sv-callers: a highly portable parallel workflow for structural variant detection in whole-genome sequence data

Arnold Kuzniar[1], Jason Maassen[1], Stefan Verhoeven[1], Luca Santuari[2], Carl Shneider[2], Wigard P. Kloosterman[2] and Jeroen de Ridder[2]

[1] Netherlands eScience Center, Amsterdam, Netherlands
[2] Center for Molecular Medicine, University Medical Center Utrecht, Utrecht, Netherlands

ABSTRACT

Structural variants (SVs) are an important class of genetic variation implicated in a wide array of genetic diseases including cancer. Despite the advances in whole genome sequencing, comprehensive and accurate detection of SVs in short-read data still poses some practical and computational challenges. We present *sv-callers*, a highly portable workflow that enables parallel execution of multiple SV detection tools, as well as provide users with example analyses of detected SV callsets in a Jupyter Notebook. This workflow supports easy deployment of software dependencies, configuration and addition of new analysis tools. Moreover, porting it to different computing systems requires minimal effort. Finally, we demonstrate the utility of the workflow by performing both somatic and germline SV analyses on different high-performance computing systems.

Subjects Bioinformatics, Genomics, Data Science
Keywords Scientific workflow, Cancer genomics, Variant calling, Structural variants, High-performance computing, Cloud computing, Snakemake, Xenon, Research software, Open science

Submitted 15 May 2019
Accepted 14 November 2019
Published 6 January 2020

Corresponding authors
Arnold Kuzniar, a.kuzniar@esciencecenter.nl
Jeroen de Ridder, j.deridder-4@umcutrecht.nl

Academic editor
Joseph Gillespie

Additional Information and Declarations can be found on page 8

DOI **10.7717/peerj.8214**

INTRODUCTION

Structural variants (SVs) such as deletions, insertions and duplications account for a large part of the genomic diversity among individuals and have been implicated in many diseases including cancer. With the advent of novel DNA sequencing technologies, whole genome sequencing (WGS) is becoming an integral part of cancer diagnostics that can potentially enable tailored treatments of individual patients (*Stratton, 2011*). Despite advances in large-scale cancer genomics projects (such as the TCGA and PCAWG of the International Cancer Genome Consortium; https://icgc.org/), systematic and comprehensive analysis of massive genomic data, in particular the detection of SVs in genomes, remains challenging due to computational and algorithmic limitations (*Alkan, Coe & Eichler, 2011*; *Yung et al., 2017*; *Ma et al., 2018*; *Gröbner et al., 2018*).

Recent tools for somatic and germline SV detection (callers) exploit more than one type of information present in WGS data (*Lin et al., 2015*). For example, DELLY (*Rausch et al., 2012*) relies on split reads and discordant read pairs while LUMPY (*Layer et al., 2014*) additionally utilizes read depth information. Furthermore, callers such as Manta

(*Chen et al., 2016*) and GRIDSS (*Cameron et al., 2017*) also incorporate short-read assembly. To obtain a more comprehensive and/or accurate callset, ensemble approaches have yielded promising results (*English et al., 2015*; *Mohiyuddin et al., 2015*; *Becker et al., 2018*; *Fang et al., 2018*). In such an approach, (i) a range of SV callers are executed, and (ii) their results are combined into a single callset. While this approach has been demonstrated to improve SV callsets, the step (i) poses a major bottleneck as running multiple SV callers efficiently on the user's computational infrastructure and/or adding new SV callers (as they become available) is far from straightforward.

A common practice to couple multiple tools together is by monolithic "wrapper" scripts (*English et al., 2015*; *Fang et al., 2018*) and, to a lesser extent, by a workflow system. The latter is a recommended approach to improve the extensibility, portability and reproducibility of data-intensive analyses (*Leipzig, 2017*). Specifically, multiple command-line tools, usually written in different languages, are integrated in a workflow (instance) that is defined either by an implicit syntax (*Köster & Rahmann, 2012*; *Holmes & Mungall, 2017*) or by an explicit syntax (*Afgan et al., 2018*; *Vivian et al., 2017*). In addition, a workflow system might support parallel execution of tasks through a batch scheduler that distributes the tasks across nodes of a high-performance computing (HPC) system. For example, Snakemake is an established generic workflow system inspired by the GNU Make build automation tool (*Köster & Rahmann, 2012*). It requires a "recipe" with implicit rules to generate output from input files. Moreover, Snakemake supports Conda package manager and light-weight containers such as Docker (https://www.docker.com/) or Singularity (*Kurtzer, Sochat & Bauer, 2017*), Common Workflow Language (*Amstutz et al., 2016*) and parallel execution in HPC cluster (via DRMAA, http://www.drmaa.org/) or cloud environments (via Kubernetes, https://kubernetes.io/). Furthermore, a workflow developed on one system is not necessarily portable to or easily reusable on another system due to the complexity of software environments, system usage policies (e.g., lack of root privilege or absence of Docker) and/or the use of different batch schedulers. Typically, compute clusters rely on batch schedulers such as Grid Engine, Slurm or Torque to manage resources and to distribute tasks over the available nodes. As a result, workflows need to be adapted to use scheduler- or file transfer-specific commands, which makes the distribution of compute tasks across (heterogeneous) systems cumbersome.

