UPPSALA
UNIVERSITET

UPTEC X 18 002
Examensarbete 30 hp
Mars 2018

Identifying esophageal atresi associated variants from whole genome sequencing data

Jonas Mattisson

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Identifying esophageal atresi associated variants from whole genome sequencing data

Jonas Mattisson

Knowing the underlying cause of a genetic disorder could not only further our understanding of the disease itself, and the otherwise healthy mechanism that is disrupted. It could potentially improve people's lives. Even if whole genome sequencing has drastically improved the potential of discovering the cause, a comparison of two non-related individual's genome will find several million sequence variations. While most variants have no significant impact, it is enough for only one to functionally impact a gene, for it to cause a genetic disorder. This project therefore focused on the filtering of variants, from lists of several million possible causes, to the stage where they could feasible be manually analysed one by one. Single-nucleotide variants, indels and structural variants were filtered, based on a dataset where single-nucleotide variants and indels had already been called. The more difficult process of structural variants discovery was performed, but it required the application of four different tools to minimise the drawback of each separate discovery technique. The same three filtering approaches were applied to all variants; the intersecting of datasets that should contain the same variant, the removal of variants in common with the general population and the selection of variants impacting functionality. Each approach proved to be an efficient filtering step, with their combination reducing each list to only a couple of variants out of the original five million. Due to lower accuracy and sensitivity of the structural variant analysis, this data will likely require more extensive manual analysis.

Handledare: Niklas Dahl & Joakim Klar
Ämnesgranskare: Adam Ameur
Examinator: Jan Andersson
ISSN: 1401-2138, UPTEC X 18 002

Populärvetenskaplig sammanfattning

Sekunder till timmar och dagar till år är konkreta termer vi använder för att definiera tid, ändå blir historia som tydligast när vi använder abstrakta termer som epoker och eror. Tidigt i det nya milleniet, 2003, firade vetenskapsvärlden slutförandet av det då över ett decennium långa arbetet att kartlägga en individs fullständiga genetiska information. Idag, nästan 15 år senare, kunde tio humana genom analyseras i sökandet efter en ovanlig sjukdoms genetiska orsak, i ett examensarbete. Endast genom att se på biologi utifrån ett före och efter Next-Generation Sequencing (NGS) perspektiv blir denna otroligt snabba utveckling möjlig.

En av de absolut största fördelarna med denna nya era är den mängd data som plötsligt blivit tillgänglig. När genetiska sjukdomar tidigare studerades krävdes stora familjer som bar på åkomman, för att kunna se ett mönster över hur den faktiskt ärvdes. Utifrån detta mönster kunde man identifiera mindre områden i arvsmassan, gemensamma för de individerna som bär på sjukdomen. Dessa områden undersökte man därefter närmare med andra metoder. Nu kan vi i stället utnyttja NGS för att först sammanställa personens fullständiga genetiska information, till och med hela familjens, för att sedan hitta orsaken. Detta tillvägagångsätt minskar signifikant analystiden, och nästan alla de krav som tidigare fanns försvinner. Dock kan en enda liten orsak, som att ett *a* blivit ett *t* på en enda plats i hela arvsmassan orsaka sjukdomen, vilket ska ses i ett sammanhang där det humana genomet består av cirka tre miljarder möjliga platser. Vi söker alltså efter nålen i höstacken.

I detta projekt undersöktes tre familjer där vissa medlemmar har fötts med missbildningen esofagusatresi (EA), vilket innebär att matstrupen inte har bildats kontinuerligt, och därför inte hänger ihop hela vägen. Då dessa familjer har en ovanligt hög förekomst av sjukdomen, är det troligt att den underliggande orsaken är genetisk, en orsak som är intressant att hitta. Projektet tog därför fram hela genomet på tio familjemedlemmar, där alla bör bära på denna orsakande faktor. Det är dessa tio filer projektet undersöker, med det slutgiltiga målet att filtrera ner flera miljoner möjliga orsaker till ett markant mindre antal för vidare analys med andra metoder. För att uppnå detta, kombineras logiska steg som exempelvis uteslutningsmetoder, med bioinformatiska verktyg som kan läsa av och manipulera den genetiska datan.

Vi kan tänka på vår arvsmassa som en tjock bok, där meningar, sidor och kapitel beskriver hur vi fungerar. De vanligaste förändringarna är att enstaka bokstäver har bytts ut, lagts till eller tagits bort. Så länge denna förändring inte är i en kritisk punkt, så har den ingen markant påverkan. Vi förstår alltså fortfarande vad meningen ska säga. Kan dock vår kropp inte längre korrekt tyda sammanhanget, kan detta orsaka en genetisk sjukdom. Även större förändringar kan självklart också ske, vilka kan ses som större mutationer där hela delar av boken har ändrats. Konstigt nog gör NGS det väldigt svårt att hitta dessa förändringar. En stor del av projektet arbetade just med att först hitta stora förändringar, för att sedan även där filtrera fram de mest troliga.

Projektet fokuserade alltså huvudsakligen på att filtrera ner de listorna på möjliga orsaker så långt som möjligt. Stegen som följde därefter, vilket inkluderade att faktiskt identifiera faktorerna som leder till missbildning, utfördes alltså inte. Resultatet blev dock en markant minskning av listor, där det som kan ha varit av intresse gått från flera miljoner möjligheter till sammanlagt några hundra som faktiskt kan vara av värde att undersöka vidare. Väldigt mycket data producerades i analyserna, men det är endast antalet hittade orsaker i de olika listorna som presenteras i den här rapporten.

Table of content

Abbreviations

CNV	*Copy-number variation*
DNA	*Deoxyribonucleic acid*
EA	*Esophageal atresi*
NGS	*Next-generation sequencing*
SNV	*Single-nucleotide variant*
SV	*Structural variant*
TEF	*Tracheoesophageal fistula*
UPPMAX	*Uppsala Multidisciplinary Center for Advanced Computational Science*
WES	*Whole exome sequencing*
WGS	*Whole genome sequencing*

1 Introduction

Sequencing technologies have been making great leaps of progress over the last couple of decades, especially with the development beyond Sanger sequencing into a Next-Generation Sequencing (NGS) era. The combination of lower cost per base pair with a considerably higher throughput, which NGS brought, meant that research groups could finally feasibly utilize whole genome sequencing (WGS). The constant improvement since then, of base pair throughput and reduced costs, have now lead to a stage where a human genome potentially could be run for less than $1,000. This is a trend of development that will hopefully continue, making further research on a larger scale possible in the future (van Dijk *et al.* 2014).

Genetic disorders were previously much more difficult to study. Larger families were a necessity in order to pinpoint the location of the cause to smaller loci, as studying longer genetic sequences was far too expensive and time consuming (Buermans & den Dunnen 2014). Especially complex diseases, which have multiple genetic causes, were difficult to examine. While single genes could be isolated, it was nearly impossible to achieve a clear picture without having more genetic data available (Hofker *et al.* 2014). Due to these previous difficulties, one of the most important NGS applications became the detection and diagnosis of genetic disorders, and as NGS is getting better and cheaper, even more data can be used to study these disorders (Buermans & den Dunnen 2014).

Variants are differences in the deoxyribonucleic acid (DNA) sequence compared to what might be considered a reference for said sequence (Teng 2016). All of us carry a large variety of different variants that has either been inherited from an ancestor's mutation or is a mutation that has occurred specifically in that person. While most of the variants each of us carry have no significant impact, those mutations with an effect that either leads to the loss, or even gain, of a function are the ones of interest. These variants, with a different effect than the reference, are the ones that are likely to be negative and have an effect which we would then consider a genetic disorder (Acuna-Hidalgo *et al.* 2016). It is in the light of finding these disease-causing variants where NGS introduces a new problem; too much data. Whole exome sequencing (WES), when only protein coding DNA is analysed, finds approximately 20,000 variants of which about 100-200 might be considered damaging in some way. WGS finds about 4,000,000 variants, which means that more, potentially impactful, variants are reported. These numbers do vary a lot, but each genome introduces new unknown, *de novo*, variants that might not be possible to resolve, even with the help of additional resources. A substantial list of possible variants will therefore be likely to occur in any study using WGS, and unless adequate steps of filter are taken, a lot of time might have to be wasted on going through them by hand (Lohmann & Klein 2014).

In the same way that there exists an abundance of different genetic disorders, there are also different reasons to why they should be studied. In the case performing genetic tests to screen for certain disorders, the reason often depends on whether it is a birth defect developed during pregnancy, or if it develops later in life. Prenatal screening is used to diagnose genetic disorders in fetuses, before the defect develops, which is already done today for certain serious disorders. This brings an opportunity for medical intervention for some specific disorders, possibly improving the outcome. Other times it might allow the parents to make informed decisions, and to make preparations depending on said decision. Genetic testing of the parents themselves before conception might also inform them of the risks for passing along certain conditions. (Temming & Macones 2016). On the other side of the spectrum, those disorders that develop later in life can use screening to help individuals prepare for certain possibilities. This might enable them to make choices that minimise risks, with the help of knowledge that they would otherwise lack. Especially in families where a certain hereditary disorder is known to be present, genetic testing can say whether the causative variants can be found in a certain individual, removing worry or forcing them to be more vigilant (Ingles & Semsarian 2014). While both prenatal and adult screening revolves around giving predictions, as well as the possibility to prepare, just being able to provide patients and their families with a clear explanation of the cause of their disorder, is a justification for discovering said cause.

