to the reference genome to detect SVs. The main advantage of using SFS is that they are not limited to fixed-length seeds (unlike *k*-mers) and the algorithm can dynamically find the shortest string for covering the breakpoints of each variant, thus making SFS ideal for anchoring potential SV breakpoints.

SVDSS has three main steps as depicted in Fig. 1, sketched here and explained in more detail in the Online Methods:

(1) Read smoothing: reads are modified to remove sequencing errors, single nucleotide polymorphisms (SNPs) and small indels (<20 basepairs (bp)) that may interfere with SV calling (step (1); Fig. 1 and Supplementary Fig. 3). Smoothing greatly reduces the number of extracted SFS while increasing their specificity for the purpose of SV calling.

(2) SFS superstring construction: SFS are computed from the smoothed reads using the optimal Ping-Pong algorithm[53] (2A; Fig. 1) and then assembled into superstrings to reduce redundancy (2B).

(3) SV prediction using SFS superstrings: SFS superstrings are clustered based on position and extended to include unique anchoring sequences from the reference genome (step 3A; Fig. 1), further subclustered by length then assembled based on a partial order alignment (POA) approach to generate haplotype candidates (3B). Finally, SVs are called by aligning the resulting POA consensus(es) (3C).

In the following sections, using experimental analysis on multiple WGS samples, we demonstrate that SVDSS accurately predicts SVs and outperforms state-of-the-art approaches. We further show that the main contribution of our proposed approach is the ability to more accurately predict SVs falling in repeated regions of the genome compared with other methods.

Benchmark and evaluation callsets

One complexity in comparing different tools for calling SVs is the imperfectness of available callsets. Missing variants and potentially false predictions affect almost all published callsets, and even the most high-quality callsets have been reported to have a false discovery rate (FDR) of around 5% and a much higher false negative rate[6]. Furthermore, many callsets are constructed using state-of-the-art but imperfect SV prediction tools and are thus biased toward these methods[54]. For these reasons, we have opted out of using pre-existing callsets such as the 2020 Genome In A Bottle (GIAB) v.0.6 callset[45] in our experimental benchmarking. Instead we constructed our ground truth SV callsets from scratch using high-quality haplotype-resolved de novo assemblies generated by using many technologies (T2T CHM13 v.1.1, HG002 and HG007, described in Comprehensive detection of insertions and deletions). A similar ground truth construction strategy was employed in a 2022 GIAB benchmark[51], although focusing on a subset of medically relevant genes. We applied the assembly-to-assembly SV calling tool dipcall[54] to each assembly versus the entire GRCh38 reference genome (see Supplementary Information Section A for more details). The three VCFs built using dipcall and used as ground truth in our experimental evaluation are available at https://github.com/ldenti/SVDSS-experiments. For a detailed comparison of the HG002 callset built with dipcall and the v.0.6 callset provided by the GIAB project, we refer the reader to Supplementary Information Section B.

Comprehensive detection of insertions and deletions

We experimentally validated the accuracy of the SVDSS pipeline in calling SVs from three whole-genome sequenced samples sequenced using PacBio HiFi technology: the homozygous CHM13 sample from the telomore-to-telomere (T2T) project[52] and the HG002 and HG007 samples corrected using DeepConsensus[55]. These samples were chosen because of the availability of high-quality and effectively complete assemblies for them. Furthermore, the DeepConsensus corrected HG002 and HG007 samples show higher accuracy than

https://doi.org/10.1038/s41592-022-01674-1

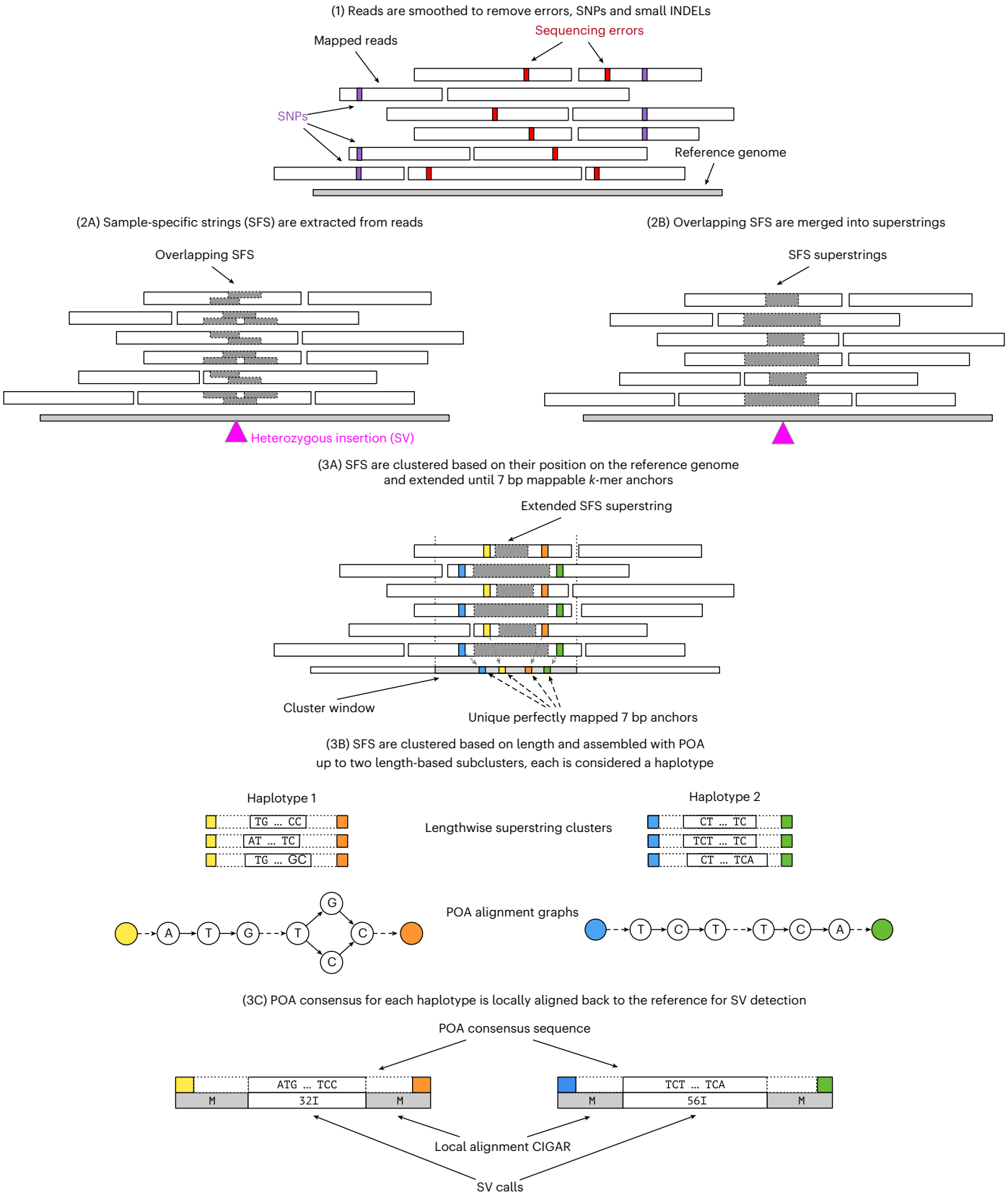

Fig. 1 | Overview of the SVDSS SV prediction pipeline. In step (1), reads are smoothed to remove SNPs and sequencing errors. SFS are extracted from reads (step 2A) and assembled into superstrings (step 2B). In step 3A, superstrings (gray) are clustered based on their placements on the reference genome and extended to uniquely mappable 7 bp anchors on each side (colored). Each cluster is further clustered into up to two subclusters based on length of the superstring (step 3B). Each subcluster represents a potential haplotype. The subclusters are assembled with POA to generate a consensus sequence (step 3C). The POA consensus for each cluster is aligned locally to the reference genome and SVs are called from the mapping information.

