Charles University

Faculty of Science

Study programme: Bioinformatics

Branch of study: Bioinformatics

Gesa-Maret Neitzert

Ambiguous representation of genetic variants in the VCF format

Ambivalentní reprezentace genetických variant ve formátu VCF

Bachelor´s thesis

Supervisor: Mgr. Petr Daněček, Ph.D.

Prague, 2020

I would like to thank my supervisor, Mgr. Petr Daneček Ph.D., for his advice and patience, during the process of writing this thesis. Additionally, I would like to thank my family and friends for their patience and especially my partner, Radovan Vacek, for his continuous emotional support.

Prohlášení

Prohlašuji, že jsem závěrečnou práci zpracovala samostatně a že jsem uvedla všechny použité informační zdroje a literaturu. Tato práce ani její podstatná část nebyla předložena k získání jiného nebo stejného akademického titulu.

V Praze, 7.6.2020 Gesa-Maret Neitzert

Abstract

The variant call format (VCF) is a file format used to represent and store information about DNA variation. Genetic variants in VCF can be represented in multiple ways because the VCF specification allows for ambiguity, which can arise because of different variant calling pipelines or differences in sequence alignment. Ambiguities interfere with the comparison of VCF files and the variants therein, leading to complications in further analysis of variants.

This thesis explores the differences in the representation of genetic variants that can occur, as well as their causes and impacts on further analysis. Furthermore, the normalization of VCF files is addressed and an algorithm for the atomization and deatomization of VCF files is shown.

Keywords: VCF, variant call format, ambiguous variant representation, variant comparison, variant atomization, variant deatomization

Abstrakt

Variant call format (VCF) je formát souborů používaný k reprezentaci a ukládání informací o variantách. Genetické varianty ve VCF mohou být reprezentovány více způsobů, protože specifikace VCF umožňuje nejednoznačnost, která může nastat kvůli různým variant call pipelinům nebo rozdílům v alignmentech sekvencí. Nejednoznačnosti narušují srovnávání souborů ve VCF a jejich variant, což vede ke komplikacím při další analýze variant.

Tato práce zkoumá rozdíly v reprezentaci genetických variant, které se mohou vyskytnout, a také jejich pravděpodobné příčiny a dopady na další analýzu. Dále je zkoumána normalizace souborů VCF a je uveden algoritmus pro atomizaci a deatomizaci souborů VCF.

Klíčová slova: VCF, variant call format, ambivalentní reprezentace variant, srovnání variantů, atomizace variantů, deatomizace variantů

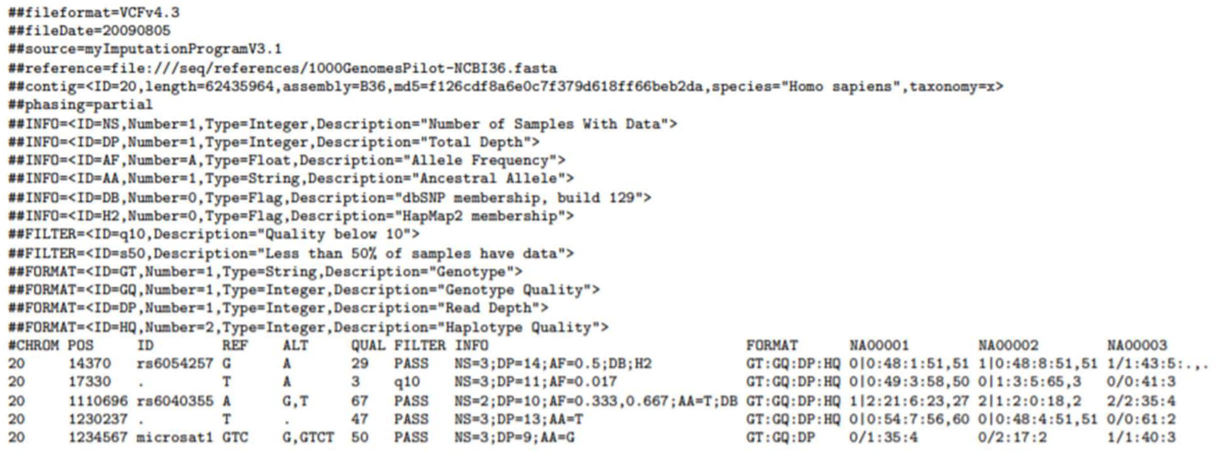
Table of Contents

Introduction

The variant call format, or VCF, is a text file format that allows for storing variations in DNA as well as annotating these variations (Danecek et al., 2011). Its specification is maintained by the File Formats group of the GA4GH consortium and available at http://samtools.github.io/hts-specs/. Files start with meta-information and a tab-delimited header, specifying the columns for the tab-delimited data lines that follow (Figure 1). The mandatory columns are:

- chromosome name, CHROM
- position, POS
- identifier, ID
- reference base(s), REF
- alternate base(s), ALT
- quality, QUAL
- filter, FILTER
- and additional information, INFO

These can be followed by the format (FORMAT) and an arbitrary number of sample IDs, if genotype data is present. VCF files are sorted by CHROM and the POS field in ascending order. VCF can represent all types of variation, SNVs, etc. (Figure 2), including structural variations. This thesis will focus on the representation of genetic variants such as SNVs (single nucleotide variants), indels (short insertions and deletions), or MNVs (multiple nucleotide variants) and the obstacles in processing these variations.



```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS     ID        REF    ALT     QUAL FILTER INFO                              FORMAT      NA00001        NA00002        NA00003
20     14370   rs6054257 G      A       29   PASS   NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,.
20     17330   .         T      A       3    q10    NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3   0/0:41:3
20     1110696 rs6040355 A      G,T     67   PASS   NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2   2/2:35:4
20     1230237 .         T      .       47   PASS   NS=3;DP=13;AA=T                   GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20     1234567 microsat1 GTC    G,GTCT  50   PASS   NS=3;DP=9;AA=G                    GT:GQ:DP    0/1:35:4       0/2:17:2       1/1:40:3
```

Figure 1 VCF file example from specification with SNVs, multi-allelic variants, missing alternative alleles and multi-allelic representation of insertion and deletion (microsat1). First two samples are majorly phased, last sample entirely unphased.

VCF files are commonly used and over the years many tools have been developed to visualize and analyze these files, such as VCFtools (Danecek et al., 2011), SMASH (Talwalkar et al., 2014), BrowseVCF (Salatino & Ramraj, 2016), VCF/Plotein (Ossio et al., 2019), VCF.Filter (Müller et al., 2017), CircosVCF (Drori et al., 2017), VCF-Miner (Hart et al., 2015), Vcfanno (Pedersen et al., 2016), Variant Tool Chest (Ebbert et al., 2014), VarMatch (Sun & Medvedev, 2016), and vt (Tan et al., 2015). Most of these tools require the input VCF files to simply follow the specification, occasionally recommending to validate the file (Hart et al., 2015) or even validating files automatically upon loading (Ossio et al., 2019). Several have further conditions, such as requesting the file to be cleaned and normalized as a standard preprocessing step (Talwalkar et al., 2014), recommending only using files with a single allele per row (Hart et al., 2015) or sometimes the opposite, requiring only biallelic sites.

This is to allow simplifying assumptions and to avoid difficulties arising from handling sites with multiple alternate alleles or ambiguous representation of variants (Talwalkar et al., 2014). Normalization is meant to eliminate this ambiguity to some extent and adapt the file according to best practices for the given task, such as left shifting variants. Actual best practices, however, are not generally agreed upon and the most convenient and practical form depends on the type of the analysis. To start with, different pipelines used in next-generation sequencing do not produce comparable results (Nekrutenko & Taylor, 2012).

