Analysis Protocol:

## 1) Sort and index the bam files

- Sort a BAM file:

*samtools sort Control.bam > Control.sorted.bam*

The default is to sort by position. Did the same for the Tumor.bam

- Create index:

*samtools index Control.sorted.bam*

The index file is used by other softwares to retrieve information about the bam and sam files

- Count reads in BAM file:

*samtools view -c Tumor.sorted.bam*

-> we have 15039503 reads

*samtools view -c Control.sorted.bam*

-> we have 19720171 reads

Reads that have a mapping quality >x:

x=30:

*samtools view -c -q 30 Tumor.sorted.bam*

-> we have 11678731 reads

*samtools view -c -q 30 Control.sorted.bam*

-> we have 15210703 reads

-q -> threshold for the quality of the reads -> here we filter for reads with quality > 30

x=25:

*samtools view -c -q 25 Tumor.sorted.bam*

-> we have 12020882 reads

*samtools view -c -q 25 Control.sorted.bam*

-> we have 15703445 reads

We can easily state that the reads are associated with high quality.

- Explore statistics-> nb no info on duplicates or quality because we didn't do an analysis on duplicates yet

**a) General statistics :**

*samtools flagstat Control.sorted.bam*

Control:

99.75% mapped

0.23% singletons

10030 reads with mate mapped to a different chr (with mapQ>=5)

Tumor:

99.96% mapped

0.03% singletons

7572 reads with mate mapped to a different chr (with mapQ>=5)

We need to operate an analysis on the transversion-have a look in IGV to see if we have something

**b) Detailed statistics:**

*samtools stats Control.sorted.bam > stat.control.txt; less stat.control.txt*

Control:

**avarage quality: 30.4**

pairs with other orientation: 3280

inser size standard deviation: 79.4 with an avarage insert size of 235.5

Tumor:

**avarage quality: 31.4**

pairs with other orientation: 2586

inser size standard deviation: 75.3 with an avarage insert size of 216.1

**c) Explore coverage statistics**

- Single base sum coverage per region -> samtools bedcov is used to have an idea on how many reads of the .bam map on the region kept in the .bed file:

*samtools bedcov Captured\_Regions.bed Control.sorted.bam > BEDCov.Control.CR.txt*

*samtools bedcov Captured\_Regions.bed Tumor.sorted.bam > BEDCov.Tumor.CR.txt*

**Captured\_Regions.bed** -> contains info on the chromosomes: 15, 16, 17, and 18 -> includes the regions that have been selected for this experiment -> reads have been selected to be in this region -> simulate a target sequence -> the .bed that tell us that these are the regions we want to target with this protocol

**DNA\_Repair\_Genes.bed** -> info on genes -> .bed file that may help us in the late analysis -> using VCF Tools and limit the analysis to these regions. We have the coordinates of DNA damage genes -> to annotate and give a name to the regions in which we see mutations or indels etc.

*less BEDCov.Control.CR.txt*

*less BEDCov.Tumor.CR.txt*

Looking at the .txt we can see that we need to keep an eye on possible deletions and duplications

Output file: same region of the .bed plus another column indicating the number of reads mapping in that region

## 2) Realignment

Realign the .bam – no info on the indels (=hidden delitions)

**a) Create the .intervals -> tell us which regions can be realigned:**

*java -jar ../tools/GenomeAnalysisTK.jar -T RealignerTargetCreator -R ../Annotations/**human\_g1k\_v37.fasta -I Tumor.sorted.bam -o realigner.intervals.Tumor* *-L Captured\_Regions.bed*

*java -jar ../tools/GenomeAnalysisTK.jar -T RealignerTargetCreator -R ../Annotations/human\_g1k\_v37.fasta -I Control.sorted.bam -o realigner.intervals.Control -L Captured\_Regions.bed*

---> -o is used to tell the software the name of the output file -> the output is a series of intervals

---> always same syntax

---> -R -> path of the reference (fasta of the human genome)

**b) Perform the realignment:**

*java -jar ../tools/GenomeAnalysisTK.jar -T IndelRealigner -R ../Annotations/human\_g1k\_v37.fasta -I Tumor.sorted.bam -targetIntervals Tumor.realigner.intervals -o Tumor.sorted.realigned.bam -L Captured\_Regions.bed*

*java -jar ../tools/GenomeAnalysisTK.jar -T IndelRealigner -R ../Annotations/human\_g1k\_v37.fasta -I Control.sorted.bam -targetIntervals Control.realigner.intervals -o Control.sorted.realigned.bam -L Captured\_Regions.bed*

All the positions in which there is a hidden deletion are now realigned -> all contained in the file .realigned.bam

**c) Count how many reads were realigned:** possible using the OC tag that is used to maintain the original CIGAR

*samtools view Tumor.sorted.realigned.bam | grep OC | wc -l*

number of reads that have been realigned: 2267

*samtools view Control.sorted.realigned.bam | grep OC | wc -l*

number of reads that have been realigned: 3158

## 3) Quality control -> Recalibration

Base quality score recalibration involves assigning accurate confidence scores to each sequence. Quality scores are critical for all the downstream analysis and systematic biases are a major contributor to bad calls. Systematic biases are a major contributor to lower quality control scores. Base quality score recalibration or BQRS is a method that adjust the PHRED quality scores to be more accurate by looking at every base in a BAM file.