https://doi.org/10.1038/s41592-022-01674-1

Table 1 | Comparison of performance of SVDSS and other methods on calling SVs

		HG002			HG007			CHM13		
Region	**Tool**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Full Genome	SVDSS	88.4	**78.2**	**83.0**	**90.1**	**76.5**	**82.7**	87.3	**84.6**	**86.0**
	cuteSV	86.0	68.6	76.3	88.3	68.1	76.9	87.1	79.7	83.2
	pbsv	86.9	68.8	76.8	84.9	68.6	75.9	84.6	82.7	83.6
	sniffles	82.0	67.3	73.9	86.7	64.1	73.7	86.4	81.4	83.8
	SVIM	83.5	65.1	73.2	84.9	64.7	73.4	**90.1**	79.9	84.7
	debreak	**88.6**	67.5	76.6	**90.1**	64.2	75.0	83.7	79.6	81.6
Tier 1	SVDSS	95.2	**85.5**	**90.1**	95.2	**82.7**	**88.5**	95.3	93.4	**94.5**
	cuteSV	90.9	82.9	86.7	93.0	79.9	86.0	94.8	93.1	93.9
	pbsv	95.7	83.1	89.0	89.7	80.5	84.9	94.0	93.7	93.9
	sniffles	87.7	81.1	84.3	92.3	75.9	83.3	87.2	93.6	90.3
	SVIM	90.1	81.1	85.4	91.5	77.9	84.2	**96.6**	92.5	**94.5**
	debreak	**96.8**	82.5	89.1	**96.2**	76.4	85.2	93.7	93.0	93.3
Extended Tier 2	SVDSS	**82.7**	**72.3**	**77.2**	**84.6**	**70.2**	**76.7**	80.3	**77.4**	**78.8**
	cuteSV	80.9	57.0	66.9	82.3	56.0	66.6	79.9	68.1	73.6
	pbsv	78.4	57.2	66.1	78.8	56.4	65.7	76.0	73.3	74.6
	sniffles	77.8	56.1	65.2	80.3	52.1	63.2	72.7	73.2	72.9
	SVIM	76.4	52.0	61.9	76.2	51.2	61.2	**83.4**	69.3	75.7
	debreak	80.4	55.3	65.5	82.3	51.9	63.7	74.4	68.1	71.1

Results are shown in terms of P, R and F1 with bold faced numbers indicating best performance. Results are further broken down by considered regions of the genome. Tier 1 accounts for nearly half of SVs and consists of 86% of the genome. Extended Tier 2 accounts for the remaining 14% of the genome and 50% of SVs and includes repetitive regions that are more difficult to genotype. See Supplementary Fig. 5 for more detail on tiers. F1, F-measure; P, precision; R, recall

standard HiFi samples corrected using only pbccs[55]. The use of both homozygous (CHM13) and heterozygous (HG002 and HG007) samples allows for more comprehensive analysis and comparison of SV calling methods.

We mapped each sample against the reference genome using pbmm2 and then called SVs on each sample using the SVDSS pipeline. We compared our approach to five state-of-the-art mapping-based SV callers: pbsv, cuteSV[56], sniffles[41], SVIM[57] and a recent preprint on a POA-based method, debreak[58]. We ran each caller setting the minimum SV support to four when analyzing the 30× CHM13 sample and to two when analyzing the 15× HG002 and HG007 samples. We then examined their insertions and deletions calls. We validated the calls of each tool against the set of SVs constructed with dipcall using Truvari[59], a SV evaluation framework that reports precision, recall and F1 score for each method. We ignored genotype-level accuracy, that is, we checked only for the presence of the corrected allele (see Supplementary Information Section D for more information on how we ran Truvari, as well as other tools used in our analysis). From this comparison, we further exclude calls made in regions of the reference genome not covered by both haplotypes, as any such call would be classified as false positive regardless of correctness.

On HG002 and HG007 samples, SVDSS outperforms the recall of the other callers by 5–10% while achieving the highest (or second highest) precision on the full genome (Table 1, Full Genome rows). SVDSS has been able to report 2,342 (+10% relative to second-best approach) more correct calls on HG002 and 1,631 (+8%) more calls on HG007 without introducing many false calls. SVDSS also achieves the highest recall on CHM13 and reports 782 (+2%) more true positive calls than other methods while maintaining a very high precision. While SVDSS has the highest F1 score on CHM13, we note that the whole-genome improvements achieved by SVDSS over other approaches is less pronounced for this sample compared with the other two samples (improvement of 2–5% in recall and 1% in F1 while achieving similar precision to other tools). This is probably due to the homozygous nature of CHM13 making SV calling relatively easier for all approaches.

Figure 2a reports the length distribution of the SVs called by each tool on the HG007 sample. On HG007, the number of SVs reported by each tool ranges from 34,827 to 38,659, with SVIM reporting the lowest number of SVs and SVDSS reporting the highest number. Overall, all the tools report more insertions than deletions with shorter SVs (≤100 bp) being more frequent than longer SVs. Moreover, all the tools show a clear peak at around 300 bp, reflecting *Alu* mobile elements.

We also repeated the above experiment on HG007 using different aligners to test how SV callers are influenced by how reads are aligned. We tested all six callers in combination with minimap2 (ref. [60]) and ngmlr[41] (Supplementary Table 2). We also noticed that SVDSS substantially improves our ability to predict SVs in comparison with state-of-the-art approaches using minimap2 mapper, while being one of top performer tools using ngmlr mapper (Supplementary Table 2).

We also investigated how read coverage affects SV calling performance. To this aim, we subsampled the HG007 sample (coverage 15×) down to 5× and 10× and we ran the six considered approaches on these two newly created samples. Our SVDSS approach was also able to outperform other approaches using 10× sequencing coverage in all the metrics of interest (precision, recall and F1; Fig. 2b and Supplementary Table 3). When sample coverage is low (5×), pbsv achieves the highest recall (63.2%) at the expense of lower precision (58.6%), whereas other tools achieve similar high precision (ranging from 87.4% of SVIM to 92.9% of SVDSS) but low recall (ranging from 46.2% achieved by SVDSS to 51.6% achieved by cuteSV). As already pointed out by Chen et al.[58], debreak works poorly with low-coverage samples. On the other hand, with higher coverages of 10× and 15×, SVDSS achieves the best precision and recall, outperforming other approaches.

Finally, our pipeline has the second-lowest runtime among the considered methods behind cuteSV. More details on runtime and performance are available in Online Methods.

