*Published online 31 January 2022*

*Nucleic Acids Research, 2022, Vol. 50, No. 9* ***e53***
*https://doi.org/10.1093/nar/gkac039*

# Dysgu: efficient structural variant calling using short or long reads

**Kez Cleal*** **and Duncan M. Baird***

Division of Cancer and Genetics, School of Medicine, Cardiff University, Heath Park, Cardiff CF14 4XN, UK

Received June 03, 2021; Revised December 20, 2021; Editorial Decision January 11, 2022; Accepted January 24, 2022

## ABSTRACT

**Structural variation (SV) plays a fundamental role in genome evolution and can underlie inherited or acquired diseases such as cancer. Long-read sequencing technologies have led to improvements in the characterization of structural variants (SVs), although paired-end sequencing offers better scalability. Here, we present dysgu, which calls SVs or indels using paired-end or long reads. Dysgu detects signals from alignment gaps, discordant and supplementary mappings, and generates consensus contigs, before classifying events using machine learning. Additional SVs are identified by remapping of anomalous sequences. Dysgu outperforms existing state-of-the-art tools using paired-end or long-reads, offering high sensitivity and precision whilst being among the fastest tools to run. We find that combining low coverage paired-end and long-reads is competitive in terms of performance with long-reads at higher coverage values.**

## INTRODUCTION

Analysis of structural variants (SVs) with whole genome or targeted enrichment sequencing is used in the clinic for diagnosing acquired or inherited genetic diseases (1) and for investigating mechanisms of genomic complexity in cancer and other pathologies (2–6). Sequencing using short paired-end reads (PE) is well established for genomic analysis due to mature workflows and low sequencing costs, although increasingly, long-read (LR) sequencing technologies are being utilized for these purposes. These LR sequencing platforms permit much longer read-lengths which can potentially lead to improvements in mapping to repetitive or complex regions of the reference genome, and advantages for detecting SVs. However, the better scalability of paired-end technologies, with further improvements in development (7), means that SV calling with shorter reads is likely to remain an area of interest.

SVs are usually defined as genomic rearrangement events over an arbitrary size of 50 bp, falling into categories such as deletions (DEL), insertions (INS), duplications (DUP), inversions (INV) or translocations (TRA) (1). SVs below this threshold are often termed indels, although these can sometimes result from more complex events such as duplication, inversion or translocation. These labels are useful in conceptualizing simple genome rearrangements in terms of the reference genome structure, although complex SVs occurring in the germline or during cancer progression, can complicate interpretation.

SVs can be detected in sequencing data using a variety of methods. For PE data, single alignments only span relatively small within-read SVs (indels) due to limited read-length, so information of SVs must be gleaned from assessing discordant mappings, changes in read-depth and the occurrence of split-reads which straddle breaksites (8). Recent methods also employ de novo assembly of SV-derived reads and further rounds of SV discovery through re-mapping of derived contigs to the reference genome (9,10). Alignment free methods are also possible, by analysing differences in k-mer content between a sample and reference (11). For LR sequences, SVs up to several kb can be detected within alignments due to the long read-lengths involved, and split-reads, changes in read depth and assembly of SV-reads can be utilized (8,12). Additionally, SVs can also be jointly discovered using LR and PE data by de novo assembly (13).

A large number of bioinformatics tools have been developed for detecting SVs using PE or LR data, although recent benchmarking studies highlight that existing algorithms are often limited in their ability to detect all classes and sizes of SVs, and there is still considerable room for improvement (14–16). The approach of quality filtering of putative SVs also differs widely between tools. In the simplest case variants are filtered based on the weight of evidence or number of supporting reads, although choosing suitable thresholds can be difficult and higher read-depths have also been associated with false positives (15). Statistical methods for quality scoring have been employed, for example the PE caller Manta employs Bayesian inference using read fragments supporting an allele to estimate a likelihood, followed by manual filtering (9). The LR caller nanovar utilizes

*To whom correspondence should be addressed. Tel: +44 2920 687855; Email: clealk@cardiff.ac.uk
Correspondence may also be addressed to Duncan M Baird. Email: bairddm@cardiff.ac.uk

a neural network classifier trained on simulated datasets, where 14 input features of each putative SV are used to classify events (17). To build on these advances, we considered that performance may be enhanced from training using non-simulated datasets. Additionally, we identified that there is an unmet need for an SV caller capable of analysing both PE and LR datasets.

Here, we present our SV calling software dysgu, which can rapidly call SVs from PE or LR data, across all size categories. Conceptually, dysgu identifies SVs from alignment cigar information as well as discordant and split-read mappings. Dysgu employs a fast consensus sequence algorithm, inspired by the positional de Brujin graph, followed by remapping of anomalous sequences to discover additional small SVs. A machine learning classifier is then employed to generate a useful quality score which can be used to prioritize variants.

## MATERIALS AND METHODS

### Overview

Dysgu has been designed to work with aligned reads in BAM or CRAM formats, and can analyse PE reads with lengths in the range 100–250 bp, or single-end LR such as PacBio Sequel II (HiFi), or Oxford Nanopore Technologies (ONT). By default, events with a minimum size of ≥30 bp are reported. Depending on the sequencing platform, dysgu offers pre-set options which apply recommended settings and a specific machine learning model (e.g. use '–mode pe' or '—mode pacbio' for PE or PacBio settings, respectively).

Dysgu provides a 'run' command which will produce a vcf file for a single input file, which is recommended for PE reads. However, depending on read-type the stages of the pipeline can differ. For PE reads (and optionally long reads), dysgu first partitions SV candidate reads into a temporary bam file (compression level set to zero by default), which is achieved using the 'fetch' command. Dysgu will then apply the 'call' command to SV candidate reads and produce an output. Depending on the length of input reads, the 'fetch' command may be redundant, as for very long reads such as ONT, a large proportion of reads harbour multiple SV candidates, which effectively leads to the input file being duplicated. Therefore the 'fetch' command is not needed for some LR datasets, and the 'call' command is recommended instead.

### Identifying SV candidate reads

For PE reads, library insert metrics are collected from the input file by scanning the first $200 \times 10^3$ reads. If the 'fetch' command is utilized, single reads, or all alignments from a read-pair, that are deemed to be candidates, are partitioned into a temporary file. However, if the 'fetch' command is not run, then input reads are simply marked as SV candidates. A read is defined as a candidate if a read is found with either, map-quality ≥1, a soft-clip ≥15 bp (PE only), a discordant insert size or read orientation (PE only), a supplementary mapping, an alignment gap ≥30, or a mate on another chromosome. A discordant insert size is defined as $insert\ size \geq insert\ median + (5.insert\ stdev)$. Reads in high coverage regions of the genome are also not analysed by default, defined as regions with a mean depth ≥200 ('–mode pe') or ≥150 ('—mode pacbio' or '—mode nanopore').

### Genome coverage

Dysgu collects several quality control metrics for use as features in the machine learning model. Genome coverage is calculated according to (18), except coverage is binned into 10 bp non-overlapping segments. The genome coverage tracks are saved in the temp folder during execution.

### Alignment clustering

Reads are initially clustered using an edge-coloured undirected graph $G$. Nodes in the graph represent SV-signatures and correspond to events listed in the cigar field of an alignment, or the properties of a read. SV-signatures are enumerated as either 'discordant', 'split', 'deletion', 'insertion' or 'breakend', and are associated with a 'genomic-start' and 'genomic-end' position. 'Breakend' types indicate a read that has a normal mapping orientation and no supplementary mappings, but has a soft-clipped sequence, which potentially corresponds to an unmapped breakpoint. Edges correspond to either 'white edges' that link together all alignments in a template with the same query name, or 'black' edges that are added between nodes that share a compatible SV signature.

Clustering is split into two phases. Initially, genomic reads are converted into a series of SV-signatures, with each item corresponding to a separate candidate event. For example, a deletion identified in the alignment cigar, a discordant read, or a read with an unmapped soft-clipped are converted into SV-signatures as nodes in $G$.

The local genomic region is then searched for events with a compatible signature. We use a red-black tree to search for items with a similar 'genomic end' position before checking if the 'genomic start' position is also similar. A search depth of 4 is used to search forwards and backwards in the data structure for other nodes. We find that using the 'genomic end' position permits a shallow search depth as datapoints are often sparser at the distant 'genomic end' position. Edges are not permitted between 'deletion' or 'insertion' types, although edges between other types are allowed.

When searching for other nodes to add 'black' edges between, nodes that are closer in the genome to the query are preferred, so if multiple candidates are found, edges are only formed between nodes passing a more stringent threshold. SV-signatures are checked to make sure that they have a reciprocal overlap of 0.1, and a separation distance between 'genomic start' and 'genomic end' positions below a clustering threshold. For PE reads, the clustering threshold is $lt; insert\ median + (5.insert\ stdev)\ bp$, while for PacBio the threshold is $lt; 35$ bp, and ONT $lt; 100$ bp. If another SV-signature is found with a 'genomic start' $lt; 35$ bp, these nodes pass the more stringent threshold, and a 'black' edge is added to the graph. For single-end reads or 'split' reads, if any of these conditions fail we also check the span position distance (19) between signatures. Span position distance between signatures $S_1$ and $S_2$ is defined as $SPD = SD(S_1 + S_2) + \frac{PD(S_1 + S_2)}{N}$ where SD is the span distance between signatures

$SD = \frac{|(E_1 - B_1) - (E_2 - B_2)|}{\max(E_1 - B_1,\ E_2 - B_2)}$, and PD is the position distance $\min(|B_1 - B_2|,\ |E_1 - E_2|,\ |\frac{B_1 + E_1}{2} - \frac{B_2 + E_2}{2}|)$. N is a normalization constant which is set at 100 for PE reads, 600 for PacBio and 900 for ONT reads. For all read types the SPD threshold used is $tlt; 0.3$. For PE reads that do not have a 'split' SV signature, we use a modified formula, only adding 'black' edges between nodes if $\frac{PD}{\max(E_1 - B_1,\ E_2 - B_2)} lt; t$ and $SDlt; t$.

If no edges are found for a PE read, a second phase of clustering is used to try and find edges between reads that share similar soft-clipped sequences. As pairwise sequence comparison between neighbouring alignments is computationally costly, we devised a novel algorithm based on clustering of the minimizer sketch of soft-clipped reads (20). Minimizer sampling involves computing the list of minimum kmers derived from consecutive windows over a sequence. We use a kmer length of 6 and a window length of 12. The minimum kmer is selected using a hash function and computed in linear-time O($n$) (see: https://people.cs.uct.ac.za/~ksmith/articles/sliding_window_minimum.html). Additionally, in a modification of the minimizer sketching algorithm, we compute only the unique set of minimum kmers $S_k$ for each soft-clipped portion of a read. Each kmer in the set $S_k$ is associated with a genomic position that corresponds to the left-most or right-most base in the alignment for left or right soft-clipped sequences, respectively.

Kmers are added to a hashmap $M$ with the key given by the kmer hash, and the value pair corresponding to a set of tuples, of (genomic position, read name). Kmers that are >150 bp from the query genomic position are dynamically removed from the hashmap during processing.

For each incoming read, the kmer set $S_k$ is first computed, then for each kmer a corresponding set $Z$ of reads and genomic positions is obtained by indexing $M$. The set $Z$ consists of a collection of local reads that share the same minimizer kmer as the query. Entries in $Z$ are then compared to the current genomic position and if the separation is $< 7$ bp, the number of found minimizers $a$ is incremented. Additionally, the number of minimizers shared between reads with the same name $b$ is counted. The total minimizer support is defined as $(\frac{a}{2} + b)$ and a threshold of $\geq 2$ is utilized. Once the minimizer support threshold is exceeded, found nodes are added to a set and returned.

Finally, 'black' edges are added to the graph between the returned set of nodes and the query node. Utilizing the minimizer clustering algorithm, pairwise sequence alignment is avoided, instead sequence matches between two sequences can be inferred from computing a minimizer sketch and utilizing hashmap queries.