4 steps:

**a.) BaseRecalibrator**: model the error modes and recalibrate qualities. Its inputs are a BAM file and the known sites. → based on the covariates, generate a BaseRecalibrator Table that tells us how much correction was needed.

*java -jar ../tools/GenomeAnalysisTK.jar -T BaseRecalibrator -R ../Annotations/human\_g1k\_v37.fasta -I ../Realignment/Tumor.sorted.realigned.bam -knownSites ../Annotations/**hapmap\_3.3.b37.vcf -o recal.table.Tumor -L Captured\_Regions.bed*

*java -jar ../tools/GenomeAnalysisTK.jar -T BaseRecalibrator -R ../Annotations/human\_g1k\_v37.fasta -I ../Realignment/Control.sorted.realigned.bam -knownSites ../Annotations/hapmap\_3.3.b37.vcf -o recal.table.Control -L Captured\_Regions.bed*

-> we limit the execution only to certain region of the genome -> specifically, it is a genes correlated to camcer-> possible via the -L parameter

-> In output we have a file called recalTable -> a text file we can explore: beginning with general info, going down in the file we have all the recalibration values for teh groups, with event type (M for Mismatches, D for deletion, I for insertions), there is also the info stratified by the scores. Plus, the empirical quality scores are usually lower than the original quality score.

**b.) PrintReads**: write recalibrated data to a BAM file thanks to the recalibration table produced in the previous step. Original qualities are retained with the OC flag.

*java -jar ../tools/GenomeAnalysisTK.jar -T PrintReads -R ../Annotations/human\_g1k\_v37.fasta -I ../Realignment/Tumor.sorted.realigned.bam -BQSR Tumor.recal.table -o Tumor.sorted.realigned.recalibrated.bam -L Captured\_Regions.bed --emit\_original\_quals*

*java -jar ../tools/GenomeAnalysisTK.jar -T PrintReads -R ../Annotations/human\_g1k\_v37.fasta -I ../Realignment/Control.sorted.realigned.bam -BQSR recal.table.Control -o Control.sorted.realigned.recalibrated.bam -L Captured\_Regions.bed --emit\_original\_quals*

**c.) The process is repeated to build the after model to evaluate remaining error**.

*java -jar ../Tools/GenomeAnalysisTK.jar -T BaseRecalibrator -R ../Annotations/human\_g1k\_v37.fasta -I Tumor.sorted.realigned.bam -knownSites ../Annotations/hapmap\_3.3.b37.vcf -BQSR Tumor.recal.table -o Tumor.after\_recal.table -L Captured\_Regions.bed*

*java -jar ../Tools/GenomeAnalysisTK.jar -T BaseRecalibrator -R ../Annotations/human\_g1k\_v37.fasta -I ../Realignment/Control.sorted.realigned.bam -knownSites ../Annotations/hapmap\_3.3.b37.vcf -BQSR recal.table -o after\_recal.table.Control -L Captured\_Regions.bed*

**d.) AnalyzeCovariates**: before and after plots are made based on recalibration tables.

*java -jar ../Tools/GenomeAnalysisTK.jar -T AnalyzeCovariates -R ../Annotations/human\_g1k\_v37.fasta -before recal.table.Tumor -after after\_recal.table.Tumor -csv recal.Tumor.csv -plots recal.Tumor.pdf*

*java -jar ../Tools/GenomeAnalysisTK.jar -T AnalyzeCovariates -R ../Annotations/human\_g1k\_v37.fasta -before recal.table.Control -after after\_recal.table.Control -csv recal.Control.csv -plots recal.Control.pdf*

-> we can have a look at the recal.pdf -> we can see that there are different context possibles -> some biases can be observed when dealing with machine artifacts -> we can see that the after calibration line is more uniform.

We could filter for the quality?

## 4) Duplicates

Duplicates are non-independent measurements of a sequence, as they are sampled from the exact same template of DNA, violating the assumptions of variant calling. Duplicates come from the same input DNA template, so they should have the same start position on reference. This is even more true for paired end, in which both the reverse and the forward read should have the same starting position. Duplicate sets are first identified, then the representative (”best copy”) read based on base quality scores an other criteria is chosen for each set.  
Marking the duplicates can be done in two different ways:

* MarkDUplicates from Picard, the golden standard.
* markdup from samtool. It requires the addition of mate tags to the BAM file through the fixmate command.

The samtools command is faster than the Picard one as it exploits the Cigar of the mate read to correct with a simple iteration. However the Picard command retains more reads because samtools’ command removes all the reads that have a mate mapped to a different chromosome, removing in this way structural variants.