https://doi.org/10.1038/s41592-022-01674-1

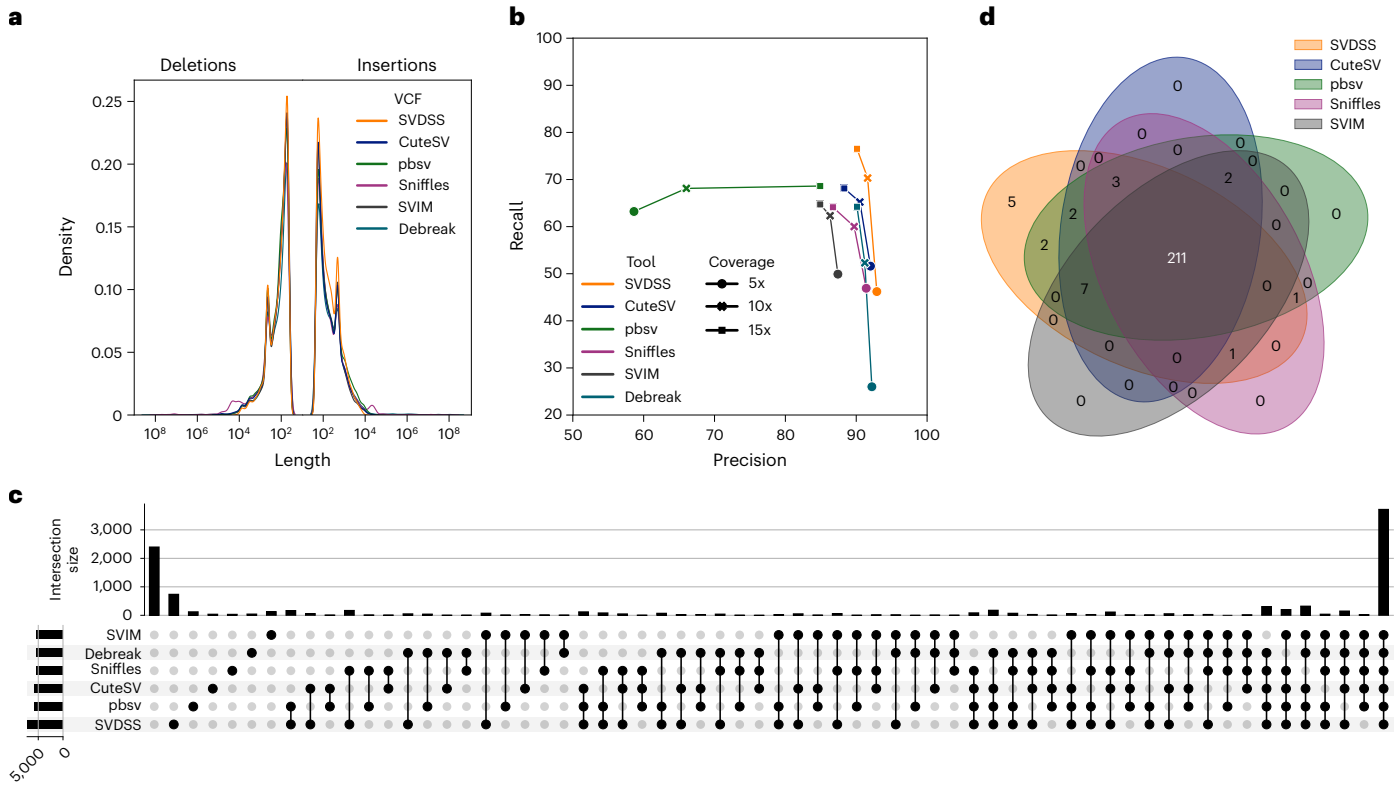

Fig. 2 | Extended comparative analysis of SV calls across methods. a, Distribution of SVs lengths reported by different tools on HG007 (Full Genome). **b**, Lineplot presenting results of the coverage titration for 5×, 10× and 15×. **c**, Analysis of shared calls (True Positives) between different tools on HG007 (Extended Tier 2). **d**, Venn diagram showing shared calls (True Positives) between different tools on the 273 medically relevant genes considered in the CMRG callset. To keep the Venn diagram cleaner, we decided to exclude debreak since it called the fewest True Positives. A supervenn figure including all tools is shown in Supplementary Fig. 7.

Improved SVs calling in hard-to-analyze regions

For further analysis, we partitioned the genome into two sets of intervals (tiers) as previously done by GIAB[45]. Tier 1 accounts for nearly 86% of the genome spanning 2.51 Gbp, includes 50% or less of the total expected number of SVs and is probably biased toward easy-to-call SVs (as stated in the README of the GIAB v.0.6 callset provided at https://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/analysis/NIST_SVs_Integration_v0.6/README_SV_v0.6.txt). Tier 2 accounts for nearly 0.8% of the genome and consists of around 6,000 difficult-to-genotype sites. The remaining 13% of the genome consists mostly of centromeres, telomeres and microsatellite regions (for example, short tandem repeats), which are generally more difficult to genotype because of their repeat structure and due to the ambiguities of the reference genome. Because the high-quality assemblies that are the basis of our analysis include effectively complete genomes for each individual, we decided to extend Tier 2 to also include these remaining 13% regions (Extended Tier 2). This way, we are able to more thoroughly evaluate the accuracy of each method across the entire human genome and we do not limit our analysis to easier-to-call regions (that is, Tier 1). Our final partitioning consists of Tier 1 and Extended Tier 2, represented in Supplementary Fig. 5.

In this analysis, we considered the callsets produced by SVDSS, cuteSV, pbsv, sniffles, debreak and SVIM starting from pbmm2 alignments. Table 1 reports the results of this analysis. Results on both tiers follow the same trend as with the full genome, with SVDSS managing to call more correct SVs without introducing many false calls. As expected, all tools achieve higher accuracy on Tier 1 regions, which are easier to analyze. Furthermore, we observed that the improvement between performance of SVDSS and other tools widens in the Extended Tier 2 regions of the genome (Table 1). Remarkably, on difficult-to-analyze regions (that is, Extended Tier 2), SVDSS achieves the highest recall, outperforming other callers by 15%, 14% and 4% on the HG002, HG007 and CHM13 samples, respectively.

To further provide evidence of correctness for true positive calls in these hard regions, we analyzed how these calls are shared among the tested callers using an upset plot[61]. Upset plots are an alternative to Venn diagrams that represent more conveniently the intersections of multiple sets. Figure 2c shows that, out of the 10,333 total SVs in the truth set for HG007 (that is, the dipcall callset), 3,720 (36%) are correctly called by all the tested approaches, whereas 2,399 (23%) are not detected by any tool. Remarkably, 739 SVs (7%) are detected only by our pipeline, partially explaining the higher recall it is able to achieve. SVIM has the second highest number of specific calls at 130. Supplementary Fig. 14 shows the distribution of SVDSS-specific versus SVIM-specific calls on chr1, chr2 and chr3 of the HG007 sample. SVDSS also detects the highest number of SVs that would have been exclusive to other tools, that is, 172 (1.6%) calls are shared by SVDSS and sniffles, and 169 (1.6%) are shared between SVDSS and pbsv.

We manually investigated some of the SVs that are exclusively called by SVDSS. Some of these calls are SVs that exhibit two different alleles on the two haplotypes. These SVs account for heterozygous SVs with two nonreference alleles (as defined in Denti et al.[62]), that is, SVs genotyped 1/2 (see two examples in Supplementary Figs. 10 and 11) as well as pairs of close SVs whose alleles come from different haplotypes (see an example in Supplementary Fig. 12). We observed that a total of 343 SVs called exclusively by SVDSS and matching dipcall predictions on the HG007 genome were located at exactly the same position as another called SV and are heterozygous SVs with two nonreference

alleles, while a total of 227 SVs are close (≤ 100 bp) to another predicted SV (Supplementary Fig. 6).

Hard-to-analyze regions harbor SVs of clinical importance

To perform a more thorough analysis of the HG002 individual, we considered the Challenging Medically Relevant Genes (CMRG) callset provided in Wagner et al.[51] and we evaluated callers' accuracy against it. The CMRG callset consists of 250 SVs falling in 126 challenging and medically relevant genes that were excluded from the previously published GIAB benchmark[45] due to their complexity: compound heterozygous insertions, complex variants in segmental duplications and long tandem repeats. The CMRG callset was created by diploid assembly of the haplotypes using hifiasm and then dipcall, proving once again the effectiveness of assembly-based methods for detecting hard-to-analyze SVs when well-curated assemblies are available.

As done previously, we computed the accuracy of SVDSS and the other five SV callers using Truvari. Out of the 250 SVs contained in the CMRG callset, SVDSS correctly called 232 SVs followed by pbsv (228) and cuteSV (225), SVIM (221), debreak (220) and sniffles (218). As shown in Fig. 2d (and Supplementary Fig. 7, where all tools are considered), five SVs are exclusive to SVDSS, while two are missed exclusively by SVDSS: one was reported but with a length just under the evaluation threshold of Truvari; the other was missed due to being detectable only in clipped reads, which SVDSS does not consider by default. We then manually investigated the SVs that were exclusively called by SVDSS, discovering that all exhibited two alleles, one per haplotype (that is, heterozygous SVs with two nonreference alleles). This result confirms previous findings[51] that heterozygous insertions in tandem repeats are among the most challenging classes of SVs to discover with current methods.

Figure 3 shows one of the SVDSS-exclusive SVs, a double insertion inside the *SLC27A5* gene on chromosome 19. Although the two haplotypes can be distinguished easily by visual inspection of adjacent heterozygous SNPs, the tested callers disagree on which allele to call. For instance only SVDSS calls two alleles of length 168 bp and 224 bp agreeing with the CMRG callset, whereas pbsv and sniffles report only one of the two (168 bp). Surprisingly, cuteSV, SVIM and debreak report a single allele of length 185 bp, which does not match any of the evidence from read alignment. Additionally, we considered the portion of the high-quality HG002 assembly covering that locus (chr19:58487900–58488500) and we checked its alignment against the reference genome (Fig. 3 and Supplementary Fig. 8). Although the considered locus is in a repetitive region (as also proven by the noisiness of the dotplots shown in Supplementary Fig. 8), the haplotype alignment confirms the presence of two allelic insertions of different lengths.