Another aspect to consider is what could be learnt further from studying the cause behind a genetic disorder. Our understanding today of several physiological and molecular processes have been discovered by looking deeper into their related abnormities and disorders. This can be done by finding the causative variant and comparing it to the normal, non-disorder related one. Through the differences between these, as well as other related information such as corresponding pathways, mechanism underlying their development or function, could be discovered (Blüher & Mantzoros 2009). The final, and in some cases the most important, reason to study genetic disorders, would be so that either gene therapy, changing the actual variants, or some other method that negates the negative effects, could be performed. This has already been done on several monogenic diseases by applying viral vectors, but even more potential could be achieved with future development (Williams 2014). However, before any of the previously described reasons can be applied, we have to actually discover the causative variants, which will always be essential if we want to fully understand any genetic disorder.

1.1.1 Aim

The goal of this project is to find, or at least minimise the possible variants that might be the cause of EA, based on a certain set of data. While factors have previously been found for syndromes containing EA, this project looks at the WGS from families where EA only occurred as cases without any other malformations.

Those born today with this birth defect can live relative normal lives, suffering only from a small set of possible discomforts. It might still be interesting for these individuals to be given an explanation of why this oesophageal malformation occurred. This information could potentially also be used to provide prenatal screening for families where EA is hereditary, preparing for complications after birth. The discovered information might ultimately be used in future studies to examine the mechanisms and processes that mediates the formation of the oesophagus, as the gene affected by the variant is instrumental in that process, considering the malformation that occurs in EA patients.

2 Background

2.1 Variants

Even if variants can be seen as locations where a genome diverge from what is considered a human reference genome, there are different ways that mutation events can occur, effectively resulting in separate types of variants. As this project analyses a genetic disorder found across families, variant that cannot be inheritable are not of interest. Somatic variations, only found in specific cells, not in the germline, will therefore not be analysed. Cell specific mutations can occur every time said cell replicates, and each individual has them in abundance, but most studies analysing somatic variants do so in relation to cancer cells, of which they are the cause. Compared to hereditary variants, somatic variants not only require other steps in their analysis, but a completely different dataset. The variants that will be analysed can be further separated into two categories, single-nucleotide variants (SNV) and structural variants (SV). Analysing these two different types of variants will require slightly different steps, but the general idea of filtering them will be the same (Teng 2016).

2.1.1 Single-nucleotide variants

Running a site by site comparison between corresponding positions in an aligned genome compared to its reference allows us to find a plethora of small variations. The first type of variant found will be SNVs. These are variations of singular nucleic bases, one nucleotide base exchanged for another. The second type of variants is indels, slightly larger events of 1-50 base pair in size. This event is either an insertion or a deletion, which means that a segment of DNA has either been added or removed at a specific position. Even if SNVs and indels are different types of variants, the fact that they can be found using the same methods, while also being short enough to filter using the same steps, means that they from a practical point of view belongs to the same category (Teng 2016).

SNVs positioned in the coding regions of genes are commonly analysed as their impact will heavily depend on their synonymous or non-synonymous nature. A non-synonymous SNV will result in another amino acid, which might drastically affect the transcripted protein. However, to predict the impact of a SNV many more aspects have to be considered, not only within protein coding regions, as many up- and downstream regions affect the transcription of the related gene. Indels have the same possibilities of impacting function, with the added risk of causing frameshifts, moving the codon reading frame. Depending on the position of the frameshift, loss of function for the gene might be a very likely outcome (Cingolani *et al.* 2012).

2.1.2 Structural variants

Genetic mutations larger than 50 bases are categorised as SV, which can occur in multiple ways. Insertions and deletions can also occur in the form of SVs, with larger segments added or removed compared to indels. A DNA sequence can also be copied into another location, which is called duplication. Variants due to insertions, deletions and duplications are often categorised as copy-number variations (CNV) as they are unbalanced, altering the length of genomic segments. Mutation events can also occur where DNA sequences have simply been moved to another location, called translocation, or partly reversed, called transversion. These two events are considered balanced, as genomic lengths are kept (Guan & Sung 2016).

While SNVs can be found in the comparison between the alignment and the reference, SVs are much harder to detect due to the nature of NGS. By fragmenting DNA into short reads, and running these reads in parallel, NGS achieves a high sequencing throughput. Even if steps and methods vary between different NGS technologies, the fundamental idea remains the same. The sequenced fragments must then be reassembled, which is done in reference to another genome, unless additional steps are taken to produce a *de novo* assembly. SVs that are identical to other segments of the used reference, such as duplications, translocations and transversions will be placed in the reference location when aligned. In the alignment, deletions will just be empty segments, while inserts can possibly be unmapped. Additional sequencing runs, using longer read technologies, as well as several other steps required to produce a *de novo* assembly will therefore be necessary to achieve a more correct assembly regarding SVs (Guan & Sung 2016).

Detecting SVs in genomes mapped to a reference is however not impossible. Four different general techniques exist for SV detection. A multitude of tools are available for this purpose, each caller adopting different approaches to the techniques, as well as applying several techniques per tool. Clustering is the most direct technique where reads that are not mapped to the reference correctly, are grouped together based on their proximity to each other. Groups of these discordant reads are then used to find the SVs that cause their disagreement to the reference. Split-read alignment is the second technique that looks at continuous reads that could be partially mapped to different positions in the reference, and tries to either find or redefine the breakpoint. The third technique called contig assemblies tries to assemble paired reads into contigs, which with enough length detects the exact SVs. The final technique tries

to find CNVs by statistically analysing read-depth, which is a number for how many reads were mapped to each position in the reference. It is therefore called statistical testing. Each technique has its own advantages and drawbacks, which further means that the sensitivity and accuracy of each SV detection tool vary significantly. Considering that each tool has its own approach, combined with the fact that each approach will struggle with certain SVs, means that a combination of different callers can achieve higher sensitivity or accuracy compared to each individually. It is therefore advised to use several SV detection tools, which together apply all four techniques. They can either be used to strengthen each other's accuracy, removing false positives only found by a single caller, or to increase sensitivity by keeping all possible variants. The accuracy and sensitivity of SV detection will however still be lacking due to genetic regions that are difficult to analyse the nature of NGS, and other sequencing errors (Guan & Sung 2016).

2.2 Esophageal atresi

Esophageal atresi (EA) denotes a collection of oesophageal malformations, in all of which the oesophagus is in some way discontinuous. These malformations are birth defects that prevents the newly born from swallowing, a condition which is lethal unless surgically rectified. Even if the oesophagus is fixed, patients are likely to experience inconveniences for the rest of their lives, such as acid reflux and swallowing difficulties (Malmfors & Stenström 2015).

In only 8 % of all EA cases can a discontinuous oesophagus be solely used to explain the malformation. The rest of the cases also consist of a Tracheoesophageal Fistula (TEF), where the oesophagus has, as a defect, formed one or two connections to the airways. The most common type of EA is when a TEF has formed between the lower of the disconnected oesophagus ends and the airways. This type occurs in 84 % of all cases (Scott 2014). A simple representation of different EA types can be seen in *Figure 1*.

In Sweden, EA occurs approximately once in every 4,000 newly born, according to the National Board of Health and Welfare (Malmfors & Stenström 2015). The majority of cases (60 %) of EA occurs related to a syndromic diagnose. These are disorders where the patient suffers from several different kinds of harmful conditions, not only a disconnected oesophagus (Scott 2014). This project only analyses non-syndromic cases of EA.

2.2.1 What is known so far

While it is today unknown how many of the EA cases that occurs due to hereditary factors, compared to environmental causes, certain genetic anomalies have been found that occurs with the malformation, as well as some proposed environmental causes that slightly correlate. All hereditary causes found so far seem to have been discovered in patients suffering from an actual syndromic diagnose, not isolated cases. It is therefore generally believed that EA that is hereditary, but not syndrome related, is caused by multiple different factors (Scott 2014).

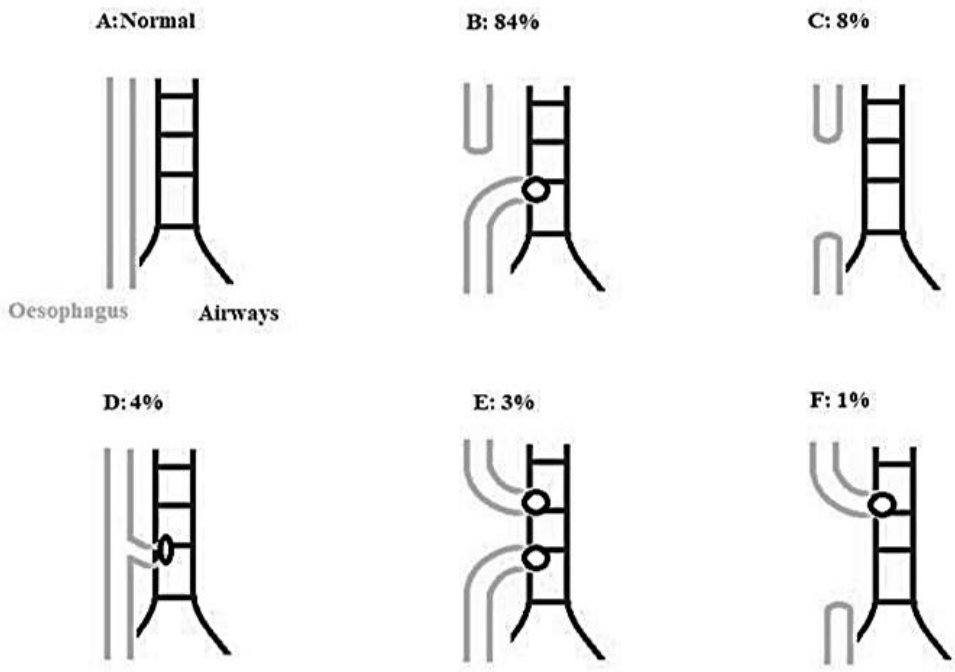

Figure 1: An overview representing the different types of EA. Grey is used for the oesophagus, while black is used for the airways. ***A*** *indicates a normal oesophagus.* ***B*** *indicates the most common version of EA, where a TEF has been formed from the second, lower end of the disconnected oesophagus to the airways.* ***C*** *indicates EA without TEF.* ***D****,* ***E*** *and* ***F*** *represent other types of EA with TEF. All individuals analysed in this report had version* ***B****.*

The gene *SSH* have previously been found to correlate to multiple malformations, among them EA. One study found the deletion of a locus which included the *SSH* gene in an EA patient (Busa *et al.* 2016). A transcription factor affecting *SSH* is *GLI3*, in which another study found a variant that was predicted to being harmful. This variant was found in a single patient with the Pallister-Hall syndrome, and further mice model tests of *GLI3* actually showed oesophageal malformations, among other effects (Yang *et al.* 2014). Other syndromes where EA can be found as one of the defects, and where a single-gene cause has been found are Anophthalmia-esophageal-genital syndrome, CHARGE syndrome and Feingold syndrome, where *SOX2*, *CHD7* and *MYCN* are the related genes found respectively. Other genes where causative variants also have been found are *FANCA*, *FANCB*, *FANCC*, *ERCC4* and *MID1* (Scott 2014). Other, more general factors, have also been associated to EA in the form of trisomy and multiple large deletions (Felix *et al.* 2007), as well as uncommon CNV. In the case of CNV, these were only found in a small percentage out of 375 studied patients (Brosens *et al.* 2016). None of the associated factors described, explains isolated cases of EA.

