CdLM “Bioinformatics for Computational Genomics” 2023-2024
ELIA COVINO and **ARIANNA RIGAMONTI**

GENOMICS COURSE - TRIOS EXAM PROJECT

INTRODUCTION

Rare genetic diseases, occurring in fewer than 1 in 2,000 individuals, stem from genetic mutations inherited from one or both parents, but they can also happen spontaneously. Based on the way they are inherited, rare genetic diseases can be categorized into **autosomal dominant (AD)** and **autosomal recessive (AR)**: in AD only one copy of the mutated gene is sufficient to cause the disease, while AR are caused by mutations in both copies of the gene (mother and father). Next Generation Sequencing (**NGS**) enables rapid, cost-effective genome sequencing and, in particular, whole exome sequencing (**WES**) detects single nucleotide variants (**SNVs**), aiding early diagnosis.

Our project focuses on **10 trios genome** data analysis for rare mendelian diseases. We were assigned the following cases:

Case Number	715	732	667	659	619
Inheritance model	AD	AR	AR	AR	AD

Case Number	728	707	587	700	604
Inheritance model	AD	AR	AD	AD	AR

MATERIALS AND METHODS

The analysis was conducted with a **Unix pipeline** and **Variant Effect Predictor (VEP)** annotation. We developed a shell script named **pipeline.sh** *(code reported in the last pages of this document)* which takes the type of inheritance and case numbers (e.g ./pipeline.sh -AD 715 619 728 587 700 -AR 732 667 659 707 604) as input, hence automating the analysis for multiple genetic disease cases. The script has to be executed on the server (IP: 159.149.160.7) and performs the following steps:

1. **Setting variables**:
   1.1. **`cases_path**="/home/BCG2024_genomics_exam/"`: sets the path to the directory containing the cases.
2. **Checking for empty input**: checks if there are no input arguments provided. If so, it displays an error message and exits the script.
3. **Iterating over command arguments**: loops through each command-line argument provided.
4. **Printing help information**: if the argument is `-h`, it prints usage information about the script.
5. **Setting disease type**:
   5.1. If the current argument is `-AR`, it sets the recessive variable as true.
   5.2. If the current argument is `-AD`, it sets the recessive variable as false.
6. **Checking case number existence**:
   6.1. Checks if the current argument corresponds to an existing case number file.
   6.2. If the file exists, it proceeds to analyze the data for that case.
7. **Current case analysis:**
   7.1. **Creating case directory**: if the case directory does not exist, it creates one with the case number and displays a message indicating the start of analysis for that case.
   7.2. **Moving to case directory:** changes the working directory to the directory for the current case.
   7.3. **Alignment**: performs alignment using Bowtie2 for child, father, and mother sequences.
   7.4. **Variant calling**: calls variants using Freebayes and generates a Variant Call Format (VCF) file.
   7.5. Filtering VCF:
      7.5.1. Filters the VCF file based on the disease type (recessive or dominant). **Recessive** cases have the mutations on both alleles of the gene, so we put the filter “**1/1.*0/1.*0/1**” because each parent is a healthy carrier of the disease (0/1) and the child has the mutation on both alleles (1/1). For dominant cases mutation on just one allele is enough to cause the disease. For the purpose of this analysis we considered **dominant** cases due to *de novo* mutations, so we used the filter “**0/1.*0/0.*0/0**”. Both parents are healthy (0/0) and the child has just one mutated allele (0/1). In a more general context, we could also have considered dominant cases due not to *de novo* mutation but inherited from one parent, who has the disease as well (0/1).

7.5.2. Saves the filtered variants to a new VCF file.

7.6. **UCSC and coverage analysis**:
- 7.6.1. Calculates coverage using bedtools for child, mother, and father BAM files.
- 7.6.2. Generates coverage files for UCSC Genome Browser visualization.

7.7. **FastQC analysis**: performs quality analysis using FASTQC for child, mother and father FASTQ files.

7.8. **Qualimap analysis**: performs quantity analysis using QUALIMAP for child, mother and father BAM files.

7.9. **MultiQC analysis**: generates a combined report using MULTIQC for all files related to the current case.

