UNIVERSITY OF COPENHAGEN
Department of Biology

Bachelor in Biology

Rasmus Alex Buntzen-Christensen

Structural variant calling in Human genomes.

Supervisor: Shilpa Garg

Submitted on: June 17th 2022

Type Units: 54.938 characters (with spaces)

Contents

Abstract

Structural variants are large scale mutations from 50 bp of length to chromosome scale. Structural variants affect more than 10% of the nucleotides in a human genome and is therefore the main contributor to the genomic variations found in human populations. Improvements in DNA sequencing technologies has enabled the discovery these large scale mutations. Structural variants are important to understand the mechanisms of genetic diseases such as cancer. In this thesis two bioinformatic pipelines to find structural variants are presented based on read mapping and de novo assembly methods. These methods found 30.280 and 51.801 structural variants in the HG002 (GM24385 Ashkenazi Son) dataset. Presenting a clear step-by-step workflow for structural variant calling will help further research in creating a comprehensive understanding of the mechanisms behind cancer and other genetic diseases, that is needed to cure and prevent the illnesses.

1 Introduction

1.1 Why structural variation in the human genome is important

Structural variants are the main contributor to genome evolution and diseases, it is important to study and understand structural variants, hereunder the types of structural variants, their phenotypic affect, and their relative abundance in the human genome.

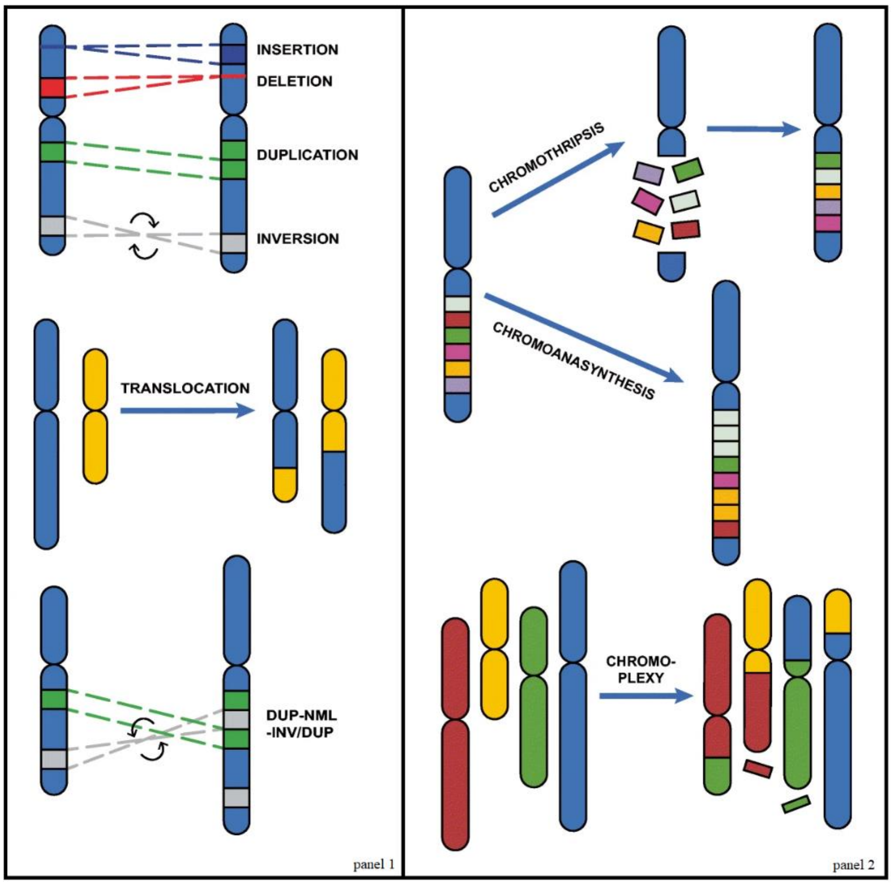


Figure 1
Panel 1: Structural variants can be categorized into categories based on their affect on the genome, here the 5 most common structural variant types are shown (insertion, deletion, duplication, inversion and translocation), aswell as an example of a more complex structural variant both involoving a duplication and inversion.
Panel 2: Many mechanisms can create structural variants, here the two most devistationg mechanisms are shown, nemely chromothripsis and chromoplexy
Figure adapted from Figure 1 in Structural variant identification and characterization by Parithi Balachandran et. al. https://doi.org/10.1007/s10577-019-09623-z

Structural variants have been strongly associated with human disease, especially cancers[1]. Cancer is caused by an accumulation of mutations in somatic cells that lead to extreme cell proliferation[2]. In Denmark one out of every three people will be diagnosed with cancer before they turn 75, in 2020 362.515 Danish citizens had a cancer diagnosis an around a third of these people will die of cancer[3].

Structural variants have been found with higher frequency than on average across the genome, in certain genes from cancerous tumors[1]. These genes include the BCL2 gene family[1] which is essential for the regulation of programmed cell death (apoptosis)[4], as well as many other oncogenes[5, 6]. Chromothripsis (see Mechanisms of SVs for explanation) is found in 3% of all cancer tumors, 25% of all bone tumors, and 10% of all brain tumors[7].

People with autism spectrum disorders have significantly more copy-number variants in their genome compared to human reference genomes[8]. Structural variants in 27 gene have been found to correlate with autism[8], an example is the oxytocin gene OXT which is responsible for the production of oxytocin[9]. A change in oxytocin levels have been found to alter social behavior in humans and mice[10, 11], corresponding to behaviors observed in people with autism spectrum disorders[8].

Structural variation in the human genome is known to be a significant factor in the development of human diseases. The study of structural variation is therefore an important contribution to understanding, preventing and curing human disease[12]. Furthermore, more reference genomes for structural variants are needed, to better understand the population structures seen in humans[13].

A simple bioinformatic pipeline from DNA sequencing all the way to structural variant calling/detection is needed in order to help create fast, cheap and precise studies on structural variation

1.2 The human genome

Deoxyribonucleic acid (DNA) contains the genetic code, and is fundamental to all life[14]. DNA consists of two chains of nucleotides coiled around each other to form a double helix, this structure is stabilized by hydrogen bonds between the nucleotides[14]. There are four nucleotides, adenine (A), cytosine (C), guanine (G), and thymine (T). The order of these four nucleotides is what determines the genetic code. Figuring out the nucleotide order has therefore been a fundamental challenge in biology ever since the discovery of DNA, by Watson and Crick in 1953[15].
A genome is the entire genetic code found in a cell[16]. A genome is not one long continuous DNA strand, but several shorter DNA strands called chromosomes, the human genome consists of 23 pairs of chromosomes[16]. As human somatic cells are diploid, the human genome consists of two copies of each chromosome, each copy is referred to as a chromosome-scale haplotype[16]. The entire human genome consists of two genome-scale haplotypes, each with a length of approximately 3.140.000.000 nucleotides[17]. The beginning and end of chromosomes consists of highly repetitive regions called telomeres, that consists of a 6 bp repeat TTAGGG, repeated over and over again with a length of 8.000-10.000 bp[18].

1.3 Single nucleotide variation

Single nucleotide variation (SNV) is the most fundamental type of variation found in the human genome, this type of variation consists of single nucleotide differences between humans [19]. This type of variation is found in more than 10 million nucleotide locations throughout the human genome [20], which roughly corresponds to 1 out of every 300 nucleotides. SNV has therefore previously been thought to contribute with the majority of variation found in the human genome.

1.4 Structural variation

Structural variants (SVs) are historically categorized as intraspecific polymorphic DNA regions larger than 1.000 nucleotides (1 kb) [20–22], though the cutoff minimum length of what is considered a structural variant varies a lot from project to project depending of the purpose. Most new literature categorize the minimum cutoff length as 50 bp, which is also the minimum length that is used throughout this thesis.

As sequencing technologies and genome assembly algorithms have improved, more SVs have been found in the human genome. Currently known SVs affect many fold more nucleotides in the average human genome than point mutations[21], a population genetic study of 2.504 human genomes by Sudmant PH et al. shows that approximatley 13% of nucleotides in the human genome is affected by structural variants [12]. Two humans have an average genomic variation of 0.1% if only SNV is compared[23], if SVs are compared two human genomes have an average genomic variation of 1.5%[23]. Structural variation is therefore the main contributor to the total genomic variation found in humans.

Structural variation is an umbrella term and can be further categorized based on the mutations effect on the DNA, see Figure 1 panel 1. Copy-number variant (CNV) is used for structural variants that changes the number of copies of a certain part of the DNA[22]. Copy-number variants includes deletions, where a part of the DNA is removed[20], thereby reducing the copy-number; duplications where a part of the DNA is copied and inserted into another part of the genome[24], thereby increasing the copy-number. These CNVs can be called Copy-number polymorphism (CNP) if the haplotype is found in more than 1% of the population[20].

Mutations that do not change the copy-number can also be the cause of structural variation, these types of SVs are called balanced variants[12], because they do not affect copy numbers. Mutations causing balanced variants include inversions, where part of the DNA is inverted in reference to the rest of the genome[25]; Translocations, where a piece of DNA is detached from its chromosome and then reattached to a different chromosome[25], this type of mutation is called cut&paste insertion if the detached DNA is reattached in a different location on the same chromosome[26]. Mobile genetic elements such as LINEs, SINEs and retrotransposons can also be categorized as structural variants[27].

1.4.1 Mechanisms of SVs

Many mechanisms have been found to generate structural variation. One of the mechanisms resulting in structural variation is double strand breaks, where both DNA strands are damaged at the same time[28]. Cells have evolved two strategies for fixing double strand breaks, nonhomologous end joining and homologous recombination[28]. Homologous recombination repairs the break without loss of nucleotides[28], but this process is only possible if there is a template strand that can be used to repair the break, this is often the case right after DNA replication. The template DNA for the DNA replication can be used as a template for the break and the DNA can be ligated back together[28]. However, sometimes a highly similar but not identical DNA sequence is used as the repair template[29], which can results in structural variants such as insertion, duplications and inversion[30].

If no repair template is present, nonhomologous end joining is the only option to repair the break[28]. Most double strand breaks do not happen at exactly the same location on both strands, the breaks however happen staggered by a few nucleotides[31]. This results in the breaks having overhangs of single stranded DNA which can be joined together. If the break happens in such a way that there is no overhang or the overhangs are damaged, there is no template for the repair present[28]. When this happens nucleases starts removing nucleotides from the broken ends until the strands are compatible[28], this results in deletions, and if they are large enough they can be categorized as structural variants.

Double strand breaks are a single event that creates a single structural variant. Mechanisms exists that can create more than a hundred double strand breaks simultaneously in the same chromosome[32]. This processes is called chromothripsis[6]. Hundreds of double strand breaks occurring clustered together in the same chromosome is detrimental to the cell, DNA repair is possible, but the broken fragments are stitched together seemingly at random[32], this of course produces numerous structural variants. The exact biological mechanism is still unknown, however most likely the breaks occur during mitosis caused by an environmental factor such as free radicals or radiation[33].

Chromoplexy is another documented but relatively unknow mechanism that produces several double strand breaks at the same time and is therefore also a likely structural variant contributor[6].

1.5 DNA sequencing

DNA sequencing determines the order of the four nucleotides in a given piece of DNA[34]. In 1977 Frederick Sanger invented the first method for DNA sequencing, he called it Sanger Sequencing[34]. The Sanger chain termination method and the chain degradation method proposed by Maxam and Gilbert is collectively called first generation sequencing[35]. This was a major biological breakthrough, that now for the first time allowed scientists to directly analyze the genetic code. However the technology was slow and cumbersome, extremely expansive and throughput was low[36, 37]. New sequencing methods were therefore continually invented and improved, and by 2005 the era of second generation sequencing (also called next generation sequencing) started[36, 37].

The 454 Life Sciences company essentially started the second generation of DNA sequencing, with their 454 method[38]. 454 allows for parallel sequencing, meaning that a DNA molecule can be fragmented into reads of 50 to 500 base pairs using restriction enzymes, whereafter each DNA read can be sequenced separately but simultaneously[39]. The original DNA sequence can be computationally assembled after sequencing[39]. Second generation sequencing brought higher accuracy, lower cost and much faster sequencing, and is therefore a clear improvement compared to first generation sequencing[36]. Nevertheless, the technologies still had problems, the relatively short reads made it difficult to do whole genome sequencing, and the process of assembling the reads are very computationally demanding.

Third generation sequencing methods primarily invented by Oxford Nanopore and Pacific Biosciences uses much longer reads[36, 37], averaging 10 kb and up. Longer reads theoretically eases the assembly process and helps spanning the highly repetitive regions of the genome[40]. Third generation sequencing is therefore optimal for whole genome sequencing. Third generation sequencing is currently under heavy development, one if the biggest issues is a relatively high error rate compared to second generation sequencing[40].

