INSTITUT FÜR INFORMATIK
Algorithmische Bioinformatik

Universitätsstr. 1 D–40225 Düsseldorf

Applying trio phasing to identify germline de novo mutations in children with cancer

Frederik Oehl

Bachelorarbeit

Beginn der Arbeit:	13. November 2019
Abgabe der Arbeit:	13. Februar 2020
Gutachter:	Prof. Dr. Gunnar Klau
	Prof. Dr. Martin Lercher

Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 13. Februar 2020

Frederik Oehl

Abstract

This thesis is concerned with using haplotype phasing based on trios of related individuals (mother, father, child) to detect germline de novo mutations in the child's genome. For this purpose, two human chromosomes are phased, and two methods to discover positions on which Mendelian conflicts occur are developed and tested. The first approach aims at detecting conflicts in the trio phased VCF file provided as output by WhatsHap. The second method consists of comparing the reads selected by WhatsHap for the phasing with the two haplotypes produced by WhatsHap in order to identify positions where WhatsHap placed the haplotypes differently than suggested by the reads based on information obtained from the parents' genome.
The phasings performed during the course of this thesis confirm that WhatsHap is fast and precise in conducting haplotype phasings. The results of the first method to detect mutations show a high number of probable false positives, but also a group of conflict positions that might be interpreted as a cluster of mutations on the processed chromosome in question. The results of the second method show a lesser amount of conflict positions, yet the number of positions classified as possible mutations by it is still higher than expected from the frequency of germline de novo mutations in humans. Though for some of the results, it seems unlikely that they have occurred by chance, it requires further examination to determine which of them might be the result of actual germline de novo mutations.
In addition to the tests on the two chromosomes mentioned above, the methods were later on used to process genome data which originates from the *Universitätsklinikum* Düsseldorf. The results of this second experiment differ from those of first one to the effect that a smaller number of possible mutations was detected. This is likely due to the comparatively smaller number and greater mutual distance of the variant positions in the VCF files from this dataset.

Contents

1 Introduction

In this thesis, it will be attempted to detect germline de novo mutations on human genome data through haplotype phasing involving a trio of mother, father, and child. While the data used in the main section of this thesis to test the methods outlined here is available for the public (provided by GIAB[ZCM+16]), the methods presented here are developed to conduct a search for germline de novo mutations on data from children with leukemia at the *Universitätsklinikum* Düsseldorf (UKD). At a later stage in the construction of the thesis, data from UKD became available, and the results from the processing of some of it could be included into the thesis.

Even though cancer is a multi-stage process in which, usually, a multitude of somatic mutations is involved, mutations which arise in the germline of the parents may contribute to it ([GWCD15], p.609). Also, in some cases, they can be the cause of other dangers for human health ([GWCD15], p.18). It is therefore a worthwile goal to develop methods for detecting such mutations in human genome data.

The approach taken here is based on haplotype phasing conducted with the tool WhatsHap[PMP+15] [MPG+16]. The term "haplotype phasing" refers to the process of determining on which of the two haplotypes of the human genome each variant (nucleotides which are different from the reference genome) is located, or, as Mäkinen et al.([MBCT15]) put it, "to identify the variants which are co-located on the same haplotype". The method of trio phasing or pedigree-based phasing, which is implemented in WhatsHap ([GMM16]), uses information from reads from the child's genome as well as information from the parents' genome and the rules of Mendelian inheritance to deduce the haplotypes of the child. The hope is that trio phasing can be applied to spot position-wise differences and conflicts between the genome of parents and child and can therefore be used to identify mutations in the child.

The first part of the thesis presented here consists of a summary of the method implemented in WhatsHap for the purpose of phasing haplotypes (section 2). Thereby, the Minimum Error Correction (MEC) problem and its extension, the PedMEC problem [GMM16] (MEC involving pedigree information) will be outlined. In section 3, a pipeline to conduct trio phasings on genomic data with WhatsHap will be depicted. This involves a description of the various tools involved in the processing of the data, and the way the data has to be piped through this variety of tools. Subsequently, in chapter 4, two methods for seaching positions on the genome that might contain germline de novo mutations will be presented and described (4.2 and 4.3). To clarify the issues at hand, chapter 4 also features a description of Mendelian inheritance and Mendelian conflicts (4.1). Chapter 5 features the testing of the methods described in this thesis. It begins with a description of the data used for the experiment (5.1), which consists of two human chromosomes from the Ashkenazim trio sequenced by the Genome In A Bottle (GIAB) consortium([ZCM+16]). Next follows a presentation of the results of the trio phasing conducted on the chromosomes (5.2). Then, both of the methods outlined in chapter 4 are tested on the two chromosomes, and the results of this experiment will be described (5.3 and 5.4). Next, we move to section 6, which contains a (comparatively brief) description of the results obtained from the processing of data from UKD. At the end of the thesis, the results will be summed up and evaluated in a discussion, which also features some suggestions for further work on the problem presented here in the future (section 7).

2 WhatsHap, the MEC problem, and trio phasing

2.1 WhatsHap

WhatsHap[PMP+15] [MPG+16] is the crucial tool used here for phasing the haplotypes of a genome (see also the online documentary for for the tool[1]). It takes as input a VCF file[DAA+11] containing the variants to be phased, and BAM (Binary Alignment/Map) files[LHW+09] containing reads of the genome(s) involved. WhatsHap can be run in pedigree mode, which means that genomic data from related individuals is being used to obtain a more accurate phasing of the sample under examination - speaking more formally, WhatsHap takes variants and reads from related individuals to solve the Minimum Error Correction problem on pedigrees (see subsections 2.2 and 2.3 on MEC and PedMEC below). In this thesis, the tool will be used for the phasing of a single trio (mother,father and child) with the goal of determining the haplotypes of the child. For the phasing, WhatsHap takes as input a multi-sample VCF file containing the variants of mother, father and child, BAM files for all three individuals, and a PED file containing the information about the family relations. The latter simply consists of one line of plain text, splitted into colums through tabs or white spaces. The IDs of child, mother and father have to be inscribed into colums 2,3, and 4 (from left to right) of the PED file. In order to counteract errors in the reads which may reduce the quality of the phasing, WhatsHap can additionally take a reference genome as input and re-align the reads based on the addidtional information provided by it.

As output, WhatsHap delivers a VCF file containing haplotype information in the "sample" column of the file. On positions where the sample individual is heterozygous (i.e. where two different alleles occur), reference and alternative allele are delimited by a "/" symbol in an unphased sample, and by "|" in a phased sample. "Phased" means that WhatsHap could determine on which of the two haplotypes each of the two variants is located. In other words, it is known with which alleles from other positions it shares the location on one and the same chromosome (out of the pair of chromosomes under examination). "Unphased" means that the two alleles found on the respective position are known, but could not be assigned to a specific haplotype. At phased positions in a trio phased VCF, the haplotypes of the child are given in the format "father|mother". In the VCF, the reference allele (the allele which is commonly found in the human genome on this position) is marked by "0", while alternative variants are marked by "1" . Therefore, if a child's genotype is marked as "0|1" by WhatsHap ("0" being the reference allele, "1" being the alternative allele), it means that the child has received the reference allele from the father, and the alternative allele from the mother.

[1]https://whatshap.readthedocs.io/en/latest/guide.html

2.2 The Minimum Error Correction problem

WhatsHap phases haplotypes by means of solving the weighted Minimum Error Correction (MEC) problem[GMM16].
The following definition of the MEC problem is borrowed from Mäkinen et al, 2015[MBCT15], p.316:

> Given an $r \times s$ matrix M with values $\{0, 1, -\}$, where reads are rows, columns are SNPs, and
>
> $m_{i,j} = 0$ if the ith read does not support the jth SNP,
> $m_{i,j} = 1$ if the ith read supports the jth SNP,
> $m_{i,j} = -$ if the ith read does not overlap the jth SNP,
>
> find
>
> (i) a partition of the rows of M (the reads) into two subsets R^0 and R^1, and
> (ii) two binary sequences A and B of length s (the two haplotypes),
>
> which minimize the number of bit flips in M that are required to make the reads in R^0 compatible with A, and the reads in R^1 compatible with B.
> We say that a read i is compatible with a haplotype $X = x_1, ...xs$ if, for all $j \in \{1, ..., s\}$, $m_{i,j} = -$ or $m_{i,j} = x_j$.

Concerning WhatsHap, it is important to note two additional points:
1. WhatsHap takes as input a VCF file which contains the variants for every bi-allelic position in the (part of the) genome under examination. Therefore, instead of $m_{i,j}$ equaling 0, 1 or $-$ under the conditions given in the definition, we have $m_{i,j} = 0$ if the ith read supports the reference allele, $m_{i,j} = 1$ if the ith read supports the alternative allele, and $m_{i,j} = -$ else. Apart from this, the given definition is applicable here.
2. WhatsHap solves the weighted version of the MEC problem, meaning that in addition to the $r \times s$ matrix M, an $r \times s$ matrix W, which contains costs for every position in the matrix M, is given - if a given position $m_{i,j}$ in M is being flipped, position $w_{i,j}$ in W thus describes the cost for the flipping operation. In order to solve the weighted MEC problem, the MEC problem is being solved under the additional constraint that the sum of the bitflip costs has to be minimized[GMM16].

In order to solve the weighted MEC problem, WhatsHap uses a dynamic programming algorithm, which runs in $O(2^c * M)$, with c being the maximum read coverage among all the positions to be phased, and M being the number of variants contained in the chromosome to be phased (see [GMM16] for a detailed description). It therefore gives a fixed-parameter tractable solution to the NP-hard[GMM16][MBCT15] problem (note that [MBCT15] provides a detailed example of how a dynamic programming algorithm solves the MEC problem).

