SOFTWARE TOOL ARTICLE

Methods developed during the first National Center for Biotechnology Information Structural Variation Codeathon at Baylor College of Medicine [version 1; peer review: 1 approved with reservations]

Medhat Mahmoud1*, Alejandro Rafael Gener2-5*, Michael M. Khayat6*, Adam C. English7, Advait Balaji8, Anbo Zhou9, Andreas Hehn10, Arkarachai Fungtammasan11, Brianna Sierra Chrisman12, Chen-Shan Chin11, Chiao-Feng Lin11, Chun-Hsuan Lo13, Chunxiao Liao14, Claudia M. B. Carvalho6, Colin Diesh15, David E. Symer16, Divya Kalra1, Dreycey Albin14, Elbay Aliyev17, Eric T. Dawson18,19, Eric Venner1, Fernanda Foertter20, Gigon Bae10, Haowei Du6, Joyjit Daw10, Junzhou Wang11, Keiko Akagi21, Lon Phan22, Michael Jochum23, Mohammadamin Edrisi18, Nirav N. Shah24, Qi Wang25, Robert Fullem6, Rong Zheng2, Sara E Kalla1, Shakuntala Mitra26, Todd J. Treangen8, Vaidhyanathan Mahaganapathy9, Venkat Sai Malladi27, Vipin K Menon1, Yilei Fu8, Yongze Yin8, Yuanqing Feng28, Tim Hefferon22, Fritz J. Sedlazeck1*, Ben Busby11,22

1Human Genome Sequencing Center, Baylor College of Medicine, Houston, Texas, 77030, USA
2Integrative Molecular and Biomedical Sciences Program, Graduate School of Biomedical Sciences, Baylor College of Medicine, Houston, Texas, 77030, USA
3Margaret M. and Albert B. Alkek Department of Medicine, Nephrology, Baylor College of Medicine, Houston, Texas, 77030, USA
4Department of Genetics, University of Texas MD Anderson Cancer Center, Houston, Texas, 77030, USA
5School of Medicine, Universidad Central del Caribe, Bayamón, Puerto Rico, 00960, USA
6Molecular and Human Genetics Department, Baylor College of Medicine, Houston, Texas, 77030, USA
7Spiral Genetics, Seattle, Washington, 98101, USA
8Department of Computer Science, Rice University, Houston, Texas, 77005, USA
9Department of Genetics, Rutgers University, Piscataway, New Jersey, 08854, USA
10NVIDIA, Santa Clara, California, 95051, USA
11DNAnexus, Mountain view, California, 94040, USA
12Department of Bioengineering, Stanford University, Stanford, California, 94305, USA
13Human Genetics Laboratory, National Institute of Genetics, Mishima City, Shizuoka Prefecture, 411-8540, Japan
14Rice University, Houston, Texas, 77005, USA
15Bioengineering Department, University of California, Berkeley, Berklely, California, USA
16Department of Lymphoma & Myeloma, University of Texas MD Anderson Cancer Center, Houston, Texas, 77030, USA
17Research Department, Sidra Medicine, Ar-Rayyan, Doha, Qatar
18Division of Cancer Epidemiology and Genetics, National Cancer Institute, Bethesda, Maryland, USA
19Department of Genetics, University of Cambridge, Cambridge, UK
20The BioTeam, Inc., Middleton, Massachusetts, 01949, USA
21Department of Thoracic Head and Neck Medical Oncology, University of Texas MD Anderson Cancer Center, Houston, Texas, 77030, USA
22
Abstract
In October 2019, 46 scientists from around the world participated in the first National Center for Biotechnology Information (NCBI) Structural Variation (SV) Codeathon at Baylor College of Medicine. The charge of this first annual working session was to identify ongoing challenges around the topics of SV and graph genomes, and in response to design reliable methods to facilitate their study. Over three days, seven working groups each designed and developed new open-sourced methods to improve the bioinformatic analysis of genomic SVs represented in next-generation sequencing (NGS) data. The groups’ approaches addressed a wide range of problems in SV detection and analysis, including quality control (QC) assessments of metagenome assemblies and population-scale VCF files, de novo copy number variation (CNV) detection based on continuous long sequence reads, the representation of sequence variation using graph genomes, and the development of an SV annotation pipeline. A summary of the questions and developments that arose during the daily discussions between groups is outlined. The new methods are publicly available at https://github.com/NCBI-Codeathons/, and demonstrate that a codeathon devoted to SV analysis can produce valuable new insights both for participants and for the broader research community.

Keywords
Structural Variant, Graph Genome, Human Genomics, Clinical Annotation, Quality Control, Codeathon

This article is included in the Sidra Medicine gateway.
This article is included in the Hackathons collection.
Corresponding authors: Alejandro Rafael Gener (itspronouncedhenner@gmail.com), Tim Heffero (theffero@nih.gov), Fritz J. Sedlazeck (fritz.sedlazeck@bcm.edu), Ben Busby (ben.busby@gmail.com)