Matching variants in databases is one example of analysis, which requires the ability to compare VCF files that are formatted differently. This is not a trivial task, because equivalent genetic variants can be represented differently across multiple VCF files. Sequence alignment can vary and the representation of variants is not standardized (Bayat et al., 2016; Tan et al., 2015). This ambiguity in the representation of variants affects further analysis and makes the comparison of VCF files a complex problem (Tan et al., 2015). Furthermore, variations in representation, especially indels in highly variable regions, that depend on the number of samples included in the VCF file, cannot be standardized. Variant records become more complex with more samples.

Another issue affecting simpler comparison methods that is not commonly addressed by normalization algorithms are multi-allelic lines. While data lines are sorted by chromosome number and location, variants that occur in the same position can be represented either in one data line with comma-separated alternate bases, or each on their own data line. For example, this can cause MNVs to appear in a single record instead of several lines of SNVs. Separating alternate bases from each other including splitting these MNVs into SNVs is a process referred to as atomization (coined by *bgt atomize*, Li, 2015) or sometimes simply splitting (*vt split*, Tan et al., 2015). Its counterpart, deatomization, allows merging several data lines into one if their locations overlap, which can provide an alternative representation and perspective, and depending on the goal of the analysis be more convenient. While algorithms for atomization and deatomization exist, they are not commonly included in distributed software for VCF file analysis (Hart et al., 2015).

This thesis explores the causes of ambiguities in VCF files and the impact they can have on the further analysis of VCF files. Finally, I will describe an algorithm that allows both for the atomization and deatomization of alternative alleles in a VCF file.

Main

About variants

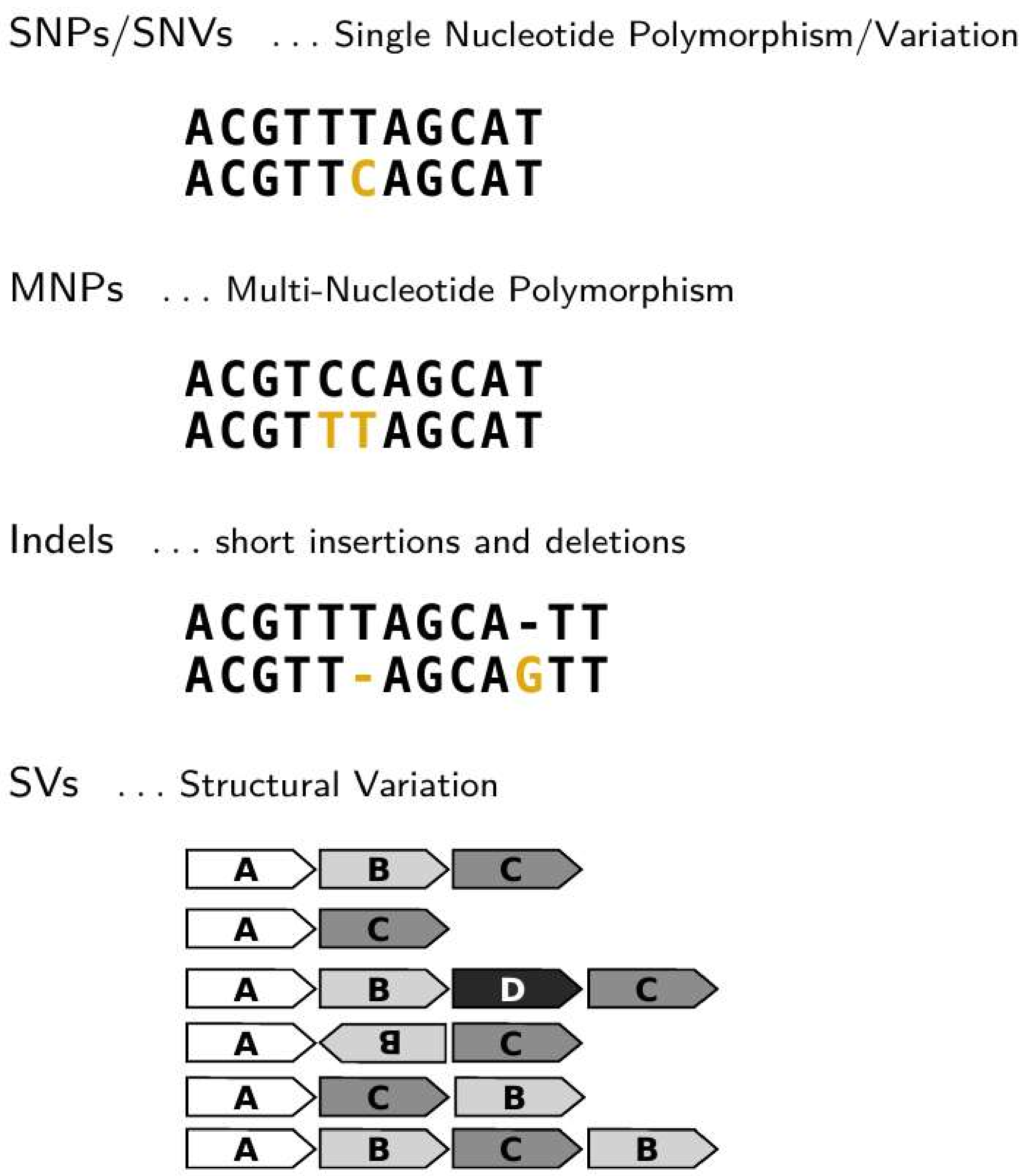

SNPs/SNVs ... Single Nucleotide Polymorphism/Variation
ACGTTTAGCAT
ACGTTCAGCAT
MNPs ... Multi-Nucleotide Polymorphism
ACGTCCAGCAT
ACGTTTAGCAT
Indels ... short insertions and deletions
ACGTTTAGCA-TT
ACGTT-AGCAGTT
SVs ... Structural Variation
A B C
A C
A B D C
A B C
A C B
A B C B

Figure 2 Illustration of different variants, image taken with permission from lecture slides.

SNV and MNV

Single nucleotide variants and multi nucleotide variants are substitutions of a single nucleotide or a few nucleotides in proximity. If these variants are common, they are referred to as single or multi nucleotide polymorphisms. The example below shows an SNV, where the guanosine at position 101 of chromosome 11 was substituted by a cytosine, and an MNV at the position 105, where a guanosine and following cytosine were substituted by adenosine and thymine, presented in VCF without sample information.

```
#CHROM  POS     ID REF     ALT     QUAL    FILTER INFO
11      101     .  G       C       199     PASS   .
11      105     .  GC      AT      199     PASS   .
```

Figure 3 Example of an SNV and an MNV in VCF.

Indels

Insertions and deletions are variations in genetic material that either introduce additional nucleotides or remove nucleotides. Indels are short insertions and deletions. Their defined size limits tend to depend on experimental methods rather than biology. The example below portrays a deletion of the sequence CGT following a guanosine at position 101 and an insertion of two thymines following a guanosine at position 105.

```
#CHROM  POS     ID REF     ALT     QUAL    FILTER INFO
11      101     .  GCGT    G       199     PASS   .
11      105     .  G       GTT     199     PASS   .
```

Figure 4 Example of two indels in VCF.

Multi-allelic variants

Variants are not always shown on separate lines. If two or more variants affect the same position, they can be joined into one line. This could be for example several SNVs in the same location or a deletion that deletes a nucleotide where an SNV could occur. Generally multi-allelic variants can be found in VCF files with genotype data for more than one sample sequence. The examples below show the representation of a deletion of the sequence CGT at the position 101 and an SNV at the position 102, first on separate lines (atomized), then combined (deatomized) into a multi-allelic variant line, as well as two different SNVs at the position 101, again first atomized, then deatomized.

```
#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO
11      101    .  GCGT   G      199    PASS   .
11      102    .  A      T      199    PASS   .

#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO
11      101    .  GCGT   G,GAGT 199    PASS   .

#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO
11      101    .  G      C      199    PASS   .
11      101    .  G      T      199    PASS   .

#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO
11      101    .  G      C,T    199    PASS   .
```

Figure 5 An indel and SNV atomized, as multi-allelic variants (deamotized), two SNVs atomized, and the same SNVs deatomized.