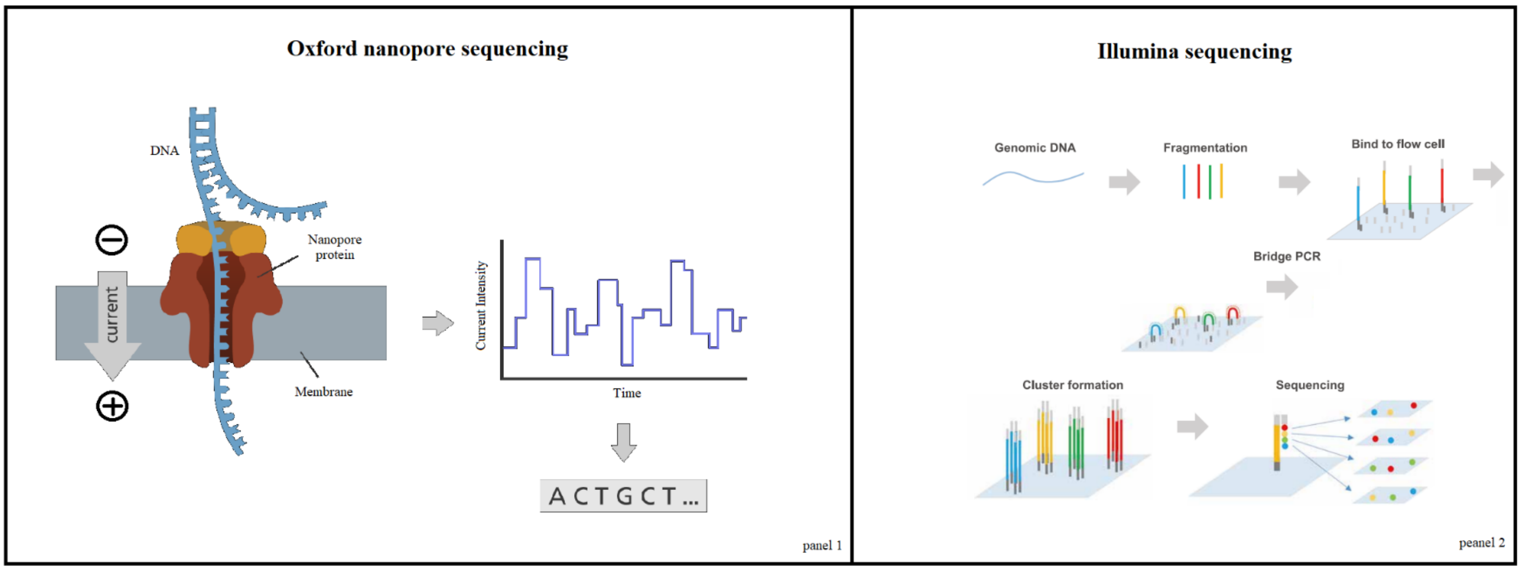


Figure 2 Overview of sequencing technologies.
Panel 1: Overview of ONT sequencing. Membrane potential changes based on the nucleotides passing through the nanopore protein.
Panel 2: Overview of Illumina sequencing. The complementary DNA sequence of the fragments are synthesized using fluorescent nucleotides which illuminates when excited by a light source.
Figure adapted from Figure 1, by Genome Research Limited, https://www.yourgenome.org/facts/what-is-oxford-nanopore-technology-ont-sequencing and Figure 2, by GenScript, https://www.genscript.com/advancing-genomics-medicine-and-health-together-by-semiconductor-dna-synthesis-technology-summary.html

1.5.1 Short read sequencing

Short read sequencing is primarily produced by second generation sequencing methods. Currently the most used short read sequencing method is Illumina dye sequencing[37, 41]. DNA fragments are ligated to a flow cell, here each fragment is amplified by bridge PCR[42]. The fragments are denatured and sequencing by synthesis proceeds. The single stranded DNA fragments complementary strand is synthesized one nucleotide at a time using fluorescent nucleotides[42]. For each nucleotide added, the fragments are excited by a light source, and the fluorescent nucleotides light up[42], and the color of the nucleotide is recorded[42]. The fluorophore which gives the nucleotides its fluorescens, acts as a blocking group which ensures that only one nucleotide is added in the elongation process at a time[42]; this ensures that all nucleotides of the fragment is sequenced one by one. See Figure 2 panel 2 for a graphical overview of the method.

During bridge amplification copies of the forward DNA strand and the reverse DNA strand is produces. Only one strand direction is sequenced at a time, but by introducing a second bridge amplification after the first strand is sequenced, the complementary strand can also be sequenced[42]. Using an index sequence, the forward and reverse strand for each fragment is linked. This is called paired end sequencing, and ensures a very low error rate since each fragment essentially is sequenced twice[40].

Each of the four florescent nucleotides has a distinct color. So, by analyzing the sequence of colors produced in each flow cell the genetic code can be inferred. Each DNA fragment produces a nucleotide sequence, these sequences are called reads.

1.5.2 Long read sequencing

Long read sequencing is produced by third generation sequencing methods such as PacBio and Nanopore[40]. In this thesis Nanopore sequencing data Oxford Nanopore Technologies is used.

The core principle of nanopore sequencing is that the four nucleotide bases adenine (A), cytosine (C), guanine (G), and thymine (T) have distinct ion charges when they pass through an electro-resistant membrane, this ionic charge can be measured and used to identify the DNA sequence[43], see Figure 2 panel 1 for a graphical overview of the method.

The sequencing machine is packed with one or more flow cells, these flow cells contain an electro-resistance membrane with a large array of nanopores embedded[44]. Each nanopore contains an electrode that measures the charge running through the nanopore at the given time[40, 44]. Motor proteins move DNA fragments through the nanopores one nucleotide at a time, as each nucleotide passes through the nanopore it affects and thereby changes the voltage running through the nanopore. The change in electric charge as each nucleotide passes through the nanopore is measured and stored in a .FAST5 file[45].

The so called electric “squiggle” recorded in the .FAST5 file can computationally be translated to a nucleotide sequence, this process is called basecalling. There are numerous basecalling algorithms, Oxford Nanopore Technologies have developed a basecalling algorithm called Guppy which is currently the state of the art, and is therefore also used for GM24385 dataset[46] used in the later presented pipeline. The output of basecalling is typically a .FASTA or .FASTQ file containing a description of each read followed by the nucleotide sequence.

Nanopore sequencing can produces 1D and 2D sequences. 1D sequencing only sequences the forward strand of the DNA fragment, where 2D sequencing both sequences the forward and reverse strand of the DNA fragment[47]. 2D sequencing have information from both DNA strands when basecalling and therefor has higher accuracy[47]. 1D sequencing on the other hand allows for longer DNA fragments which in the end results in longer reads, the tradeoff being lower accuracy than 2D sequencing. Oxford Nanopore is currently developing what they call 1D squared sequencing which they claim to have the accuracy of 2D sequencing while producing reads comparable in length to 1D sequencing[47, 48].

1.5.2.1 Sequencing summary statistics

For quality control of the raw sequencing data a number of statistics can be used. The statistics used in this thesis will now be explained.
Phred is a quality score given to each nucleotide by the basecalling algorithm, the Phred score is logarithmically related to the basecallers error probability for that nucleotide[49, 50]. This means that the Phred score can be directly transformed to an error probability of the basecaller by the following function:

$Error\ probability\ of\ sequncing = 10^{-\frac{Phred}{10}}$ [49, 50]

Sequencing depth/depth of coverage can be calculated using the Lander/Waterman equation[51]:

$$depth\ of\ coverage = \frac{number\ of\ reads \cdot average\ read\ length}{genome\ size}$$

Depth of coverage is how many reads covers or aligns to an arbitrary part of the reference, i.e. the number of times an arbitrary part of the genome on average has been sequenced.

2 Computational pipeline

In this chapter two computational pipelines for structural variant calling of ONT long reads are presented[52], as well as the mechanisms and the algorithms behind the computations. For a quick overview of the pipelines see the flowchart in Figure 3.

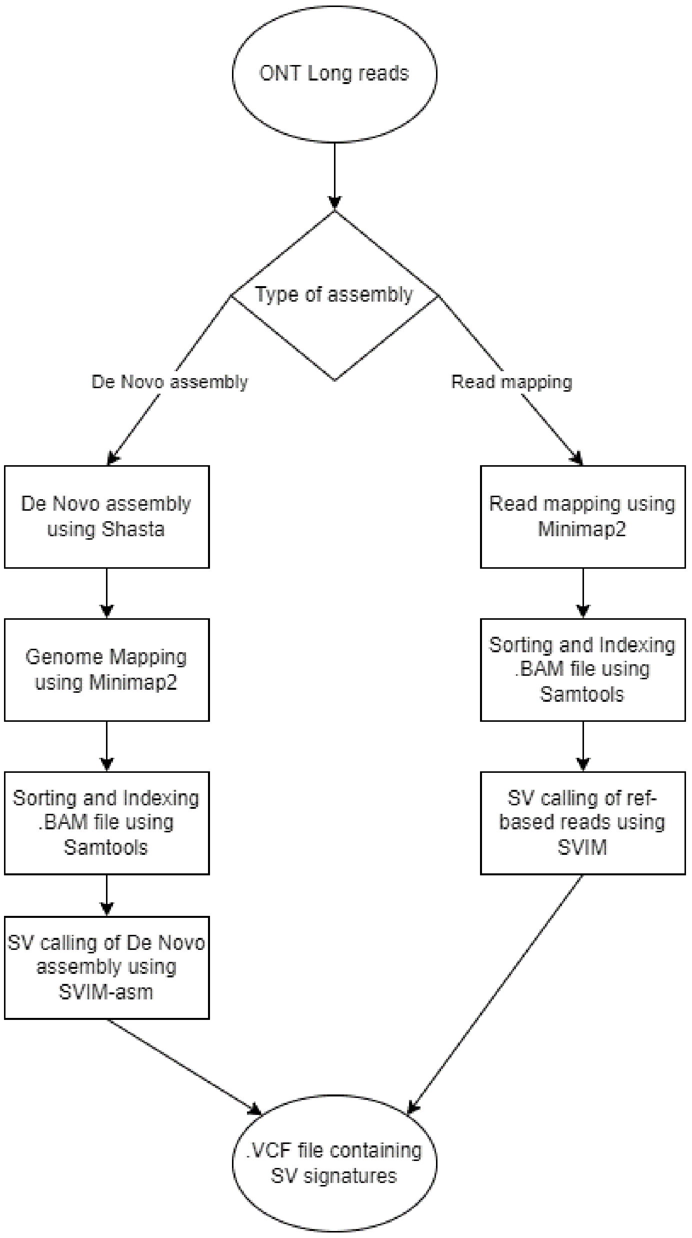

ONT Long reads
Type of assembly
De Novo assembly
Read mapping
De Novo assembly using Shasta
Genome Mapping using Minimap2
Sorting and Indexing .BAM file using Samtools
SV calling of De Novo assembly using SVIM-asm
Read mapping using Minimap2
Sorting and Indexing .BAM file using Samtools
SV calling of ref-based reads using SVIM
.VCF file containing SV signatures

Figure 3 Flowchart of the structural variant calling pipeline. Figure created with https://app.diagrams.net/

The programs used throughout this thesis are chosen based on two factors. Programs directly designed for long ONT reads are chosen, to ensure an optimal performance of the pipelines. Programs that receive regular updates are chosen to ensure the pipelines are future-proof.

2.1 Genome assembly

After sequencing with second or third generation sequencing, the next step is assembling the reads in order, so the full nucleotide sequence of the unfragmented DNA can be found. There are two overall strategies of human genome assembly, read mapping and de novo assembly[53].

Read mapping uses a reference genome or a panel of genomes to probabilistically assign each sequenced read to its correct place in the genome[54]. Theoretically each read can be compared to all locations in the reference and the location with the highest alignment score is chosen as the correct location for that read. In reality this process is far too slow when scaled[55]. Mapping programs using different algorithmic procedures have been developed for different lengths of reads and specifically for DNA or RNA sequences to speed up the mapping. Some of the most used mappers are BWA, BLAT, ZOOM, SeqMap, MAQ, TopHat, CloudBurst and Bowtie2[56]. What most mappers have in common is that they use some kind of indexing to speed up the mapping procedure[57]. The most used indexing method is FM-indexing[57]. Read mapping requires a reference genome, distinguishing true genetic variation from sequencing and mapping errors are therefore inherently difficult[53]. To combat the biases created by having a reference, genome assemblers that only uses the raw reads have been developed, this method is called de novo assembly.

De novo assembly does not use a reference genome, it uses overlaps between the sequenced reads to assemble the genome. De novo assembly is extremely complicated and computationally intensive. Multiple assemblers have been created, each using slightly different algorithms to speed up the computation. Popular de novo assemblers include Shasta, Flye, Canu, Hifiasm, Falcon and Wtdbg2[53, 58]. The exact algorithms used depends on the assemblers, but the overall workflow of the assembly looks something like this: Overlapping parts between reads is a sign that the two reads should be positioned next to each other in the assembly. Using this method the reads are assembled into an overlap graph which represents all the reads positions to each other and their overlaps, using the overlap graph contigs are created[59], this step is extremely computationally complex and memory intensive[60]. Contigs are groups of reads which have been predicted to orient from neighboring positions in the genome, using the overlap graph[61]. The contigs are hereafter oriented and placed in the correct order. Reads predicted to orient from the same location may have minor nucleotide differences, the assembler chooses the most likely nucleotide for that given position and hereby creates a consensus sequence[42]. The goal of the assembler is to create one long continues contig for each chromosome[42], this requires complete coverage of the entire genome which is not possible with current technology. Certain parts of the DNA is especially difficult to sequence, such as long repeating sequences seen in centromeres and telomeres[62]. A so-called scaffold is therefore created, a scaffold consists of contigs of varying length and gaps where the sequencing and assembly failed.