2.3 The families

Whole genome sequencing data, sampled from ten Swedish individuals, was used to perform the analysis in this project. The ten samples originated from three different families where the occurrence of EA is unusually high, to such degree that hereditary factors become likely. An overview of the families can be seen in *Figure 2*.

Consent was given by all partaking individuals, or their legal guardian, based on provided information regarding both the study itself, as well as their rights as participants. While the genetic samples are stored in a secure biobank, no other personal information was collected for this study. Ethical approval was obtained from the Regional Ethical committee of Uppsala 2010-08-11 (Dnr 2010/236) and according to the declaration of Helsinki.

All the cases of EA within the three families were of the most common kind, where a TEF has been formed between the airways and the second lower part of the discontinuous oesophagus. EA was also the only malformation found in any patient, which suggests that these are isolated cases of EA and not part of a more significant syndrome.

Niklas Dahl's group performed whole exome sequencing on four of the patients prior to this project. No significant variations could be found in this data, which led the group to later on sequence the entire genome of 10 individuals. The abbreviation WES in *Figure 2* marks the individuals that the whole exome analysis were performed on, compared to WGS which marks those studied in this project.

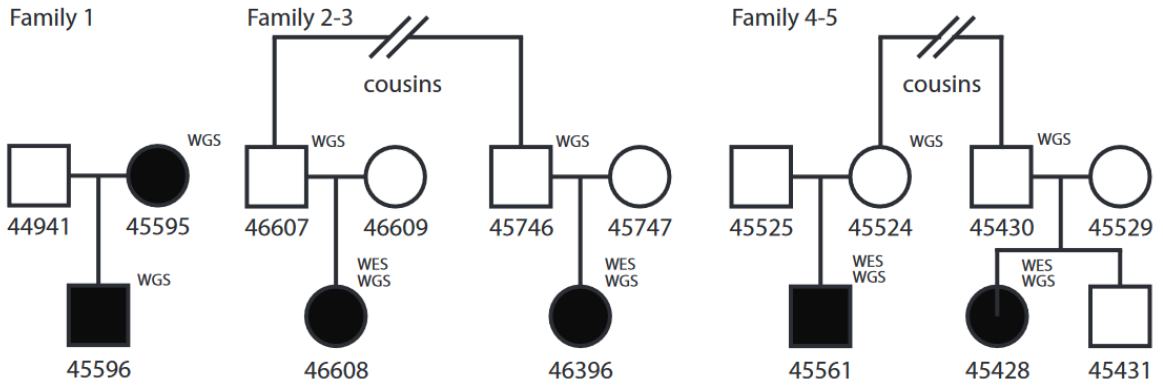

Figure 2: A schema representing the families analysed in this project. Squares are male individuals, while circles represent females. Each individual is marked by a number, which is used as an identifier for that individual. Black indicates individuals born with EA. Those individuals marked with WGS were whole genome sequenced, while those marked with WES were whole exome sequenced for a previous analysis.

3 Materials & Methods

3.1 The Dataset

The analysis performed in this project was done on a dataset consisting of the WGS of 10 individuals, of which further information can be seen in *Figure 2*. After WGS, this data also underwent several steps of processing and analysis before being used in this project. These steps were run with National Genomics Infrastructures' pipeline called Piper, which follows the best practices according to the Broad Institute (Dahlberg 2016). This meant that the project could start off with already aligned genomes and called SNVs. Short descriptions detailing the datasets pre-processing steps can be seen below in sections 3.1.1 and 3.1.2.

3.1.1 Sequencing

All the samples were sequenced by SNP&SEQ Technology Platform in Uppsala, using Illumina HiSeqX technology. The sequencing was done with paired-ends and to 30X coverage.

3.1.2 Mapping & variant calling

The entirety of the alignment, as well as, SNV and indel calling, was done with the Piper pipeline (Dahlberg 2016). The reference genome used in both steps was the human reference genome hg19, version g1k_v37 in UPPMAX. Another thing to note is that the SNV annotation tool snpEff was applied in the pipeline, which led to the corresponding tool snpSift being used in the analysis. Further information relating to the pipeline, mainly the steps taken, as well as tools and versions used, can be found in S*upplement 1*. *Supplement 1* is the content of the version report, which came attached to the delivered data.

3.1.3 File formats

Three different file formats were used in the analysis steps of this project. The data produced from the mapping to the reference came in the form of BAM-files. The BAM format is the binary version of the Sequence Alignment/Map format (SAM), which is used to store sequence reads based on the corresponding mapped position in the reference. This element of presenting positions relative to the reference genome, enables other procedures, such as variant calling, to be done on BAM and SAM files (Zhang 2016). The SNV and indel variants found from the variant calling are given in the variant call format (VCF), which was specifically developed for said purpose. The VCF format can handle a large variety of annotations and information per variant, something that the tools can then utilize to better analyse and visualise genetic differences (Danecek *et al.* 2011). The last format used was BED, which is simply a text file where each row is a single feature, and each tab-delimited column shows information for said feature (Zhang 2016). This is a very flexible format that was used to store SV data applied in filtering steps.

All the resulting data from this project were in the form of VCF-files, where each file contained a set of variants found by filtering according to certain sets of criteria.

3.2 UPPMAX

Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX) is the resource available at Uppsala University to provide the data storage and computational power required to run projects and problems of a certain size and complexity. It is hosted by the Department of information technology, and is an absolute necessity for this type of bioinformatics.

All the data storage, from the alignments to the resulting variant lists, was kept at the Milou cluster of UPPMAX. It was also at this cluster that every step of the analysis was run. Through the cluster it was possible to access all except one of the tools that was used in this project. Accessing UPPMAX itself, to analyse the data, could be done in either one of two ways; interactively for smaller tests by writing shell commands through a terminal, or by sending a shell script through a queuing system. The latter was done for larger and longer runs.

3.3 SweGen

2017 a dataset containing all of the variants found in 1,000 Swedish genomes, accompanied by their frequencies in said population, was published. This was done due to the fact that different populations had shown to contain a significant variety of variants, especially in studies where a large quantity of population specific *de novo* variants were found. One of the fundamental steps of finding mutations that causes genetic disorders are to eliminate those variants that can also be found in the healthy part of the populations. Therefore, if we lack necessary knowledge regarding the variant setup of the studied population, it will weaken the predictive power of the analysis (Ameur *et al.* 2017).

With the purpose of creating this Swedish population specific dataset, WGS were performed on 1,000 individuals, mainly from the Swedish Registry (Tvillingregistret), but also from the Northern Sweden Population Health Study. These individuals were handpicked primarily based on principle component analysis of SNP data in order to fully represent the Swedish population, north to south. 506 males and 494 females were sequenced, with the average age of 65.2 years. Due to the age, as well as other criterias, variants related to severe genetic disorders should be highly unlikely or possibly only occurring at a very small frequency (Ameur *et al.* 2017).

The pipeline created to find the SVs in the SweGen dataset, or to call and compare SVs in other datasets to SweGen, was also tested in this project (Viklund 2017). The SweGen dataset was applied to filtering steps in the form of a VCF-file for SNVs, and as a BED file for SVs. The same reference was used to create both the SweGen dataset and the alignments in this project.

3.4 Tools

A set of multiple tools were necessary in order to perform the different steps of the data analysis. Below follows a very brief description of each prominent tool, categorised on its functional usage in the project.

3.4.1 VCF-handling

BCFtools is a continuation of VCFtools which was originally developed for manipulating VCF-files (Danecek *et al.* 2011). The same functionality available in VCFtools is retained in BCFtools, with the added benefit of being able to handle the more efficient binary variant call format, BCF. BCFtools was primarily used to compare, manipulate and to visualise compressed VCF-files.

HTSlib is a library developed for accessing a selection of common bioinformatic file formats. It was therefore used in the project for the bgzip and tabix functionality, which enables the compression and indexing of file formats like VCF.

vt is a tool that can manipulate VCF-files in several ways, the main use being the normalization of variants. It is common for variant callers to inconsistently represent discovered variants, making comparison unnecessarily difficult. vt can rectify this by normalizing all variants uniformly, re-evaluating them based on certain rules (Tan *et al.* 2015). vt was used for normalization, as well as its function to deduplicate variant entries i.e. separating rows containing several different variants into one row per variant. Deduplication was a required step to ensure that BCFtools could correctly compare duplicated variants, which was done using vts decompose functionality.