Ambiguous alignment

Aside from the ambiguity of atomized and deatomized representations of variants shown above, further ambiguities, such as the one shown below, can occur. Here the mutations from the sequence GCAT to the sequence GATT can be described either directly as such, as the deletion of the cytosine and insertion of another thymine, as two SNVs, or as one MNV.

Reference sequence	G C A T
Sample sequence	G A T T
Alignment 1 (Indels)	G C A T – G – A T **T**
Alignment 2 (SNVs or MNV)	G C A T G **A** **T** T

Table 1 Different alignments causing different the different VCF records below.

```
#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO
11      1      .  GCAT   GATT   199    PASS   .

#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO
11      1      .  GC     G      199    PASS   .
11      4      .  T      TT     199    PASS   .

#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO
11      2      .  C      A      199    PASS   .
11      3      .  A      T      199    PASS   .

#CHROM  POS    ID REF    ALT    QUAL   FILTER INFO
11      2      .  CA     AT     199    PASS   .
```

Figure 6 The same sequence represented in four different ways.

Phasing

VCF files are frequently used for representing human genome information, which means they often contain diploid samples. This data is commonly unphased due to the sequencing methods used, but some analyses require phased data (Browning & Browning, 2011). To determine haplotypes from unphased genotypes, phasing methods have been developed, which either determine the haplotype through further laboratory-based work, by genotyping additional family members, or by inferring the phase statistically by using computational methods.

Phased genotypes give additional information about the origin of the genotypes and thus variations therein. Autosomal chromosomes in offspring consist of one paternal and one maternal copy, which can result in four possible genotypes (Figure 7).

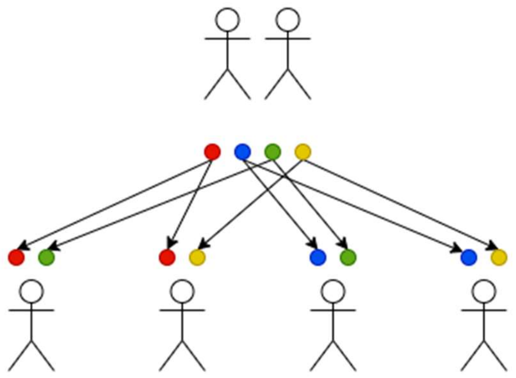


Figure 7 Four possible inheritance options.

Thus each allele in the genome originates from either a paternal or a maternal genetic sequence and furthermore is often inherited within a haplotype, meaning several alleles grouped into a sequence originating from the same chromosome. A phased variant sample is a sample where the haplotype for the alleles is known, giving information not just about the presence of a specific allele of a certain variant, but also about adjacent alleles that stem from the same haplotype. Knowing the haplotype is important for example when inferring disease status, assessing patterns of inheritance, or even filling in gaps, where genotype information is missing (Rao et al., 2013).

POS	REF	ALT	Sample
1	A	C	0/1
2	A	C	0/1
Reference sequence:	A A		

	Maternal	Paternal
Possible haplotypes:	A A	C C

	C C	A A
	C A	A C
	A C	C A

Table 2 Various possible haplotypes for unphased VCF sample. Statistical information could allow phasing of the sample, adding to the information that can be drawn from this sample.

The computational phasing of unphased genotypes can be achieved in several ways, but most approaches rely on reference panels of haplotypes from a large number of unrelated samples (McCarthy et al., 2016). The phasing methods usually use statistical approaches such as hidden Markov models to determine haplotypes (Browning & Browning, 2011; Loh et al., 2016). The main limitation of statistical phasing methods is their computational complexity (Tewhey et al., 2011), since the accuracy of phasing tends to be higher with larger sample sizes (Loh et al., 2016). Phasing is a complex problem, and while important for the analysis of genetic variants, it falls beyond the scope of this thesis.

Causes for ambiguity

The VCF generally contains one line per variant, consisting of one REF allele and zero or more alternative alleles. If there is one alternative allele in the ALT column, it is referred to as being in biallelic form. This flexibility allows for several alternative alleles on one line, which more commonly occurs with increased sample size. These lines are referred to as multi allelic and could occur for example when a sample genotype is heterozygous for two alternative alleles. Multi allelic variants can be found for example the publicly available VCF files generated by the 1000 Genomes Project (Auton et al., 2015). The VCF ALT field contains a list of non-reference alleles separated by commas, if several alternative alleles were called for this position. Which samples contain which alleles is then expressed in the genotype for the samples. For the genotype field in the samples of a VCF file ‘0’ is used to denote the reference allele and alternative alleles are numbered from 1.

```
#CHROM  POS     ID REF    ALT    QUAL   FILTER INFO   FORMAT S1     S2
11      101     .  G      C,T    199    PASS   .      GT     0|1    2|1
```

Figure 8 Diploid sample S1 has the reference allele G for one and alternative allele C for the other allele, S2 has the alternative alleles T for the first and C for the second allele.

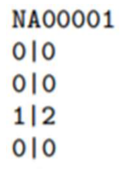
If the genotype is unphased its values are separated by ‘/’, while phased genotypes are separated by ‘|’, meaning the haplotype is known.

```
#CHROM   POS     ID REF     ALT     QUAL    FILTER INFO    FORMAT  S1      S2
11       101     .  G       C       199     PASS   .       GT      0|1     0/1
11       102     .  A       G,T     199     PASS   .       GT      1|2     1/2

Haplotypes S1: GG and CT
Haplotypes S2: GG and CT or GT and CA
```

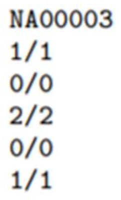
Figure 9 Example of effect of phasing on possible haplotypes.

In the case of unphased genotypes, there is no specific order the allele values need to follow, and genotypes can represent haploid, diploid, triploid and higher ploidy calls. Missing genotypes (and missing values in VCF in general) are represented by '.'.



```
NA00001
0|0
0|0
1|2
0|0
```

Figure 10 Phased samples are denoted by '|', from specification example VCF file.



```
NA00003
1/1
0/0
2/2
0/0
1/1
```

Figure 11 Unphased samples are denoted by '/', from specification example VCF file.

POS	REF	ALT	Unphased	Phased
1	A	T	0/1	0|1
2	G	GG	0/1	0|1
Possible sequences:	Sequence A		Sequence B	
Phased haplotypes	AG		TGG	
The four possible unphased haplotypes	AG		TGG	
	TG		AGG	
	AGG		TG	
	TGG		AG	

Table 3 Example of unphased and phased sample and their respective possible haplotype sequences.

This means the VCF is flexible and VCF files can appear vastly different, depending on which algorithm and parameters were used to obtain it (Bayat et al., 2016). Primary data and software version and parameters should be stated, because they all affect the output, leading to complication when comparing VCF files (Nekrutenko & Taylor, 2012). Sandmann et al. (2017)

found that some tools prefer calling MNVs in places where several mutations occur in proximity, whereas other tools prefer to report them separately. Furthermore, the calling of indels also varies greatly between different tools and depends on the number of samples present. Single- or multi-threading sample variant calling also seems to affect how variants are called (Sandmann et al., 2017). One cause for different variant representation in VCF is this flexibility of the VCF, because it allows for ambiguities such as expressing the same variants as several SNVs or one MNV (Danecek et al., 2011). Further ambiguities can be due to indels within repetitive sequences or homopolymers, where alignment can place the indel anywhere in the repeat without changing the underlying biological sequence:

```
REF       ACACACAC
ALT1      AC--ACAC
ALT2      ACAC--AC
```

or generally the combinations of simpler variants into more complex (Cleary et al., 2015). This ambiguity leads to difficulties when comparing VCF files.