2.3 PedMEC and trio phasing

In pedigree mode, WhatsHap solves the Minimum Error Correction problem on Pedigrees (PedMEC) to provide a faster and more exact result in comparison to classical phasing with a single sample. The (weighted) PedMEC problem, as described by Garg, Martin and Marschall[GMM16], is a generalization of the (weighted) MEC problem. The following definition is based on the one given in the paper mentioned above, yet a few aspects which can be found there but exceed the scope of the specific problem described in this bachelor thesis have been left out or abridged (i.e. the question of recombination costs):

As input, PedMEC takes a set of individuals $I = \{1, ..., n\}$, a read Matrix M and a weight Matrix W for each individual, a set T of triples describing the relationships of the individuals in I, and a vector χ describing the costs for recombination events at each column (or position containing a SNP). Note that for the purpose of this paper, only the phasing of a single trio is relevant. In this case, if i_1 is the mother, i_2 the father and i_3 the child, $I = \{i_1, i_2, i_3\}$, and $T = \{(i_1, i_2, i_3)\}$.
As output, PedMEC produces a set of two haplotypes $h0_i, h1_i \in \{0,1\}^s$, a set of matrix entries $E_i \subset \{1, ..., r\} \times \{1, ...s\}$ for each $i \in I$ which have to be flipped to make the respective read matrix compatible with the haplotypes, and two transmission vectors $t_{mother->child}, t_{father->child} \in \{0,1\}^s$ for each trio under examination. The transmission vectors describe which of the two alleles of a parent is transmitted to the child, with 0 denoting the copy the parent inherited from his or her father (the child's grandfather), and 1 denoting the copy the parent inherited from his or her mother (the child's grandmother). In solving the PedMEC problem, we therefore minimize

$\sum_{i \in I} \sum_{(j,k) \in E_i} W_i(j,k) + \sum_{(m,f,c) \in T} \chi(t_{m->c}) + \chi(t_{f->c})$ for a given trio $(m, f, c) \in T$

s.t. all haplotypes are compatible with the transmission vectors for their respective column[GMM16].

This means that we obtain the most cost-efficient set of bitflips and recombination events for each individual in the sample, whose haplotypes are then phased accordingly. Concerning the transmission vectors, compatibility means that the alleles on the child's haplotypes have to match the alleles transferred to the child by mother and father. Consider, for example, a case in which the bipartition of the reads indicates a child genotype of "0|1" (father|mother) at a given position, while the parents' genotypes are indicated to be "0|1" for the father and "1|0" for the mother, and the transmission vectors are $t_{f->c} = 1$ and $t_{m->c} = 0$. According to the transmission vectors, the reference allele has to be transmitted from father to child (since it is located on the fathers' second haplotype, which is the one transmitted), and the reference allele from mother to child. The child's genotype as deduced from the reads, however, indicates that the reference allele was obtained from the father, and an alternative allele from the mother. We therefore have a mismatch that has to be adjusted. This will later become important when attempting to extract germline de novo mutations via trio phasing.

As Garg, Martin and Marschall could show, phasing trios by solving the PedMEC problem significantly improves phasing results in comparison to the phasing of a single individual by solving the MEC problem: Solving the PedMEC problem (i.e. conducting a trio phasing) with a maximum read coverage of 5 per sample produced fewer phasing errors and unphased haplotypes (the latter corresponds to the length of the continuous strands of phased positions, called blocks) than solving the MEC problem with a maximum read coverage of 5 and even 15 on real and simulated data. Since the runtime of the fixed-parameter tractable algorithm used to solve the PedMEC problem, like the one solving the MEC problem, depends on the maximum read coverage[GMM16], it therefore can produce better results in less time.

3 A pipeline to conduct trio phasing on genome data

In this chapter, we will discuss how to use a variety of tools to prepare the data for processing by WhatsHap, and how to obtain a trio phased VCF file out of a triplet of VCF files describing the variants contained in the genomes of three individuals (mother, father, child), and BAM files containing reads of the three individuals' genome. Note that it is also possible to start with BAM files and a reference genome, and obtain the VCFs of the individuals by processing the reference genome (Bcftools[Li11a] provides the functionality). To prepare the data for the actual phasing by WhatsHap, a number of intermediary steps is required. I will therefore first give an overview of tools and their respective function, and then proceed to describe the process of piping the data through the different tools involved.

3.1 Tools involved in the process

3.1.1 Samtools

Samtools[LHW$^+$09] is comprised of a set of tools by means of which SAM (Sequence Alignment/Map), BAM and CRAM files can be manipulated. This includes indexing, splitting, merging and editing the files as well as changing the metadata. In the experiment presented here, samtools is primarily used to index the newly formed BAM files with modified metadata (via command "samtools index alignment.bam"), thus enabling WhatsHap to correctly process them. Note, however, that Samtools provides a host of other functions that might become useful when difficulties in the preprocessing of data occurs (see also: the online documentary for Samtools[2]).

3.1.2 Bcftools

Bcftools[Li11a][DAA$^+$11] is a sibling of Samtools, providing means for manipulating VCF and BCF files in much the same manner as the latter does for files in the BAM format (and others). In the processing of VCF files, it is necessary for the purpose described here to extract the variants of a single chromosome or limited areas of the genome ("bcftools

[2]http://www.htslib.org/doc/

filter"), merge several VCFs together to form a multi-sample file ("bcftools merge"), and also to change the metadata in reaction to difficulties with WhatsHap in processing the VCF ("bcftools annotate").
If no VCF of a sample is available, bcftools (commands "bcftools mpileup -f reference.fa alignments.bam" and "bcftools call -mv -Ob -o calls.bcf") can be used to construct one based on the BAM file(s) containing the reads and a reference genome in fasta format (see the explanation in the GitHub manual for Samtools and Bcftools[3]).

3.1.3 Tabix and Bgzip

Another tool from the same family as Samtools and Bcftools, Tabix[Li11b], can be used for indexing VCFs, BAMs and other files which contain positional genome data. For merging VCF files into multi-sample files with Bcftools, it is necessary to provide compressed index files in the gz.tbi format, requiring the Bgzip compression tool Tabix, which takes files in the GZ format as input, as part of the pipeline presented here (documentation can be found at[4]).

3.1.4 Picard

In order to make it possible for WhatsHap to match the reads from the BAM files to the VCFs, the read groups specified in the headers of the BAMs have to be identical with the sample names given in the merged VCF. For example, if the sample name belonging to the mother in a trio is "HG123", the "SM" attribute of the "@RG" tag in the header of the BAM belonging to the mother's genome also has to be referred to by "HG123". This equivalence between the names is, however, not always given in advance, and it might be necessary to adjust the tag manually (as was the case in the experiment with the Ashkenazim Trio Chromosomes 11 and 22). For this, I used the "AddOrReplaceReadGroups" tool out of Broad Institute's Picard tool set (documentation can be found at[5]), which provides various tools for working with BAM, VCF and other files. Note that adjusting the tags in the BAMs can be time-consuming.

3.1.5 Snakemake

In order to automate the processing of data and to avoid having to type in command line arguments for every single step in the workflow, Snakemake[KR12] can be used. Snakemake allows to specify rules for each step of the workflow, which (in their most basic form) consist of an input file, an output file and a shell command, which otherwise would have to be typed in by hand. For example, if you want to extract chromosome 3 from a VCF file called x.vcf via bcftools, you define a rule called "bcftools_filter" (other names are also possible), specify the input as x.vcf, define a VCF as output, and state the appropriate bcftools command in the category "shell". The rules are placed in a Snakefile

[3]https://samtools.github.io/bcftools/howtos/variant-calling.html
[4]http://www.htslib.org/doc/bgzip.1.html
[5]http://broadinstitute.github.io/picard

by the user, and a language in python format is used to define them. Snakemake connects multiple rules by a directed acyclic graph, through which it works its way recursively from the final rule ("rule all"), which takes as input the result of the entire pipeline, to the first rules which take the unprocessed files as input. This means that in order to execute "rule all", Snakemake will search for the input required for the rule, and will use the output by one or more other rules for this purpose if it matches the input's name (and if it exists). In order to execute the rule which delivers this output, Snakemake proceeds in the same way (i.e. searches for the fitting input), until it has worked its way back to the original input, which is not produced by a rule.
During the process of constructing the Snakemake pipeline for the experiment described in this thesis, I decided to exclude the processing of the BAM files with Picard from the Snakefile. The first reason for the decision is that this part of the preprocessing can take a long time, and should be done separately so that the phasing can be conducted on small sized jobs with short waiting time on the university's High Performance Cluster (HPC). The second reason is that it is not always necessary to process the BAMs - it depends on whether the data by means of which the read groups are identified is consistent with the information in the VCF or not. I therefore decided to conduct the processing of the BAM files manually, via command line. Included into the Snakemake pipeline are: the merging of the VCFs (which requires a multitude of intermediate steps, all with low runtime), the phasing itself (which includes the code extension for WhatsHap described in section 4.3), and the program described in section 4.2.

3.2 Conducting the phasing

Depending on the properties of the data, different steps on processing can be necessary to prepare it for the phasing with WhatsHap. The following description is largely based upon experience with the phasing of data from the Ashkenazim trio, which is provided by the Genome in a Bottle consortium[ZCM+16], and available online (source for the data:[6]).