To alleviate these problems, we developed *sv-callers*, a user-friendly, portable and scalable workflow based on the Snakemake and Xenon (middleware) software. The workflow includes state-of-the-art somatic and germline SV callers, which can be easily extended, and runs on HPC clusters or clouds with minimal effort. It supports all the major SV types (i.e., deletions, insertions, inversions, duplications and translocations) as detected by the individual callers. In this paper, we focus on the implementation and computational performance of the workflow as well as provide examples to improve the reproducibility of downstream analyses.

METHODS

Experimental data and setup

The *sv-callers* workflow was tested with two human WGS datasets, one for the single-sample mode (i.e., germline SV detection) and the other for the paired-sample mode (i.e., somatic SV detection in tumor with matched normal). The *benchmark* data were obtained by Illumina HiSeq 2500 sequencing of the NA12878 genome (BioProject: PRJNA200694, ENA:SRX1049768–SRX1049855, 148 GiB BAM file downsampled to 30× coverage) released by the NIST Genome in a Bottle Consortium (*Zook et al., 2016*). The *cell lines* data were obtained by Illumina HiSeq X Ten sequencing of a cancer cell line derived from malignant melanoma (COLO829, EFO:0002140) at 90× coverage (ENA: ERX2765496, 184 GiB BAM file) and a matching control, lymphoblastoid cell line (EFO:0005292) at 30× coverage (ENA:ERX2765495, 67 GiB BAM file). Illumina short reads were mapped to the human reference genome (i.e., b37 and GRCh37 were used for the *benchmark* and the *cell lines* data, respectively) using the BWA-MEM algorithm (*Li, 2013*). To increase the amount of jobs, the WGS samples in BAM files were copied ten times, resulting in 140 jobs per dataset (of which 80 and 60 were SV calling and post-processing jobs, respectively). This allowed us to reliably estimate compute resources used as well as to assess if the results are correct and/or reproducible across the sample copies using the UpSet(R) plots (*Conway, Lex & Gehlenborg, 2017*).

Structural variation detection

Four SV callers were included in the workflow namely Manta (v1.1.0), DELLY (v0.7.7), LUMPY (v0.2.13) and GRIDSS (v1.3.4), taking into account the state-of-the-art methods (e.g., used in the ICGC projects), active development and software/documentation quality. Moreover, each of the callers have been extensively validated in their respective publications. Most callers make use of more than one programming language and take advantage of multi-core architectures (Table 1). Further, we accommodated the Conda package manager (https://conda.io/) including the 'bioconda' channel (*Da Veiga Leprevost et al., 2017*) to ease the installation of required bioinformatics software. Structural variant detection involved the callers' default settings and 'best practices' for post-processing the callsets in VCF/BCF files: (i) genomic regions with anomalous mappability (ENCODE: ENCFF001TDO in BED file) were filtered using the SURVIVOR software (v1.0.6; *Jeffares et al., 2017*) with `filter` sub-command, (ii) low-quality calls and/or (iii) germline calls in the paired-sample (somatic) mode were filtered using BCFtools (v1.9; *Li et al., 2009*), and finally (iv) the resulting SV calls of the four detection tools were merged into one (union) set using SURVIVOR with `merge` sub-command with user parameters defined in the workflow configuration file (`analysis.yaml`). In our example analyses (*Kuzniar & Santuari, 2019*), two SVs were considered overlapping and merged into one entry if the distance between the relative breakpoints at each end is smaller than 100 bp, regardless of the SV type. Note that this threshold is within the estimated insert sizes of the *benchmark* (552 bp) and the *cell lines* (518/531 bp for tumor/normal) datasets. Furthermore, the germline SV calls of each tool including the derived (merged) callsets

Table 1 **SV detection tools included in the workflow.**

Software	Implementation	Parallelism	I/O files
Manta	C++, Python	pyFlow tasks[1], SIMD[2]	FASTA, BAM or CRAM/VCF
DELLY	C++	OpenMP threads[3]	FASTA, BAM/BCF
LUMPY	C/C++, Python	–	FASTA, BAM/VCF
GRIDSS	Java, R, Python	Java threads, SIMD[2]	FASTA, BAM or CRAM/VCF or BCF

Notes:
[1] http://illumina.github.io/pyflow.
[2] Single instruction multiple data vector processing.
[3] Depends on the number of input samples used (i.e., max. two in the paired-sample mode).

were evaluated in terms of precision (TP/(TP + FP)) and recall (TP/(TP + FN)) using the StructuralVariantAnnotation R package (v1.0.0; *Cameron & Dong, 2019*) with high-confidence deletions (truth sets) published by *Layer et al. (2014)* (PacBio/Moleculo data, n = 4,095) and *Parikh et al. (2016)* (Personalis/1000 Genome Project data, n = 2,676).

Workflow implementation and deployment

We implemented the *sv-callers* workflow (v1.1.0; *Kuzniar et al., 2018*; *Kuzniar, 2019*) using the Snakemake workflow system (v4.8.0; *Köster & Rahmann, 2012*) and the *Xenon* software suite (v3.0.0). Figure 1A shows a schematic diagram of the workflow including input/output files. To perform an analysis, a user provides a CSV file with the (paired) samples and adjusts the workflow configuration file(s) accordingly. The file `analysis.yaml` is concerned with the analysis itself (such as the mode, reference genome, exclusion list, callers, post-processing, resource requirements, etc.) while the file `environment.yaml` includes software dependencies/versions. Moreover, the workflow parameters are also accessible through the command-line interface (see `snakemake -C [params]` argument).