### Event partitioning

Once all alignments have been added into the main graph $G$, the graph is simplified to a undirected quotient graph $Q = (V_q,\ E_q)$ whose vertices consists of blocks or partitions of vertices from the main graph $G$. The vertices (partitions) $V_q$ are found by finding connected components in $G$ using 'black' edges only. Edges $E_q$ are then defined between partitions using 'white' edge information from $G$, thus linking together read templates that map one or more SV.

Connected components in $Q$ are processed together. These components can be composed of one or more partitions, harbouring potentially multiple SV events. In the simplest case, a component will consist of a single partition, which is processed for one or more SV. Components with a single edge are processed for a single SV only. For components with multiple edges, each edge is processed for a single SV, and additionally, each node partition is processed as a single partition if the number of 'black' intra-partition edges exceeds the number of 'white' out-edges, according to the main graph $G$. Thus, all components of $Q$ are processed as a series of single-edges or single-partitions.

Single-edges in $Q$ are assumed to represent a single SV, with reads from the $u$ partition corresponding to one breaksite and reads from the $v$ partition corresponding to the other. Single-partition nodes are assumed to map a single SV if a spanning alignment is found (e.g. a deletion event in the alignment cigar field). If no-spanning alignments are found, reads in the single-partition are further clustered using hierarchical clustering with the Nearest Point Algorithm (21), using the genomic start and end points of reads in the partition. This step helps disentangle SVs with large overlaps and similar reference coordinates. Identified subclusters are then processed for a single SV.

### Consensus sequence generation

We generate consensus sequences at each breakpoint, from which read properties can be derived, such as repeat score or expanded polymer bases (see SV metrics section for further details), and to determine soft-clipped sequences for potentially remapping to the reference genome. We utilize a novel algorithm that borrows concepts from the positional de Brujin graph (22), and partial order alignment graphs (POA) (23). In a positional de Brujin graph $G$, the vertex set $V$encodes each sequence kmer in addition to genomic location, which helps leverage information provided by the mapper and localizes assembly. Edges $E$ are permitted between kmers adjacent in the reference genome, which generally leads to a directed acyclic graph. However, it is possible that some bases do not have a genomic location, such as insertions within a read, or soft-clipped sequence. In such cases, genomic location can be inferred, for example using the expected mapping position if the whole read was aligned without gaps (10).

Partial order alignment graphs (23) are used to perform multiple sequence alignments, with vertices representing bases, and edges added between neighbouring bases in a sequence. Additional Sequences can be pairwise-aligned and incorporated into a POA using dynamic programming, and a consensus can be extracted by back-tracing through the maximum weighted path (23).

In our algorithm, we also represent vertices as bases and employ back-tracing through the longest path. However, similar to a positional de Brujin graph, we take the ordering of the graph from the genomic locations determined by the mapper. Utilizing this approach gives an approximation of a multiple sequence alignment between local genomic reads,

and makes usage of information given by the mapper, whilst being simple and efficient to compute.

Let vertices correspond to a tuple $(b_i,\ i,\ f,\ c) \in V$, where $b_i$ is the base aligned at genome position $i$, $i$ is the genome position, $f$ is an offset describing the distance to the closest aligned base, and $c$ is a flag to indicate if the base is part of a left or right soft-clip (or neither). For left soft-clipped bases $c = 1$, right soft-clipped bases $c = 2$, whilst $c = 0$ otherwise. Bases that are not aligned to the reference genome may thus belong to three categories, when $f > 0$, for insertions $c = 0$, for left soft-clips $c = 1$, and for right soft-clips $c = 2$.

Edges are added between adjacent bases in a sequence $(u_j, v_{j+1})$, and vertices are weighted according to the sum of base qualities for a given node. Graph construction leads to a directed acyclic graph, that is then topologically sorted in linear time (24).

To read the consensus sequence, the graph is first traversed using breadth-first search and for each vertex $v$, the longest path ending at $v$ is determined by choosing the highest scoring predecessor vertex and adding to the running total. The consensus sequence is read by back-tracing from the vertex with the highest score, and recursively selecting the best predecessor node.

The worst-case time complexity for consensus sequence generation is linear with the number of input sequence bases. This follows, as graph construction, topological sorting, breadth-first search and back-tracing all have worst case complexities of $O(V + E)$ time.

### Consensus sequence quality trimming

For the described consensus sequence algorithm, problems can arise at unmapped bases (e.g. soft-clipped sequences) if the underlying reads have a high indel error rate. In this situation, indels in unaligned bases cause neighbouring sequences to be shifted out of sync and can result in collapsing of indel errors in the consensus sequence. To address this problem, we trim soft-clipped sequences at bases with an alternative high scoring path. For each node $v$ on the consensus path, with predecessor $u$ and successor $w$ also on the consensus path, a path quality metric is calculated. $I_{total}$ is defined as the total weight of all incoming edges to $v$. The in-edge quality is defined as $q_{in} = \frac{I_{(u,\ v)}}{I_{total}}$, where $I_{(u,\ v)}$ is the weight of the consensus path edge $(u,\ v)$. Similarly, $O_{total}$ is defined as the total weight of all outgoing edges from $v$. The out-edge quality is defined as $q_{out} = \frac{O_{(v,\ w)}}{O_{total}}$, where $O_{(v,\ w)}$ is the weight of $(v,\ w)$. The path quality metric for $v$ is defined as $P_q = \min(q_{in},\ q_{out})$. Soft-clipped sequences are trimmed at bases with a path quality metric $< 0.5$.

The soft clip weight (scw) parameter is defined for subsequent filtering, as the total base quality of nodes in the soft-clipped portion of the sequence divided by the length of the soft-clip.

### Re-mapping of contigs

After generating consensus sequences, if an end co-ordinate could not be determined, an attempt is made to align the soft-clipped sequence to the reference genome. Soft-clipped sequences are remapped to a window ±500 bp from the anchored breakpoint. We utilize edlib (25) (parameters: mode = 'HW') to find an approximate location, before refining the alignment using Striped Smith-Watermen (26) (parameters: match_score = 2, mismatch_score = −8, gap_open_penalty = 6, gap_extend_penalty = 1) using the scikit-bio library (found online at: http://scikit-bio.org/). For deletion events, if less than 40% of the soft-clip could be remapped and the alignment span is < 50 bp, the alignment is rejected. For insertion events, if >20 bp of sequence could not be mapped the alignment is rejected.

If no alignment is identified, dysgu can still call an unanchored insertion event at the identified break point, however, only events that have support > min_support +4 and a soft-clip length ≥18 bp. The min_support parameter can be user supplied and takes a value of 3 for PE data or 2 for LR data.

### Sequence repeat score

Dysgu calculates repetitiveness scores for aligned regions of contigs as well as reference bases between deletions, and soft-clipped sequences. To calculate this metric, the sequence of interest is broken into kmers of increasing lengths from 2 – 6 bases. For each kmer of length $k$, a hashtable is used to record the last seen position of each kmer. If a kmer is seen more than once, the distance in bases to the last seen position is retrieved $d$. The repeat score is then calculated as a mean according to $\frac{1}{n}\ (\sum \frac{kx}{m})$ where $k$ is the kmer length, and $x$ and $m$ have the form $v.e^{-\frac{\lambda}{k}}$, where $e$ is Euler's number, $\lambda$ is a decay constant set at 0.25, and $v = k$ for the denominator $m$, and $v = d$ for $x$. For perfect tandem repeats $\frac{kx}{m} = 1$, whilst sequencing errors, interspersed patterns or random sequence lead to lower values.

### Base quality score correlation at soft-clipped reads

For short-read input data we calculate a metric referred to as 'soft-clip quality correlation' (SQC), which is aimed at quantifying a sequence-specific error profile we observed in Illumina data (27). During sequencing, it is though that certain genomic sequences can promote dephasing, that gives rise to read base-qualities that correlate with the underlying sequence, and can result in frequent mismatches in alignments at specific bases (27). In our data, we observed a pattern consistent with this model but occurring at soft-clipped reads. These sites were frequently identified adjacent to homopolymer sequences and displayed base-quality scores that fluctuated with the underlying soft-clipped sequence. These soft-clip sequences often appeared to contain many errors as neighbouring soft-clipped reads showed many differences. Finally, these sites also frequently gave rise to false-positive calls at one-end anchored SV calls. The SQC metric was devised to quantify this phenomenon and is utilized as a feature in machine learning classification.

For each query read from the putative SV, the quality values of soft-clipped bases are added to a hashmap $H$, with the relative genomic position $pos$ as the key, and a list $L_{pos}$ of base-qualities as values. The relative genomic position is taken as the position of the base if the whole soft-clipped portion of the read was mapped to the genome.

Once all reads have been added, the 'local mean' is calculated as the absolute difference from the mean of each list $d_{pos} = |x_j - \mu|$ where $x_j$ is each item in $L_{pos}$ and $\mu$ is the mean of $L_{pos}$. The sum of all calculated values of $d_{pos}$ is stored in a variable $v_{local} = \sum d_{pos}$, and the global mean across all $d_{pos}$ is calculated $m = \frac{v_{local}}{n}$. Finally, for each list in $H$, the sum of differences with the global mean is calculated $v_{global} = \sum |x_j - m|$. The SQC metric is calculated as the ratio $sqc = \frac{v_{local}}{v_{global}}$. When the positions of low-quality bases are distributed randomly with genomic position $sqc$ values will be close to 1.0. However, when low quality bases are clustered at certain positions, this results in smaller differences in base qualities at the local scale, giving smaller $v_{local}$ values and lower $sqc$ values.

### Fold change in coverage across SVs

We calculate the fold change in coverage (FCC) across putative SVs according to (28) with minor modifications. We utilize a genomic bin size of 10 bp and analyse 1 kb sequence flanking the left and right breaksites. The fold change in coverage is calculated as the median coverage of the interior SV region divided by the median of the flanking sequence. The FCC metric was the most important feature after SV length for classifying SVs by machine learning, however we considered that this metric may not be suitable for non-diploid samples, or complex clonal mixtures such as those encountered during tumour sequencing, as lower allelic fractions only give rise to small changes in FCC. For this reason, we also provide an additional machine-learning model for use with non-diploid or complex tumour SV discovery.

### Polymer repeats at breaksites

Dysgu searches for simple repeat patterns with a unit length of 1–6 bp that directly overlap a break. These sites could arise from the joining of directed repeats (e.g. deletion event) or by the extension of the polymer at the break (e.g. insertion), or perhaps a more complex event. The length of the identified repeat sequence and the stride of the simple repeat are also utilized as features in the machine learning model.

For each base in the input sequence, a search is initiated for a repeat pattern starting at that base. Repeat lengths $l$ of between 1–6 bp are tested in increasing length. To identify a repeat pattern, successive kmers are tested for identity with the starting kmer, using a step size of $l$. If a matching kmer is found the count $c$ is incremented. If >3 non-matching kmers or >1 successive non-matching kmer is found the search is stopped. If $c \geq 3$ when the search is stopped, and the spanning sequence identified is > 10 bp, the repeat sequence is set aside. Finally, if the repeat sequence overlaps the breaksite then the SV event is annotated with the breaksite repeat and stride length.

### SV event metrics

Dysgu annotates each putative SV event with a number of metrics. In Table 1, we list metrics utilized in the diploid paired-end model by decreasing feature importance.

### Classifier training

To train a machine learning classifier for the different read-types (PE, PacBio and ONT) we constructed several 'gold-sets'. Gold-sets consisted of manually curated SV loci or SV loci found using other calling software. Primarily, gold-sets were based on the well-studied HG001 sample (Female, Western European ancestry). However, for PacBio data, gold-sets were also derived from the HG005 sample (Male, Chinese ancestry). The read data utilized in constructing the gold-sets are listed in Table 2.

The overall strategy was to quantify dysgu performance on smaller subsets of data, and then combine these smaller benchmarks into a larger set for training. We employed this strategy as it meant that manual curation of smaller subsets was more feasible (as opposed to annotating events genome wide), and also multiple methods for annotating true-positive calls could be integrated into the training set, e.g. relying on manual curation, labelling using a third party SV caller, or utilizing previously publish call sets, or utilizing different DNA mappers.

