MSc Data Science & Information Technologies

Bioinformatics - Biomedical Data Science Specialization

Course: Introduction to Bioinformatics

Professors: Martin Reczko - Alexandros Dimopoulos

Introduction to Bioinformatics Final Project (11) GATK Variant Calling Analysis

Name	*Student ID*
Konstantinos Giatras	**7115152300005**

DEPARTMENT OF INFORMATICS & TELECOMMUNICATIONS

HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

2023-2024

Contents

1 Introduction

Research in genome-wide association studies (GWAS) and population genetics typically includes the task of detecting genetic differences among individuals or groups. Variant calling is a key technique used for this purpose, where the goal is to identify genetic variants that differ between individual samples and a reference genome. A popular tool for this task is the Genome Analysis Toolkit (GATK) [8], which offers a comprehensive set of tools for uncovering variants and performing genotyping. In our specific case, we are going to conduct a GATK-based variant calling analysis using exome data from chromosome 11. These samples are sourced from 27 individuals participating in the 1000 Genomes project [2]. Chromosome 11 is notable for harboring numerous genes linked to diseases, particularly various cancers and neurological conditions [1]. Creating a variant callset for this chromosome is significant, since it can be utilized to enhance our understanding of the genetic foundations of these illnesses and potentially identify novel avenues for drug development.

2 Methodology

In the process of detecting germline short variants (SNPs and Indels) against the GRCh38 human reference genome, the approach is divided into three core stages:

1. Pre-Processing
2. Variant Discovery
3. Callset Refinement

The following pipeline, as shown in figure 1 was used as a guideline.

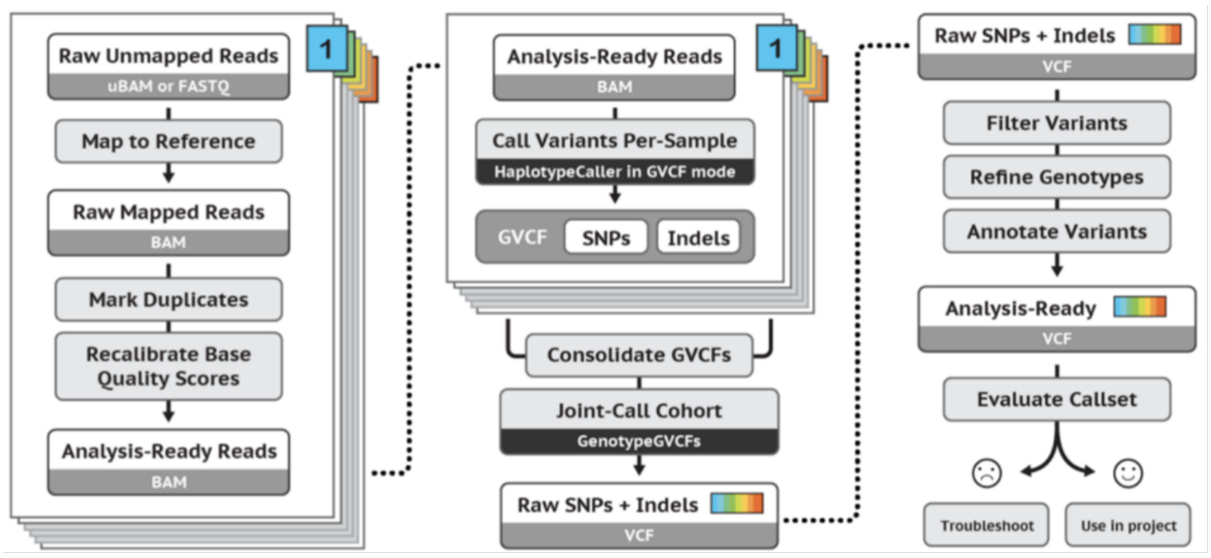


Figure 1: Pipeline for GATK germline variant calling analysis [9]

For the needs of the project, a main folder is created, containing the following sub-folders:

- "files": contains the reference genome and BAM files
- "vcf": will contain the cohort VCF file produced by variant discovery
- "statistics": will contain intermediary statistics for GATK pipeline monitoring
- "scripts": will contain the necessary scripts for executing the analysis

2.1 Pre-Processing

2.1.1 Data Retrieval

A script was constructed for the selective download of chromosome 11 exome alignments from the 1000 Genomes Project via File Transfer Protocol (FTP), targeting the specific range of samples HG00100 to HG00114. The aim was to optimize the number of samples for enhanced accuracy within the constraints of time and computational resources. The script, executed within the "files" directory, pieced together URLs for potential samples from HG00100 up to HG00140 and employed `wget` for retrieval. Absences in the sequence meant some `wget` attempts yielded no files. Despite the intention to download 41 samples, the non-sequential nature of sample numbering resulted in a total of 27 samples, including the 15 samples of interest.