7.10. **Completion message**: displays a message indicating the completion of analysis for the current case.

8. **Error handling**: if the provided argument is invalid, it displays an error message and exits the script.

RESULTS AND DISCUSSION

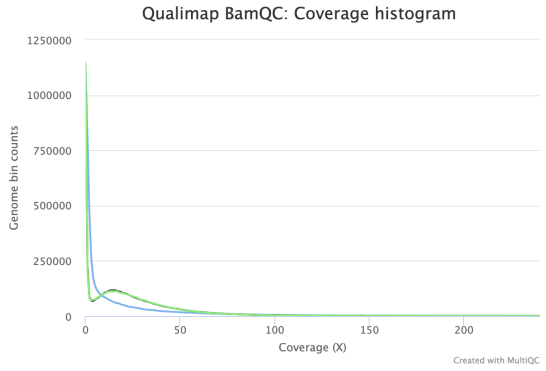
QUALITY CONTROL:

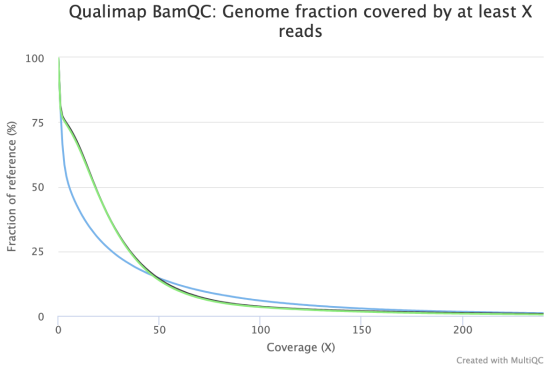
Data quality assessment was performed using MultiQC. As an example here is one of the **MultiQC** (case 587), with the respective **FastQC** and **Qualimap.**

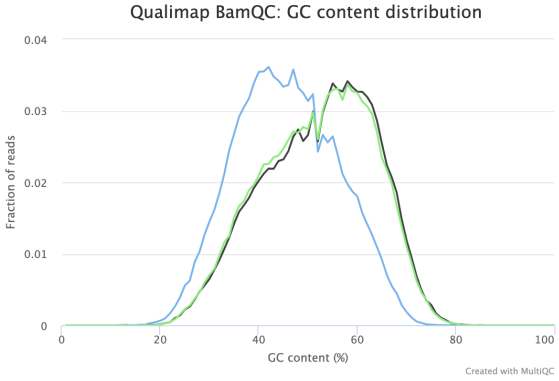
General Statistics

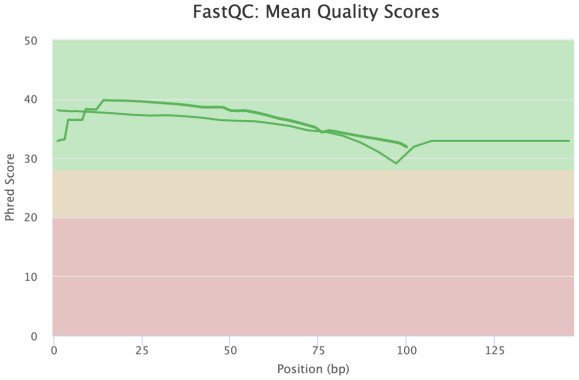
Copy table | Configure Columns | Plot | Showing 3/3 rows and 8/18 columns.

Sample Name	% GC	≥ 30X	Median cov	Mean cov	% Aligned	% Dups	% GC	M Seqs
case587_child	46%	22.8%	5.0X	24.1X	99.8%	5.4%	43%	3.0
case587_father	52%	31.2%	18.0X	27.3X	99.9%	6.1%	50%	2.2
case587_mother	52%	31.0%	18.0X	26.3X	99.8%	8.5%	50%	2.1









The **coverage histogram** provides information on the coverage across the genome and in our cases shows a relatively good distribution. The **genome fraction covered by at least X reads** provides information on the proportion of the genome that meets a certain coverage threshold. In our analysis a substantial portion of the genome has sufficient coverage for reliable variant calling. The **GC content distribution** describes the distribution of GC content across the genome and in our data the sequencing exhibits a relatively uniform GC content across the genome (except for a *warning* in the child case). The **mean quality score** provides an overall assessment of the sequencing data quality, reflecting the average base call accuracy across all reads. We obtained a high mean quality score for mother, father and child, suggesting that the majority of bases in the sequencing data have high confidence in their accuracy, supporting reliable downstream analysis such as variant calling.