Firstly, we constructed a gold-set based on PacBio Sequel II reads. Nanovar was run on HG001 minimap2-aligned reads and insertion calls from chr1 and chr10 in the size range 30–500 bp were added to the set ($n = 1808$). The choice of chromosome to utilize was arbitrary. We also utilized a previously published list of deletion and insertion calls made using pbsv ($n = 27\,662$) on PacBio CCS data at around 30× coverage (downloaded from GIAB ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/ChineseTrio/analysis/PacBio_pbsv_05212019/HG005_GRCh38.pbsv.vcf.gz).

Next we added a collection of manually curated SV loci that were identified by visually inspecting calls made by dysgu using the Integrative Genomics Viewer (IGV) (29). Multiple read-types were assessed, simultaneously viewing alignments of PacBio Sequel II, PacBio CCS and ONT reads. If the SV showed support in more than one technology the SV loci was labelled as true. If a call made by dysgu was plausible, but showed strong evidence of being below the minimum size threshold < 30 bp, then the call was labelled as false. All deletion and insertion calls for chr1, 10 and 11 for HG001 minimap2-aligned reads were manually labelled in this way (n = 2973). Additionally, large insertion calls ('large-INS') made by dysgu (≥500 bp, whole genome) using HG001 minimap2 and ngmlr aligned reads were also assessed ($n = 1661$). Calls made by dysgu were then compared to these smaller benchmark sets separately and labelled as true or false using SVBench (available online at https://github.com/kcleal/svbench).

These smaller benchmarks were then concatenated before training a gradient boosting classifier using the lightgbm package (30) (boosting type 'dart'). Features were first selected using recursive feature-selection with cross-validation using scikit-learn (31). Hyperparameters were tuned using grid search with cross-validation using Stratified $K$-fold ($n = 5$) (31). The learning-rate, max-bin, max-depth, $n$-estimators and number-of-leaves were optimized in this way, whilst other parameters were left as default.

Events labelled using the PacBio classifier with probability ≥0.5 were then leveraged to help construct additional

**Table 1.** Overview of the features used in machine learning classification

| Abbreviation | Long name | Description |
|---|---|---|
| **SVLEN** | SV length | The length in base-pairs of the SV |
| **FCC** | Fold change in coverage | A measure of the change in sequencing coverage across the SV |
| **SQR** | Soft-clip base quality ratio | The quality ratio of soft clipped bases to aligned bases |
| **SU** | Support | The total evidence in terms of reads supporting the SV |
| **RMS** | Re-mapping score | The alignment score of the re-mapped soft-clipped sequence for one-end anchored SVs |
| **CMP** | Compressibility | The mean compressibility of both consensus sequences, defined as the compressed sequence length divided by the length of the uncompressed sequence. Zlib is used as the sequence compressor. |
| **BCC** | Bad clip count | The number of reads within 500 bp of breaksites that do not have a high quality soft-clip. A sliding window of 10 bp is used to scan soft-clip sequences. If the average base quality of the window is $>10$, a counter is incremented. If $\geq 15$ windows are found above this threshold, the read is deemed to have a high quality soft-clip. |
| **NEIGH10** | Neighbours within 10 kb | The total number of neighbouring break points within 10 kb of each end of the SV. |
| **REPSC** | Repeat score for soft-clipped sequences | The mean repeat score for the soft-clipped portion of consensus contigs. See the 'Repeat score calculation' section for details. |
| **MCOV** | Maximum sequence coverage within 10 kb | The maximum sequencing coverage within 10 kb of SV breaksites |
| **SWC** | Soft-clip weight | The average base quality weight of the soft-clipped portion of consensus contigs. See the 'Consensus sequence generation' section for more details. |
| **RB** | Reference bases | The total number of reference-aligned bases in consensus sequences |
| **RAS** | Reverse soft-clip to alignment score | The soft-clipped portion of a consensus contig is reverse complemented and aligned to the reference-aligned portion of the contig. RAS is the score of any alignment found using Striped Smith-Waterman using scikit-bio. |
| **MAPQP** | Map quality primary | The mean mapping score of primary alignments. |
| **RR** | Reference repeat score | For deletion events $<150$ bp, the repeat score for the deleted reference sequence is calculated. See the 'Repeat score calculation' section for details. |
| **COV** | Mean coverage within 10 kb | The mean sequencing coverage within 10 kb of both break sites. |
| **FAS** | Forward soft-clip to alignment score | The soft-clipped portion of a consensus contig is aligned to the reference-aligned portion of the contig. FAS is the score of any alignment found using Striped Smith-Waterman using scikit-bio. |
| **SQC** | Soft-clip quality correlation | See the section 'Base quality score correlation at soft-clipped reads' |
| **SVTYPE** | Structural variant type | The major SV category, DEL – deletion, INS – insertion, INV – inversion, DUP – duplication, TRA – translocation. |
| **NP** | Normal pairs | The total number of reads with a 'normal' mapping orientation and spacing determined by the mapper |
| **GC** | GC % | The mean GQ percentage of consensus contigs |
| **NEXP** | Number of expanded repeat bases at break | See the 'Repeat expansion at break sites' section |
| **REP** | Repeat score of aligned bases | The mean repeat-score of reference-aligned sections of consensus contigs. See the 'Repeat score calculation' section for details. |
| **NMP** | Mean NM score or alignments | Mean edit-distance of primary alignments supporting the variant, determined by the mapper |
| **BND** | Number of break-end reads | The total number of reads with a breakend signature, arising when a PE read is mapped in a normal orientation with no supplementary mappings, but also has a soft-clipped sequence |
| **MAS** | Maximum alignment score | Maximum alignment score of supplementary reads supporting the variant |
| **STRIDE** | - | The unit size in bp of the polymer extension sequence at the break site |
| **MS** | Minus strand | The total number of reads found on the minus strand |
| **NMB** | - | Mean edit distance excluding gaps $\geq 30$ bp |
| **OL** | Overlap | The overlap in bp of query alignments from each breaksite |
| **RED** | Re-map edit distance | The edit distance of the re-mapped soft-clip sequence |
| **PS** | Plus strand | The total number of reads found on the plus strand |
| **NEIGH** | Neighbours | The number of other putative breakpoints within 1 bp of the current SV |
| **WR** | Within-read support | The number of reads with an alignment gap supporting the SV |
| **RPOLY** | Reference polymer | Number of polymer bases identified in the reference-aligned portion of consensus contigs |
| **CIPOS95** | Confidence-interval | The confidence-interval around the POS breaksite |
| **MAPQS** | Map-quality supplementary | The mean mapping quality of supplementary alignments |
| **SC** | Soft-clips | Number of reads with soft-clips supporting the variant |
| **SR** | Split-reads | Number of split-reads supporting the variant |
| **BE** | Block edge | Categorical variable indicating if the component of the quotient graph from which the call was made, had an edge |
| **NDC** | Number of double clips | The number of reads that had left and right soft-clips |
| **STL** | Short template length | The number of reads that displayed an insert size blow the 0.05% percentile. |

**Table 2.** Overview of datasets used in model training

| Sample | Read type | Alignment information | Coverage | Source |
|---|---|---|---|---|
| HG001 | PacBio Sequel II 11 kb library | GRCh37 minimap2 GRCh37 ngmlr | 5–6 | SRA accession SRR9001772 |
| HG001 | ONT | GRCh37 minimap2 | 13 | SRA accession SRR10965087 |
| HG001 | Illumina 148 bp x2 HiSeq 2500 | GRCh37 bwa mem | 40 20 | ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/NA12878/NIST_NA12878_HG001_HiSeq_300x/RMNISTHS_30xdownsample.bam |
| HG001 | PacBio CCS | GRCh37 minimap2 | 24 | ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/NA12878/NA12878_PacBio_MtSinai/merged_ec_output_primary.bam |
| HG005 | PacBio Sequel II 11 kb library | GRCh38 minimap2 | 5–6 | SRA accession SRR9001776 |

gold-sets for PE and ONT read-types. For the PE gold-set, deletion and insertion loci identified using the PacBio model were taken as true-positive loci (chromosomes 1, 2, 10, 11, 12, $n = 8258$). Additionally, the 'large-INS' set derived from PacBio reads was utilized. Finally, events called by dysgu using PE reads (HG001, bwa mem) were manually curated, corresponding to deletions ($n = 5984$ true) from chromosomes 1–5 and 10–22, plus insertions ($n = 2250$ true) from chromosomes 1–14. The choices of chromosomes were arbitrary.

For the ONT gold-set, we utilized deletion and insertion loci identified using the PacBio model (probability $\geq 0.5$, whole genome $n = 25\,072$ true). To this, we used regions identified by Nanovar ($n = 23\,581$ true), and the 'large-INS' manually curated set. Additionally, we added manually curated dysgu calls from ONT data from chr1 and chr10 ($n = 4265$).

### Benchmark datasets

For the HG002 benchmark, variants were downloaded from GIAB ftp://ftptrace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/analysis/NIST_SVs_Integration_v0.6. For HG001, variants were downloaded from GIAB ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/technical/svclassify_Manuscript/Supplementary_Information/Personalis_1000_Genomes_deduplicated_deletions.bed.

The 'syndip' CHM1-CHM13 benchmarking analysis files were downloaded from https://github.com/lh3/CHM-eval/releases/download/v0.5/CHM-evalkit-20180222.tar. Short read alignment files for this sample were downloaded from the Sequence Read Archive (SRA) accession ERR1341796. PacBio (HiFi) data for CHM13 was downloaded from SRA from accession SRR11292120, and PacBio data for CHM1 was downloaded from SRA accession SRR14407676. One million PacBio reads from each of CHM13 and CHM1 were combined for analysis of the 'syndip' benchmark.

Data from the 1000 Genomes project was obtained from ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/1000G_2504_high_coverage/data, and reference call sets were downloaded from ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/phase3/integrated_sv_map/ALL.wgs.integrated_sv_map_v2_GRCh38.20130502.svs.genotypes.vcf.gz.

### Structural variant simulation

SVs were simulated using Visor (32). Two haplotypes with random SVs were created from the base human reference genome hg38. For each haplotype the command used was 'randomregion.r -n 20000 -l 50 –standarddev 1000 -v 'deletion,insertion,tandem duplication,inversion' -r '25:25:25:25 -d chrom.dim.tsv -x exclude.bed'. The chrom.dim.tsv and exclude.bed file listed chromosome lengths and centromere/telomere regions. Single nucleotide mutations were randomly added to each haplotype using a custom script with a rate of ∼1 mutation per 2 kb. 150 bp paired end reads ($n = 200 \times 10^6$) were sampled from each haplotype using wgsim for a coverage of ∼38×. Simulated reads were mapped to the base hg38 reference genome using bwa mem (33). SV callers were run using default parameters.

### Benchmarking SV calls using svbench

We developed a python software library 'svbench' to facilitate rapid benchmarking of SV datasets, as well as to facilitate exploration and comparison of SV calls as an aide during software development. Svbench performs a similar role to other benchmarking programs such as truvari from GIAB (34) (Supplemental_Table_S1.pdf), although as data structures can be held in memory and explored interactively, significant speedups can be obtained for benchmarking which can be helpful during software development and analysis.

Svbench also optionally adds a weighting to input SVs that can be used to break ties between multiple query and reference SVs. The weighting or 'strata' can be specified during loading of SVs, and usually takes the value of a quality metric set by the caller, or if this is absent, the variant support in terms of read evidence. Stratifying SV calls in this way is also necessary to generate a precision-recall curve.

Another difference between svbench and truvari, is that svbench can optionally classify duplicate true-positive calls, which can arise when one reference SV in the sample gives rise to multiple calls in the output. There are several ways to classify duplicates, such as labelling all duplicates as false-positives, true-positives, or ignoring them from precision calculation. By default, svbench utilizes the latter option. Although this can lead to optimistic precision and F1 scores, we consider this approach often leads to a clearer un-

derstanding of the underlying performance of an SV caller. For example, if duplicates are labelled as false-positives then a caller that identifies the correct genomic loci but has a high duplication rate is penalized, while a caller that identified incorrect loci but also has a low duplication rate could end up with a similar overall precision and F1 score. Furthermore, removing duplicates bioinformatically, might be less of a challenge than removing genuine false positives, by for example filtering SVs with low weight but found nearby other SVs.