3.2.1 Merging the VCFs

Given are three VCF files from individuals which form a trio, each containing the variants from the entire genome of the individual it belongs to. Here, it is assumed that the VCFs are available in compressed format, and indexed (the first rules of the Snakefile take gzipped VCFs as input). Otherwise, they have to be compressed by Bgzip and then indexed by Tabix at the beginning of the process. First, the subsection of the genomes we actually want to phase has to be extracted from the VCFs, what can be done with Bcftools. For example, if we want to phase chromosome 1 of a trio, "bcftools filter -r 1" can be used (it is also possible to extract smaller regions of a chromosome, reaching from a specific position which can be given to bcftools via command line to another). The resulting VCFs then (again) have to be compressed into the gz format using Bgzip to enable the next step of processing. In this next step, the information contained in the FORMAT

[6] ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/

column of the VCFs is being removed using the "bcftools annotate -x" command. The reason for this is that it proved to be a source of error when trying to merge the VCFs, since some positions in the files differ from the format specified in this column (for a description of a frequently occurring error, see for example[7] in the samtools manual). Since the information presented in the FORMAT column is not important for the purpose of this experiment, I concluded that removing it is the best option (for a description of the information contained in the FORMAT column see the VCF format specification[8]).
Next, the modified VCFs received as output of the "annotate" step again have to be compressed into the gz format using Bgzip, and an index file has to be created for each with the help of Tabix. Now, we are ready to merge the three VCFs into a multi-sample file, again using Bcftools. With the "merge" command, Bcftools takes as input the three compressed files (vcf.gz format) and gives the multi-sample VCF as output. Note that if the "-0" flag is set, Bcftools interprets missing genotype information at a given position in a VCF as indication that the individual involved is homozygous and carries the reference allele. For example, if on position x, we have genotype information for the mother and the child, but not for the father (i.e. the father's VCF does not contain genotype information), we will get "0/0" for the father in the multi-sample file if the flag is set, and "./." (missing genotype) if it is not set. This is important since WhatsHap does not phase positions on which one or more genotypes are missing.
The merging of the VCFs has been automated with the help of Snakemake, avoiding the repetition of all the intermediate steps via command line arguments for every phasing.

3.2.2 Processing the BAMs

For the BAM files to be feasible for processing by WhatsHap, it is important that the sample names of the read groups provided in the "SM" subsection of the "@RG" header in each respective BAM file match the sample names in the multi-sample VCF and the PED file. Only then can WhatsHap identify which reads belong to which genome, and successfully phase them. Since this condition is not always fulfilled in the data, the names have to be adjusted by using the "AddOrReplaceReadGroups" function from the Picard tool set. When using "AddOrReplaceReadGroups", a single BAM file has to be provided as input, and all subsections (RGLB, RGPL, RGPU, RGSM) of the "@RG" tag in the file have to be overwritten (RGSM refers to the SM subsection, where the old name has to be replaced by the sample name used in the VCFs and the PED file). Note that the BAM files are very large (the combined size of the three files for chromosome 22 of the Ashkenazim trio is about 50GB, while those of chromosome 11 comprise about 140GB), and processing them can therefore take a considerable amount of time.
In the next step, an index file for each of the modified BAMs has to be created ("samtools index" can be used here). Afterwards, the BAMs are ready to be processed by WhatsHap.

[7]http://samtools.github.io/bcftools/howtos/FAQ.html#incorrect-nfields
[8]http://samtools.github.io/hts-specs/VCFv4.3.pdf

3.2.3 Phasing with WhatsHap

In pedigree mode, WhatsHap takes as input the VCF multi-sample file, the BAM files, and a PED file containing information about how the samples are related (for one trio, one line of plain text, designating one sample as the child, one as the mother, and one as the father, is sufficient). In addition, a reference sequence can be given to WhatsHap in order to enable the tool to re-align reads during the phasing and reduce the number of errors.
Given correct input, Whatshap then first reduces the number of reads to be used for the phasing in a preprocessing step. To ensure a quicker phasing process, WhatsHap thereby reduces the maximum coverage (i.e. the maximum number of reads from the BAM files covering a singe position) to 5 per sample as a default option. Moreover, only reads that cover at least two variants are included in the phasing. Note that positions with Mendelian conflicts that can be detected before the phasing are not phased by WhatsHap (see section 4.2).
By solving the PedMEC problem, whatshap then creates a VCF file which contains the desired phasing as output, sorting the child's alleles accordingly as father|mother for every heterozygous position (and the parents' haplotypes are being phased as well). It is this file we will work with in section 4.2. For a graphical illustration of the phasing process, see Figure 1.

4 Detecting mutations

4.1 Mendelian inheritance and conflicts

Mendelian inheritance, which, despite its age, still forms the basis of the modern-day understanding of inheritance ([GWCD15]), provides the framework for the search for mutations applied in this thesis. According to the patterns of genetic inheritance first described by Mendel, every unit containing hereditary information (i.e. every gene) appears twofold in living organisms (exceptions do exist, but the rule holds for *Homo sapiens*). These two copies of the same gene, called alleles, can be identical, yet it is also possible that one (or both) genes carry mutations, which can change the function of a gene and therefore the characteristics found in an organism's phenotype. Since genes are ultimately composed of nucleotides, the condition of diploidity (i.e. the existence of two copies) also holds true for nucleotides. Concerning the genome as a whole, it follows that a human individual (as many other organisms) possesses two haplotypes (i.e. sets of chromosomes which comprise the genome). The goal of haplotype phasing can in turn be described as to determine for each heterozygous position (i.e. each position where two different nucleotides appear on the two strands), which of the two variants belongs to which haplotype, and therefore "which variants are co-located on the same haplotype" ([MBCT15], p.315).

In the process of inheritance, organisms receive one copy of a gene from each of their respective parents, since the germline cells which form eggs and sperms are haploid, not diploid as the cells which comprise the organism([GWCD15]). Phasing

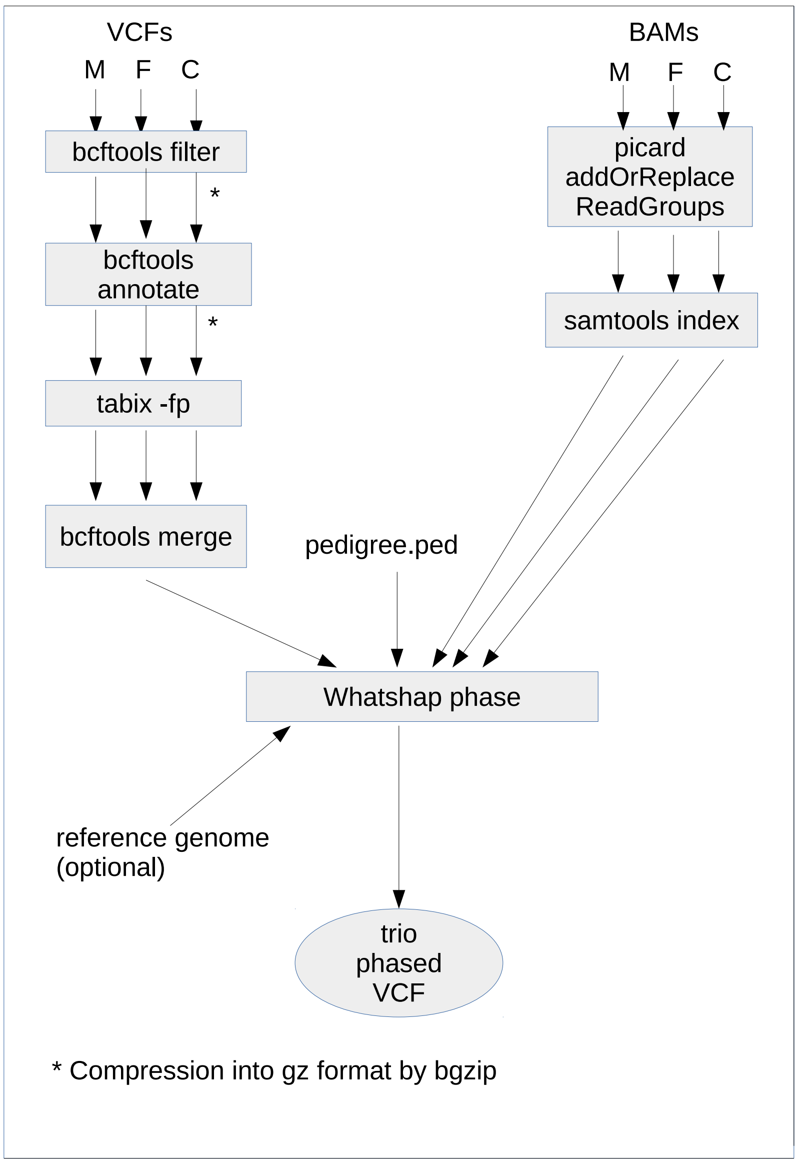
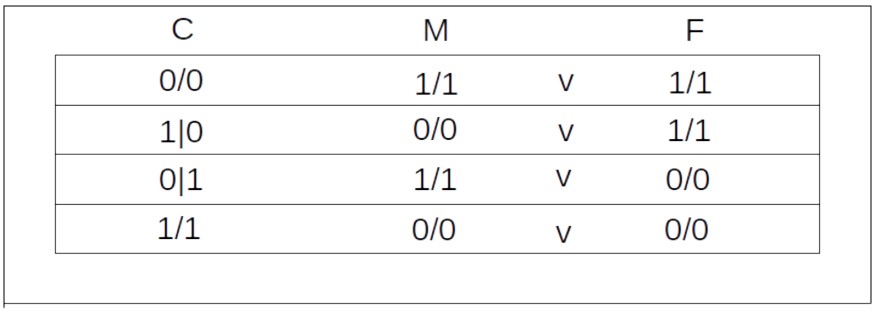


Figure 1: *A graphical illustration of the phasing process*

haplotypes based on trio information allows to determine which of the two nucleotides on each heterozygous position in the genome stems from the father, and which from the mother of a given child. This in turn opens the possibilty to compare the child's nucleotides at a given position directly to those that can be found in the parents. If, on a given position, a nucleotide appears in the child which is not present in the respective parent, this can be classified as a *Mendelian conflict*, since it contradicts what is expected to be the result of an inheritance process (Note that the focus here is on point mutations, or single nucleotide polymorphisms (SNPs), and insertions and deletions, which can also be traced through variant calling, are not included here). If we assume that the phasing leading to such a discovery gave correct results, and if we furthermore can assume that the given mutation did not appear as part of a biological process that occured at a later stage of the child's development, we have reason to believe that it originated in the germline of the parents. Collecting positions which contain Mendelian conflicts, and are therefore candidates for the occurrence of a germline de novo mutation, from the results of haplotype phasing with WhatsHap is the purpose of the programs described in the next subsections.