Author roles: Mahmoud M: Conceptualization, Formal Analysis, Investigation, Methodology, Project Administration, Software, Supervision, Validation, Writing – Original Draft Preparation, Writing – Review & Editing; Gener AR: Conceptualization, Data Curation, Investigation, Methodology, Project Administration, Resources, Writing – Original Draft Preparation, Writing – Review & Editing; Khayat MM: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Project Administration, Resources, Software, Validation, Visualization, Writing – Review & Editing; English AC: Conceptualization, Formal Analysis, Investigation, Methodology, Software; Balaji A: Data Curation, Methodology, Software, Validation, Visualization; Zhou A: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Project Administration, Resources, Software, Validation, Visualization; Hneh A: Methodology, Resources, Software; Fungtammasan A: Conceptualization, Formal Analysis, Investigation, Methodology, Project Administration, Resources, Software, Validation, Visualization; Chrisman BS: Data Curation, Formal Analysis, Investigation, Methodology, Software; Chinn CS: Conceptualization, Formal Analysis, Investigation, Methodology, Resources, Supervision, Validation, Visualization, Writing – Original Draft Preparation; Lin CF: Data Curation, Writing – Original Draft Preparation; Lo CH: Conceptualization, Investigation, Methodology, Software; Liao C: Conceptualization, Methodology, Validation, Visualization; Carvalho CMB: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Resources, Supervision, Validation, Writing – Review & Editing; Diesch C: Methodology, Software, Visualization, Writing – Review & Editing; Symer DE: Conceptualization, Methodology, Writing – Original Draft Preparation, Writing – Review & Editing; Kalra D: Methodology, Resources, Software; Albín D: Data Curation, Methodology, Software, Validation, Visualization; Aliyev E: Conceptualization, Formal Analysis, Investigation, Methodology, Resources, Software, Validation, Visualization; Dawson ET: Conceptualization, Methodology, Software, Supervision; Venner E: Resources, Software; Foerster F: Supervision; Bae G: Resources, Software; Du H: Methodology, Software, Writing – Review & Editing; Daw J: Methodology, Resources, Software; Wang J: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Resources, Software, Validation; Akagi K: Conceptualization; Phan L: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Project Administration, Validation, Writing – Original Draft Preparation; Jochum M: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Resources, Software, Validation, Visualization; Edrisi M: Methodology, Resources, Software; Shah NN: Conceptualization, Data Curation, Formal Analysis, Investigation, Methodology, Resources, Software, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; Wang Q: Conceptualization, Data Curation, Formal Analysis, Investigation, Resources; Fullem R: Conceptualization, Software, Validation; Zheng R: Conceptualization, Data Curation, Investigation, Methodology; Kalla SE: Data Curation, Methodology, Resources, Writing – Original Draft Preparation; Mitra S: Conceptualization, Methodology, Resources, Visualization, Writing – Original Draft Preparation, Writing – Review & Editing; Treangen TJ: Conceptualization, Project Administration, Supervision, Writing – Review & Editing; Mahaganapathy V: Conceptualization, Formal Analysis, Methodology, Software, Supervision; Malladi VS: Conceptualization, Formal Analysis, Investigation, Methodology, Software, Validation, Writing – Review & Editing; Menon VK: Conceptualization, Methodology, Resources, Writing – Original Draft Preparation; Fu Y: Conceptualization, Data Curation, Investigation, Resources, Software, Validation, Visualization; Yin Y: Conceptualization, Data Curation, Investigation, Resources, Software, Validation, Writing – Original Draft Preparation; Feng Y: Conceptualization; Heffero T: Conceptualization, Data Curation, Resources; Sedlazeck FJ: Conceptualization, Funding Acquisition, Project Administration, Resources, Writing – Original Draft Preparation, Writing – Review & Editing; Busby B: Funding Acquisition

Competing interests: ARG received travel awards and bursaries from Oxford Nanopore Technologies, Oxford, UK. FJS received sponsored travel from PacBio and Oxford Nanopore and received the Pacbio SMRT Grant in 2018. NNS is an employee at Element Genomics, Inc., a UCB subsidiary. C-SC, AF, JW, and C-FL are employees of DNAExpress Inc. This material should not be interpreted as representing the viewpoint of the U.S. Department of Health and Human Services, the National Institutes of Health, National Library of Medicine, National Center for Biotechnology Information, National Cancer Institute, Human Genetics Laboratory at the National Institute of Genetics (Japan).

Grant information: ARG is funded in part by institutional support from Baylor College of Medicine; private funding by East Coast Oils, Inc., Jacksonville, Florida, and ARG’s own private funding, including Student Genomics (manuscripts in prep). KA was supported by the National Cancer Institute (NCI) [R50 CA211533]. ARG also received the PFLAG of Jacksonville scholarship for multiple years. CMBC is supported by the National Institute of Neurological Disorders and Stroke (NINDS) [R35 NS105078], the Eunice Kennedy Shriver National Institute of Child Health & Human Development (NICHD) [R03 HD092569] and National Institute of General Medical Sciences (NIGMS) [R01 GM106373]. FJS and MM were supported, and multiple data sets were generated, by NIH [UM1 HG008898]. The work of LP, TH and BB was supported by the Intramural Research Program of the National Library of Medicine at the NIH. YF is supported by Ken Kennedy Institute Computer Science & Engineering Enhancement Fellowship from Rice University.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2020 Mahmoud M et al. This is an open access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The author(s) is/are employees of the US Government and therefore domestic copyright protection in USA does not apply to this work. The work may be protected under the copyright laws of other jurisdictions when used in those jurisdictions.

How to cite this article: Mahmoud M, Gener AR, Khayat MM et al. Methods developed during the first National Center for Biotechnology Information Structural Variation Codeathon at Baylor College of Medicine [version 1; peer review: 1 approved with reservations] F1000Research 2020, 9:1141 https://doi.org/10.12688/f1000research.23773.1

First published: 16 Sep 2020, 9:1141 https://doi.org/10.12688/f1000research.23773.1
Introduction

Structural variants (SVs) are large-scale genomic alterations, frequently defined as greater than 50 bases (bp) in length, involving deletions, duplications, insertions, inversions and/or translocations, and can occur in combinations. SVs have been linked to multiple phenotypic differences between organisms, as well as within populations of the same species. SVs are known to play roles in myriad diseases, including neurological (e.g. Parkinson, Huntington), Mendelian, and other genomic alterations such as those seen in many cancers and constitutional diseases. In contrast to single nucleotide variants (SNVs), involving substitution of a single nucleotide, SVs remain understudied due to their more complex nature. Our understanding of these larger forms of genomic alterations is limited by the sequencing technology and computational methods available to analyze ever-increasing amounts of sequence or similar data. A special type of SV is copy number variant (CNV). These are unbalanced SVs that could either increase or decrease total DNA content through duplications and deletions, respectively. CNVs of high importance are associated with different physical diseases like obesity, type 1 diabetes, rheumatoid arthritis, and neurological disorders such as autism, schizophrenia, Mendelian diseases and other genomic disorders. More general studies examine the relationship between copy number variation and a range of diseases.