**a) Prepare the .bam file: sort + index**

*samtools sort Tumor.bam > Tumor.sorted.bam*

*samtools index Tumor.sorted.bam*

Same for Control

**b) Use Picard MarkDuplicates**:

*java -jar ../../Tools/picard.jar MarkDuplicates I=Control.sorted.bam O=Control.sorted.dedup.bam REMOVE\_DUPLICATES=true TMP\_DIR=/tmp METRICS\_FILE=Control.picard.log ASSUME\_SORTED=true*

*java -jar ../../Tools/picard.jar MarkDuplicates I=Tumor.sorted.bam O=Tumor.sorted.dedup.bam REMOVE\_DUPLICATES=true TMP\_DIR=/tmp METRICS\_FILE=Tumor.picard.log ASSUME\_SORTED=true*

--> input file: .bam file sorted and indexed plus realigned and recalibrated

--> REMOVE\_DUPLICATES as true because we want to remove them

--> ASSUME\_SORTED is true because the .bam is sorted

--> METRICS\_FILE -> File to write duplication metrics to

open the .log file using calc (excel) - the most important lines are the 7 and the 8, columns E and on, correspond to the **summary of the removal of duplicates**: contains the number of reads that are unpaired, the number of duplicates found, the number of unmapped. Plus also the percentage of duplicates and the optical duplicates and the non optical duplicates.

Control -> tells us the number total reads: 21355813 -> the percentage of duplicates where 13.8%

Tumor -> tells us the number total reads: 18084804 -> the percentage of duplicates where 12.2%

*samtools index Control.sorted.dedup.bam*

*samtools index Tumor.sorted.dedup.bam*

--> needed to index again the .bam file -> because it is a new .bam file so also the index need to be done again

## 5) Variant Calling

What are variants?

* SNV → Single Nucloetide Variant → change in one nucleotide. The most common variation.
* Insertion
* Deletion

Even though the SNVs are the most frequent events the weight of the structural variance (duplicates and insertion) is higher because they deal with a large number of nucleotides and not only one as SNVs.

We can detect Variants via:

* Computation of the Allelic Fraction (AF) across all position in our genome. Thresholds are needed to distinguish genotypes → calcluate ratios of the reads that support the reference genotype (=the reference base) and reads that support the alternative allele → calculate the allele fraction → Based on the value of the allelic fraction we can infer if we are looking at a homozygous genome or a heterozygous genome: Homozygus if AF=0 or 1 Heterozygus if the AF≥0.2 or ≤0.8 → difficult to find an AF of 0.5!

Use basic statistic to distinguish genotypes: Binomial test → possible in R via **binom.test()**. → to obtain a statistic value of the AF we get! → we need to specify the number of successes (=number of reads that are hetero or homozygus), the number of unsuccess (=total number of reads) and the probability → here the probability correspond to the AF frequency. → based in the value of the test we can see if what we are observing is really an SNV  
NB. quality threshold can be added

* Computation of the Likelihood:
  + **Bayes’ rule for variant calling ->** Refined methods can find the genotype of each sample by calculating via Bayes’ rule the probability of each possible genotype
  + **Likelihood estimation for variant calling ->** The inference used as a gold standard relies on a likelihood function to estimate the probability of sample data given the proposed haplotype. The probability is computed by calculating the support of the alternative base based on quality.

Available tools to perform this task are:

* Bcftools
* GATK

**a) Operate the variant calling using bcftools**-> Bcftool pileup combined with a variant calling bcftool

*bcftools mpileup -Ou -a DP -f ../../Annotations/human\_g1k\_v37.fasta Control.sorted.bam | bcftools call -Ov -c -v > Control.BCF.vcf*

*bcftools mpileup -Ou -a DP -f ../../Annotations/human\_g1k\_v37.fasta Tumor.sorted.bam | bcftools call -Ov -c -v > Tumor.BCF.vcf*

-> Sample.BCF.vcf is the output file -> for the optional data we have all of them ???? look at the created file, thisi is for the old one ????, specifically all the info PASSED and the reads map in the chromosomes 20

-> as output we get also a .log file that tells us if there were some errors in the computation

**b) Operate the variant calling using GATK**

*java -jar ../../Tools/GenomeAnalysisTK.jar -T UnifiedGenotyper -R ../../Annotations/human\_g1k\_v37.fasta -I Control.sorted.bam -o Control.GATK.vcf -L chr20.bed*

*java -jar ../../Tools/GenomeAnalysisTK.jar -T UnifiedGenotyper -R ../../Annotations/human\_g1k\_v37.fasta -I Tumor.sorted.bam -o Tumor.GATK.vcf -L chr20.bed*

-> we obtain as output Sample.GATK.vcf -> map on chr 20. Quality -> low quality

**c) The output of the analysis is a VCF file so let's use vcftool to analyze it and work with it**

VCF is a text file format, most likely stored in a compressed matter, that contains meta-information lines, a header line and data lines, each containing information about a position in the genome.