It is important to note that the number of germline de novo mutations is small in comparison to the size of an entire genome: In a 2012 study (a summary can be found in [GWCD15], p.17f.) conducted on 219 individuals from 78 trios (including 78 children), an average of 63 de novo mutations where found in each child. 62 of the 4933 mutations found in total, were, however, classified as a potential health risk.

In a VCF file, the value "0" is assigned to an allele which is identical to the one found in a reference genome on a given position, whereas "1" is assigned to an alternative allele (values greater than 1 are also possible, if more than one alternative allele is assigned to a given position). Figure 2 gives a short overview of Mendelian conflicts on a given position in a trio. Conflicts on phased heterozygous positions of the child (C) are not visible in the VCF (see 4.3 for a more detailed description and the method for finding these mutations), while conflicts on positions where the child is homozygous are visible. However, it should be noted that mutations can occur which do not lead to a visible conflict at all: If, at a given position, a parent is heterozygous with "0/1", and we know the allele the child inherited from the parent is "0", it is possible that the child has not received a "0" allele from the parent, but a "1" allele that by pure chance mutated to "1" in the germline (analogous if the child's allele is "1").



C	M		F
0/0	1/1	v	1/1
1|0	0/0	v	1/1
0|1	1/1	v	0/0
1/1	0/0	v	0/0

Figure 2: *Directly visible Mendelian conflicts between child, mother and father. The conflicts on positions where the child is heterozygous are not visible in a trio phased VCF and have to be accessed differently.*

4.2 Tagging conflict positions in the phased VCF

In the phasing process, WhatsHap skips those mutations conataining Mendelian conflicts, if the conflicts are visible prior to the phasing (for example: "0/1" in the child and "1/1" in both the parents). in order to regain them, a self-written Python script is used here. As input, it takes the trio phased VCF and the sample names of child, mother, and father, and as an output, it produces two text files containing the positions and variants of candidates for mutations. The choice of candidates is based on whether a given position has been excluded from the phasing by Whatshap, and whether there exists a Mendelian conflict on it, whereby two different categories of mutation candidates (meaning: more and less likely candidates for mutation) are being distinguished here.

As a first step of processing, the program writes the data from the phased VCF into a two-dimensional array with the help of the pyVCF library (see pyVCFs online user guide[9] for detailed information). The further steps of the preprocessing and the subsequent search for mutations are being conducted on this array, which contains the child's reference and alternative allele as well as all genotypes of the trio for every position in the trio phased VCF, while the original VCF remains unchanged. To simplify the subsequent search, the alleles are then grouped according to the results of the phasing for every heterozygous position: If, for example, the child is "1|0" (meaning: alleles transferred from "father|mother"), the alternative allele will be switched into the second, and the reference allele in the third column of the array (the first column contains the position on the chromosome). In the end, the second column in the array contains all alleles inherited from the father, and the third one all alleles stemming from the mother. Next, all positions containing insertions and deletions are being removed from the array. Note that this step is optional - it is possible to include those positions of the VCF that contain indels into the search.

Now, the search for Mendelian conflicts can begin, whereby the program collects those conflicts sorted out by WhatsHap. As depicted in Figure 2, there are a number of different, directly visible mutation candidates. Those on which the child is heterozygous are, unfortunately, not identifiable in the phased VCF (see section 4.3.1 below), and therefore, the program looks for all positions where the child is homozygous ("0/0" or "1/1") or heterozygous and unphased, and Mendelian conflicts appear. However, practical experience with scanning the trio phased VCFs from the two chromosomes regarded in this thesis strongly suggests a further distinction. To be precise: in those positions where missing genotype information has been replaced by "0/0" (as described in section 3.2.1), many more conflicts than expected appear (namely several thousands on both chromosomes from the Ashkenazim child; compare to the empirical data referenced above in section 4.1), leading to the conclusion that those positions are error prone (for more details: see below in section 5.3). Since among the probably erroneous positions, there might still be actual mutations, I decided to not simply leave them out. Instead, the program now distinguishes between two categories: In the first, all those conflicts where the child is heterozygous and unphased, and the parents are homozygous on the alternative allele ("0/1", "1/1", "1/1") are collected, and all positions involving an individual with "0/0" in the second. Additionally, the program also provides a function that separates the three kinds of conflicts involving "0/0" positions, in order to count and collect them separately.

Note that it is also possible to not interpret missing positions as "0/0" in the merged VCF, though, as a consequence, WhatshHap skips over all those positions where the geno-

[9]https://pyvcf.readthedocs.io/en/latest/

type for either child, father, or mother is missing. They are therefore included into the VCF as unphased. In the course of writing the paper, I tried several different heuristical approaches to obtain an approximate phasing of those unphased positions - the results, however, were not promising (i.e. there always occurred a much higher number of apparent conflicts than expected) and have therefore not been included into the thesis.

4.3 Tagging conflicts by comparing reads and haplotypes

In order to obtain more mutations from the phasing than described above, we have to move our focus away from the VCF and towards intervening into the phasing itself. This is due to the method used by WhatsHap when constructing the VCF, as will be described in the following.

4.3.1 WhatsHap's compatibility constraint

As mentioned above in section 2.3, WhatsHap constructs the haplotypes of all positions under the constraint that they are compatible with the transmission vectors, which describe which alleles are transferred from the parents to the child. This means that if, on a given position, the bipartition of the read matrix suggests two haplotypes that are different from the ones obtained through the transmission vectors (i.e. more than half of all non-empty entries in a column of one of the two subsets would have to be flipped to obtain haplotypes identical with the ones which match the transition vectors), the entries which have to be flipped to match the transition vectors are all included into the set of entries E (see 2.3.). Now, if a position contains a germline de novo mutation, this means precisely that the allele present in the child on a given position is different from the allele inherited from the parent. From this, it follows that mutations on positions where the child is heterozygous and not sorted out by WhatsHap before the actual phasing (and which are therefore phased) are lost in the phasing process.
For example, if on a given column in the read matrix, haplotypes of "0 | 1" (father | mother) would require the minimum amount of bit flips, yet according to the transmission vector, the allele transmitted from the father had to be "1" and the one from the mother had to be "0", WhatsHap would flip the alleles to "1 | 0", even though in the subset of the matrix column associated with the alleles, there are more 0's than 1's in the subset assigned to the haplotype inherited from the father (and vice versa for the subset assigned to the one inherited from the mother).
As a consequence, the described kind of mutation cannot be detected by searching through the trio phased VCF, what severely limits the usefulness of post-phasing attempts to detect germline de novo mutations. It therefore has to be attempted to catch more mutations during the phasing process itself - a method to do this will be described in the following.

4.3.2 A method to regain the lost conflict positions

If the positions of possible mutations where the child is phased as "0|1" or "1|0" (heterozygous) are to be obtained, an intervention into the logic of WhatsHap offers new opportunitiues (the source code of WhatsHap can be found on Bitbucket[10]). In WhatsHap, the different parts of the phasing process are executed by a single method (the method "run_whatshap" in the "phase" class), and a number of different objects and data structures provided by WhatsHap's core logic are used.

By adding suitable code to the method, the child's haplotypes as well as the variants of the reads which are used to conduct the phasing can be extracted from the program during the trio phasing itself. Obtaining the data from WhatsHap opens up the possibility for a direct comparison between the haplotypes determined by WhatsHap and the information in the reads. If, at a given position, a lot of the variants found in the reads do not match with the variants on the haplotypes determined in the phasing (for example, a large number of reference alleles or 0's in the reads while, according to the phasing, the child has the alternative allele (1) on the given haplotype, and a large number of 1's on the haplotype where the child has the reference allele), this is an indication that the transmission vector does not match with what the reads indicate. We would therefore have found a position where a mutation, invisible in the trio phased VCF, might possibly have occurred.

Now follows a more detailed description of the method. All functionality described here is placed in the "run_whatshap" method in the class "phase.py" in WhatsHap. It is executed after the PedMec problem has been solved on the input given to WhatsHap, since the haplotypes and reads selected by WhatsHap are taken as an input.

First, the phased haplotypes of the trio have to be accessed. They can be obtained via the "PedigreeDPTable" object from Whatshap's Cython wrapper class ("core.pyx"), which returns a list of three haplotypes (which in turn consist of the two haplotype strands 0 and 1 of the given individuals). In the course of a series of test runs, it came to light that the three haplotypes are ordered differently in every run, and therefore, it has to be checked which of the three haplotypes is the child's (this can be checked since the haplotypes are built out of reads which have a sample ID ("0" stands for the child)).

