Master´s Thesis in Bioinformatics

SVenX: A highly parallelized pipeline for structural variation detection using linked read whole genome sequencing data

Vanja Börjesson[1,2]◕

1 Department of Molecular Medicine and Surgery, Karolinska Institutet, Solna, Stockholm, Sweden
2 Masters Program in Bioinformatics, Lund University, Lund, Sweden

va6745bo-s@student.lu.se
Friday 26th January, 2018

Abstract

Genomic rearrangements larger than 50 bp are called structural variants. As a group, they affect the phenotypic diversity among humans and have been associated with many human disorders including neurodevelopmental disorder and cancer. Recent advances in whole genome sequencing (WGS) technologies have made it possible to identify many more disease-causing genetic variants relevant in clinical diagnostics and sometimes affecting treatment. Numerous approaches have been proposed to detect structural variants, but to acquire and filter out the most significant information from the multitude of called variants in the sequencing data has shown to be a challenge. Another obstacle is the high computational cost of data analyses and difficulties in configuring and operating the softwares and databases. Here, we present SVenX, a highly automated and parallelized pipeline that analyzes and call structural variants using linked read WGS data. It performs variant calling using three different approaches, as well as annotation of variants and variant filtering. We also introduce a new tool, SVGenT, that reanalyzes the called structural variants by performing *de novo* assembly using the aligned reads at the identified breakpoint junctions. By comparing assembled contigs and analyzing the read coverage between the breakpoint junctions, SVGenT improves both variant and genotype classification and the breakpoint localization.

Introduction

It is well known that variations in the human genome are responsible for differences between individuals and specific traits. The size of variants vary from single nucleotide variants (SNV) to large chromosomal rearrangements, and are associated with both common and rare human diseases, including autism [1], schizophrenia [2] and other neuropsychiatric disorders [3]. Genomic rearrangements larger than 50 base pairs (bp) are commonly referred to as structural variants (SVs) and are the source of more base alterations compared to SNVs [4]. Different classes of SVs have been defined and include copy number variants (CNV) such as deletions (DEL) and duplications (DUP), and balanced SVs that do not effect the number of copies such as insertions (INS), inversions (INV) and translocations. To understand genetic disorders and complex traits, it is important to identify and genotype all different SV types and size [5] [6]. However, the majority of SVs in the human genome are not linked to any disorders, and several attempts to create databases of SVs have been made with the goal of define a gold standard. The database of Genomic Variants (DGV) [7] focuses on mapping SVs in healthy humans to provide curated information to the Genome Reference Consortium (GRC) [8], while the NCBIs database of SVs (dbVar) [9] also includes clinical data and data from other species. Despite the large number of human genomes being sequenced and many databases being built, there are still challenges in correctly calling and interpreting SVs.

The challenges in the detection and accurate prediction of SVs includes short-read sequencing technologies, an accurate reference genome, and optimal computational tools to analyze these data. No sequencing method can read the whole genome from start to end, and several different methods have been developed to re-sequence the human genome by fragmenting the DNA, sequencing and finally aligning reads to a reference genome (currently hg19). Despite several attempts of finding SVs with high reliability using short-read sequencing data, the methods struggle within repetitive regions where short-reads often are mapped incorrectly. Furthermore, SV discovery depends on information both in and between reads. By applying the 10x Genomics Chromium™ barcode strategy with Illumina sequencing to link reads into contigs, some of these obstacles may be overcome [10] [11] [12].

The variant calling format (VCF) was developed by the 1000 genomes project [13] and may be used both for the annotation of SVs as well as other DNA polymorphisms. The VCF includes both start and end position of the specific variant, the reference allele, the called non-reference allele, the quality score, and additional information such as genotype and filtering information. The filtering information is defined in the heading, and can include a quality score or number of samples with data. The filter status in the file is set to PASS if the position has passed all filters, and otherwise it is set to FAIL. Available computational tools are good at detecting different types of variants with different precision [14], and therefore WGS data is often analyzed using several callers to identify different types of SVs. This means that the generated VCF files often contain predictions from multiple tools where the information about SV classification, breakpoints and genotypes can differ significantly. Today's software lack in their ability of genotyping with high precision, and some of the SVs discovered are not genotyped at all.

To improve the analysis of genomic SVs, we present a new highly parallelized pipeline, SVenX (Structural Variants pipeline for tENX-data) and a new genotyper SVGenT. SVenX takes one or multiple samples of linked-read sequencing data and generates a VCF file of predicted SVs. Three variant caller softwares are included (1) Long Ranger WGS, (2) CNVnator ([15]) and (3) TIDDIT ([16]). Long Ranger WGS was developed by 10x-genomics and is a pipeline that performs variant calling, de-duplication and filtering, it also performs alignment by mapping the reads to a reference genome. CNVnator is a variant caller that uses read depth information to call CNVs, and TIDDIT is an in-house developed high-performance variant caller using discordant pair and supplementary alignment (SA) to discover and call variants [16]. Two softwares for annotations are included in the pipeline (1) VEP ([17]) and (2) SVDB query ([18]). VEP is a variant effect predictor and adds information about location, possible effected genes or transcripts, if the variant effect a regulatory site, etc. SVDB query is a second annotation tool which in addition to VEP also give variant frequency information. As a last step in the pipeline, a new software SVGenT was developed to improve genotype prediction and SV classification. Finally, SVGenT predicts new breakpoints by *de novo* assembly of predicted SV-regions. It considers all SVs found in the VCF file, first by running *de novo* assembly on reads covering the SV region, and second by re-analyzing the read depths over the region, and returns an updated VCF file containing new SV predictions.

SVenX have been tested on a high performance cluster, analyzing 13 samples of human 10x Genomics WGS data obtained from blood samples as input. The pipeline produces annotated and filtered VCF files for all samples within a 5 day period. The final VCF file contains between 2000 to 3000 individual SV calls, and VCF files from each variant caller can be found in a separate sample specific folder. The new tool SVGenT was compared with three variant callers (TIDDIT, CNVnator and DELLY [19]) using two different datasets, one public large scale sequencing dataset (NA12878) from a Utah woman sequenced using several different techniques to obtain benchmark reference material [20], and one simulated dataset. The tool performs well in finding true SVs, and improves the SV classification. SVGenT also generates and returns important information regarding read coverage between and around the breakpoints, which could be useful in understanding more complex structural variations and rearrangements.