The script (**../scripts/data_retrieval.sh**):

```
# Static parts of the URL
link_prefix="ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/"
link_suffix_1=".chrom11.ILLUMINA.bwa.GBR.exome.20120522.bam"
link_suffix_2=".chrom11.ILLUMINA.bwa.GBR.exome.20121211.bam"
link_middle="/exome_alignment/"

# Static sample prefix
sample_prefix="HG00"

# wget query for all the possible link combinations
for i in $(seq -w 100 140); do
  wget ${link_prefix}${sample_prefix}${i}${link_middle}${sample_prefix}
     ${i}${link_suffix_1}
  wget ${link_prefix}${sample_prefix}${i}${link_middle}${sample_prefix}
     ${i}${link_suffix_2}
done
```

2.1.2 BAM File Quality Control and Reference Genome Retrieval

Initially, the integrity of the BAM files was confirmed using a SAMtools command [5] that assesses the quality of the reads, which reported zero failures:

```
for file in *.bam; do samtools flagstat $file | grep "QC";done
```

To ascertain the reference assembly to which the reads are aligned, another command was employed, yielding the reference assembly's identifier, which was hs375d:

```
for file in *.bam; do echo $file; samtools view -H $file | grep '^@SQ'|
     grep 'UR';done
```

Subsequently, to align the reads with the GRCh38 reference genome, we obtained the reference genome FASTA file from the GATK resource bundle available at `https://console.cloud.google.com/storage/browser/genomics-public-data/resources/broad/hg38/v0/`. Since we are only interested in the genomic area of chromosome 11, we used the following commands to extract only the relevant region from the downloaded reference genome file and generate its `.fai` index file:

```
samtools faidx hg38.fa chr11 > chr11.fa
samtools faidx chr11.fa
```

Additionally, a sequence dictionary was generated, which is crucial for downstream processes:

```
gatk CreateSequenceDictionary R=my_sequence.fa O=my_sequence.dict
```

From the same GATK resource bundle, we also get the following files, and generate their corresponding index files with `tabix`, as we will need them in following steps:

- hapmap_3.3.hg38.vcf.gz
- 1000G_omni2.5.hg38.vcf.gz
- 1000G_phase1.snps.high_confidence.hg38.vcf.gz
- Homo_sapiens_assembly38.dbsnp138.vcf
- Mills_and_1000G_gold_standard.indels.hg38.vcf.gz
- Homo_sapiens_assembly38.known_indels.vcf.gz

2.1.3 Realignment to Reference Genome GRCh38, Chromosome 11

The existing BAM files, already aligned to the hs375d reference genome, require reformatting to FASTQ for realignment to another reference genome. To accomplish this, we execute a script within the "files" directory, which employs `samtools collate` to convert the paired-end BAM files to compressed interleaved FASTQ format. This also shuffles the read order, grouping them by name in the FASTQ file, which is instrumental in mitigating biases introduced during the initial alignment.

The script (**../scripts/convert_bam_to_fastq.sh**):

```
# Loop converting BAM files to fastq format and shuffling reads
echo "Beginning conversion from BAM to fastq"
for file in *.bam; do
  echo "Starting conversion of $file"
  # Use samtools collate to shuffle and convert BAM to fastq, then
     compress with gzip
  samtools collate -uOn 128 "$file" "tmp_${file}" | samtools fastq |
     gzip >  "interleaved_${file%.bam}.fq.gz"
  echo "Conversion of $file complete"
done
```

For the alignment of the chromosome 11 exome FASTQ files to the GRCh38 reference genome, the BWA-MEM version of the Burrows-Wheeler Aligner [7], which is suitable for long reads, was employed. For this purpose, the reference genome needs to be indexed. We use the following command to do that:

```
bwa index chr11.fa
```

Unique read group information from each BAM file, indicating specifics like flowcell details, sequencing method, and DNA preparation library identifier, is used to distinguish between samples in the GATK pipeline. A script was used to extract this read group data from each BAM file and incorporate it into the BWA alignment process using the "-R" parameter. This resulted in sorted BAM files aligned by coordinates. The script, run from the "files" directory, also gathered important metrics for each sample file. These metrics and details will be further discussed in the subsequent section.