In the next step, the variants from all the reads selected by WhatsHap for phasing have to be matched to the positions of the child's haplotypes. Note that both of the child's haplotypes are stored within a single list, of which the first half stands for haplotype 0, and the second half for haplotype 1. The reads are provided by an object named "all_reads", in which all those reads from the BAM files which were selected by WhatsHap as optimal for the phasing are stored. By checking the sample ID of each read, it can be found out which reads belong to the child. In order to be able to detect positions which contain conflicts between the optimal partitioning (the optimal sets of reads selected by WhatsHap for each haplotype), and the variants on the haplotypes, the optimal partitioning itself is accessed. It is assumed here that on positions where

[10]https://bitbucket.org/whatshap/whatshap/src/master/

half of the variants provided by the optimal partitioning do not match with the variant on the haplotype they are assigned to, the haplotypes have been flipped because of a conflict with the information provided by the transmission vectors. To obtain the optimal partitioning, a getter from "PedigreeDPTable", which provides an ordered list describing to which of the two partitions each read from the reads selected by WhatsHap belongs, is used. Based on this information, it is assessed with which of the two haplotypes on a given position a variant on a given read has to be compared.

After the variants on the reads have been matched to the variants on the haplotype and saved in an array, conflicts have to be searched. "Conflicts" or "mutation candidates" here mean such positions where, for both variants on the haplotype, at least half of the variants which have been assigned to it in the previous step do not match it. In order to avoid false positives, only positions with a minimum coverage on both haplotypes are included into the output are being examined (the minimum coverage can, for example, be set to 2, or be determined to be 3 on one haplotype, and 2 on the other). This means that at least a certain, fixed amount of variants has to cover each of the two positions. "False positives" here means conflicts that appear by chance due to low coverage. For example, if, on a position x, the haplotypes are "0|1", with two variants "0" and "1" covering the first, and two variants "0" and "1" covering the second haplotype, this would technically be a conflict position, yet it is questionable whether it should be classified as a mutation candidate.
This restriction can, however, easily be changed so that positions with lower coverage are examined if desired. After the selection step, the conflict positions are handed out in a text file. In order to verify whether the program works correctly, the entire array containing the haplotypes, their positions and all variants assigned to them, is provided in another textfile.

It has to be mentioned that placing code within the "run_whatshap" method is accompanied by some difficulties regaridng the time needed for testing and improving the code. Since objects from the core of WhatsHap (e.g. reads and readsets), written in C++ and accessible through wrapper classes written in Cython, have to be accessed and processed by the new code, it is necessary to execute a phasing for every test run that is being conducted to look for possible errors and things to improve. The process can, however, be fastened by using smaller amounts of data (e.g. only a merged VCF of a limited part of the chromosome). Despite the restriction described, the code was tested to the best of the author's abilities using the university's High Performance Cluster (HPC).

Concerning runtime, it is also important to note that since all the reads belonging to the child have to be phased in order to extract every existing variant for each position in the haplotype, the method described here is altogether slow (a complete phasing for chromosome 22 took about 3:10 hours with the code extension, compared to 23 min without it), and when working with larger chromosomes, it is necessary to divide them into several parts which have to be processed in succession, since the runtime is not proportional to the size of the input data, but depends on the number of variants on the two haplotypes times the total number of variants on the reads examined.

4.4 "Sneaky" mutations

Another challenge is posed by the kinds of mutations which are not visible as Mendelian conflicts between parent and child because the parent is heterozygous (e.g. where the "1 | 0" child receives a "0" from the "1 | 0" father, which mutates into a "1" in the germline). Since I have, as for now, not worked on a practical method to catch those mutations, and for reasons of brevity, I do not include ideas of how to detect them into this thesis. It can, however, be noted, that this is a possible field of interest for future modifications on WhatsHap.

5 Experiment with the GIAB data and results

5.1 The data

In order to conduct phasings and to test the methods for finding germline de novo mutation candidates, data from the Ashkenazim trio, provided by the Genome In A Botte (GIAB) Consortium[ZCM+16], has been used. The data can be openly accessed online under the address provided in section 3.2.
The VCF files used (to be found at[11]) were constructed from variant callings conducted in two different technologies and contain Single Nucelotide Polymorphisms (SNPs) as well as insertions and deletions. The reads (BAM files) are provided by Pacific Biosciences (PacBio), and can be downloaded at[12].

For the experiment, the chromosomes 11 and 22 of the Ashkenazim trio have been used. The combined VCF for chromosome 22 (merged from the VCFs of child, mother, and father as described in section 3.2) contains 61.984 variant positions in total (counting provided by WhatsHap), which are distributed over a length of 35.167.312 base pairs (bp) on the chromosome, with the first base pair contained in the VCF being found on position 16.054.740 on the chromosome and the last one on position 51.222.052 (as visible in the merged VCF itself). On average, therefore, one out of 567 base pairs (or, respectively, positions on the chromosome) contains a variant, if one counts from the first to the last position containing a variant, not the first and the last position on the entire chromosome. From the fact that the first position containing a variant described in the VCF is about 16 million bp distant from the beginning of the chromosome, it can be deduced that a large part of the chromosome is not covered by the VCF. This is already the case in the online data (where position 16.054.740 is already the first one on chromosome 22 where a variant can be found), and not a result of the merging or phasing process. The reason for this is unknown to me, though it seems unlikely that not a single variant should be found on that vast stretch of the chromosome, if you compare with the frequency of variants on the other parts of the chromosome.

For chromosome 11, the merged VCF contains 246.465 variants in total, with the

[11]ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/analysis/NIST_CallsIn2Technologies_05182015/
[12]ftp://ftp-trace.ncbi.nlm.nih.gov/giab/ftp/data/AshkenazimTrio/HG002_NA24385_son/PacBio_MtSinai_NIST/MtSinai_blasr_bam_GRCh37/

first base pair contained in the VCF being found on position 192.155 on the chromosome and the last one on position 134.946.070. In the case of chromosome 11, thus, one out of about 547 base pairs contains a variant (again, not counting the beginning of the chromosome before the first detected variant). Note that in the case of chromosome 11, the missing part at the beginning of the chromosome is much smaller than in the case of chromosome 22, comprising only about 200.000 bp.

In the course of the experiment, first, trio phasings of the two chromosomes have been conducted according to the method described in section 3. Afterwards, the trio phased VCFs have been searched for Mendelian conflicts according to the procedure described in section 4.2. Later, the method for finding mutation candidates described in 4.3 has been tested on both chromosomes - on chromosome 11, the method was only be used on the first 30 percent of the VCF after splitting the file up. Splitting larger VCFs into parts is necessary due to the long runtime required otherwise. All phasings have been conducted on the university's High Performance Cluster (HPC).

5.2 Results of the trio phasings

WhatsHap attempts to phase a maximum number of heterozygous positions from the merged VCF (homozygous positios cannot be phased for the trivial reason that the alleles on both haplotypes are the same). If positions contain Mendelian conflicts or insertions and deletions, they are not phased, but are still included in the trio phased VCF (they are marked with "x/y" in the GT column, in contrast to the "x | y" of the phased positions, x and y standing for 1 or 0 here).

For chromosome 22, out of the variants, 27.920 are heterozygous in the child, with 25.005 being SNPs and 2.915 insertions or deletions. Of the variants, 24.328, or 87.1%, have been phased, while 3.592 remained unphased (see Table 1). Of the unphased variants, 2.631 have been left out due to Mendelian conflicts. All the phased variants have been assigned to a single block by WhatsHap, meaning that on the whole chromosome, all neighbouring variants are connected by a read covering both of them (cf. [GMM16]; Note that the phasing has been conducted with a maximum read coverage of 5 reads per position, which is the default for WhatsHap in trio phasing mode). Therefore, one can say that the phase information is consistent for the entire chromosome 22. Regarding the extent covered by the largest (and only) block, it does, according to the numbers given by WhatsHap after the phasing process, cover the positions from 16.933.993 and 51.222.052. It therefore stretches right to the last position contained in the VCF, while between the first position contained in the VCF and the first position on the block (which is also the first phased position), there is a gap of about 900.000 bp were the positions contained in the VCF are either homozygous or unphased. A look inside the VCF file reveals that it contains only nine positions for this part of the chromosome.

It is interesting to note that if the "-0" flag of Bcftools is not set in the process of merging the VCFs, and missing genotypes in any individual are interpreted as "./.", this changes the result considerably (see Table 2). In this case, only 8.562 out of the 27.920 blocks, or 30.7%, have been phased, and the variants were assigned to 19 different blocks with an average size of about 450 variants, and a median size of 9 variants. Even though the largest block still consists of 8312 variants, and therefore comprises most of the phased positions (on the chromosome, it covers a total length of 34.021.750 bp), the overall result is clearly less satisfying. Therefore, even though Bcftools' "-0" flag is of limited use for the purpose of detecting germline de novo mutations, it can be considered helpful for the phasing itself.