3.4.2 SNV filtering specific tools

snpSift is a tool that manipulates VCF-files based on the annotation previously created by snpEff (Ruden *et al.* 2012). The annotation takes into account several factors, but mainly the impact that the differences between variant and reference will cause at each position (Cingolani *et al.* 2012). While all large set of functions are available to snpSift, it was used in this project to filter variants based on their snpEff predicted impact, the frequency in the SweGen dataset, and also to extract certain information from the VCF fields.

3.4.3 SV calling tools

Manta is a SV caller that pride itself on discovering SVs and indels, both accurately and quickly. It works in two phases, first by creating a graph over all related breakpoints in the genome, which are then analysed in order to find SVs. Manta therefore discovers and also scores SVs based on supporting paired and split-read evidence (Chen *et al.* 2016).

Delly requires paired-end reads to call SVs, which it uses to discover all discordantly mapped read-pairs that are abnormal. It then clusters paired-ends that show the same pattern to use as breakpoints, to screen for split-read support. It is this combination of breakpoints and split-reads that are further analysed to find SVs (Rausch *et al.* 2012).

CNVnator takes a new approach to already established read depth techniques, enabling it to run on much larger datasets. The focus for this tool lies on CNVs, as the name implies (Abyzov *et al.* 2011).

TIDDIT is a SV caller that can detect any SV, but special focus is put on larger variants of sizes above 1,000 base pairs. This is only beaten by SV callers that employ the contig assembly technique, which remains far more time consuming to run. TIDDIT starts its search by computing coverage and insert distribution, which are used to form genomic coordinates. Based on these genomic coordinates, discordant reads and split-reads are analysed to find SVs. Neighbouring areas are searched for further evidence (Eisfeldt *et al.* 2017).

3.4.4 SV filtering specific tools

BEDTools is a tool made with the intention of manipulating BED-files. Even if the tool itself has a lot of possible uses, only its intersect function was applied in this project. BEDTools' intersect keeps only certain genomic features depending on the pattern of features in other files, effectively filtering according to what options are currently chosen. The complementary intersect option, used in all runs for this project, removed any features in the first file that in any way overlapped with features in the second file. Another advantage of working with BEDTools is that it can also handle other file formats, such as VCF (Quinlan 2014).

VEP is an annotation tool that can annotate both SNVs and SVs. Developed by Ensembl, it combines the common route of looking at each variant's position and change, with further steps of going through a variety of available reference data. This additional data is partly specific for the reference genome used, and might have to be downloaded separately (McLaren *et al.* 2016). VEP was used to annotate all SVs, except the lists with unfiltered variants discovered by TIDDIT, on which it could not be successfully run.

3.4.5 Programming language

Python was used when no other easy solutions were available, and scripts had to be written to solve it. The choice of programming language was made due to personal preferences instead of other potential advantages, such as computational speed. Those kinds of aspects were not necessary to consider in this case as all computationally heavy steps are performed by published tools.

3.5 SNV Filtering

The filtering of SNVs used the VCF-files produced by the variant calling steps in the Piper pipeline. The goal of this analysis was to filter the lists down to a size range manageable by future manual steps, while still keeping as many potential causes as possible.

The first step was to prepare the VCF-files, as well as the SweGen data, to be correctly compared in later stages of the filtration. They were therefore deduplicated and then normalized using vt. For the deduplication, vt decompose used the smart decomposition option (-s) and output option (-). vt normalize was used for the normalization with the quiet (-q) option to avoid unnecessary messages, and the same reference (-r) was used as for the alignments. With one core on UPPMAX, using vt with those options, it took 1 minute and 13 seconds for a single VCF-file to be run, compared to 13 minutes and 33 seconds for the SweGen file.

The second step was to only keep those variants that could be found in every member of each family necessary to explain the inheritance pattern of the disorders. The choice of taking this as the second step was twofold. First it decreased the number of files to handle in every step from ten to three, making the rest of the project a bit easier and clearer. Secondly, it was the filtering step that would bring the most certainty. If there is a hereditary cause for the disorder in a family, it must be present in all these individuals. This comparison between family members was done using the intersect (isec) function of BCFtools, which simply finds identical variants in different files. For each family the number of files that the variant had to be present in (-n) was equal to the number of family members. Only one out file was required since variants of interest should be found in all samples, which meant that the index of the out file option (-w) could be set to 1 for that specific file. The output format option (-O) was set to compressed VCF (z). Lastly an output directory was marked (-p). The results from this step can be seen in *Table 1*. The following steps, after joining all the family members together, was done in a single shell script, as previous output was unaffected by varying variables. With six cores, this script took about 45 minutes to run. Fewer cores might throw a memory exceeded error.

The next filtering step removed those variants that were deemed too common in Swedish populations, more specifically the SweGen dataset. While this can be used to, more or less, remove all those variants that are not specific for the families, it's important to keep in mind that variants causing the disorder can exist in healthy individuals without leading to malformations. The script for SNV filtering was therefore modified to easily allow different values for a cut off frequency, up to which a variant can still be allowed to remain within the dataset. Based on the decided frequency, snpSift was used to filter the SweGen dataset, keeping variants with a frequency (AF) above that threshold. This step took about 25 minutes with six cores. All of the three families' lists were thereafter run against this new, filtered, SweGen dataset using the complementary BCFtools intersect option (-C). This complementary option removed any variants also found in the second file, which are all now above the decided frequency. This run also used the file index (-w), output file format (-O) and output directory (-p) options. Running each family took approximately ten minutes with six cores. The resulting filtering effects can be seen in *Table 2*.

The third filtering step tried to only keep variants that would cause a significant impact, which were decided by annotated predictions done by the snpEff tool in the Piper pipeline. While this step removed a lot of probably unnecessary variants, including most variants located in intergenic regions, the annotated impact here was still just a statistical prediction and not decisively true. To pick specifically those variants annotated with either high or moderate impact, which are those that might actually have an effect, snpSift targeted all impact annotations (ANN[*].impact) with the capitalized version of these words. Doing this took less than five minutes with six cores, per family. These can be seen in *Table 3*.

Under the assumption that one or more specific variants cause this birth defect, it is likely that they are shared amongst the families. While it is also a possibility that each family carries their own mutation that leads to EA, the scenario with shared variants is still worth exploring. BCFtools intersect was therefore used, similarly to the second step of the SNV filtering, to create four different family combinations, Family 1 & 2, Family 1 & 4, Family 2 & 4, as well as all families combined. The resulting variants found in each set of families can be seen in *Table 4*.

Finally, if the disorder is caused by the loss of function of a certain gene, it might not be necessary for all families to share exactly the same variant causing said loss of function. By first extracting both the gene names and their corresponding impact factor with snpSift from all families, as well as the family combinations, a python script was written that kept the impactful gene names. Another python script was thereafter written that intersected two of the gene name lists, as well as one script that removed variants from one script found in the other. These scripts could then be used to create impactful gene lists corresponding to different family combinations, which could later in turn be filtered toward the lists for the actual family combination. The resulting lists from these runs only contained genes that had not previously been suggested for said family combination, but where the genes function is significantly impacted in all members. All these scripts ran in only a couple of seconds when using six cores. Numbers indicated how many genes were found per family combination can be seen in *Table 5*.

3.6 SV Calling

All of the four SV calling tools described in section 3.4.3, were used to detect SVs. The combination of specifically these four tools was used in the disorder related variant calling pipeline developed by a group at Karolinska Institutet (Stranneheim *et al.* 2014), where they were chosen based on different benchmarks. The previously described list of SV-calling techniques had four different types: clustering, split-read alignment, contig assembly and statistic testing. This combination of tools uses different approaches to three types of techniques, lacking the contig assembly approach. However, due to the fact that methods applying contig assembly are extremely slow and that TIDDIT also has the ability to handle very large SVs, which otherwise usually is the advantage of those methods, this combination of tools should still work well. Each tool was run separately instead of applying the pipeline,

to avoid the difficulties of setting up the pipeline in the UPPMAX environment. All of these runs used the BAM-files created through the Piper mapping. A summary of the output from the callers can be seen in *Table 6*.

Running Manta first required the creation of a configuration file per sample. This was done using the configManta.py file, with an option for the input BAM-file (--bam), reference file (--referenceFasta), and the working directory path (--runDir). The workflow itself was thereafter run, using the appropriately named file runworkflow.py. This was run quietly to avoid unnecessary output (--quite), with a specified number of threads (-j). As it was on UPPMAX, it was marked as local for that running perimeter (-m). This resulted in three different output, one for diploid SVs, another for small indel candidates and finally one for SV candidates. Both candidate files contained variants with low scores, which can easily be disregarded if they remain after future steps. Manta was run with twelve cores, each sample requiring approximately 45 minutes to run.

The Delly caller only required a single line to run per sample, where the germline option (-g) was used, as we were not studying somatic variants. Other options included the output file (-o), a list of regions to exclude (-x), as well as the option not to look for smaller SVs (-n). While the overall sensitivity would have increased if the last two options had not been used, without them running each sample would have taken several days, due to the fact that threading of the run in UPPMAX did not seem to work properly. With the limiting options applied, running the samples took approximately three hours each.

To run CNVnator, several steps were required per sample, the first of which were to create a root file based on the samples input file (-tree). Secondly, a histogram (-his) had to be created, which required both the value for a bin size and the reference (-d), separated by chromosome. A very simple python script was created to split the reference genome, and 100 was chosen as bin size, as this was the authors recommendation for 30X coverage files (Abyzov *et al.* 2011). Thereafter statistics had to be calculated (-stat), then read-depth signal petitioned (-partition) which was GC-content corrected (-ngc). Finally, the root file (-root), which had been used in all steps so far, were called (-call) to detect variants. This step was also GC-content corrected and resulted in variants being printed one by one. After all these steps, the file cnvnator2VCF.pl had to be used to get the variants in the VCF format. Running all these steps took approximately an hour per sample, with four cores.