Materials and methods

SVenX: pipeline for detection of SVs using 10x Genomics data

SVenX is written in Nextflow [21], a domain-specific language (DSL) for data-driven computational pipelines, and Python and Bash. The pipeline takes one sample-folder of 10x Genomics FASTQ-files R1 (forward), R2 (reverse complementary) and I1 (index), or alternatively one folder containing multiple sample folders as input. The user also specifies which variant calling tools (TIDDIT and/or CNVnator) to use in the Pipeline, except for Long Ranger WGS, and also specifies if an annotation by VEP should be performed on small indels. The folders are checked for required FASTQ-files; R1, R2 and I1, before the user defined programs are launched. If several samples have been selected, SVenX will channel all samples for parallel execution.

Structural variant calling

In the default setting the 10x Genomics software Long Ranger WGS is the first variant caller to be launched. This program will call SVs and SNVs, and for each category generate two files, one barcoded and indexed BAM file and one phased and indexed VCF file. Long Ranger WGS is performed using 16 cpus. As soon as the Long Ranger WGS process is completed, the BAM file is copied and processed in parallel by TIDDIT and CNVnator for additional variant calling, and VEP is run to annotate small indels. TIDDIT is executed with the following settings: -i 20000, -r 100, -p 6, -q 10, meaning that the number of SAs have to be 100 at least to call a small SV, and 5 to call a large SV. It will generate a TAB file containing read coverage information for every 100th base (1 bin) in the sample, and a VCF file with the predicted SVs. CNVnator is executed with a bin size of 1000bp and generates a VCF file with the predicted CNVs. Finally, the VCF files containing SV calls are merged together using SVDB merge.

Annotation and filtering

First, the merged VCF file is annotated using VEP, and second by SVDB. The annotated VCF file is filtered by only selecting variants with a PASS in the VCF filter field. As a last step in the pipeline, the output files are sorted into sample specific folders (Fig 1).

A flowchart of SVenX is shown in Fig 2.

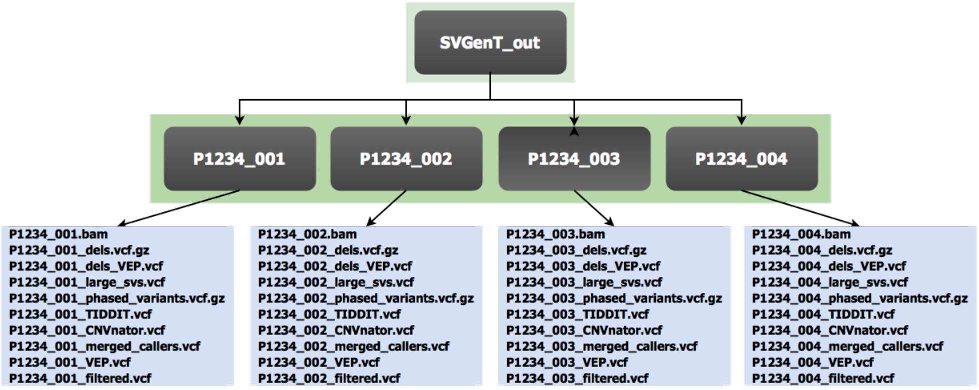

Fig 1. SVenX output; graph representation of sorted folders and files. SVenX saves the output files from each variant caller, annotation and filtering step in the pipeline. These files are then sorted into sample specific folders inside the SVGenT_out folder.

Evaluating the performance of SVenX

The performance of SVenX was evaluated by executing the pipeline with 13 samples of WGS data generated through linked read sequencing (10x Genomics) as input. The DNA was collected from human blood samples. The total performance of the pipeline as well as each component (Long Ranger WGS (using 16 cores), VEP for small indels, CNVnator and TIDDIT) was timed and the number of calls was calculated for each process and for the final VCF file. SVenX was executed on UPPMAX, a high performance cluster using Intel Xeon E5-2630 v3 processors.

SVGenT: new tool for genotype and breakpoint prediction

SVGenT requires a BAM file, an annotated VCF file and a TAB file generated by TIDDIT, with read depth information per 100 base-pairs, as input. SVGenT also require a SQLite database (SVGenT.db) containing mappability and GC-content information.

SVGenT database

In order for SVGenT to perform read depth normalization, the SQLite database SVGenT.db is required. The SVGenT database script is written in python, and requires the human reference genome hg19 [22] and mappability data in bigWig-format from the Centre for Genomic Regulation (CRG) [23]. The mappability data contains measures how often a kmer of size 100 base-pairs is aligned in the reference genome and gives a score between 0 and 1, calculated by dividing 1 by the number of alignments observed [24]. The mappability data are presented as a score for every position in the genome. The bigWig-file is converted to bedGraph format before it can be used as input in the SVGenT database script. SVGenT then calculates the average mappability for every 100 bp bin (position 1-100, 101-200,...,(n-99)-n) in the genome and the results are saved along with chromosome, start and end positions to the database SVGenT.db. The script will also calculate the GC-content of the same 100bp bins using the reference FASTA file. The GC-content information is added to SVGenT.db where chromosome and positions are equivalent.