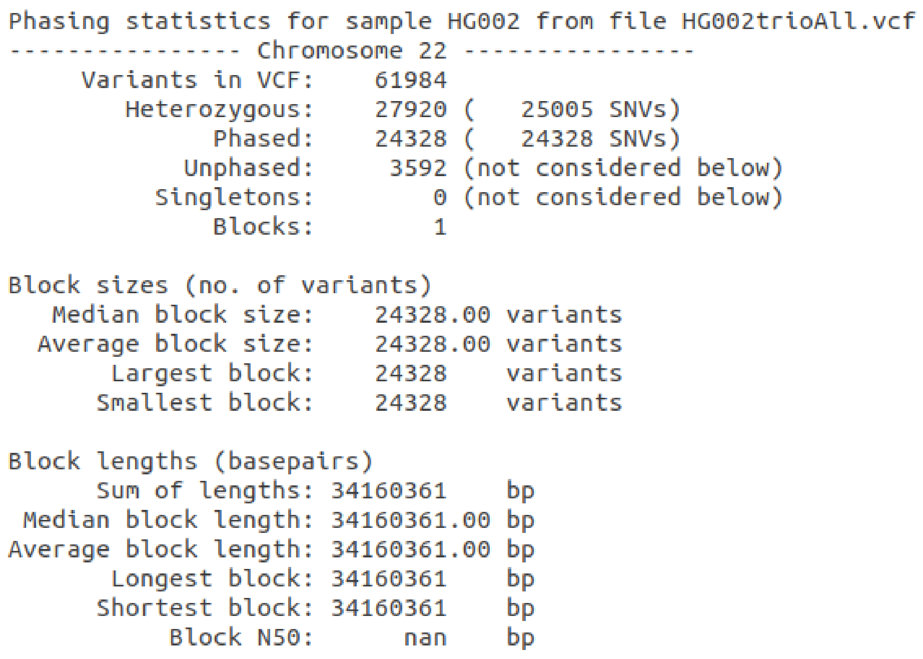
For chromosome 11, out of the variants, 100.457 are heterozygous in the child, with 90.168 being SNPs and 10.289 insertions or deletions. Of the variants, 88206, or 87.8%, have been phased, while 12251 remained unphased (see Table 3) Of the unphased variants, 9133 have been left out due to Mendelian conflicts. In the case of this chromosome, which contains almost four times as many variants as chromosome 22, the phasing resulted in two blocks, the smaller one of which consists of only three variants. The larger of the two blocks consists of 88203 variants (on the child) and stretches from position 193.096 to 134.945.765 on the chromosome, containing all variants save for small amounts, which can be found within a range of about 1.000 bp from the very first and about 300 bp from the very last variant contained in the VCF.

Concerning runtime (with both of the phasings having been conducted on the HPC, on a job involving a maximum memory of 25 GB), the phasing of chromosome 22 took a total time of 1336.5 seconds (22:17 min), whereby 822.6 seconds (13:43 min) were needed for reading the BAM files, 476.6 seconds (7:57 min) for the phasing itself (i.e. constructing the haplotypes from the information in the BAM and merged VCF files), and 36.5 seconds for other purposes such as selecting the reads used for the phasing, and parsing the VCF. This means about 61.5% of the time were used for reading the BAMs, 35.5% for the phasing itself, and 4% for other purposes.
The phasing of chromosome 11 took 5248.0 seconds (87:28 minutes) in total, of which 3198.2 seconds (53:18 min) were consumed by reading the BAMs, 1849.0 (30:49 min) by the phasing itself, and 200.7 (3:21 minutes) by other operations. Thus, about 60.9% of the time were used for reading the BAMs, 35.2% for the phasing itself, and 3.9% for other purposes.

5.3 Results from tagging conflict positions in the VCF file

On the whole, the program which searches mutations within the trio phased VCF detected 2632 conflicting positions on the VCF belonging to chromosome 22, and 9136 conflicting positions on chromosome 11. Both runs have been conducted on VCFs where the missing genotypes had been replaced by "0/0" (for a test run on a VCF where the missing genotypes had not been replaced see the end of the subsection). In both cases, the number of discovered conflicts is slightly higher than the number of conflict positions skipped by WhatshHap (2631 and 9133). The reason for this is unclear; duplicate positions in the VCF file are a possible explanation.

More importantly, the number of conflict positions is extremely high compared to what would be expected from possible mutations (see section 4.1). In order to identify the likely reason for the high number, it is hepful to look for the number of occurrences

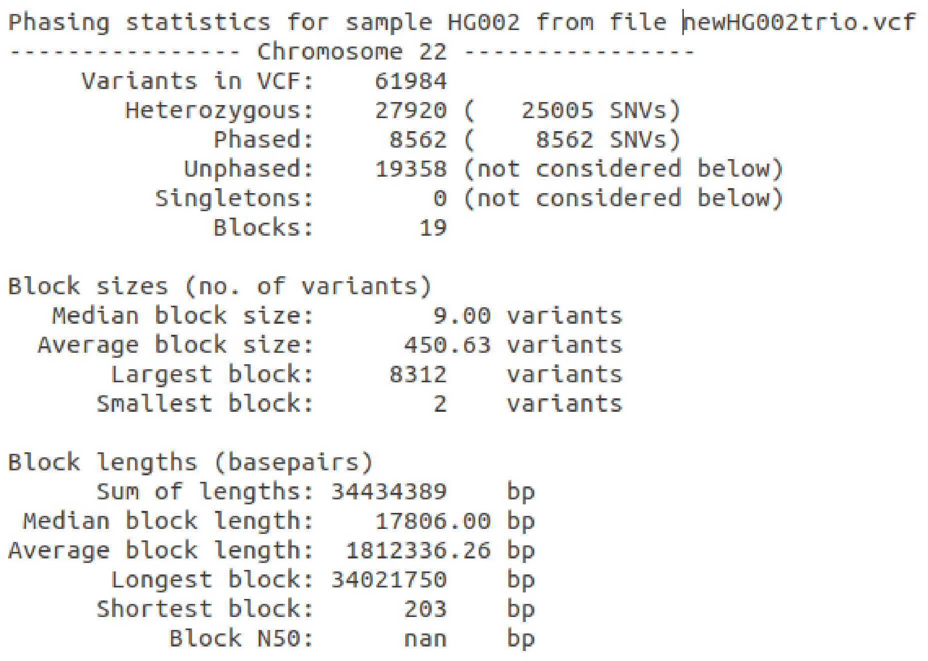


```
Phasing statistics for sample HG002 from file HG002trioAll.vcf
---------------- Chromosome 22 ----------------
     Variants in VCF:    61984
        Heterozygous:    27920 (   25005 SNVs)
              Phased:    24328 (   24328 SNVs)
            Unphased:     3592 (not considered below)
          Singletons:        0 (not considered below)
              Blocks:        1

Block sizes (no. of variants)
   Median block size:    24328.00 variants
  Average block size:    24328.00 variants
       Largest block:    24328    variants
      Smallest block:    24328    variants

Block lengths (basepairs)
      Sum of lengths: 34160361    bp
 Median block length: 34160361.00 bp
Average block length: 34160361.00 bp
       Longest block: 34160361    bp
      Shortest block: 34160361    bp
           Block N50:      nan    bp
```

Table 1: *The results of the phasing of chromosome 22 (obtained via the "stats" command of whatshap). Note that all phased variants have been assigned to a single block. The unphased positions are mostly due to conflicts involving missing variants interpreted as "0/0", and due to insertions or deletions being found on the position in question.*

of the specific types of conflicts detected by the program. The four different types are listed in Table 4 below, and show a clear pattern. Conflicts where the child is "0/1" and both parents are "1/1" are altogether rare, and those that do occur are concentrated in a small region of chromosome 22. In contrast, all those types of conflicts which involve positions on which the genotypes were missing in at least one individual, and where the genotypes in said individuals were replaced by "0/0" for the sake of the phasing, occur much more often than would reasonably be expected. It can therefore be said that the strategy of replacing missing genotypes by "0/0", useful as it is for the phasing itself (see section 5.2), does not prove useful in the search for mutations. In order to get feasible information about the positions where genotypes are missing, at least concerning those where more than one individual is affected, the actual genotypes on the missing positions would first have to be regained.

Yet, as mentioned above, the program also identifies conflict positions in which the child is "0/1", while both parents are "1/1". As Table 4 shows, the results for this type of conflict are quite different. While on the entire chromosome 11, none could be localized, 24 occur on chromosome 22. A look at the exact positions of the conflicts (Table 5) reveals that they all occur in close neighborhood, namely within 13.000 bp on the chromosome, while a length of over 34.000.000 bp was covered by the phasing on the chromosome. In other words, all conflict positions occur within an area that covers less than 0.04% of chromosome 22. On the rest of the two chromosomes phased, conflicts of the type discussed here are absent (which is a result that is in line with the low frequency



```
Phasing statistics for sample HG002 from file newHG002trio.vcf
---------------- Chromosome 22 ----------------
     Variants in VCF:    61984
        Heterozygous:    27920 (   25005 SNVs)
              Phased:     8562 (    8562 SNVs)
            Unphased:    19358 (not considered below)
          Singletons:        0 (not considered below)
              Blocks:       19

Block sizes (no. of variants)
   Median block size:        9.00 variants
  Average block size:      450.63 variants
       Largest block:     8312    variants
      Smallest block:        2    variants

Block lengths (basepairs)
      Sum of lengths: 34434389    bp
  Median block length:    17806.00 bp
Average block length:  1812336.26 bp
       Longest block: 34021750    bp
      Shortest block:      203    bp
           Block N50:      nan    bp
```