As TIDDIT was not yet available at UPPMAX it had to be installed before being used. The git clone command, combined with the running of the file INSTALL.sh, was enough to set it up. It was thereafter run using python for SV calling (--sv), for which it required the input file (--bam), output directory (-o) and the reference (--ref). With eight cores, each sample took a bit more than an hour to run.

The so-called WGSstructvar pipeline, developed to assist the finding of SVs and then their comparison to the SVs found in the SweGen dataset, was also tested in this project. It was

performed by exactly, step by step, following their manual, applying the SweGen SVs file to the mask cohorts originally called by Manta (Viklund 2017).

3.7 SV Filtering

The general approach behind the filtering of SVs is very similar to the one employed for SNVs, which means that this part of the method often will refer to previous explanations in the SNV filtering section (3.5). All steps below were written, and run, in a single shell script. The running time for this script was between eight minutes with two cores, and six hours with eight cores. This significant variety in execution time depended mostly on the frequency variable chosen for the SweGen dataset, as strict filtering left few variants for VEP to annotate, while no filtering meant that VEP had to annotate all discovered variants.

Similarly, to the filtering of SNVs, an initial step of running vt on all VCF-files, both with decompose and normalize, was necessary. The normalization step could however not be performed, and therefore had to be skipped on those variants called by CNVnator and TIDDIT, as unexplained errors were thrown.

In the first step of filtering, the family members were intersected using BCFtools. While this was executed in the same manner as for SNVs, the variants found by each tool remained separated, effectively resulting in six lists per family. The combining of family members, as well as families, were also performed on the SV files produced by the SweGen related pipeline.

To frequency filter the BED-file used to store the SweGen SV data, a python script was written. BEDTools complementary intersect (-v) function was used to filter the variants in every family- & tool-specific VCF-file. An option was used to make sure headers were kept (-header), and the first input file (-a) was the VCF-file, while the second input file (-b) was the BED-file.

VEP was thereafter used to annotate the still remaining variants in each file. Compared to the other tools where the newest version available on UPPMAX was used, VEP version 87 was the latest release that I could get to work. For this run a cache (--cache) was used for reference, which had to be downloaded specifically and uncompressed in a folder (--dir_cache). The merged file was used for this (--merged), where other options were quiet run (-q), vcf for the format (--format) and output (--vcf), a specified buffer size of 1,000 not to exceed memory (--buffer_size), both sift (--sift) and polyphen (--polyphen) usage set to b, and offline run (--offline). Annotation option also included canonical, biotype, ccds, regulatory, total_length, numbers and domains, which were set like the other options. For the run without frequency filtering, VEP could for some unknown reason not handle TIDDIT, making the process kill itself. It was therefore not used for that specific run, resulting in no impact filtering for TIDDIT without frequency filtering.

To at least try to remove some of the false positives reported by each separate tool, BCFtools intersect function was used to keep only those variants reported by a minimum of two different tools. As Manta had produced three files that partially contained the same information, these were first turned into one file using the BCFtools merge function. The options used for this was to first allow overlapping regions (--force-samples), then setting the options to how duplications should be handled (-m) to none. All other settings were the same as for intersect. Next the intersect was applied to the three tools, as well as the merged file. The number of intersection were set to two or more (-n +2), and no index was specified, keeping all files. BCFtools merge function was thereafter applied again to join the output files from the intersect, using the same options, resulting in a file with slightly more accurate variants.

Different family combinations were also created with BCFtools intersect, the same way that it was done in the SNV filtering. This step was however done on all the separate tool files, as well as the tool intersection file.

The last step of filtering was the impact filtering based on the VEP annotation. This was the final action for the SVs due to the large quantity of files already produced, which was further increased by additional files containing the impact filtering. Instead of using VEPs own filtering tool, which could not handle the TIDDIT files, the shell command grep was used. Extended grep functionality (-E) was applied, which allowed easy search for both the header and impact annotation simultaneously, keeping the VCF format intact. In *Table 7* the result from a collection of SV filtering steps can be seen. A quick side by side comparison the quantity of Manta SVs, with or without the pipeline, is showed in *Table 8*.

4 Results

In this section of the report, numbers indicating the quantity of features remaining in the filtered lists are shown. Each number can therefore be thought of as the representation of one list, which remains accessible through UPPMAX. No actual variants or genes, only their quantity, are presented in this section, due to the fact that this report does not focus on the manual analysis of variants.

4.1 Single-nucleotide variants

The results from different stages of the filtering process are displayed in Tables *1-5*. The complete table, showing all numbers simultaneously, can be found in *Supplement 2*. The first result, which can be seen in *Table 1*, shows the effect of working with complete families instead of separate individuals.

Table 1: This table represent the number of variants found in the different samples, as well as the number remaining when the family members were joined. The second row indicates sample identifiers in bold. The family above shows which family said sample belongs to. The numbers below show how many variants were found for that sample, where the last row indicates how many variants each family shared, out of those variants found per sample.

Family 1		**Family 2**				**Family 4**			
45595	**45596**	**45746**	**46607**	**46608**	**46396**	**45428**	**45430**	**45524**	**45561**
4974238	4900293	4913483	4776653	4937813	4973004	4974715	4892805	5016778	4924784
3886181		2515525				2492099			

The second filtration step, utilizing the SweGen dataset to remove common variants, is demonstrated in *Table 2*. Here the difference between varying the frequency threshold, has been tested for 1 %, 5 % and 10 %. Strict filtering is also included where no occurrence in the SweGen dataset was allowed.

Table 2: This table represents the filtering effect from using the SweGen dataset to frequency filter each family, to an allowed percentage in the general population. Each row represents a frequency level from before any filtering was applied, to strict filtering where any variant also found in the SweGen dataset was removed. The last column shows the percentage of remaining variants for that frequency filtering, compared to the Before Filtering row.

	Family 1	**Family 2**	**Family 4**	**% Remaining**
Before Filtering	3886181	2515525	2492099	-
10% Filtering	266462	38842	26646	3,16
5% Filtering	143980	21886	16208	1,74
1% Filtering	57650	12486	10429	0,80
Strict Filtering	26919	10163	9889	0,50

The effect of the functional impact filtering step can be seen in *Table 3*. The same frequency steps used for *Table 2* are also demonstrated here, as it continues to provide insight into the effect of the frequency filtering as well as impact filtering under these conditions

Table 3: This table represent the filtering effect when filtering is based on functional impact. Each family has a row, as well as another separate row for after having been filtered. Different population filtering frequencies are shown in each column. Strict indicate that no variants found in the SweGen dataset was allowed to remain after filtering. The last column indicates the effect of the filtering per family, in the form of the average percentage remaining between the two rows.

	Strict Filtering	1% Filtering	5% Filtering	10% Filtering	% Remaining
Family 1	26919	57650	143980	266462	-
Family 1, Impactful	103	269	649	1113	0,43
Family 2	10163	12486	21886	38842	-
Family 2, Impactful	6	30	75	128	0,24
Family 4	9889	10429	16208	26646	-
Family 4, Impactful	4	5	36	73	0,15

The number of variants that remain when combinations of families were formed can be seen in *Table 4*. Both the effects of the frequency filtering and the functional impact filtering were displayed to give a more complete picture. Another thing to note is that the variants in the list for a combination of families, Family 1 & 2 for example, were unique for that family combination. If this was not the case, the variants found in all families, would also be found in every other family combination. These lists were created with the purpose of avoiding unnecessary cluttering of data, making the cells content clearer with the unique aspect. This is the only dataset in this project with which the unique approach was applied.

Table 4: This table shows the results from combining the families into different combinations. It also accounts for both impact filtering, as well as frequency filtering, like in table 2 & 3. Important to note is that the number of variants for each family combination is unique. This means that, those variants found in all families have been removed from the other combinations to make the remaining result in those groups clearer.

Family Combination Unique Variants	Strict Filtering	1% Filtering	5% Filtering	10% Filtering
Family 1 & 2	1278	1289	1443	3037
Family 1 & 4	1012	1021	1217	1909
Family 2 & 4	409	409	413	597
All Families	7459	7459	7459	7463
Family 1 & 2, Impactful	2	2	2	8
Family 1 & 4, Impactful	0	0	1	2
Family 2 & 4, Impactful	0	0	0	0
All Families, Impactful	0	0	0	0

The number of genes with impactful variants, where the variants themselves are not shared in different family combinations, but where a certain set of families still have an impactful variant for said gene, are shown in *Table 5*. These lists have not been made unique, like the ones in *Table 4*, but it was not deemed necessary considering the low quantity of features.

Table 5: This table shows how many genes that can be found in the analysis of impacted genes found in family combinations, without having shared variants for said genes in those families. These are not unique like in table 4, but still shows the frequency filtering.

	Strict Filtering	1% Filtering	5% Filtering	10% Filtering
Family 1 & 2, Impactful Genes	0	0	3	5
Family 1 & 4, Impactful Genes	1	1	1	5
Family 2 & 4, Impactful Genes	0	0	1	1
All Families, Impactful Genes	0	0	1	1

4.2 Structural variants

Due to the number of files produced during the SV calling and filtering stages of this analysis, the SV result section is more condensed than that of SNVs. Complete tables, displaying all the various feature quantities, can be found in *Supplement 3-7*. The first result displayed in *Table 6* lists the number of variants found per tool for each sample.

Table 6: This table represents how many variants were found per tool for each sample. The second row indicates sample identifiers in bold. The family above it represents which family said sample belongs to. Each row shows a single output file.