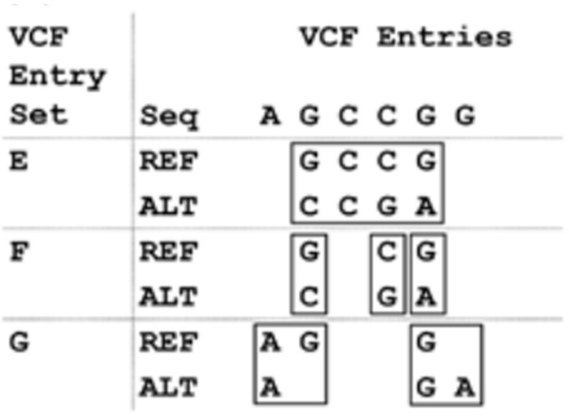
Indels			SNVs			MNV		
POS	REF	ALT	POS	REF	ALT	POS	REF	ALT
1	GC	C	2	C	A	2	CA	AT
4	T	TT	3	A	T			
Reference				GCAT				
Alternative				GATT				

Table 4 Example of VCF ambiguity: Three different representations of the same sequence.

Implicit equivalencies, variants that produce the same sequence when applied to the reference sequence but are not represented the same way, among different VCF files are common (Bayat et al., 2016) and while cheaper next-generation sequencing methods produce vast amounts of raw data waiting to be analyzed, this often requires specialists (Mardis, 2010). Over the past decade many tools were developed to improve the analysis and processing of data (Talwalkar et al., 2014), many of them rely on the command-line and a few provide a graphical user interface.

The impact of ambiguity

The comparison of variants in VCF files has remained an issue, because of the different possibilities to represent the same variant (Tan et al., 2015). Comparators would need the ability to rebuild the alternative sequences for the respective samples, since a direct variant to variant comparison can be close to impossible. This ambiguity also causes redundancies in variant archives and databases (Watkins et al., 2019). The overlap between variants from different databases increases when normalizing the variants prior to comparison (Tan et al., 2015), which reveals the necessity of normalization not just to allow for comparisons during the analysis of data, but also to prevent redundancies. Normalization, however, is not sufficient as the example in Figure 12 (Sun & Medvedev, 2016) shows. Thus, the comparison of the sequences that result from applying variants to the reference genome is the best way to compare VCF files.



VCF Entry Set		VCF Entries					
	Seq	A	G	C	C	G	G
E	REF		G	C	C	G	
	ALT		C	C	G	A	
F	REF		G		C	G	
	ALT		C		G	A	
G	REF	A	G			G	
	ALT	A				G	A

Figure 12 Three normalized variants representing the same alternative sequence ACCGAG, Sun & Medvedev (2016).

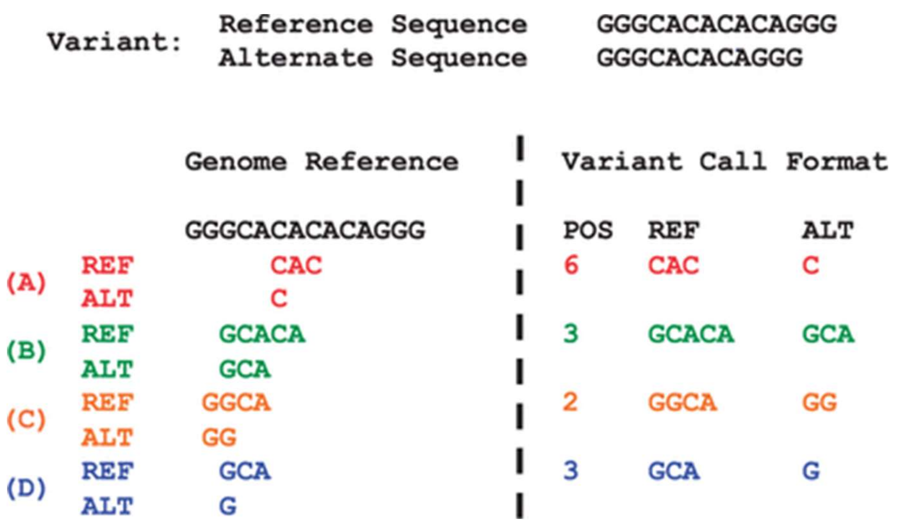
One example of where the comparison of VCF files is also essential is the development and validation of new pipelines and algorithms, because it enables benchmarking of these new methods (Cleary et al., 2015). Generally comparison allows to measure similarity to popular variants, and compare variants to those in databases, as well as facilitating the evaluation of the accuracy of tools (Sun & Medvedev, 2016) to mention a few applications.

The analysis and processing of data can limit research in genome sequencing (Talwalkar et al., 2014), even after the cost of sequencing was reduced. The inconsistencies in VCF files also affect the reproducibility of research (Nekrutenko & Taylor, 2012). Consistent representation of variants is a key aspect of further analysis, influencing quality, integration and functional interpretation of the data available (Tan et al., 2015). It could add to the quality of data stored in archives and thus provide a baseline for further research in genomics, but using various tools to call the variants for the same sequence will lead to varying VCF files and sometimes even reproducing the same VCF with the same tool is not possible (Sandmann et al., 2017).

The analysis of variant call data from VCF files can be limited further by the requirements some tools set for their VCF input files. Tools such as the VCF-Miner can implicitly include atomized alternative alleles in their requirements and request multi-allelic files to be converted into biallelic (Hart et al., 2015), even though atomization algorithms are usually not provided within the tool and if they are, then usually with limitations. In the case of the VCF-Miner, samples are stripped of '.', '/', '|', and '0' to determine homo- or heterozygosity, which in multi-allelic VCF files can lead to inaccuracies, because it does not distinguish between different numbers, but rather counts the leftover number of values leading to heterozygotes being dismissed as homozygous. This eliminates potentially essential information about the genotype from further analysis. Generally samples containing several different alternative alleles are frequently considered as homozygotes (Ebbert et al., 2014; Hart et al., 2015). Methods of the Variant Tool Chest for the intersection of multi-allelic data lines consider the intersection as true if at least one of the alternative alleles matches (Ebbert et al., 2014). This could lead to inaccurate results and shows that the analysis involving these and similar tools can be finetuned by the atomization of VCF files.

VCF normalization

Generally variants can be considered equivalent if they match in position, reference and alternative bases (Pedersen et al., 2016), which is a complex task, because equivalent variants could be represented differently, like for example two SNVs as one MNV or two SNVs as two indels as shown in Figure 6. Therefore it is more reasonable to consider VCF files as equivalent if their sample sequences are the same, even if the actual data lines differ (Bayat et al., 2016; Sun & Medvedev, 2016). Having a more standardized representation of variants could ease the comparison of two VCF files, because comparators frequently recognize equivalencies among uniformly represented variants (Bayat et al., 2016). This is what normalization methods try to address.



Variant:
Reference Sequence GGGCACACACAGGG
Alternate Sequence GGGCACACAGGG
Genome Reference
Variant Call Format
GGGCACACACAGGG
POS REF ALT
(A) REF CAC ALT C | 6 CAC C
(B) REF GCACA ALT GCA | 3 GCACA GCA
(C) REF GGCA ALT GG | 2 GGCA GG
(D) REF GCA ALT G | 3 GCA G

Figure 13 A: Not left-aligned; B: Neither left-aligned, nor parsimonious; C: Left-aligned, not parsimonious; D: Left-aligned and parsimonious (Tan et al., 2015).