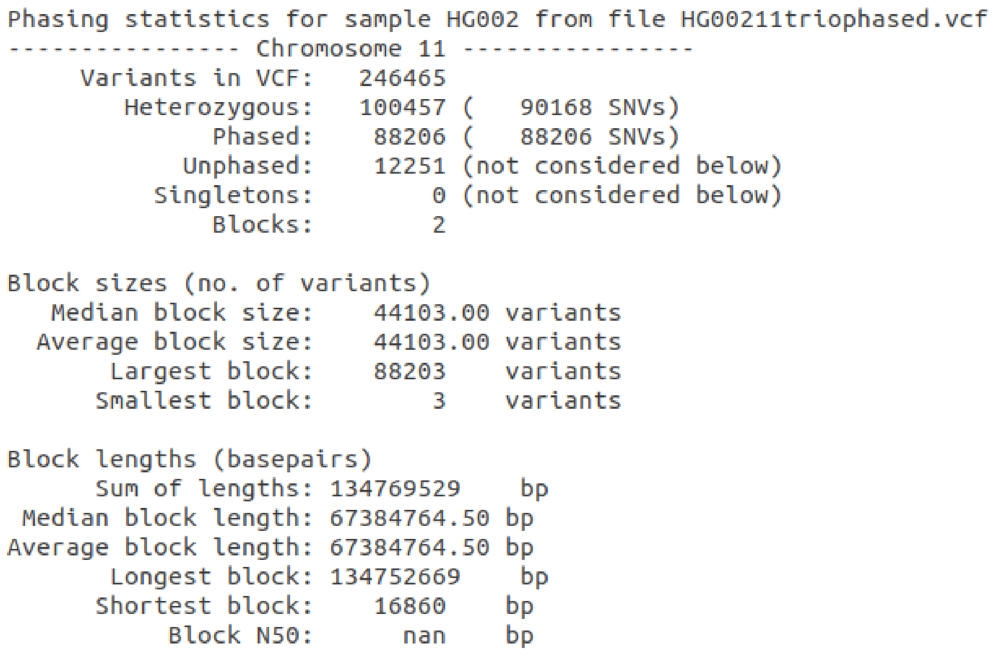
Table 2: (referring to 5.2) *The results for chromosome 22, when missing genotypes are interpreted as "./.", and not as homozygouos for the reference allele. Far fewer positions have been phased (8.562 compared to 24.328), and the phasing is considerably more fragmented (19 blocks instead of one).*

of germline de novo mutations in humans mentioned in 4.1). The conflicts might be the result of errors in the variant calling, yet it is also possible that the conspicuity lies in the genome itself, what would make the "0" alleles found in the child the result of mutations.

For chromosome 22, the code for tagging mutations has also been tested on a VCF where the missing genotypes had not been replaced in the merging process, and were therefore interpreted as "./.". In this case, the program found only the aforementioned 24 conflicts of the type ("0/1","1/1","1/1"), and all the other conflicts were, as expected, absent (since positions containing "./." are not counted as conflicting by the program).

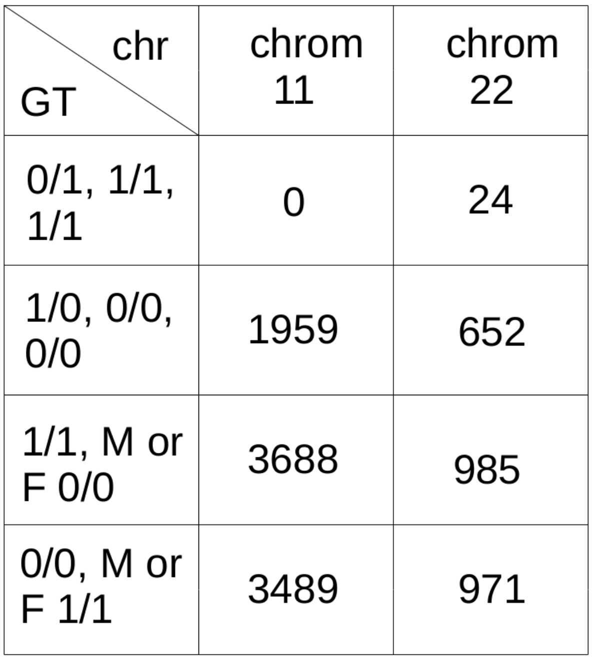
5.4 Results from extracting conflicts directly from WhatsHap

Both chromosome 11 and chromosome 22 were processed with the code extension for WhatsHap described in 4.3.2. While chromosome 22 was processed fully, for chromosome 11, only positions within the first 40 million base pairs (slightly less than 30% of the chromosome) were included, since the time for a complete run did not suffice. All positions on the chromosomes with a read coverage of at least 2 on one haplotype, and a coverage of 3 on the other haplotype were examined. The maximum read coverage for any given position, which is identical with the read coverage used by WhatsHap for the phasing, was 5.



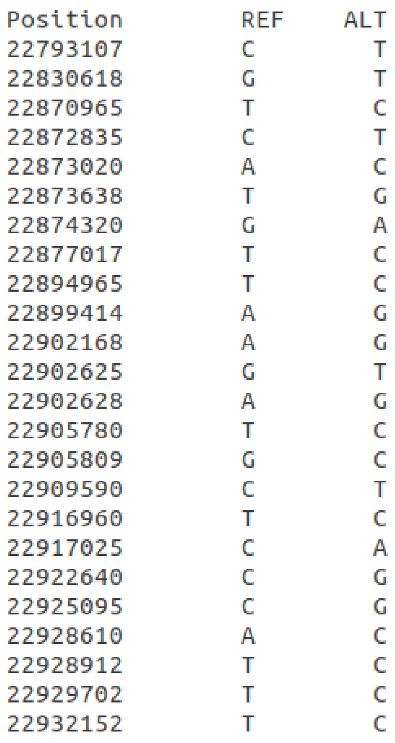
```
Phasing statistics for sample HG002 from file HG00211triophased.vcf
---------------- Chromosome 11 ----------------
     Variants in VCF:   246465
        Heterozygous:   100457 (   90168 SNVs)
              Phased:    88206 (   88206 SNVs)
            Unphased:    12251 (not considered below)
          Singletons:        0 (not considered below)
              Blocks:        2

Block sizes (no. of variants)
   Median block size:    44103.00 variants
  Average block size:    44103.00 variants
       Largest block:    88203    variants
      Smallest block:        3    variants

Block lengths (basepairs)
      Sum of lengths: 134769529    bp
 Median block length: 67384764.50 bp
Average block length: 67384764.50 bp
       Longest block: 134752669    bp
      Shortest block:    16860    bp
           Block N50:      nan    bp
```

Table 3: (referring to 5.2) *The results of the phasing for chromosome 11. While, compared to chromossome 22, more than three times as many variants have been phased, WhatsHap still assigned almost all of them to a single block (a second block consists of only three variants).*



GT \ chr	chrom 11	chrom 22
0/1, 1/1, 1/1	0	24
1/0, 0/0, 0/0	1959	652
1/1, M or F 0/0	3688	985
0/0, M or F 1/1	3489	971

Table 4: (referring to 5.3) *Frequency of different types of conflicts on the VCFs of the two chromosomes. M stands for mother, F for father.*



Position	REF	ALT
22793107	C	T
22830618	G	T
22870965	T	C
22872835	C	T
22873020	A	C
22873638	T	G
22874320	G	A
22877017	T	C
22894965	T	C
22899414	A	G
22902168	A	G
22902625	G	T
22902628	A	G
22905780	T	C
22905809	G	C
22909590	C	T
22916960	T	C
22917025	C	A
22922640	C	G
22925095	C	G
22928610	A	C
22928912	T	C
22929702	T	C
22932152	T	C

Table 5: (referring to 5.3) *All positions in chromosome 22 where the parents are homozygous with the alternative allele ("1/1"), while the child is heterozygous ("1/0"). Note that they all appear in short succession on the chromosome, the last one only about 13000bp after the first.*

For the two chromosomes phased, the method presented here required a large amount of runtime, with about 3:10 hours for a complete phasing of chromosome 22 with the method included, and slightly more than 5 hours for the first 40 million bp of chromosome 11 (the experiments were conducted on the HPC with a memory of 8 GB).

In the experiment, 81 mutation candidates were found on chromosome 22, and 64 were found on the examined part of chromosome 11. The frequency of mutations found by this method is therefore clearly lower than what we saw during the examination in 5.3, but still higher than expected from the frequency of germline de novo mutations in humans (p.17f. in [GWCD15], see also 4.1). Concerning the distribution of the candidate positions, it might be noted that on both chromosomes, some of them occur in very close proximity to each other. The most striking of those formations can be found on chromosome 22, where three mutation candidates are located on positions 24.000.528, 24.000.549 and 24.000.566, and four on positions 23.761.166, 23.761.171, 23.761.185, and 23.761.306. On chromosome 11, there is a "pair" of mutation candidates which are direct neighbours (17.841.561 and 17.841.562), and two other candidates which are only 114 bp apart from each other (23.210.691 and 23.210.805). The conflicts found by the described method could possibly result from errors in the reads or the phasing, but might also be intrinsic to the genome itself. It is beyond the scope of this thesis to interpret the patterns noticed in the genome data - yet, this kind of grouping is worth noting.

As a side remark, it should be mentioned that in the output of the program all positions differ from the positions in the VCF by the value of 1 - for example, 23.761.306 in the VCF is represented as 23.761.305 in the output. The reason for this is as yet unknown. Above, the positions are noted in the way they appear in the VCF - subtract one to get to the position found in the output.

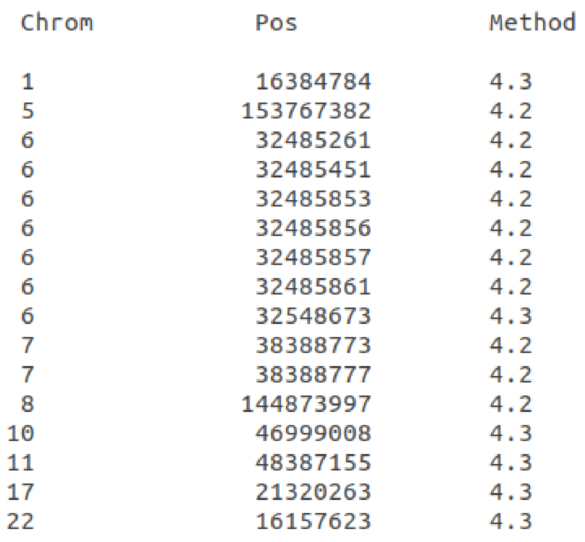
6 Results from the processing of UKD data

At a later stage in the construction of this thesis, trio data from *Universitätsklinikum* Düsseldorf (UKD) could be phased and processed. Due to the shorter amount of time available, the processing of the data is depicted here in a less detailed fashion compared to section 5. However, it is hoped that this synoptic description can still be uselful as an overview.