SVDSS has extremely low baseline error rate

Finally, we further investigated the lower bound on baseline FDR of SVDSS by comparing the HiFi reads from CHM13 against the high-quality T2T assembly[52] of the same sample. Given the almost perfect T2T CHM13 assembly produced using multiple orthogonal technologies, it is expected that an ideal SV caller would predict no SVs when comparing CHM13 reads against this assembly. Thus, we propose an experiment to establish a lower bound on the baseline FDR of different methods by comparing how many SV calls they report on the CHM13 HiFi reads against its T2T assembly.

Ideally, the SVDSS pipeline should generate zero SVs calls in this scenario as no SFS should be extracted when querying smoothed CHM13 reads against the T2T assembly. However, this will not be the case in practice due to mapping ambiguities in repetitive regions of the genome. Still, we expect the method to produce very few variant calls.

As a side-objective, we will also investigate the resulting SV calls to find if our method has discovered any true SVs missing from the T2T assembly. Due to the effectively homozygous nature of the CHM13 genome, any true variant discovered must be homozygous. However, it is possible that artifacts accumulated in the cell-line and actual heterozygosities in the genome may result in heterozygous SVs being reported.

We built the FMD index for v.1.1 of the CHM13 assembly and extracted SFS from CHM13 HiFi reads smoothed against the T2T assembly using this index. We then passed the SFS through the SVDSS pipeline for SV discovery. Our pipeline discovers a total of 102 SVs. For comparison, we repeated the above experiment with the other tools pbsv, cuteSV, SVIM, debreak and sniffles. Table 2 includes a summary of the results. We calculated the baseline FDR for each tool as the number of calls it makes against T2T divided by the number of calls it makes against GRCh38. SVDSS has the lowest number of calls against the T2T assembly and also has the lowest baseline error rate.

We further investigate if any of our calls are indeed true variants. The T2T project provides a list of known heterozygous sites on CHM13 (refs. [52,63]) and 13 of our SV calls intersect these regions, suggesting that they may be actual heterozygous alleles missing from the homozygous assembly. We also report the number of intersecting calls in Table 2 for every tool. SVDSS has the highest ratio of calls intersecting known heterozygous regions. We performed additional filtering of the calls using Merfin[64]—a variant call polishing tool that filters VCF files based on whether the variants introduce *k*-mers not found in the sequencing reads. Only one of our calls passes Merfin's filtering and we verify that the call seems to be a heterozygous site (Supplementary Fig. 9).

In summary, SVDSS produces only 102 calls using CHM13 HiFi reads against the T2T CHM13 assembly, some of which may be actual true heterozygous variants. Furthermore, with our earlier experiments showing an average of 33,000 SV calls per sample, this amounts to a baseline error rate of less than 0.4% showing that SVDSS is robust to false detection of variants.

Discussion

We introduced SVDSS—a method for SV discovery that combines advantages of different SV discovery approaches to achieve considerable improvements in SV calling. A highlight of SVDSS is its much higher recall compared with other approaches in repeated regions of the genome (that is, Extended Tier 2), and also its overall higher accuracy, in particular in repetitive and traditionally hard-to-genotype regions of the genome. We also observed that reducing sequencing coverage impacts SVDSS less than other approaches. Thus SVDSS can accurately predict SVs in low-coverage sequenced samples. Furthermore, using the recent CHM13 assembly produced by T2T consortium, we could estimate baseline error rare for each methods and further observed that SVDSS has the lowest baseline error rate, followed by sniffles.

While the availability of low-error long-read data enables more extensive variant discovery on new samples, SV discovery in repetitive regions of the genome such as STRs and microsatellites remains challenging but also hard to evaluate. This is evidenced by comparisons presented in this manuscript. Despite the considerable performance improvements of SVDSS in repetitive regions, precision and recall in these regions are still lower than in the rest of the genome.

SVDSS currently supports the discovery of unbalanced SVs, that is, deletions and insertions; however, as the underlying SFS signatures capture nearly all variation in the genome, a next step could be to extend the method to finding other classes of SVs such as inversions and duplications. Our current best technique for creating SV truth sets (dipcall) does not evaluate inversions and duplications, yet a recent study[28] provides one of the first gold standards.

Throughout this work, we highlight the importance of accurate benchmarks of SV calling methods. We evaluated SVDSS on a recent benchmark extensively curated over the HG002 sample[51] with the specific purpose of producing SVs occurring in genes of medical relevance. These genes are considered challenging for mapping-based and assembly-based SV prediction methods even from highly accurate long reads. This benchmark revealed that other methods fail to call

https://doi.org/10.1038/s41592-022-01674-1

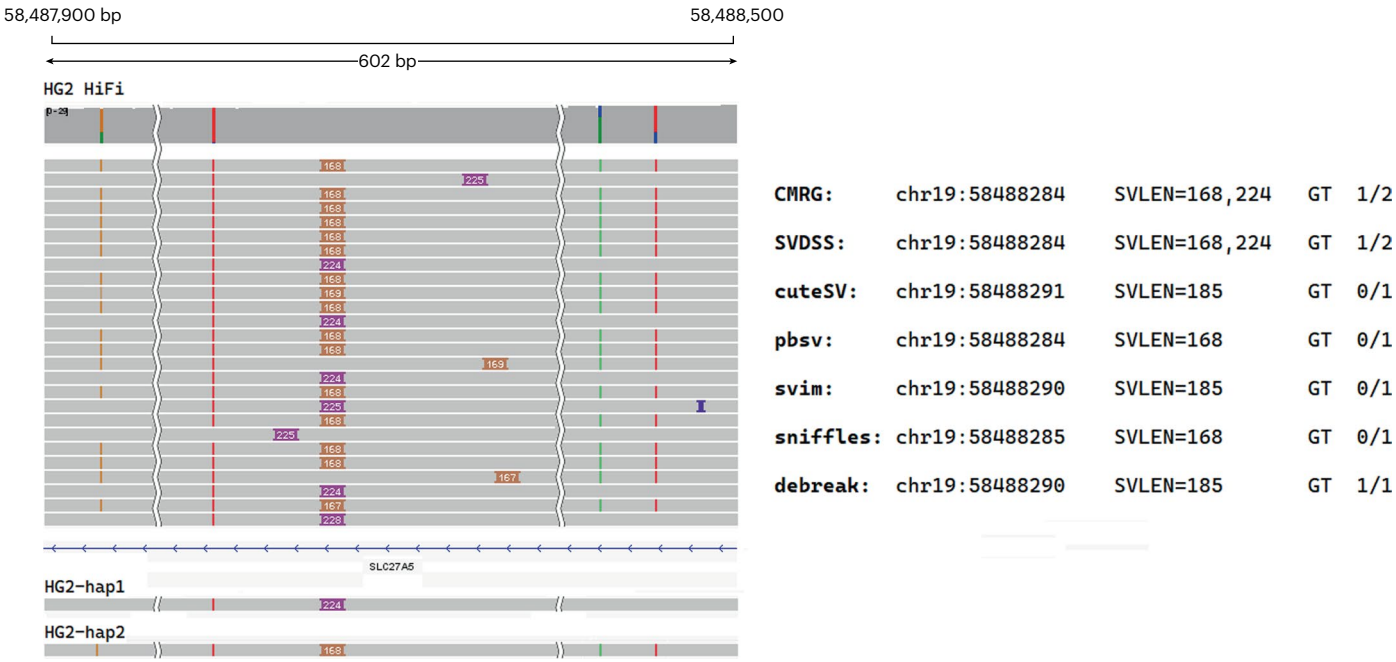

Fig. 3 | Example of an SV at a medically relevant gene that has been correctly called exclusively by SVDSS. Right, IGV sketch of the 602 bp region around the SV (full region reported in Supplementary Fig. 13). The sketch reports the HiFi reads alignment along with the haplotype alignment performed using `minimap2` (as part of the dipcall pipeline). Left, details of the SVs reported by the CMRG callset, SVDSS, and the other alignment-based callers considered in our evaluation.