The recommended representation is one reference base for SNVs and insertions, and one alternate base for deletions, referring to the lowest position when there is any ambiguity (Danecek et al., 2011). This is described as left shifting in normalization methods (Talwalkar et al., 2014). Normalization methods define a variant as normalized if it is left-aligned and parsimonious, which is able to represent biallelic variants unambiguously (Tan et al., 2015). Because most variants are biallelic, this normalization technique already helps to filter out redundancies and makes the processing of raw sequencing data more efficient (Talwalkar et al., 2014). This does not have to affect the representation of multi-allelic variants, in fact *vt normalize*, a tool for the normalization of the representation of genetic variants in VCF (Tan et al., 2015), takes both bi-allelic and multi-allelic variants as input, which adapts well to the sometimes unavoidable ambiguity of VCF files. The ability to process all representations of variants is not the rule for all tools available though, like for example VCF-Miner (Hart et al., 2015) described above.

Normalization methods such as the Best Alignment Normalization (BAN) (Bayat et al., 2016) can adapt to multi allelic variants. BAN for example introduces an algorithm that splits diploid samples into haploid prior to normalization, because part of the normalization method does not support heterozygous samples. One VCF file is represented by one sequence during the normalization, and since heterozygotes have two different alleles, they require the file to be split into two sequences. For heterozygous samples in multi-allelic variants, where two different alternative alleles are present in the same sample, this means that they are essentially atomized into two separate variant records:

POS	REF	ALT	Genotype	POS	REF	ALT	Genotype
1	T	C,G	1/2	1	T	C	0/1
				1	T	G	0/1

Table 5 Example of a heterozygous multi-allelic variant being split into two heterozygous variants as part of BAN's normalization method (Bayat et al., 2016).

During this the sample becomes a heterozygote for each alternative allele, but the other allele refers back to the reference (Bayat et al., 2016), as can be seen in Table 5. This process leads to samples containing reference alleles in loci that originally were all alternative variants. During further processing the alternative alleles are each assigned to separate sequences. If the sample is not phased, a random phase is assigned to temporarily phase the sample and standardize it enough for the following splitting of the diploid sample into haploid (Bayat et al., 2016). This can lead to further inconsistencies among VCF files, as Bayat et al. (2016) state that the randomness can cause inaccuracies.

A random assignment of phase as used in BAN (Bayat et al., 2016) can cause problems for example when a deletion in one position is immediately followed by a substitution in the next. Assigning both these alleles to the same haplotype should be avoided, because their overlap would pose an unfeasible situation where a nucleotide is simultaneously deleted and substituted. The random phase assignment step during BAN only guarantees the assignment of alleles from multi-allelic variants to opposite haplotypes (Bayat et al., 2016), but not their neighboring variants. Deatomization of a VCF file prior to applying the BAN method for normalization, could prevent errors caused during the random phase assignment of overlapping variants.

The best practice seems to be assigning as many variants to one haplotype as possible, which should be considered during the normalization of VCF files. If variants are then joined during deatomization, the minimum number of alternative alleles are created. With increasing numbers of samples, the minimization of alternative alleles per variant can become computationally expensive.

Atomization and deatomization

The ambiguity of VCF files lies not just in the representation of variants themselves, but also in the way variants are grouped together into multi-allelic data lines or not in a VCF file. Resolving this part of variations in representation is not commonly addressed by normalization algorithms. Some tools, such as vcflib (Garrison, 2012) or vt (Tan et al., 2015), include methods intended for atomization, but are limited either by not producing genotype allele values, producing them incorrectly, or not atomizing combinations of variants such as MNVs further. VCF-Miner, one of the tools requiring atomized alleles, does not provide an atomization tool, but offers a script for atomization upon request (Hart et al., 2015).

Deatomization on the other hand does not seem to be explicitly required for further analysis, but it can help in reducing ambiguity similarly to atomization. While tools to merge entire VCF files are available, such as bcftools merge (Li et al., 2009), tools and algorithms to deatomize variants in VCF files are not.

Implementation

Description of algorithms

The problem at hand can be seen as two inverse processes: atomization and deatomization. For both the input is a file in VCF, read line by line. The output for atomization is a VCF file where multi-allelic variants were split. The output for deatomization is a VCF file where variants that affect the same positions were combined into a multi-allelic variant.

```
Reference sequence:       GCGT (0)         GCGT (0)
S1:                       GCGT (0)         G--- (1)
S2:                       G--- (1)         GCGA (2)
S3:                       GCGA (2)         GTGA (3)
S4:                       GCGA (2)         CCGT (4)

#CHROM   POS    ID REF    ALT    QUAL   FILTER INFO   FORMAT S1     S2     S3     S4
12    101  .  GCGT   G,GCGA,GTGA,CCGT   199 PASS   .  GT    0|1    1|2    2|3    2|4

#CHROM   POS    ID REF    ALT    QUAL   FILTER INFO   FORMAT S1     S2     S3     S4
12      101     .  G      C      199    PASS   .      GT     0|.    .|0    0|0    0|1
12      101     .  GCGT   G      199    PASS   .      GT     0|1    1|.    .|.    .|.
12      102     .  C      T      199    PASS   .      GT     0|.    .|0    0|1    0|0
12      104     .  T      A      199    PASS   .      GT     0|.    .|1    1|1    1|0

#CHROM   POS    ID REF    ALT    QUAL   FILTER INFO   FORMAT S1     S2     S3     S4
12    101  .  GCGT   G,GCGA,GTGA,CCGT   0  .       .  GT    0|1    1|2    2|3    2|4
```

Figure 14 A possible input for atomization, a possible input for deatomization (which is also the output of the atomization of the previous), and a possible output for deatomization.

Atomization

```
WHILE not end of file:
   IF ALT contains comma:                            #this line is multi-allelic
      set alts = split ALT
      FOR alt in alts:
         #cut off unchanged nucleotides from alternative and reference alleles
         newALT = trim alt
         newREF = trim REF
         newPOS = position of new REF
         positions = positions affected by newALT    #used to update genotype
         IF genotype present:
            update genotype
         build new line                              #using newALT, newREF, newPOS, updated genotype
      print new lines
```

Figure 15 Atomization in pseudocode.

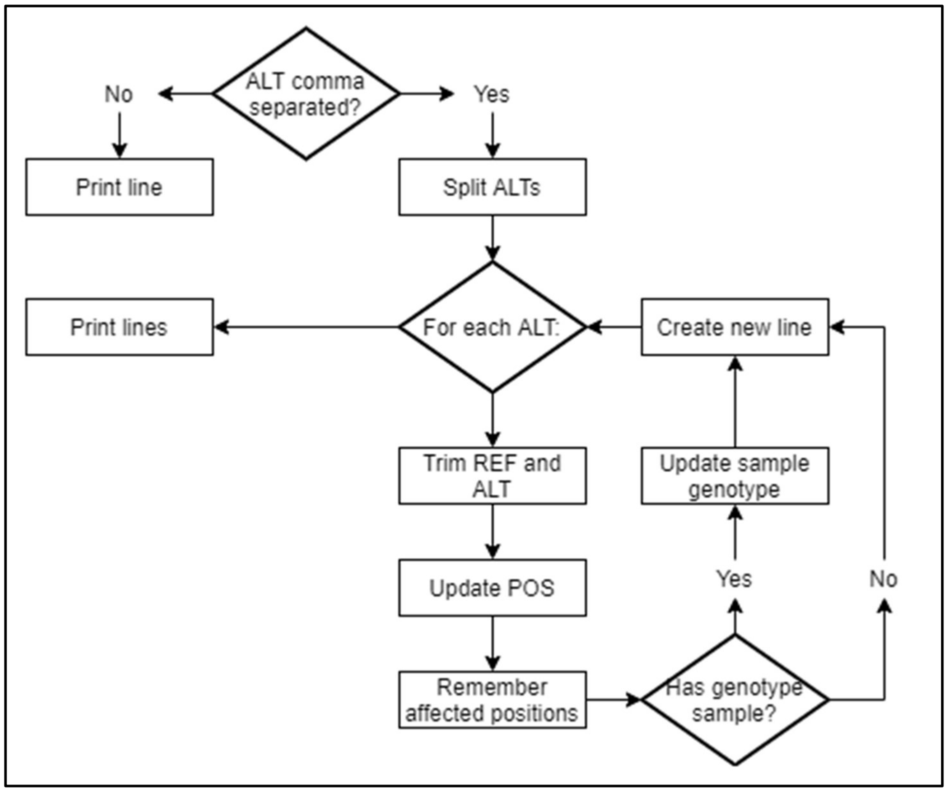


Figure 16 A flowchart of atomization.