	Family 1		Family 2				Family 4			
	45595	**45596**	**45746**	**46396**	**46607**	**46608**	**45428**	**45430**	**45524**	**45561**
Manta: Small Indels	108070	103960	107927	109530	92563	104404	108730	105273	113179	107766
Manta: Candidate SV	126642	121621	126302	128474	106082	122182	126985	122288	133854	126454
Manta: Diploid SV	11063	10456	10950	11225	8026	10469	10775	10278	12291	11144
Delly	7357	6814	6876	6472	5212	6830	6823	6817	7005	6450
CNVnator	5143	4677	2098	4647	3754	5044	4575	4536	5130	4609
Tiddit	76732	67959	68483	71897	46739	68384	66664	63307	77920	66041

The filtering effect from allowing different frequencies in the SweGen dataset, predicting the functional impact by creating different family combinations, can be seen in *Table 7*. This was shown on the CNVnator dataset, which was chosen as it resulted in the clearest progression of filtering. To get the clearer picture, as well as to avoid potential tool bias, the other tools can be seen in the supplements.

Table 7: This table combines the result from all types of filtering the CNVnator dataset. It is therefore separated according to frequency filtering, from strict where no overlap with SweGen data is allowed, to the non-filtered column. Rows indicate what family combinations are used, as well as if they have been impact filtered. Impact is here divided into two levels, as each variant could be either moderate (potentially impactful) or high (likely impactful).

CNVnator	Strict Filtering	1% Filtering	5% Filtering	10% Filtering	Non-filtered
Family 1	94	175	262	339	1170
Family 2	49	79	103	114	280
Family 4	53	83	114	131	398
Family 1 & 2	47	74	94	103	224
Family 1 & 4	49	76	95	109	259
Family 2 & 4	43	68	87	97	196
All Families	43	67	85	94	182
Family 1 Moderate Impact	5	10	18	19	37
Family 2 Moderate Impact	2	6	9	9	14
Family 4 Moderate Impact	5	9	12	12	20
Family 1 & 2 Moderate Impact	2	5	7	7	12
Family 1 & 4 Moderate Impact	3	6	7	7	14
Family 2 & 4 Moderate Impact	2	6	9	9	13
All Families Moderate Impact	2	5	7	7	12
Family 1 High Impact	11	23	35	44	148
Family 2 High Impact	6	12	18	20	53
Family 4 High Impact	8	13	19	20	65
Family 1 & 2 High Impact	5	9	13	14	40
Family 1 & 4 High Impact	7	11	13	14	44
Family 2 & 4 High Impact	6	11	15	16	36
All Families High Impact	5	9	12	13	32

The same type of dataset, but where the tools have been intersected to increase accuracy, can be seen in *Table 8*. *Table 7* and *Table 8* tries to show as much relevant filtering data as possible, while keeping information only of interest for someone planning to work further with this dataset, as supplements.

Table 8: The same type of table as Table 7, which is used to show the effect of all the filtering. This is done on the merged dataset, where an intersection between the tools was made.

Merged Tools	Strict Filtering	1% Filtering	5% Filtering	10% Filtering	Non-filtered
Family 1	0	2	6	7	370
Family 2	0	1	2	2	220
Family 4	0	1	3	3	208
Family 1 & 2	0	1	2	2	154
Family 1 & 4	0	1	2	2	152
Family 2 & 4	0	1	2	2	139
All Families	0	1	2	2	118
Family 1 Moderate Impact	0	0	0	0	9
Family 2 Moderate Impact	0	0	0	0	6
Family 4 Moderate Impact	0	0	0	0	6
Family 1 & 2 Moderate Impact	0	0	0	0	1
Family 1 & 4 Moderate Impact	0	0	0	0	1
Family 2 & 4 Moderate Impact	0	0	0	0	2
All Families Moderate Impact	0	0	0	0	0
Family 1 High Impact	0	0	0	0	16
Family 2 High Impact	0	0	0	0	12
Family 4 High Impact	0	0	0	0	11
Family 1 & 2 High Impact	0	0	0	0	6
Family 1 & 4 High Impact	0	0	0	0	7
Family 2 & 4 High Impact	0	0	0	0	8
All Families High Impact	0	0	0	0	5

The difference between using Manta as a stand-alone tool, compared to the output produced by the WGSstructvar pipeline, can be seen in *table 9*. The stand-alone approach filtered discovered variants through the steps described in this report, while the WGSstructvar pipeline both called SVs and filtered them towards the SweGen SV dataset directly.

Table 9: This table compares the difference between the pipelined Manta calling in WGSstructvar, and the one from just running the tool. The Manta diploid SV dataset was chosen as it was supposed to have the highest accuracy. The table shows all the different family combinations, and also the number of impactful variants found in each dataset.

	WGSstructvar Pipeline	Manta: DiploidSV
Family 1	17	240
Family 2	11	80
Family 4	8	78
Family 1 & 2	9	46
Family 1 & 4	7	48
Family 2 & 4	5	36
All Families	5	27
Impactful	0	0

5 Discussion

The first thing to note in *Table 1* is that each sample has consistently about five million variants, which were discovered through the Piper pipeline. For those numbers that should be consistent to actually be so, is a good first sign. The fact that Family 1 contains a significantly higher quantity of variants is due to the reality that fewer and closer family members were being used than for the other two families (*Figure 2*). In the case when having only one parent and one child, many of the rare germline mutations, specific for the mother in this case, will still remain in the son. This can be seen further in *Table 2*, where the difference factor between Family 1 and the other two families, increases the more of the SweGen dataset is used, which is most likely due to the fact that Family 1 has a larger fraction of *de novo* mutations remaining. The impact filtering in *Table 3*, also supports this, partially due to the fact that more variants were kept from the frequency filtering step, but those kept were also more likely to be impactful. This is indicated by the much higher fractions of variants in the strict frequency with functional impact cells of the table in Family 1. This shows that those absent in the SweGen dataset are also especially high of impact, a pattern that would match that of entirely new mutations. All of this could mean that larger families, with more distant family members, drastically increase the efficiency of the filtering, not only for the initial step where shared variants are found.

The comparison between the results in *Table 2* to the results in *Table 3* shows that filtering based on impact removed a higher fraction of variants than the frequency filtering. Even if the result in these cases looks more efficient, impact filtering makes more assumptions, and it might just be more efficient when frequency filtering has already been applied. The initial step of combining family members (*Table* 1) is arguably the weakest of the filtering steps as it keeps about half, for both Family 2 and Family 4, of the initial five million variants per sample. As each consecutive filtering step is more effective for Family 2 and Family 4 than for Family 1, I would argue that having these large families to begin with is very effective. This is also the only step, which we with absolute certainty can say, does not remove any crucial variants, as long as the disorder actually follows an inheritance pattern within the family. Creating different sets of family combinations also proved effective (*Table 4*) leaving only a few impactful variants. This test does however only work under the assumption that the families might share the defects cause, which might not be the case. The same can be said for *Table 5*, showing the impactful genes without shared variants, which was a quick test looking into patterns that might be more difficult to see under manual circumstances.

Even if we cannot conclude anything with full certainty without further analysing the variants actually found for either SNVs or SVs, the SNV list should be considered probably more consistent and analysed first. While many pipelines and tools have been fully developed to analyse SNVs, SVs remains lacking in both the calling and the filtering aspects of disorder discovery. This is simply due to the fact that SVs are so much more difficult to find with today's technology, and therefore also bring a lot of uncertainty to the following steps, where tools often developed for straightforward SNVs are applied. People working further with this dataset, should therefore keep in mind that the callers will have missed some of the SVs, while also having added false positives. Looking at the caller output in *Table* 6, Manta still finds the most variants, but both the candidate files partially overlap and contain low score variants. In this project, all the filtering steps still removed enough variants making an additional step of filtering low-quality SVs unnecessary. Someone will have to keep this in mind when looking at each remaining variant. As most of the tools for VCF-files were developed for SNVs, it is likely that they might have been unable to handle some variants in the SV-files. Each caller practically had their own way of indicating SVs, to such degree that normalization with vt did not work for all tools, introducing uncertainties to the filtering. VEP also struggled with TIDDIT for some unknown reason, which hopefully will be fixed, seeing as TIDDIT probably found the most high-quality variants (*Table 6)*. In the future all these problems might be fixed with new or updated tools, both for calling and filtering, if better sequencing with longer reads is not yet developed. Until that time, we will have to keep the drawbacks in mind when going through SV-files, as luck might actually be on our side. However, as I never trust in luck and therefore did not want to throw anything of interest away due to the workings of a single tool, almost all combinations were created and stored, which resulted in *Supplement 3-7*.

Looking at the content of *Table 7*, both the impact and frequency filtering steps seem to perform decently and corresponding more or less to what could be seen for SNVs. The combining of families into different sets seems to be the weakest filtering step by far, which might be caused by population shared CNVs not being filtered properly in the frequency step. The SweGen dataset was only run with Manta, leaving variants found by CNVnators approach undiscovered. However, the main weakness of the entire analysis is how BEDTools intersect, which is found by just examining if features overlap position wise, not necessarily if they are actually the same variants. This means that it might also be worth to take a closer look at the non-frequency filtered column, in those cases where these contain a feasible quantity of variants. Compared to SNVs, where impact filtering was done for highly and moderately impactful variants being combined, high and moderate was separated as a test for SVs. Considering that both the moderate and high files should be checked, one might argue that the separation is either unnecessary or in fact clearer, depending on personal preferences. If we end by comparing *Table 7* to *Table 8*, we see the effect of trying to increase accuracy at the expense of sensitivity. One should keep in mind that the intersected files contain variants from all tools, not just CNVnator, which means that many variants, especially when frequency filtering is applied, are removed. Even if close to nothing remains after the frequency filtering in *Table 8*, taking into account how BEDTools operates, the non-filtered column is worth analysing further. If a more extensive picture is sought, this can be found in *Supplement 3-7*.

While it was just a quick comparison, *Table 9* shows that more SVs remained in the diploid SV file, than in the pipeline output. No impactful variants were found by either tool, but with the independent Manta call other files were also given. It was however much easier to use the pipeline. Following the description step by step, it simply did not provide enough variants for this analysis.