Conceptually, svbench loads input files (vcf, bed, bedpe or csv format) into a 'CallSet' object. Internally, SV records are held in a pandas dataframe (35), which support a rich set of data wrangling capabilities, making common data operations straightforward such as filtering, splitting, combining, grouping and plotting precision-recall curves.

To compare one dataset with another, i.e. a benchmark dataset with a query dataset, both sets of SV loci are loaded into an svbench CallSet object. The benchmark dataset is then prepared by adding intervals (add_intervals function) around each breaksite, adding one interval for each start and end coordinate. Intervals are held in a nested containment list using the ncls library (36). Utilizing an interval at both start and end sites, rather than a single interval, means translocations can be naturally compared, and for large SVs, nesting of small SV intervals within larger SVs is avoided which can reduce the search space when comparing records.

Query SVs are then checked against prepared intervals. If a benchmark record overlaps both the start and end of a query SV, and the percent size similarity, reciprocal overlap and svtype match criteria, then the records are considered to match. Percent size is defined as $\frac{\min(size_{ref},\ size_{query})}{\max(size_{ref},\ size_{query})}$. Query and benchmark records that pass provided thresholds are then clustered on an undirected graph $G$, using the network library (37,38).

Edges $(u,\ v) \in G$ are added to the graph between benchmark vertices $u$ and query vertices $v$ with the edge weight given by the 'strata', or weight property of the query event, which is parsed during loading of the data. If a query vertex $v$ matches multiple benchmark vertices $u$, then the chosen benchmark call $u$ is determined by the closest absolute genomic distance between $u$ and $v$, defined as $|start_{query} - start_{ref}| + |end_{query} - end_{ref}|$. Once all query records have been added to the graph, connected components are then processed. If a benchmark vertex has multiple edges, a highest scoring edge is selected as the true-positive call, whilst other query vertices are labelled as duplicates. If duplicate classification is permitted then precision scores are calculated as $precision = \frac{true\ positives}{total-duplicates}$. If duplicate classification is turned off then duplicates are treated as false positives. Recall is assessed as $recall = \frac{true\ positives}{true\ positives + false\ negatives}$ and F1 score is calculated as $F1 = 2.\frac{precision.\ recall}{precision + recall}$.

We utilized svbench to assess performance of dysgu compared to other SV callers. For benchmarking calls against the HG002 benchmark (34), we filtered query calls by a minimum size of 30 bp (whole genome benchmark), or 50 bp (Tier 1 benchmark). We utilized a reference interval size of 1000 bp, and a percent size similarity threshold of 15%. Deletion and insertion calls were analysed separately, filtering both query and reference calls by svtype before comparison. Additionally, only query calls on the 'normal' chromosomes were analysed $\{chr1..chr\ Y\}$. To match the definition of the GIAB benchmark, we converted DUP calls <500 bp to insertions.

SV callers were applied to datasets using default settings. Version numbers for tested callers were as follows: dysgu v1.3.0, gatk v4.1.2.0, strelka v2.9.2, manta v1.6.0, svim v2.0.0, sniffles v1.0.12, nanovar v1.4.0, delly v0.8.5. Versions number of mappers were: minimap2 v2.17, ngmlr v0.2.7, bwa v0.7.17. SV calls were also filtered by removing calls without a 'PASS' in the filter field (if applicable). The 'strata' metric utilized for each of the SV callers was as follows: lumpy – 'SU', delly – 'QUAL', dysgu – 'PROB', manta – 'QUAL', strelka – 'QUAL', gatk – 'QUAL', nanovar – 'QUAL', sniffles – 'RE', svim – 'SUPPORT'. Events with a minimum support <2 were filtered out.

## RESULTS

Dysgu is a general purpose*de novo* SV and indel caller that can analyse PE or LR sequencing datasets (Figure 1). SV-associated reads are first identified by assessing alignment gaps, split-read and discordant mappings, soft-clipped reads and read-depth changes. SV signals are clustered on a graph and contigs are generated for putative breakpoints (Figure 1B–E). One-end anchored SVs—events with a single soft-clipped sequence without a corresponding mapping, are re-aligned to the reference genome to identify additional small SVs. Putative SV events are labelled with a rich set of features describing sequencing or mapping error metrics and supporting evidence. Events are further classified using a machine learning model to prioritise variants with higher probability (Figure 1F). Dysgu also supports merging of SVs across datasets and read types, regenotyping of samples, and working with capture data.

### Testing datasets

To assess precision and recall statistics we utilized benchmark datasets provided by the Genome in a Bottle (GIAB) consortium. Primarily, we assesses a germline call set derived from the Ashkenazi son sample (HG002) that combines five sequencing technologies and 68 call sets plus manual curation into a high quality and comprehensive benchmark (34). The HG002 benchmark is stratified into high confidence regions (Tier 1), where precision and recall can be confidently determined, as well as less confident regions (Tier 2, followed by 'all' regions) which potentially involve more complex genomic regions, or the completeness of the benchmark is uncertain. However, as only SVs ≥50 bp appear in Tier 1 regions, we also analysed all unfiltered SVs in the GIAB dataset which has a minimum SV size threshold ≥20 bp, appreciating that the 'All-regions' benchmark shows lower completeness compared to Tier 1 regions.

In addition, we assessed recall on the HG001 cell line that has corresponding deletion calls (≥50 bp) provided by GIAB (39). As the machine-learning classifier that dysgu

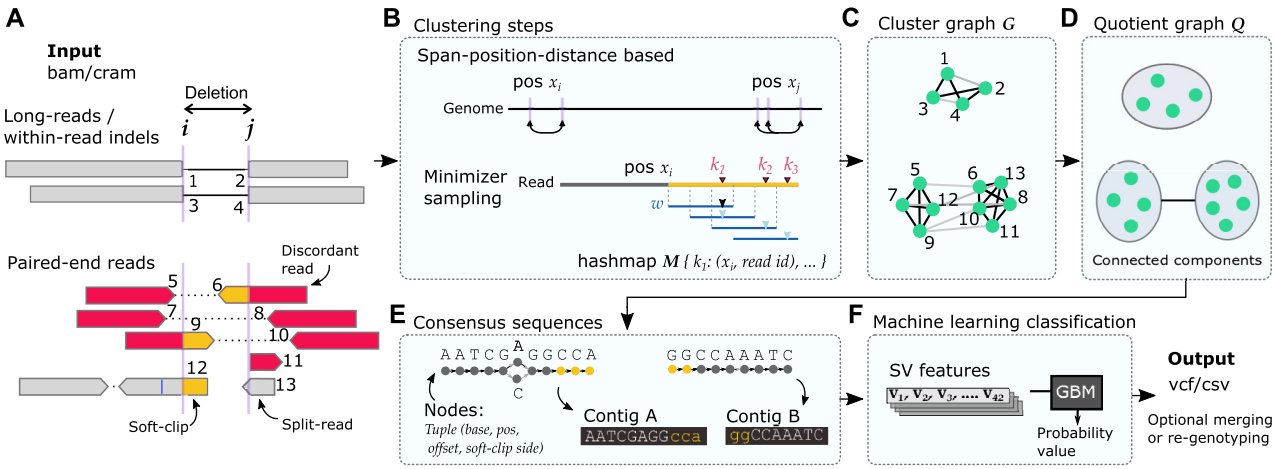


**Figure 1.** Overview of dysgu pipeline. Dysgu analyses long-read (top pane, **A**) or paired-end data (bottom pane, **A**), assessing alignment gaps, discordant and split-reads, and soft-clipped alignments. An example deletion is shown for the different read types, with numbers representing SV signatures assessed by dysgu. Candidate SV signatures are clustered by their proximity on the reference genome (span-position-distance based) and by using a minimizer sketch of the soft-clipped portion of reads (**B**). SV evidence is clustered on a graph $G$, using black edges to link matching signatures from B, and grey edges to link alignments from the same read (**C**). The top and bottom panes in (C) and (**D**) depict example graphs from analysing LR or PE data, respectively. The graph $G$ is simplified to a quotient graph $Q$, where nodes represent partitions of $G$ that are linked by black edges, and edges in $Q$ are induced by grey edges between partitions. Connected components in $Q$ are then further analysed. A fast consensus sequence algorithm is employed to generate break-site sequences (**E**), which may be re-aligned to the reference genome to discover additional small SVs. Finally, up to 42 features are calculated for each candidate SV, depending on read type and scored using a gradient boosting machine classifier (**F**). Output vcf files may then be optionally merged/unified with other samples or read types.

employs was trained using calls derived from HG001 (see Materials and Methods), we did not assess precision using this dataset. For PE data we also assessed SV callers on the 'syndip' benchmark (40), and a subset of 10 samples from the 1000 Genomes project (41).

### Performance using paired-end short reads

Dysgu was tested on simulated data (Supplemental_Fig_S1.pdf), and on the HG002 benchmark at coverages of 20× (Figure 2, Table 3, 2, Supplemental_Table_S2.pdf, Supplemental_Table_S3.pdf) and 40× (Supplemental_Fig_S2.pdf, Supplemental_Table_S4.pdf - Supplemental_Table_S6.pdf). Performance was compared to the popular SV callers manta (9), delly (42), and lumpy (43). We also compared indel calling performance with strelka (44) and gatk down to a size of 30 bp. Strelka calls indels up to 50 bp whilst gatk calls deletions and insertions to around the insert size.

On simulated data, all SV callers performed well across most SV classes except for novel sequence insertions (Supplemental_Fig_S1.pdf). Dysgu was the only caller tested that could reliably call this class of SV, calling 4483 true positives, whereas the next best caller delly called only 218 true positives. Dysgu showed the highest $F1$ scores for deletions and novel sequence insertions, whereas lumpy showed the highest F1 scores for duplications and inversions.

To gauge performance on real data, Tier 1 SVs of the HG002 benchmark (20× coverage) were assessed. Dysgu called the largest number of true deletions and insertions ($n = 3913$), with 727 more variants called than the next best caller manta ($n = 3186$) (Table 3). Precision-recall curves indicated that probability values estimated by dysgu using machine learning were useful for stratifying variants by quality, with higher probability values correlating with precision (Figure 3A–D). Dysgu had the highest precision for deletion calls (95.7%), as well as the highest recall for deletions (62.2%) and insertions (23.7%). Manta showed the highest precision for insertion variants (97.6% versus dysgu 95.1%) but had a lower recall (14.2%) than dysgu. As a percentage value, dysgu called 8.8% more deletions and 66% more insertions than manta. Overall, dysgu showed higher $F1$ scores than the next best caller, manta, with an $F1$ score 4.7% higher for deletions and 13.1% higher for insertions. Dysgu also showed better genotype calling $F1$ scores compared to manta, although this was due to higher recall with manta showing the highest genotype calling precision (Table S2).

We also assessed the level of duplication, defined as the ratio of duplicated true-positive calls relative to unique true-positive calls. The problem of duplication arises when a single SV event leads to multiple calls in the output file. Generally, all PE callers displayed a low level of duplication below <1.5% (Table 3).

We also stratified variants by size using the All-regions benchmark to investigate size constraints of SV calling (Table 4, Supplemental_Table_S3.pdf). For deletions in the 30–50 bp range, dysgu showed similar performance to gatk with similar precision, recall and $F1$ scores. For insertions in the 30–50 bp range, dysgu showed higher precision (97.1%) and recall (28.2%) than strelka and gatk.

For SVs ≥50 bp, dysgu showed a good balance of precision and recall across all size ranges with the highest $F1$ scores among callers (Table 4). For deletion SVs dysgu generally displayed the highest precision but showed a lower recall for large SVs. For example, delly showed a higher recall than dysgu for deletions ≥5000 bp (41.1% versus 33.1%), but only had a precision of 36% versus dysgu 97%.