2.1.1 Minimap2

In this thesis Minimap2 v2.24-r1122[63] is used for read mapping. Minimap2 uses a standard seed-chain-align procedure[63]. Explaining the algorithm of Minimap2 is beyond the scope of this thesis, I therefor refer the reader to the original paper for Minimap2 by Heng Li[63]. Minimap2 have been chosen because it is specifically designed for mapping ONT long read data to the human genome[63], and therefore in theory should perform better than its competitors (Bowtie2, SNAP and BWA-MEM). When ONT published the GM24385 dataset they included Minimap2 read mapping in their Snakemake pipeline, which means mapped and indexed .bam files are readily available for the dataset used in this thesis. Furthermore, Minimap2 is also capable of assembly-to-assembly alignment[63], which is needed in one of the presented workflows (this is further explained later in the thesis).

2.1.2 Shasta

In this thesis Shasta v0.8.0[58] is used for de novo genome assembly. Shasta is designed for assembling Oxford Nanopore sequences and is extremely fast compared to other assembly programs[58]. Shasta uses more or less the same procedure as described in the previous segment; it does however differ from its competitors in a couple of key areas. Instead of using the raw reads as they appear in the inputted FASTA/Q file, Shasta uses run-length encoding[64]. Run-length encoding substitutes repeating nucleotides with a repeat count (TTAC would be represented as T2A1C1), this generally shortens each read with 25% and thereby speed up the assembly[64]. Shasta holds all data in Random-access memory (RAM)[64], hereby eliminating the slow process of reading and writing to memory, which further speeds up the assembly. This on the other hand also means that large amounts of RAM is needed to effectively run the assembler, 1-2Tb of RAM is recommended for assembly of human genome at x60 coverage[64].

2.1.2.1 Shasta summary statistics

To compare assemblies performed with Shasta, N50, L50 and genome fraction are used.
Genome fraction is the ratio of how long the assembly is compared to the reference genome[65]. Genome fraction can therefore be used to assess how gapped an assembly is.
To compute N50 and L50, all contigs are sorted based on their length. Starting from the longest contig, contig lengths are summarized until the summarized contig length is ≥ 50% of the total assembly length. Of these contigs that have had their lengths summarized, N50 is the length (in nucleotides) of the shortest contig in this group[65]. When summarizing the contig lengths, L50 is the number of contigs needed before the summarized contig length is ≥ 50% of the total assembly length[65]. N50 and L50 might at first sight seems like odd metrics to compare assemblies with, but they are a great way of assessing the size of the contigs in an assembly. A low N50 and a high L50 indicates an assembly containing many short contigs, and therefore a very fragmented assembly. A high N50 and a low L50 indicates that the assembly is comprised of few long contigs, which in most cases also means that the assembly has a high genome fraction.

2.2 Structural Variant Calling

Structural variants can be found using programs and algorithms generally known as structural variant callers. The main reason why this thesis uses long sequenced reads is that long reads have been shown to produce better and more accurate structural variant calls than short reads[61, 66–68]. Short read mappers and SV callers have been widely developed and used, some note worth SV callers for short reads are BreakDancer, DELLY, LUMPY and MANTA, for further explanation of short read SV calling please see the review articles written by Medhat Mahmoud et al. [69].

As earlier explained, long reads can fully span regions that are difficult to sequence and assemble, and thereby improves accuracy in these regions[66]. Long reads also means that a single read can encapsulate an entire SV, which leads to easier and more accurate SV calling[69]. For SV calling of long reads there is two overall procedures, each requiring a certain type of input. Either a de novo assembled genome can be used as the input for SV calling, or a read map[69]. The two procedures have in common that before SV calling, the assembly needs to be mapped to a reference genome. Contigs or reads that partially or does not map at all to the reference is potential structural variants[70]. The contigs or reads flagged by the mapper as not mappable is then further examined by the SV caller to determine if they are SVs. In this thesis two SV callers produced by The Max Planck Institute for Molecular Genetics are used.

2.2.1 SVIM

For SV calling of the read mapped data SVIM v1.4.2[26] is used. SVIM can recognize six different SV classes; deletions, inversions, cut&paste insertions, tandem duplications, interspersed duplications, and novel element insertions.

SVIM expects a mapped, sorted and indexed .bam file of read maps as input. The SVIM algorithm consists of three phases, the first phase is collection of SV signatures[26]. Collection of SV signatures gathers the mapped reads produced by Minimap2, reads with unusual characteristics are then flagged, these flagged reads that are unusual and could not be optimally mapped is then passed on to phase two[26]. There can be many causes of why a read is flagged, but the most common reason is that the read is chimeric, meaning that is has been mapped to two distinct locations in the reference genome by the genome mapper[26]. This behavior of a read is consistent with multiple types of structural variation and obvious candidate for further analysis by SVIM.
Phase two is Clustering of SV signatures. In phase one, reads containing possible structural variants were identified, first step of phase two is to merge all similar reads, this is done by graph-based clustering and distance matrices[26]. Now all possible SVs are represented by a cluster of reads. Step two of phase two is scoring how likely each cluster is to be a SV, this score is based on four features[26]. 1: A score is given based on the number of reads in the cluster; more reads in a cluster decreases the probability of sequencing or mapping error being the origin of that cluster. 2: A bonus score is given to clusters containing intra-alignment signatures and inter-alignment signatures. Inter-alignment signatures are reads where part of the nucleotide sequence is inverted (which could be caused by an inversion). Intra-alignment signatures are reads where there are large gaps between the reads on the reference genome

(could be caused by insertions or deletions). 3: A score is given based on the standard deviation of the genomic position of the reads in the cluster, if the reads are all mapped roughly to the same genomic region, the cluster is considered more significant. 4: A score is given based on the standard deviation of the genomic span of the suspected SV in each read of the cluster, if all suspected SVs are roughly the same length the cluster is considered more significant. Summing up the scores of the four features we get a probability of how likely the cluster is caused by a SV. Trustworthy clusters has many reads in the cluster, has many intra-alignment signatures and inter-alignment signatures and has a small standard deviation of genomic position and span[26].
In phase 3, all clusters are classified into one of the six different SV classes recognized by SVIM, this classification is based directly on signatures in each cluster[26].

2.2.2 SVIM-asm

For SV calling of de novo assemblies SVIM-asm v1.0.2[70] is used. SVIM-asm expects a mapped, sorted and indexed assembly in .bam format[70]. SVIM-asm has two separate pipelines for haploid and diploid data. In this thesis only the haploid pipeline is used. SVIM-asm's haploid algorithm have minor changes compared the algorithm described for SVIM, however the overall idea is the same[70].

2.2.3 Truvari

For quality control of the structural variant callers Truvari v3.2.0[71] is used. The overall idea of Truvari consists of cross referencing two files containing information about structural variants. One of the files containing SVs are called the base set or truth set, this file contains information such as size of SV, type of SV, genomic location of SV and so on for known structural variants. The other file is the comparison file, it contains the same type of information but for structural variants generally found using a structural variant caller. The comparison file can then be cross referenced with the truth set, and in this way, you can get a quantitative assessment of the performance of the structural variant caller used to produce the comparison file. Structural variants are generally stored in variant call format files (.vcf). Since the same SV rarely is express 100% identical in the truth set and in the comparison set[71] (due to differences in sequencing and mapping), cross referencing can therefore not use identical pattern matching search[71]. When the Truvari algorithm cannot look for identical matches a question arises, how similar should two structural variants be, before they can be declared as the same SV? And furthermore, how do you compare two structural variants? Truvari compares structural variants on the following characteristics: 1) size similarity, the SVs length must be within 70% of each other to de considered identical[71]. 2) Reference distance, the SVs genomic location cannot be located further than 500 bps from each other to be considered identical[71]. 3) SV type, the SVs must be of the same type to be considered identical[71]. These three comparison statistics are easy to calculate, and when they have been calculated cross referencing the truth set, and the comparison file is trivial.

2.2.3.1 Truvari summary statistics

Truvari summarizes the performance of the structural variant callers, it does this by identifying true positives (TP), false positives (FP), and false negatives (FN). True positives are SVs both found in the comparison set and in the truth set. False positives are SVs found in the comparison set but not in the truth set. False negatives are SVs found in the truth set but not in the comparison set. The number of true positives, false positives, and false negatives can be used to calculate the SV caller's precision, recall and a f1 score. These metrics are calculated in the following way:

$Precision = \frac{TP}{TP+FP}$ [72]. This ratio shows how many of the SV calls in the comparison set is true positives.

$Recall = \frac{TP}{TP+FN}$ [72]. This ratio shows how many of the SVs in the truth set is identified in the comparison set.

$f1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$ [72]. As both precision and recall are important for assessing the performance of the SV caller, they are incorporated into a single score (f1) that can be used to compare different SV caller performances.

3 Experiments and results

3.1 Dataset

The sequence data used in this thesis is the EPI2ME dataset of Genome in a Bottle sample GM24385, updated October 2021, from here on out this dataset will be referred to as the GM24385 dataset. The data is available from an Amazon Web Services S3 bucket at: s3://ont-open-data/gm24385_q20_2021.10/

The sequence data is obtained using Oxford Nanopore Technologies PromethION sequencing device, Simplex Kit 12 sequencing chemistry and R10.4 flow cells[73], this sequencing technology performs 1D sequencing. The sequencing has been performed in four independent flow cells.

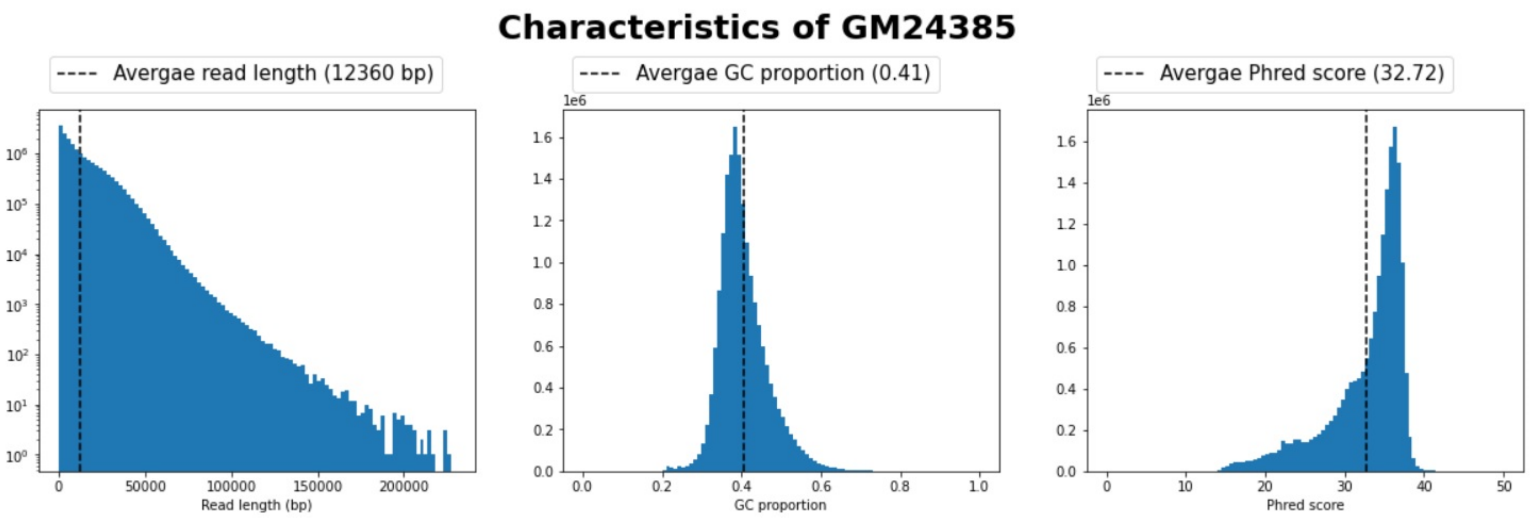
3.2 Raw sequencing reads

We use the human genome HG002 that is openly consented and a widely used genome. The PromethION sequencing device produces a sequencing_summary.txt file, this file can be used to for a quick overview of how the sequencing went, for this purpose pycoQC v0.11.9[74] is used. The FASTQ file produced by the basecaller contains the actual reads and is therefore also used for quality control, for this SAMtools v1.15[75], Seqkit v2.1.0[76] and Bioawk v1[77] are used. All four quality control programs give the same data characteristics, Bioawk and Python are used for the visualization seen in Figure 4. Please see the GitHub page with supplementary data for the output of the other QC programs.
The sequencing resulted in 18.125.023 reads, with read length ranging from 10 bp to 314.548 bp with an average read length of 12.360 nucleotides.
Before sequencing, all fragments longer than 35kb was depleted[73], in light of this an average read length just above 10 kilobases is a bit lower than expected but tolerable.

The average GC content across all reads are 0,41. Based on highly accurate Human reference genomes it is believed that the actual GC content of the human genome is 0,409[17].

We compute the sequencing error probability to $5,3*10^{-4}$, which is as expected and well within the accuracy of >99% as reported by ONT[78].

An article by Zhou et al[79] describes an experiment they have run to find out how depth of coverage affects read mapping and the accuracy of SV calling. They found that at least 20 times coverage was needed to produce accurate SV calls (judged by the f1 score). A depth of coverage of 72 times as we see on the raw reads is therefore very satisfactory as it lets us eliminate the shortest reads which theoretically should speed up the assembly.