The header line names 8 fixed, mandatory columns:

* CHROM: chromosome.
* POS: position.
* ID: semi-colon separated list of unique identifiers.
* REF: reference bases.
* ALT: comma separated list of alternative non-reference alleles called on at least one of the samples.
* QUAL: prhed-scaled quality score for the assertion made in ALT.
* FILTER: pASS if the position has passed all filters. "." when filters are not applied.
  + INFO: additional information. Genotype data are followed by a FORMAT column header and an arbitrary number of sample. Info on the depth (DP), the allelic frequency (AF), the ancestral allele (AA), the number of samples (NS), and so on.
* IDs. The header line is tab-delimited.

The metadata correspond to annotations that are made on the reference and on the reads, we can use this info to operate some comparison plus we have an idea on the quality and on the filter. Sometimes it is possible to not have the FILTER checked as PASSED, instead we find a string that indicate the test that the read didn’t pass, like for example the quality check → example if the quality is below 10 do not pass the filter (q10 instead of PASSED in the column FILTER).

The package vcftools can be used to perform a number of operations on VCF files:

* Filter out specific variants for quality, mean depth or allelic fraction.
* Compare files.
* Summarize variants.
* Convert to different file types.
* Validate and merge files.
* Create intersections and subsets of variants.

A variant calling protocol consists of two steps. In the first bcftools pileup provides the supported bases and their quality. The highest support is checked through a Bayesian model. A position with an alternative allele is reported in the sample VCF file. Then in the second step vcftools or GATK are used to produce a new VCF with the corresponding informations. GATK keeps more variants with respect to vcftools, but there is no method clearly better than the other. Usually the consensus is used to compare the output of the two tools and an intersection is built. Moreover the two tools show a lot of similarities in their output. Vcftools are more confident in making calls while GATK can introduce a portion of false positives. This can be reduced by increasing the filter on minimal coverage and reaching a set of more trustable parameters.

**a) Filter variants on the quality** -> treshold on the minimum quality at 20 and the coverage. First on the BCF.vcf

*vcftools --minQ 20 --min-meanDP 30 --remove-indels --vcf Tumor.BCF.vcf --out Tumor.BCF --recode --recode-INFO-all*

-> we get a .filtered.vcf

**--min-meanDP** *<float>*  
**--max-meanDP** *<float>*

Includes only sites with mean depth values (over all included individuals) greater than or equal to the "--min-meanDP" value and less than or equal to the "--max-meanDP" value. One of these options may be used without the other. These options require that the "DP" FORMAT tag is included for each site.

**--minQ** *<float>*

Includes only sites with Quality value above this threshold.

**b) Filter variants on the quality** -> treshold on the minimum quality at 20 and the coverage. On the GATK.vcf

*vcftools --minQ 20 --min-meanDP 30 --remove-indels --vcf Control.GATK.vcf --out Control.GATK --recode --recode-INFO-all*

**c) Operate the comparison between the two files .vcf-> file. outdiff\_files**

*vcftools --vcf Control.BCF.recode.vcf --diff Control.GATK.recode.vcf --diff-site*

*vcftools --vcf Tumor.BCF.recode.vcf --diff Tumor.GATK.recode.vcf --diff-site*

-> we get the info on each position for the read in both files -> we can compare the two files easily-> we can see the same event in both files? Same base call?

TASK2: Dedup, realign, recalibrate Sample.sorted.bam file e recompute variant calling

TASK3: Change --min-meanDP value to 20 and compare results

*vcftools --minQ 20 --max-meanDP 200 --min-meanDP 20 --remove-indels --vcf Control.BCF.vcf --out Control.BCF --recode --recode-INFO-all*

*vcftools --minQ 20 --max-meanDP 200 --min-meanDP 20 --remove-indels --vcf Tumor.BCF.vcf --out Tumor.BCF --recode --recode-INFO-all*

*vcftools --minQ 20 --max-meanDP 200 --min-meanDP 20 --remove-indels --vcf Control.GATK.vcf --out Control.GATK --recode --recode-INFO-all*

*vcftools --minQ 20 --max-meanDP 200 --min-meanDP 20 --remove-indels --vcf Tumor.GATK.vcf --out Tumor.GATK --recode --recode-INFO-all*

-> the limit on the coverage is higher so the number of reads in the .vcf file is dofferent

## 6) Variant Annotation

Variant annotation is a crucial step in linking sequence variants with changes in phenotype. Annotation results can have a strong influence on the ultimate conclusions of disease studies.

SnpEff is a variant effect predictor program categorizing each variant based on its relationship to coding sequenced in the genome and how it may change the coding sequence and affect the gene product.

Variant annotation depends on the set of transcripts used as the bases for annotation (= depends on the background so on the reads used, the sequences used, and the databases used to operate the annotations. Plus depends on the set of transcripts used). Widely used annotation databases such as ENSEMBL, RefSeq, and UCSC contain sets of transcripts that can be used for variant annotation.