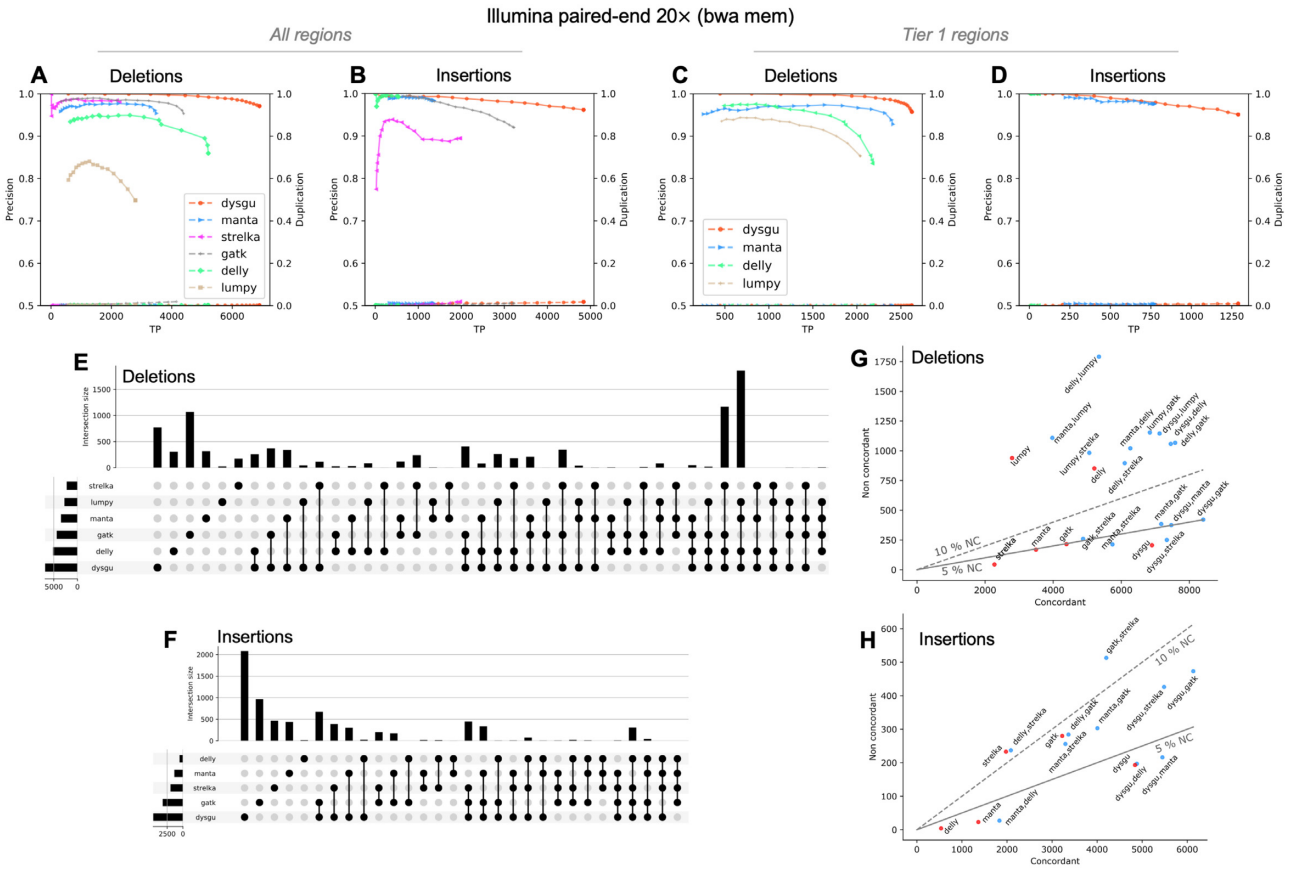


**Figure 2.** Performance of dysgu using 20 × PE reads. Dysgu was compared to SV callers manta, delly and lumpy, and indel callers strelka and gatk, using the HG002 benchmark. Precision-recall curves are shown for all genomic regions (**A, B**), as well as high-confidence Tier 1 regions (**C, D**). The secondary y-axis indicates duplicate true-positives (TP) as a fraction of true-positive calls. Intersections and aggregates of intersections of SV calls for the all-regions benchmark are displayed using an upset plot (**E, F**). To investigate combinations of SV callers, the union of true-positives between callers (labelled concordant), was plotted against the sum of false-positives (labelled non concordant) (**G, H**). The 5 and 10% non-concordance (NC) is also illustrated as a solid or dashed line, respectively.

**Table 3.** Performance using PE 20× data on the HG002 'Tier 1 regions' benchmark. The numbers of deletion (DEL) and insertion (INS) variants are quantified. Duplication is defined as the ratio of duplicate true-positive calls to the number of true-positive calls. TP = true-positive, FP = false-positive. Best scores are shaded blue

| | TP | | FP | | Precision | | Recall | | Duplication | | *F*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEL | INS | DEL | INS | DEL | INS | DEL | INS | DEL | INS | DEL | INS |
| dysgu | 2624 | 1289 | 117 | 66 | 0.957 | 0.951 | 0.622 | 0.237 | 0.000 | 0.009 | 0.754 | 0.379 |
| manta | 2411 | 775 | 163 | 12 | 0.937 | 0.985 | 0.572 | 0.142 | 0.000 | 0.008 | 0.710 | 0.249 |
| delly | 2178 | 58 | 429 | 0 | 0.835 | 1.000 | 0.517 | 0.011 | 0.001 | 0.000 | 0.638 | 0.021 |
| lumpy | 2037 | | 256 | | 0.888 | | 0.483 | | 0.001 | | 0.626 | |

For insertion SVs ≥50 bp, dysgu showed the highest recall, but manta displayed the best precision of 98.2%. Dysgu was the best caller for identifying loci with large insertions (≥500 bp) finding $n = 426$, versus manta $n = 23$ and gatk $n = 52$. However, as dysgu utilizes insert size statistics to estimate large insertions length, calculated insertion sizes are expected to be less accurate compared to *de novo* assembly-based callers such as manta and gatk (data not shown).

At 40× coverage, all callers displayed improved recall and *F*1 scores although at the expense of lower precision (Supplemental_Fig_S2.pdf, Supplemental_Table_S4.pdf - Supplemental_Table_S6.pdf). Interestingly, this phenomenon was also reported in a recent benchmarking study suggesting that at higher coverage values, absolute numbers of sequencing and mapping artifacts are more likely to be mistaken for SV events with low allelic fraction (14). Overall, at 40× coverage dysgu maintained a good balance of precision and recall compared to other callers, in line with 20 × coverage, showing the highest *F*1 score for deletions and insertion calls.

We next investigated the intersection of variant calls between tools, or the set of SVs shared between tools, and displayed results using an upset plot (Figure 2E, F), which quantifies the sizes of SV call sets, their intersections, and aggregates of intersections (45). Assessing Tier 1 SVs in the HG002 benchmark, dysgu showed the largest number of

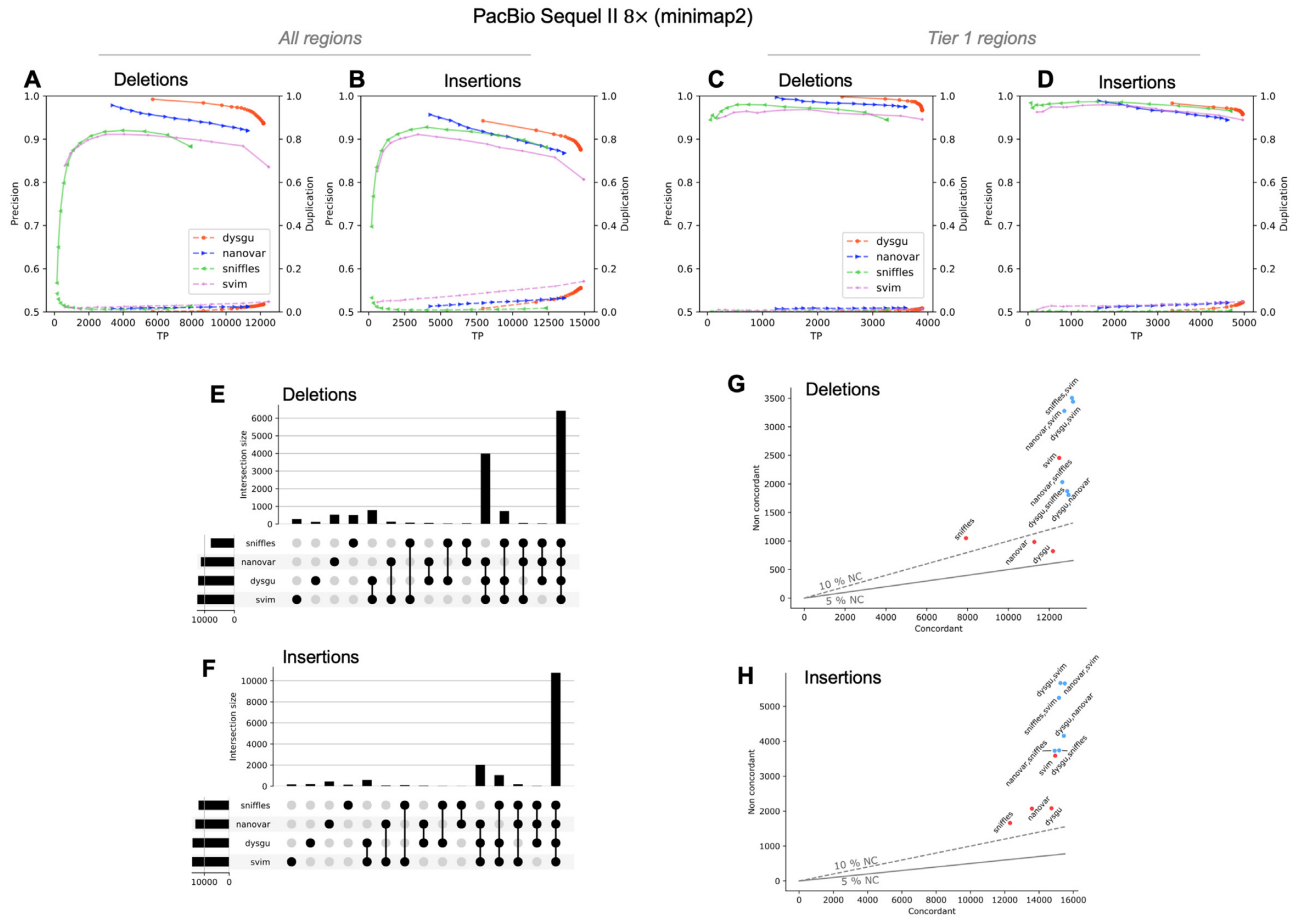


**Figure 3.** Performance of dysgu using PacBio reads. Precision-recall curves are shown for all genomic regions (**A, B**), as well as high-confidence Tier 1 regions (**C, D**). Analysis of SV intersections and aggregates of intersections for the all-regions benchmark are displayed using an upset plot (**E, F**). Combinations of SV callers were assessed by plotting the union of true-positives (labelled concordant), against the sum of false-positives (labelled non concordant) (**G, H**). The 5 and 10% non-concordance (NC) are shown as a solid or dashed line, respectively.