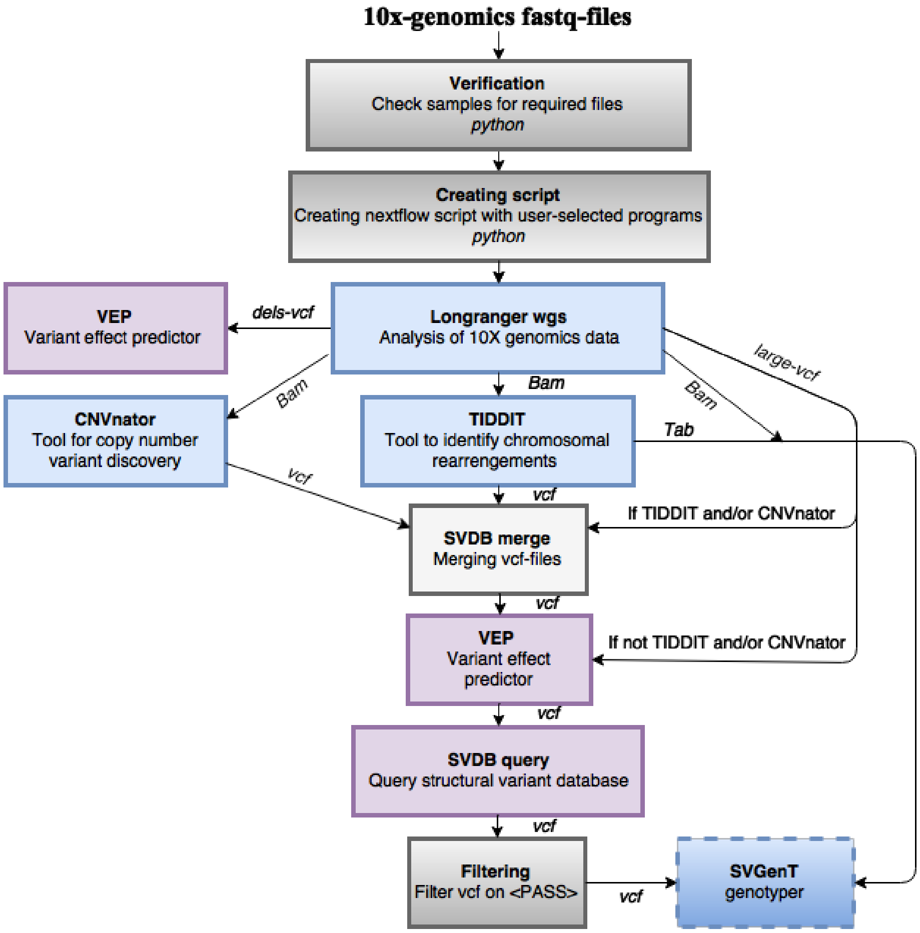
10x-genomics fastq-files
Verification
Check samples for required files
python
Creating script
Creating nextflow script with user-selected programs
python
VEP
Variant effect predictor
dels-vcf
Longranger wgs
Analysis of 10X genomics data
large-vcf
Bam
Bam
Bam
CNVnator
Tool for copy number variant discovery
TIDDIT
Tool to identify chromosomal rearrengements
Tab
vcf
vcf
If TIDDIT and/or CNVnator
SVDB merge
Merging vcf-files
vcf
VEP
Variant effect predictor
If not TIDDIT and/or CNVnator
vcf
SVDB query
Query structural variant database
vcf
Filtering
Filter vcf on <PASS>
vcf
SVGenT
genotyper

Fig 2. SVenX pipeline flowchart. The SVenX pipeline with the variant callers presented in light blue boxes, annotation tools presented in purple boxes and genotype tools presented in a darker blue box. SVenX starts by confirming that all 10x Genomics sequencing output files have been added correctly as input, then the Nextflow script is created based on user specified requests about which variant callers to run. The script always starts by executing Long Ranger WGS for de-duplication, filtering, alignment and variant calling. If the variant callers CNVnator and/or TIDDIT, and annotation using VEP for small indels were selected, these programs are executed in parallel. The variant callers save called SVs as VCF files, which are merged together before annotated using VEP and SVDB query. As a last step, the VCF file is filtered and only calls with a quality of PASS are retained.

Genotype, breakpoint prediction and SV-type classification using *de novo* assembly

SVGenT generates a sample specific SQLite database (not the same as SVGenT.db) containing chromosome, start position, end position and read coverage for every 100bp bins, by reading the TAB file containing read coverage information. The database is joined with SVGenT.db, adding GC-content and mappability information for every 100bp where both chromosome and genomic positions are identical in the two databases. From the VCF file, the SV breakpoint start and end positions are retrieved and a 2kb region for the called positions are calculated e.g. if the called SV is on chromosome 1 position 3451100 and end position 3454000, the region will be chromosome 1 position 3450100-3452100 and 3453000-3455000. Sequencing reads within the regions are extracted from the BAM file using SAMtools [25], and creates a separate BAM file for this SV. The created BAM file is then converted to a FASTA file using Bash. If the breakpoints in an SV are located within the same 2kb region, only reads from one region will be used to avoid duplicated reads (Fig 3). The region specific BAM file are converted to a FASTA format before executing *de novo* assembly. Two *de novo* assembly tools, ABYSS 2.0 [26], and SSAKE 3.8.5 [27], are used to analyze the region specific FASTA file. ABySS assembles the sequencing reads by using the Bruijn graph [28]. ABySS use multiple kmer and trim settings. The minimum kmer coverage used for the assemble is 3 and kmer sizes used are: 30, 50, 70 and 90, and for every kmer size, three different maximum length of blunt contigs being trimmed are; 10, 30 and the same length as the used kmer. SSAKE assembles the reads by iteratively searching for reads mapping to the end of a contig, and the minimum depth of coverage used is 10.

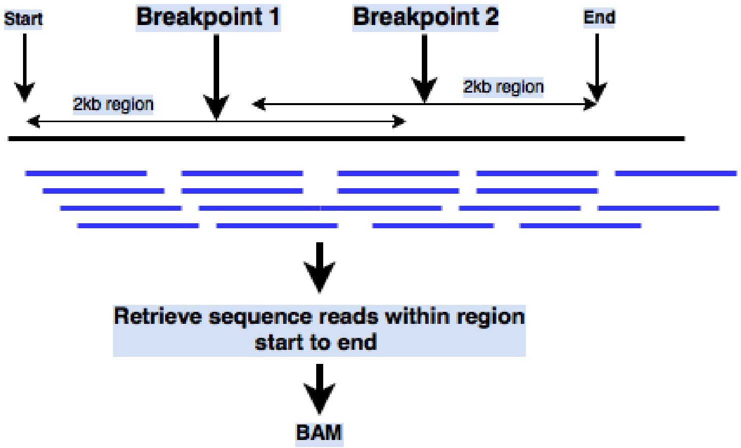

Fig 3. Extracting reads for overlapping breakpoint regions. The two breakpoint regions (2kb) are overlapping. In this case, the regions will be merged and sequence reads will be retrieved within the flanking ends of the regions (start and stop in the figure).