Table 2 | Comparison of baseline FDR rate of SVDSS with other methods

Tool	GRCh38 calls	T2T calls	Baseline FDR	Het Intersections	Het Precision
svdss	**23,777**	**102**	**0.4%**	13	**12.7%**
cuteSV	22,654	667	2.94%	23	3.4%
pbsv	23,707	616	2.59%	28	4.5%
sniffles	22,680	314	1.38%	22	7.0%
SVIM	22,176	948	4.27%	**29**	3.0%
debreak	23,432	834	3.55%	24	2.8%

Number of SV calls against both the reference genome and the CHM13 assembly is included. Baseline FDR is calculated as division of first two columns for each tool. The last two columns report the number of known CHM13 heterozygous (Het) sites covered by each method and the precision of the method calculated as the number of covered heterozygous sites divided by the number of predicted calls. Boldface indicates best performance

heterozygous indels in highly homozygous regions or erroneous indels interpreted by a consensus approach. SVDSS is the only method able to discover five such SVs in medically relevant gene regions. We believe the current examples of accurate prediction of multiallelic heterozygous events based on SVDSS indicates the merit of extending this approach for genotype prediction of SVs.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41592-022-01674-1.

References

1. Alkan, C., Coe, B. P. & Eichler, E. E. Genome structural variation discovery and genotyping. *Nat. Rev. Genet.* **12**, 363–376 (2011).
2. Feuk, L., Carson, A. R. & Scherer, S. W. Structural variation in the human genome. *Nat. Rev. Genet.* **7**, 85–97 (2006).
3. Ho, S. S., Urban, A. E. & Mills, R. E. Structural variation in the sequencing era. *Nat. Rev. Genet.* **21**, 171–189 (2020).
4. Mills, R. E. et al. Mapping copy number variation by population-scale genome sequencing. *Nature* **470**, 59–65 (2011).
5. Sudmant, P. H. et al. An integrated map of structural variation in 2,504 human genomes. *Nature* **526**, 75–81 (2015).
6. Chaisson, M. J. et al. Resolving the complexity of the human genome using single-molecule sequencing. *Nature* **517**, 608–611 (2015).
7. Ebert, P. et al. Haplotype-resolved diverse human genomes and integrated analysis of structural variation. *Science* **372**, eabf7117 (2021).
8. Khayat, M. M. et al. Hidden biases in germline structural variant detection. *Genome Biol.* **22**, 347 (2021).
9. Sekar, S. et al. Complex mosaic structural variations in human fetal brains. *Genome Res.* **30**, 1695–1704 (2020).
10. Carvalho, C. M. & Lupski, J. R. Mechanisms underlying structural variant formation in genomic disorders. *Nat. Rev. Genet.* **17**, 224–238 (2016).
11. Audano, P. A. et al. Characterizing the major structural variant alleles of the human genome. *Cell* **176**, 663–675 (2019).
12. Zhao, X. et al. Expectations and blind spots for structural variation detection from long-read assemblies and short-read genome sequencing technologies. *Am. J. Human Genet.* **108**, 919–928 (2021).
13. Stankiewicz, P. & Lupski, J. R. Structural variation in the human genome and its role in disease. *Annu. Rev. Med.* **61**, 437–455 (2010).
14. Sharp, A. J., Cheng, Z. & Eichler, E. E. Structural variation of the human genome. *Annu. Rev. Genomics Hum. Genet.* **7**, 407–442 (2006).
15. Collins, R. L. et al. A structural variation reference for medical and population genetics. *Nature* **581**, 444–451 (2020).

16. Sudmant, P. H. et al. Global diversity, population stratification, and selection of human copy-number variation. *Science* **349**, aab3761 (2015).
17. Sudmant, P. H. et al. Evolution and diversity of copy number variation in the great ape lineage. *Genome Res.* **23**, 1373–1382 (2013).
18. Fortna, A. et al. Lineage-specific gene duplication and loss in human and great ape evolution. *PLoS Biol.* **2**, e207 (2004).
19. Hurles, M. Gene duplication: the genomic trade in spare parts. *PLoS Biol.* **2**, e206 (2004).
20. Wala, J. A. et al. Svaba: genome-wide detection of structural variants and indels by local assembly. *Genome Res.* **28**, 581–591 (2018).
21. Walsh, T. et al. Rare structural variants disrupt multiple genes in neurodevelopmental pathways in schizophrenia. *Science* **320**, 539–543 (2008).
22. Conrad, D. F. et al. Origins and functional impact of copy number variation in the human genome. *Nature* **464**, 704–712 (2010).
23. Marshall, C. R. et al. Structural variation of chromosomes in autism spectrum disorder. *Am. J. Human Genet.* **82**, 477–488 (2008).
24. The, I., of Whole, T. P.-C. A. & Consortium, G. et al. Pan-cancer analysis of whole genomes. *Nature* **578**, 82 (2020).
25. Li, Y. et al. Patterns of somatic structural variation in human cancer genomes. *Nature* **578**, 112–121 (2020).
26. Ye, K. et al. Systematic discovery of complex insertions and deletions in human cancers. *Nature Med.* **22**, 97–104 (2016).
27. Scott, E. C. et al. A hot l1 retrotransposon evades somatic repression and initiates human colorectal cancer. *Genome Res.* **26**, 745–755 (2016).
28. Porubsky, D. et al. Recurrent inversion polymorphisms in humans associate with genetic instability and genomic disorders. *Cell* **185**, 1986–2005 (2022).
29. Porubsky, D. et al. Recurrent inversion toggling and great ape genome evolution. *Nature Genet.* **52**, 849–858 (2020).
30. Wang, S. et al. Long read sequencing reveals sequential complex rearrangements driven by hepatitis B virus integration. Preprint at *bioRxiv* https://doi.org/10.1101/2021.12.09.471697 (2021).
31. Zook, J. M. et al. A robust benchmark for detection of germline large deletions and insertions. *Nat. Biotechnol.* **38**, 1347–1355 (2020).
32. Abyzov, A., Urban, A. E., Snyder, M. & Gerstein, M. CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome Res.* **21**, 974–984 (2011).
33. Iqbal, Z., Caccamo, M., Turner, I., Flicek, P. & McVean, G. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nat. Genet.* **44**, 226–232 (2012).
34. Chen, S. et al. Paragraph: a graph-based structural variant genotyper for short-read sequence data. *Genome Biol.* **20**, 291 (2019).
35. Lin, J. et al. Mako: a graph-based pattern growth approach to detect complex structural variants. *Genomics Proteomics Bioinformatics* **20**, 205–218 (2022).
36. Gardner, E. J. et al. The mobile element locator tool (melt): population-scale mobile element discovery and biology. *Genome Res.* **27**, 1916–1929 (2017).
37. Soylev, A., Le, T. M., Amini, H., Alkan, C. & Hormozdiari, F. Discovery of tandem and interspersed segmental duplications using high-throughput sequencing. *Bioinformatics* **35**, 3923–3930 (2019).
38. Ebler, J., Schönhuth, A. & Marschall, T. Genotyping inversions and tandem duplications. *Bioinformatics* **33**, 4015–4023 (2017).
39. Michaelson, J. J. & Sebat, J. forestSV: structural variant discovery through statistical learning. *Nat. Methods* **9**, 819–821 (2012).
40. Layer, R. M., Chiang, C., Quinlan, A. R. & Hall, I. M. LUMPY: a probabilistic framework for structural variant discovery. *Genome Biol.* **15**, R84 (2014).
41. Sedlazeck, F. J. et al. Accurate detection of complex structural variations using single-molecule sequencing. *Nat. Methods* **15**, 461–468 (2018).
42. Sindi, S., Helman, E., Bashir, A. & Raphael, B. J. A geometric approach for classification and comparison of structural variants. *Bioinformatics* **25**, i222–i230 (2009).
43. Medvedev, P., Stanciu, M. & Brudno, M. Computational methods for discovering structural variation with next-generation sequencing. *Nat. Methods* **6**, S13–S20 (2009).
44. Mahmoud, M. et al. Structural variant calling: the long and the short of it. *Genome Biol.* **20**, 246 (2019).
45. Zook, J. M. et al. A robust benchmark for detection of germline large deletions and insertions. *Nat. Biotechnol.* **38**, 1347–1355 (2020).
46. Belyeu, J. R. et al. De novo structural mutation rates and gamete-of-origin biases revealed through genome sequencing of 2,396 families. *Am. J. Human Genet.* **108**, 597–607 (2021).
47. Khorsand, P. & Hormozdiari, F. Nebula: ultra-efficient mapping-free structural variant genotyper. *Nucleic Acids Res.* **49**, e47–e47 (2021).
48. Maretty, L. et al. Sequencing and de novo assembly of 150 genomes from Denmark as a population reference. *Nature* **548**, 87–91 (2017).
49. Zhang, J.-Y. et al. Using de novo assembly to identify structural variation of eight complex immune system gene regions. *PLoS Comput. Biol.* **17**, e1009254 (2021).
50. Zhang, L., Zhou, X., Weng, Z. & Sidow, A. De novo diploid genome assembly for genome-wide structural variant detection. *NAR Genom. Bioinform.* **2**, lqz018 (2020).
51. Wagner, J. et al. Curated variation benchmarks for challenging medically relevant autosomal genes. *Nat. Biotechnol.* **40**, 672–680 (2022).
52. Nurk, S. et al. The complete sequence of a human genome. *Science* **376**, 44–53 (2022).
53. Khorsand, P. et al. Comparative genome analysis using sample-specific string detection in accurate long reads. *Bioinform. Adv.* **1**, vbab005 (2021).
54. Li, H. et al. A synthetic-diploid benchmark for accurate variant-calling evaluation. *Nat. Methods* **15**, 595–597 (2018).
55. Baid, G. et al. DeepConsensus improves the accuracy of sequences with a gap-aware sequence transformer. *Nat. Biotechnol.* https://doi.org/10.1038/s41587-022-01435-7 (2022).
56. Jiang, T. et al. Long-read-based human genomic structural variation detection with cuteSV. *Genome Biol.* **21**, 189 (2020).
57. Heller, D. & Vingron, M. SVIM: structural variant identification using mapped long reads. *Bioinformatics* **35**, 2907–2915 (2019).
58. Chen, Y. et al. DeBreak: deciphering the exact breakpoints of structural variations using long sequencing reads. *Res. Square* https://doi.org/10.21203/rs.3.rs-1261915/v1 (2022).
59. English, A. C., Menon, V. K., Gibbs, R., Metcalf, G. A. & Sedlazeck, F. J. Truvari: refined structural variant comparison preserves allelic diversity. Preprint at *bioRxiv* https://doi.org/10.1101/2022.02.21.481353 (2022).
60. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* **34**, 3094–3100 (2018).
61. Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R. & Pfister, H. UpSet: visualization of intersecting sets. *IEEE Trans. Vis. Comput. Graph.* **20**, 1983–1992 (2014).
62. Denti, L., Previtali, M., Bernardini, G., Schönhuth, A. & Bonizzoni, P. Malva: genotyping by mapping-free allele detection of known variants. *iScience* **18**, 20–27 (2019).