Number of reads	Avergae read length [bp]	N50	Average Phred score	Average GC content	Depth of coverage
18.125.023	12.360	22.531	32,7	0,41	72

Figure 4 General characteristics for GM24385 sequence data. Figure created in Python

3.3 Shasta assembly

First Shasta assembly run used the newest standard Shasta configuration for ONT long read sequence data, which pr. May 2022 is --Nanopore-Oct2021 (see supplementary data for summary statistics). The L50 value is 4053 and N50 is 148.139, this indicates that the assembly contains a lot of short contigs, which can be caused by many factors. The genome fraction of 71% is lower than expected since we have a depth of coverage of 72 times. The low genome fraction and the unusual high number of short contigs leads us to believe that the assembly is fragmented with longer gaps between most contigs. By inspecting the assembly summary produced by the Shasta run we see that over 10.000.000 reads are discarded because they are shorter the set minim read length of 10.000 nucleotides. This means that over 50% of the input reads are discarded before the assembly starts, this explains the low genome fraction. It could also explain why the assembly is fragmented.

We therefore decided to run Shasta again with the --Nanopore-Oct2021 configuration but this time we manually set the minimum read length to 5.000 nucleotides. We see in the assembly summary (see supplementary data) that this resulted in roughly 35% of the reads being discarded. Lowering the minimum read length has therefore had the intended outcome of including more raw reads in the assembly process. However, if we look at the rest of the quality control metrics, we see that the overall quality of the assembly is much lower than for our first Shasta run. We see that the genome fraction is 68% and that the N50 value is now under 100.000 nucleotides. We can therefore conclude that the configuration change that we made did not have the intended outcome, we therefore need to tweak the configurations further.

For the third Shasta run there is two problems that needs to be addressed. The first problem is the fact that even after the minimum read length was lowered, over 35% of the reads are still discarded. Lowering the minimum read length is a tradeoff between including more reads, and thereby increasing the amount of input to the assembly algorithm, which should theoretically increase the genome fraction; and on the other hand, including more short reads which decreases the assembly precision and slows down the assembly algorithm. A minimum read length of 5.000 nucleotides is already short since long read sequenced data is used, and since the raw reads has a depth of coverage of 72 times, the minimum read length will not be lowered any further in this run. Other than an assembly FASTA file, a Shasta run also produces many diagnostic plots that can be used for further tweaking the assembly configurations. With help from the admin of the Shasta GitHub page[80, 81] we figured out that the default minimum and maxim values for the size of the hash bucket population potentially lowered the performance of the assembly. Shasta uses hash buckets for storing similar reads[64], only hash buckets containing a certain number of reads are marked as valid. This validity check ensures that buckets with few reads in them are discarded since the reads most likely are due to sequencing error; and on the other hand, buckets with a lot more reads then expected is most likely due to reads containing unrelated common repeats, these buckets are also discarded. To figure out the hash bucket population size Shasta crates a LowhashBucketHistogram.csv diagnostic file (available at supplementary data) which have been used to create the histogram seen in Figure 5.

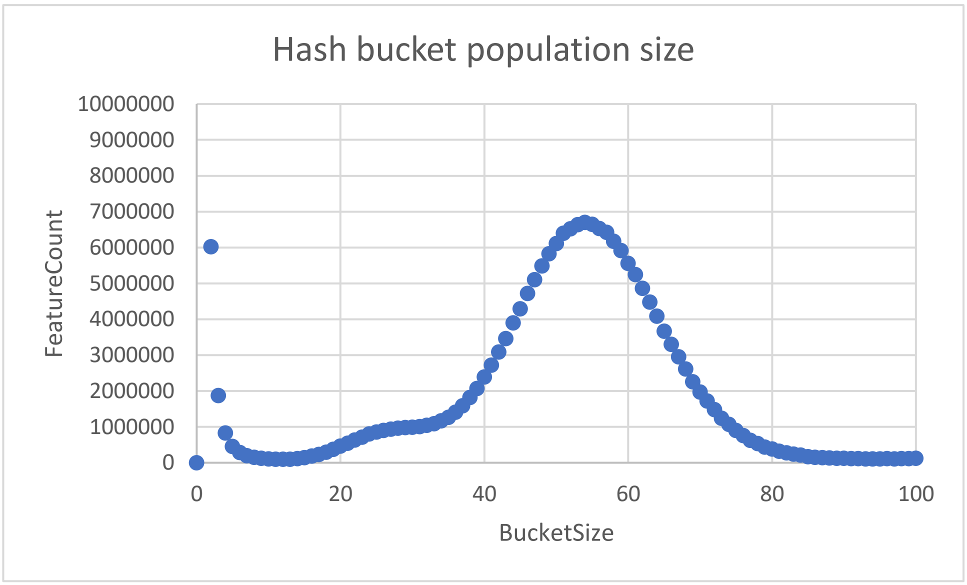


Figure 5 Diagnostic plot from the 2nd Shasta Run.
On the first axis is the number of reads in each bucket and on the second axis is the number of buckets containing x number of reads. Plot created in Microsoft Excel.

The default hash bucket size range is from 0-10[64]. In Figure 5 we see the healthy hash bucket population size from the 2nd Shasta run with a peak around a bucket size of 55 reads (the first peak is most likely caused by sequencing errors). Keeping in mind that all buckets containing more than 10 reads have been discarded in the first two Shasta runs, it is not surprising that the genome fraction is low since almost all healthy buckets were discarded. The sequencing resulted in a depth of coverage of 72 times, larges bucket sizes are therefore to be expected. For the third Shasta run the valid hash bucket range will be set to 30-70 items.

The third Shasta run resulted in a greatly improved assembly compared to the two first runs. The genome fraction is 96%, N50 is 7.307.892 bp, L50 is 124 and a total number of contigs of 4.333, this indicates that the assembly consists of few long contigs with a minimal number of gaps between each contig, where the earlier Shasta runs produced assemblies containing many short contigs with large parts of the genome not covered.

For the quality control of the Shasta assembly QUAST v5.0.2[65] is used, see supplementary data for the full report.

As mentioned in the background for the Shasta algorithm, the computation is very RAM intensive. As the server I have access to only have 500Gb of RAM I have run Shasta with '--memoryMode filesystem --memoryBacking disk' options which reads and writes to the servers storages as need be when the available RAM is running low[64], this comes with a very big performance penalty resulting in each assembly taking approximately 13 days to run, where the Shasta documentation claims a comparable dataset takes around 5 hours to run under optimal RAM conditions[64].

3.4 Genome mapping

The raw reads undergo read mapping as explained in the background section and the Shasta assemblies undergo assembly-to-assembly alignment. The output of Minimap2 (whether read mapping or assembly-to-assembly alignment has occurred) is a .sam/.bam file[63], this output file contains all the raw reads/contigs given as input to Minimap2 as well as a genomic location for all successfully mapped reads/contigs. These genomic locations have been calculated using a reference genome, and they indicate from where in the genome each read/contig originates. As reference genome GCA_000001405.15 GRCh38 reference genome sequence[82] is used. This older patch of the GRCh38 genome are chosen because this is the reference used by ONT in the GM24385 dataset, by using the same reference genome it is therefore possible to directly use the read mapping they performed on the raw ONT reads and compare it to the mapping I have performed on the Shasta assembly.

Genome mapping is notoriously difficult to quality control because the true genomic location for each read/contig is unknown, this means we do not have a truth set to compare our alignment against. A quick and very superficial way of checking whether the genome mapping procedures were successful is to find the percentage of successfully mapped reads/contigs, any major complications will result in a lower-than-expected success rate. We see that the read mapping and the Shasta assembly has a very high success rate of above 99%.

For the quality control of the genome mapping performed by Minimap2 SAMtools' flagstat command is used, see supplementary data for full summary.

3.5 Structural variant calling

Structural variant calling in the two pipelines resulted 30.280 structural variant candidates in the read mapping pipeline and 51.801 structural variant candidates in the de novo assembly pipeline.

	Deletion candidates	Inversion candidates	Insertions	Tandem duplication candidates	Novel insertion candidates	Total number of SV candidates
SVIM	12.860	15	17.405	0	0	30.280
SVIM-asm	16.957	34	34.810	0	0	51.801

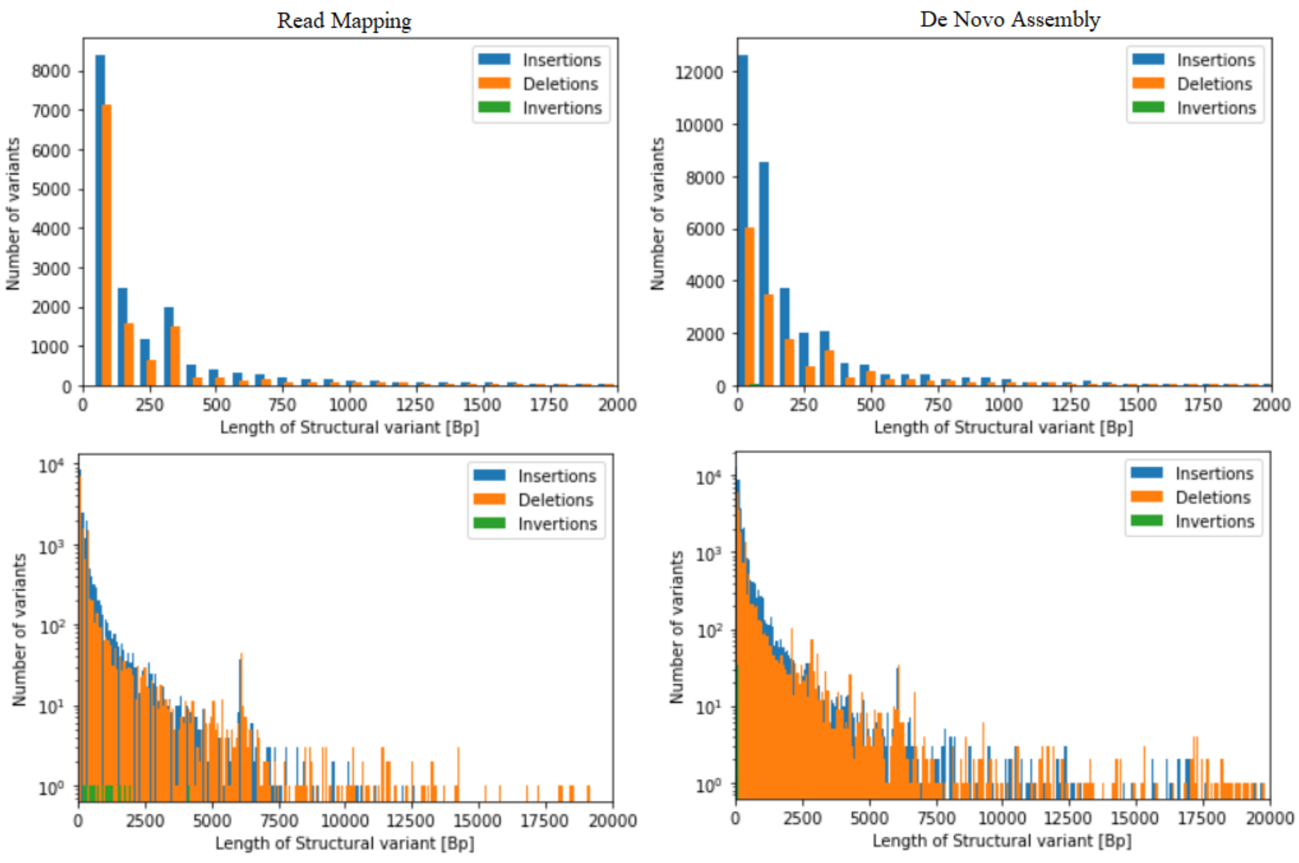


Figure 6
Structural variants found by SVIM-asm and SVIM.
On the left is SVs found by SVIM-asm on the Shasta assembly, on the right is SVs found by SVIM on the read mapped data. The y-axis of the two bottom plots is logarithmic.

Truvari scored the performance of the two SV callers, the newest HG002_NA24385_son (v4.2.1 SV) dataset has been used as truth set, which resulted in the following performance scores.

	Precision	Recall	F1
SVIM	0,926	0,963	0,944
SVIM-asm	0,913	0,750	0,823

For the de novo assembly a total of 51.801 structural variant candidates were found. SVIM found 121.506 structural variant candidates on the read mapped data. This means that SVIM predicts over 130% more structural variant then SVIM-asm on what is the exact same dataset, the only difference being the mapping method.
When SVIM SV calls, each predicted SV is accompanied with a quality score based on how many reads support the structural variant candidate and the Phred score. As default SVIM

returns all structural variants candidates no matter how many reads support the SV, filtering the SVs depending on the quality score is therefore beneficial. After a bit of testing, a minimum quality score of 15 was found to produce the best results, this is in the high end of what the SVIM guide recommends, but as the datasets has a genome coverage of 70 times, more reads are also expected to support each true SV.
After the filtering process SVIM found 30.280 structural variant candidates. Due to the small differences in the algorithms used by SVIM and SVIM-asm, SVIM-asm is not able to produce a quality score for each SV candidate, and filtering therefore is not an option. Which as we will see becomes an issue that creates false positives for the pipeline.