Some common annotations are:

* **Putative\_impact/impact:** The most interesting for us! A simple estimation of putative impact/deleteriousness : (HIGH, MODERATE, LOW, MODIFIER). Estimation of how much the variant can have on the protein quantity → the variant is causing some trouble with the protein? High → yes a lot Moderate → Yes but not too much Low → Nope Modifier → we don’t know
* **Gene Name**: Common gene name (HGNC). Optional: use the closest gene when the variant is “intergenic”. Useful if we didn’t previously annotate.
* **Feature type**: Which type of feature (e.g. transcript, motif, miRNA, etc.)? It is preferred to use Sequence Ontology (SO) terms, but ‘custom’ (user-defined) is allowed.
* **Feature ID**: Depending on the annotation sources, this may be: Transcript ID (preferably using version number), Motif ID, miRNA, ChipSeq peak, Histone mark, etc. Depends on the annotation databases used. Note: Some features may not have ID (e.g. histone marks from custom Chip-Seq experiments may not have a unique ID)
* **Biotype**: The bare minimum is at least a description of whether the transcript is (“Coding”, or “Noncoding”).

1. **Call SNPEffector using VCF files, for both BCF and GATK**

*java -Xmx4g -jar ../Tools/snpEff/snpEff.jar -v hg19kg ../05\_VariantCalling/Data/Control.BCF.recode.vcf -s Control.BCF.recode.ann.html > Control.BCF.recode.ann.vcf*

-> We generate an annotate.vcf and a .html

-> let's explore the annotated vcf file: it is a normal VCF file with all teh info and other info lines that hae been hadded to the header of teh metadata -> info added by SNPEffector -> the most important is the annotation lines -> at a certain point of the INFO string added by SNPEffector, we have a code ANN that contain the info, divided by the pipe |. Some type of annotations are: protein\_coding, intron\_variant, missense\_variant, moderate\_effect, etc

-> it is a text file so it can be read!

-> we can extract the data creating a python code

-> look at the html file to have an overview -> tells us the command line lounches, the warnings (=not a big deal), errors (= top if it is equal to 0), number of variants and number of lines -> there can be differences in these two lines, here for example, we analyze 7432 lines and 7447 variants -> this difference is because of teh multialleles -> here we have 50 cases that have more than 1 variant, so a multiple alleles. The analysis of annotation is operated on the variants so we can hav emore variants than lines in the files.

We also have the genome length, the effective genome length -> differences because we used only the info on the chr 20 to generate the VCF files and the variant rate.

Table of Number variant by types -> all SNPs -> we filtered against indels and insertions

Summary and classifications of number of effects stratified by the impact -> biggest part of the effect predicted is modifier -> so unable to predict. After we have low and moderate effect -> not easy to predict the effect.

Summary and classifications of number of effects stratified by the functional class -> majority are missense

More detailed stratification of the effects divided by type and region (downstream, upstream, introns, splising sites, UTR regions, etc) -> here a big percentage of the calls fall in the intron region, 53%. Only 30% fall in the exonic regions -> a lot of time the calls are not happening in a sequence that give rise to a protein. So ot can be difficult to repdict the effect of these modification. Other modification can be found in the regulatory region, so the upstream regions -> change the level of expression of the proteins.

Graph of the variations divided by type

Quality summary -> min, max, mean, median, standard deviation, values and counts per values. Associated to a plot of the distribution of the values.

Insertion and deletion length -> histogram -> here we don't have them because there are no ins or del -> we left them out when creating the VCF file.

Matrix of the base changes for the SNPs -> we can see how often a base is changed into another base -> not an equal presentation -> not the same numbers for every change -> here a lot of time the C became A and G become T.

Summary of teh changing of the bases in transition and transvertion. Can be used to operate Mutational signals analysis: association of specific mutation signature to specific diseases -> like smoke or liver damage -> the type of changes observed is the same.

Allele frequency -> not super useful sometimes

Allele count -> not super useful sometimes

Hom/Het per sample -> not super useful sometimes

Codon change -> the changes in codons that pairs with the amino acids changes table -> we can see which amino acid changed -> as the color gets more dark, the change of codon is observed more.

Amino acids changes table

With all these info we can get an idea of what is happening in the sample -> bases that are changing and so on.

Variants by chromosome -> specific for each chromosome -> we can see where the variants are accumulating in the actual sequence

*java -Xmx4g -jar ../Tools/snpEff/snpEff.jar -v hg19kg ../05\_VariantCalling/Data/Control.GATK.recode.vcf -s Control.GATK.recode.ann.html > Control.GATK.recode.ann.vcf*

-> compare the results with the VCF file -> the number of variants here is lower. In this case, we have 84% of modifier and 61% of the SNPs are missense. The majority of teh variants are found in introns (=51%).

We can say, that even if the numbers are a little different, the results are comparable and both tools give us an info that is comparable -> there is not a best one respect to the two of them.

The main difference can be seen in the amino acid table.

1. **Call Annotate -> additional passage on the vcf files. We use databases to add annotations. We use two different annotation -> hapmap and clinvar**

SnpSift help you filter large genomic datasets in order to find the most significant variants. Basically another step in the filtering and annotation.