**Table 4.** SV calling stratified by size using PE 20× data on the HG002 the 'All-regions' benchmark. Best scores are shaded blue

| | | Precision | | | | Recall | | | | F1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | [30–50) | [50–500) | [500–5000) | ≥5000 | [30–50) | [50–500) | [500–5000) | ≥5000 | [30–50) | [50–500) | [500–5000) | ≥5000 |
| Deletions | dysgu | 0.969 | 0.969 | 0.979 | 0.970 | 0.358 | 0.232 | 0.370 | 0.331 | 0.522 | 0.374 | 0.537 | 0.494 |
| | manta | 1.000 | 0.967 | 0.953 | 0.837 | 0.008 | 0.219 | 0.286 | 0.335 | 0.015 | 0.358 | 0.441 | 0.479 |
| | gatk | 0.966 | 0.932 | 1.000 | | 0.361 | 0.105 | 0.001 | | 0.526 | 0.189 | 0.002 | |
| | strelka | 0.982 | 1.000 | | | 0.262 | 0.003 | | | 0.413 | 0.005 | | |
| | delly | 0.974 | 0.906 | 0.766 | 0.360 | 0.242 | 0.164 | 0.377 | 0.411 | 0.388 | 0.277 | 0.505 | 0.384 |
| | lumpy | 0.895 | 0.930 | 0.745 | 0.311 | 0.002 | 0.148 | 0.378 | 0.409 | 0.004 | 0.255 | 0.502 | 0.353 |
| Insertions | dysgu | 0.971 | 0.935 | 0.988 | 1.000 | 0.282 | 0.153 | 0.119 | 0.119 | 0.437 | 0.263 | 0.213 | 0.212 |
| | manta | 1.000 | 0.988 | 1.000 | | 0.012 | 0.100 | 0.007 | | 0.023 | 0.182 | 0.014 | |
| | gatk | 0.923 | 0.910 | 1.000 | 1.000 | 0.250 | 0.101 | 0.014 | 0.028 | 0.393 | 0.182 | 0.027 | 0.054 |
| | strelka | 0.888 | 0.938 | 1.000 | | 0.225 | 0.006 | 0.003 | | 0.358 | 0.013 | 0.005 | |
| | delly | 0.991 | 1.000 | | | 0.057 | 0.006 | | | 0.108 | 0.012 | | |

unique calls (both deletions $n = 154$, and insertions $n = 814$) followed by manta ($n = 127$ deletions, $n = 290$ insertions). Including indel callers and analysing all SVs changed the conclusion slightly. In this case, gatk found the most unique deletions events ($n = 1084$, versus dysgu $n = 766$) and the second highest number of unique insertion events ($n = 944$) after dysgu ($n = 2034$).

Recent studies have investigated combining the output of different SV callers to boost performance (46–48). To gauge the performance of different combinations of callers we assessed the union of true positive calls (labelled as concordant) and compare with the sum of false positives (labelled non-concordant) as a proxy for the false positive rate (Figure 2G, H). The best combination of callers using the All-

regions benchmark appeared to be dysgu and gatk which together found 8408 deletions and 6134 insertions.

We additionally tested the recall of tools against the HG001 deletion call set, comparing unfiltered variants for all callers. Dysgu demonstrated the highest recall (93.61%), followed by manta (89.84%), delly (84.38%) and lumpy (81.61%).

Additionally, dysgu was tested on the synthetic diploid ('syndip') benchmark (40). This benchmark was previously created by de novo assembly of two near haploid cell lines (CHM1 and CHM13). SV callers were tested against deletion and insertion reference calls and performance was stratified by size (Supplemental_Fig_S3.pdf, Supplemental_Table_S7.pdf – Table_S8.pdf). Performance of SV callers were consistent with the HG002 benchmark, with dysgu showing the highest *F*1 scores. Dysgu called about twice as many true variants as the next most sensitive caller delly ($n$ = 12 229 for dysgu versus 6122 for delly), although manta had the highest average precision (0.892 versus dysgu 0.883). For deletion calls ≥50 bp manta also showed good performance, with *F*1 scores close to dysgu (Supplemental_Table_S8.pdf).

Finally, we also tested dysgu on ten randomly chosen samples from the 1000 genomes project for which a reference set of large deletions (≥1000 bp in size) was available (Supplemental_Fig_S4.pdf, Supplemental_Table_S9.pdf). Dysgu demonstrated the highest precision and *F*1 scores on every sample, although delly and lumpy showed better recall than dysgu, with delly having an average recall of 52.3% versus dysgu 48.9%. However, delly only had a mean precision of 23.7% vs 63.3% for dysgu. The next best caller after dysgu was manta, with an *F*1 score on average 3.2% lower than dysgu.

To summarise, using PE data, dysgu was generally the most performant tool showing a good balance of precision and recall across SV types and size ranges.

### Performance using long reads

We tested dysgu against the HG002 benchmark using PacBio HiFi reads at approximately 8× (Figure 3, Tables 5–6, Supplemental_Fig_S5.pdf, Supplemental_Table_S10.pdf - Supplemental_Table_S13.pdf) and 15× coverage (Supplemental_Fig_S6.pdf, Supplemental_Table_S14.pdf - Supplemental_Table_19.pdf), and using Oxford nanopore reads at 13× coverage (Supplemental_Fig_S7.pdf, Supplemental_Table_S20.pdf - Supplemental_Table_S27.pdf). Performance was compared against recently published LR callers nanovar (17), sniffles (49) and svim (19), using reads aligned by minimap2 (50) (Figure 3, Tables 5 and 6), or ngmlr (49) (Supplemental_Fig_S5.pdf, Supplemental_Table_S10.pdf). Aligning reads using ngmlr tended to give slightly higher precision among all SV callers although F1 scores were also slightly reduced, particularly for insertion variants (Supplemental_Table_S10.pdf).

Assessing Tier 1 SVs from the HG002 benchmark, dysgu had the highest recall for deletions (92.4%) and insertions (91.2%) and the highest precision for insertion calls (95.8%). Dysgu also had the highest *F*1 score for deletions (0.945) and insertions (0.934) but was closely followed by svim with *F*1 scores of 0.935 and 0.928 for deletions and insertions, respectively (Figure 3 and Table 5).

Expanding the testing set to all regions and a minimum size of 30 bp, svim showed the highest recall (0.334 for deletions and 0.409 for insertions) (Supplemental_Table_S11.pdf). Dysgu and nanovar displayed similar precision scores, but overall dysgu displayed the highest *F*1 scores (0.483 for deletions and 0.551 for insertions) (Supplemental_Table_S11.pdf). Svim showed marginally lower *F*1 scores (0.477 for deletions and 0.542 for insertions), although we noticed that svim showed a higher level of duplication. Additionally, for some callers this problem was more acute when analysing Oxford nanopore reads, with for example, svim showing a duplication ratio of 0.58 for insertion calls in Tier 1 regions (Supplemental_Figure_S8.pdf, Supplemental_Table_S22.pdf). Among callers, dysgu generally showed a higher level of duplication than sniffles and nanovar, although dysgu had a consistently higher recall.

Analysing the intersection of SVs, we found that most callers seemed to identify similar sets of SVs indicating that combining SV callers might only lead to small gains in sensitivity (Figure 2E–H).

Similar to Illumina data, increasing the coverage of PacBio HiFi data increased the recall of SV callers and *F*1 scores, but at the expense of reduced precision. At 15× coverage, dysgu had the highest F1 scores for deletions and insertions for Tier 1, whilst showing a low level of duplication (Supplemental_Table_S14.pdf).

Sensitivity of SV detection was also assessed using the HG001 deletion benchmark (≥50 bp in size). Using PacBio reads at 5× coverage dysgu showed the highest recall (77.35%) compared to other callers (nanovar 75.97, sniffles 70.52, svim 73.73%). Likewise, dysgu showed the highest recall using 13× ONT reads (96.41%) compared to other callers (nanovar 91.67, sniffles 95.89, svim 95.25%).

We further assessed dysgu on the syndip benchmark by combining PacBio reads from CHM1 and CHM13 samples to a coverage of around 12× (Supplemental_Fig_S7.pdf, Supplemental_Table_S20.pdf – Supplemental_Table_S21.pdf). Results were consistent with the HG002 benchmark, with dysgu generally showing higher precision and F1 scores compared to other callers, and svim demonstrating a slight advantage for recall.

In summary, dysgu demonstrated a high level of performance on LR datasets, with generally the best balance of precision and recall across SV sizes and categories.

### Combining short and long reads for improved performance

Dysgu supports merging of SVs from different runs using a 'merge' command making it trivial to integrate calls from different sequencing technologies. After merging, additional tags are added to the output file corresponding to the maximum and mean probability across samples, with the probability determined by the machine learning classifier.

We used dysgu to assess different combinations of sequencing technology including PacBio (8× and 15×), ONT (13×) and Illumina paired-end reads (20 × and 40×), by filtering calls with a maximum model probability ≥0.5 for PacBio, or ≥0.35 for ONT combinations (Table 7). Test-

**Table 5.** Performance using PacBio Sequel II reads at 8 × coverage on HG002 Tier 1 regions. Duplication is defined as the ratio of duplicate true-positive calls to the number of true-positive calls. TP = true-positive, FP = false-positive. Best scores are shaded blue

| | TP | | FP | | Precision | | Recall | | Duplication | | *F*1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DEL | INS | DEL | INS | DEL | INS | DEL | INS | DEL | INS | DEL | INS |
| dysgu | 3896 | 4962 | 132 | 218 | 0.967 | 0.958 | 0.924 | 0.912 | 0.014 | 0.040 | 0.945 | 0.934 |
| nanovar | 3593 | 4613 | 93 | 272 | 0.975 | 0.944 | 0.852 | 0.848 | 0.018 | 0.042 | 0.909 | 0.893 |
| svim | 3899 | 4965 | 224 | 292 | 0.946 | 0.945 | 0.925 | 0.912 | 0.014 | 0.048 | 0.935 | 0.928 |
| sniffles | 3251 | 4680 | 190 | 166 | 0.945 | 0.966 | 0.771 | 0.860 | 0.011 | 0.006 | 0.849 | 0.910 |

**Table 6.** Long-read performance as a function of SV size. PacBio Sequel II reads at 8× coverage was assessed using the HG002 'all-regions' benchmark. Best scores are shaded blue

| | | Precision | | | | Recall | | | | *F*1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | [30, 50) | [50, 500) | [500, 5000) | ≥5000 | [30, 50) | [50, 500) | [500, 5000) | ≥5000 | [30, 50) | [50, 500) | [500, 5000) | ≥5000 |
| Deletions | dysgu | 0.938 | 0.938 | 0.900 | 0.921 | 0.549 | 0.501 | 0.467 | 0.360 | 0.693 | 0.654 | 0.615 | 0.518 |
| | nanovar | 0.930 | 0.924 | 0.830 | 0.800 | 0.489 | 0.457 | 0.443 | 0.327 | 0.641 | 0.611 | 0.578 | 0.464 |
| | svim | 0.881 | 0.806 | 0.767 | 0.856 | 0.560 | 0.512 | 0.487 | 0.339 | 0.685 | 0.626 | 0.596 | 0.486 |
| | sniffles | 0.930 | 0.903 | 0.759 | 0.567 | 0.261 | 0.361 | 0.439 | 0.354 | 0.408 | 0.516 | 0.557 | 0.436 |
| Insertions | dysgu | 0.834 | 0.866 | 0.941 | 0.929 | 0.580 | 0.600 | 0.561 | 0.360 | 0.685 | 0.709 | 0.703 | 0.519 |
| | nanovar | 0.845 | 0.871 | 0.838 | 0.558 | 0.515 | 0.552 | 0.529 | 0.304 | 0.640 | 0.676 | 0.649 | 0.394 |
| | svim | 0.782 | 0.773 | 0.890 | 0.929 | 0.589 | 0.608 | 0.573 | 0.364 | 0.672 | 0.681 | 0.697 | 0.523 |
| | sniffles | 0.862 | 0.874 | 0.892 | 0.868 | 0.452 | 0.541 | 0.469 | 0.182 | 0.593 | 0.668 | 0.615 | 0.301 |

ing against the All-regions benchmark, the addition of Illumina reads consistently led to performance improvements when combined with PacBio or ONT, especially for deletion calls (Table 7). The largest increases in recall were seen from adding 40× Illumina calls, although 20× Illumina calls also led to noticeable increases. For example, adding 40× Illumina calls to 8× PacBio calls identified an additional 1286 deletions and 894 insertions for the All-regions benchmark, or 150 deletions and 33 insertions for Tier 1 regions. *F*1 scores improved for the All-regions benchmark, increasing by 3.4% for deletions and 2.1% for insertions. Surprisingly, combining Illumina calls with PacBio 8×, appeared to be similar in performance to PacBio calls at a higher coverage value 15×.

However, Tier 1 regions generally did not show increased *F*1 scores despite increased recall, which was caused by an inflation of the false-positive rate (Supplemental_Table_S28.pdf). Additionally, we assessed Tier 1 + 2 regions which include more complicated genomic loci than Tier 1. Tier 1 + 2 regions also showed improved *F*1 scores, with 8 × PacBio + 40× Illumina *F*1 scores increasing by 3.1 points for deletions and 1.5 for insertions (Supplemental_Table_S29.pdf). We speculate that Illumina data may enhance SV calling at complicated genomic regions that are not trivial to map for LR mappers. Additionally, PE data may help fill-in the gaps for LR datasets in regions of low or zero coverage.