Plotting the structural variants found by SVIM and SVIM-asm in a histogram by SV length allows us to get on overall idea of the SV types found and their length, as well as visually comparing the two SV calling methods.
In Figure 6 we see a direct comparison between structural variant calling of a de novo assembly (on the right) and read mapped data (on the left). We see that the proportion of the five structural variant types (here insertions and novel element insertions have been grouped together, as seen in the peak of insertions around 5800 bp's of length) are quite similar across the two methods. We do however see that SVIM predicts more relatively short inversions, and SVIM-asm predicts more SVs above 15.000 bp of length.
Looking at Figure 6 provides an understanding of the SV candidates but does not provide any information of the structural variant caller's performance. In order to asse the performance of the callers and thereby the accuracy of the SV candidates Truvari is used. Truvari are run with custom configurations, which differs from the standard configurations in the following way; the configuration used changed the distance an SV is allowed to vary from the truth set from 500 bp to 5000 bp. Multi-matching is allowed, meaning a single SV in the truth set can be matched with multiple SVs from the comparison set. This allows for correcting errors in the structural variant caller that wrongly marked two identical SVs as different.
The structural variant calling using SVIM on the read mapped data resulted in a precision of ~0,93, meaning 93% of the SV calls performed by SVIM are accurate. A recall of ~0,96 means that SVIM identified 96% of all SVs in the truth set. Precision and recall can be combined into a single metric called the f1 score, SVIM had a f1 score of ~0,94 which is considered a good score.
SVIM-asm on the de novo assembled data has precision of just over 91%, correctly identifying 75% percent of the SVs in the truth set (recall), which results in a f1 score of 0,823. An f1 score over 0.8 is a decent score, but there is still room for improvement which mainly will come from improving recall.

3.6 Structural variant inspection in IGV

Big parts of the remaining analysis will rely on visual inspection of the structural variants and the reads for the corresponding region in the Integrative Genomics Viewer (IGV) by Broad Institute and the Regents of the University of California. Being able to read and understand the information in IGV is therefore important for understanding this thesis, which is why the basics are now be explained.

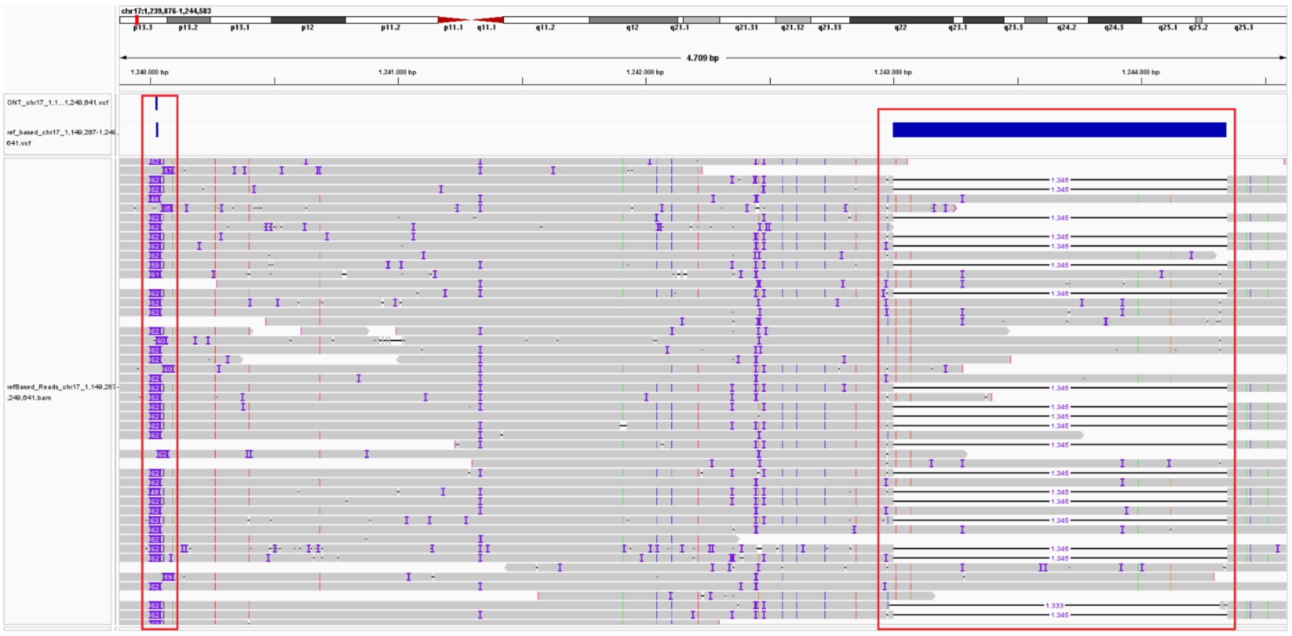

chr17:1,239,876-1,244,583
4,709 bp
1,240,000 bp
1,241,000 bp
1,242,000 bp
1,243,000 bp
1,244,000 bp
ONT_chr17_1.1...1,249,641.vcf
ref_based_chr17_1.149,287-1,249,641.vcf
refBased_Reads_chr17_1.149,287-1,249,641.bam

Figure 7 Here is the IGV representation of the chr17:1,239,876-1,244,583 region. Two structural variants are marked by red boxes. The left SV is an insertion, the right SV is a deletion.

Figure 7 shows a typical representation of an insertion and a deletion in IGV. The blue bars towards the top of the figure shows that a structural variant is present in the loaded vcf file, we see that two vcf files are loaded in Figure 7, the top vcf track shows SVs discovered by the de novo pipeline where the bottom vcf track shows SVs discovered by the read mapping pipeline, here we see the left SV is discovered by both pipelines where the right SV only is found in the read mapping pipeline. Under the vcf tracks the alignment track is shown, here we see the read alignments. Insertions are shown as purple regions as see in the left SV; deletions are shown as black horizontal bars as seen in the right SV. Completely white regions are gapped regions in the alignment.

Now that we know how to interpret the structural variants in IGV we can begin to analyze the structural variants found by the two pipelines.

3.6.1 False positives

By inspecting the false positives from the two pipelines, we notice that the two structural variant sets have 10 false positives in common. By inspecting some of these false positives and the reads covering the regions we can get an understanding of why the algorithm falsely identified the structural variants.

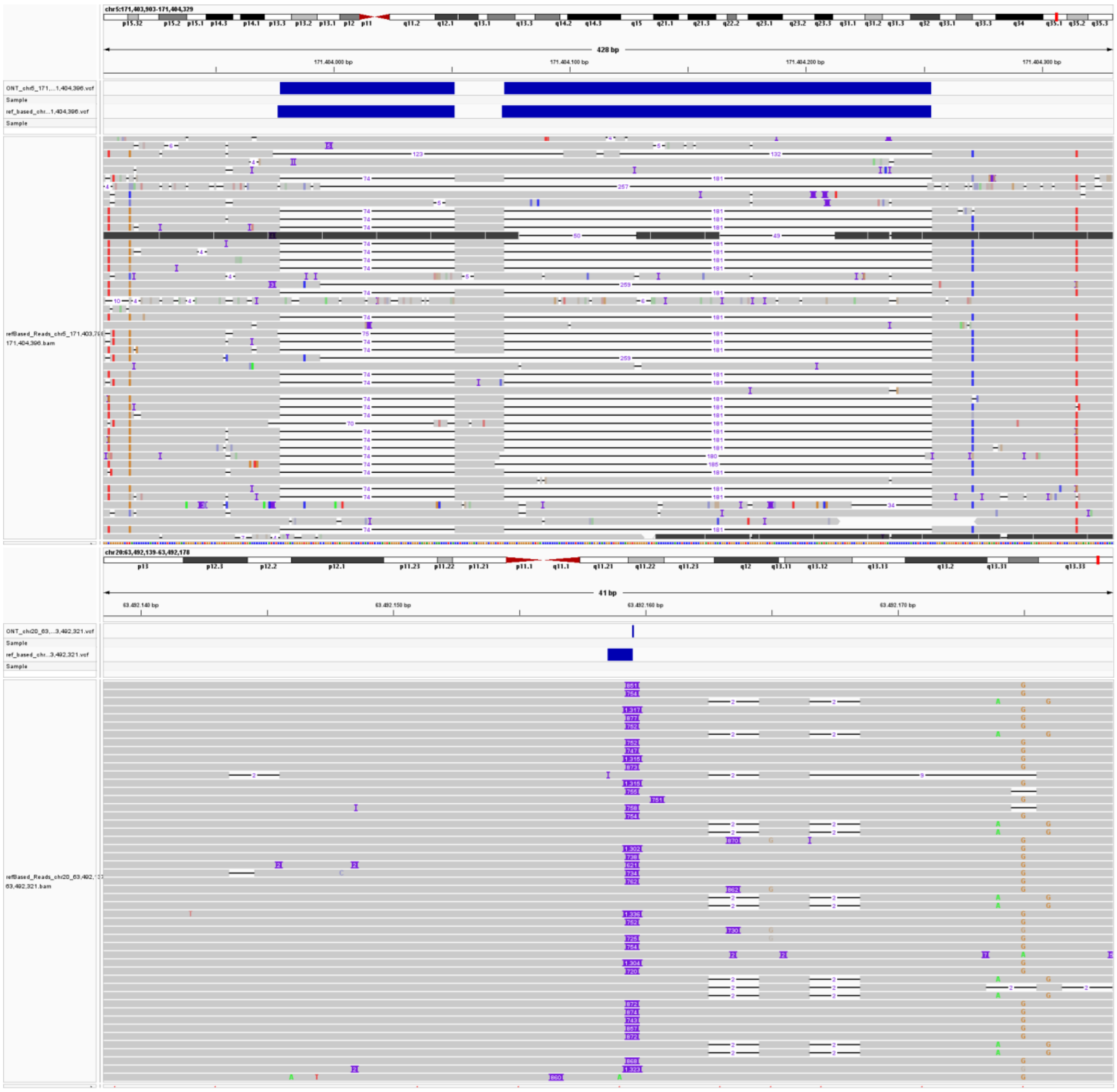

chr5:171,403,903-171,404,329
428 bp
ONT_chr5_171....1,404,396.vcf
Sample
ref_based_chr...1,404,396.vcf
Sample
refBased_Reads_chr5_171,403,798-171,404,396.bam
chr20:63,492,139-63,492,178
41 bp
ONT_chr20_63,...3,492,321.vcf
Sample
ref_based_chr...3,492,321.vcf
Sample
refBased_Reads_chr20_63,492,137-63,492,321.bam

Figure 8 SVs marked as false positives by Truvari.
Top panel: Two deletions in the chr:5171,403,798-171,404,396 region.
Bottom panel: An insertion in the chr:2063,492,137-63,492,321 region.
Figure created with IGV

In Figure 8 we see that the reads indicate the 3 SVs actually are true positives, and not false positives as they were marked by Truvari. We come to this conclusion because the majority of the read alignments contains the structural variant. This error occurs because the SV are not present in the truth set. As shown, this however does not mean that the SV is not present in the raw reads.

This stands to show that there is a lot of information to gain from visually inspecting the structural variants. The next step is therefore to find and inspect structural variants that are inconsistent across the two pipelines, this will allow us to understand which types of SVs the two pipelines struggle with.

3.6.2 Inconsistent structural variants

We will start by focusing on the structural variants found by the de novo assembly pipeline but not by the read mapping pipeline. These structural variants can either be true structural variants or false positives. By looking at these inconsistent structural variants we will get an understanding of the strength and weaknesses of the de novo assembly pipeline.
Many of the true positives identified by the de novo pipeline but not by the read mapping pipeline are short insertions.