Recent studies report between 20,000 to 25,000 SVs per human genome. Although the number of SVs per individual is smaller than that for SNVs, SVs account for more altered nucleotides per diploid genome. Recently, there has been a prominent shift towards studying SVs at the population level. The most prominent and early example of this was the 1000 Genomes Project, representing different ethnic groups. Other projects include National Institutes of Health (NIH)-sponsored programs such as the National Human Genome Research Institute (NHGRI)’s Centers for Common Disease Genomics (CCDG), and National Heart, Lung, and Blood Institute (NHLBI)’s Trans-Omics for Precision Medicine (TOPMed) program that includes ~155,000 participants sequenced. With data from newer, high-throughput sequencing platforms, investigators are able to capture the genetic variability at a genomic scale of SVs across more geographically and genetically distinct populations. The Icelandic project studied 1,817 Icelandic individuals, including 369 trios (mother+father+child). Accordingly, many tools exist to study SVs in cohorts (e.g. svtools and the StructURal Variant majorIty VOtE (SURVIVOR)).

Our understanding of genomic variability in humans is tied to the technologies used to study those genomes, typically involving DNA sequencing. Short-read (SR) DNA sequencing (resulting in reads usually less than 150 bp in length) has been the most common way to evaluate DNA samples directly, and RNA samples indirectly after cDNA conversion. When short reads are mapped to reference genomes, they either map entirely or partially. Partial mapping can be accomplished by locally aligning part of the read and dropping the rest. This is known as soft-clipping. However, short reads do not align well to large variants when there is a significant gap between the last anchored reference position of the read and position of the soft-clipped portion (e.g. a variant with repetitive sequences longer than the read itself). Therefore, the aligner may not produce a global alignment between the read and reference, either choosing not to map the entire read or leaving the soft-clipped portion unmapped.

The linear reference genome (LRG), which is based on a linear coordinate system, is a common way to represent variability within a genome. While this may be efficient for applications using one or few genomes, more complex applications using multiple LRGs do not scale well. For example, when mapping reads from RNA sequencing (RNA-seq) to a single genome, the use of an LRG requires a baseline amount of compute resources. However, if mapping to multiple linear reference genomes, the task scales roughly linearly, (increasing by a factor n, where n is the number of LRGs to which data can be mapped).

By contrast, graph genomes can explicitly encode many alternate paths. Variants can be represented in this data structure, and reads can then be mapped exactly to both a reference path and variant sequences. Graphs have been shown to reduce reference bias and improve read mappings to variants. Variants can also be quickly genotyped from such structures. However, adding more variations can rapidly increase the complexity of the graph. In order to make the problem computationally tractable, one must carefully choose which genomic regions to include. Tools for working with graphs are still nascent, and there are few open-source graph implementations that enable mapping directly to SVs (e.g. vg and Paragraph). Graphs are a non-linear alternative to represent genomes; different paths are presented in the graph data structure. Consequently, 1.) they can decrease reference bias and improve read mappings to variants; and 2.) variants can be efficiently genotyped. Adding more genomes to the graph enhances it, but increases its complexity. To decrease complexity and make this problem computationally tractable, the added regions must be chosen carefully.

Regardless of how drastically human individuals may differ in features such as their physical characteristics and behaviors, the human genomes between any two people are actually relatively well-conserved across individuals. The completion of de novo individual genomes is costly, and population-scale comparisons impossible with current technology infrastructure. There are a small number of individuals represented in the reference human genome, while in comparison any given human individual has on average ~4 million single-nucleotide polymorphisms (SNPs) and ~2,500 CNVs. Consequently, there is a burgeoning awareness that the current standard reference genome assemblies do not include available human variation data. Moreover, there is an urgent need for improved tools, which more precisely represent rare genotype(s) of individuals bearing haplotype(s) absent from the human genome. This is particularly relevant for genomic variants leading to clinical phenotypes such as in Mendelian diseases and genomic disorders where rare and ultra-rare variants have a prominent role. A
reliable and scalable solution to this comprises developing a more comprehensive reference genome data structure that represents variations that exist in a given population, such as a graph genome\(^4\). The variants in a given cohort’s genomic sequences are represented as independent walks along the graph, allowing it to represent the reference cohort.

Overall, the quality assessment, representation, and annotation of SVs across multiple disciplines (e.g., whole-genome sequencing (WGS) and metagenomic sequencing) remain challenging. Thus, our SV codeathon groups focused on seven topics, which led to development of seven new computational methods. These topics included: 1.) quality assessment of population-scale VCF files; 2.) metagenomic assembly quality assessment; 3.) CNV detection and identification of de novo SVs using long-reads; 4.) fast genome graph generation; 5.) SV annotation; 6.) graph making and graph-based read-mapping with GPUs; and 7.) CNV detection quality control. Here we describe our progress, giving details about our tools’ implementations and applications to foster continuous development beyond the current scope of the tools as they were at the end of this codeathon. All methods are open source licensed, and have been made available on GitHub (https://github.com/NCBI-Codeathon/).

**Methods**

**Implementation**

**Clouseau: rapid quality assessment of population-scale VCF files.** As we progress from comparing SVs between a sample and its corresponding reference (capturing differences between a given sample and its reference genome) to large cohort genomic datasets (variants across multiple hundreds to thousands of samples), large variant call format (VCF) files are being generated. Generating these large VCF files often involves customized scripts or analysis methods leaving the possibility of human or other runtime errors. For example, undetected errors such as incomplete files or artificially missing data might be mistaken for real biological phenomena (deletions/truncations). Leaving these unchecked may lead to erroneous conclusions. We developed Clouseau to address these challenges. Clouseau is a command line tool allowing users to rapidly validate VCF file formatting, generate multiple QC statistics, providing rapid QC insights from the input VCF file.