All the contigs created are then merged to a single FASTA file. The contigs are aligned to the reference genome using Burrow's wheeler alignment algorithm [29], and the mapped results are presented as a new SAM file. The mapped contigs from the SAM file are first checked if they have a cigar field with soft clipping (S), an indication of an true SV event. If a soft clipping is present, the distance from contig start position to the soft clipping event is calculated, and this is the new predicted breakpoint (Fig 4). The same procedure is carried out for the SA.

If no information concerning supplementary mapping is available, or if the regions fall outside the source 2kb regions, the contig will be ignored and SV classification and genotyping will only be carried out by read depth information. If SA information is provided, the contig parts mapped with soft clipping are also checked for strand direction by converting SAM flag to binary numbers with factor 2 and examine if 2^4 is present. If no soft clipping is found, the contigs are checked for matching (M) contigs spanning the breakpoints. If more than one soft clipping contig is found, the longest contig is used for further predictions. The first genotype predictions are performed by comparing soft clipping and completely matched regions. If the two contigs span over the same region, SVGenT will genotype this variant as heterozygous (symbolized as 0/1). Or, if no matching region span over the soft clipping breakpoints the genotype

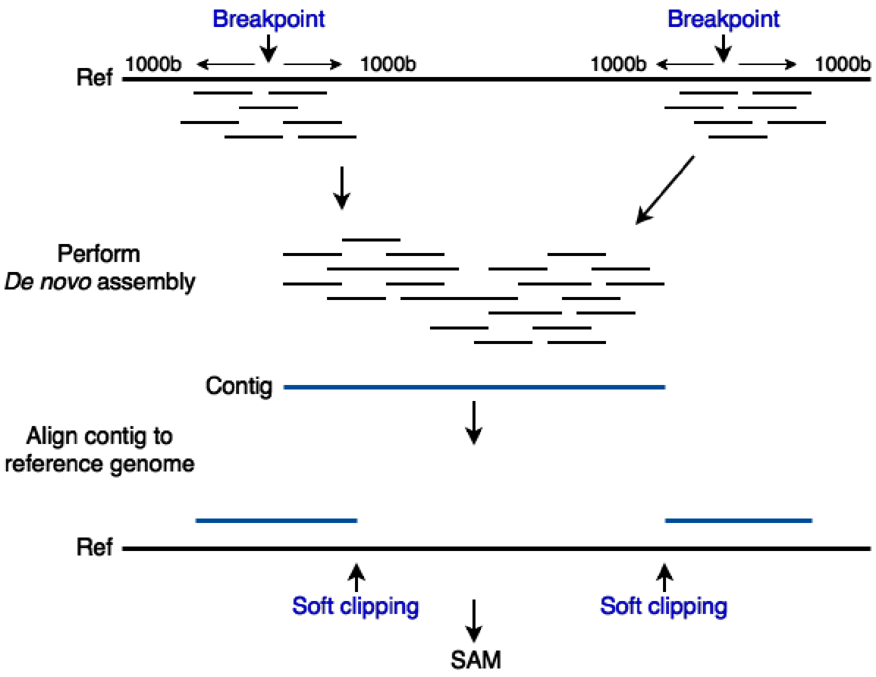

Fig 4. *De novo* assembly and alignment to reference. The sequence reads are retrieved within a 2kb region centered on the breakpoint positions. The reads are then *de novo* assembled into contigs, and mapped back to the reference genome. The result is presented in a SAM file. If the events of both an alignment and a SA are reported in a contig; soft clipping (S) will be present for the primary alignment, and hard clipping for the SA in the SAM file. In the example presented in the figure, the contig has been mapped with a SA. The cigar field in the SAM file could look like this: primary alignment; 500M500S, and for the SA; 500H500M (1000kb contig).

will be classified as homozygous (symbolized as 1/1) (Fig 5).

For the second genotype prediction and SV classification, read depth information is used. Three regions are considered, the region between the breakpoint junctions, the 2kb region surrounding the first breakpoint and the second. If the breakpoint junctions are located on different chromosomes, no read coverage information is collected between breakpoints and only two regions will be analyzed. The read coverage and GC bias for the regions are then corrected by the equation Eq 1 also implemented by [30] and [15], where RD_i is the raw read coverage for SV region found in sample-specific SQLite database, RD_{GC} is the average read coverage for bins with the same GC content as the breakpoint bin, RD_{all} is the average read coverage for all bins in the same chromosome as the region with a mappability score above a threshold of 0.5, and $\widetilde{RD_i}$ is the normalized read coverage for the SV region. All data were retrieved from the sample-specific SQLite database.

$$\widetilde{RD_i} = RD_i \cdot \frac{RD_{all}}{RD_{GC}} \tag{1}$$

Genotype and CNV classification by read-depth

If the region between breakpoint bins has a mappability score below 0.5, This SV will not be analyzed by SVGenT and the original SV will be kept in the new VCF. If the breakpoint bin has a mappability score equal to or above 0.5, the read depth