Deatomization

```
IF this REF overlaps previous REF:
    store line
ELSE:
    IF lines are stored > 1:                        #deatomize stored lines
        newREF = join REFs
        IF genotype available:
            IF genotype unphased:
                right-swap samples                  #temporary phasing
            FOR each column in samples:
                base = newREF
                IF item in column not REF:
                    replace REF with ALT in base
                newALT = base
            update genotype
        ELSE:
            FOR each ALT in lines:
                IF next ALT not overlap this ALT:
                    newALT = this ALT + newREF[this ALT end, next ALT start] + next ALT
                ELSE:
                    extend newALT
                    start new newALT
        build new line          #using newALTs, newREF, smallest POS, updates genotype
        print new line
        clear stored lines
        store line
    ELSE:
        print stored line
        store line
```

Figure 17 Deatomization in pseudocode.

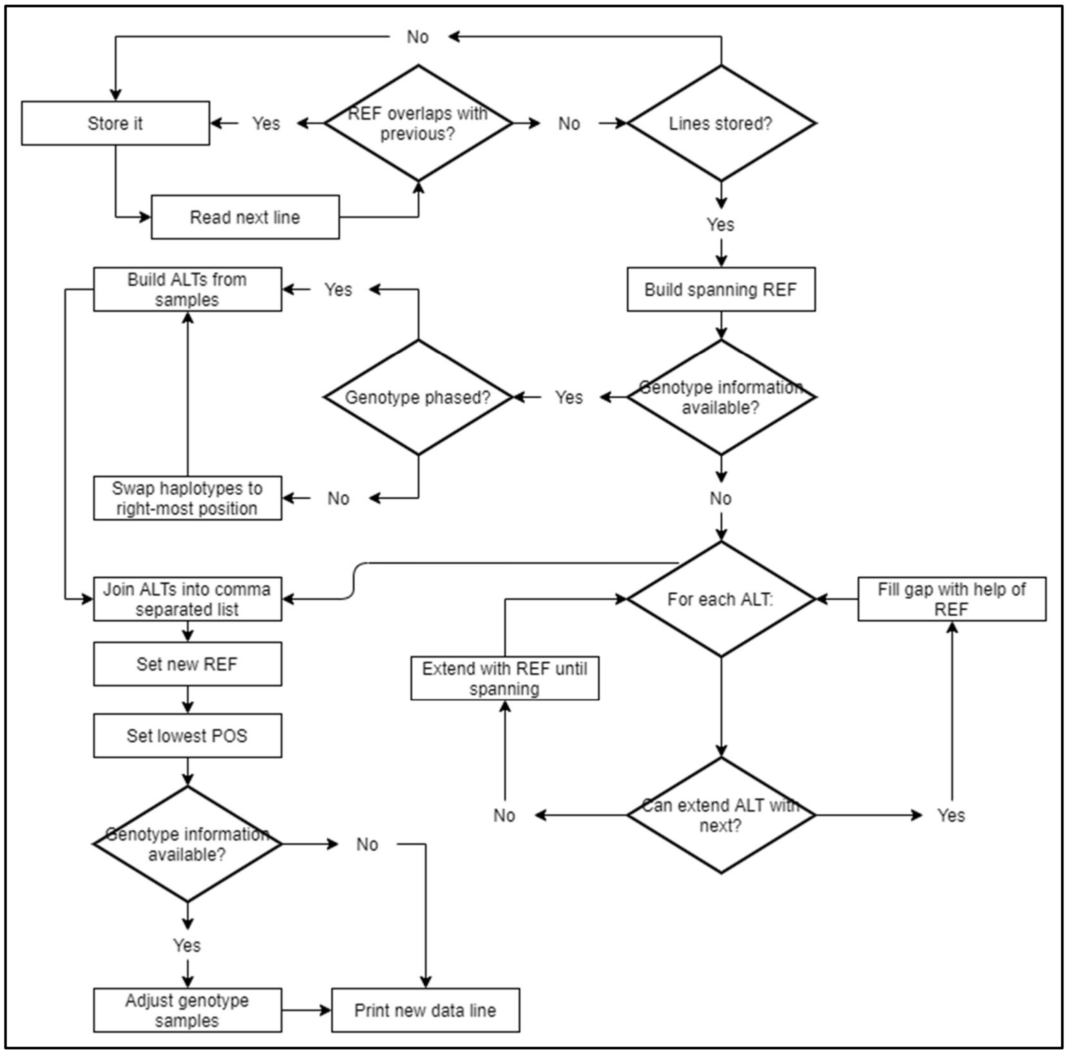

No
Store it
Yes
REF overlaps with previous?
No
Lines stored?
Read next line
Yes
Build ALTs from samples
Yes
Build spanning REF
Genotype phased?
Yes
Genotype information available?
Swap haplotypes to right-most position
No
No
Join ALTs into comma separated list
For each ALT:
Fill gap with help of REF
Set new REF
Extend with REF until spanning
Set lowest POS
No
Can extend ALT with next?
Yes
Genotype information available?
No
Yes
Adjust genotype samples
Print new data line

Figure 18 A flowchart for deatomization.

Implementation details

Both the atomization and deatomization start by copying the meta-information header and the header line containing column names. Then the input file is read line by line with help of the Java BufferedReader, which allows the reading from VCF files, which tend to be large files, to be more efficient. Should an output file be given, then the output is written with a PrintWriter wrapped around a BufferedWriter. This combination should give the formatting advantages of the PrintWriter while still buffering the output.

Data lines from the VCF file are represented by the class VCFDataRow, which has a field for every column of the type specified in the VCF specification and overwrites the toString() method to print a single line as a tab-delimited String. The algorithms implemented only address the atomization and deatomization of alternative alleles, which means that annotation information from columns such as INFO are dropped during deatomization and copied during atomization. Adjusting this additional information during deatomization was not within the scope of this thesis.

The program was written in Java 11 on Windows 10.