The script (**../scripts/alignment.sh**):

```
echo "Beginning alignment"
for file in H*.bam; do
  echo "Starting alignment of $file"
  # Extract the sample name by removing the expected patterns from the
     BAM filename
  samplename=$(echo $file | sed -E 's/.chrom11.ILLUMINA.bwa.GBR.exome
     .(20120522|20121211).bam//')

  # Construct the read group information string with the -R option
  read_group="@RG\tID:${samplename}\tLB:${samplename}\tSM:${samplename
     }\tPL:ILLUMINA\tCN:unknown"

  # Look for the FASTQ files with both possible dates
  fq1=$(ls interleaved_${samplename}.chrom11.ILLUMINA.bwa.GBR.exome
     .20120522.fq.gz 2> /dev/null)
  fq2=$(ls interleaved_${samplename}.chrom11.ILLUMINA.bwa.GBR.exome
     .20121211.fq.gz 2> /dev/null)

  # Select the available FASTQ file
  if [[ -n $fq1 ]]; then
    fastq=$fq1
  elif [[ -n $fq2 ]]; then
    fastq=$fq2
  else
    echo "No FASTQ file found for $samplename"
    continue
  fi

  echo "Running BWA MEM for $samplename with FASTQ file $fastq"
  bwa mem -t 7 -M -R "$read_group" chr11.fa $fastq | samtools view -bS
     - | samtools sort -o "aligned${samplename}.bam"
  echo "Alignment of $file complete"
done

#This script handles the collection of total and mapped reads before
   and after the alignment
touch old_mapped_stats.csv
echo "sample,old bam mapped,old bam reads" >> old_mapped_stats.csv
for file in H*.bam;
do
  sample=${file:0:7}
  previous_unmapped_reads=$(samtools view -F 4 -c $file)
  previous_total_reads=$(samtools view -c $file)
  echo "$sample,$previous_unmapped_reads,$previous_total_reads" >>
     old_mapped_stats.csv
done

touch new_mapped_stats.csv
echo "new bam mapped,new bam reads" >> new_mapped_stats.csv
for file in aligned*.bam;
do
  reference_unmapped_reads=$(samtools view -F 4 -c $file)
  reference_aligned_reads=$(samtools view -c $file)
  echo "$reference_unmapped_reads,$reference_aligned_reads" >>
     new_mapped_stats.csv
done
#paste the files together
paste -d "," old_mapped_stats.csv new_mapped_stats.csv > mapped_stats.
   csv
```

2.1.4 Mark Duplicates

Prior to analyzing the mapped BAM files, it is essential to handle duplicate reads, which often emerge from PCR duplication during library construction or sequencing. These duplicates need marking to prevent them from skewing results in later GATK pipeline stages. GATK's `MarkDuplicates` tool is used for this purpose. A script, run from the "files" directory, is utilized to apply this tool on each aligned BAM file. This process also generates important metrics, such as the duplication percentage, to inform the quality of the dataset.

The script (**../scripts/mark_duplicates.sh**):

```
# Locate and tag duplicate reads in the aligned bam files, as well as
   create an index bam file
for file in aligned*.bam; do
```

```
  echo "Beginning duplicate marking of $file"
  gatk MarkDuplicates \
    I="$file" \
    O="marked_dup_${file}" \
    M="marked_dup_metrics_${file%.bam}.txt" \
    CREATE_INDEX=true
  echo "Duplicate marking of $file complete"
done

# Create a file to store duplication percentage
echo "Sample,Duplication_Percentage" > duplicate_percentage.csv
for file in marked_dup_metrics_*.txt; do
  sample_name=$(basename "$file" .txt | sed 's/marked_dup_metrics_//')
  duplication_percentage=$(awk 'NR>7 {print $9}' "$file")
  echo "$sample_name,$duplication_percentage" >> duplicate_percentage.
     csv
done
```

2.1.5 Base Quality Score Recalibration (BQSR)

The next stage in the workflow is the recalibration of base quality scores, crucial for the precision of downstream analysis. The base quality scores are assigned to each base by the sequencing machine and represent the confidence in the base call. However, the scores can be overconfident or under-confident, which can lead to errors in downstream analysis. To recalibrate the scores, the GATK Base Quality Score Recalibration (BQSR) tool is used, which consists of two main steps:

1. The first step is to create recalibration tables for each BAM file using the `gatk BaseRecalibrator` command. This tool analyzes various features of the bases in the file, such as read group, machine-assigned quality score, machine cycle, and the previous base. It excludes known variants, which are identified via an input VCF file.
2. The second step is to adjust the quality scores of each base in the reads using the recalibration tables generated by the first tool. This is carried out by the `gatk ApplyBQSR` command.

The process involves running scripts from the "files" directory, employing dbSNP variants of GRCh38 for reference as known variation sites for the BQSR tool, and generating CSV files with recalibration metrics using the `AnalyzeCovariates` GATK tool.

The script (**../scripts/bqsr.sh**):

```
# First pass - Creating recalibration tables for each marked duplicate
    BAM file
for file in marked_dup_aligned*.bam; do
  echo "Creating recalibration table corresponding to $file"
  recal_table="recal_${file%.bam}.table"
  gatk BaseRecalibrator \
    -R chr11.fa \
    -I "$file" \
    --known-sites Homo_sapiens_assembly38.dbsnp138.vcf \
    -O "$recal_table"
  echo "Recalibration table created for $file"
done

# Data CSV retrieval for each recalibration table
for table in recal_*table; do
  echo "Analyzing covariates for $table"
  csv_file="../statistics/${table%.table}.csv"
  gatk AnalyzeCovariates \
    -bqsr "$table" \
    -plots "$csv_file"
  echo "Covariate analysis complete for $table, CSV file created at
     $csv_file"
done

# Second pass - Applying recalibration to each marked duplicate BAM
   file
for file in marked_dup_aligned*.bam; do
  echo "Applying recalibration to $file"
  recal_table="recal_${file%.bam}.table"
  output_bam="recalibrated_${file}"
  gatk ApplyBQSR \
    -R chr11.fa \
    -I "$file" \
    --bqsr-recal-file "$recal_table" \
    -O "$output_bam"
  echo "Recalibration applied to $file"
done
```

The pre-processing step as a whole resulted in BAM files that are base score recalibrated, duplicates marked, and aligned to the GRCh38 reference, making them ready for further analysis in the GATK pipeline.

2.2 Variant Discovery

In this phase of germline variant calling, the objective is to spot DNA sequence variations between the sample reads in the BAM files and the GRCh38 reference genome [6]. Unlike somatic variant calling, which compares variants to a related tissue of the same individual, germline variant calling focuses on identifying genetic variations inherent in an individual's germline. These variations include:

- Single Nucleotide Polymorphisms (SNPs), which are genetic variations where alternative alleles at the same loci occur in at least 1% of the population
- Insertions & Deletions (INDELs) of bases

2.2.1 HaplotypeCaller

The GATK HaplotypeCaller tool is instrumental in this phase, utilizing a statistical approach to evaluate the probability of various genotypes at each genomic locus. It accounts for sequencing errors and the presence of genetic variations in only some of the reads, which is crucial for avoiding false positive variant calls. HaplotypeCaller applies local *de novo* assembly to create haplotypes near each variant, enhancing its ability to differentiate real genetic variants from sequencing errors or artifacts.

In our study, we employed HaplotypeCaller on chromosome 11 in genomicVCF (GVCF) mode. The generated GVCF files list all loci, not just the variant ones, preparing them for merging with other samples for a comprehensive joint analysis later on. The execution of this procedure was carried out from the "files" directory. The script also gathers and consolidates key metrics from the GVCFs using the "`stats`" function from bcftools, integrating them into a CSV file. These compiled metrics cover all 27 samples and are detailed in the results section.

The script (**../scripts/HaplotypeCaller.sh**):

```
# Creation of GVCF files using GATK HaplotypeCaller
for file in recalibrated_*.bam; do
  echo "Starting creation of GVCF for $file"
  output_vcf="../vcf/${file##*/}"
  output_vcf="${output_vcf%.bam}.g.vcf"
  gatk HaplotypeCaller \
    -R chr11.fa \
    -I "$file" \
    -O "$output_vcf" \
    --native-pair-hmm-threads 7 \
    -ERC GVCF \
    -L chr11
  echo "Created GVCF for $file"
done

# Collection of summary statistics from the GVCF files and appending to
    a single CSV file
stats_file="../statistics/HC_stats.csv"

echo "Sample,number of samples,number of records,number of no-ALTs,
   number of SNPs,number of MNPs,number of indels,number of others,
   number of multiallelic sites,number of multiallelic SNP sites" > "
   $stats_file"

for file in ../vcf/*.g.vcf; do
  sample_name=$(basename "$file" .g.vcf)
  echo "Processing $sample_name"
  stats=$(bcftools stats "$file" | grep "^SN" | head -n 9 | awk -F':' '
     {printf("%s,", $2)}' | sed 's/,$//')
  echo "$sample_name,$stats" >> "$stats_file"
done
```

2.2.2 Consolidate GVCFs

The GenomicsDBImport tool is responsible for merging individual GVCF files from each sample into a collective database directory. A sample cohort map in the form of "`$sample\t$file`", which lists each sample in a new row, is generated and used as the input for creating this database. Similar to previous steps, the script for this process is run from the "files" directory.

The script (**../scripts/GenomicsDBImport.sh**):

```
# Create the sample map for GenomicsDBImport
cohort_sample_map="../vcf/cohort.sample_map"
touch "$cohort_sample_map"

# Loop through the g.vcf files in the 'vcf' directory to create the
   sample map
for file in ../vcf/*.g.vcf; do
  sample=$(basename "$file" .g.vcf)
  echo -e "$sample\t$file" >> "$cohort_sample_map"
done

# Execute GenomicsDBImport
gatk GenomicsDBImport \
  --genomicsdb-workspace-path ../database \
  --sample-name-map "$cohort_sample_map" \
  --reader-threads 7 \
  -L chr11
echo "GenomicsDBImport complete"
```

2.2.3 Genotype GVCFs

Once the database containing variant data from the HaplotypeCaller is prepared, we proceed to genotype the samples. Utilizing the "`GenotypeGVCFs`" tool from GATK enables the merging of these files into a consolidated VCF file that encompasses multiple samples. This method improves the precision and accuracy of variant calling by integrating data from the entire sample set and considering the genetic variation within the group. The genotyping phase is executed on the GVCF files stored in the database by running a script from the "files" directory.

The script (**../scripts/GenotypeGVCFs.sh**):

```
# GenotypeGVCFs
echo "Running GenotypeGVCFs"
gatk GenotypeGVCFs \
  -R chr11.fa \
  -V gendb://../database \
  -O ../vcf/chr11_cohort.vcf \
  -L chr11
echo "GenotypeGVCFs complete"

# Retrieving metrics for the cohort VCF
cohort_stats_file="../statistics/cohort_stats.csv"
echo "cohort_file,number of samples,number of records,number of no-ALTs
    ,number of SNPs,number of MNPs,number of indels,number of others,
    number of multiallelic sites,number of multiallelic SNP sites" > "
    $cohort_stats_file"

echo "Processing chr11_cohort.vcf"
stats=$(bcftools stats ../vcf/chr11_cohort.vcf | grep "^SN" | head -n 9
     | awk -F':' '{printf("%s,", $2)}' | sed 's/,$//')
echo "chr11_cohort.vcf,$stats" >> "$cohort_stats_file"
echo "Metrics for chr11_cohort.vcf saved to $cohort_stats_file"
```

2.3 Callset Refinement

2.3.1 Variant Quality Score Recalibration

Upon generating a joint cohort of 27 samples featuring initial variant calls, it is common to encounter errors. To discern true variants from these inaccuracies with high precision, we employ the GATK Variant Quality Score Recalibration (VQSR) process. This tool builds a model that recalibrates variant quality scores by training on a selected dataset using `VariantRecalibrator`, followed by the filtering of the VCF variants into "passed" or "failed" using `ApplyVQSR`. We have delineated the model training and application process for both SNPs and INDELs, producing an output file that is refined for both types.

For SNPs, we train our model using data from the HapMap [3] and 1000 Genomes projects and evaluate it against DbSNP entries. The model refines scores by examining several metrics for the cohort's SNPs, including:

- Quality by Depth (QD): Evaluating the quality of the variant call in relation to the sequencing depth
- Mapping Quality (MQ): The root mean square of the mapping quality across all reads
- MQRankSum: Comparing mapping quality differences between alleles
- ReadPosRankSum: Assessing positional differences of the alternate allele within reads
- Fisher's Strand (FS): Measuring strand bias in allele calls
- Strand Odds Ratio (SOR): Evaluating the disparity in allele representation between forward and reverse strands

For INDELs, the process mirrors that of SNPs but differs in the training resources. We utilize validated INDELs from the 1000 Genomes project and pertinent DbSNP INDELs for assessment.

The script run in the "files" directory implements the above VQSR steps and compiles the model's evaluation metrics. It is pivotal to recognize that this process is not one-size-fits-all. The weighting of prior parameters and the calibration of the "`-truth-sensitivity-filter-level`" (set to 99.0 here to balance the identification of high-quality variants against the reduction of false positives) can be adjusted to fine-tune results according to the unique demands of the dataset and the research objectives.

The script (**../scripts/vqsr.sh**):

```
# Beginning SNP Filtering
echo "Beginning SNP Filtering"
gatk VariantRecalibrator \
  -R chr11.fa \
  -V ../vcf/chr11_cohort.vcf \
  --resource:hapmap,known=false,training=true,truth=true,prior=15.0
     hapmap_3.3.hg38.vcf.gz \
  --resource:omni,known=false,training=true,truth=false,prior=12.0 1000
     G_omni2.5.hg38.vcf.gz \
  --resource:1000G,known=false,training=true,truth=false,prior=10.0
     1000G_phase1.snps.high_confidence.hg38.vcf.gz \
  --resource:dbsnp,known=true,training=false,truth=false,prior=2.0
     Homo_sapiens_assembly38.dbsnp138.vcf \
  -an QD -an MQ -an MQRankSum -an ReadPosRankSum -an FS -an SOR \
  -mode SNP \
  -O ../vcf/SNP.recal \
  --tranches-file ../statistics/SNP.tranches \
  --rscript-file ../statistics/SNP.plots.R

# Beginning INDEL Filtering
echo "Beginning INDEL Filtering"
gatk VariantRecalibrator \
  -R chr11.fa \
  -V ../vcf/chr11_cohort.vcf \
  --resource:mills,known=false,training=true,truth=true,prior=12.0
     Mills_and_1000G_gold_standard.indels.hg38.vcf.gz \
  --resource:indels,known=true,training=false,truth=false,prior=2.0
     Homo_sapiens_assembly38.known_indels.vcf.gz \
  -an QD -an MQ -an MQRankSum -an ReadPosRankSum -an FS -an SOR \
  -mode INDEL \
  -O ../vcf/INDEL.recal \
  --tranches-file ../statistics/INDEL.tranches \
  --rscript-file ../statistics/INDEL.plots.R

# Applying SNP filters
echo "Applying SNP filters"
gatk ApplyVQSR \
  -R chr11.fa \
  -V ../vcf/chr11_cohort.vcf \
  -O ../vcf/SNP_filtered_chr11_cohort.vcf \
  --tranches-file ../statistics/SNP.tranches \
  --recal-file ../vcf/SNP.recal \
  --truth-sensitivity-filter-level 99.0 \
  -mode SNP

# Applying INDEL filters
echo "Applying INDEL filters"
gatk ApplyVQSR \
  -R chr11.fa \
  -V ../vcf/SNP_filtered_chr11_cohort.vcf \
  -O ../vcf/recal_filtered_chr11_cohort.vcf \
  --tranches-file ../statistics/INDEL.tranches \
  --recal-file ../vcf/INDEL.recal \
  --truth-sensitivity-filter-level 99.0 \
 -mode INDEL
```

2.3.2 Relevant Sample Isolation

After filtering the variants, we can now isolate only the relevant samples: HG00100 - HG00114 from the cohort and create a new VCF file, by running the following commands from the 'files' directory:

```
# Create a relevant_samples.txt file
cat << EOF > relevant_samples.txt
recalibrated_marked_dup_alignedHG00100
recalibrated_marked_dup_alignedHG00101
recalibrated_marked_dup_alignedHG00102
recalibrated_marked_dup_alignedHG00103
recalibrated_marked_dup_alignedHG00107
recalibrated_marked_dup_alignedHG00108
recalibrated_marked_dup_alignedHG00109
recalibrated_marked_dup_alignedHG00110
recalibrated_marked_dup_alignedHG00111
recalibrated_marked_dup_alignedHG00113
recalibrated_marked_dup_alignedHG00114
EOF

# Filtering out the relevant samples from the VCF file
echo "Filtering relevant samples from recalibrated VCF"
bcftools view --samples-file relevant_samples.txt ../vcf/
    recal_filtered_chr11_cohort.vcf > ../vcf/
    relevant_recal_filtered_chr11_cohort.vcf
```

2.3.3 Variant Annotation

The last and crucial phase in the GATK variant calling workflow is variant annotation. This step enriches the set of identified variants with critical details, including their genomic position, variant type, and the anticipated impact on gene function. Such enrichment with descriptive information is vital as it equips researchers with the means to assess the biological significance of each variant. This is especially important for pinpointing variants that might cause diseases, influence drug efficacy, or contribute to drug resistance.

After refining the variants through the VQSR process, we employ "`snpEff`" [4] for annotation. "`snpEff`" leverages an extensive database of genomic annotations and employs predictive algorithms to assess the functional implications of the variants. This step is complemented by the generation of annotation metrics, further aiding in the evaluation of variant consequences. We run the following command from the 'vcf' directory:

```
# Running snpEff for variant annotation
echo "Annotating filtered VCF with snpEff"
java -jar ../../../apps/snpEff/snpEff.jar -v -stats ../statistics/
    annot_metrics.html -csvStats ../statistics/annot_metrics.csv hg38
    relevant_recal_filtered_chr11_cohort.vcf >
    annotated_relevant_recal_filtered_chr11_cohort.ann.vcf
```

3 Results

In this chapter, we provide a detailed compilation of statistics derived from the data metrics obtained in the methodology section. Moreover, a significant portion of these statistics has been further processed using Python to facilitate their graphical representation. Each subsection is dedicated to a distinct analytical method or statistical insight and is accompanied by a corresponding figure where necessary.

3.1 Intermediate Step Results

3.1.1 Mapped & Total Reads Before and After Alignment to Reference Genome

Using the '`mapped_stats.csv`' file generated by the alignment script described in the previous chapter, along with the Seaborn Python library, we have compiled the following data:

- The number of reads that were mapped in the original samples (blue)
- The total number of reads present in the original samples (yellow)
- The number of reads that were successfully mapped post-alignment to the GRCh38 reference genome (green)
- The total count of reads following the alignment to the GRCh38 reference genome (red)

The comparative analysis of mapped reads against the total reads reveals a consistent ratio both before and after the alignment process. Furthermore, the quantity of reads remains relatively stable, ranging from around 2.5 million reads for sample HG00127 to over 11 million reads for sample HG00131. Notably, the post-alignment total read count exhibits only minor variations, which can be attributed to the removal of single-end reads or unpaired reads from paired-end sequencing that are flagged as secondary in the Burrows-Wheeler Alignment process. These findings are visually presented in Figure 2.

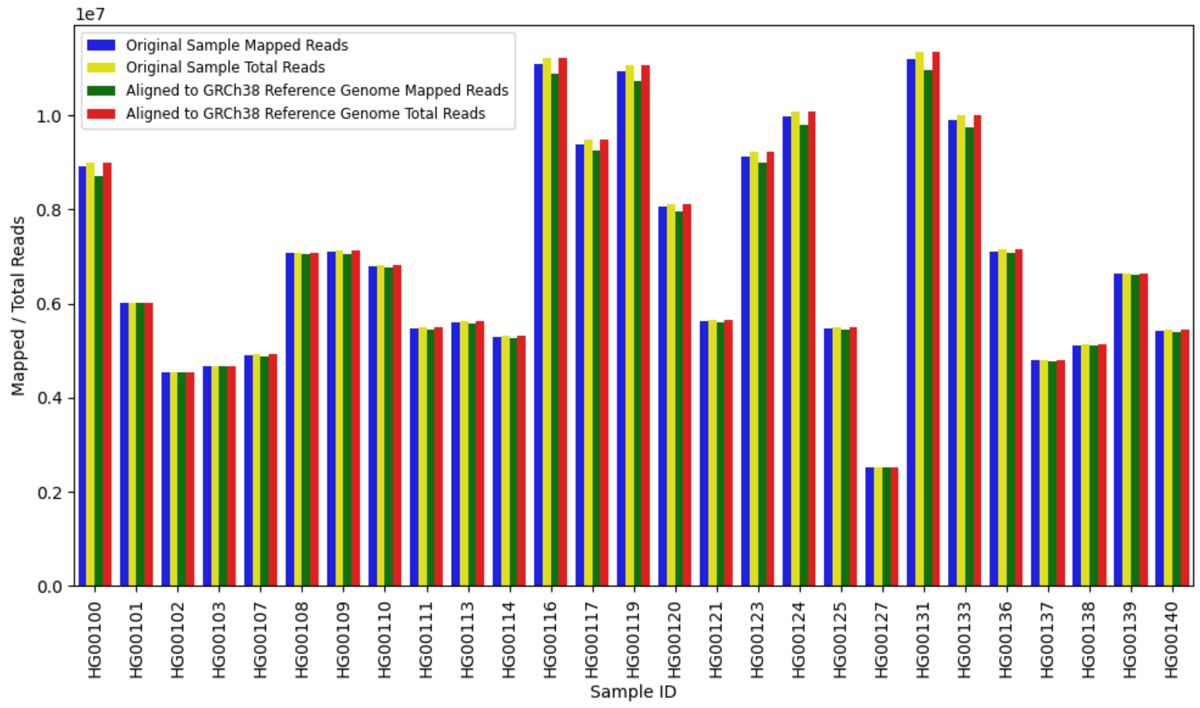


Figure 2: Results from the Alignment of the Samples to the GRCh38 Reference Genome

3.1.2 Percentage of Duplicates per Sample

By examining the '`duplicate_percentage.csv`' file generated from our script in the duplicate marking portion of the methodology and employing the Seaborn Python library, Figure 3 illustrates that the percentage of duplicate reads for all samples ranges between 25% and 40%.