Annotate is the command. The input files are two .vcf file → samples we annotated with SNPEffector. But it works with any annotated .vcf file. Output: a .vcf file with some new info

*java -Xmx4g -jar ../Tools/snpEff/SnpSift.jar Annotate ../Annotations/hapmap\_3.3.b37.vcf Control.BCF.recode.ann.vcf > Control.BCF.recode.ann2.vcf*

*java -Xmx4g -jar ../Tools/snpEff/SnpSift.jar Annotate ../Annotations/clinvar\_Pathogenic.vcf Control.GATK.recode.ann2.vcf > Control.GATK.recode.ann3.vcf*

*java -Xmx4g -jar ../Tools/snpEff/SnpSift.jar Annotate ../Annotations/clinvar\_Pathogenic.vcf Control.BCF.recode.ann2.vcf > Control.BCF.recode.ann3.vcf*

*java -Xmx4g -jar ../Tools/snpEff/SnpSift.jar Annotate ../Annotations/hapmap\_3.3.b37.vcf Control.GATK.recode.ann.vcf > Control.GATK.recode.ann2.vcf*

-> let's open one file: annotate2 of BCF: We have a normal vcf file -> the info fields are populated one after the other, we have some new info and new annotations -> addition of annotation from the databases.

We can see that the info lines are different between annotation 2 and annotation 3 files from teh same tool -> the string of the annotations are different because we used different databases and tools

**c) Under, we ask specifically for the annotation relative to the clinical significance -> CLNSIG**

*cat Sample.BCF.recode.ann3.vcf | java -Xmx4g -jar ../Tools/snpEff/SnpSift.jar filter "(exists CLNSIG)"*

-> there is 1 variant, that is associated to clinical significance: Pathogenic.

We have a BRACA1 both in control and tumor samples

*cat Sample.GATK.recode.ann3.vcf | java -Xmx4g -jar ../Tools/snpEff/SnpSift.jar filter "(exists CLNSIG)"*

-> there is the same -> hereditary breast amd ovarian cancer syndrome

## 7) Ancestry Analysis

It is important to study ancestry because of the mixture of the genome of the human population.

Methods:

* 1. **Principal Component Analysis (PCA) based methods**

In principal component analysis methods, the low-dimensional projection of the data allows for maximally retained variance-covariance structure among the genotypes. Fast algorithms are available to solve the problem, but interpretation might be non-trivial.

Examples: SMARTPCA, EThSeq

* 1. **Model-based methods**

The explicit generative model for the data is based on Hardy-Weinberg equilibrium and linkage equilibrium. Probabilistic methods estimate ancestry information from the inference of the best model parameters that fit the data.

Examples: FastStructure

1. **Run SMARTPCA -> non possiamo farlo perché non abbiamo i file ped!!**

go in SMARTPCA folder

cd smartpca

Convert the PED and MAP file into SMARTPCA input files:

./EIG5.0.2/bin/convertf -p par.PED.EIGENSTRAT.100s

./EIG5.0.2/bin/convertf -p par.PED.EIGENSTRAT.16s

3) Run SMARTPCA analysis -> when we run it, it print the inpout files, the samples analyzed and the number of SNPs used. Plus, if a part of teh SNPs were lefted out/killed.

# This analysis is PCA based so a summary of the PCA is returned, with the eigenvector and the dimentions of SNPs and individuals.

# Plus, a plot is returned, containing the SNPs

./EIG5.0.2/bin/smartpca -p 1000GP\_Genotypes100s.pca.par

./EIG5.0.2/bin/smartpca -p 1000GP\_Genotypes16s.pca.par

**b) Run EthSEQ -> possiamo farlo, dobbiamo creare i vcf su cui vogliamo lavorare!**

go in the folder for ethseq

cd ethseq

run the script

Rscript Run.R # there is a R script already implemented to operate the EthSEQ anlysis

# what the script look like? Look under in the R SCRIPT PART!

# we get the outputs in teh folders:

# -> RGraph.Output -> the PCA plots of teh target, model and teh refinement process. We can understand teh stratification of teh ethnicity ansd see if the samples fall in a specific subspace for a specific ethnicity

# the 16s fiel is smaller tha n the 100s file -> only 16 points in the PCA plots

# -> the file RReport.PCA.Acord -> data of the report

# -> file Rreport.txt-> file that tell us if each samples are inside a specific population subspace or near a specific population, each row is a sample

**c) Run fastSTRUCTURE -> non possiamo farlo, non abbiamo i filess**

cd faststructure

## run the examples and create plots

# the -k parameter is used to define the number of ethnicity we should start our analysis with -> deciding k can be difficult -> if we know that there is for sure a certain number of population, it will be easier. Defining k is difficult in unsupervised clustering, usually different approaches are used and a statistical analyisi is often associated to teh result to understand in the k is good. A treshold on the max k can be inserted -> so it is not needed to set a minimum k but a max k is selected

# needed to run both structure.py and distruct.py code

# the code can be find in the folder or at the end of the page.

python fastStructure/structure.py -K 4 --input=1000GP\_Genotypes100s --output=1000GP\_Genotypes100s --full --seed=100

python fastStructure/distruct.py -K 4 --input=1000GP\_Genotypes100s --output=1000GP\_Genotypes100s.svg

python fastStructure/structure.py -K 4 --input=1000GP\_Genotypes16s --output=1000GP\_Genotypes16s --full --seed=100

python fastStructure/distruct.py -K 4 --input=1000GP\_Genotypes16s --output=1000GP\_Genotypes16s.svg

## 8) Somatic Variant Calling

Somatic Variant calling is highly relevant in the field of cancer and in blood-related diseases. Cancer often is obtained via alterations not present in the germinal cells → so alterations are accumulated in somatic cells.

Somatic variant calling consider single nucleotide aberrations, which can be distinguished in:

* Single nucleotide polymorphisms SNPs: mutations that are shared amongst a population. These are aberrations expected at a particular position for any member of the species. They are well-characterized and catalogued in dbSNP (databases).
* Single nucleotide variations SNVs: private mutations. They occur at low frequencies and are not common. They typically are non-synonymous mutations that result in amino acid change, impacting protein sequence and function. They are somatic mutations and are typically tumour-specific. → not strictly true that the aberration are specific to a single individual and that the frequency are low → recurrent mutations can happen in hotspots!

Substantial discrepancies exist among the calls from different callers. Callers appear to be less concordant for calling somatic SNVs than germline SNPs. Sensitivity and specificity vary across callers and along the genome within any caller.

cd Data

### Varscan

**## SNPs**

1) We perform a pileup -> we generated a pilep file because ot is needed for the tool we are using in the point 2

samtools mpileup -B -f ../../Annotations/human\_g1k\_v37.fasta ../../05\_VariantCalling/Data/Sample.sorted.bam > Sample.sorted.pileup

samtools mpileup -B -f ../Annotations/human\_g1k\_v37.fasta Control.sorted.realigned.recalibrated.debup.bam > Control.pileup

2) We use the tool mpileup2snp -> generate a cvf fiel that contains the SNPs that are found in the pileup bam file

java -jar ../Tools/VarScan.v2.3.9.jar mpileup2snp Control.pileup --p-value 0.01 --output-vcf 1 > Control.VARSCAN.vcf

# it is possible to change some threshold -> like the coverage and the minimum variance frequency. Plus we can change the statistical analysis using teh p-values. All teh filters used can be changed and all are printed to terminal.

# tell us the number of SNPs found and the number of indels found

3) Filter the file using vcftools

vcftools --max-meanDP 200 --min-meanDP 5 --remove-indels --vcf Control.VARSCAN.vcf --out Control.VARSCAN --recode --recode-INFO-all

-> we retain almost all the possible sites of teh SNPs : retain 2736 sites over 2829

4) Compare two files: the Control.BCF.recode.vcfobtained in the lesson 5 and the one obtained now

vcftools --vcf Control.BCF.recode.vcf--diff Control.VARSCAN.recode.vcf --diff-site

# create the diff-site file in output > contains the following columns: chromosome, position in first file, position in the second file, in\_file (1 or B=both), reference 1, reference 2, alternative 1, alternative 2. A dot is inserted if there is no call for that base

# we get a print to the terminal -> found 2126 sites in common in both files

**## Somatic point mutations**

1) mpileup of the tumoral and normal .bam files -> we use two files: the tumor and the normal .bam file, we only select the chromosome 15-18 because these analysis are really long -> we focus on specific positions

samtools mpileup -q 1 -f ../Annotations/human\_g1k\_v37.fasta Control.sorted.realigned.recalibrated.debup.bam > Control.sorted.pileup

Same for the tumor

2) We run VarScan tool with the impostation somatic on the pileup files

java -jar ../Tools/VarScan.v2.3.9.jar somatic Control.sorted.pileup Tumor.sorted.pileup --output-snp somatic.pm --output-indel somatic.indel --output-vcf 1

# if we only have a tumor sample and not a normal sample we have the risk to misinterpret the data -> what can we do? We can use a pannel of normal samples -> we use like 1 0samples that are normal and the "normal event" is defined as a distribution of what is observed in the pannel.

# The thresholds are printed

# Some estimations are performed

# There is a summary: number of position in tumor and shared in normal, number of reads with sufficien coverage, number of abberations: here we have: 2558 Germline, 269 LOH, 16 somatic and 12 unknown, 0 variant.

**## Annotation-> to annotate the aberration events we found out**

1) Filter a bit more the somatic.pm.vcf

vcftools --max-meanDP 200 --min-meanDP 30 --remove--indels --vcf somatic.pm.vcf --out somatic.pm --recode --recode-INFO-all

2) we call snpEff-> create the annotation of the SNPs-> predict the SNPs

java -Xmx4g -jar ../Tools/snpEff/snpEff.jar -v hg19kg somatic.pm.vcf -s somatic.pm.vcf.html > somatic.pm.ann.vcf

# -Xmx4g -> tells java how many giga of RAM can be used -> we can increase it if we have enough RAM

2) We have a look at the annotation file -> ann.vcf

cat somatic.pm.ann.vcf

## 9) Somatic Copy Number Calling

Copy number mutations involve large portions of the genome. We can distinguish between deletion or insertion (=amplifications). The spectra of deletions i much simpler to characterize: homozygous deletions are clear to spot, as no coverage is observed. For higher copy numbers the pattern is more difficult to identify as the combinatorial distribution in allelic distribution makes it difficult to recognize.

How can we quantify?

We can think of calculating a ratio and transforming it in log → better if the ratio is transformed in log2(ratio) because the data is more stratified and less compressed towards small or high numbers, specifically in the case of ratios.

Immagine che contiene testo, schermata, linea, Carattere

Descrizione generata automaticamente

NB.

If we are in a condition of normality → wild type and we register a hemizygous, the signal will give us a negative value for the log2 ratio If we are looking at a gain, the log2 ratio will be higher than the expected one → positive number. We are in a normal condition → with a quantity of copy equal from the two samples only when the ratio is equal to 1 meaning a log2 ratio equal to 0.

The effect of contamination of a tumor sample with normal cells is to bring the segmentation signal closer to 0 → there is a contribution from normal cells then the log2 ratio is lower and closer to 0.

The distribution of the log2 ratios will be closer to 0 → shift closer to 0 → see appunti a mano pag 46 of the lecture appunti.

When dealing with tumor data we need correction of the signal, otherwise there will be a problem.

**Circular binary segmentation**

One idea is to operate segmentation of the data → we can divide the data into more segments and compare the coverage between the segments → in this way we can understand if the segmentation is correct or not → we then proceeded to try to get more segments as possible → process of segmentation, We stop when we do not find significant segments anymore or when we arrive to a granularity high enough (=granularity=number of segments).

**Visualizing CNVs**

1. **IGV**

By writing data as a table, it is possible to export it to IGV as a seg file. The mean log2 R-value for each sample projected on the reference genome coordinate is visible. If blue it represents deletion, while if red amplification. A consensus for a specific region can be built.

1. **Histogram**

In order to better visualize the distribution of log2 R segments in the sample a histogram can be built. A peak in 0 is expected as the majority of the samples have the same copy number. The more signal, the more secondary peaks are expected for deletions and amplification.

1. **Browsing a CNV across many individuals**

For browsing a CNV across many individuals the getSegments function can be used. A shift according to the reference is expected for germline variants. In the case of a rare CNV, a clearly higher zero peak is expected, while different peaks with similar heights indicate a pretty common CNV.

**Applying CBS**

In order to apply CBS data needs to be cleaned and the log2 R needs to be computed. This can be done on the coverage of the sample and the reference coverage once they have been normalized. Some tools include all the necessary frameworks, while others only modules. For example, VarScan can take as input mpileup | copynumber and then applies copyCaller.

1. **RCode\_DNAcopy.R**

To generate the data necessary for teh second part of R script we need to enter in teh folder Data -> cd Data -> Here we have two files: normal.bam and a tumor.bam, both already filtered to contain data on chr 13 and 20 and sorted.

1. **Somatic copy number**

Generate a pileup of the files-> one column of teh pileup is the coverage -> we are just making teh coverage available for the tool we are using after -> we are extracting teh info of the coverage

samtools mpileup -q 1 -f ../Annotations/human\_g1k\_v37.fasta Control.sorted.realigned.recalibrated.debup.bam Tumor.sorted.realigned.recalibrated.debup.bam | java -jar ../Tools/VarScan.v2.3.9.jar copynumber --output-file SCNA --mpileup 1

We use VarScan and explicit the copynumber tool -> to transform teh coverage in Copy Numbers

java -jar ../../Tools/VarScan.v2.3.9.jar copyCaller SCNA.copynumber --output-file SCNA.copynumber.called

# we can see the number of the regions that have a specific call: amplification, deletion, neutral. Plus we get the number of called that passed the thresholds

# teh file contains the agjusted\_logratio -> adjusted on the Copy number

# the files now have a column with the call association for eeach read

# here we are trying to call an event for each part of teh genome, we are not oerating a segmentation -> we are just analyzing the log2 ratio.

# We can use this info to perform Circular Segmentation!

1. **Run R script with DNAcopy segmentation**

Rscript CBS.R

## 10) IGV

## 11) Purity Ploidy estimation

CLONET METTI I GRAFICI

## 12) SPIA

NB FAI LA SOMATIC VARIANT CALLING -> PER GLI SNP SOLO SU NORMAL E PER SNV SU TUMOR -> COSI CONTROLLI CHE GLI SNV TROVATI NEL TUMOR NON FOSSERO GIA PRESENTI COME SNP