To understand how PE datasets might increase the performance of SV calling when combined with LR datasets, we assessed true-positive deletion calls made using PE data that were absent from LR data and vice-versa (Supplemental_Fig_S9.pdf, Supplemental_Fig_S10.pdf). Around 72% of PE calls not found in LR call sets appeared to result from low or absent read support in the LR data. We estimated only around 3.5% of PE calls would be detectable in the in LR data by altering default runtime parameters of dysgu, for example by lowing the map quality threshold (Supplemental_Fig_S9.pdf). However, a further 24% of PE calls showed ambiguous alignments in the LR dataset that were inconsistent with the reference SV. A few examples are shown in Supplemental_Fig_S10.pdf, showing single deletion events that are mapped as multiple gaps on the reference. Using blat (51) to align the LR data sometimes recovered the expected reference SV event, suggesting that using a different aligner might help resolve mapping ambiguities.

These data indicate that PE data can improve SV calling in combination with LR data mainly through evening-out coverage of the genome, but also by providing an alternative interpretation at ambiguous or complex SV sites. Combining sequencing technologies for improved SV discovery has not received much attention, although with the increasing prevalence of LR sequencing, and other non-standard techniques such as linked-read or HiC, we suggest that this would be an interesting avenue for future research.

## Runtime

We tested runtime using an Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz Linux machine with 256GB of system memory. For Ilumina data, dysgu was the fastest tool using a single-core, analysing 40× coverage data in 57 min and using 5.6GB memory (Table 8), which was more than twice as quick as the next fastest tool, delly. Manta was over 6 times slower than dysgu to run on a single core, but used the least memory (0.244), and can also be run in parallel efficiently (data not shown). However, dysgu trades speed for temporary disk space by copying SV-associated reads to an intermediate file for fast access and required 7.8GB disk space using default parameters. As large intermediate files can cause I/O problems in high performance computing environments, dysgu can be run without generating intermediate alignment files although runtime increased to

**Table 7.** Performance of combinations of sequencing platforms using the HG002 'all-regions' benchmark. pb = PacBio, ill = Illumina, ont = Oxford Nanopore Technologies. Best scores are shaded blue

| | TP | | Precision | | Recall | | Duplication | | *F*1 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | DEL | INS | DEL | INS | DEL | INS | DEL | INS | DEL | INS |
| pb 8x | 12155 | 14690 | 0.936 | 0.878 | 0.325 | 0.402 | 0.029 | 0.097 | 0.483 | 0.551 |
| pb 8x + ill 20x | 12999 | 15226 | 0.929 | 0.873 | 0.348 | 0.416 | 0.051 | 0.118 | 0.506 | 0.564 |
| pb 8x + ill 40x | 13441 | 15584 | 0.920 | 0.868 | 0.360 | 0.426 | 0.064 | 0.131 | 0.517 | 0.572 |
| pb 15x | 12821 | 15599 | 0.939 | 0.868 | 0.343 | 0.427 | 0.029 | 0.111 | 0.502 | 0.572 |
| pb 15x + ill 20x | 13399 | 15956 | 0.930 | 0.863 | 0.358 | 0.436 | 0.052 | 0.133 | 0.518 | 0.580 |
| pb 15x + ill 40x | 13749 | 16189 | 0.922 | 0.857 | 0.368 | 0.443 | 0.066 | 0.147 | 0.526 | 0.584 |
| ont 13x | 13716 | 13531 | 0.891 | 0.880 | 0.367 | 0.370 | 0.041 | 0.028 | 0.520 | 0.521 |
| ont 13x ill 20x | 13261 | 14456 | 0.918 | 0.869 | 0.355 | 0.395 | 0.056 | 0.117 | 0.512 | 0.543 |
| ont 13x ill 40x | 13928 | 15031 | 0.902 | 0.855 | 0.373 | 0.411 | 0.072 | 0.158 | 0.527 | 0.555 |
| ont 13x pb 8x | 13758 | 15616 | 0.892 | 0.841 | 0.368 | 0.427 | 0.095 | 0.220 | 0.521 | 0.567 |

**Table 8.** Resource requirements of SV callers. Best scores are shaded blue. Temporary hard-disk space requirements are only shown for dysgu. Dysgu can be run in two different modes – the 'run' mode first separates SV-associated reads into a temporary file for faster access before running the 'call' program. For very long read data such as ONT, the 'call' program is recommended

| Reads | Caller | Min | Mem (GB) | Space (GB) |
|---|---|---|---|---|
| Illumina 40X | dysgu 'run' | 57.3 | 5.54 | 7.8 |
| | dysgu 'call' | 97.8 | 5.39 | 0.6 |
| | Manta | 365.1 | 0.24 | |
| | delly | 150.0 | 6.42 | |
| | lumpy | 211.5 | 12.00 | |
| PacBio 8X | dysgu 'run' | 7.5 | 0.35 | 4.3 |
| | dysgu 'call' | 10.9 | 1.12 | 0.6 |
| | nanovar | 46.5 | 17.15 | |
| | svim | 15.25 | 0.34 | |
| | sniffles | 19.5 | 0.71 | |
| ONT 13X | dysgu 'call' | 68.0 | 0.94 | 0.6 |
| | nanovar | 83.4 | 17.58 | |
| | svim | 66.0 | 0.90 | |
| | sniffles | 68.0 | 2.01 | |

around 98 min for this sample. We tested dysgu on additional samples with different coverage values (Supplemental_Table_S30.pdf). Resource requirements generally scaled with coverage, although sample complexity appeared to play a role.

For PacBio HiFi reads analysed on a single core, dysgu was the fastest tool, analysing 8× coverage sample in <8 min and using 0.35GB memory, but used 4.3GB of temporary space. ONT reads at 13× coverage were analysed by dysgu in 68 min using 0.94GB memory, which was slower than the fastest caller svim (66 min and 0.9GB memory).

## DISCUSSION

We developed dysgu to facilitate SV and indel discovery using PE or LR sequencing platforms in a computationally efficient manner. Dysgu analyses several forms of evidence to detect events including alignment gaps, discordant reads, read-depth, soft-clipped and supplementary mappings. For PE data, remapping of anomalous soft-clipped reads is also utilized to identify additional small SVs. Putative events are then labelled with a useful probability value using a gradient boosting classifier (30).

Stratifying events by probability has several potential benefits over manually filtering. For example, machine learning classifiers can learn non-linear relationships between variables, and potentially capture large numbers of interactions between variables that would be difficult to reproduce through a manual approach. However, machine-learning raises additional challenges such as feature engineering, collation of appropriate training sets, and assessing how well a model will generalize to new data.

Dysgu models SV events using a vector of up to 41 features depending on read-type, with each feature designed to quantify different aspects of an SV signature, or error patterns of the respective read-type. The current list of features is non-exhaustive and can potentially be expanded in future releases to enhance calling performance.

Features incorporate more obvious signals such as read-support and sequencing depth, as well as novel patterns such as 'soft-clip quality correlation' (PE data only) and repetitiveness scores (see Materials and Methods). To facilitate the calculation of features that capture sequence-contextual information, we also developed a novel linear-time consensus sequence algorithm, that is used to rapidly collapse reads at each break site into consensus contigs for further analysis. We trained our classifier using a large collection of manually labelled SV loci and combined these sites with loci identified by other SV callers. Manually labelling induces an obvious bias in the training set, where the correctness is a matter of opinion of the human observer. However, using a manual approach also allowed us to generate training sets with high completeness, which was not the case when relying on third party SV callers. Construction of quality training sets is a perennial challenge in machine learning and we expect that improving the quality and size of training sets will yield further performance improvements for SV classification.

We validated performance using benchmark datasets provided by GIAB (34,39), and provide a software library 'svbench' to facilitate benchmarking and exploration of results. Primarily we assessed the HG002 benchmark, analysing in detail high-confidence Tier 1 regions, as well as all genomic regions. At Tier 1 regions we find that dysgu outperforms existing tools for both PE reads (Table 3) or third generation long-reads (Table 5) using the F1 metric for comparisons. Tier 1 regions cover 2.51 Gbps of the genome although more complicated regions and smaller indel SVs (<50 bp) are absent. Analysis of all genomic regions largely supported the conclusion that dysgu matches or outperforms existing tools, with dysgu often showing the best *F*1

scores across read types (Supplemental_Table_S4.pdf, Supplemental_Table_S11.pdf). Notably, svim showed higher *F*1 scores than dysgu in some benchmarks, although this was at the expense of considerably lower precision values and often increased duplication of true-positives.

Another novel feature of dysgu is that calls from separate sequencing technologies can be merged using a single command. Particularly, we found that adding calls made using Illumina data to either PacBio or ONT led to improved recall (Table 7). However, this appeared to occur mainly outside Tier 1 regions, suggesting Tier 1 regions are an 'easy-case' for LR platforms. Nevertheless, for applications that require higher recall, adding PE data to lower coverage LR data is a cost-effective approach for SV discovery that dysgu can support.

To improve dysgu performance in future releases there are several avenues to explore. Dysgu relies on a several heuristics such as span position distance, minimizer support and others that could potentially be optimized using a large test set. As discussed, feature engineering and improving training sets for machine learning are also likely to be beneficial. Finally, incorporation of a *de novo* assembly stage could also help better resolve multi allelic sites and full-length insertion sequences in PE data, and help resolve duplication of true-positve events. For LR datasets, dysgu relies heavily on the DNA mapper to generate meaningful alignments, although at complex SV sites problems can arise with ambiguous alignments (Supplemental_Fig_S10.pdf). *De novo* assembly, or re-alignment procedures may additionally help resolve these issues.

In conclusion, dysgu is *de novo* SV caller that performs favourably compared to existing tools using PE or LR datasets. Dysgu is also computationally efficient to run, being the fasted tool using PE data, or second fastest using LR data. We provide dysgu as an open-source package for use in basic and applied research applications.

## ABBREVIATIONS

SV structural variant, PE paired-end, LR long-read, DEL deletion, DUP duplication, INV inversion, INS insertion, TRA translocation, ONT Oxford Nanopore Technologies, GIAB Genome In A Bottle consortium, SRA Sequencing read Archive, POA partial order alignment.

## DATA AVAILABILITY

Dysgu is released as free and open source under the Massachusetts Institute of Technology (MIT) licence. Source code and distributions can be downloaded at https://github.com/kcleal/dysgu. Data used to train the classifier is available online at https://zenodo.org/record/4761527. Svbench is also released under the MIT license and can be found at https://github.com/kcleal/svbench. Analysis scripts used to reproduce results found in this paper can be found under https://github.com/kcleal/svbench. Illumina sequencing data for Ashkenazim HG002 (34) sample was downloaded from GIAB (ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/HG002_NA24385_son/NIST_HiSeq_HG002_Homogeneity-10953946/HG002Run01-11419412/HG002run1_S1.bam). Two lanes of PacBio data were downloaded from SRA (https://www.ncbi.nlm.nih.gov/sra) under accessions SRR10188368 and SRR10188369. ONT data were downloaded from SRA under accession SRR11537600.

## SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

## ACKNOWLEDGEMENTS

*Author contributions*: K.C. devised methodology, performed experiments, wrote the software and drafted the manuscript. D.M.B. contributed design ideas, provided feedback and performed manuscript editing. All authors read and approved the final manuscript.

## FUNDING

Work in the Baird laboratory is funded by Cancer Research UK [A18246/A29202]; Wales Cancer Research Centre. Funding for open access charge: Cancer Research UK.
*Conflict of interest statement.* None declared.