63. Mc Cartney, A, M. et al. Chasing perfection: validation and polishing strategies for telomere-to-telomere genome assemblies. *Nat. Methods* **19**, 687–695 (2022).
64. Formenti, G. et al. Merfin: improved variant filtering, assembly evaluation and polishing via k-mer validation. *Nat. Methods* **19**, 696–704 (2022).

Methods

Sample-specific string computation and assembly

Sample-specific SFS are defined as sequences that are specific to a 'target' set of strings (a genome or sequencing sample) with respect to another 'reference' set of strings (another genome or sequencing sample)[53]. The 'substring-free' part means that they do not occur as substrings of each other. Note that, in the context of SV discovery, the 'reference' will always be an assembled reference genome, for example, GRCh38, and the 'target' here is a set of reads. SFS can be optimally computed using the Ping-Pong algorithm, presented in Khorsand et al.[53]. Ping-Pong builds the FMD index[65] of the reference genome and queries the reads of the target sample against this index to report substrings that are not present in the index. The FMD index is a bidirectional text index with constant-time forward and backward search operations, thus allowing for efficient computation of SFS.

When SFS are computed between a reference genome and a target sample, they capture nearly all variations expressed in the sample with respect to the reference genome, as shown in Khorsand et al.[53]. Indeed, each sequencing read including a variant produces at least one SFS supporting the variant; hence, a variant will be supported by at least one SFS per read covering it. SV breakpoints usually result in new sequences that are captured as SFS. However, due to the 'shortest' property of SFS, the entire SV sequence is not necessarily covered by a single SFS: a read may produce several overlapping SFS for long variations. To remove unnecessary redundancy in the information captured by overlapping SFS, we newly assemble all such overlapping SFS into longer strings called 'superstrings.' Assembling SFS into superstrings also reduces the number of SFS by an order of magnitude, making any downstream analysis more efficient.

As SFS on each read are naturally sorted based on their start positions, the assembly stage can be implemented as a single pass over the SFS on each read, merging each SFS with the next one if they overlap. The resulting superstring can further be merged with the next SFS if they also overlap, and so on. More formally, on a read R where k consecutive SFS are overlapping such that $R[i_1,j_1]$ overlaps with $R[i_2,j_2]$ and $R[i_2,j_2]$ overlaps with $R[i_3,j_3]$ and ... $R[i_{k-1},j_{k-1}]$ overlaps with $R[i_k,j_k]$, we merge the strings into the single superstring $R[i_1,j_k]$.

The SFS assembly procedure effectively merges all the SFS belonging to the same variant into a single long superstring. This results in superstrings from the same variant to have similar length, sequence and position with respect to the reference genome which allows them to be clustered easily for SV prediction.

Read smoothing

The SFS extraction step (Ping-Pong algorithm) requires reads with low error rates for optimal performance as sequencing errors can result in millions of undesirable SFS. While most such SFS can be filtered later on, they can negatively affect the accuracy and will increase runtime by adding excess processing. Furthermore, the presence of millions of SNPs and small indels in a sample also results in tens of millions of additional SFS being extracted that are not directly useful for genotyping SVs. To solve both of the above problems, we introduce a preprocessing step called 'read smoothing' that aims to eliminate both sequencing errors and short variants from input reads. The smoothing algorithm starts from read alignments (a BAM file) and uses information from the CIGAR strings of each alignment to remove any short mismatch between a read and the reference genome.

In more detail, for segments reported as a match between a read and the reference genome (CIGAR operation 'M'), the algorithm replaces the read sequence with the corresponding sequence from the reference genome, automatically removing any single-base mismatches (that is, sequencing errors or potential SNPs) in the process. For short deletions (CIGAR operation 'D'), the algorithm removes the deletion from the read by copying back the deleted bases from the reference sequence. Short insertions (CIGAR operation 'I') are similarly smoothed by removing the inserted bases from the read. Using the default parameters, deletions and insertions are smoothed if they are shorter than 20 bp. Note that smoothing insertions or deletions, that is, removing them from the alignment, results in the extension of the `'M'` sections of the CIGAR string. Finally, soft-clipped regions (CIGAR operation 'S') are retained as they include potentially long inserted or deleted sequences: any SNP or sequencing error inside clipped regions cannot be corrected as a result. As a result of the smoothing algorithm, a smoothed read's CIGAR strings will have fewer edit operations than that the original read and it will consist of one or more very long 'M' segments with large INDELs in between, potentially surrounded with soft-clipped regions. Supplementary Fig. 3 illustrates the smoothing procedure on an example read. We note that the Ping-Pong algorithm will not produce any SFS that is entirely contained in an 'M'section of a smoothed read as the corresponding sequence has been replaced base-by-base with reference genome sequence. Therefore, the number of SFS extracted from smoothed reads is substantially smaller than the number of SFS extracted from original reads.

The smoothing algorithm only works with primary alignments and nonprimary alignments are ignored. This is to avoid complications arising from having multiple different smoothed version of reads with multiple alignments.

Smoothing relies on correctness of read alignments. If an alignment is thought to be inaccurate, the smoothing algorithm does not modify it. To this aim, during its execution, the algorithm keeps track of the average number of mismatches between the 'M' segments of alignments and the corresponding reference sequence: any read that has more than three times the average mismatch rate is ignored, that is, is not modified.

On a more technical note, we point out that the above modifications do not change the overall mapping of the read as the mapping positions (begin and end) remain the same. As a result, the algorithm will not change the order of the reads in a sorted BAM file. This allows us to quickly reconstruct a sorted BAM file without the need to sort it again. However, because the size of the reads may have changed, the index of the original BAM files is no longer valid for the smoothed BAM and it has to be indexed again with samtools index.