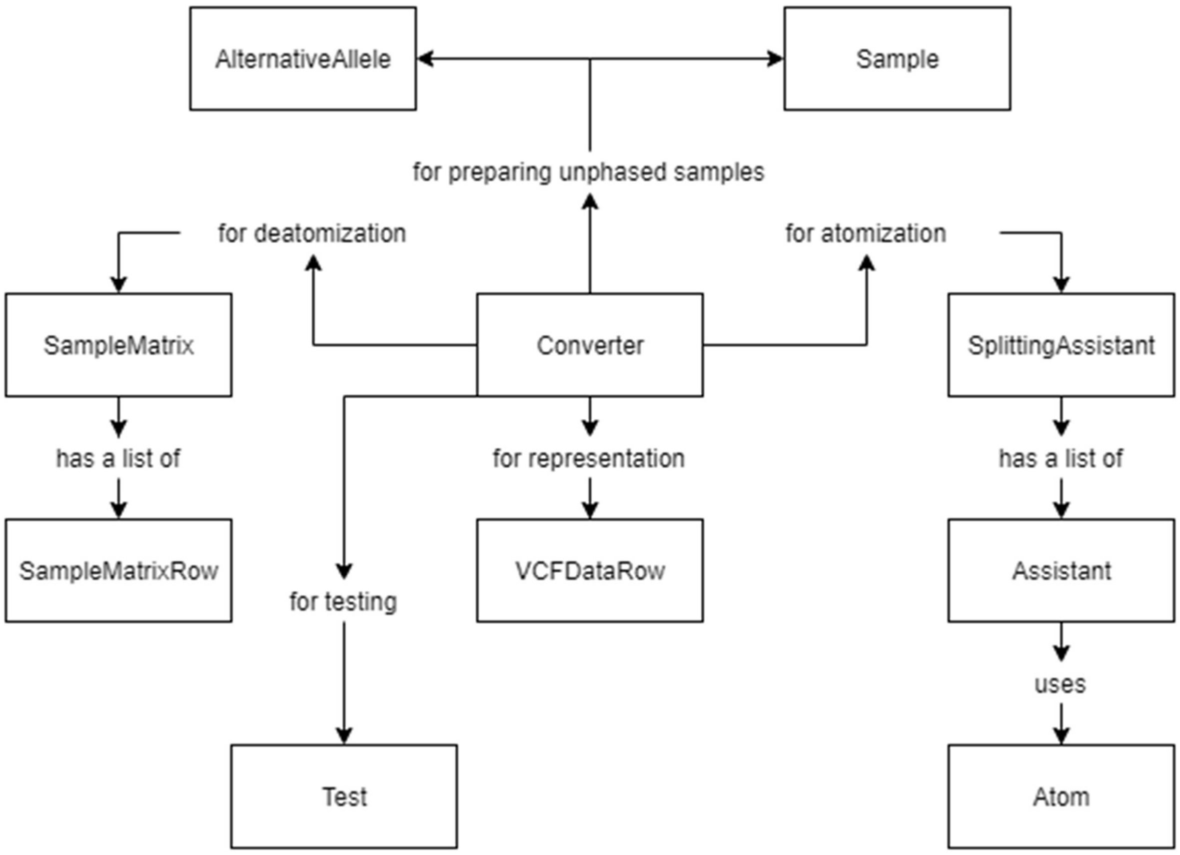


Figure 19 A simplified diagram to show class relations and purposes.

Atomization

Once a VCFDataRow object is created for the line, the ALT column is inspected for commas. If there are any present, this indicates that the line has more than one alternative allele. Else the line is written to the output unchanged.

A line with multiple alternative alleles is atomized into as many new lines as there are alternative alleles. For this the class Atom is used, which can store all the necessary information about the allele, such as the original form of the reference and alternative allele, and original position, as well as the new reference and alternative allele, which are trimmed from both ends for all characters they have in common, and the beginning and exclusive end position. Furthermore, it stores the allele's positional number, so that samples can be adjusted if present. Should an allele be an indel, it will not be trimmed, as this could require further computation for the best alignment. Should an allele, however, be an MNV, then the MNV will be split into SNVs, each of which will be treated as their own new data line for the output file.

If the input file contains samples with genotype information, these must be adjusted. For this the SplittingAssistant and Assistant classes are used. An Assistant object contains both the Atom it refers to, reference and alternative alleles, the position, and an array, representing a column for the sample after atomization. This array only contains '0', '1', or '.' as values and is inserted as the new sample values wherever the original data line was referring to this allele.

After the alternative alleles have been atomized, the new block of data lines is merged into one String, separating single lines with the new line character, and send to the output.

Deatomization

To find a block of lines to be deatomized, each line is inspected for an overlap with the position of the previous. Should two lines either have the same position, or one ends after the position of the other, these lines are stored as VCFDataRow objects in a list, referred to as the block for deatomization. Lines are added to this block as long as the overlap is not overcome. Should a line not share any position with any other line, it will be directly printed to the output. A block for deatomization is complete once no new lines can be added to the overlap. This relies on the VCF file being sorted by chromosome and position, as per specification.

For the new line of data, the smallest position is taken out of all lines in the block, which is the POS for the deatomized line. A new reference is constructed to span all position in the block, based on the reference alleles of each single line, rather than extracting the original reference genome, since the exact version of such genomes is often not listed (Nekrutenko & Taylor, 2012). The alternative alleles must span the entire reference, but if samples are present, they need to be constructed based on the samples. For this the classes SampleMatrix and SampleMatrixRow are available. They help split apart the genotype in the samples, from which the right alternative alleles are inserted into the newly constructed reference allele. This works by using the new reference as a base and replacing its substring with the alternative allele in the same indexes when the sample genotype is the alternative allele.

Should the genotypes not be phased, the class Sample is used to represent a single column of the unphased genotype from a sample. Rather than assigning phase randomly as done by BAN (Bayat et al., 2016), as many alternative alleles are pushed onto the same haplotype as possible within the Sample object, while keeping track of the consistency of the haplotype. This affects the resulting alleles, since it is not an actual phasing algorithm. Then the genotypes are temporarily represented as phased, to allow the algorithm to construct the alternative alleles.

If the file does not have genotype information available, the alternative alleles are constructed greedily. This means that alternative alleles are combined into one if feasible. Gaps are filled with information from the newly constructed reference allele. If the next alternative allele affects a position that already was covered by a previous alternative allele, then this alternative allele is appended with the substring of the reference allele until it spans the entire length if necessary, and a new alternative allele gets constructed for the following alternative alleles. This does not necessarily minimize the number of alternative alleles, and is heavily dependent on the order of the original alternative alleles, but it can construct MNVs, even with indels, and thus likely reduces the number of alternative alleles in the deatomized new data line.

Once alternative alleles are constructed based on sample information, the genotype information in the samples themselves are adjusted to fit the new alternative alleles. This is done by matching the columns in the samples to the alleles that were constructed from them and assigning the number of that allele to the new sample. Then the a new VCFDataRow object is created based on the information from the deatomization and send to the output.

Usage documentation

The program takes a command – ‘atomize’, ‘deatomize’ or ‘test’ – as argument, followed by an input file (with exception for the ‘test’ command) and an optional output file. Should no output file be given, then the output is written to the standard output. If the input file is missing for atomization or deatomization or the command does not match any of the three mentioned above, the program informs the user of this and ends.

Limitations

For simplicity the program relies only on the default installation of Java, but packages for reading and analyzing VCF files are available, such as htsjdk (https://github.com/samtools/htsjdk) which are for example used by VCF.Filter (Müller et al., 2017) and VTC (Ebbert et al., 2014). The idea is to be able to run it on any computer with Java installed, without requiring the download and installation of further dependencies.

Further limitations of the program are the use of a greedy algorithm instead of an actual phasing algorithm for the deatomization of VCF files containing unphased samples. This and the addition of algorithms resolving further annotation information could improve this program, but were not part of the task at hand. The current greedy algorithm is heavily dependent on the order of the alleles and could be further improved by sorting them in order to create fewer alternative alleles.

Conclusion

The specification of the VCF allows for various representations of the same sequence. These ambiguous representations can be due to differences in sequence alignment, different combinations of alternate alleles observed across samples, or differing methods of variant calling, and can take the form of an MNV instead of SNVs, indels instead of SNVs, multi-allelic lines instead of bi-allelic, and even more complex variations.

Ambiguity in VCF files poses a problem when analyzing them, since analysis often involves the comparison of VCF files or comparison to variants in databases and variant to variant comparisons are affected by different representations. This leads to comparison methods relying on the reconstruction of sequences, an algorithmically and computationally difficult task. Normalization methods help to standardize the representation of variants by for example left-shifting indels in repetitive regions (Talwalkar et al., 2014; Tan et al., 2015), but do not commonly address multi allelic representation of variants.

Analysis tools either adapt to ambiguities in VCF files or require preprocessing of input files. Adaptations to multi-allelic representations of variants include considering heterozygous alternative genotypes as homozygous or including all alternative alleles in multi-allelic lines into sets based on the presence of one of these alternative alleles. In both cases tools could produce more accurate results, if the VCF files were atomized prior to analysis. Preprocessing of input files tends to focus more on the normalization of variants, but occasionally requires atomization of multi-allelic lines as well, even though methods and algorithms for atomization are not regularly included.