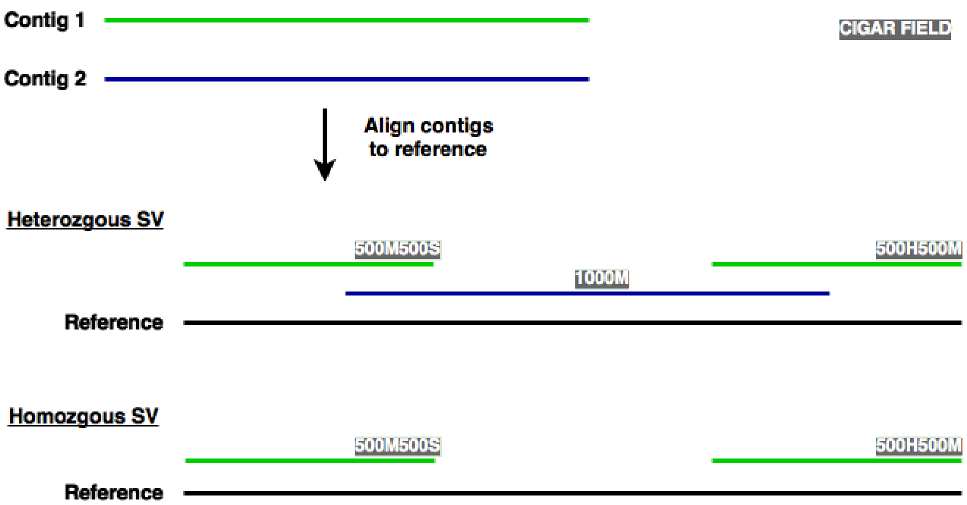

Fig 5. Genotyping called variants using contigs. The contigs produced through *de novo* assembly are used to classify the SV as homozygous or heterozygous. If two contigs span the same region, one with soft clipping and the other with complete match, this SV is classified as heterozygous. If only a single contig with soft clipping and SA is detected, the SV will be classified as homozygous.

information is used to classify SV type and genotype by comparing the region bins between breakpoints to the global average. The classification is described in Table 1.

Table 1. SV classification by read depth

Genotype	Read depth comparison
Homozygous DEL	$\widetilde{RD_i} < 0.25 \cdot RD_{all}$
Heterozygous DEL	$\widetilde{RD_i} > 0.25 \cdot RD_{all}$ and $\widetilde{RD_i} < 0.75 \cdot RD_{all}$
Homozygous DUP	$\widetilde{RD_i} > 1.75 \cdot RD_{all}$
Heterozygous DUP	$\widetilde{RD_i} > 1.25 \cdot RD_{all}$ and $\widetilde{RD_i} < 1.75 \cdot RD_{all}$

The strand information is also checked. If the directions of contig alignments are in two different directions, the SV will be classified as an INV. Otherwise, if the breakpoints are on different chromosomes or have not been classified as DEL, DUP or INV, they will be classified as BND.

The new SV genotypes, breakpoints and SV types are together with the statistics printed to an updated VCF file also containing the original SVs SVGenT failed to classify. A flowchart of SVGenT is presented in Fig 6.

Benchmarking the sensitivity and specificity of SVGenT

The performance of SVGenT SV-classification, breakpoint position and genotype prediction were evaluated first by using the public large-scale sequencing data NA12878 [20], and second by using a simulated dataset generated by CreateTranslocations [31]. A detailed description of the benchmarking is given in the supplementary S1 File.

Benchmarking using NA12878

SVGenT was benchmarked using the public large scale sequencing dataset NA12878 [20] retrieved as a BAM file. Variant calling was performed by TIDDIT (default

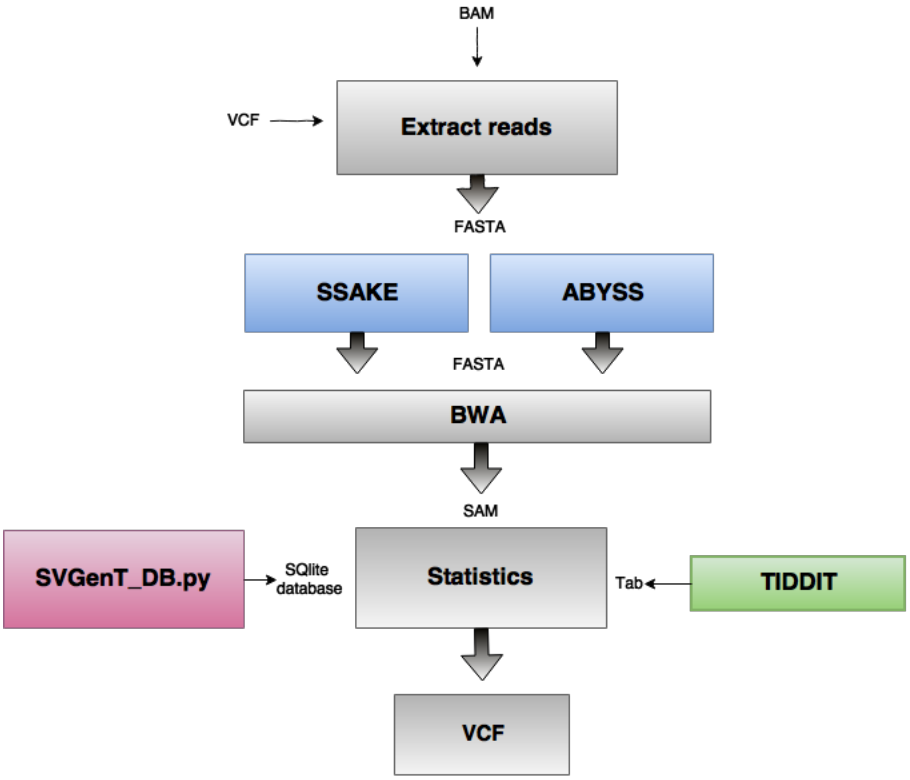

Fig 6. Flowchart of SVGenT. SVGenT takes a BAM, VCF and TAB file as input. From the VCF file, SV breakpoint start and end positions are retrieved. A 2kb region surrounding the breakpoint positions are calculated, and sequencing reads within this regions are collected from the BAM file, creating a separate BAM file for each SV. The BAM files are then converted to FASTA files and used as input for *de novo* assembly. Two assembly tools are used (ABYSS and SSAKE), generating new FASTA files with the assembled contigs. The contigs are mapped to the reference genome using BWA, presenting the result as SAM files. The alignment result are analyzed by looking at the mappability, GC content and read coverage over the region using SVGenT SQLite database and TIDDIT TAB file. The re-calibrated SVs are reported in a new VCF file together with calls SVGenT failed to classify

settings), CNVnator (bin size 100) and DELLY (only looking for DELs) separately, generating VCF and TAB files. SVGenT was then executed on the VCF files. The new VCF files containing updated variant breakpoints, SV types were compared to corresponding validated gold standard dataset of deletions [32] by merging the files using SVDB. The merging was performed using 10000 as maximum distance between breakpoints, and required overlap was 0.6. The performance was calculated using the equation Eq 2. Two different approaches for merging were performed, one by variant type adding –no_var as argument, and the other without. The performances of TIDDIT and DELLY were evaluated for comparison. SVGenT was also executed directly on the validated dataset to examine how well it performs excluding possible error sources in previous steps of variant calling.

$$Y = \frac{S_n - (M_n - T_n)}{T_n} \tag{2}$$

Y is the rate of correctly classified calls, S_n is the number of calls from the sample

VCF, T_n number of true variants in true VCF and M_n is the number of calls after merging sample VCF and true VCF.