## REFERENCES

1. Stankiewicz,P. and Lupski,J.R. (2010) Structural variation in the human genome and its role in disease. *Annu. Rev. Med.*, **61**, 437–455.
2. Cleal,K. and Baird,D.M. (2020) Catastrophic endgames: emerging mechanisms of telomere-driven genomic instability. *Trends Genet.*, **36**, 347–359.
3. Cleal,K., Jones,R.E., Grimstead,J.W., Hendrickson,E.A. and Baird,D.M. (2019) Chromothripsis during telomere crisis is independent of NHEJ, and consistent with a replicative origin. *Genome Res.*, **29**, 737–749.
4. Escudero,L., Cleal,K., Ashelford,K., Fegan,C., Pepper,C., Liddiard,K. and Baird,D.M. (2019) Telomere fusions associate with coding sequence and copy number alterations in CLL. *Leukemia*, **33**, 2093–2097.
5. Turro,E., Astle,W.J., Megy,K., Gräf,S., Greene,D., Shamardina,O., Allen,H.L., Sanchis-Juan,A., Frontini,M., Thys,C. *et al.* (2020) Whole-genome sequencing of patients with rare diseases in a national health system. *Nature*, **583**, 96–102.
6. Marshall,C.R., Chowdhury,S., Taft,R.J., Lebo,M.S., Buchan,J.G., Harrison,S.M., Rowsey,R., Klee,E.W., Liu,P., Worthey,E.A. *et al.* (2020) Best practices for the analytical validation of clinical whole-genome sequencing intended for the diagnosis of germline disease. *npj Genomic Med.*, **5**, 47.
7. Qin,Y., Koehler,S., Zhao,S., Mai,R., Liu,Z., Lu,H. and Xing,C. (2020) High-throughput, low-cost and rapid DNA sequencing using surface-coating techniques. bioRxiv doi: https://doi.org/10.1101/2020.12.10.418962, 11 December 2020, preprint: not peer reviewed.
8. Mahmoud,M., Gobet,N., Cruz-Dávalos,D.I., Mounier,N., Dessimoz,C. and Sedlazeck,F.J. (2019) Structural variant calling: the long and the short of it. *Genome Biol.*, **20**, 246.
9. Chen,X., Schulz-Trieglaff,O., Shaw,R., Barnes,B., Schlesinger,F., Källberg,M., Cox,A.J., Kruglyak,S. and Saunders,C.T. (2016) Manta: rapid detection of structural variants and indels for germline and cancer sequencing applications. *Bioinformatics*, **32**, 1220–1222.
10. Cameron,D.L., Schröder,J., Penington,J.S., Do,H., Molania,R., Dobrovic,A., Speed,T.P. and Papenfuss,A.T. (2017) GRIDSS: sensitive and specific genomic rearrangement detection using positional de bruijn graph assembly. *Genome Res.*, **27**, 2050–2060.
11. Khorsand,P. and Hormozdiari,F. (2021) Nebula: ultra-efficient mapping-free structural variant genotyper. *Nucleic Acids Res.*, **49**, e47.

12. Ebert,P., Audano,P.A., Zhu,Q., Rodriguez-Martin,B., Porubsky,D., Bonder,M.J., Sulovari,A., Ebler,J., Zhou,W., Serra Mari,R. *et al.* (2021) Haplotype-resolved diverse human genomes and integrated analysis of structural variation. *Science*, **372**, eabf7117.
13. Fan,X., Chaisson,M., Nakhleh,L. and Chen,K. (2017) HySA: a hybrid structural variant assembly approach using next-generation and single-molecule sequencing technologies. *Genome Res.*, **27**, 793–800.
14. Kosugi,S., Momozawa,Y., Liu,X., Terao,C., Kubo,M. and Kamatani,Y. (2019) Comprehensive evaluation of structural variation detection algorithms for whole genome sequencing. *Genome Biol.*, **20**, 117.
15. Cameron,D.L., Di Stefano,L. and Papenfuss,A.T. (2019) Comprehensive evaluation and characterisation of short read general-purpose structural variant calling software. *Nat. Commun.*, **10**, 3240.
16. Sarwal,V., Niehus,S., Ayyala,R., Chang,S., Lu,A., Darci-Maher,N., Littman,R., Chhugani,K., Soylev,A., Comarova,Z. *et al.* (2020) A comprehensive benchmarking of WGS-based structural variant callers. bioRxiv doi: https://doi.org/10.1101/2020.04.16.045120, 18 April 2020, preprint: not peer reviewed.
17. Tham,C.Y., Tirado-Magallanes,R., Goh,Y., Fullwood,M.J., Koh,B.T.H., Wang,W., Ng,C.H., Chng,W.J., Thiery,A., Tenen,D.G. *et al.* (2020) NanoVar: accurate characterization of patients' genomic structural variants using low-depth nanopore sequencing. *Genome Biol.*, **21**, 56.
18. Pedersen,B.S. and Quinlan,A.R. (2018) Mosdepth: quick coverage calculation for genomes and exomes. *Bioinformatics*, **34**, 867–868.
19. Heller,D. and Vingron,M. (2019) SVIM: structural variant identification using mapped long reads. *Bioinformatics*, **35**, 2907–2915.
20. Roberts,M., Hayes,W., Hunt,B.R., Mount,S.M. and Yorke,J.A. (2004) Reducing storage requirements for biological sequence comparison. *Bioinformatics*, **20**, 3363–3369.
21. Virtanen,P., Gommers,R., Oliphant,T.E., Haberland,M., Reddy,T., Cournapeau,D., Burovski,E., Peterson,P., Weckesser,W., Bright,J. *et al.* (2020) SciPy 1.0: fundamental algorithms for scientific computing in python. *Nat. Methods*, **17**, 261–272.
22. Ronen,R., Boucher,C., Chitsaz,H. and Pevzner,P. (2012) SEQuel: improving the accuracy of genome assemblies. *Bioinformatics*, **28**, i188–i196.
23. Lee,C., Grasso,C. and Sharlow,M.F. (2002) Multiple sequence alignment using partial order graphs. *Bioinformatics*, **18**, 452–464.
24. Knuth,D.E. (2011) In: *The Art of Computer Programming: Combinatorial Algorithms, part 1*. 1st edn., Addison-Wesley Professional.
25. Šošić,M. and Šikić,M. (2017) Edlib: a C/C ++ library for fast, exact sequence alignment using edit distance. *Bioinformatics*, **33**, 1394–1395.
26. Farrar,M. (2007) Striped smith–waterman speeds database searches six times over other SIMD implementations. *Bioinformatics*, **23**, 156–161.
27. Nakamura,K., Oshima,T., Morimoto,T., Ikeda,S., Yoshikawa,H., Shiwa,Y., Ishikawa,S., Linak,M.C., Hirai,A., Takahashi,H. *et al.* (2011) Sequence-specific error profile of illumina sequencers. *Nucleic Acids Res.*, **39**, e90.
28. Pedersen,B.S. and Quinlan,A.R. (2019) Duphold: scalable, depth-based annotation and curation of high-confidence structural variant calls. *GigaScience*, **8**, giz040.
29. Robinson,J.T., Thorvaldsdóttir,H., Winckler,W., Guttman,M., Lander,E.S., Getz,G. and Mesirov,J.P. (2011) Integrative genomics viewer. *Nat. Biotechnol.*, **29**, 24–26.
30. Ke,G., Meng,Q., Finley,T., Wang,T., Chen,W., Ma,W., Ye,Q. and Liu,T.-Y. (2017) LightGBM: a highly efficient gradient boosting decision tree. In: *Advances in Neural Information Processing Systems*. Vol. **30**.
31. Pedregosa,F., Varoquaux,G., Gramfort,A., Michel,V., Thirion,B., Grisel,O., Blondel,M., Prettenhofer,P., Weiss,R., Dubourg,V. *et al.* (2011) Scikit-learn: machine learning in python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
32. Bolognini,D., Sanders,A., Korbel,J.O., Magi,A., Benes,V. and Rausch,T. (2020) VISOR: a versatile haplotype-aware structural variant simulator for short- and long-read sequencing. *Bioinformatics*, **36**, 1267–1269.
33. Li,H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv doi: https://arxiv.org/abs/1303.3997, 26 May 2013, preprint: not peer reviewed.
34. Zook,J.M., Hansen,N.F., Olson,N.D., Chapman,L., Mullikin,J.C., Xiao,C., Sherry,S., Koren,S., Phillippy,A.M., Boutros,P.C. *et al.* (2020) A robust benchmark for detection of germline large deletions and insertions. *Nat. Biotechnol.*, **38**, 1347–1355.
35. McKinney,W. (2010) In: *Data Structures for Statistical Computing in Python*. Austin, Texas, pp. 56–61.
36. Alekseyenko,A.V. and Lee,C.J. (2007) Nested containment list (NCList): a new algorithm for accelerating interval query of genome alignment and interval databases. *Bioinformatics*, **23**, 1386–1393.
37. Hagberg,A., Schult,D. and Swart,P. (2008) Exploring network structure, dynamics, and function using networkX. In: *Proceedings of the 7th Python in Science Conference*. pp.11–15.
38. Proceedings of the python in science conference (SciPy): exploring network structure, dynamics, and function using networkX. https://networkx.org(13 January 2022, date last accessed).
39. Parikh,H., Mohiyuddin,M., Lam,H.Y.K., Iyer,H., Chen,D., Pratt,M., Bartha,G., Spies,N., Losert,W., Zook,J.M. *et al.* (2016) svclassify: a method to establish benchmark structural variant calls. *BMC Genomics*, **17**, 64.
40. Li,H., Bloom,J.M., Farjoun,Y., Fleharty,M., Gauthier,L., Neale,B. and MacArthur,D. (2018) A synthetic-diploid benchmark for accurate variant calling evaluation. *Nat. Methods*, **15**, 595–597.
41. Sudmant,P.H., Rausch,T., Gardner,E.J., Handsaker,R.E., Abyzov,A., Huddleston,J., Zhang,Y., Ye,K., Jun,G., Hsi-Yang Fritz,M. *et al.* (2015) An integrated map of structural variation in 2,504 human genomes. *Nature*, **526**, 75–81.
42. Rausch,T., Zichner,T., Schlattl,A., Stütz,A.M., Benes,V. and Korbel,J.O. (2012) DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, **28**, i333–i339.
43. Layer,R.M., Chiang,C., Quinlan,A.R. and Hall,I.M. (2014) LUMPY: a probabilistic framework for structural variant discovery. *Genome Biol.*, **15**, R84.
44. Kim,S., Scheffler,K., Halpern,A.L., Bekritsky,M.A., Noh,E., Källberg,M., Chen,X., Kim,Y., Beyter,D., Krusche,P. *et al.* (2018) Strelka2: fast and accurate calling of germline and somatic variants. *Nat. Methods*, **15**, 591–594.
45. Lex,A., Gehlenborg,N., Strobelt,H., Vuillemot,R. and Pfister,H. (2014) UpSet: visualization of intersecting sets. *IEEE Trans. Vis. Comput. Graph.*, **20**, 1983–1992.
46. Fang,L., Hu,J., Wang,D. and Wang,K. (2018) NextSV: a meta-caller for structural variants from low-coverage long-read sequencing data. *BMC Bioinf.*, **19**, 180.
47. Becker,T., Lee,W.-P., Leone,J., Zhu,Q., Zhang,C., Liu,S., Sargent,J., Shanker,K., Mil-homens,A., Cerveira,E. *et al.* (2018) FusorSV: an algorithm for optimally combining data from multiple structural variation detection methods. *Genome Biol.*, **19**, 38.
48. Zarate,S., Carroll,A., Mahmoud,M., Krasheninina,O., Jun,G., Salerno,W.J., Schatz,M.C., Boerwinkle,E., Gibbs,R.A. and Sedlazeck,F.J. (2020) Parliament2: accurate structural variant calling at scale. *GigaScience*, **9**, giaa145.
49. Sedlazeck,F.J., Rescheneder,P., Smolka,M., Fang,H., Nattestad,M., von Haeseler,A. and Schatz,M.C. (2018) Accurate detection of complex structural variations using single-molecule sequencing. *Nat. Methods*, **15**, 461–468.
50. Li,H. (2018) Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, **34**, 3094–3100.
51. Kent,W.J. (2002) BLAT—the BLAST-Like alignment tool. *Genome Res.*, **12**, 656–664.