In our experiments, smoothing effectively reduces the number of extracted SFS by over 90%, while having effectively no impact on the SV calling pipeline's recall. Out of the 6.2 million reads for the CHM13 samples, around 5 million are smoothed and the rest are deemed to have unreliable mappings and are discarded. The 1.2 million nonsmoothed reads from CHM13 are responsible for more than 82% of all SFS extracted from that sample after smoothing. However, the SFS extracted from nonsmoothed reads do not contribute to increasing the method's recall at all. Indeed excluding the SFS extracted from nonsmoothed reads increases the method's precision while leaving the recall unaffected. This justifies the exclusion of nonsmoothed reads from the SVDSS pipeline. Further analysis shows that nearly all nonsmoothed reads map to centromere regions of the CHM13. Supplementary Fig. 4 shows the distribution of mapping positions of reads from chr1 on both CHM13 and GRCh38. The large gap around the centromere when mapping to GRCh38 explains the poor performance of nonsmoothed reads when predicting SVs against the reference genome.

In summary, read smoothing is a critical preprocessing step of the SVDSS pipeline. It reduces the number of retrieved SFS and increases the specificity of the extracted SFS which results in higher precision in predicting SVs without deteriorating recall. The procedure is also computationally very lightweight, as it essentially rewrites the BAM file in a single pass with minor modifications. As a result, smoothing is an effective method for increasing the specificity of SFS for SV calling and improving the computational efficiency of the pipeline.

SV Discovery

The main SV calling algorithm consists of three main steps (Fig. 1 steps 3A, 3B and 3C):

(1) Superstrings constructed from the SFS strings are 'placed' on the reference genome by extracting their alignments from read alignments. The superstrings are then clustered based on their aligned loci. Each cluster represents one or more SVs that are close to each other and may also contain multiple alleles.
(2) Each cluster is further clustered based on length to generate up to two haplotype candidates (taking into account the diploidy of the human genome). Each haplotype cluster candidate is assembled with POA to yield a consensus sequence.
(3) Each haplotype candidate is locally realigned back to the reference genome region corresponding to its cluster and SVs are called based on the alignment.

We will explain each step in more details in the following subsections.

Superstring placement and clustering. Aligning superstrings back to the reference genome would be time-consuming and error-prone due to their relatively short lengths. However these superstrings were already (indirectly) mapped as part of the mapping of the reads they are part of. Hence, in practice we do not align superstrings directly to the reference genome but instead their alignment is extracted from the alignment of their originating reads. We refer to this as superstring placement. Assuming $R[i,j]$ is a superstring that spans positions $i…j$ on read R, by knowing the mapping position of R, we can easily place the superstring on the reference genome by analyzing the corresponding CIGAR portion (that is, CIGAR sections covering positions $i…j$). As already pointed out, thanks to read smoothing, SFS (and consequently superstrings) cannot be contained entirely in an 'M' section of a smoothed read alignment (CIGAR) and therefore span its 'I' and 'D' sections. For superstrings spanning a 'D' section, all the bases are already placed on the reference genome and no additional computation is necessary. On the other hand, when a superstring spans an 'I' section, it often covers just a portion of the inserted sequence. In such a case, since the inserted sequence cannot be placed on the reference genome, it is challenging to fully place the superstring. To deal with this issue, we extend each superstring that does not fully cover an 'I' section until it fully covers it. In other words, we extend the superstring until it covers (on each side) a base that can be placed on the reference genome (that is, that is not part of the inserted sequence).

To further boost the informative content of the superstrings and to make the following steps of the pipeline easier and more accurate, each placed superstring is further extended on the read on both sides until we reach a perfectly mappable (can be mapped to the reference genome with no errors) and locally unique (not repeated in the considered window) k-mer anchor. The default value for k is 7 and the default window size is 100 bp on each side of the superstring. The superstrings that cannot be extended in this manner are ignored. Figure 1, step 3A shows this extension procedure. The k-mer anchoring idea was influenced by LongShot[66].

Finally, we cluster the superstrings based on their mapping locations: superstrings that have close enough mappings (by default less than 500 bp apart) are placed in the same cluster. The resulting cluster's interval is defined as the smallest interval in the genome that completely includes all of its superstrings and the includes either a single SV or several close or overlapping SVs possibly from different haplotypes.

POA assembly and SV detection. Each cluster so far includes one or more close SVs. However, as the human genome is diploid, the SVs might indeed be from different haplotypes. To resolve the different haplotypes, we further split each cluster into subclusters of superstrings of similar size and sequence. This is based on the assumption that different alleles at each site have different length and sequence. The similarity of sequences is calculated using rapidfuzz (available at https://github.com/maxbachmann/rapidfuzz-cpp). The two largest resulting subclusters (in terms of number of superstrings) are selected as haplotype candidates (considering the human genome is diploid). If only one subcluster is returned, it suggests a homozygous variant. SVDSS then computes a consensus sequence for each subcluster using POA.

Assume that a cluster c spans the interval $G[s_c, e_c]$ of the reference genome G. Most strings of the cluster only partially cover this interval (that is, they align to positions $[s,e]$ with $s_c \le s < e \le e_c$) while some others span the entire interval (that is, they align to positions covering at least $[s_c,e_c]$). To perform a more accurate POA, SVDSS extends all the strings in a cluster to be of the same length. Therefore, SVDSS fills the gaps preceding or following a superstring using the reference genome. For instance, if a superstring S aligns to $[s,e]$ with $s_c < s < e < e_c$, then the resulting sequence will be $G[s_G, s-1] + S + G[e+1, e_G]$ (where + is the string concatenation operator). The main goal of this extension is to summarize the information contained in a cluster and to minimize the difference between the superstrings coming from different reads. The extended superstrings in each subcluster are then aligned to each other using abPOA[67] to generate a consensus (Fig. 1 step 3B).

Finally, each POA consensus sequence is realigned locally to the reference genome window corresponding to its cluster using parasail[68]. The alignment's CIGAR information is analyzed to call and detect insertion and deletion SVs (Fig. 1 step 3C). A weight is assigned to each SV prediction based on the number of superstrings that support it. A higher support indicates a more confident call. By default, we filter out SV calls having less than four supporting superstrings. The confidence threshold can also be set at runtime using the --min--cluster--weight option.

SV chain filtering. Reads originating from loci in repetitive parts of the genome such as STRs may map to slightly different coordinates due to the similarity of the local sequence. This will result in multiple clusters (relatively close to each other) and multiple SV calls for the same variant but at slightly different positions. To reduce the number of false positives and eliminate such redundant calls, we perform a 'chain filtering' postprocessing step. This step sorts all predicted SVs based on coordinates and filters out consecutive SVs of the same type with similar sizes, keeping only the one with the highest number of supporting superstrings.

Implementation details

As a result of its many steps and the complexity of extraction SFS, SVDSS is more compute-intensive than other SV discovery methods, yet remains fast due to heavy optimization and deep parallelization. In this section, we elaborate on the performance of each of the steps and compare our runtime with other methods.

The FMD index creation and querying are handled internally by the FMD implementation from Li[65]. FMD index creation for the GRC38 reference genome takes around 30 min on 16 cores. The index can be reused for any number of samples so its creation is a one-time expense.

Read smoothing is an IO-intensive step and greatly benefits from enabling the multithreaded BAM decoding functionality built into htslib[69] by setting the bgzf_mt flag when opening a BAM file. To further improve BAM decompression performance, we require that htslib is built with libdeflate in place of the default BAM decoder. For HiFi data at 30× coverage the smoothing algorithm takes about 15 min to run on 16 cores.

SFS extraction is the most computationally intensive step and takes about 45 min on 16 threads for the CHM13 HiFi data. Finally, the SV calling steps is very fast and takes less than 8 min to run despite the computational load of POA and local alignment. Overall, the runtime of the SVDSS pipeline is less than 70 min for a high-coverage HiFi sample on 16 cores, excluding index creation time. In comparison, the fastest SV caller was cuteSV, taking 5 min, and the slowest was sniffles, taking upwards of 3 h. The remaining method debreak, pbsv and SVIM each took between 90 and 100 min to run.