VARIANT CALLING:

The output of our shell script is a **.vcf** file (one for each case) that we uploaded to *Variant Effect Predictor (VEP)* for Human GRCh37. VEP is a tool used to interpret genetic variations, particularly single nucleotide variants (SNVs) and insertion/deletion variants (indels) in human genomes and other organisms. It can indicate whether a particular variant is located in a coding region of a gene (which could alter the encoded protein), in a regulatory region (which could influence gene expression), or in a non-coding region (which may not have a notable impact).

The selected **parameters** in uploading the VCF file on the VEP website are:

- **Transcript:** Refseq
- **Allele Freq data**: 1000 genomes global, gnomAD
- **Additional notations**: phenotypes
- **Pathogenicity predictions**: prediction + score, SIFT + PolyPhen

We then applied the following **filters** to look for variants that cause rare mendelian autosomal disease:

- **Impact is not LOW**: i.e is MODERATE or HIGH, suggesting that the variant is likely to alter the function of the encoded protein.
- **Impact is not MODIFIER**: the variant is expected to have a notable effect on the function of the encoded protein.
- **Phenotype or disease is DEFINED**
- **SIFT <= 0.2**: SIFT stands for Sorting Intolerant from Tolerant and it is used to predict whether an amino acid substitution in a protein will have an impact on its function. If a substitution is predicted to be deleterious, it is assigned a low SIFT score, typically below a certain threshold, in our case 0.2.
- **PolyPhen > 0.6**: PolyPhen (Polymorphism Phenotyping) is another tool used to predict the functional effects of genetic variants, particularly amino acid substitutions, by considering various features such as sequence homology, protein structure, and physicochemical properties. A PolyPhen score above 0.6 suggests a high probability that the variant is damaging or deleterious.

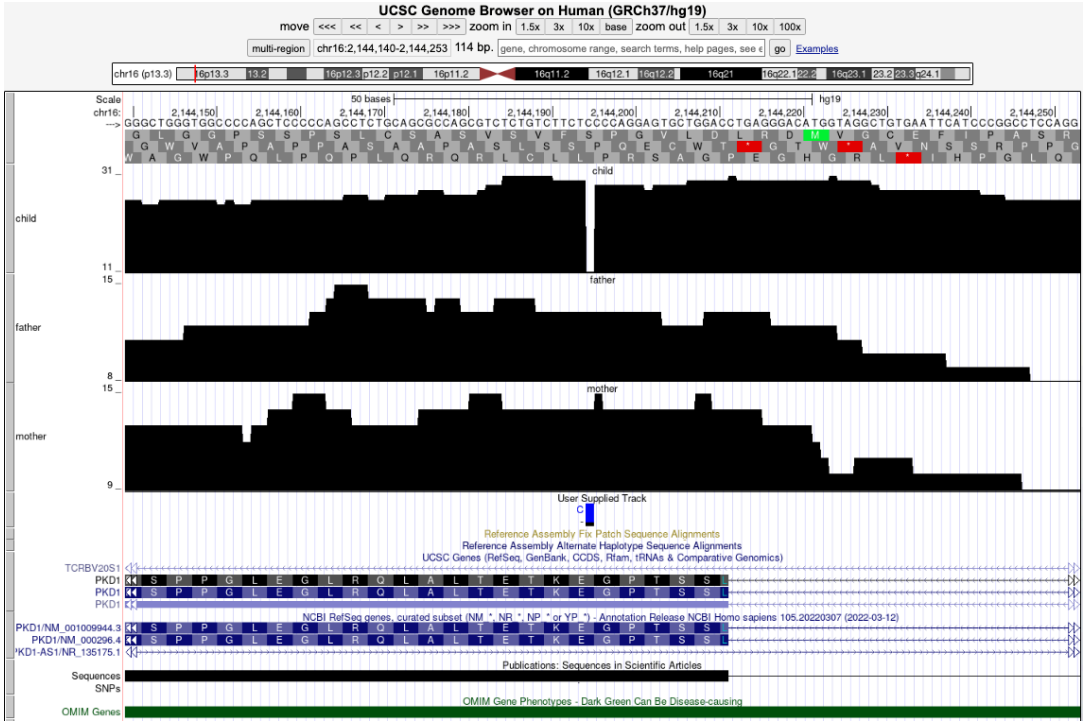
We finally considered rare Mendelian diseases if the allele frequency **AF** and **gnomADe AF** are **lower than 10^{-4} or '-'** (absent), obtaining the following diagnosis:

Case Number	Inheritance model	Location	Consequence	Gene	Allele	Disease
715	AD	16:2140953-2140953	stop_gained	PKD1 (5310)	A	Autosomal dominant polycystic kidney disease
732	AR	16:89858416-89858416	stop_gained	FANCA (2175)	A	Fanconi anemia
667	AR	16:53705451-5370545	frameshift_variant	RPGRIP1L (23322)	AA	Joubert syndrome; Meckel-Gruber syndrome
659	AR	16:88889041-88889043	frameshift_variant	GALNS (2588)	A	Mucopolysaccharidosis, MPS-IV-A
619	AD	16:51174260-51174260	stop_gained	SALL1 (6299)	A	Townes-Brocks syndrome 1

Case Number	Inheritance model	Location	Consequence	Gene	Allele	Disease
728	AD	-	-	-	-	(healthy)
707	AR	16:3639684-3639684	stop_gained	SLX4 (84464)	A	Fanconi anemia
587	AD	16:2144194-2144200	frameshift_variant	PKD1 (5310)	C	Autosomal dominant polycystic kidney disease
700	AD	16:89350120-89350132	frameshift_variant	ANKRD11 (29123)	TCTC	KBG syndrome
604	AR	16:88923285-88923285	start_lost	GALNS (2588)	A	Mucopolysaccharidosis, MPS-IV-A

COVERAGE:

We finally uploaded the **candilistTG.vcf** file and **.bg** files of mother, father and child for every case in **UCSC Genome Browser** in order to check sequencing coverage and the location of the variant. As an example we report *case 587*, where the child has been diagnosed with a frameshift variant starting from the nucleotide C, causing *Autosomal dominant polycystic kidney disease*.