We developed the *Xenon* library (*Maassen et al., 2018*) and the associated command-line interface, *xenon-cli* (*Verhoeven & Spaaks, 2019*) to provide uniform access to different batch schedulers and file transfer clients. *Xenon* is comparable to the Simple API for Grid Application (SAGA) (*Merzky, Weidner & Jha, 2015*) and its predecessor, the Grid Application Toolkit (GAT) (*Van Nieuwpoort, Kielmann & Bal, 2007*). These APIs are client-side solutions rather than a server-side standardization attempt such as the Distributed Resource Management Application API (DRMAA) (*Troger et al., 2007*). Specifically, *Xenon* supports several batch schedulers such as Grid Engine, Slurm and Torque, local or remote execution on a server via SSH, as well as local or remote file access via the (S)FTP and WebDAV transfer protocols, Amazon S3 cloud storage or Apache Hadoop Distributed File System (HDFS). As shown in Figure 1B, the *Xenon* library translates the *xenon-cli* commands to scheduler-specific commands used by a target cluster. When the cluster is changed or multiple clusters are used, *Xenon* will simply use a different translation, while the *xenon-cli* commands will remain essentially the same. A user needs to modify only a scheduler type or a queue name. For this reason, *xenon-cli* was configured as the cluster submission command for our Snakemake-based workflow, which improved the portability by making the workflow independent of the

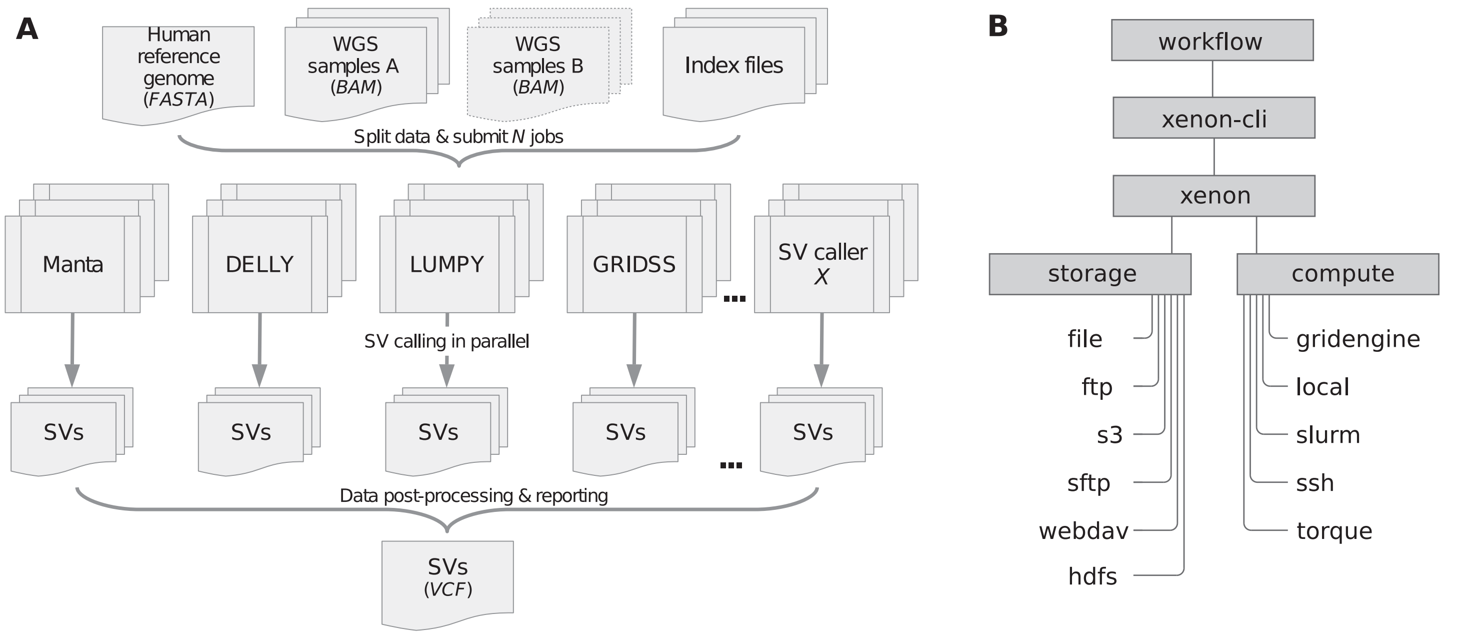

Figure 1 **Workflow/Xenon software architecture.** (A) The workflow includes four SV callers namely Manta, DELLY, LUMPY and GRIDSS to detect SVs in (paired) samples given a reference genome. Submitted jobs execute in parallel across compute nodes. SVs detected per caller are filtered and merged into one (union) call set. Note: DELLY requires separate runs for each SV type (i.e., DUP, duplication; DEL, deletion; INS, insertion; INV, inversion; BND, translocation). (B) The workflow uses the Xenon command-line interface (xenon-cli) that abstracts away scheduler- and file transfer-specific commands; the Xenon library translates each command to the one used by the target system.
Full-size DOI: 10.7717/peerj.8214/fig-1

system-specific solutions (see `snakemake --cluster` '`xenon scheduler [type]`' argument).