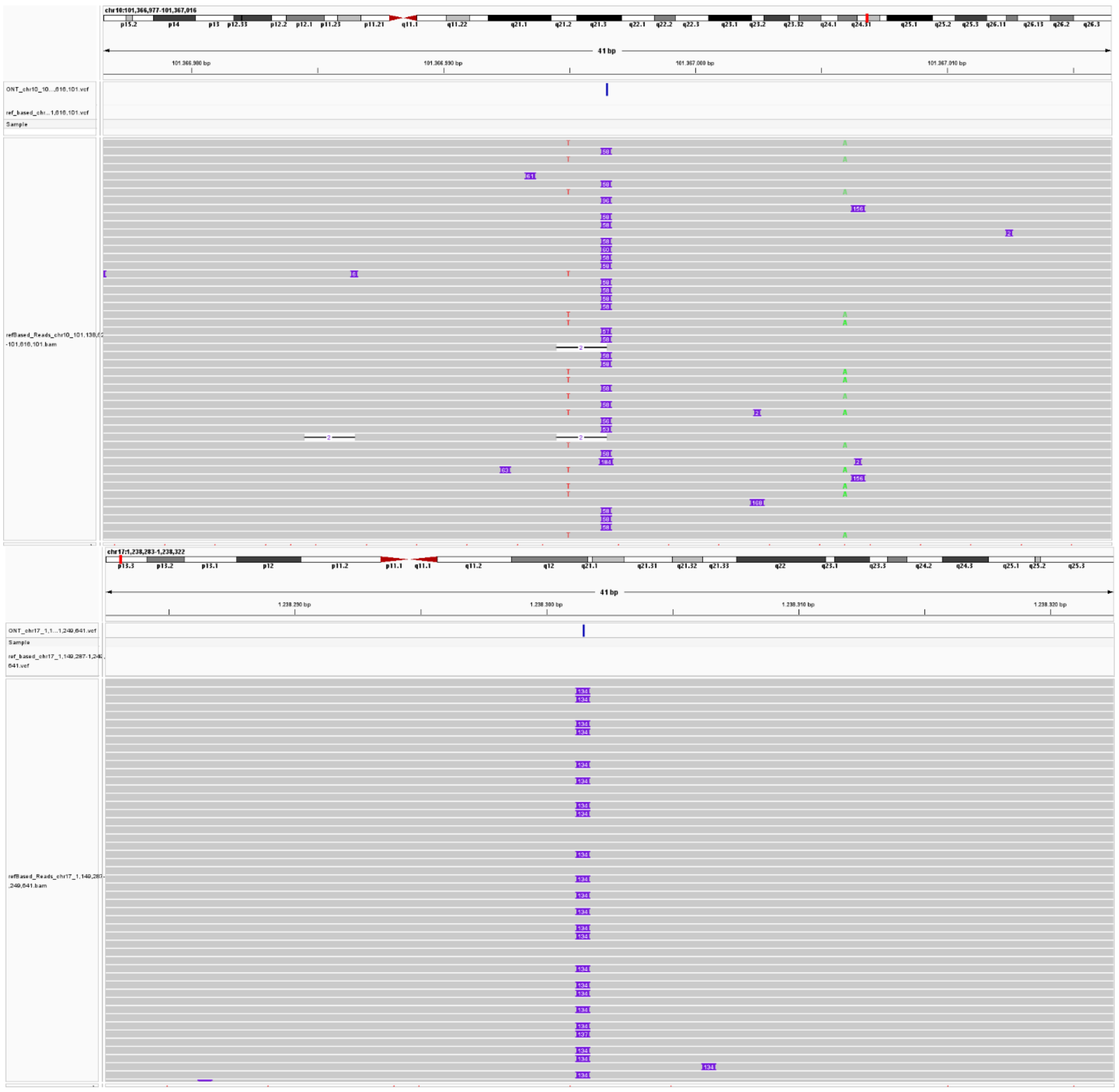


Figure 9 Two examples of typical true structural variants only identified by the de novo assembly pipeline. Figure created with IGV

In Figure 9 we see two examples of a typical SV only identified by the de novo pipeline. It is short insertions (generally under 150 bp) in regions with good coverage and non-complex read alignments.

As mentioned SVIM-asm structural variant candidates does not have a quality score upon which they can be filtered, which becomes obvious by looking at the pipeline's unique false positives.

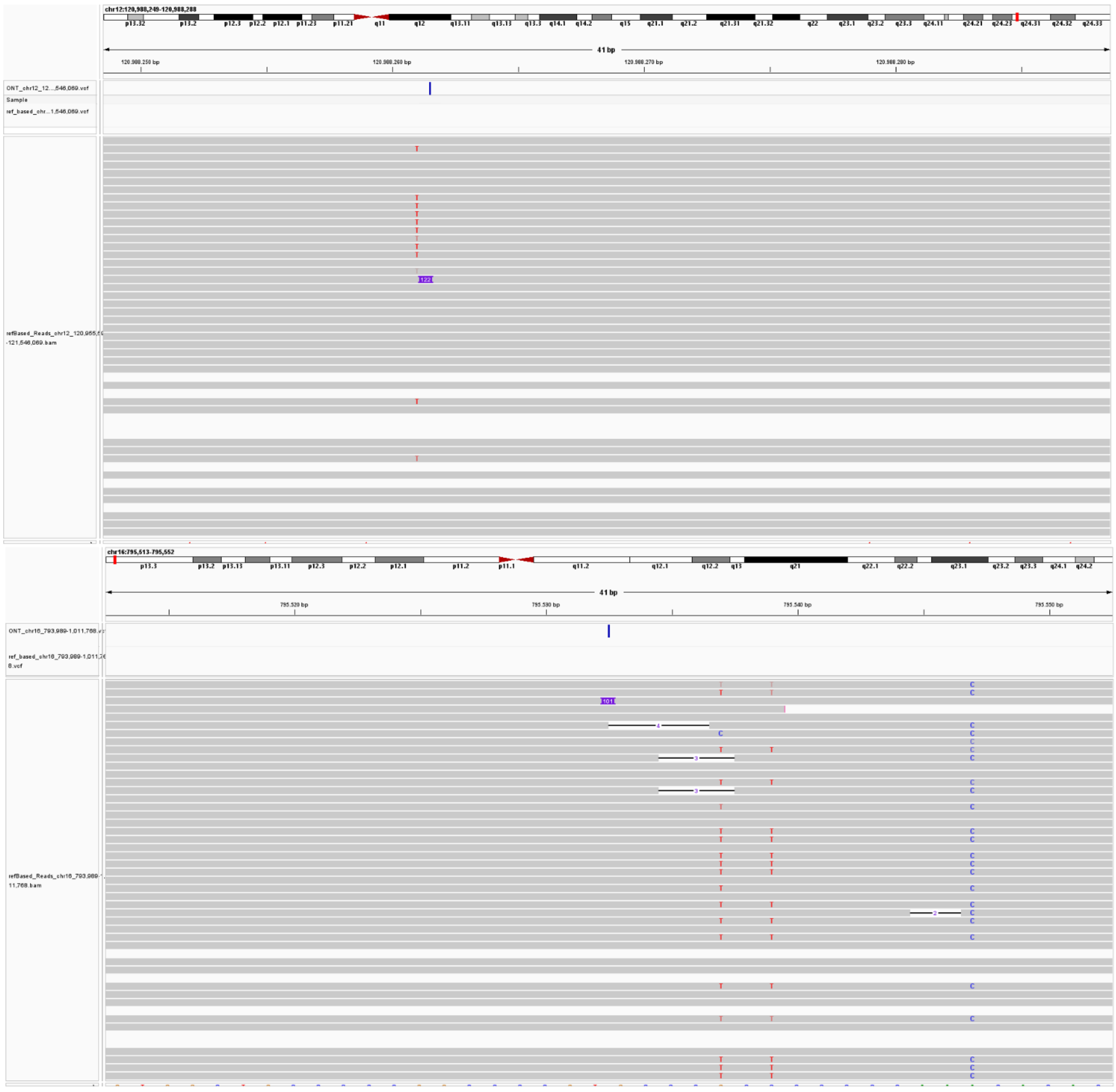

chr12:120,988,249-120,988,288
41 bp
ONT_chr12_12....546,069.vcf
Sample
ref_based_chr...1,546,069.vcf
refBased_Reads_chr12_120,955,5
-121,546,069.bam
chr16:795,513-795,552
41 bp
ONT_chr16_793,989-1,011,768.vc
ref_based_chr16_793,989-1,011,76
8.vcf
refBased_Reads_chr16_793,989-1,
11,768.bam

Figure 10 Typical unique false positives by the de novo assembly pipeline. Figure created with IGV

As seen in Figure 10 SV candidates can be based upon a single read. This means that all sequencing or alignment errors (longer than 50 bp) will be identified as a SV candidate, this problem is most common with insertions but the problem have also been detected with deletions. These structural variant candidates supported by one or very few reads are rarely if ever true structural variants, which means that this issue is basis for most of the pipeline's false positives. Despite this issue, the pipeline still has a precision above 90%, but it most likely could be significantly improved if a filtering mechanism is added to SVIM-asm.

Let us now shift focus to the read mapping pipeline. We will again look at the correct and wrongly predicted structural variants in order to draw conclusion about the pipeline's strengths and weaknesses.
Tightly clustered structural variants can be difficult to predict by the SV callers because the alignments in these regions can become quite complex.

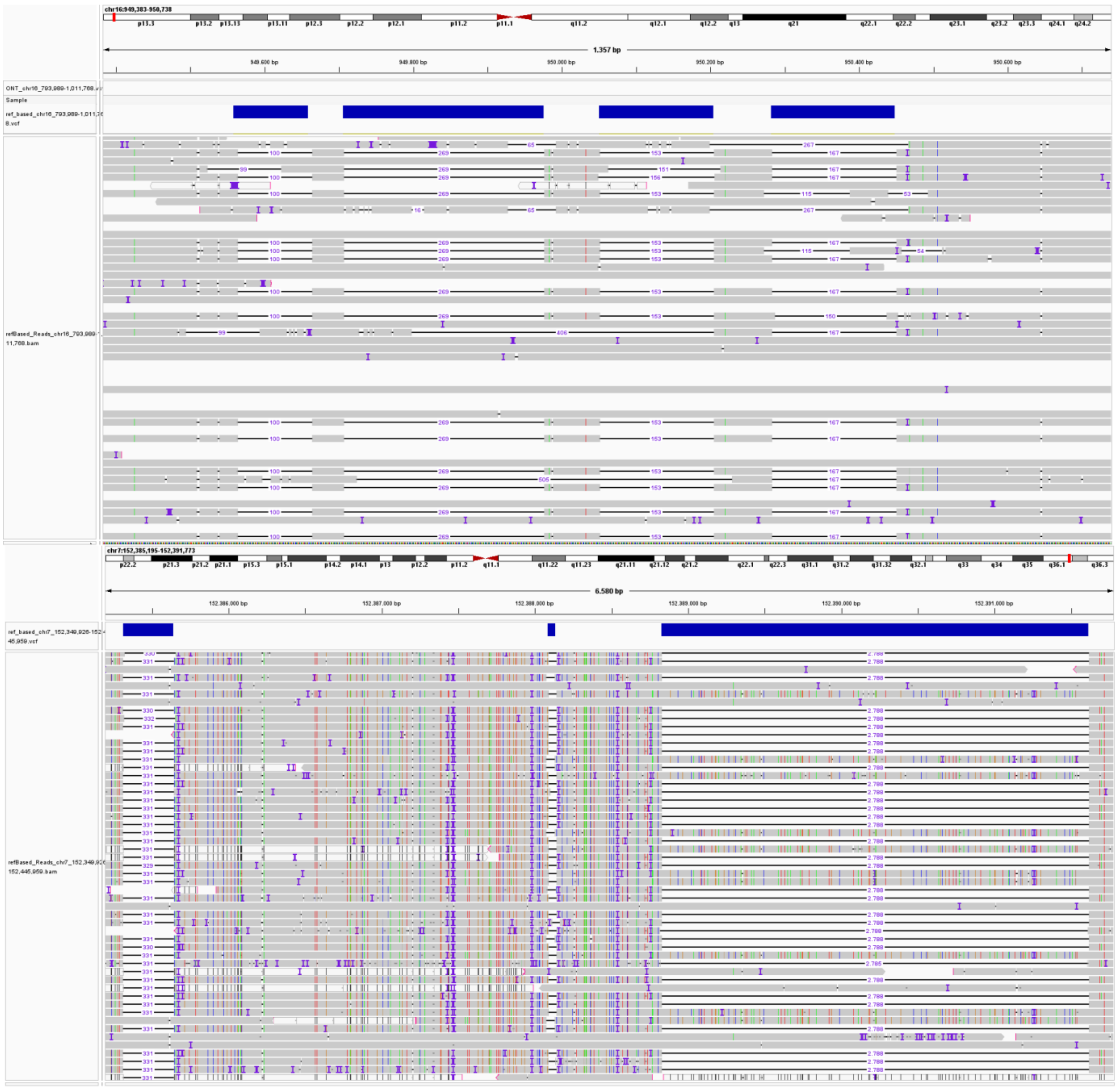


Figure 11 Typical true structural variants in the read mapping pipeline. Figure created with IGV

In Figure 11 we see deletions clustered together within the same genomic region. The read mapping pipeline correctly identifies all SVs in the region, this is most likely explained by the core differences between the two pipelines, the more direct path from sequencing to SV calling is in this scenario beneficial, since the very complex regions tends to be skipped or wrongly assembled by Shasta, resulting in wrong SV calls.

The read mapping pipeline struggles to predict the length of short mutations, resulting in mutations below the 50 bp cutoff being wrongly identified as structural variants.