What is worth to remember is that the field of bioinformatics is currently developing at an incredible speed, with new sequencing methods and tools available yearly. I consider this both a blessing and a curse. In the next couple of years, new technologies might trivialize this entire problem. This could be achieved by longer sequencing reads being able to find all SVs and finished pipelines that successfully handles this process from start to finish. It is at the same time these advances with new formats that the tools cannot handle, and a sea of different tools to wade through to find the right one and the specific version you need for your file, which was the most time-consuming part of this project. With this development also comes the issue of security, as the whole genome is basically the most private information possible. As we do not know what the future will entail, a good first step is to focus on a more secure storage for this type of information. This is where we are today. While it might feel restricting, it gives us the opportunity to later adapt regulations, which would not be possible if all information was already publicly available.

In this project, precautions were taken to ensure the security and integrity of the dataset. First and foremost, the stored genomic information was only identified using assigned numbers, ensuring that no individual could be directly linked to one dataset. Secondly, all the actual data was always kept at UPPMAX, minimising the risk of released personal information. The entire project was also recently relocated to an even more secure cluster at UPPMAX, following new stricter guidelines.

Most of the remaining work that need to be done to identify a candidate variant that could explain the malformation in EA, is the manual analysis of potential variants. There are still many other steps that can be performed, like using the genes previously discovered to relate to EA and running a comparison between those and lists of variants, or other types of pathway analysis. I would also have liked to further analyse each filtration step, comparing how effective they could be under different circumstances, but there was no time left to do either the filtration comparison or to further analyse the dataset. Another step I would have liked to perform if time had allowed me to, would have been to redo the entire project after remapping the dataset to the newer hg39 reference genome, which should have decreased the number of contigs.

6 Acknowledgements

First of all, I would like to thank my supervisors Niklas Dahl and Joakim Klar for entrusting me with this incredibly fun and challenging project. Thank you Joakim for all the great support and assistance along the way, especially for letting me run all my different ideas by you. Thank you Niklas for all the constant feedback and input that you have provided, making sure I did not neglect the biology behind the project, while also giving me free reins to run with what I thought was best. I would also like to express my deepest gratitude to Adam Ameur for taking on the duties of reviewing my work, ensuring its quality. I am very thankful that you took time out of your schedule to meet with me, and that you always replied with great answers to all of my various questions.

Thank you Jan Andersson and Lena Henriksson for examining and evaluating this project, and actually making it possible through you planning. I am however even more grateful for all the support and feedback you have given me during these last couple of years. This program would not be what it is, if it was not for all your incredible work.

Lastly, I would like to express my immense gratitude and love for my family and friends that have supported me through all my endeavours. From all the moments you have tried to understand what I am actually doing, to all the times that you have made me forget, even for a single moment, about reports I had to write, thank you. Even if I am almost out of thanks, a special one goes to everyone that took the time to read through this report and give me feedback. Both it and I would not be the same without you.

7 References

Abyzov A, Urban AE, Snyder M, Gerstein M. 2011. CNVnator: An approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. Genome Research 21: 974–984.

Acuna-Hidalgo R, Veltman JA, Hoischen A. 2016. New insights into the generation and role of de novo mutations in health and disease. Genome Biology, doi 10.1186/s13059-016-1110-1.

Ameur A, Dahlberg J, Olason P, Vezzi F, Karlsson R, Martin M, Viklund J, Kähäri AK, Lundin P, Che H, Thutkawkorapin J, Eisfeldt J, Lampa S, Dahlberg M, Hagberg J, Jareborg N, Liljedahl U, Jonasson I, Johansson Å, Feuk L, Lundeberg J, Syvänen A-C, Lundin S, Nilsson D, Nystedt B, Magnusson PK, Gyllensten U. 2017. SweGen: a whole-genome data resource of genetic variability in a cross-section of the Swedish population. European Journal of Human Genetics, doi 10.1038/ejhg.2017.130.

Blüher S, Mantzoros CS. 2009. Leptin in humans: lessons from translational research1234. The American Journal of Clinical Nutrition 89: 991S–997S.

Brosens E, Marsch F, de Jong EM, Zaveri HP, Hilger AC, Choinitzki VG, Hölscher A, Hoffmann P, Herms S, Boemers TM, Ure BM, Lacher M, Ludwig M, Eussen BH, van der Helm RM, Douben H, Van Opstal D, Wijnen RMH, Beverloo HB, van Bever Y, Brooks AS, IJsselstijn H, Scott DA, Schumacher J, Tibboel D, Reutter H, de Klein A. 2016. Copy number variations in 375 patients with oesophageal atresia and/or tracheoesophageal fistula. European Journal of Human Genetics 24: 1715–1723.

Buermans HPJ, den Dunnen JT. 2014. Next generation sequencing technology: Advances and applications. Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease 1842: 1932–1941.

Busa T, Panait N, Chaumoitre K, Philip N, Missirian C. 2016. Esophageal atresia with tracheoesophageal fistula in a patient with 7q35–36.3 deletion including SHH gene. European Journal of Medical Genetics 59: 546–548.

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

Cingolani P, Platts A, Wang LL, Coon M, Nguyen T, Wang L, Land SJ, Lu X, Ruden DM. 2012. A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff. Fly 6: 80–92.

Dahlberg J. 2016. piper: A genomics pipeline build on top of the GATK Queue framework. National Genomics Infrastructure. https://github.com/NationalGenomicsInfrastructure/piper

Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, Handsaker RE, Lunter G, Marth GT, Sherry ST, McVean G, Durbin R, 1000 Genomes Project Analysis Group.

2011. The variant call format and VCFtools. Bioinformatics (Oxford, England) 27: 2156–2158.

Eisfeldt J, Vezzi F, Olason P, Nilsson D, Lindstrand A. 2017. TIDDIT, an efficient and comprehensive structural variant caller for massive parallel sequencing data. F1000Research, doi 10.12688/f1000research.11168.2.

Felix JF, Tibboel D, de Klein A. 2007. Chromosomal anomalies in the aetiology of oesophageal atresia and tracheo-oesophageal fistula. European Journal of Medical Genetics 50: 163–175.

Guan P, Sung W-K. 2016. Structural variation detection using next-generation sequencing data: A comparative technical review. Methods 102: 36–49.

Hofker MH, Fu J, Wijmenga C. 2014. The genome revolution and its role in understanding complex diseases. Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease 1842: 1889–1895.

Ingles J, Semsarian C. 2014. The value of cardiac genetic testing. Trends in Cardiovascular Medicine 24: 217–224.

Lohmann K, Klein C. 2014. Next Generation Sequencing and the Future of Genetic Diagnosis. Neurotherapeutics 11: 699.

Malmfors G, Stenström P. 2015. Esofagusatresi. WWW-dokument 2015-: http://www.socialstyrelsen.se/ovanligadiagnoser/esofagusatresi. Collected 2017-09-18.

McLaren W, Gil L, Hunt SE, Riat HS, Ritchie GRS, Thormann A, Flicek P, Cunningham F. 2016. The Ensembl Variant Effect Predictor. Genome Biology 17: 122.

Quinlan AR. 2014. BEDTools: the Swiss-army tool for genome feature analysis. Current protocols in bioinformatics / editoral board, Andreas D Baxevanis . [et al] 47: 11.12.1-11.12.34.

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

Ruden DM, Cingolani P, Patel VM, Coon M, Nguyen T, Land SJ, Lu X. 2012. Using Drosophila melanogaster as a Model for Genotoxic Chemical Mutational Studies with a New Program, SnpSift. Frontiers in Genetics, doi 10.3389/fgene.2012.00035.

Scott DA. 2009 [Updated 2014]. Esophageal Atresia/Tracheoesophageal Fistula Overview. GeneReviews(®). Available from: https://www.ncbi.nlm.nih.gov/books/NBK5192/

Stranneheim H, Engvall M, Naess K, Lesko N, Larsson P, Dahlberg M, Andeer R, Wredenberg A, Freyer C, Barbaro M, Bruhn H, Emahazion T, Magnusson M, Wibom R, Zetterström RH, Wirta V, von Döbeln U, Wedell A. 2014. Rapid pulsed whole genome

sequencing for comprehensive acute diagnostics of inborn errors of metabolism. BMC Genomics 15: 1090.

Tan A, Abecasis GR, Kang HM. 2015. Unified representation of genetic variants. Bioinformatics 31: 2202–2204.

Temming LA, Macones GA. 2016. What is prenatal screening and why to do it? Seminars in Perinatology 40: 3–11.

Teng S. 2016. NGS for Sequence Variants. Translational Biomedical Informatics, s. 1–20. Springer, Singapore,

van Dijk EL, Auger H, Jaszczyszyn Y, Thermes C. 2014. Ten years of next-generation sequencing technology. Trends in Genetics 30: 418–426.

Viklund J. 2017. wgs-structvar: Whole Genome Sequenceing Structural Variation Pipelines. NBIS -- National Bioinformatics Infrastructure Sweden. https://github.com/NBISweden/wgs-structvar

Williams DA. 2014. Curing Genetic Disease with Gene Therapy. Transactions of the American Clinical and Climatological Association 125: 122–129.

Yang L, Shen C, Mei M, Zhan G, Zhao Y, Wang H, Huang G, Qiu Z, Lu W, Zhou W. 2014. De novo GLI3 mutation in esophageal atresia: Reproducing the phenotypic spectrum of Gli3 defects in murine models. Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease 1842: 1755–1761.

Zhang H. 2016. Overview of Sequence Data Formats. SpringerLink 3–17.

8 Supplements

8.1 Version Report

The version report, included with the delivered files can be seen below. The main use of this report would be if someone wants to replicate the steps without the pipeline, or to see what version was used to create this dataset.