Out of the UKD data, 22 chromosomes from the trio which is referred to as "KB0001" in the dataset could be processed. The number of variants contained in the VCFs and the size of the BAM files (for each entire genome, there was only one BAM file of a size of about 50 GB) in the examined sample was small compared to the data described in section 5.1. Therefore, the time required for the phasing and the search for mutations could be greatly reduced, providing the opportunity to process a larger number of chromosomes. To give an example with regard to the sample size: the VCF for chromosome 1, the largest chromosome on the phased genome, contained 10.358 variants on the child, of which 4.782 were heterozygous, and of the heterozygous variants, 3.928 were phased. In comparison, chromosome 22 of the Ashkenazim child contained 24.328 phased variants and 61.984 variants in total. The results of the phasings were, as a rule, less exact than those obtained from the chromosome data described in section 5 (this is probably owed to the smaller number of reads in the BAM files and the greater distances between variants in the VCFs compared to the Ashkenazim trio data). On chromosome 1, the phasing resulted in 86 blocks of interconnected variants. Yet, it should also be mentioned that the largest of those blocks comprises alomost the complete part of the chromosome (248.381.326 out of 248.396.089 bp, or 3.716 out of the 3.928 phased variants). Therefore, in the case of chromosome 1, an altogether high degree of reliability can still be attributed to the trio phasing.

The two methods for detecting possible germline de novo mutations both succeeded in the discovery of candidate positions. With the first method, described in section 4.2, 10 positions where the child is "0/1" and both parents are "1/1" have been discovered on the whole genome (the positions containing "0/0" are being ignored here due to the problems described in section 5.3). With the second method (see section 4.3), six positions where the reads do not match the variants on the haplotypes have been discovered. Only positions with a coverage of at least 3 on one and at least 2 on the other haplotype were included. It should be noted that of the 16 candidates discovered in total, seven are located on chromosome 6 (of which six have been discovered with the first, and one with the second method). The six positions discovered there with the

first method are all in close proximity to each other, and even the position tagged by the second method is located at a distance of about 65.000 bp from the others, and thus in relatively close neighborhood.
For chromosome 1, the phasing was repeated with a maximum coverage of 7 per sample instead of WhatsHap's default maximum coverage of 5, in order to re-examine the chromosome with the second method. The findings of the second method were identical to those from the first run, on which one mutation candidate had been discovered. All 16 identified candidate positions, the chromosomes on which they were found, and the method with which they were detected (4.2 standing for the first, and 4.3 for the second method), are listed in Table 6 below.
Concerning runtime, it can be noted that on the sample from UKD, the second method worked much faster than on the chromosomes from the Ashkenazim trio. Even the larger chromosomes, such as chromosome 1, were processed within 15 minutes, while for some of the smaller chromosomes (such as 22), the processing only took approximately one minute (counting only the time required by the method, not the time required for solving PedMEC). This is due to the fact that the number of reads and the number of identified variants on the haplotypes, on which the method's runtime depends, were both much smaller.



Chrom	Pos	Method
1	16384784	4.3
5	153767382	4.2
6	32485261	4.2
6	32485451	4.2
6	32485853	4.2
6	32485856	4.2
6	32485857	4.2
6	32485861	4.2
6	32548673	4.3
7	38388773	4.2
7	38388777	4.2
8	144873997	4.2
10	46999008	4.3
11	48387155	4.3
17	21320263	4.3
22	16157623	4.3

Table 6: (referring to 6) *The positions on the child's genome (KB0001_c) which were identified as conflicting. To synchronise the data (see the last paragraph of section 5.4), 1 has been added to all positions discovered by the second method (4.3.), so they can be found in the VCF.*

7 Discussion

In the previous chapters, the possibilities for using trio phasing to detect germline de novo mutations on genome data have been explored. For this purpose, a trio phasing with the tool WhatsHap has been conducted on two chromomosomes, and two methods to detect possible mutations have been described and subsequently tested on the chromosome data. The genome data used can be accessed openly and is provided by GIAB [ZCM+16]. In addition, the presented methods have then been applied to data form the *Universitätsklinikum* Düsseldorf (UKD).

The phasings themselves can be regarded as successful for both chromosomes from the GIAB data, since the whole chromosomes could be phased in a short amount of time (22 minutes for chromosome 22, 87 minutes for chromosome 11), with all the phased variants having been assigned to a single, continuous block on chromosome 22, and to only two blocks in the case of the larger chromosome 11. Concerning runtime, the preprocessing of the BAM files (the adjustment of the read groups) took longer than the phasing itself (approximately one hour for chromosome 22, and three hours for chromosome 11). The merging of the VCF files required many different intermediate steps, yet, as part of the Snakemake pipeline, it could be done in a matter of seconds.

Coming to the first method for detecting mutation candidates, it has to be said that it produces a high number of false positives, which relate to positions where, in the original data, the genotype had been missing for at least one of the three individuals in the trio. In the preprocessing, the missing genotypes had been replaced by "0/0". Since, when this step is left out and the missing genotypes are not replaced, most positions in the VCF are not included in the phasing, this poses a problem.
Thus, with the first method, only conflict positions where the child is "0/1" and both parents are "1/1" can currently be reliably detected. Those kinds of candidate positions, however, do seem to yield some information value, as the findings on chromosome 22 suggest, where 24 of those positions have been found, all in close proximity to each other. Though it is beyond the scope of this paper to interpret what is the meaning of patterns on the genome, the finding appears worth noting.

The second method, in which the reads selected by WhatsHap were compared to the two haplotypes of the child to identify positions where the information given by the reads might be in conflict with the transmission vectors, was altogether more successful. A number of candidates were discovered on both chromosomes from the Ashkenazim child, and again, the feature that some of them appear in groups on the genome was noted. It must, however, be stated, that the number of conspicuous positions discovered on the genome is still higher than to be expected from the frequency of germline de novo mutations in humans, and that the method, as it is currently implemented, is slow and inefficient.

Compared to the findings on the Ashkenazim data, less mutation candidates (16 on 22 chromosomes in total, with 10 being found by the first and six by the second method) where detected on the UKD data. This is probably due to the smaller number of variants contained in the VCFs for the examined trio, and the smaller size of the read (BAM) files used.

With regards to future work, it might be a promising strategy to obtain the DP matrix with the columns of variants for each position on the haplotype directly from WhatsHap's core, which is written in C++, since this could possibly increase the speed of the processing. Concerning further methods for detecting mutations, a position-wise comparison between a trio phased VCF and single phased VCF on which the phasing has been conducted with a high maximum coverage might be another feasible possibility.

For positions on the trio phased VCF where the genotypes were missing in the original data, a new strategy to regain the genotype information is needed, since interpreting the missing positions as homozygous is not precise enough to allow for a promising search for mutations. Possibly, this is a field of application for WhatsHap's function for predicting genotype likelihoods.

8 Acknowledgements

I hereby would like to express my gratitude for my advisor Sven Schrinner and for Prof. Dr. Gunnar W. Klau for the opportunity to compose this thesis, and for the discussions which greatly enhanced my knowledge about the topics dealt with here, and helped introducing me to the methods of Bioinformatics. I would furthermore like to thank Philipp Spohr for the organisation of the regular meetings and presentations, in which important background information on the HPC and other tools was provided in a form that is accessible and easy to understand.

References

[DAA$^+$11] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A Albers, Eric Banks, et al. The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158, 2011.

[GMM16] Shilpa Garg, Marcel Martin, and Tobias Marschall. Read-based phasing of related individuals. *Bioinformatics*, 32(12):i234–i242, 2016.

[GWCD15] Anthony J F Griffiths, Susan R Wessler, Sean B Carroll, and John Doebley. *Introduction to Genetic Analysis (eleventh edition)*. W.H. Freeman and Company, 41 Madison Avenue, New York, NY 10010, 2015.

[KR12] Johannes Köster and Sven Rahmann. Snakemake — a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012.

[LHW$^+$09] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[Li11a] Heng Li. A statistical framework for snp calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27(21):2987–2993, 2011.

[Li11b] Heng Li. Tabix: fast retrieval of sequence features from generic tab-delimited files. *Bioinformatics*, 27(5):718–719, 2011.

[MBCT15] Veli Mäkinen, Djamal Belazzougui, Fabio Cunial, and Alexandru I. Tomescu. *Genome-Scale Algorithm Design.* Cambridge University Press, Cambridge, 2015.

[MPG$^+$16] Marcel Martin, Murray Patterson, Shilpa Garg, Sarah O Fischer, Nadia Pisanti, Gunnar W Klau, Alexander Schönhuth, and Tobias Marschall. Whatshap: fast and accurate read-based phasing. *bioRxiv*, 085050, 2016.

[PMP$^+$15] Murray Patterson, Tobias Marschall, Nadia Pisanti, Leo van Iersel, Leen Stougie, Gunnar W Klau, and Alexander Schönhuth. Whatshap: Weighted haplotype assembly for future-generation sequencing reads. *Journal of Computational Biology*, 22(6):498–509, 2015.

[ZCM$^+$16] Justin Zook, David Catoe, Jennifer McDaniel, Lindsay Vang, Noah Spies, et al. Extensive sequencing of seven human genomes to characterize benchmark reference materials. *Scientific Data*, 3(160025), 2016.

List of Figures

List of Tables