Finally, the software reliability has been ensured by regular unit and continuous integration tests in the cloud (Travis CI, https://travis-ci.org/), which involve Docker images of different batch systems including real test data (NA12878). After successful tests, the workflow was deployed on different academic HPC systems namely the Grid Engine-based cluster at Utrecht Bioinformatics Center (UBC, https://ubc.uu.nl/), the Slurm-based Distributed ASCI Supercomputer 5 (*Bal et al., 2016*; https://www.cs.vu.nl/das5/) and the Dutch national supercomputer (Cartesius, https://userinfo.surfsara.nl/systems/cartesius/).

RESULTS

The *sv-callers* workflow readily automated parallel execution of the tools across compute nodes and enabled streamlined data analyses for example, in a Jupyter Notebook. We tested the workflow with each WGS dataset (i.e., *benchmark* and *cell lines)* on both Grid Engine- and Slurm-based HPC systems, processing in total about 4 TiB of data. Overall, the datasets were processed at different rates on each system due to differences in data size and coverage, and system load or availability at the time of use (data not shown). Submitted jobs of the *benchmark* and of the *cell lines* datasets completed successfully (100%) in about twelve hours and in 4 days, respectively. Table 2 shows the compute resources used by the SV callers (on the UBC cluster). We found that the detection of somatic SVs required significantly more compute time or resources than that of germline

Table 2 **Compute resources used by the SV callers.**

Software	Dataset	Threads per job	Wall-clock/CPU times* (HH:MM)	Peak mem. (GiB)	Output
Manta	Cell lines	24	01:45/14:31	5.3	94 MiB
	Benchmark	24	00:49/05:42	2.8	225 MiB
DELLY	Cell lines	2	19:23/25:47 66:03/85:20	3.6	13 MiB
	Benchmark	1	00:54/00:49 09:07/08:35	0.9	28 MiB
LUMPY	Cell lines	1	13:40/20:55	24.4	5.9 GiB
	Benchmark	1	00:29/00:45	5.7	<1 MiB
GRIDSS	Cell lines	24	29:31/214:13	44.4	73 GiB
	Benchmark	24	08:24/26:04	38.5	342 MiB

Notes:
* Timings per sample or per job are averaged using median over ten WGS (paired) samples.
DELLY calls per sample were summed over jobs of distinct SV types.
LUMPY is mostly single-threaded except its I/O related code, which results in CPU time > wall-clock time.

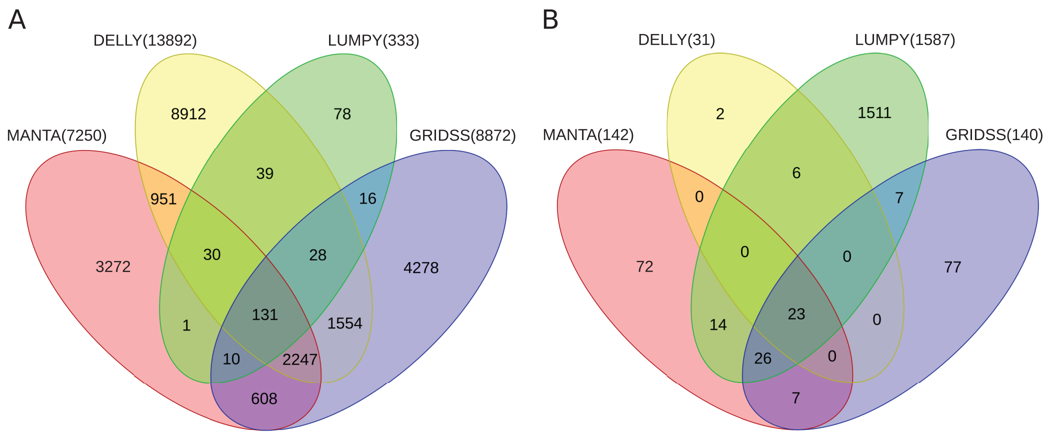

Figure 2 **Venn diagrams of intersecting SV call sets.** (A) Germline and (B) somatic SVs detected in the *benchmark* and in the *cell lines* samples, respectively, using Manta, DELLY, LUMPY and GRIDSS. Most SVs are caller-specific, followed by SVs common to three of the four callers. SVs detected by the callers were filtered and merged into one set (see Section Methods). Note: The Venn diagrams include the largest GRIDSS sets as the GRIDSS output varies slightly each run using the same input. Figs. S1 and S2 show the comparisons across sample copies. Full-size DOI: 10.7717/peerj.8214/fig-2

SVs for the datasets used (335 vs. 41 CPU core hours). In particular, the analyses by DELLY or GRIDSS were computationally more costly per sample than those by Manta or LUMPY (see Data S1).