All tools needed less than 64 GB of memory with SVDSS peaking at 34 GB of memory during the SV calling stage. Our SFS extraction and smoothing stages each use constant memory; however, the SV calling stage uses the most memory due to simultaneous handling of several (depending on the number of threads) POA graphs and local alignment dynamic programming tables in memory.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

All described datasets are publicly available through the corresponding repositories. In our experimental evaluation we used data publicly available at: GRCh38 reference genome: https://hgdownload.cse.ucsc.edu/goldenpath/hg38/bigZips/hg38.fa.gz; GRCh37 reference genome: http://ftp-trace.ncbi.nih.gov/1000genomes/ftp/technical/reference/phase2_reference_assembly_sequence/hs37d5.fa.gz; HG002 PacBio HiFi data: https://storage.googleapis.com/brain-genomics-public/research/deepconsensus/publication/deepconsensus_predictions/hg002_15kb/two_smrt_cells/HG002_15kb_222723_002822_2fl_DC_hifi_reads.fastq; HG002 assembly: https://console.cloud.google.com/storage/browser/brain-genomics-public/research/deepconsensus/publication/analysis/genome_assembly/hg002_15kb/two_smrt_cells/dc; CMRG callset: https://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/release/AshkenazimTrio/HG002_NA24385_son/CMRG_v1.00/GRCh38/StructuralVariant/; HG007 PacBio HiFi data: https://storage.googleapis.com/brain-genomics-public/research/deepconsensus/publication/deepconsensus_predictions/hg007_15kb/three_smrt_cells/HG007_230654_115437_2fl_DC_hifi_reads.fastq; HG007 assembly: https://console.cloud.google.com/storage/browser/brain-genomics-public/research/deepconsensus/publication/analysis/genome_assembly/hg007_15kb/two_smrt_cells/dc; CHM13 PacBio HiFi data: https://github.com/marbl/CHM13#hifi-data; CHM13 T2T assembly v1.1: https://s3-us-west-2.amazonaws.com/human-pangenomics/T2T/CHM13/assemblies/chm13.draft_v1.1.fasta.gz. The three callset built using dipcall are available at https://github.com/ldenti/SVDSS-experiments.

Code availability

SVDSS is open source and publicly available at https://github.com/Parsoa/SVDSS. Scripts to reproduce the experimental evaluations described in the manuscript are available at https://github.com/ldenti/SVDSS-experiments. Other software tools used in the study are either referenced or provided as links here: `pbmm2` (https://github.com/PacificBiosciences/pbmm2) and pbsv (https://github.com/PacificBiosciences/pbsv).

References

65. Li, H. Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly. *Bioinformatics* **28**, 1838–1844 (2012).
66. Edge, P. & Bansal, V. Longshot enables accurate variant calling in diploid genomes from single-molecule long read sequencing. *Nat. Commun.* **10**, 4660 (2019).
67. Gao, Y. et al. abPOA: an SIMD-based C library for fast partial order alignment using adaptive band. *Bioinformatics* **37**, 2209–2211 (2021).
68. Daily, J. parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments. *BMC Bioinformatics* **17**, 81 (2016).
69. Bonfield, J. K. et al. Htslib: C library for reading/writing high-throughput sequencing data. *Gigascience* **10**, giab007 (2021).

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grants agreements No. 872539 and 956229 (P.B. and R.C.). This work has also been supported in part by NSF award DBI-2042518 to F.H. R.C was supported by ANR Transipedia, SeqDigger, GenoPIM, Inception and PRAIRIE grants (ANR-18-CE45-0020, ANR-19-CE45-0008, ANR-21-CE46-0012, PIA/ANR16-CONV-0005, and ANR-19-P3IA-0001). This project has received funding from the European Union's Horizon Europe programme for research and innovation under grant agreement No. 101047160. The funding body had no role in the design of the study and collection, analysis, and interpretation of data and in writing the manuscript.

Author contributions

L.D. and P.K. devised and implemented the approach. L.D. and P.K. performed the experimental evaluation. P.B., F.H. and R.C. conceived the study, supervised and coordinated the work. All authors wrote, reviewed, edited and approved the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41592-022-01674-1.

Correspondence and requests for materials should be addressed to Paola Bonizzoni, Fereydoun Hormozdiari or Rayan Chikhi.

Peer review information *Nature Methods* thanks Andrew Carroll and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available. Primary Handling Editor: Lin Tang, in collaboration with the *Nature Methods* team.

nature research

Corresponding author(s): Paola Bonizzoni, Fereydoun Hormozdiari, Rayan Chikhi

Last updated by author(s): July 29, 2022

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☒	☐ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☒	☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☒	☐ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	The software dipcall (v0.3) has been used to produce the SVs callsets used in our experimental evaluation.
Data analysis	The manuscript describes the SVDSS tool, which is freely available at https://github.com/Parsoa/SVDSS . In the experimental evaluation, we used the following softwares, which are all openly available: - minimap2 (v2.22-r1101), https://github.com/lh3/minimap2 - pbmm2 (v1.7.0), https://github.com/PacificBiosciences/pbmm2 - ngmlr (v0.2.7), https://github.com/philres/ngmlr - pbsv (v2.6.2), https://github.com/PacificBiosciences/pbsv - cuteSV (v1.0.11), https://github.com/tjiangHIT/cuteSV - svim (v1.4.2), https://github.com/eldariont/svim - sniffles (v1.0.12), https://github.com/fritzsedlazeck/Sniffles - debreak (v1.0.2), https://github.com/Maggi-Chen/DeBreak - truvari (v3.0.1), https://github.com/ACEnglish/truvari Extensive description on on experiments reproducibility is available at https://github.com/Identi/SVDSS-experiments .

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

All described datasets are publicly available through the corresponding repositories. In our experimental evaluation we used data publicly available at:
- GRCh38 reference genome: https://hgdownload.cse.ucsc.edu/goldenpath/hg38/bigZips/hg38.fa.gz
- GRCh37 reference genome: ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/technical/reference/phase2_reference_assembly_sequence/hs37d5.fa.gz
- HG002 PacBio HiFi data: https://storage.googleapis.com/brain-genomics-public/research/deepconsensus/publication/deepconsensus_predictions/hg002_15kb/two_smrt_cells/HG002_15kb_222723_002822_2fl_DC_hifi_reads.fastq
- HG002 assembly: https://console.cloud.google.com/storage/browser/brain-genomics-public/research/deepconsensus/publication/analysis/genome_assembly/hg002_15kb/two_smrt_cells/dc
- CMRG callset: https://ftp-trace.ncbi.nlm.nih.gov/ReferenceSamples/giab/release/AshkenazimTrio/HG002_NA24385_son/CMRG_v1.00/GRCh38/StructuralVariant/
- HG007 PacBio HiFi data: https://storage.googleapis.com/brain-genomics-public/research/deepconsensus/publication/deepconsensus_predictions/hg007_15kb/three_smrt_cells/HG007_230654_115437_2fl_DC_hifi_reads.fastq
- HG007 assembly: https://console.cloud.google.com/storage/browser/brain-genomics-public/research/deepconsensus/publication/analysis/genome_assembly/hg007_15kb/two_smrt_cells/dc
- CHM13 PacBio HiFi data: https://github.com/marbl/CHM13#hifi-data
- CHM13 T2T assembly v1.1: https://s3-us-west-2.amazonaws.com/human-pangenomics/T2T/CHM13/assemblies/chm13.draft_v1.1.fasta.gz

The three callset produced in this work are available at https://github.com/ldenti/SVDSS-experiments .

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	The sample size is the one determined by the publicly available human long-read sequencing data from Genome In A Bottle (GIAB) consortium and the Telomere-to-Telomere (T2T) consortium for this study.
Data exclusions	No data were excluded. We have used publicly available human long-read sequencing data from Genome In A Bottle (GIAB) consortium and the Telomere-to-Telomere (T2T) consortium for this study.
Replication	Not applicable since this study describes a deterministic algorithms without statistical analysis. Moreover, this study does not involve wet lab experiments.
Randomization	Not applicable since this study have no experimental or control groups. Moreover, this study introduces a method and does not include biological hypothesis analysis.
Blinding	Not applicable since this study does not involve statistical analysis and data acquisition. The method introduced in this study and all softwares used in the experimental evaluation are deterministic and do not take advantages from knowning the origin of the input data.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Human research participants
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging