Università degli Studi di Napoli Federico II

Dipartimento di Medicina Molecolare e Biotecnologie Mediche
Area Didattica di Scienze Matematiche Fisiche e Naturali

Corso di Laurea Magistrale in Biotecnologie Mediche

Tesi sperimentale in Genetica Medica

Statistical analysis of negative selection using pangenomics data models

Analisi statistiche per detectare la selezione negativa usando dati pangenomici

Relatore interna:
Dott.ssa Gabriella De Vita
Relatrice esterna:
Dott.ssa Vincenza Colonna

Candidato:
Flavia Villani
Matricola
N79001438

A.A. 2019/2020

Contents

Chapter 1

Sommario / Abstract(1 pagina)

Studies of genomic selection typically assume a single linear reference genome. However, structural and complex variation can render these simplified models inapplicable. To address this limitation, in my thesis, I am working to implement a software library for the statistical analysis of negative selection using pangenomic data models. Typically represented in the Graphical Fragment Assembly (GFA) format, these models can represent whole genome alignments in a compact graphical structure. Because they embed the linear genomes from which they are constructed, we can choose a particular reference genome and project the variants (which appear as bubbles in the graph) back into any reference frame. Thus far, I have focused on algorithms for bubble detection that allow us to generate Variant Call Format (VCF) files from graphs. We can use this projection to drive standard population genetic analyses, or eventually as a mechanism to communicate results that we obtain from pangenome graph based population genetic analyses which I am designing.

Chapter 2

Introduction/(5-10 pagine)

2.1 Graphical Fragment Assembly (GFA) Format Specification

Format

2.2 Variant Call Format (VCF)

Format

2.3 Simulation

MP F-test

2.4 Application

Chapter 3

Results of the Research Group (5-10 pagine)

Descrive il progetto di ricerca di cui fa parte l'attività sperimentale svolta dal candidato. Vanno qui riportati i risultati ottenuti dal gruppo di ricerca in precedenza e/o in contemporanea rispetto all'attività svolta dal candidato che siano rilevanti per l'argomento della tesi. Nel caso in cui il candidato abbia svolto, nell'ambito del gruppo di appartenenza, un progetto di ricerca nuovo e autonomo, questa sezione può essere eliminata. In questa sezione è richiesto l'uso di illustrazioni.

Chapter 4

Aims of the thesis/(1-2 pagine)

The aims of the thesis is how drive standard population genetic analyses of negative selection using pangenomic data models. Typically represented in the Graphical Fragment Assembly (GFA) format, these models can represent whole genome alignments in a compact graphical structure. Thus far, I have focused on algorithms for bubble detection that allow us to generate Variant Call Format(VCF) files from graphs.

Chapter 5

Results achieved by the Candidate(5-20 pagine)

5.1 From Graphical Fragment Assembly (GFA) format to VCF

Data consists of GFA format from simulation of genomic sequences. From a GFA I'm obtained VCF.

DNA sequence graphs represented by (Figure 5.1) have two strands, we need a more precise way of addressing elements in the graph than nodes, which implicitly represent both strands. The handle, allows us to refer to one strand of a single node, which is the smallest addressable unit in a variation graph. Nodes have numeric identifiers (or IDs), and associated sequences. Edges link two handles. Paths are a series of steps which link a path identifier and a handle. **(**garrison2020).

I obtained the element of the VCF from GFA:

- CHROM: name of the path in the GFA
- POS: lengh of sequence in that position
- REF: path of reference that I chose
- ALT: sequence that not present in the ref, alt present in the bubble
- TYPE: SNV, INDEL, INSERTION

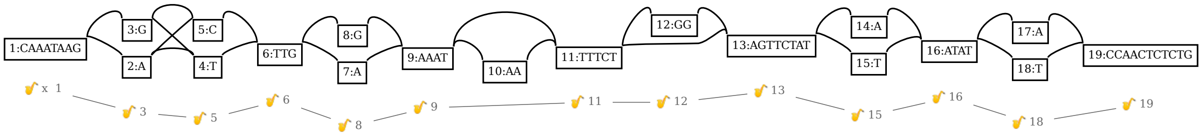


FIGURE 5.1: **Fig:** node id, sequence, insertion, deletion

```
H VN:Z:1.0
S 5 C
S 16 ATAT
S 12 GG
S 8 G
S 17 A
S 1 CAAATAAG
S 19 CCAACTCTCTG
S 6 TTG
S 11 TTTCT
S 9 AAAT
S 14 A
S 3 G
S 7 A
S 4 T
S 15 T
S 2 A
S 10 AA
S 18 T
S 13 AGTTCTAT
P x 1+,3+,5+,6+,8+,9+,11+,12+,13+,15+,16+,18+,19+ 8M,1M,1M,3M,1M,4M,5M,2M,8M,1M,4M,1M,11M
L 5 + 6 + 0M
L 16 + 17 + 0M
L 16 + 18 + 0M
L 12 + 13 + 0M
L 8 + 9 + 0M
L 17 + 19 + 0M
```

L 1 + 2 + 0M
L 1 + 3 + 0M
L 6 + 7 + 0M
L 6 + 8 + 0M
L 11 + 12 + 0M
L 11 + 13 + 0M
L 9 + 10 + 0M
L 9 + 11 + 0M
L 14 + 16 + 0M
L 3 + 4 + 0M
L 3 + 5 + 0M
L 7 + 9 + 0M
L 4 + 6 + 0M
L 15 + 16 + 0M
L 2 + 4 + 0M
L 2 + 5 + 0M
L 10 + 11 + 0M
L 18 + 19 + 0M
L 13 + 14 + 0M
L 13 + 15 + 0M

5.2 Identification of euploid samples for genetic analyses

vedere note per articolo da citare

In half of pregnancies in the first trimester the cause of PL are large chromosomal aneuploidies, such as trisomies or deletions of large chromosomal chunks (**goddijn2000genetic**, **zhang2009genetic**). With this project we want to focus on cases in which the embryo is euploid when analyzed with current diagnostic techniques i.e. comparative genomic hybridization (arrayCGH). Hence, samples were screened for chromosomal aneuploidies prior to whole-genome sequencing. This task was performed before my thesis work started, nevertheless I am describing it here because of its importance.

In 44 samples when screened for maternal contamination 11.4% drops off the analysis, due to technical challenges during sample collection. A first round of detection of aneuploidies on chromosomes 13, 15, 16, 18, 21, 22, X, and Y through Short Tandem Repeats analysis discarded 47.7% of samples. These types of repeats (tetra- or penta-nucleotide) are often expected to be found in heterozygosis, therefore triploidies are assumed when three alleles are found at several markers along a chromosome (complete) or part of it (partial). Similarly, uniparental disomy for a targeted region or chromosome is assumed when only one parental allele is amplified. Subsequent analysis through array-CGH and copy number variation detection form low-coverage sequencing discarded another 11.4% of samples (Figure 5.2).

The most common aneuploidy in our data set is the trisomy of chromosome 22 (15.9%), followed by trisomy of chromosome 16 (9%) and 18 (4.5%)(Figure 5.3).

Overall 29.5% of samples were euploid (Figure 5.3) and could proceed to whole-genome sequence analysis. By the time I started my thesis I had access to six samples for which sequence data was ready and available to be sequenced.

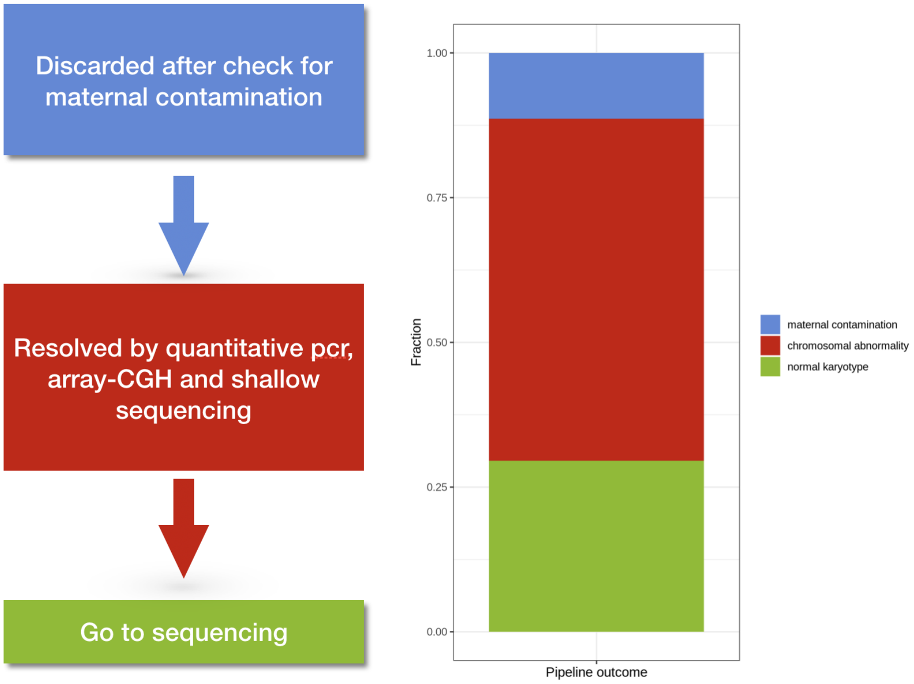


FIGURE 5.2: **Pipeline outcome.** Percentage of samples that was discarded/go to sequencing

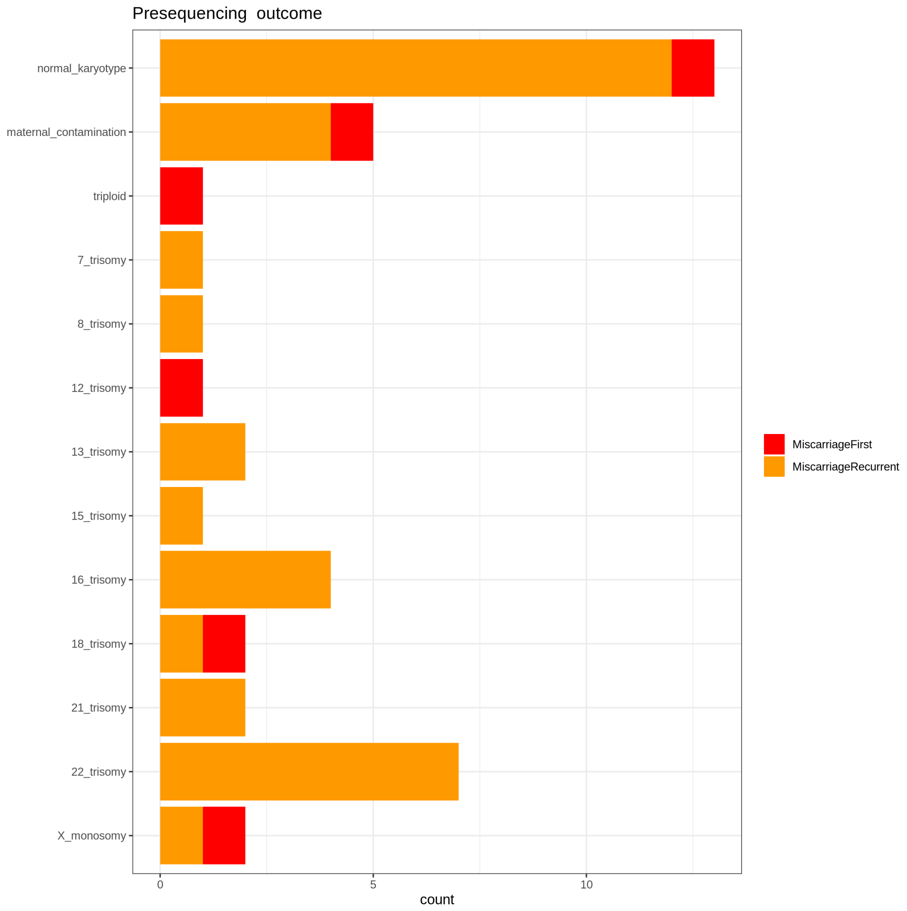


FIGURE 5.3: **Number of aneuploidies on Samples.** 29.5% of cases are diploid. The most common aneuploidies is a trisomy on chromosome 22 followed by trisomy on chromosome 16 and 18. 11.4% was discarded for maternal contamination.

5.3 Analysis of whole-genome sequence of embryos from pregnancy loss

5.3.1 Pipeline for the identification of genetic variants

The sequencing pipeline produces sequence data in the fastq format that constitute the raw sequence data and can be used to perform the analysis from scratch and customized pipelines on high performance computing machines. Figure 5.4 provides an overview of the pipeline used in this study.

I performed the **alignment** of the raw sequence data in form of reads against the most recent version of the human reference Genome (GRChg38.p12, **rosenbloom2015ucsc**) using BWA-MEM (**li2013aligning**) and SAMTOOLS (**li2009sequence**). The alignment produces files in the bam format that provide information on the quality of the alignemnt and enable some quality controls. In our data set XX% of reads were correctly aligned to the reference genome. The average coverage is XXXX INSERIRE DATI DI STATISTICHE sul BAM. Before proceeding to the next step I used SAMBAMBA (**tarasov2015sambamba**) to **refine** the data and remove PCR duplicates, i.e. reads and they came from the same DNA fragment that bias variant detection through increased homozygosity.

I used the refined bam file to perform **variant calling** using FREEBAYES (**garrison2012haplotype**) that produces data in the vcf format. The variant calling is the process of identification of variants from sequence data, where variants are chunks of sequence that differ from the reference genome. Genetic variants are classified as single nucleotide variants (SNVs), small insertions and deletions (indels), and structural variants (SVs, large genomic rearrangements). In addition to call variants, the software specifies whether samples are homozygous or heterozygous (genotype) at the variable sites and the genotype likelihood, i.e. the probability that the observed genotype is correct.

The final step is a second round of **refining** that includes three steps.

- I used VCFFILTER (**vcflib**) to filter variants for quality score(QUAL)>20. The QUAL is an estimate of how likely it is to observe a call by chance, and a value of 20 corresponds to 1% probability of having an incorrect genotype.

- I used VT (**tan2015unified**) to do the normalization that consist of two parts: parsimony and left alignment. Parsimony is the representation of a variant in as few nucleotides as possible without reducing the length of any allele to 0. Left alignment is the shift of the start position of a variant to the left till it is no longer possible to do so (Figure 5.5) (**tan2015unified**).

- I used VT to deconstruct multiallelic variants in a VCF to allow for allelic comparisons between call sets (Figure 5.6) (**tan2015unified**).

Overall, the variant calling identified on average 4.7M variant site *per* sample. This number corresponds to the expectation in individuals of European ancestry(**1000genome2015global**). The most represented class is single nucleotide variant (83.2%) followed by insertions (6.85%) and deletions (6.84%)(Figure 5.7)(Table **??**).

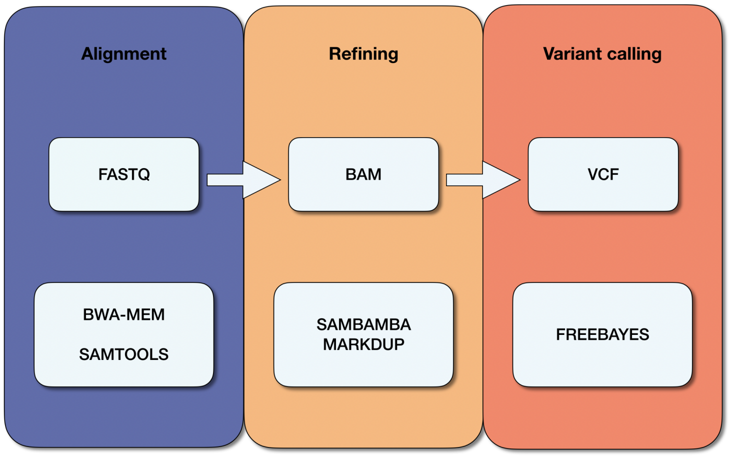


FIGURE 5.4: **Pipeline.** From FASTQ to VCF.

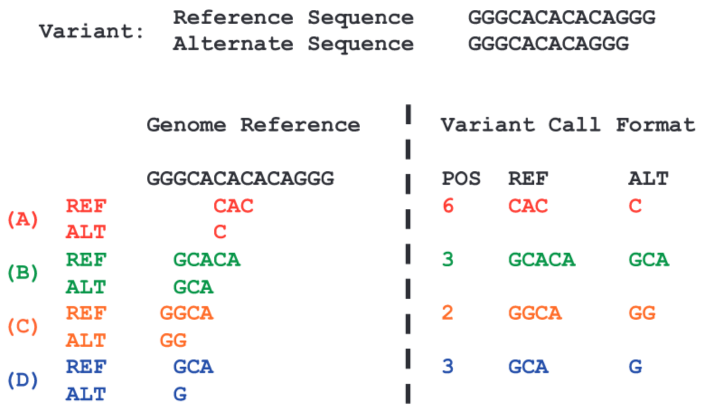
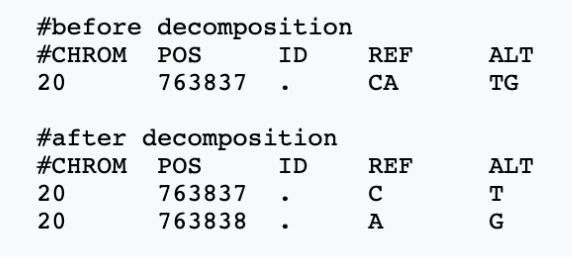


FIGURE 5.5: **Example of VCF entries representing the same variant.** *Adapted from* **tan2015unified**. Left panelaligns each allele to the reference genome, and the right panel represents thevariant in VCF. (A) is not left-aligned (B) is neither left-aligned nor parsimoni-ous, (C) is not parsimonious and (D) is normalized



```
#before decomposition
#CHROM   POS      ID    REF     ALT
20       763837   .     CA      TG

#after decomposition
#CHROM   POS      ID    REF     ALT
20       763837   .     C       T
20       763838   .     A       G
```

FIGURE 5.6: **Decomposes biallelic clumped variant** for allelic comparisons between call sets
Adapted from **tan2015unified**.

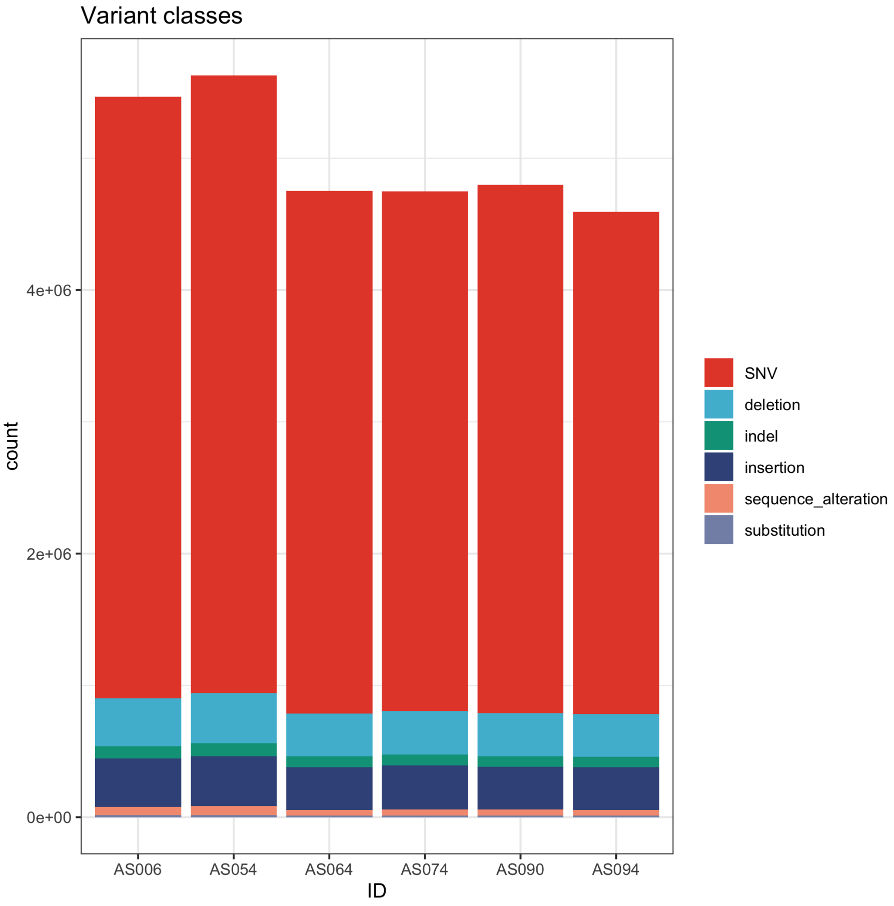


FIGURE 5.7: **Variant classes among all samples**

5.3.2 Functional annotation of genetic variants

Once the genetic variants are identified, the following challenge is to understand their functional consequences, i.e.the effect that they produce on the gene and if they cause a specific phenotype. I have used VARIANT EFFECT PREDICTOR (VEP, **mclaren2016ensembl**) provided by Ensembl (**howe2020ensembl**) to annotate the genetic variant discovered in the six GREP samples. VEP is currently the most updated and comprehensive toolset for the analysis, annotation, and prioritization of genomic variants in coding and non-coding regions based on an extensive collection of genomic annotations. VEP can determine the effect of variants (SNPs, insertions, deletions, CNVs or structural variants) on genes, transcripts, and protein sequence, as well as regulatory regions.

Table **??** (adapted from **mclaren2016ensembl**) describes the classification of the consequences in the Ensembl Variation data base. The impact of the variant on the gene product is classified in four categories: high, moderate, low, and modifier. Examples of variations with high impact are mutations that cause premature termination of the transcription (stop gained) or that suppress transcription (start lost). The classification includes variants in genic (e.g. missense), regulatory (e.g. located in transcription factor binding site) and intergenic regions. In the analysis of the GREP samples we refer to these categories.

Overall, TABELLA sample #high #moderate #low #modifier

Figure 5.8 shows the outcome for the most deleterious variant of VEP stratified by categories of impact. Among the variants with high impact ....

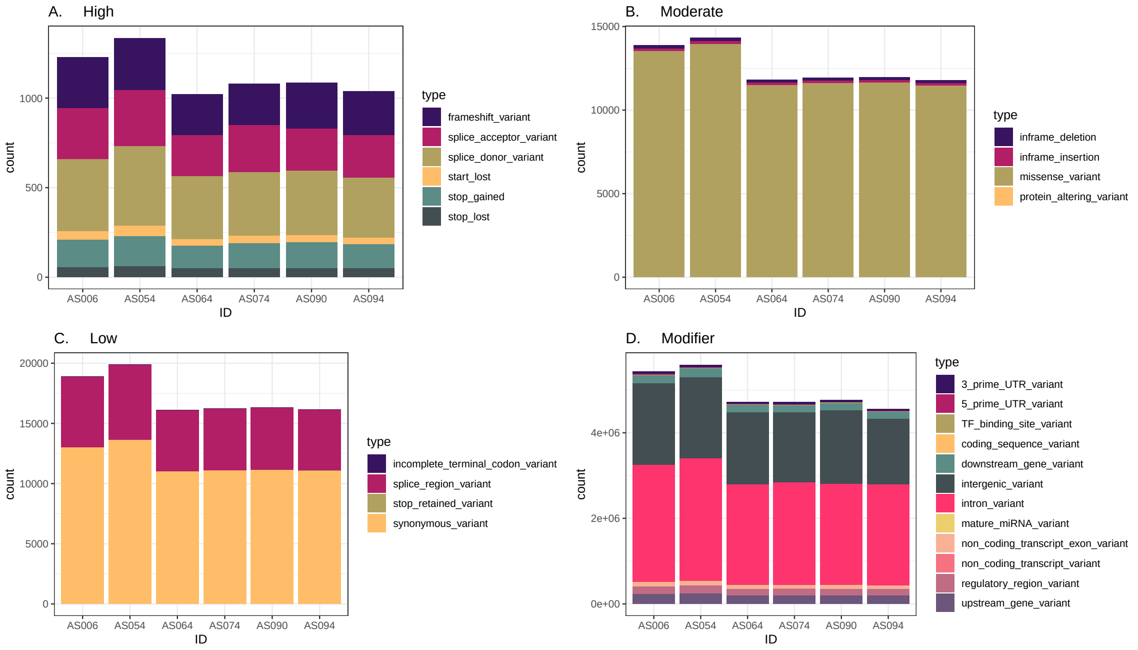


FIGURE 5.8: **Fig A High:** XXXX **Fig B Moderate:** XXX.**Fig C Low:** XXX.**Fig D Modifier:** XXX.

5.3.3 Identification of causative variants

Development of a pipeline for variant prioritization

The rationale behind the identification of genetic variants responsible for PL is based on the hypotheses that the causative variants are likely to be among those classified as detrimental and that more than one mutation can contribute to the phenotype. Furthermore, we assume that the phenotype is a complex trait, i.e. the combination of variants at more than one gene might have caused the miscarriage, therefore we seek for a list of variants in more than one gene or regulatory region, rather than a single variant. Under these assumptions I used the information contained in the variant annotations to search for those more likely to be deleterious and possibly causative.

I contributed to develop a pipeline (`https://github.com/ezcn/grep`) that analyze genomic variants within individual genome sequences to prioritize those possibly causative. A scheme of how the algorithm works is presents in Figure 5.9. The pipeline is optimized for genomic regions containing coding sequences, while the analysis of regulatory regions is under development. In particular, in this first version of the pipeline we use a combination of the following information:

- **Allele that cause the consequence.** We consider the **count** of the allele, i.e. if the individual has one or two allele, assuming that having two alleles is worst than having only one. We consider **allele frequency** in the 1000Genomes (**1000genome2015global**) and gnomeAD (**karczewskigenome**) data bases, representing reference population; we assume that if the allele is common than it is less likely to be detrimental.
- **Type and severity of the consequence.** We use the ranking illustrated in Table **??** to give priorities to consequences with higher impact on the phenotype.
- **pLI score.** pLI is the probability of a gene of being loss-of-function intolerant (**lek2016analysis**). A value of pLI>0.9 suggests that the gene is likely to not tolerate mutations that alter its function, therefore variants located in these genes are more likely to be prioritized.
- **CADD score.** The Combined Annotation Dependent Depletion (CADD) is a score that integrates multiple annotations into one metric by contrasting variants that survived natural selection with simulated mutations (**kircher2014general**, **rentzsch2019cadd**). The highest the CADD score, the more likely is the pathogenicity of a variant and it applies both to coding and non-coding variants.
- **Genes associated with miscarriages or involved in embryonic development.** We assign high priority to variants located in these sets of genes: 4651 essential (**blomen2015gene**, **wang2015identification**, **hart2015high**); 3801 lethal (**dawes2019gene**); genes involved in the embyonic development (GO:0009790); genes DDD (**firth2011deciphering**, **firth2009decipher**); a manually curated list of genes that have been associated with miscarriages, obtained from literature (**laisk2019genetic**, **pereza2017systematic**, **qiao2016whole**, **rull2012genetics**, **colley2019potential**, **quintero2017novel**).

The first part of the pipeline works at *per*-individual, *per*-site level producing annotations like those illustrated in Figure 5.10. Annotations are filtered or ranked in the second part of the pipeline. Overall, the pipeline takes as input the the genetic variants in vcf format and produces a table with sites that pass the filtering or have high rank (Figure 5.9).

Identification of putatively detrimental variants in the *AHNAK2* gene by raw filtering of annotated variants

Meanwhile, as a proof of principle, we applied raw filtering based on the combination of several criteria like allele frequencies in the general population (selected to be extremely rare), homozygosity, impact of the allele with consequences. Our preliminary results identify a number of putatively detrimental mutations. Among those, three samples carry two homozygous missense mutations in the *AHNAK2* cancer-related gene, coding a cytoplasmic nucleoprotein whose high expression is associated with negative prognosis of several cancers.

The criteria with which the first filter was made are the following:

- Rare variants (allele frequency <1%)
- VEP impact Moderate or High
- Homozygous for the allele with consequences

The resulting after this kind of filtering is that all samples have a low number of candidate sites and their consequence is only Missence variant (Figure 5.11).

Identification of putatively detrimental variants in the *LAMA5* gene by ranking of annotated variants

Within genic regions we identified three to ten genetic variants per sample of high (splice-donor) and moderate (missense) impact with frequency lower than 5% in gnomAD and 1000 Genomes populations, located in twenty-eight genes involved in early embryonic development. Four genes with these variants are involved in the gonadotropin-releasing hormone receptor pathway and two in the integrin signalling pathway. Of particular relevance four samples have multiple hetrozygous missense mutations at four sites in *LAMA5*, a gene that regulates the attachment, migration, and organization of cells into tissues during embryonic development.

After this fist attempt we made another filter not by limiting for homozygous variants but by checking if the variant is into a lists of genes important for embryonic development or for the cell cycle. This lists was taken by GENEONTOLOGY (GO), we have searched for a specific category and taken all genes below it.

We have more "candidate" sites and for each site we know if there is in a gene involved in Embryo development (GO:0009790) or in the cell cycle (GO:0007049) or in both (Figure 5.12). The type of Consequence is much similar to the previous filter with most of all sites that have a Missence variant (Figure 5.13), furthermore, the number of sites is on the order that we want and it will be relevant for the future development of this software.

With this lists, we have repeat the first filter and we have seen that the homozygous sites are less enriched then heterozygous sites (Figure 5.14).

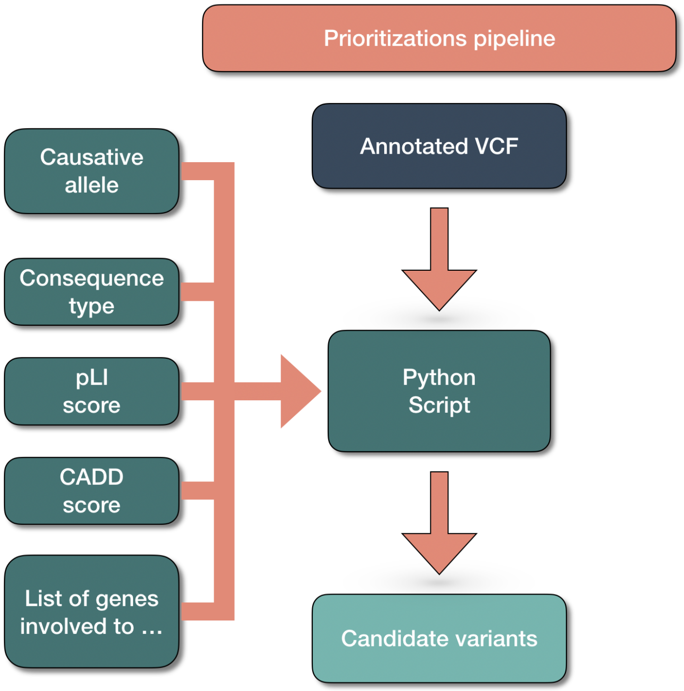
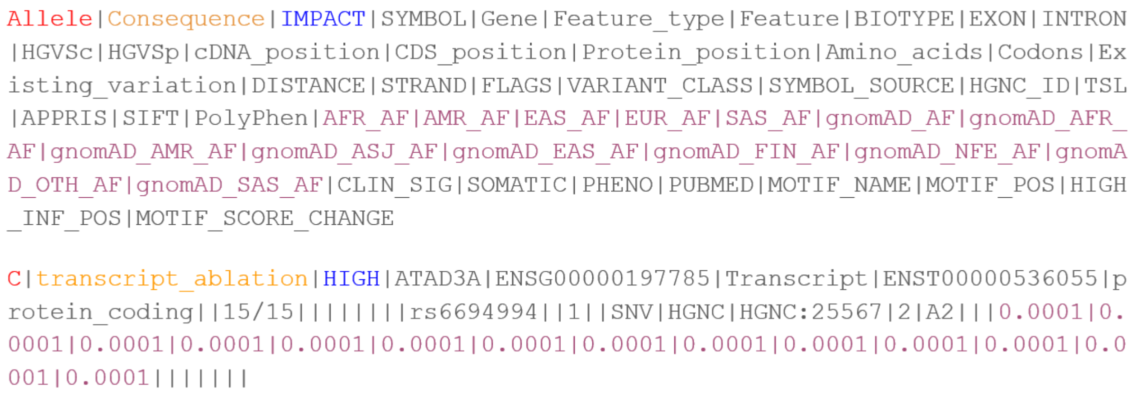


FIGURE 5.9: **Prioritization pipeline with combined scores and lists of genes.**



```
Allele|Consequence|IMPACT|SYMBOL|Gene|Feature_type|Feature|BIOTYPE|EXON|INTRON
|HGVSc|HGVSp|cDNA_position|CDS_position|Protein_position|Amino_acids|Codons|Ex
isting_variation|DISTANCE|STRAND|FLAGS|VARIANT_CLASS|SYMBOL_SOURCE|HGNC_ID|TSL
|APPRIS|SIFT|PolyPhen|AFR_AF|AMR_AF|EAS_AF|EUR_AF|SAS_AF|gnomAD_AF|gnomAD_AFR_
AF|gnomAD_AMR_AF|gnomAD_ASJ_AF|gnomAD_EAS_AF|gnomAD_FIN_AF|gnomAD_NFE_AF|gnomA
D_OTH_AF|gnomAD_SAS_AF|CLIN_SIG|SOMATIC|PHENO|PUBMED|MOTIF_NAME|MOTIF_POS|HIGH
_INF_POS|MOTIF_SCORE_CHANGE

C|transcript_ablation|HIGH|ATAD3A|ENSG00000197785|Transcript|ENST00000536055|p
rotein_coding||15/15|||||||rs6694994||1||SNV|HGNC|HGNC:25567|2|A2|||0.0001|0.
0001|0.0001|0.0001|0.0001|0.0001|0.0001|0.0001|0.0001|0.0001|0.0001|0.0001|0.0
001|0.0001||||||
```

FIGURE 5.10: Header and example output of VEP software

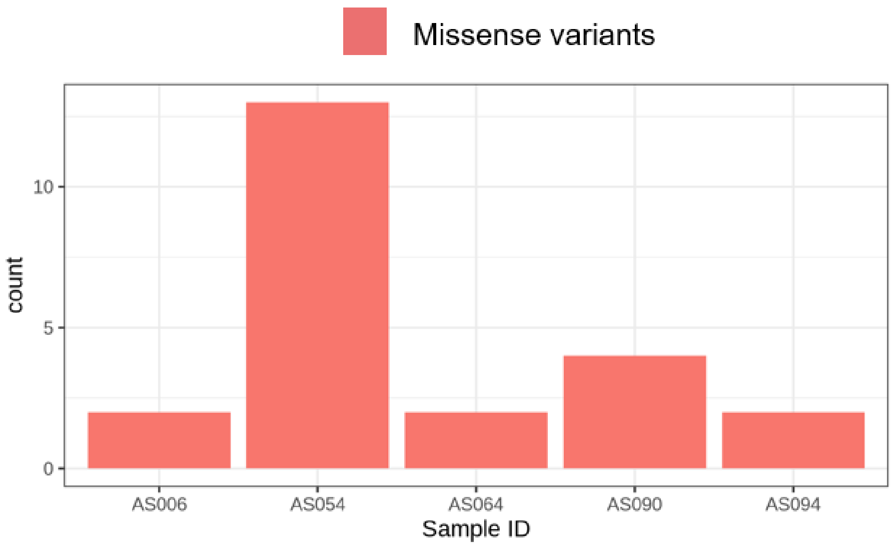


FIGURE 5.11: All the sites have only Missence variant.

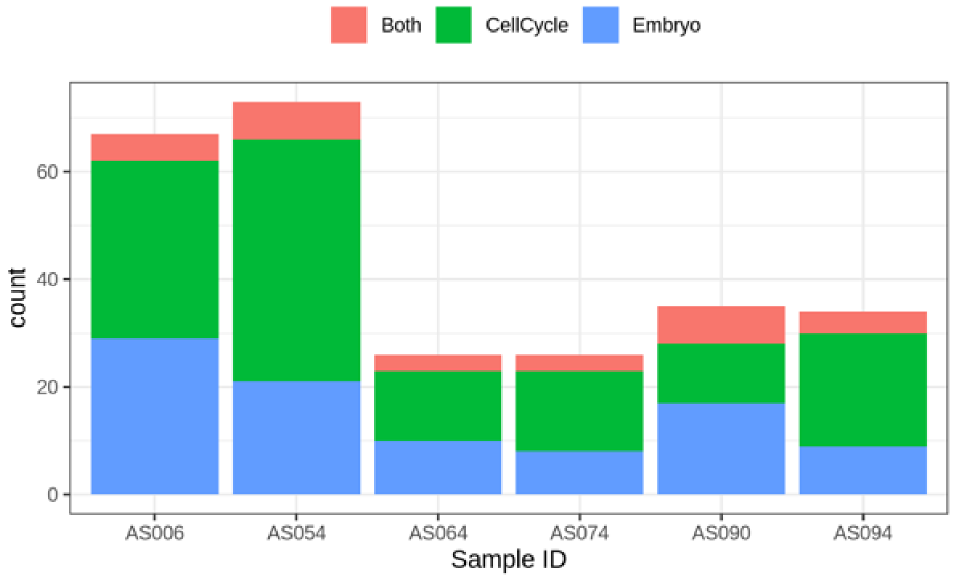


FIGURE 5.12: Distribution of consequence after checking in Embryo or/and CellCycle lists.

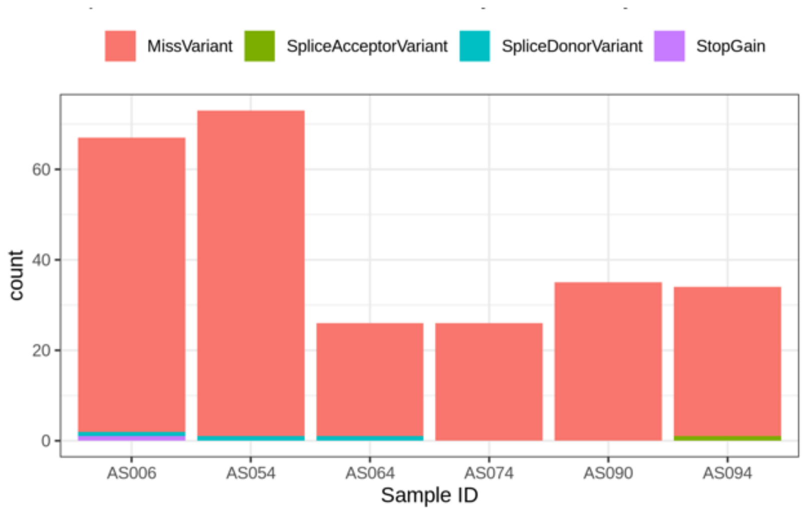


FIGURE 5.13: Type of Consequence after filtering with gene lists.

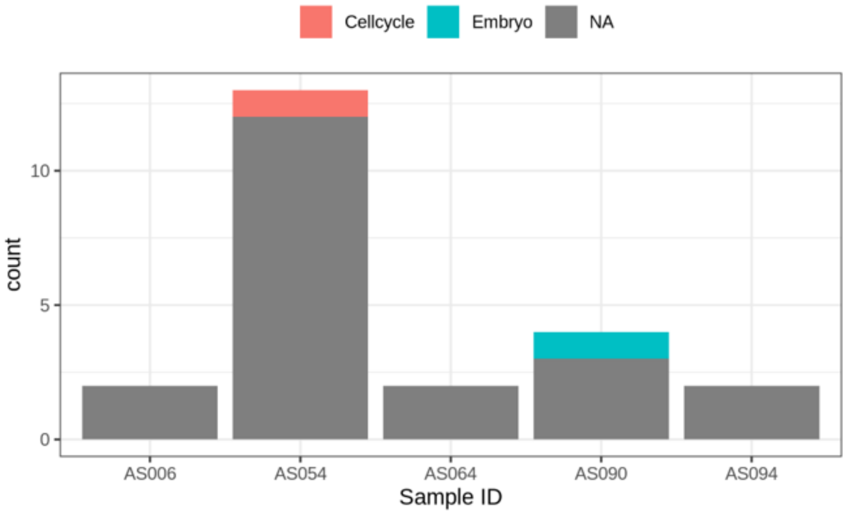


FIGURE 5.14: Precence or not in gene lists for samples that are homozygous.

5.3.4 Overlapping.

With the results provided by our software, we have checked if there is an overlapping between samples in the homozygous sites. In half of the samples, we have overlapping in the same gene and transcript (Table **??**).

Wiht Ensembl transcript we know the gene is *AHNAK2*. In two of three is the same variant, change of Glutammic acid in Aspartic acid, (supported by Variant ID) the remaining sample has another variant, change of Leucine in Valine, further upstream than the others (Figure 5.15).

Nella discussione parlare maggiormente sul perchè l'overlap sia una cosa molto positiva in soli 3 campioni. Parlare anche di cosa si conosce del gene.

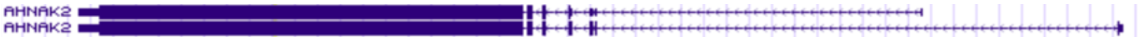


FIGURE 5.15: *AHNAK2* overlapping transcripts (below).

Chapter 6

Conclusions and Future Perspectives(3-6 pagine)

Chapter 7

Materials and Methods(5-10 pagine)

7.1 Explore little graph and detection bubble

Depth first traversal or Depth first Search is a recursive algorithm for searching all the vertices of a graph or tree data structure. The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The DFS algorithm works as follows:

-Start by putting any one of the graph's vertices on top of a stack. -Take the top item of the stack and add it to the visited list. -Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of stack. -Keep repeating steps 2 and 3 until the stack is empty.

```
def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    print(start)
    for next in graph[start] - visited:
        dfs(graph, next, visited)
    return visited

graph = {’0’: set([’1’, ’2’]),
         ’1’: set([’0’, ’3’, ’4’]),
         ’2’: set([’0’]),
         ’3’: set([’1’]),
         ’4’: set([’2’, ’3’])}

dfs(graph, ’0’)
```

Breadth first traversal or Breadth first Search is a recursive algorithm for searching all the vertices of a graph or tree data structure. he algorithm works as follows:

-Start by putting any one of the graph’s vertices at the back of a queue. -Take the front item of the queue and add it to the visited list. -Create a list of that vertex’s adjacent nodes. Add the ones which aren’t in the visited list to the back of the queue. -Keep repeating steps 2 and 3 until the queue is empty.

```
def bfs(graph, root):
    visited, queue = set(), collections.deque([root])
    visited.add(root)
    while queue:
        vertex = queue.popleft()
        for neighbour in graph[vertex]:
            if neighbour not in visited:
                visited.add(neighbour)
                queue.append(neighbour)

if __name__ == ’__main__’:
    graph = {0: [1, 2], 1: [2], 2: [3], 3: [1,2]}
    breadth_first_search(graph, 0)
```

I implement this code that is availabe on the git repository of the project (https://github.com/ezcn/grep)

https://www.programiz.com/dsa/graph-bfs (guardare qui)

7.2 Explore large graph

For this part I use a library of Python.

Odgi representing large genomic variation graphs with minimal memory overhead requires a careful encoding of the graph entities. Odgi is based on a node-centric encoding of the graph that is designed to improve cache coherency when traversing or modifying the graph. This encoding is split between graph topology and paths, which was found to be important to yielding a good balance of runtime performance and memory usage on real-world graphs with large path sets. Each node's its sequence and inbound and outbound edges are encoded in a byte array using a variable-length integer encoding scheme. Edges are described in terms of a relative offset between the rank of this node in the sorted array of nodes of the graph and the node to which the edge arrives (or from which it emanates). The set of steps traversing each node is recorded in a second dynamic integer vector of fixed width entries, compressed so that only largest integer entry is stored at full width. Each step links a path identifier, relative ranks of the previous and successive node traversals on the path, and the rank of the next or previous step among the path steps recorded at that node. **garrison2020**

Given a graph in GFA format(vedere come inserire il formato e mettere il link), we can build the odgi format using odgi build.

```
odgi build -g lil.gfa -o lil.odgi
PYTHONPATH=~/odgi/lib python3

Now we can load the graph and check how big it is:

import odgi

g = odgi.graph() # instatiate a graph
g.load('file.odgi')
```

For convert GFA format in VCF the code is availabe on the git repository of the project (https://github.com/ezcn/grep)

7.3 Whole-genome sequence analyses

7.3.1 Sequencing

Sequencing was done through a service provider (Macrogen s.r.l). In particular, libraries for sequencing were prepared using the Illumina TruSeq DNA PCR-free Library (insert size 350bp) and samples were sequenced at 30X mapped (110Gb) 150bp PE on HiSeqX. Data were released in as fastq files.