Further, the resulting VCF files with SV calls were checked for completeness and for overlap among the callers and sample copies. Most SVs were caller-specific while the second largest category of SVs were common to three of the four callers (Fig. 2). Importantly, merging SV calls supported by the majority of tools resulted in improved precision (but not recall) compared to the best caller according to the PacBio/Moleculo (94.8% vs. 81.6% [LUMPY]) and the Personalis/1kGP (88.1% vs. 76.8% [GRIDSS]) truth sets (Fig. 3; Data S2).

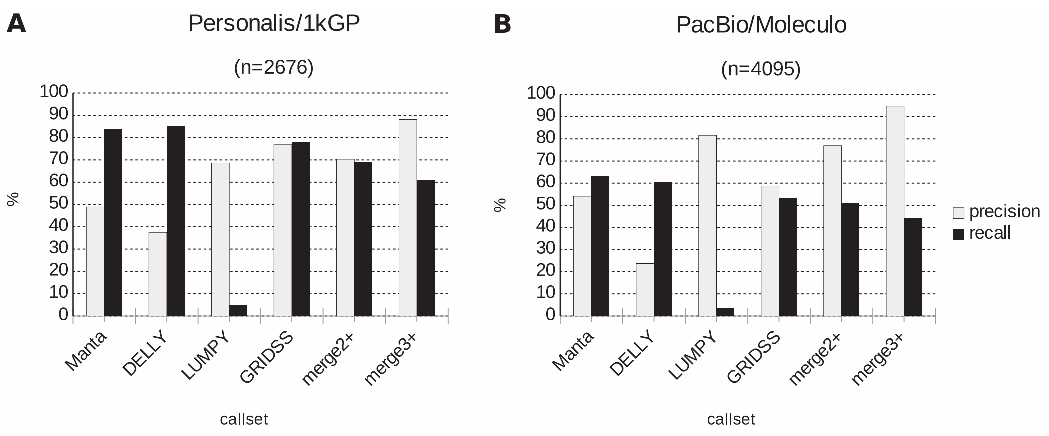

Figure 3 Precision and recall of SV detection based on a single vs. multiple caller(s) according to two truth sets. (A) For Personalis/1kGP data, the merged calls of at least three SV callers (denoted as "*merge3+*") have best precision (88.1%) while DELLY calls have best recall (85.3%). (B) For PacBio/Moleculo data, the "*merge3+*" calls have best precision (94.8%) while Manta calls have best recall (63%). Full-size DOI: 10.7717/peerj.8214/fig-3

CONCLUSIONS

Relying on a single SV detection algorithm or caller is insufficient for comprehensive and accurate detection of SVs in cancer genomes (*English et al., 2015*; *Fang et al., 2018*; *Kosugi et al., 2019*). Therefore, there is an increasing need for flexible and portable bioinformatics workflows that—with minimal effort—integrate multiple SV detection and post-processing tools while making efficient use of different HPC cluster or cloud infrastructures. To address these needs, we developed the *sv-callers* workflow according to best practices in research software (*Jiménez et al., 2017*; *Maassen et al., 2017*; https://guide.esciencecenter.nl/). Further improvements to the workflow itself and/or its 'backend' are possible. For example, adding multiple read mapping tools to the workflow would enable analyses directly from raw sequencing data. Moreover, the SV reporting step could be enhanced with interactive visualization to facilitate manual inspection of SVs in genomic regions of interest. Furthermore, the SV callers' binaries (currently distributed *via* bioconda) are suboptimal regarding the performance for target systems used, and therefore the codes might benefit from further optimization (e.g., using the Intel C/C++ Compiler). We aim to extend cloud support in Xenon with schedulers such as Microsoft Azure Batch and Amazon AWS Batch as well as to improve the coupling between Xenon and Snakemake software. Finally, the *sv-callers* workflow is an open-source software that is freely available from the ELIXIR's bio.tools registry (https://bio.tools/sv-callers) and Research Software Directory (https://research-software.nl/software/sv-callers).

ACKNOWLEDGEMENTS

The authors would like to thank HPC system administrators and SURFsara consultants in particular Jeroen Engelberts and Lyke Voort for technical support, and the reviewers for their valuable suggestions.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by the Netherlands eScience Center (Grant Number: 027016G03) and was carried out on the Dutch national e-infrastructure with the support of SURF Foundation (NWO Project Number: 16669). The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Grant Disclosures

The following grant information was disclosed by the authors:
The Netherlands eScience Center: 027016G03.
Dutch National e-infrastructure with the support of SURF Foundation (NWO): 16669.

Competing Interests

Jeroen de Ridder is co-founder of Cyclomics B.V.

Author Contributions

- Arnold Kuzniar conceived and designed the experiments, performed the experiments, developed the software, analyzed the data, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Jason Maassen (co)developed the software, analyzed the data, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Stefan Verhoeven (co)developed the software, reviewed drafts of the paper, and approved the final draft.
- Luca Santuari contributed to the software, analyzed the data, performed the experiments, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Carl Shneider contributed to the software prototype, reviewed drafts of the paper, and approved the final draft.
- Wigard P. Kloosterman conceived and designed the experiments, reviewed drafts of the paper, and approved the final draft.
- Jeroen de Ridder conceived and designed the experiments, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The source code including test data were deposited at the Zenodo archive (DOI 10.5281/zenodo.1217111 and DOI 10.5281/zenodo.2663307).