**MasQ: Metagenomic assemblies-focused Quality assessment.**

While metagenomic assemblies have significantly improved since the early days of the Human Microbiome Project (HMP), intragenomic and intergenomic repetitive sequences remain as confounders. Individual reads spanning microbial strains (either via long-read technology or by generating synthetic long-reads) may be pieced together to resolve variation within a given microbial community. However, this process is imperfect. A major concern is that detected structural variants could actually result from misassembled genomic data instead of actual strain-specific variation. The goal of this project is to identify errors in any metagenomic assembly based on both short and long read mapping, in the hope of eliminating some of the uncertainty and error in metagenomic studies. Examples of errors to detect include falsely called (false positive) inversions, chimeras (translocation), INDELs (less than 50 bases long), and replacements (large substitutions). We created a containerized quality control pipeline called MasQ to locate, classify, and rectify errors in metagenomic assemblies. Out of concern for the integrity of current and future metagenomic studies from short-read and long-read genome sequencing data, we found that the quality control of metagenomic assemblies could be substantially improved by using a combination of sequence alignment tools and VCF file outputs from SV callers such as Sniffles v1.0.8\(^3\) and Manta v1.5.0\(^4\). The resulting SV call sets are subsequently compared and merged using packages like Truvari v0.1.2018.08.1 and SURVIVOR\(^5\). The current version of MasQ neatly packages this workflow and integrates it with novel correction and validation steps to fix any erroneous contigs found.

**DeNovoSV: CNV detection and identification of de novo SV using long reads.** We developed a pipeline to identify de novo structural variants (SVs) from long-read (LR) sequencing data collected from trios (proband + parents). As described below, in a pilot study, we analyzed data from an individual who carries multiple de novo CNVs, initially identified by array-based comparative genomic hybridization (aCGH). Prior to this work, we lacked an integrated bioinformatics pipeline to identify high-confidence, de novo structural variants called from LR DNA sequencing (e.g., Oxford Nanopore Technology - ONT). Accordingly, to select further SV calls for orthogonal validation, we merged de novo LR SV calls with de novo SV calls independently identified by short-read (SR) DNA sequencing (i.e. PCR-free paired-end Illumina DNA sequencing, 150 paired-end reads, 40x depth of coverage) in this trio. DeNovoSV facilitates identification and prioritization of high-quality de novo SV calls that can be further validated by using targeted orthogonal methodologies such as Sanger sequencing or droplet digital PCR (ddPCR).

**SWIGG (SWIFT Genomes in a Graph): fast genome graph generation.** There is a growing consensus across genomics that linear genome representation is suboptimal for representing variants across populations. While graph genomes have been steadily gaining popularity, many challenges remain (complexity, visualization, etc.). In this project, we developed a heuristic approach that quickly generates graphs and represents genomes in an efficient and succinct way. We created a simple algorithm and tool to build a graphical model that captures variability in the genomes at multiple scales. Moreover, there are regions across the human genome that are conserved among species while bearing modest amounts of variability that are suitable for understanding relationships of genome structure among individuals and/or organisms. We used a k-mer approach to create a sparse representation of such regions (anchors) at a large scale so as to allow visualizing the entire genome easily. These “anchored” graphs can then be further iteratively improved to include local sequence differences, and in turn, to help with genotyping existing variants and identifying new variants in new genomes.

**ASAP (Automated Structural Variation Annotation Pipeline).**

ASAP is an automated and robust pipeline to annotate structural variations. The pipeline integrates annotations including allele frequency, collocated gene, functional features, domains,
regulatory elements, and transcription levels. To facilitate further development, during stage one, we pull the main program, AnnotSV v2.2.31 from its remote source, as we expect this step to change relatively infrequently and the program itself is large. In stage two, based on the previously-built base image, we pull in its dependencies including BEDTools v2.29.0^1 and build the docker. The workflow is as follows: 1.) User provides a VCF containing SV as input into AnnotSV 2.) AnnotSV annotates the variations and outputs a tab-delimited file 3.) the R script “postprocess.R” is used to process the TSV file generated by AnnotSV and extracts essential annotations like ranking score. AnnotSV has many default annotation sources (https://lbgi.fr/AnnotSV/), and can also accept user-provided annotations as input. The output of this pipeline used the default annotation.

Super-minityper: graph making and graph-based read-mapping with GPUs in the cloud. We present a set of cloud-based workflows, composed mostly of preexisting and optimized tools, to 1.) construct graphs containing structural variants and 2.) map reads to these graphs. Our workflows allow users to make arbitrary SV calls, construct a graph, and map reads to these graphs. This workflow prioritizes ease-of-use and speed, accepting common input formats and returning results in minutes on commodity cloud virtual machines. Our approach is an example of what can be done now, and is generalizable to newer graph tools.

ScanCNV. Many current medical genomic studies still rely on CNV calling to identify de novo events that could have led to a certain disease phenotype. Nevertheless, a common problem for CNV detection is a high rate of false positives due to multiple biases in leading short-read sequencing technologies (e.g. GC bias, repetitiveness and general unevenness of the sequence data). To identify false positives and thus improve the reliability of CNV calling pipelines, we designed ScanCNV, which includes multiple QC steps. These QC steps currently include fastQc v0.11.9, XYAlign v1.1.6^5, and Plink v2.0^4 where all the information is currently assembled and vetted within ScanCNV. Future work is still required to automate the process and fully leverage the obtained QC results.

Operation
All methods were tested on DNA-Nexus instances azure:mem1_ssd1-x, 4, cores, 8G ram or azure:mem1_ssd1_x16, 16 cores, 32G ram.