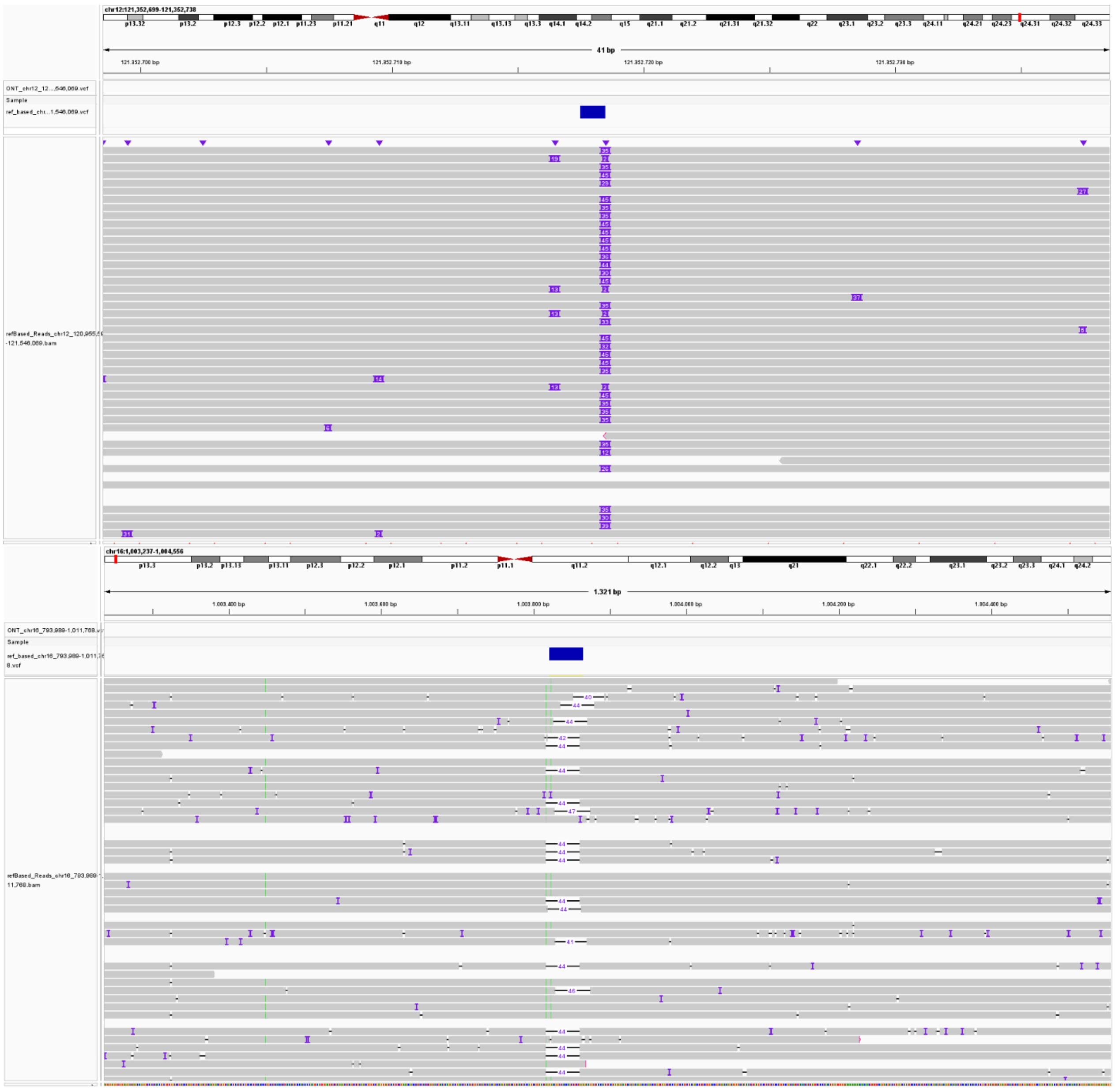


Figure 12 Typical unique false positives int he read mapping pipeline. Figure created with IGV

The top SV in Figure 12 is 45 bp long and thus should not be identified as a SV, however SVIM has marked the regions as an insertion of length 99. Similarly, the bottom SV is a deletion of length 44, but SVIM marks it as a deletion of length 112. This problem is a fundamental issue with the SVIM algorithm and is therefore a source for false positives that must be accepted when using the program. However, the problem can be fixed by raising the minimum length of what is considered a structural variant.

4 Discussion

This thesis set out to create and benchmark two bioinformatic pipelines for structural variant calling. More precisely, the two pipelines used a de novo assembly approach and a read mapping approach. Through benchmarking it has been shown in this thesis that the presented read mapping pipeline scores higher in all three performance scores used (precision, recall and f1). The de novo assembly pipeline falls relatively short when it comes to recall, demonstrating a lack of sensitivity. The read mapping pipeline excels in complex genomic regions where structural variants are tightly clustered, it however fails to accurately predict short structural variants. On the other hand, the de novo assembly pipeline accurately predicts short structural variants. Being able to filter the structural variant candidates based on a quality score after using SVIM is a big advantage and something SVIM-asm needs to implement in order to avoid the low quality structural variant candidate problem earlier described. These strength and weakness assumptions of the two pipelines are based on relatively few examples. Further analysis of the structural variants found by the two pipelines are therefore need in order draw genome wide conclusions.

More comprehensive truth sets spanning larger regions of the human genome than what is currently available, are needed for better benchmarking. Especially the recall scores have a vague importance, since around half of the structural variant candidates are in genomic regions not covered by the truth set, and are therefore not factored in during the benchmarking.
The current Shasta configurations needs a lot of manually tweaking in order produce good assemblies, which is a big disadvantage to the usability of the pipeline. Shasta is regularly updated with new configurations directly aimed at improving the assembly performance of long ONT reads, continuing the configuration updates is a vital part of improving the de novo pipeline to get it on par with the reference-based pipeline. The programs used in the two pipelines have been chosen to create simple and good performing pipelines that are future proof, this means that there likely are other configurations to the chosen programs or other programs entirely that will perform better on a case-by-case basis.

The pipelines in this thesis aims to present a simple step by step workflow for structural variant calling. Such a workflow is needed in order to promote further research in the area. Structural variation research is needed in order to create a comprehensive understanding of the mechanisms behind cancer and other genetic diseases. Cancer research is shifting from a treatment approach to a preventative approach[83], understanding the biological mechanisms behind structural variation is a key factor in the preventative approach. Cancer prevention is needed as it is the single most destructive and expansive disease in modern society[84].

A more in-depth analysis of the structural variant candidates is required to create more concise genome wide conclusion of the two pipelines. This requires more time to analyze more structural variant candidates as well as testing the workflows with multiple input datasets. This more furrow benchmarking of the workflows would ensure the stability of the pipelines across a more varied range of inputs. Future research would also benefit from a comprehensive benchmarking of the most obvious competitors to each of the programs used, and from this, choose the best performing alternative to each program. The programs used has been pre-selected based on criteria mentioned earlier in the thesis, benchmarking alternative programs for each step in the workflow would therefore ensure that the pipelines perform the best possible across a range of tested programs, and not only within the programs that has been pre-selected.

References

1. Bryant Jeri L, Boughter John D, Gong Suzhen, LeDoux Mark S HDH (2010) The landscape of somatic copy-number alteration across human cancers. Physiol Behav 32:41–52. https://doi.org/10.1038/nature08822

2. Hartwell L, Goldberg M, Fischer J, et al (2018) The Genetics of Cancer. In: GENETICS: FROM GENES TO GENOMES, Sixth. McGraw-Hill Education, New York, pp 681–712

3. Cancerregisteret (2022) Cancer nøgletal. https://www.cancer.dk/hjaelp-viden/fakta-om-kraeft/kraeft-i-tal/nogletal/. Accessed 24 May 2022

4. Alberts B, Hokin K, Johnson A, et al (2019) Control Of Cell Numbers And Cell Size. In: Twitchell B, Morales M (eds) Essential Cell Biology, Fifth. W. W. Norton & Company, New York & London, pp 639–646

5. Friedman LS, Ostermeyer EA, Szabo CI, et al (1994) Confirmation of BRCA1 by analysis of germline mutations linked to breast and ovarian cancer in ten families. Nat Genet 8:399–404. https://doi.org/10.1038/ng1294-399

6. Yi K, Ju YS (2018) Patterns and mechanisms of structural variations in human cancer. Exp Mol Med 50:. https://doi.org/10.1038/s12276-018-0112-3

7. Stephens PJ, Greenman CD, Fu B, et al (2011) Massive genomic rearrangement acquired in a single catastrophic event during cancer development. Cell 144:27–40. https://doi.org/10.1016/j.cell.2010.11.055

8. Sebat J, Lakshmi B, Malhotra D, et al (2007) Strong Association of De Novo Copy Number Mutations with Autism Interdepartmental Program in the Neurosciences, Program in. Science (80-) 316:445–449. https://doi.org/10.1126/science.1138659

9. NCBI XT oxytocin/neurophysin I prepropeptide [Homo sapiens (human)]. https://www.ncbi.nlm.nih.gov/gene/5020. Accessed 10 Mar 2022

10. Kosfeld M, Heinrichs M, Zak PJ, et al (2005) Oxytocin increases trust in humans. Nature 435:673–676. https://doi.org/10.1038/nature03701

11. Ferguson JN, Young LJ, Hearn EF, et al (2000) Social amnesia in mice lacking the oxytocin gene. Nat Genet 25:284–288. https://doi.org/10.1038/77040

12. Sudmant PH, Rausch T, Gardner EJ, et al (2015) An integrated map of structural variation in 2,504 human genomes. Nature 526:75–81. https://doi.org/10.1038/nature15394

13. Collins RL, Brand H, Karczewski KJ, et al (2020) A structural variation reference for medical and population genetics. Nature 581:444–451. https://doi.org/10.1038/s41586-020-2287-8

14. Hartwell L, Goldberg M, Fischer J, et al (2018) Genetics: The Study of Biological Information. In: GENETICS: FROM GENES TO GENOMES, Sixth. McGraw-Hill Education, New York & London, pp 1–11

15. WATSON JD, CRICK FHC (1953) Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid. Nature 171:737–738. https://doi.org/10.1038/171737a0

16. National Human Genome Research Institute (2019) Introduction to Genomics.

https://www.genome.gov/About-Genomics/Introduction-to-Genomics. Accessed 24 May 2022

17. Piovesan A, Pelleri MC, Antonaros F, et al (2019) On the length, weight and GC content of the human genome. BMC Res Notes 12:1–7. https://doi.org/10.1186/s13104-019-4137-z

18. Hartwell L, Goldberg M, Fischer J, et al (2018) DNA Structure, Replication, and Recombination. In: GENETICS: FROM GENES TO GENOMES, Sixth. McGraw-Hill Education, New York, pp 181–218

19. Syvänen AC (2001) Accessing genetic variation: Genotyping single nucleotide polymorphisms. Nat Rev Genet 2:930–942. https://doi.org/10.1038/35103535

20. Feuk L, Carson AR, Scherer SW (2006) Structural variation in the human genome. Nat Rev Genet 7:85–97. https://doi.org/10.1038/nrg1767

21. Cao H, Hastie AR, Cao D, et al (2014) Rapid detection of structural variation in a human genome using nanochannel-based genome mapping technology. Gigascience 3:1–11. https://doi.org/10.1186/2047-217X-3-34

22. Freeman JL, Perry GH, Feuk L, et al (2006) Copy number variation: New insights in genome diversity. Genome Res 16:949–961. https://doi.org/10.1101/gr.3677206

23. Pang AW, MacDonald JR, Pinto D, et al (2010) Towards a comprehensive structural variation map of an individual human genome. Genome Biol 11:. https://doi.org/10.1186/gb-2010-11-5-r52

24. Mendivil Ramos O, Ferrier DEK (2012) Mechanisms of Gene Duplication and Translocation and Progress towards Understanding Their Relative Contributions to Animal Genome Evolution. Int J Evol Biol 2012:1–10. https://doi.org/10.1155/2012/846421

25. National Center for Biotechnology Information (NCBI) Overview of Structural Variation. https://www.ncbi.nlm.nih.gov/dbvar/content/overview/#:~:text=Introduction,copy number variants (CNVs). Accessed 3 Mar 2022

26. Heller D, Vingron M (2019) SVIM: Structural variant identification using mapped long reads. Bioinformatics 35:2907–2915. https://doi.org/10.1093/bioinformatics/btz041

27. Alberts B, Hokin K, Johnson A, et al (2019) How Genes and Genomes Evolve. In: Twitchell B, Morales M (eds) Essential Cell Biology, Fifth. W. W. Norton & Company, New York & London, pp 297–328

28. Alberts B, Hokin K, Johnson A, et al (2019) Dna Repair. In: Twitchell B, Morales M (eds) Essential Cell Biology, Fifth. W. W. Norton & Company, New York & London, pp 215–224

29. Beckmann JS, Estivill X, Antonarakis SE (2007) Copy number variants and genetic traits: Closer to the resolution of phenotypic to genotypic variability. Nat Rev Genet 8:639–646. https://doi.org/10.1038/nrg2149

30. Lee JA, Carvalho CMB, Lupski JR (2007) A DNA Replication Mechanism for Generating Nonrecurrent Rearrangements Associated with Genomic Disorders. Cell

131:1235–1247. https://doi.org/10.1016/j.cell.2007.11.037

31. Moore JK, Haber JE (1996) Cell cycle and genetic requirements of two pathways of nonhomologous end-joining repair of double-strand breaks in Saccharomyces cerevisiae. Mol. Cell. Biol. 16:2164–2173

32. Manuscript A, Process S (2013) Chromothripsis and Human Disease: Piecing Together the Shattering Process. Cell 148:29–32. https://doi.org/10.1016/j.cell.2012.01.006.Chromothripsis

33. Tsai AG, Lieber MR (2010) Mechanisms of chromosomal rearrangement in the human genome. BMC Genomics 11

34. Hartwell L, Goldberg M, Fischer J, et al (2018) DNA sequencing. In: GENETICS: FROM GENES TO GENOMES, Sixth. McGraw-Hill Education, New York, pp 327–333

35. Sofi MY, Shafi A, Masoodi KZ (2022) Chapter 2 - Advances in DNA sequencing. In: Sofi MY, Shafi A, Masoodi KZ (eds) Bioinformatics for Everyone, First. Academic Press, pp 9–16

36. Yang A, Zhang W, Wang J, et al (2020) Review on the Application of Machine Learning Algorithms in the Sequence Data Mining of DNA. Front. Bioeng. Biotechnol. 8

37. Heather JM, Chain B (2016) The sequence of sequencers: The history of sequencing DNA. Genomics 107:1–8

38. Voelkerding K V., Dames SA, Durtschi JD (2009) Next-generation sequencing:from basic research to diagnostics. Clin Chem 55:641–658. https://doi.org/10.1373/clinchem.2008.112789