pipeline.sh

```
#!/usr/bin/env bash

cases_path="/home/BCG2024_genomics_exam/"
recessive=false

# check empty case
if [ ${#@} == 0 ]; then
   echo "ERROR: no input arguments"
   exit
fi

# iterate over command arguments
for arg in "$@"; do
   # print help info
   if [ "$arg" = "-h" ]; then
       echo
       echo "pipeline.sh executable by Elia Covino & Arianna Rigamonti"
       echo
       echo "./pipeline.sh [-AR recessive disease case numbers] [-AD dominant disease case numbers] [-h help]"
       echo
       echo "example: ./pipeline.sh -AR 452 453 454 -AD 703 704 705"
       echo


       exit

   # set recessive disease
   elif [ "$arg" = "-AR" ]; then
       recessive=true
   # set non-recessive disease (dominant)
   elif [ "$arg" = "-AD" ]; then
       recessive=false

   # check if the case number exists
   elif [ -f ${cases_path}"case"${arg}"_child.fq.gz" ]; then
       case_number=${arg}

       if [ ! -d "case"${case_number} ]; then
           mkdir "case"${case_number}
           echo "Running case $case_number..."
       else
           echo "Already present case $case_number."
           continue
       fi
```

```
44.
45.         cd case${case_number}
46.
47.         # ALIGNING
48.         echo "Aligning..."
49.         echo "Bowtie2 -child..."
50.         bowtie2 -U ${cases_path}case${case_number}_child.fq.gz -p 8 -x ${cases_path}uni --rg-id 'SC' --rg
    "SM:child" |         samtools view -Sb | samtools sort -o case${case_number}_child.bam
51.         echo "Bowtie2 -father..."
52.         bowtie2 -U ${cases_path}case${case_number}_father.fq.gz -p 8 -x ${cases_path}uni --rg-id 'SF' --rg
    "SM:father" | samtools view -Sb | samtools sort -o case${case_number}_father.bam
53.         echo "Bowtie2 -mother..."
54.         bowtie2 -U ${cases_path}case${case_number}_mother.fq.gz -p 8 -x ${cases_path}uni --rg-id 'SM' --rg
    "SM:mother" | samtools view -Sb | samtools sort -o case${case_number}_mother.bam
55.         echo "Freebayes..."
56.         freebayes -f ${cases_path}universe.fasta -m 20 -C 5 -Q 10 --min-coverage 10
    case${case_number}_mother.bam case${case_number}_child.bam case${case_number}_father.bam  >
    case${case_number}.vcf
57.
58.         bcftools query -l case${case_number}.vcf | sort > vcf.sort.temp
59.         bcftools view -S vcf.sort.temp case${case_number}.vcf > case${case_number}.sorted.vcf
60.         grep "^#CHR" case${case_number}.sorted.vcf
61.
62.         echo "grep # ..."
63.         grep "#" case${case_number}.sorted.vcf > candilist${case_number}.vcf
64.
65.         diseaseFilter="1/1.*0/1.*0/1" # recessive
66.         if  [ $recessive != true ]; then
67.             diseaseFilter="0/1.*0/0.*0/0" # dominant
68.         fi
69.
70.         grep ${diseaseFilter} case${case_number}.sorted.vcf  >> candilist${case_number}.vcf
71.
72.         grep "#" candilist${case_number}.vcf > ${case_number}candilistTG.vcf
73.
74.         bedtools intersect -a candilist${case_number}.vcf -b
    /home/BCG2024_genomics_exam/exons16Padded_sorted.bed -u >> ${case_number}candilistTG.vcf
75.
76.         # UCSC
77.         echo "bedtools for UCSC..."
78.         echo "bedtools -child"
79.         bedtools genomecov -ibam case${case_number}_child.bam -bg -trackline -trackopts 'name="child"' -max 100
    > case${case_number}childCov.bg
80.         echo "bedtools -mother"
81.         bedtools genomecov -ibam case${case_number}_mother.bam -bg -trackline -trackopts 'name="mother"' -max
    100 > case${case_number}motherCov.bg
82.         echo "bedtools -father"
83.         bedtools genomecov -ibam case${case_number}_father.bam -bg -trackline -trackopts 'name="father"' -max
    100 > case${case_number}fatherCov.bg
84.
85.         # FASTQC
86.         echo "fastqc..."
87.
88.         echo "fastqc -child"
89.         fastqc ${cases_path}case${case_number}_child.fq.gz -o ./
90.         echo "fastqc -mother"
91.         fastqc ${cases_path}case${case_number}_mother.fq.gz -o ./
92.         echo "fastqc -father"
93.         fastqc ${cases_path}case${case_number}_father.fq.gz -o ./
94.
95.         # QUALIMAP
96.         echo "qualimap..."
97.         qualimap bamqc -bam case${case_number}_child.bam -gff
    /home/BCG2024_genomics_exam/exons16Padded_sorted.bed -outdir case${case_number}_child
98.         qualimap bamqc -bam case${case_number}_mother.bam -gff
    /home/BCG2024_genomics_exam/exons16Padded_sorted.bed -outdir case${case_number}_mother
99.         qualimap bamqc -bam case${case_number}_father.bam -gff
    /home/BCG2024_genomics_exam/exons16Padded_sorted.bed -outdir case${case_number}_father
100.
101.        # MULTIQC
102.        echo "multiqc..."
103.        multiqc -f /home/BCG2024_genomics_exam/case${case_number}* ./case${case_number}*
104.
105.        echo "case $case_number DONE!"
106.        echo
107.
108.        cd ../
109.
110.    # error message and exit
111.    else
112.        echo
113.        echo "ERROR: wrong input value: $arg"
114.        echo "incorrect case number or unknown argument (only -r -d -h)"
115.        exit
116.    fi
117. done
```