******
README
******

Data has been aligned to the reference using bwa. The raw alignments have then been deduplicated, recalibrated and indel realigned using GATK. For deduplication PicardMarkDuplicates, available in the Picard version that is bundled with the current version of GATK, has been used. Quality control information was gathered using Qualimap. SNVs and indels have been called using the GATK HaplotypeCaller. These variants were then functionally annotated using snpEff. The pipeline used was Piper, see below for more information.

The versions of programs and references used:
piper: unknown
bwa: 0.7.12
samtools: 0.1.19
qualimap: v2.2
snpEff: 4.1
snpEff reference: GRCh37.75
gatk: 3.3-0-geee94ec

reference: human_g1k_v37.fasta
db_snp: gatk-bundle/2.8
hapmap: gatk-bundle/2.8
omni: gatk-bundle/2.8
1000G_indels: gatk-bundle/2.8
Mills_and_1000G_golden_standard_indels: gatk-bundle/2.8

indel resource file: {Mills_and_1000G_gold_standard.indels.b37.vcf version: gatk-bundle/2.8}
indel resource file: {1000G_phase1.indels.b37.vcf version: gatk-bundle/2.8}

piper

Piper is a pipeline system developed and maintained at the National Genomics Infrastructure build on top of GATK Queue. For more information and the source code visit: www.github.com/NationalGenomicsInfrastructure/piper

8.2 Complete SNV results

In the table below, the complete results from the SNV filtration can be seen. Each cell is a list in UPPMAX containing that many variants, or genes.

	Strict Filtering	1% Filtering	5% Filtering	10% Filtering
Family 1	26919	57650	143980	266462
Family 2	10163	12486	21886	38842
Family 4	9889	10429	16208	26646
Unique Family 1 & 2	1278	1289	1443	3037
Unique Family 1 & 4	1012	1021	1217	1909
Unique Family 2 & 4	409	409	413	597
All Families	7459	7459	7459	7463
Family 1, Impactful	103	269	649	1113
Family 2, Impactful	6	30	75	128
Family 4, Impactful	4	5	36	73
Unique Family 1 & 2, Impactful	2	2	2	8
Unique Family 1 & 4, Impactful	0	0	1	2
Unique Family 2 & 4, Impactful	0	0	0	0
All Families, Impactful	0	0	0	0
Impactful Genes, Family 1 & 2	0	0	3	5
Impactful Genes, Family 1 & 4	1	1	1	5
Impactful Genes, Family 2 & 4	0	0	1	1
Impactful Genes, All Families	0	0	1	1

8.3 Complete SV Manta: Candidate Small Indels results

In the table below, the complete results connected to Mantas small indel candidate files can be found. The same setup was used here as in *Table 7* and *Table* 8. Each cell is a list in UPPMAX containing that many variants.

Manta: Candidate Small Indels	Strict Filtering	1% Filtering	5% Filtering	10% Filtering	Non-filtered
Family 1	1179	8423	17333	22056	61067
Family 2	545	3730	7654	9712	26597
Family 4	510	3774	7688	9713	26734
Family 1 & 2	377	2429	4794	6112	16863
Family 1 & 4	333	2495	5008	6309	17225
Family 2 & 4	278	1951	3783	4802	13203
All Families	234	1614	3105	3927	10832
Family 1 Moderate Impact	1	11	15	24	54
Family 2 Moderate Impact	1	6	8	11	31
Family 4 Moderate Impact	1	7	8	9	26
Family 1 & 2 Moderate Impact	1	4	5	7	19
Family 1 & 4 Moderate Impact	0	4	5	6	19
Family 2 & 4 Moderate Impact	0	5	5	6	17
All Families Moderate Impact	0	3	3	4	14
Family 1 High Impact	1	4	10	12	25
Family 2 High Impact	1	1	3	3	9
Family 4 High Impact	0	0	2	2	11
Family 1 & 2 High Impact	1	1	3	3	9
Family 1 & 4 High Impact	0	0	2	2	8
Family 2 & 4 High Impact	0	0	1	1	5
All Families High Impact	0	0	1	1	5

8.4 Complete SV Manta: Candidate SV results

In the table below, the complete results connected to Mantas SV candidate files can be found. The same setup was used here as in *Table 7* and *Table* 8. Each cell is a list in UPPMAX containing that many variants.

Manta: Candidate SV	Strict Filtering	1% Filtering	5% Filtering	10% Filtering	Non-filtered
Family 1	1259	8686	17837	22712	61067
Family 2	579	3834	7837	9934	26597
Family 4	539	3872	7867	9933	26734
Family 1 & 2	401	2500	4910	6249	16863
Family 1 & 4	355	2571	5128	6459	17225
Family 2 & 4	295	2005	3867	4902	13203
All Families	250	1664	3178	4012	10832
Family 1 Moderate Impact	1	11	15	24	64
Family 2 Moderate Impact	1	6	8	11	31
Family 4 Moderate Impact	1	7	8	9	26
Family 1 & 2 Moderate Impact	1	4	5	7	19
Family 1 & 4 Moderate Impact	0	4	5	6	19
Family 2 & 4 Moderate Impact	0	5	5	6	17
All Families Moderate Impact	0	3	3	4	14
Family 1 High Impact	1	4	10	12	24
Family 2 High Impact	1	1	3	3	9
Family 4 High Impact	0	0	2	2	11
Family 1 & 2 High Impact	1	1	3	3	9
Family 1 & 4 High Impact	0	0	2	2	8
Family 2 & 4 High Impact	0	0	1	1	5
All Families High Impact	0	0	1	1	5

8.5 Complete SV Manta: Diploid SV results

In the table below, the complete results connected to Mantas diploid SVs files can be found. The same setup was used here as in *Table 7* and *Table* 8. Each cell is a list in UPPMAX containing that many variants.

Manta: Diploide SV	Strict Filtering	1% Filtering	5% Filtering	10% Filtering	Non-filtered
Family 1	36	108	240	334	5148
Family 2	14	40	80	103	2508
Family 4	14	41	78	103	2701
Family 1 & 2	8	24	46	58	1807
Family 1 & 4	8	30	48	66	1914
Family 2 & 4	8	22	36	46	1490
All Families	4	17	27	35	1282
Family 1 Moderate Impact	0	0	0	0	36
Family 2 Moderate Impact	0	0	0	0	13
Family 4 Moderate Impact	0	0	0	0	17
Family 1 & 2 Moderate Impact	0	0	0	0	7
Family 1 & 4 Moderate Impact	0	0	0	0	7
Family 2 & 4 Moderate Impact	0	0	0	0	6
All Families Moderate Impact	0	0	0	0	3
Family 1 High Impact	0	0	0	0	137
Family 2 High Impact	0	0	0	0	57
Family 4 High Impact	0	0	0	0	67
Family 1 & 2 High Impact	0	0	0	0	33
Family 1 & 4 High Impact	0	0	0	0	39
Family 2 & 4 High Impact	0	0	0	0	30
All Families High Impact	0	0	0	0	21

8.6 Complete SV Delly results

In the table below, the complete results connected to the Delly files can be found. The same setup was used here as in *Table 7* and *Table* 8. Each cell is a list in UPPMAX containing that many variants.

Delly	Strict Filtering	1% Filtering	5% Filtering	10% Filtering	Non-filtered
Family 1	23	53	236	259	1589
Family 2	5	13	109	115	619
Family 4	8	16	102	106	625
Family 1 & 2	4	11	85	87	429
Family 1 & 4	4	11	78	82	420
Family 2 & 4	3	9	64	65	337
All Families	2	8	60	61	291
Family 1 Moderate Impact	0	0	1	1	130
Family 2 Moderate Impact	0	0	1	1	31
Family 4 Moderate Impact	0	0	0	0	34
Family 1 & 2 Moderate Impact	0	0	1	1	22
Family 1 & 4 Moderate Impact	0	0	0	0	13
Family 2 & 4 Moderate Impact	0	0	0	0	6
All Families Moderate Impact	0	0	0	0	6
Family 1 High Impact	0	3	8	9	191
Family 2 High Impact	0	0	3	3	62
Family 4 High Impact	0	0	2	2	56
Family 1 & 2 High Impact	0	0	2	2	39
Family 1 & 4 High Impact	0	0	2	2	28
Family 2 & 4 High Impact	0	0	2	2	20
All Families High Impact	0	0	2	2	19

8.7 Complete SV TIDDIT results

In the table below, the complete results connected to the TIDDIT files can be found. The same setup was used here as in *Table 7* and *Table* 8. Each cell is a list in UPPMAX containing that many variants, except those marked with a dash; these are the ones that could not be run.

TIDDIT	Strict Filtering	1% Filtering	5% Filtering	10% Filtering	Non-filtered
Family 1	109	301	803	980	331
Family 2	16	35	150	175	598
Family 4	13	47	178	212	734
Family 1 & 2	8	17	83	98	358
Family 1 & 4	8	22	105	123	398
Family 2 & 4	7	14	73	84	249
All Families	4	9	54	61	186
Family 1 Moderate Impact	0	0	0	0	-
Family 2 Moderate Impact	0	0	0	0	-
Family 4 Moderate Impact	0	0	0	0	-
Family 1 & 2 Moderate Impact	0	0	0	0	-
Family 1 & 4 Moderate Impact	0	0	0	0	-
Family 2 & 4 Moderate Impact	0	0	0	0	-
All Families Moderate Impact	0	0	0	0	-
Family 1 High Impact	2	2	102	105	-
Family 2 High Impact	0	0	32	32	-
Family 4 High Impact	1	1	38	38	-
Family 1 & 2 High Impact	0	0	16	16	-
Family 1 & 4 High Impact	1	1	21	21	-
Family 2 & 4 High Impact	0	0	13	13	-
All Families High Impact	0	0	10	10	-