Supplemental Information

Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj.8214#supplemental-information.

REFERENCES

Afgan E, Baker D, Batut B, Van Den Beek M, Bouvier D, Cech M, Chilton J, Clements D, Coraor N, Grüning BA, Guerler A, Hillman-Jackson J, Hiltemann S, Jalili V, Rasche H, Soranzo N, Goecks J, Taylor J, Nekrutenko A, Blankenberg D. 2018. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research* **46(W1)**:W537–W544 DOI 10.1093/nar/gky379.

Alkan C, Coe BP, Eichler EE. 2011. Genome structural variation discovery and genotyping. *Nature Reviews Genetics* **12(5)**:363–376 DOI 10.1038/nrg2958.

Amstutz P, Crusoe MR, Tijanić N, Chapman B, Chilton J, Heuer M, Kartashov A, Leehr D, Ménager H, Nedeljkovich M, Scales M, Soiland-Reyes S, Stojanovic L. 2016. Common workflow language, v1.0. *figshare* DOI 10.6084/m9.figshare.3115156.v2.

Bal H, Epema D, De Laat C, Van Nieuwpoort R, Romein J, Seinstra F, Snoek C, Wijshoff H. 2016. A medium-scale distributed system for computer science research: infrastructure for the long term. *Computer* **49(5)**:54–63 DOI 10.1109/MC.2016.127.

Becker T, Lee WP, Leone J, Zhu Q, Zhang C, Liu S, Sargent J, Shanker K, Mil-Homens A, Cerveira E, Ryan M, Cha J, Navarro FCP, Galeev T, Gerstein M, Mills RE, Shin DG, Lee C, Malhotra A. 2018. FusorSV: an algorithm for optimally combining data from multiple structural variation detection methods. *Genome Biology* **19(1)**:38 DOI 10.1186/s13059-018-1404-6.

Cameron D, Dong R. 2019. *StructuralVariantAnnotation: variant annotations for structural variants*. R package (version 1.0.0). *Available at https://bioconductor.org*.

Cameron DL, Schröder J, Penington JS, Do H, Molania R, Dobrovic A, Speed TP, Papenfuss AT. 2017. GRIDSS: sensitive and specific genomic rearrangement detection using positional de Bruijn graph assembly. *Genome Research* **27(12)**:2050–2060 DOI 10.1101/gr.222109.117.

Chen X, Schulz-Trieglaff O, Shaw R, Barnes B, Schlesinger F, Källberg M, Cox AJ, Kruglyak S, Saunders CT. 2016. Manta: rapid detection of structural variants and indels for germline and cancer sequencing applications. *Bioinformatics* **32(8)**:1220–1222 DOI 10.1093/bioinformatics/btv710.

Conway JR, Lex A, Gehlenborg N. 2017. UpSetR: an R package for the visualization of intersecting sets and their properties. *Bioinformatics* **33(18)**:2938–2940 DOI 10.1093/bioinformatics/btx364.

Da Veiga Leprevost F, Grüning BA, Alves Aflitos S, Röst HL, Uszkoreit J, Barsnes H, Vaudel M, Moreno P, Gatto L, Weber J, Bai M, Jimenez RC, Sachsenberg T, Pfeuffer J, Vera Alvarez R, Griss J, Nesvizhskii AI, Perez-Riverol Y. 2017. BioContainers: an open-source and community-driven framework for software standardization. *Bioinformatics* **33(16)**:2580–2582 DOI 10.1093/bioinformatics/btx192.

English AC, Salerno WJ, Hampton OA, Gonzaga-Jauregui C, Ambreth S, Ritter DI, Beck CR, Davis CF, Dahdouli M, Ma S, Carroll A, Veeraraghavan N, Bruestle J, Drees B, Hastie A, Lam ET, White S, Mishra P, Wang M, Han Y, Zhang F, Stankiewicz P, Wheeler DA, Reid JG, Muzny DM, Rogers J, Sabo A, Worley KC, Lupski JR, Boerwinkle E, Gibbs RA. 2015. Assessing structural variation in a personal genome—towards a human reference diploid genome. *BMC Genomics* **16(1)**:286 DOI 10.1186/s12864-015-1479-3.

Fang L, Hu J, Wang D, Wang K. 2018. NextSV: a meta-caller for structural variants from low-coverage long-read sequencing data. *BMC Bioinformatics* **19(1)**:180 DOI 10.1186/s12859-018-2207-1.