39. Rothberg JM, Leamon JH (2008) The development and impact of 454 sequencing. Nat Biotechnol 26:1117–1124. https://doi.org/10.1038/nbt1485

40. Bleidorn C (2016) Third generation sequencing: Technology and its potential impact on evolutionary biodiversity research. Syst Biodivers 14:1–8. https://doi.org/10.1080/14772000.2015.1099575

41. Segerman B (2020) The Most Frequently Used Sequencing Technologies and Assembly Methods in Different Time Segments of the Bacterial Surveillance and RefSeq Genome Databases. Front Cell Infect Microbiol 10:1–7. https://doi.org/10.3389/fcimb.2020.527102

42. Clark DP, Pazdernik NJ, McGehee MR (2019) DNA Sequencing. In: Molecular Biology, Third. Academic Press, London, pp 240–268

43. Deamer D, Akeson M, Branton D (2016) Three decades of nanopore sequencing. Nat Biotechnol 34:518–524. https://doi.org/10.1038/nbt.3423

44. Jain M, Olsen HE, Paten B, Akeson M (2016) The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. Genome Biol 17:1–11. https://doi.org/10.1186/s13059-016-1103-0

45. Lu H, Giordano F, Ning Z (2016) Oxford Nanopore MinION Sequencing and Genome Assembly. Genomics, Proteomics Bioinforma 14:265–279.

https://doi.org/10.1016/j.gpb.2016.05.004

46. Wick RR, Judd LM, Holt KE (2019) Performance of neural network basecalling tools for Oxford Nanopore sequencing. Genome Biol 20:1–10. https://doi.org/10.1186/s13059-019-1727-y

47. Tyler AD, Mataseje L, Urfano CJ, et al (2018) Evaluation of Oxford Nanopore's MinION Sequencing Device for Microbial Whole Genome Sequencing Applications. Sci Rep 8:1–12. https://doi.org/10.1038/s41598-018-29334-5

48. National Center for Genome Analysis Support Introduction to ONT Nanopore Sequencing. In: INDIANA Univ. Bloom. https://blogs.iu.edu/ncgas/2021/11/29/introduction-to-ont-nanopore-sequencing/. Accessed 17 Mar 2022

49. Ewing B, Green P (1998) Base-Calling of Automated Sequencer Traces Using Phred. I. Accuracy Assessment. Genome Res 8:186–194. https://doi.org/10.1101/gr.8.3.186

50. Ewing B, Green P (1998) Base-calling of automated sequencer traces using phred. II. Error probabilities. Genome Res 8:186–194. https://doi.org/10.1101/gr.8.3.186

51. Port E, Sun F, Martin D, Waterman MS (1995) Genomic mapping by end-characterized random clones: a mathematical analysis. Genomics 26:84–100. https://doi.org/10.1016/0888-7543(95)80086-2

52. Heller D, Vingron M, Church G, et al (2020) SDip: A novel graph-based approach to haplotype-aware assembly based structural variant calling in targeted segmental duplications sequencing. bioRxiv 2020.02.25.964445. https://doi.org/https://doi.org/10.1101/2020.02.25.964445

53. Garg S (2021) Computational methods for chromosome-scale haplotype reconstruction. Genome Biol 22:1–24. https://doi.org/10.1186/s13059-021-02328-9

54. Loh P, Danecek P, Palamara PF, et al (2017) Reference-based phasing using the Haplotype Reference Consortium panel. 48:1443–1448. https://doi.org/10.1038/ng.3679.Reference-based

55. Schbath S, Martin V, Zytnicki M, et al (2012) Mapping reads on a genomic sequence: An algorithmic overview and a practical comparative analysis. J. Comput. Biol. 19:796–813

56. Fonseca NA, Rung J, Brazma A, Marioni JC (2012) Tools for mapping high-throughput sequencing data. Bioinformatics 28:3169–3177. https://doi.org/10.1093/bioinformatics/bts605

57. Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics 25:1754–1760. https://doi.org/10.1093/bioinformatics/btp324

58. Shafin K, Pesout T, Lorig-Roach R, et al (2020) Nanopore sequencing and the Shasta toolkit enable efficient de novo assembly of eleven human genomes. Nat Biotechnol 38:1044–1053. https://doi.org/10.1038/s41587-020-0503-6

59. Myers EW (2005) The fragment assembly string graph. Bioinformatics 21

60. Guidi G, Selvitopi O, Ellis M, et al (2021) Parallel string graph construction and

transitive reduction for de novo genome assembly. Proc - 2021 IEEE 35th Int Parallel Distrib Process Symp IPDPS 2021 517–526. https://doi.org/10.1109/IPDPS49936.2021.00060

61. Sedlazeck FJ, Lee H, Darby CA, Schatz MC (2018) Piercing the dark matter: Bioinformatics of long-range sequencing and mapping. Nat Rev Genet 19:329–346. https://doi.org/10.1038/s41576-018-0003-4

62. Lamb JC, Birchler JA (2003) The role of DNA sequence in centromere formation. Genome Biol 4:. https://doi.org/10.1186/gb-2003-4-5-214

63. Li H (2018) Minimap2: Pairwise alignment for nucleotide sequences. Bioinformatics 34:3094–3100. https://doi.org/10.1093/bioinformatics/bty191

64. Initiative CZ Shasta Documentation. https://chanzuckerberg.github.io/shasta/. Accessed 11 Mar 2022

65. Gurevich A, Saveliev V, Vyahhi N, Tesler G (2013) QUAST: Quality assessment tool for genome assemblies. Bioinformatics 29:1072–1075. https://doi.org/10.1093/bioinformatics/btt086

66. Audano PA, Sulovari A, Graves-Lindsay TA, et al (2019) Characterizing the Major Structural Variant Alleles of the Human Genome. Cell 176:663-675.e19. https://doi.org/10.1016/j.cell.2018.12.019

67. Sedlazeck FJ, Rescheneder P, Smolka M, et al (2018) Accurate detection of complex structural variations using single-molecule sequencing. Nat Methods 15:461–468. https://doi.org/10.1038/s41592-018-0001-7

68. Nattestad M, Goodwin S, Ng K, et al (2018) Complex rearrangements and oncogene amplifications revealed by long-read DNA and RNA sequencing of a breast cancer cell line. Genome Res 28:1126–1135. https://doi.org/10.1101/gr.231100.117

69. Mahmoud M, Gobet N, Cruz-Dávalos DI, et al (2019) Structural variant calling: The long and the short of it. Genome Biol 20:1–14. https://doi.org/10.1186/s13059-019-1828-7

70. Heller D, Vingron M (2020) SVIM-asm: Structural variant detection from haploid and diploid genome assemblies. Bioinformatics 36:5519–5521. https://doi.org/10.1093/bioinformatics/btaa1034

71. English AC, Menon VK, Gibbs R, et al (2022) Truvari : Refined Structural Variant Comparison Preserves Allelic Diversity. Life Sci Wkly 7464. https://doi.org/10.1101/2022.02.21.481353

72. Powers DMW (2007) Evaluation : From Precision , Recall and F-Factor to ROC , Informedness , Markedness & Correlation. J Mach Learn Technol 2:37–63. https://doi.org/10.48550/arXiv.2010.16061

73. EPI2ME GM24385 dataset. https://labs.epi2me.io/gm24385_q20_2021.10/. Accessed 23 Feb 2022

74. Leger A, Leonardi T (2019) pycoQC, interactive quality control for Oxford Nanopore Sequencing. J Open Source Softw 4:1236. https://doi.org/10.21105/joss.01236

75. Danecek P, Bonfield JK, Liddle J, et al (2021) Twelve years of SAMtools and

BCFtools. Gigascience 10:1–4. https://doi.org/10.1093/gigascience/giab008

76. Shen W, Le S, Li Y, Hu F (2016) SeqKit: A cross-platform and ultrafast toolkit for FASTA/Q file manipulation. PLoS One 11:1–10. https://doi.org/10.1371/journal.pone.0163962

77. Li H bioawk. version: 1; https://github.com/lh3/bioawk. Accessed 22 Feb 2022

78. Oxfor Nanopore Tech New nanopore sequencing chemistry in developers' hands; set to deliver Q20+ (99%+) "raw read" accuracy. https://nanoporetech.com/about-us/news/new-nanopore-sequencing-chemistry-developers-hands-set-deliver-q20-99-raw-read. Accessed 22 Apr 2022

79. Zhou A, Lin T, Xing J (2019) Evaluating nanopore sequencing data processing pipelines for structural variation identification. Genome Biol 20:1–13. https://doi.org/10.1186/s13059-019-1858-1

80. Shasta GitHub Issue #273. https://github.com/chanzuckerberg/shasta/issues/273. Accessed 22 Apr 2022

81. Shasta GitHub Issue #285. https://github.com/chanzuckerberg/shasta/issues/285. Accessed 22 Apr 2022

82. National Library of Medicine GRCh38 reference genome. https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.26/. Accessed 12 May 2022

83. Valle I, Tramalloni D, Bragazzi NL (2015) Cancer prevention: State of the art and future prospects. J. Prev. Med. Hyg. 56:21–27

84. Yabroff KR, Lund J, Kepka D, Mariotto A (2011) Economic burden of cancer in the United States: Estimates, projections, and future research. Cancer Epidemiol Biomarkers Prev 20:2006–2014. https://doi.org/10.1158/1055-9965.EPI-11-0650

Supplementary data

Please find all supplementary data file at my GitHub repository: https://github.com/GargGroup/SVasmcaller/blob/Structural-variant-calling-in-Human-genomes/README.md

Software commands

All software used is run through a Linux terminal. All the required programs are free and can be installed using Python3's pip install command or in a conda environment.

```
>	python3 -m pip install <package name>
```

or

```
>	conda install <package name>
```

Quality control of raw sequencing reads

As mentioned, the sequencing was run over four flow cells, four independent fastq files and sequencing_summary files are therefore created. Regular expression matching can be used to give the following programs multiple input files without giving the precise location for all the files.

```
>	pycoQC -f <sequencing_summary.txt> -o <output.html>

>	bioawk -t -c fastx '{print $name,gc($seq),meanqual($qual),length($seq) }'
	<input.fastq> <output.txt>

>	seqkit stats -a <input.fastq> -T > <output.txt>

>	samtools stats <input.fastq> > <output.txt>
```

Shasta Assembly

As later descripted in the discussion I experimented with different Shasta configurations, the commands used for each run is shown below.

First Shasta:

```
>	shasta-Linux-0.8.0 --input <input.fastq> --config Nanopore-Oct2021
	--assemblyDirectory <output location> --threads <thread count of CPU>
```

Second Shasta run:

```
>	shasta-Linux-0.8.0 --input <input.fastq> --config Nanopore-Oct2021
	--assemblyDirectory <output location> --threads <thread count of CPU> --
	assemblyDirectory shatsta_5kb --Reads.minReadLength 5000
```

Third Shasta run:

```
>	shasta-Linux-0.8.0 --input <input.fastq> --config Nanopore-Oct2021
	--assemblyDirectory <output location> --threads <thread count of CPU> --
	assemblyDirectory shatsta_5kb --Reads.minReadLength 5000 --
	MinHash.minBucketSize 30 --MinHash.maxBucketSize 70
```

Quality control of the Shasta assembly

```
>	quast <input.fasta> -o <output location> --eukaryote --large -r <reference ge-
	nome.fasta>
```

Genome mapping

As mentioned, two overall pipelines can be used for structural variant calling, however both pipelines expect a mapped input.

Mapping of Shasta assembly

```
>	minimap2 -a -x asm5 --cs -r2k -t <thread count of CPU> <reference ge-
	nome.fasta>   <assembly.fasta> <output.sam>
```

Mapping of raw reads

I have not performed this set as ONT already assembled their realease of the GM24385 dataset. However, this might not be true in all cases and therefore I include this command in my pipeline.

```
>	minimap2 -a -x map-ont -t <thread count of CPU> <reference genome.fasta>
	<assembly.fasta> <output.sam>
```

Sorting and indexing the mapped files

```
>	samtools sort -m4G -@<thread count of CPU> -o <output.bam> <input.sam>
```

```
>	samtools index <input.bam>
```

Quality control of the genome mapping

```
>	Samtools flagstat <input.bam> < <output.txt>
```

Structural variant calling

Structural variant calling of Shasta assembly

```
>	swim-asm haploid <output location> <input.bam> <reference_genome.fasta>
```

Structural variant calling of read mapped data

```
>	svim alignment <output location> <input.bam> <reference_genome.fasta>
```

Compressing and indexing the SV files

```
>	bcftools view <input.vcf> -Oz -o <output.vcf.gz>
```

```
>	bcftools index <input.vcf.gz>
```

Benchmarking the SV calls

First run

```
>	truvari bench -b <truth_set.vcf.gz> -c <input.vcf.gz> -f <reference_genome.fasta> -o	<output location> --includebed <truth_set.bed>
```

Second run

This run uses manually set configurations in order to obtain a better benchmark (see discussion).

```
>	truvari bench -b <truth_set.vcf.gz> -c <input.vcf.gz> -f <reference_genome.fasta> -o	<output location> --includebed <truth_set.bed> --giabreport --passonly --multimatch

	 -r 5000 -p 0.00
```

Consistency report

```
>	truvari consistency <vcf_1> >vcf_2> … <vcf_n> > <output.txt>
```