Benchmarking using a simulated dataset

The performance of SVGenT was also evaluated using a simulated dataset generated by CreateTranslocation [31]. The dataset was created with zero translocations, 1000 tandem duplications, 2000 deletions and 1000 inversions, with a random distribution over all chromosomes excluding chromosome Y (female genome). The average read coverage was set to 30, the rate of homozygouzity events was set to 0.1 and the minimum distance between events was 5000. CreateTranslocation generates a sorted and indexed BAM file plus a TAB file describing each event. Variant calling was performed by TIDDIT and DELLY separately. SVGenT was executed on the generated VCF files plus one merged VCF file from both variant callers. Calls from each tool and the merged file were compared, before and after applying SVGenT, with the TAB file describing the true events. The accuracy, sensitivity and precision were calculated for each comparison.

Results

The performance of SVenX was tested on 13 samples of human 10x Genomics WGS data. The average time for completing the pipeline was 5 days and the number of SV calls in the final VCF were 2444 in average. Long Ranger WGS took 3 days to align, sort and perform calling of the WGS data, while the other tools required only a few hours (Table 2).

Table 2. Performance of SVenX pipeline was measured in time and the number of SV calls was counted for each variant caller

Process	Time	SV calls (n)
Long Ranger WGS	3(d)	3 993
CNVnator	1-3(h)	1 095
TIDDIT	1-4(h)	23 353
VEP (small indels)	<0.1(h)	-
SVDB merge	<0.1(h)	33 096
VEP	<0.1(h)	-
SVDB query	<0.1(h)	-
filtering and sorting folders	<0.1(h)	2 444
Total	4-5(d)	2 444

The time and number of calls are an average of 13 WGS samples.

The performance of SVGenT was first evaluated using the public dataset NA12878, and compared with a validated gold standard dataset of deletions. The validation was carried out in two different analyzes. First, by running SVGenT directly on the validated dataset to examine if the tool is capable of finding the variants. The validated dataset consists of 2677 deletions of which SVGenT succeed to find and classify 2461 SVs (92%). 2011 (82%) of the 2461 SVs SVGenT found was classified as deletions, 390 (16%) as break ends and the rest as inversions or duplications. Of the 2677 validated deletions, SVGenT classify 75% correctly of which 60% were classified by *de novo* assembly contig analyses and 40% was classified using breakpoint information from input VCF file and read depth information. The performance of SVGenT on the NA12878 data compared to validated deletions in various length are presented in Table 3.

Table 3. The performance of SVGenT on the validated gold standard dataset of deletions (NA12878).

	Small DELs	Medium DELs	Large DELs	Total DELs
SVGenT DELs	0.62	0.76	0.80	0.75
SVGenT SVs	0.67	0.98	0.87	0.92

The number of SVs and correct classified deletions using SVGenT were counted for four categories and compared with the validated gold standard dataset. The four categories are: (1) Small DELs (deletions smaller than 100bp), (2) Medium DELs (deletions between 100 to 1000bp), (3) Large DELs (deletions larger than 1000bp), and a total count including all DELs. The rate of correctly classified DELs and SVs are presented in the table.

Second, SVGenT was validated by running variant callers TIDDIT, CNVnator and DELLY on the NA12878 data and then executing SVGenT. The number of correctly classified deletions based on the validated dataset was compared between the different tools (Table 4). SVGenT was also benchmarked using simulated data generated by CreateTranslocations. Variant calling was performed by TIDDIT and DELLY. SVGenT was performed on the generated VCF files, plus one merged file with calls from both TIDDIT and DELLY. Calls from each tool and the combined file were compared with the true dataset of SVs and the result are presented in Table 5.

Table 4. The performance of SVGenT on the public large scale dataset NA12878.

	NA12878					
	TIDDIT		DELLY		CNVnator	
	-	SVGenT	-	SVGenT	-	SVGenT
Correct calls	0.55	0.43	0.71	0.73	0.005	-
Correct DELs	0.29	0.42	0.71	0.71	-	-

The rate of correctly called SVs (not considering variant type) and DELs for variant callers TIDDIT, DELLY and CNVnator on the public large-scale dataset NA12878. The performance of SVGenT for each caller is presented in the SVGenT column.

Table 5. The performance of SVGenT on a simulated dataset.

	Simulated data					
	TIDDIT		DELLY		Merged calls	
	-	SVGenT	-	SVGenT	-	SVGenT
Accuracy calls	0.36	0.43	0.84	0.84	0.94	0.95
True positive rate	0.50	0.54	0.82	0.82	0.94	0.95
False negative rate	0.50	0.46	0.18	0.18	0.06	0.05
Precision	0.35	0.43	0.84	0.84	0.79	0.80
Accuracy SV-type	0.32	0.42	0.65	0.65	0.73	0.76
Accuracy genotype	0.32	0.34	0.27	0.32	0.38	0.38

The rate of correctly called SVs for variant callers TIDDIT, DELLY, and one merged file with calls from TIDDIT and DELLY on a simulated dataset. First calculated ignoring variant type (*Accurate calls*), second by considering the variant type (*Accuracy SV-type*), and third by considering the genotype (*Accuracy genotype*). The accuracy was calculated for each category using the equation $(true\ positive + true\ negative)/(true\ positive + true\ negative + false\ positive + false\ negative)$. The true positive rate $(true\ positive/(true\ positive + false\ negative))$, false negative rate $(false\ negative/(true\ positive + false\ negative))$ and precision $(true\ positive/(true\ positive + false\ positive))$ were calculated for the calls. The performance of SVGenT for each caller and are presented in the SVGenT columns.