Gröbner SN, Worst BC, Weischenfeldt J, Buchhalter I, Kleinheinz K, Rudneva VA, Johann PD, Balasubramanian GP, Segura-Wang M, Brabetz S, Bender S, Hutter B, Sturm D, Pfaff E,

Hübschmann D, Zipprich G, Heinold M, Eils J, Lawerenz C, Erkek S, Lambo S, Waszak S, Blattmann C, Borkhardt A, Kuhlen M, Eggert A, Fulda S, Gessler M, Wegert J, Kappler R, Baumhoer D, Burdach S, Kirschner-Schwabe R, Kontny U, Kulozik AE, Lohmann D, Hettmer S, Eckert C, Bielack S, Nathrath M, Niemeyer C, Richter GH, Schulte J, Siebert R, Westermann F, Molenaar JJ, Vassal G, Witt H, Burkhardt B, Kratz CP, Witt O, Van Tilburg CM, Kramm CM, Fleischhack G, Dirksen U, Rutkowski S, Frühwald M, von Hoff K, Wolf S, Klingebiel T, Koscielniak E, Landgraf P, Koster J, Resnick AC, Zhang J, Liu Y, Zhou X, Waanders AJ, Zwijnenburg DA, Raman P, Brors B, Weber UD, Northcott PA, Pajtler KW, Kool M, Piro RM, Korbel JO, Schlesner M, Eils R, Jones DTW, Lichter P, Chavez L, Zapatka M, Pfister SM, ICGC PedBrain-Seq Project. 2018. ICGC MMML-Seq Project. 2018. The landscape of genomic alterations across childhood cancers. *Nature* **555(7696)**:321–327 DOI 10.1038/nature25480.

Holmes IH, Mungall CJ. 2017. BioMake: a GNU make-compatible utility for declarative workflow management. *Bioinformatics* **33(21)**:3502–3504 DOI 10.1093/bioinformatics/btx306.

Jeffares DC, Jolly C, Hoti M, Speed D, Shaw L, Rallis C, Balloux F, Dessimoz C, Bähler J, Sedlazeck FJ. 2017. Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast. *Nature Communications* **8(1)**:14061 DOI 10.1038/ncomms14061.

Jiménez RC, Kuzak M, Alhamdoosh M, Barker M, Batut B, Borg M, Capella-Gutierrez S, Chue Hong N, Cook M, Corpas M, Flannery M, Garcia L, Gelpí JL, Gladman S, Goble C, González Ferreiro M, Gonzalez-Beltran A, Griffin PC, Grüning B, Hagberg J, Holub P, Hooft R, Ison J, Katz DS, Leskošek B, López Gómez F, Oliveira LJ, Mellor D, Mosbergen R, Mulder N, Perez-Riverol Y, Pergl R, Pichler H, Pope B, Sanz F, Schneider MV, Stodden V, Suchecki R, Svobodová Vařeková R, Talvik H-A, Todorov I, Treloar A, Tyagi S, Van Gompel M, Vaughan D, Via A, Wang X, Watson-Haigh NS, Crouch S. 2017. Four simple recommendations to encourage best practices in research software. *F1000Research* **6**:876 DOI 10.12688/f1000research.11407.1.

Kosugi S, Momozawa Y, Liu X, Terao C, Kubo M, Kamatani Y. 2019. Comprehensive evaluation of structural variation detection algorithms for whole genome sequencing. *Genome Biology* **20(1)**:117 DOI 10.1186/s13059-019-1720-5.

Kurtzer GM, Sochat V, Bauer MW. 2017. Singularity: scientific containers for mobility of compute. *PLOS ONE* **12(5)**:e0177459 DOI 10.1371/journal.pone.0177459.

Kuzniar A. 2019. *sv-callers*. Version 1.1.0. Zenodo. *Available at http://doi.org/10.5281/zenodo.1217111*.

Kuzniar A, Maassen J, Verhoeven S, Santuari L, Shneider C, Kloosterman W, De Ridder J. 2018. A portable and scalable workflow for detecting structural variants in whole-genome sequencing data. In: *2018 IEEE 14th International Conference on e-Science (e-Science)*. Amsterdam: IEEE, 303–304 DOI 10.1109/eScience.2018.00064.

Kuzniar A, Santuari L. 2019. *Test data for sv-callers workflow*. Version 1.1.0. Zenodo. *Available at http://doi.org/10.5281/zenodo.2663307*.

Köster J, Rahmann S. 2012. Snakemake: a scalable bioinformatics workflow engine. *Bioinformatics* **28(19)**:2520–2522 DOI 10.1093/bioinformatics/bts480.

Layer RM, Chiang C, Quinlan AR, Hall IM. 2014. LUMPY: a probabilistic framework for structural variant discovery. *Genome Biology* **15(6)**:R84 DOI 10.1186/gb-2014-15-6-r84.

Leipzig J. 2017. A review of bioinformatic pipeline frameworks. *Briefings in Bioinformatics* **18(3)**:530–536 DOI 10.1093/bib/bbw020.

Li H. 2013. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *Available at http://arxiv.org/abs/1303.3997*.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, 1000 Genome Project Data Processing Subgroup. 2009. Genome project data processing subgroup, the sequence alignment/map format and SAMtools. *Bioinformatics* **25(16)**:2078–2079 DOI 10.1093/bioinformatics/btp352.

Lin K, Smit S, Bonnema G, Sanchez-Perez G, De Ridder D. 2015. Making the difference: integrating structural variation detection tools. *Briefings in Bioinformatics* **16(5)**:852–864 DOI 10.1093/bib/bbu047.