Clouseau. Clouseau requires python 3.5 or above. Users supply optional parameters such as the expected maximum distance between variations to identify missing entries. Importantly, the input VCF file needs to be sorted by coordinates. The Clouseau pipeline reads in a VCF file and performs a sample-level basic statistic carried out to ensure that the VCF file is complete and was not truncated during the VCF generation. Next, the VCF is parsed and sample-level QC is carried out. This consists of checking the number of samples in the VCF, the number of chromosomes, the names of all chromosomes/contigs, the distribution and number of variants (single nucleotide variants, insertions, deletions, structural variants) in each chromosome/contig for all samples, and the coordinates for the start and end of each variant for each chromosome and sample. Clouseau further tries to identify missing entries based on long stretches of no variations, which might represent file errors (as occur in incomplete files) or real biological phenomena (as in deletion/truncation). The full workflow is shown in Figure 1.

MasQ. The MasQ pipeline comprises four stages for metagenomic study quality control and validation: assembly, classification, correction, and validation. Our pipeline takes read files in FASTQ format as input, and classifies possible assembly errors as inversions, INDELs, substitutions, chimeras, or N/A, and then compares them with the make-up of the original assembly.

The purpose of the assembly stage is to do basic quality control on the sequencing data, generate a mock assembly from alignments of the read data, and generate VCF files informing about unidentified errors. Short-reads are pre-processed and visually assessed for quality, using FastQC. Then, all the reads are aligned to the MegaHit assembly using BWA-MEM v0.7.4^19. The alignment results are processed through Manta v1.5.0^12, a structural variant caller, to produce a VCF file containing the detected errors. Optionally, if long reads are also available for the same sequencing sample, they are passed through a similar pipeline. The long reads are inspected for quality using NanoporeQC. Minimap2 v2.8^40 is used to align the long reads instead of BWA, then the long-read alignments are provided to Sniffles v1.0.8^15 to call SVs, condensing this information into a VCF file.

In the classification stage, VCF files for the short and long reads are compared to each other using Truvari and a merged VCF file is generated with SV data. Short-read and long-read data each present their own sets of challenges while building metagenomic assemblies, so comparison of the Truvari results from short and long sequencing reads of the same samples decreases false positive results.

The correction stage takes the regions of assembly error ascertained from the classification stage and performs the necessary changes to make a more accurate metagenomic assembly. From here, the validation stage uses sequence aligners such as Minimap2 and BWA to compare the edited assembly to the original inputs, looking to confirm an increase in the percentage of mapped reads in the corrected assembly, which would show the success of the pipeline in locating and correcting assembly errors.

MasQ is implemented using Docker and is freely available on DockerHub. MasQ can also be run online using DNAxenuse. All relevant parameters, as well as python scripts for the correction and validation steps, can be found on the GitHub repository. The MasQ pipeline is shown below in Figure 2.

DeNovoSV. DeNovoSV takes input VCF files produced by long-read SV callers such as Sniffles v1.0.11^23 and short-read SV callers such as Lumpy v0.2.13^23, Delly v0.8.2^23, and Manta
Clouseau Workflow. Clouseau starts with a VCF/pVCF/gVCF (variant call format) file that is assessed to ensure the completeness of the previous run. Furthermore, Clouseau assesses the overall statistics to give insights into the per sample quality control (QC). VCF, variant call format. pVCF, project variant call format. gVCF, genome variant call format. QC, quality control.

v1.6.0\textsuperscript{19}. We developed a custom shell script to remove calls that genotyped as a homozygous reference (GT=0/0), with read support less than 5 and aligned to haplotype chromosome, unplaced or unlocalized contigs (GL, KI, etc). Second, the filtered outputs are combined and compared by the pipeline using SURVIVOR v1.0.6\textsuperscript{19} merge allowing a maximum distance of 1000 bp measured pairwise from the start and end breakpoints of each SV, respectively; SVs classified as the same type; and SVs larger than 30 bp. Lastly, CNV calls are performed independently, using ONT data calculated on the resulting alignments using mosdepth v0.2.3\textsuperscript{43} with the following parameter set “-F 3588 -Q 1”. This calculates the bp coverage in 100 kb bins and includes only primary and supplemental alignments. Normalization of the coverage signal is performed based on a custom script, which generates bedgraph data as output. Each individual bedgraph file pertaining to proband and...
Figure 2. MasQ pipeline to assess the quality of metagenomic assemblies. Over multiple steps MasQ utilizes the short and long reads available to assess the quality of the before obtained metagenomic assembly.
parents is processed by dividing the 100 bp bins scores by the median of the coverage windows. Using this normalization, the majority of the genome shows a score of 1 (i.e. similar coverage between the samples) while a CNV on the proband results in a score of 2 for a duplication. We load these files into JBrowse with the multibigwig plugin for visualization, and to facilitate downstream analyses of the putative CNV calls. The resulting output is a list of high-confidence de novo SVs. The overall workflow of DeNovoSV is shown in Figure 3.

**SWIGG.** The SWIGG (SWIft Graph in a Genome) pipeline can take multiple FASTA sequences as input and outputs a graph that would be a sparse representation for a multiple sequence alignment which can be visualized for larger scale differences. First, the input FASTA sequence is processed to identify appropriate $k$-mers. Appropriate $k$-mers are those that are not repetitive within and across all FASTA sequences. The thresholds for appropriate $k$-mers can be modified using the script arguments. $K$-mers are then sorted based on their positions and collapsed into a node-edge list. This node-edge list can be visualized using a graph visualizer such as Gephi.

Our motivation to implement SWIGG (Figure 4) is that the human genome essentially contains two types of regions, those that are quite stable, and those that are hypervariable. For the stable regions, short reads can be either hashed as exact matches to these relatively static regions, or mapped with very small bubbles. Nevertheless, stable regions are interspersed by variable and hypervariable regions (e.g. Kir, MSC). We used $k$-mers to provide a backbone for these complex regions.

Likewise, we have also looked at those regions that are very unstable for mapping in the reference genome to extrapolate to graphs. We used 64 of such regions located on human chromosome 6. They are graphically depicted using the NCBI Genome Data Viewer in Figure 5. SWIGG is implemented in python 3 and publicly available both on GitHub.