SVGenT analyzes about 700 SVs per hour, using one core on the high performance cluster, UPPMAX. The sample-specific database takes about 20 minutes to build.

Discussion

In the era of advancing long read sequencing technologies, efficient and precise bioinformatic tools to analyze these data are essential. It has been proven to be a difficult task to build new pipelines and tools with high precision [33], [34], and using several approaches will in most cases improve the result. Bioinformatic knowledge and computational resources are needed in order to use and install the tools. Improving the use of software by building automated pipelines for parallel execution speeds up the analysis and is more cost-effective and approachable. SVenX executes all steps from raw WGS data to analyzed SVs by a singe command, this includes filtering, alignment, variant calling by multiple tools and annotation for multiple samples at a time. To the best of our knowledge, there are no other pipelines using multiple callers to detecting SVs by analyzing 10x Genomic Chromium data. The time for executing the pipeline was in our setup less then five days, and the number of calls in the final VCF were 2400 on average. SV calls from every process are stored and can be found in a sample specific folder, making it easy to access data for evaluations of each step.

Long Ranger WGS is the step that requires the most resources. Being a pipeline, it performs de-duplication, filtering, alignment, calling and phasing variants using molecular Chromium barcodes. There is to our knowledge no equivalent software for 10x Chromium data for comparison. Similar pipelines exists for analysis of standard Illumina WGS data (without Chromium barcode analysis), where the execution time is between 20 to 50 hours [35] [36], which indicates the relevance of resources required for Long Ranger WGS.

The pipeline is optimized on performance time and to generate a manageable number of SV calls in the final VCF file. Therefore, the variant caller TIDDIT is especially trimmed on not calling small SVs without supporting evidence (a high number of SAs). This will probably result in the loss of some true SVs, and more optimization of the variant callers is necessary to improve the accuracy of the final result. Another limitation of the script is the launching from python. Running the whole pipeline in Nextflow exclusive would be more cost effective on account of required resources needed. It would also make the transferring, configuration and troubleshooting of the pipeline more manageable.

SVGenT is a complementary tool designed to improve the breakpoint junction position, SV classification and genotype prediction. Since SVGenT does not call variants, it is dependent on an effective primary variant caller in order to obtain and improve the initial SV information. When benchmarking SVGenT, we can only consider the SVs that the variant callers have detected. Benchmarking SVGenT using the public large scale sequencing data NA12878 [20] and simulated data from CreateTranslocation [31], demonstrates the difficulties in finding an accurate variant caller effective for all data types. SVGenT was also tested on the validated NA12878 gold standard set of deletions [32], and succeeded to detect 92% of all SVs ignoring the SV type, and classified 75% as deletions. When looking more close at the read depth information over the SV region SVGenT failed to classify, nothing gives an indication of this SV being an deletion. SVGenT was also evaluated using the complete NA12878 dataset and a simulated dataset, using three variant callers TIDDIT, DELLY and CNVnator separately. The capacity in detecting validated deletions was compared before and after SVGenT was executed. The performance diverges between callers, and also if SVGenT is utilized or not. CNVnator only called 0.5% of the SVs in the NA12878 dataset, and due to the small number of observations, we therefore decided not to evaluate the performance of SVGenT using only this caller. SVGenT improved the SV classification in three out of five cases, and in two cases it classified equally good as the variant caller. The detection rate was in one case worse using SVGenT, but in the same case it improved the classification by 13% compared to the variant caller. In all other cases, it was equally good or better at finding correct SVs. SVGenT also improved the genotype prediction for both variant callers separately, and equally good when applied to the merged data. The true positive rate, also known as sensitivity or recall, indicates how well SVGenT detect the SVs. In two out of three cases SVGenT increases the sensitivity and lower the false negative rate.

SVGenT was developed to find CNVs and INV. The benchmarking was performed to evaluate the performance of SVGenT by comparing the result before and after execution. The validated gold standard and true datasets used did only contain variants SVGenT specialized in detecting, and the analyses do not consider the number of SVs SVGenT classifies as BND due to the lack of ability in classifying translocations, insertions and other more complex rearrangements. SVGenT shows promising results. However, to be confident in the performance on all datasets, further analyzes have to be done. A validated gold standard dataset containing genotype information for NA12878 could not be found.

Conclusion

SVenX is an efficient, fast and user friendly pipeline for the detection and analyzes of SVs from 10X Genomic WGS data. As a result of a highly parallelized pipeline, SVenX has the capacity of analyzing numerous 10x Genomics WGS samples within a week. The pipeline calls and annotates SVs using several techniques considering different information such as barcodes, read coverage, discordant read pairs, split reads, etc. The genotyper SVGenT has separately from the pipeline been tested on public sequencing datasets and simulated datasets for benchmarking. The results show that SVGenT has the capacity of detecting and improving the SV classification of already called variants, and provide a more accurate VCF file. SVGenT also provides important information including normalized read coverage information for both breakpoints (2kb) separately and between breakpoints, which are added to the updated VCF and can be used for more complex SV analyzes.

Supporting information

S1 File. Supplementary methods. The procedure of benchmarking SVGenT.

Acknowledgments

I would like to thank my supervisor Anna Lindstrand for giving me the opportunity to do and complete this thesis, and for introducing me to the world of medical research and structural variants. I would also like to thank Jesper Eisfeldt for the best possible bioinformatical support, interesting discussions, and for always making me a better programmer. Last I want to thank Daniel Nilsson for good bioinformatic advices and support.

References

1. Sebat J, Lakshmi B, Malhotra D, Troge J, Lese-Martin C, Walsh T, et al. Strong association of de novo copy number mutations with autism. Science. 2007;316(5823):445–449.