7.3.2 Alignment with reference.

Reads in the fastq file were aligned against the reference genome GRChg38.p12 (**rosenbloom2015ucsc**) using BWA and SAMTOOLS. BWA (https://github.com/lh3/bwa) "is a software package for mapping low-divergent sequences against a large reference genome, such as the human genome." Of the three algorithms available in BWA we used BWA-MEM that is generally recommended to analyze 70-100bp Illumina reads.

For each sample, the following command line takes in input fastq file (paired-end) and gives as output a raw-bam file.

```
~/bin/bwa mem -t 16 -R "@RG\tID:$idsample\tSM:$idsample" \
/mpbastudies3/IMMA/hg38/hg38.p12.fa \
/mpbastudies3/181113_Vincenza-Colonna/$namedir/$nameflgz1 \
/mpbastudies3/181113_Vincenza-Colonna/$namedir/$nameflgz2 | \
/mpba0/mpba-sw/samtools view -b - > \
/mpbastudies3/IMMA/samples/$idsample.raw.bam
```

- BWA
- SAMTOOLS
- SAMBAMBA
- TABIX
- BGZIP
- FREEBAYES
- script in-house github <——

- VEP
- array-CGH (?)
- quantitative PCR (?)
- python (???)
- R (???)

SAMTOOLS (https://github.com/samtools/samtools) is a set of utilities for interacting with and post-processing short DNA sequence read alignments in the SAM (Sequence Alignment/Map), BAM (Binary Alignment/Map) and CRAM formats (wiki). We have used this software downstream BWA-MEM for obtain the BAM from fastq.

Remove PCR duplicate. During PCR the machine introduces a several PCR duplicate in our sequence, for remove them we have used SAMbamba.

SAMbamba (https://github.com/biod/sambamba) is a high performance highly parallel robust and fast tool (and library), written in the D programming language, for working with SAM and BAM files.Current functionality is an important subset of samtools functionality, including view, index, sort, markdup, and depth (git).We have used the function markdup to remove PCR duplicates.

```
~/bin/sambamba markdup -t 8 -p  \
--tmpdir /scratch --overflow-list-size 500000 \
/mpbastudies3/IMMA/samples/${idflrbam}.raw.sorted.bam \
/mpbastudies3/IMMA/samples/${idflrbam}.bam
```

The command line takes in input a raw version of the BAM file for each sample and gives us as output the BAM file without PCR duplicates.

VariantCalling. Is the process that allows us to identifies sites that differ from the reference for each sample. We have choose FREEBAYES (https://github.com/ekg/freebayes) that is a Bayesian genetic variant detector designed to find small polymorphisms, specifically SNPs (single-nucleotide polymorphisms), indels (insertions and deletions), MNPs (multi-nucleotide polymorphisms), and complex events (composite insertion and substitution events) smaller than the length of a short-read sequencing alignment (git).

```
~/bin/freebayes -f /mpbastudies3/IMMA/hg38/hg38.p12.fa \
-r ${chr} --gvcf -g 2000  /mpbastudies3/IMMA/samples/${idsample}.bam
```

For use this command line is necessary the reference genome (hg38.p12) in fasta format and a BAM for each sample. Furthermore is possible for speedup the process do independent analysis for each chromosome.

Compress and indexing. When a file was generated is necessary do a compressing and indexing for each one.

Bgzip (https://www.htslib.org/doc/bgzip.html#DESCRIPTION) compresses files into a series of small (less than 64K) 'BGZF' blocks. This allows indexes to be built against the compressed file and used to retrieve portions of the data without having to decompress the entire file (wiki).

```
~/bin/bgzip /mpbastudies3/IMMA/samples/vcf/${idsample}.vcf > \
/mpbastudies3/IMMA/samples/vcf/${idsample}.vcf.gz
```

The command line take in input the output (VCF) from variant calling and give us the compressed file.

Tabix (https://www.htslib.org/doc/tabix.html) indexes a TAB-delimited genome position file in.tab.bgz and creates an index file.

```
~/bin/tabix -p vcf /mpbastudies3/IMMA/samples/vcf/${id}.${chr}.vcf.gz
```

The command line take in input the compressed file from BGZIP and give the indexed one.

Variant Effect Predictor. VEP (https://grch37.ensembl.org/info/docs/tools/vep/index.html) determines the effect of your variants (SNPs, insertions, deletions, CNVs or structural variants) on genes, transcripts, and protein sequence, as well as regulatory regions. Simply input the coordinates of your variants and the nucleotide changes to find out the:

- Alleles, Genes and Transcripts affected by the variants
- Location of the variants (e.g. upstream of a transcript, in coding sequence, in non-coding RNA, in regulatory regions)
- Consequence of your variants on the protein sequence (e.g. stop gained, missense, stop lost, frameshift)
- Known variants that match yours, and associated minor allele frequencies from the 1000 Genomes Project

- SIFT and PolyPhen-2 scores for changes to protein sequence

In our case, we have used the command-line version (v96.3) with chosen parameters reported in the command line.

```
singularity run -B /mpbastudies3/IMMA/samples/:/data \
/mpba0/mpba-sw/biocontainers/vep-96.3.img \
--af_1kg --af_gnomad --appris --biotype \
--buffer_size 5000 --check_existing --distance 5000 \
--fork 4 --polyphen b --pubmed --regulatory --sift b \
--species homo_sapiens --symbol --tsl --cache \
--dir_cache /mpba0/vcolonna/vepcache/ --offline \
--vcf --variant_class -i /data/vcfconcat/${id}.fullVcf \
-o /data/vepVCF/${id}.fullvep
```

This software takes in input a final version of the VCF (filtered) and gives to us an annotated version with all the parameters written in the command-line.

Acknowledgements