**ASAP.** The user inputs an SV VCF file(s) produced from any SV caller, and the SV file is then ranked and annotated using AnnotSV and produces a tab-delimited file which will be processed using contemporary versions of R. The complete pipeline can be executed automatically as a single WDL script on GitHub. Below is the workflow Figure 6.

**Results/Use cases Clouseau**

Before proceeding with SV analysis, the user needs to have insight into the SV file to avoid issues with downstream analyses. Clouseau facilitates this best-practice quality control check by analyzing an input SV file and providing detailed information about its content. Clouseau takes as input an SV VCF file, and returns the number of samples in the file, chromosomes analyzed, distribution of SVs in all samples, and detailed information about each sample in the file. The output is a list of files for all the samples and each sample in the input file. Clouseau was benchmarked using the SV file from the 1000 Genomes Project. The data set consists of 2,505 samples and consumed 1 minute (wall clock time) to analyze 10,000 variant lines, Figure 7 shows an example of the output results per sample.

![Figure 3. DeNovoSV workflow.](image-url)

The schematic shows the required inputs and final outputs, along with intermediate steps. DeNovoSV starts with already aligned reads (BAM) from long reads and short reads to filter candidate SVs based on a trio to identify de novo SVs in the proband.
Figure 4. SWIGG workflow diagram shows each step; extracting k-mers, finding positional information, and building graphs.

Figure 5. NCBI Genome Data Viewer showing complex regions in human chromosome 6.
MasQ
MasQ uses short- and long-read sequencing data as input. It outputs VCF files with the locations of structural variants, as well as a classification of assembly errors. Among the two existing methods of assembly validation, reference-based or de novo assembled, we chose to develop a de novo validation pipeline, and benchmarked it using two widely-used datasets, Zymo Microbial Community Standards and Shakya. The Zymo dataset consisted of both Illumina pair-ended short reads and Oxford Nanopore long reads, while the Shakya dataset was made up of pair-ended short reads. The current version of MasQ is publicly available as a Docker container that combines existing tools for assembly, creation and comparison. It includes a Python classifier and correction to identify and fix assembly errors, and is open source on GitHub.

For the Shakya data, metaSPAdes assembly, metaQUAST reports that there are 328 misassemblies and the total misassembled contigs length is 12,555,565 base pairs. For Shakya, MEGAHIT assembly, metaQUAST reports that there are 673 misassemblies and the total misassembled contigs length is 20,244,280 base pairs. For both of the finished assemblies, metaQUAST reports more than hundreds of misassemblies and the total length is more than tens of million base pairs. Also, on the Shakya dataset, according to metaQUAST, the performance of metaSPAdes is slightly better than that of the MEGAHIT. For the MasQ result of both assemblies, both Manta and Sniffles call hundreds of mis-assemblies in two assemblies, which are about the same size of the result outputted by metaQUAST. From the long-read perspective, Sniffles calls less insertions and duplications on the metaSPAdes assembly. Even though metaSPAdes required significantly more time to finish assembling, the extension of edges through repetitive regions and careful decision to remove low coverage edges contributed to a lower insertion and duplication rate.

DeNovoSV
In a prototype trial of DeNovoSV, we investigated available genomics data obtained from a trio of samples (i.e. father, mother and proband), with a particular focus on identification
of de novo SVs in the proband. Available data included ONT long-read sequences, Illumina short-read WGS from the trio and aCGH. Data from the latter were used as a positive control for known de novo CNVs for the purpose of developing this pipeline. Working with ONT long-read sequence data, we used the NGMLR\textsuperscript{32} aligner to map the data against the human genome reference assembly (hg38). Sniffles\textsuperscript{32} used BAM files from the previous step to identify candidate de novo SVs. Read depths of SVs were calculated using mosdepth\textsuperscript{43}. Sniffles outputs were used as source dataset for VCF files which were filtered using our DeNovoSV custom script to remove SVs for which supporting reads were limited in number (RE < 5); homozygous reference SVs; SV branching from autosomes to decoy; and contigs identified with GL numbers, which represent unlocalized contigs. De novo SVs in the child were defined by merging and comparing with parents in the trio using SURVIVOR\textsuperscript{19}. After filtering for de novo SVs in the child using the DeNovoSV pipeline, we compared these de novo SVs and CNV candidates with results extracted from CGH array data.

We used the DeNovoSV pipeline to prioritize high-confidence, de novo SVs called from long-read (LR) technology. Starting with about 3 million SV calls identified by Sniffles for each individual, the de novo LR pipeline detected 4,509 high-quality SV calls. We further enriched for potentially true-positive calls by merging the long-read calls with SVs calls independently identified with short-read sequencing in this trio (N = 2599 high-quality SV calls). After merging the calls from short-read sequencing to filter for consensus calls, we obtained a list of 67 high-quality SV calls (Figure 8 and Figure 9).

Eight de novo duplications in the proband were detected using normalized read depth coverage data from mosdepth and visualized in JBrowse (Figure 10).

**SWIGG**

In SWIGG we created a simple algorithm and tool to build genome graphs, which is suitable for understanding relationships of genome structure among individuals and/or organisms. Our approach captured variations in a hierarchical way. The idea was to create a sparse representation of large-scale differences (anchors) so as to allow visualizing the entire genome in a succinct way. These “anchored” graphs can then be further iteratively improved to include local sequence differences, and in turn, facilitate genotyping existing variants and identifying new variants in new genomes. We tested SWIGG by creating a graph of the human MHC region (4.5Mb in size) using 128-mers in less than three minutes on seven alternative haplotypes. To evaluate our approach at a smaller scale, we tried building a graph for 10 HIV viral genome (each ~10kb) using 10-mers (see SWIGG Github repository).