Ma X, Liu Y, Liu Y, Alexandrov LB, Edmonson MN, Gawad C, Zhou X, Li Y, Rusch MC, Easton J, Huether R, Gonzalez-Pena V, Wilkinson MR, Hermida LC, Davis S, Sioson E, Pounds S, Cao X, Ries RE, Wang Z, Chen X, Dong L, Diskin SJ, Smith MA, Guidry Auvil JM, Meltzer PS, Lau CC, Perlman EJ, Maris JM, Meshinchi S, Hunger SP, Gerhard DS, Zhang J. 2018. Pan-cancer genome and transcriptome analyses of 1,699 paediatric leukaemias and solid tumours. *Nature* **555(7696)**:371–376 DOI 10.1038/nature25795.

Maassen J, Drost N, Hage WV, Van Nieuwpoort RV. 2017. Track 2 Lightning talk: software development best practices at the Netherlands eScience Center. In: *Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE5.1), 6 September 2017, University of Manchester, Manchester, UK.* DOI 10.6084/m9.figshare.5327587.v2.

Maassen J, Verhoeven S, Borgdorff J, Spaaks JH, Drost N, Meijer C, Van Der Ploeg A, De Boer PT, Van Nieuwpoort R, Van Werkhoven B, Kuzniar A. 2018. Xenon. Zenodo. *Available at http://doi.org/10.5281/zenodo.597993*.

Merzky A, Weidner O, Jha S. 2015. SAGA: a standardized access layer to heterogeneous distributed computing infrastructure. *SoftwareX* **1–2**:3–8 DOI 10.1016/j.softx.2015.03.001.

Mohiyuddin M, Mu JC, Li J, Bani Asadi N, Gerstein MB, Abyzov A, Wong WH, Lam HYK. 2015. MetaSV: an accurate and integrative structural-variant caller for next generation sequencing. *Bioinformatics* **31(16)**:2741–2744 DOI 10.1093/bioinformatics/btv204.

Parikh H, Mohiyuddin M, Lam HY, Iyer H, Chen D, Pratt M, Bartha G, Spies N, Losert W, Zook JM, Salit M. 2016. svclassify: a method to establish benchmark structural variant calls. *BMC Genomics* **17(1)**:64 DOI 10.1186/s12864-016-2366-2.

Rausch T, Zichner T, Schlattl A, Stütz AM, Benes V, Korbel JO. 2012. DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics* **28(18)**:i333–i339 DOI 10.1093/bioinformatics/bts378.

Stratton MR. 2011. Exploring the genomes of cancer cells: progress and promise. *Science* **331(6024)**:1553–1558 DOI 10.1126/science.1204040.

Troger P, Rajic H, Haas A, Domagalski P. 2007. Standardization of an API for distributed resource management systems. In: *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07).* Rio De Janeiro: IEEE, 619–626 DOI 10.1109/CCGRID.2007.109.

Van Nieuwpoort RV, Kielmann T, Bal HE. 2007. User-friendly and reliable grid computing based on imperfect middleware. In: *Proceedings of the 2007 ACM/IEEE conference on Supercomputing - SC '07.* Reno, Nevada: ACM Press, 1 DOI 10.1145/1362622.1362668.

Verhoeven S, Spaaks JH. 2019. Xenon command line interface. Zenodo. *Available at http://doi.org/10.5281/zenodo.597603*.

Vivian J, Rao AA, Nothaft FA, Ketchum C, Armstrong J, Novak A, Pfeil J, Narkizian J, Deran AD, Musselman-Brown A, Schmidt H, Amstutz P, Craft B, Goldman M, Rosenbloom K, Cline M, O'Connor B, Hanna M, Birger C, Kent WJ, Patterson DA, Joseph AD, Zhu J, Zaranek S, Getz G, Haussler D, Paten B. 2017. Toil enables reproducible, open source, big biomedical data analyses. *Nature Biotechnology* **35(4)**:314–316 DOI 10.1038/nbt.3772.

Yung CK, O'Connor BD, Yakneen S, Zhang J, Ellrott K, Kleinheinz K, Miyoshi N, Raine KM, Royo R, Saksena GB, Schlesner M, Shorser SI, Vazquez M, Weischenfeldt J, Yuen D, Butler AP, Davis-Dusenbery BN, Eils R, Ferretti V, Grossman RL, Harismendy O, Kim Y, Nakagawa H, Newhouse SJ, Torrents D, Stein LD. 2017. Large-scale uniform analysis of cancer whole genomes in multiple computing environments. *bioRxiv. Available at https://www.biorxiv.org/content/early/2017/07/10/161638*.

Zook JM, Catoe D, McDaniel J, Vang L, Spies N, Sidow A, Weng Z, Liu Y, Mason CE, Alexander N, Henaff E, McIntyre ABR, Chandramohan D, Chen F, Jaeger E, Moshrefi A, Pham K, Stedman W, Liang T, Saghbini M, Dzakula Z, Hastie A, Cao H, Deikus G, Schadt E, Sebra R, Bashir A, Truty RM, Chang CC, Gulbahce N, Zhao K, Ghosh S, Hyland F, Fu Y, Chaisson M, Xiao C, Trow J, Sherry ST, Zaranek AW, Ball M, Bobe J, Estep P, Church GM, Marks P, Kyriazopoulou-Panagiotopoulou S, Zheng GXY, Schnall-Levin M, Ordonez HS, Mudivarti PA, Giorda K, Sheng Y, Rypdal KB, Salit M. 2016. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Scientific Data* **3(1)**:160025 DOI 10.1038/sdata.2016.25.