The ambiguity of the representation of variants in VCF files will remain a problem, because differences in variant calling methods and the number of samples present can lead to differing representations that even normalization and atomization/deatomization cannot consolidate. Thus, the comparison of VCF files still requires the reconstruction of sequences.

Sources

Auton, A., Abecasis, G. R., Altshuler, D. M., Durbin, R. M., Abecasis, G. R., Bentley, D. R., Chakravarti, A., Clark, A. G., Donnelly, P., Eichler, E. E., Flicek, P., Gabriel, S. B., Gibbs, R. A., Green, E. D., Hurles, M. E., Knoppers, B. M., Korbel, J. O., Lander, E. S., Lee, C., … National Eye Institute, N. I. H. (2015). A global reference for human genetic variation. *Nature*, *526*(7571), 68–74. https://doi.org/10.1038/nature15393

Bayat, A., Gaëta, B., Ignjatovic, A., & Parameswaran, S. (2016). Improved VCF normalization for accurate VCF comparison. *Bioinformatics*, *33*(7), 964–970. https://doi.org/10.1093/bioinformatics/btw748

Browning, S. R., & Browning, B. L. (2011). Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, *12*(10), 703–714. https://doi.org/10.1038/nrg3054

Cleary, J. G., Braithwaite, R., Gaastra, K., Hilbush, B. S., Inglis, S., Irvine, S. A., Jackson, A., Littin, R., Rathod, M., Ware, D., Zook, J. M., Trigg, L., & De La Vega, F. M. (2015). Comparing Variant Call Files for Performance Benchmarking of Next-Generation Sequencing Variant Calling Pipelines. *BioRxiv*, 23754. https://doi.org/10.1101/023754

Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., McVean, G., Durbin, R., & Group, 1000 Genomes Project Analysis. (2011). The variant call format and VCFtools. *Bioinformatics*, *27*(15), 2156–2158. https://doi.org/10.1093/bioinformatics/btr330

Drori, E., Levy, D., Smirin-Yosef, P., Rahimi, O., & Salmon-Divon, M. (2017). CircosVCF: circos visualization of whole-genome sequence variations stored in VCF files. *Bioinformatics*, *33*(9), 1392–1393. https://doi.org/10.1093/bioinformatics/btw834

Ebbert, M. T. W., Wadsworth, M. E., Boehme, K. L., Hoyt, K. L., Sharp, A. R., O'Fallon, B. D., Kauwe, J. S. K., & Ridge, P. G. (2014). Variant Tool Chest: an improved tool to analyze and manipulate variant call format (VCF) files. *BMC Bioinformatics*, *15 Suppl 7*(Suppl 7), S12–S12. https://doi.org/10.1186/1471-2105-15-S7-S12

Garrison, E. (2012). Vcflib: A C++ library for parsing and manipulating VCF files. In *GitHub*.

Hart, S. N., Duffy, P., Quest, D. J., Hossain, A., Meiners, M. A., & Kocher, J.-P. (2015). VCF-Miner: GUI-based application for mining variants and annotations stored in VCF files. *Briefings in Bioinformatics*, *17*(2), 346–351. https://doi.org/10.1093/bib/bbv051

Li, H. (2015). BGT: efficient and flexible genotype query across many samples. *Bioinformatics*, *32*(4), 590–592. https://doi.org/10.1093/bioinformatics/btv613

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., & Subgroup, 1000 Genome Project Data Processing. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, *25*(16), 2078–2079. https://doi.org/10.1093/bioinformatics/btp352

Loh, P.-R., Palamara, P. F., & Price, A. L. (2016). Fast and accurate long-range phasing in a UK Biobank cohort. *Nature Genetics*, *48*(7), 811–816. https://doi.org/10.1038/ng.3571

Mardis, E. R. (2010). The $1,000 genome, the $100,000 analysis? *Genome Medicine*, *2*(11), 84. https://doi.org/10.1186/gm205

McCarthy, S., Das, S., Kretzschmar, W., Delaneau, O., Wood, A. R., Teumer, A., Kang, H. M., Fuchsberger, C., Danecek, P., Sharp, K., Luo, Y., Sidore, C., Kwong, A., Timpson, N., Koskinen, S., Vrieze, S., Scott, L. J., Zhang, H., Mahajan, A., … Consortium, H. R. (2016). A reference panel of 64,976 haplotypes for genotype imputation. *Nature Genetics*, *48*(10), 1279–1283. https://doi.org/10.1038/ng.3643

Müller, H., Jimenez-Heredia, R., Krolo, A., Hirschmugl, T., Dmytrus, J., Boztug, K., & Bock, C. (2017). VCF.Filter: interactive prioritization of disease-linked genetic variants from sequencing data. *Nucleic Acids Research*, *45*(W1), W567–W572. https://doi.org/10.1093/nar/gkx425

Nekrutenko, A., & Taylor, J. (2012). Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. *Nature Reviews Genetics*, *13*(9), 667–672. https://doi.org/10.1038/nrg3305

Ossio, R., Garcia-Salinas, O. I., Anaya-Mancilla, D. S., Garcia-Sotelo, J. S., Aguilar, L. A., Adams, D. J., & Robles-Espinoza, C. D. (2019). VCF/Plotein: visualization and prioritization of genomic variants from human exome sequencing projects. *Bioinformatics*, *35*(22), 4803–4805. https://doi.org/10.1093/bioinformatics/btz458

Pedersen, B. S., Layer, R. M., & Quinlan, A. R. (2016). Vcfanno: fast, flexible annotation of genetic variants. *Genome Biology*, *17*(1), 118. https://doi.org/10.1186/s13059-016-0973-5

Rao, W., Ma, Y., Ma, L., Zhao, J., Li, Q., Gu, W., Zhang, K., Bond, V. C., & Song, Q. (2013). High-resolution whole-genome haplotyping using limited seed data. *Nature Methods*, *10*(1), 6.

Salatino, S., & Ramraj, V. (2016). BrowseVCF: a web-based application and workflow to quickly prioritize disease-causative variants in VCF files. *Briefings in Bioinformatics*, *18*(5), 774–779. https://doi.org/10.1093/bib/bbw054

Sandmann, S., de Graaf, A. O., Karimi, M., van der Reijden, B. A., Hellström-Lindberg, E., Jansen, J. H., & Dugas, M. (2017). Evaluating Variant Calling Tools for Non-Matched Next-Generation Sequencing Data. *Scientific Reports*, *7*(1), 43169. https://doi.org/10.1038/srep43169

Sun, C., & Medvedev, P. (2016). VarMatch: robust matching of small variant datasets using flexible scoring schemes. *Bioinformatics*, *33*(9), 1301–1308. https://doi.org/10.1093/bioinformatics/btw797

Talwalkar, A., Liptrap, J., Newcomb, J., Hartl, C., Terhorst, J., Curtis, K., Bresler, M., Song, Y. S., Jordan, M. I., & Patterson, D. (2014). SM a SH: a benchmarking toolkit for human genome variant calling . *Bioinformatics*, *30*(19), 2787–2795. https://doi.org/10.1093/bioinformatics/btu345

Tan, A., Abecasis, G. R., & Kang, H. M. (2015). Unified representation of genetic variants. *Bioinformatics*, *31*(13), 2202–2204. https://doi.org/10.1093/bioinformatics/btv112

Tewhey, R., Bansal, V., Torkamani, A., Topol, E. J., & Schork, N. J. (2011). The importance of phase information for human genomics. *Nature Reviews Genetics*, *12*(3), 215–223. https://doi.org/10.1038/nrg2950

Watkins, M., Rynearson, S., Henrie, A., & Eilbeck, K. (2019). Implementing the VMC Specification to Reduce Ambiguity in Genomic Variant Representation. *AMIA ... Annual Symposium Proceedings. AMIA Symposium*, *2019*, 1226–1235.