**ASAP**

ASAP was benchmarked using 53 non-diseased tissue sites across nearly 1000 individuals. GTEx RNA-seq data (retrieved from RNA-seq gene read counts GCT file https://gtexportal.org/home/datasets) and GTEx Metadata (manually curated from https://www.ebi.ac.uk/arrayexpress/files/E-MTAB-5214/E-MTAB-5214.sdrf.txt) from NA12878 Germline Whole Genome were used. V2 Chromium Genome dataset by Long Ranger 2.2.1 downloaded WGS v2 deletions VCF file from

---

**Figure 8.** Number of de novo high-quality SV calls detected using ONT LR sequences, Illumina short-read sequences, and filtered consensus calls detected by both methods.
Figure 9. Categories of consensus de novo SV calls from DeNovoSV. Del: deletion; Dup: duplication; Inv: inversion; INS: insertion; TRA: translocation; UNK: unknown.

Figure 10. DeNovoSV outputs. JBrowse screenshots displaying normalized Oxford Nanopore read coverage for four out of eight de novo duplications spanning 900 kb to 1 Mb genomic segments from Chromosomes 4, 5, 6 and 10. Parent 1 is represented in a blue line, parent 2 in a light blue line and proband in a dark red line. Red rectangles denote duplications.

https://support.10xgenomics.com/genome-exome/datasets/2.2.1/NA12878_WGS_v2. Messenger RNA features were extracted from NCBI latest RefSeq Annotation\(^3\) on GRCh38 using instructions provided here https://www.ncbi.nlm.nih.gov/refseq/functionalelements/. The workflow for running benchmarking and results are shown in Figure 12 below.

Conclusions and next steps

Clouseau

The current version of Clouseau sought to validate VCF integrity and to provide informative statistics. Clouseau can read an SV file and return relevant reports. A next step would be to implement modules to accept SNVs, more than one file as
an input, visualize the statistics results using publicly available modules, and reimplement the tool to guarantee better performance.

MasQ

MasQ currently provides a smooth, convenient workflow to locate genome assembly errors and a solid foundation for a machine learning model to classify assembly errors. In the future, MasQ would benefit from further testing with simulated data, larger datasets, and manual annotation of training data for a more sophisticated classifier. We intend to treat this problem as a multiclass text classification problem, which could be successfully handled using an RNN, LSTM, SVM, or HAN framework (Figure 13). Additionally, the outputs from the
classification model must be visualized with histograms of the assembly error types, as well as with visual indications of where the errors are located within the assembly. The assembly errors could possibly be organized so that each type of error could be queried within the given assembly and made easily accessible to users.

**DeNovoSV**  
In a pilot study to analyze *de novo* SVs identified in a proband from a genetic trio, we evaluated capabilities of DeNovoSV to confirm those SVs and potentially to discover additional SVs. Using the DenovoSV pipeline with parameters described in Figure 3, all eight genome-wide chromosomal segments with known *de novo* duplications identified by aCGH in the proband were also identified by short- and long-reads. In addition, 59 additional potential *de novo* SVs were identified, which need further validation. Independently, the read depth information generated by mosdepth using LR data from ONT also successfully identified all eight genomic duplications, suggesting that analysis of LR data alone can be valuable for *de novo* CNV detection.

**SWIGG**  
Once simple and variable regions are solved for graphs, including those with many structural variants and those with complex regions across a number of populations and diseases, then we will be able to phase new genomes automatically. Thus, we will be able to observe traversals of previously unmanageable paths through graphs made from both long and short read data as shown in Figure 14.

Now that we are developing tools for generating genome graphs quickly and efficiently, it is desirable to find optimal

![Random Forest to Classify Assembly Errors](image1.png)

**Figure 13.** MasQ future work using advanced machine learning approaches to identify misassemblies.

![Comparison between simple and complex genomic regions showing structural variant graph islands (bubbles) and linear portions.](image2.png)

**Figure 14.** Comparison between simple and complex genomic regions showing structural variant graph islands (bubbles) and linear portions.
parameters (e.g., size of k-mer and occurrence of a k-mer) to determine whether a given graph should be retained or not. This remains challenging because there is no simple way to evaluate genome graphs. Numbers of nodes and edges can be starting points. Comprehensive investigations of potential factors are needed, such as determining to what extent known features are recapitulated in the graphs.

Data availability
Source data
Clouseau - SV data from 2,505 samples from the 1000 Genomes Project (http://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/).


DeNovoSV - For the development of the pipeline we used in-house control data from patients.

SWIGG - For data to construct the graph, we used the alternative sequences of the MHC region given in the latest reference Human Genome (HGR38). These were downloaded using NCBI accession numbers GL000250, GL000251, GL000253, GL000254, GL000255 and GL000256. HIV sequences were obtained from Los Alamos National Laboratories\textsuperscript{3}, \textsuperscript{8}, [with specific accession numbers] Human immunodeficiency virus type 1 (HXB2), complete genome; HIV1/HTLV-III/LAV reference genome [K03455], HIV-1 isolate 2106 from Democratic Republic of the Congo, complete genome [MH705158], HIV-1 isolate 50 from Democratic Republic of the Congo, complete genome [MH705161], HIV-1 isolate 70641 from Democratic Republic of the Congo, complete genome (MH705151), HIV-1 isolate P4039 from Democratic Republic of the Congo, complete genome [MH705157], HIV-1 isolate PBS6126 from Democratic Republic of the Congo, complete genome [MH705133], HIV-1 isolate HIV/CH/BID-V3538/2003 from Switzerland, partial genome [Q403028], Human immunodeficiency virus 1 proviral env gene for envelope protein, isolate 98CM.MP1014 [AM279354], HIV-1 isolate 09NG010499 from Nigeria gag protein (gag) gene, complete cds; pol protein (pol) gene, partial cds; and vif protein (vif), vpr protein (vpr), tat protein (tat), rev protein (rev), vpu protein (vpu), envelope glycoprotein (env), and nef protein (nef) genes, complete cds. [XX389622].