2. Stone JL, O'Donovan MC, Gurling H, Kirov GK, Blackwood DH, Corvin A, et al. Rare chromosomal deletions and duplications increase risk of schizophrenia. Nature. 2008;455(7210):237–241.

3. Gasull-Camos J, Tarres-Gatius M, Artigas F, Castane A. Glial GLT-1 blockade in infralimbic cortex as a new strategy to evoke rapid antidepressant-like effects in rats. Transl Psychiatry. 2017;7(2):e1038.

4. Pang AW, MacDonald JR, Pinto D, Wei J, Rafiq MA, Conrad DF, et al. Towards a comprehensive structural variation map of an individual human genome. Genome Biol. 2010;11(5):R52.

5. Alkan C, Coe BP, Eichler EE. Genome structural variation discovery and genotyping. Nat Rev Genet. 2011;12(5):363–376.

6. Raphael BJ. Chapter 6: Structural variation and medical genomics. PLoS Comput Biol. 2012;8(12):e1002821.

7. MacDonald JR, Ziman R, Yuen RK, Feuk L, Scherer SW. The Database of Genomic Variants: a curated collection of structural variation in the human genome. Nucleic Acids Res. 2014;42(Database issue):D986–992.

8. Church DM, Schneider VA, Graves T, Auger K, Cunningham F, Bouk N, et al. Modernizing reference genome assemblies. PLoS Biol. 2011;9(7):e1001091.

9. Phan L, Hsu J, Tri LQ, Willi M, Mansour T, Kai Y, et al. dbVar structural variant cluster set for data analysis and variant comparison. F1000Res. 2016;5:673.

10. Zheng GX, Lau BT, Schnall-Levin M, Jarosz M, Bell JM, Hindson CM, et al. Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. Nat Biotechnol. 2016;34(3):303–311.

11. Elyanow R, Wu HT, Raphael BJ. Identifying structural variants using linked-read sequencing data. Bioinformatics. 2017;.

12. Garcia S, Williams S, Xu AW, Herschleb J, Marks P, Stafford D, et al. Linked-Read sequencing resolves complex structural variants. bioRxiv. 2017;doi:10.1101/231662.

13. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, et al. The variant call format and VCFtools. Bioinformatics. 2011;27(15):2156–2158.

14. Tattini L, D'Aurizio R, Magi A. Detection of Genomic Structural Variants from Next-Generation Sequencing Data. Front Bioeng Biotechnol. 2015;3:92.

15. Abyzov A, Urban AE, Snyder M, Gerstein M. CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. Genome Res. 2011;21(6):974–984.

16. Eisfeldt J, Vezzi F, Olason P, Nilsson D, Lindstrand A. TIDDIT, an efficient and comprehensive structural variant caller for massive parallel sequencing data. F1000Research. 2017;6:664. doi:10.12688/f1000research.11168.1.

17. McLaren W, Gil L, Hunt SE, Riat HS, Ritchie GR, Thormann A, et al. The Ensembl Variant Effect Predictor. Genome Biol. 2016;17(1):122.

18. Eisfeldt J. SVDB; 2017. `https://github.com/J35P312/SVDB.git`.

19. Rausch T, Zichner T, Schlattl A, Stutz AM, Benes V, Korbel JO. DELLY: structural variant discovery by integrated paired-end and split-read analysis. Bioinformatics. 2012;28(18):i333–i339.

20. Zook JM, Catoe D, McDaniel J, Vang L, Spies N, Sidow A, et al. Extensive sequencing of seven human genomes to characterize benchmark reference materials. Sci Data. 2016;3:160025.

21. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. Nat Biotechnol. 2017;35(4):316–319.

22. Tyner C, Barber GP, Casper J, Clawson H, Diekhans M, Eisenhart C, et al. The UCSC Genome Browser database: 2017 update. Nucleic Acids Res. 2017;45(D1):D626–D634.

23. Derrien T, Estellé J, Sola SM, Knowles DG, Raineri E, Guigó R, et al. Fast Computation and Applications of Genome Mappability. PLoS ONE. 2012;7(1):e30377. doi:10.1371/journal.pone.0030377.

24. Derrien T, Estelle J, Marco Sola S, Knowles DG, Raineri E, Guigo R, et al. Fast computation and applications of genome mappability. PLoS ONE. 2012;7(1):e30377.

25. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009;25(16):2078–2079.

26. Jackman SD, Vandervalk BP, Mohamadi H, Chu J, Yeo S, Hammond SA, et al. ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter. Genome Res. 2017;27(5):768–777.

27. Warren RL, Sutton GG, Jones SJ, Holt RA. Assembling millions of short DNA sequences using SSAKE. Bioinformatics. 2007;23(4):500–501.

28. Compeau PE, Pevzner PA, Tesler G. How to apply de Bruijn graphs to genome assembly. Nat Biotechnol. 2011;29(11):987–991.

29. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv. 2013;doi:arXiv:1303.3997v2[q-bio.GN].

30. Yoon S, Xuan Z, Makarov V, Ye K, Sebat J. Sensitive and accurate detection of copy number variants using read depth of coverage. Genome Res. 2009;19(9):1586–1592.

31. Eisfeldt J. CreateTranslocation; 2016. `https://github.com/J35P312/CreateTranslocations.git`.

32. Parikh H, Mohiyuddin M, Lam HY, Iyer H, Chen D, Pratt M, et al. svclassify: a method to establish benchmark structural variant calls. BMC Genomics. 2016;17:64.

33. Hwang S, Kim E, Lee I, Marcotte EM. Systematic comparison of variant calling pipelines using gold standard personal exome variants. Sci Rep. 2015;5:17875.

34. Cornish A, Guda C. A Comparison of Variant Calling Pipelines Using Genome in a Bottle as a Reference. Biomed Res Int. 2015;2015:456479.

35. Li H. FermiKit: assembly-based variant calling for Illumina resequencing data. Bioinformatics. 2015;31(22):3694–3696.

36. Guo Y, Ding X, Shen Y, Lyon GJ, Wang K. SeqMule: automated pipeline for analysis of human exome/genome sequencing data. Sci Rep. 2015;5:14283.