ASAP - Data sources and Data Description: 53 non-diseased tissue sites across nearly 1000 individuals: GTEx RNASeq Data: Downloaded RNA-Seq gene read counts GCT file from https://gtexportal.org/home/datasets. GTEx Metadata: manually curated from https://www.ncbi.nlm.nih.gov/geo/query/CEL?acc=GPL1350. Whole Genome v2 Dataset by Long Ranger 2.2.1: Downloaded WGS v2 dels VCF file from https://support.10genomics.com/genome-exome/datasets/2.2.1/NA12878_WGS_v2. RefSeq Features: mRNA features are extracted from NCBI latest RefSeq Annotation on GRCh38 using instructions provided here: https://www.ncbi.nlm.nih.gov/refsseq/functionalelements/.

ScanCNV - Simulated CNV 6 samples (males and females) FASTQ data (truth set: “In silico CNVs - Sheet1.csv”), and 1000 Genomes 10 random samples (males and females) mapped data (in file names) in GitHub repository (see below).

Software availability
Clouseau is available from Github: https://github.com/NCBI-Codeathons/Clouseau.
Archived source code at time of publication: https://doi.org/10.5281/zenodo.3950421\textsuperscript{3}
License: MIT

MasQ is freely available from Github: https://github.com/NCBI-Codeathons/MASQ.
Archived source code at time of publication: https://doi.org/10.5281/zenodo.3950441\textsuperscript{4}
License: MIT
DeNovoSV is freely available from GitHub: https://github.com/NCBI-Codeathons/DeNovoSV.
Archived source code at time of publication: https://doi.org/10.5281/zenodo.3950439
License: MIT

SWIGG is freely available from GitHub: https://github.com/NCBI-Codeathons/SWIGG.git.
DockerHub: ncbicodeathons/swigg.
Archived source code at time of publication: https://doi.org/10.5281/zenodo.3950425
License: MIT

ASAP is freely available from GitHub: https://github.com/NCBI-Codeathons/ASAP.
Archived source code at time of publication: https://doi.org/10.5281/zenodo.3950444
License: MIT

Super-minityper is freely available from GitHub: https://github.com/NCBI-Codeathons/super-minityper.
Archived source code at time of publication: https://doi.org/10.5281/zenodo.3977283
License: MIT

Acknowledgements

We would like to thank the administrative staff of the Human Genome Sequencing Center at Baylor College of Medicine (especially Meagun Elizabeth Sam and Chelette Darlene Gaskin) who helped to organize and run the event. We thank DNAnexus Inc. for sponsoring cloud computing resources. We thank Drs. Pengfei Liu and James R. Lupski for generously providing data from their Multiple CNVs project for the DenovoSV pipeline development and Oxford Nanopore Technology team for helping with data generation and data analysis. We thank Dr. Nick Matinayan for helpful suggestions during manuscript review.

References

Open Peer Review

Current Peer Review Status: ?

Version 1

Reviewer Report 12 November 2020

https://doi.org/10.5256/f1000research.26233.r74008

© 2020 Albert I. This is an open access peer review report distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Istvan Albert
Department of Biochemistry and Molecular Biology, The Pennsylvania State University, University Park, USA

This paper and other similar works fulfill an important need of demonstrating cutting edge methodologies, recommending new practices, and educating the broader community on what may be considered state of the art in a given subfield of bioinformatics.

That being said I wish the authors did a better job at organizing and distributing the code. The current practice of creating a separate disconnected repository for every single (sometimes exploratory) project feels verbose, counterproductive and long term misses the whole point of sharing code.

To make a better point of this criticism I took the time to rework the repository for this paper in what I think is far superior approach. See it here:

https://github.com/ialbert/Structural-Variation-Baylor-College-Codeathon-2019

The important difference in the structure I recommend is there is that more information should be shared right there on Github to avoid the constant referral back to the paper, chasing down links (by the way the links in the abstract do not match the text!) and scrolling the paper back and forth.

There should be a single entry point for each Codeathon that contains all the code developed during the event. During the events, the speed of development is of critical importance, thus it is very unlikely that the product would be a package that is worth maintaining separately. In addition, a lot of value can be gained from seeing what someone actually did during the Codeathon.

Copying over the code to a new project, if that is worthy would be a trivial exercise. It helps no one that from now on there will be an abandoned repository called Clouseau on Github that hangs out there with no context.
I would strongly urge NCBI in general, and these authors in particular, to start sharing the results of Codeathons not as a random list of seemingly unrelated packages but as complete, standalone event snapshots that are worth visiting and stand the test of time (as I show in my repository). The current organization neither scales nor allows Codeathons to reach their full potential.

As for the code I noted several shortcomings, all summarized as a lack of educational focus. Education starts with not just telling someone what to do but also showing how to do it. Each project developed during the Codeathon should have a simple example (with a simple data published there as well) that shows the full usage.

1. Closeau: needs a simple example run shown.
2. ASAP: there is a title header saying Execution Example but does not contain the example.
3. DenovoSNV: please show an example run.
4. MASQ: has run code but please do not hardcode the full link to it, instead use relative links otherwise cloned repositories will be incorrect.
5. SCANCVN: please compress your data files! In my repository I compressed the VCF and BED files and that dropped the repository size from 340MB to 40 MB! Makes an enormous difference. Also please do not use an IMAGE to demonstrate numbers or content in a shell. Just copy over the text into a markdown codeblock. There is a usage example, so that is covered.
6. SWIGG - is done nicely and it is how one imagines a complete report. It is informative, uses images as Mother Nature intended ;-) , has usage instructions. Well done.

Overall I think the paper is a valuable addition and a good fit for the Hackathons series if the authors take time to make their code more usable and meaningful to the larger community.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Partly

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Partly

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Partly

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

**Competing Interests:** No competing interests were disclosed.

**Reviewer Expertise:** Bioinformatics, Software development

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

---

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com