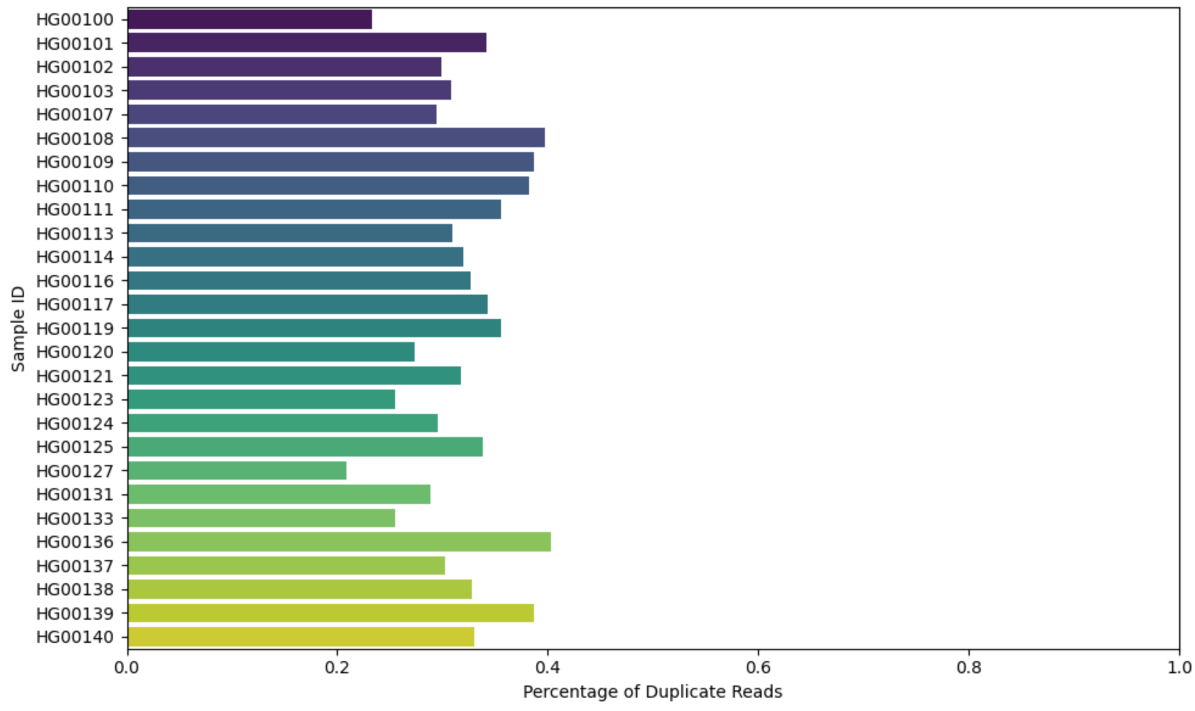


Figure 3: Percentage of Duplicate Reads per Aligned Sample

3.1.3 SNPs & INDELs Identified by HaplotypeCaller

When it comes to the statistics derived from the GVCF files generated by the HaplotypeCaller for each sample, we have consolidated all pertinent data into a single file called '`HC_stats.csv`'. This compilation was facilitated through the script outlined in the corresponding methodology chapter. The CSV file is structured with the following headers: Sample, number of samples, number of records, number of no-ALTs, number of SNPs, number of MNPs, number of indels, number of others, number of multiallelic sites, and number of multiallelic SNP sites. To specifically focus on identifying SNPs and INDELs, we have visually represented these particular CSV columns in Figure 4.

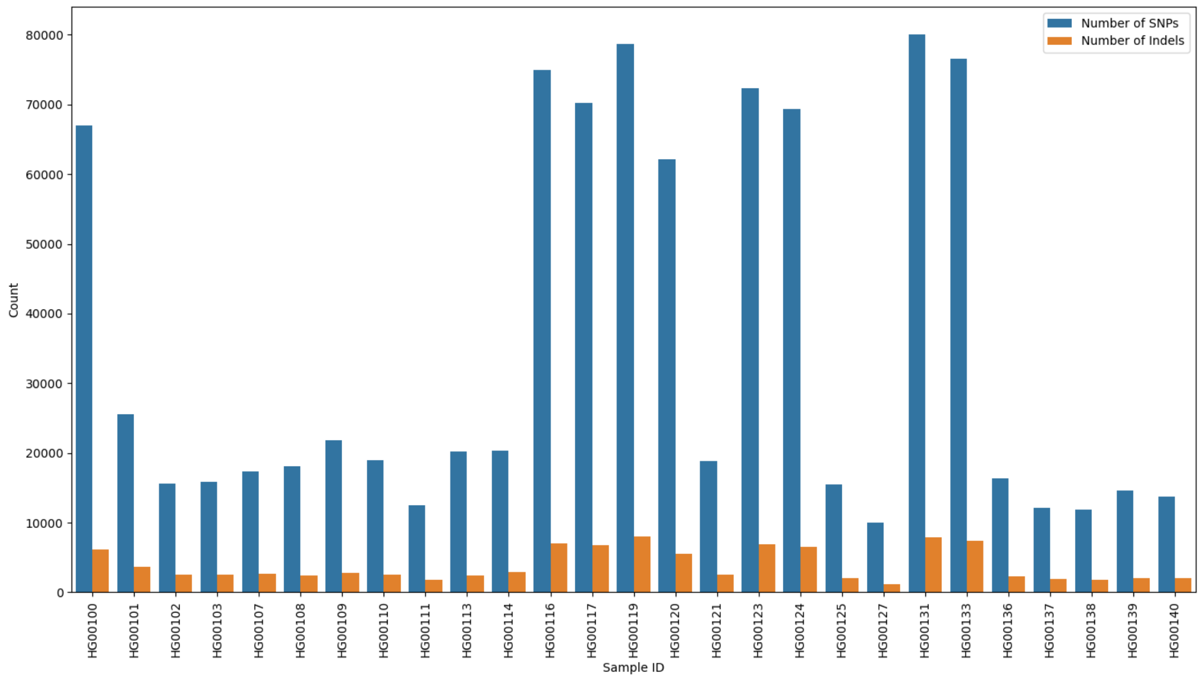


Figure 4: SNPs & INDELs Counts per Sample (HaplotypeCaller)

3.1.4 SNPs & INDELs Count After Cohort Genotyping

The GenotypeGVCFs tool from GATK, as detailed in the methodology chapter, takes into account the genetic variation across the entire cohort when determining variants. Observing the metrics we collected in '`cohort_stats.csv`', it is evident that the number of variants significantly decreases when we shift the scope of variation analysis from a single sample to encompass 27 samples simultaneously:

- Initially, the total number of small variations identified by HaplotypeCaller across all 27 samples amounted to 950,523 SNPs and 104,266 INDELs.
- However, when we performed cohort genotyping, these numbers reduced to 308,061 SNPs and 34,294 INDELs.

This substantial reduction is primarily due to our criteria, which require evidence of variation at specific loci from more than one sample. Moreover, an SNP site identified across all samples is counted as a single SNP site for the cohort.

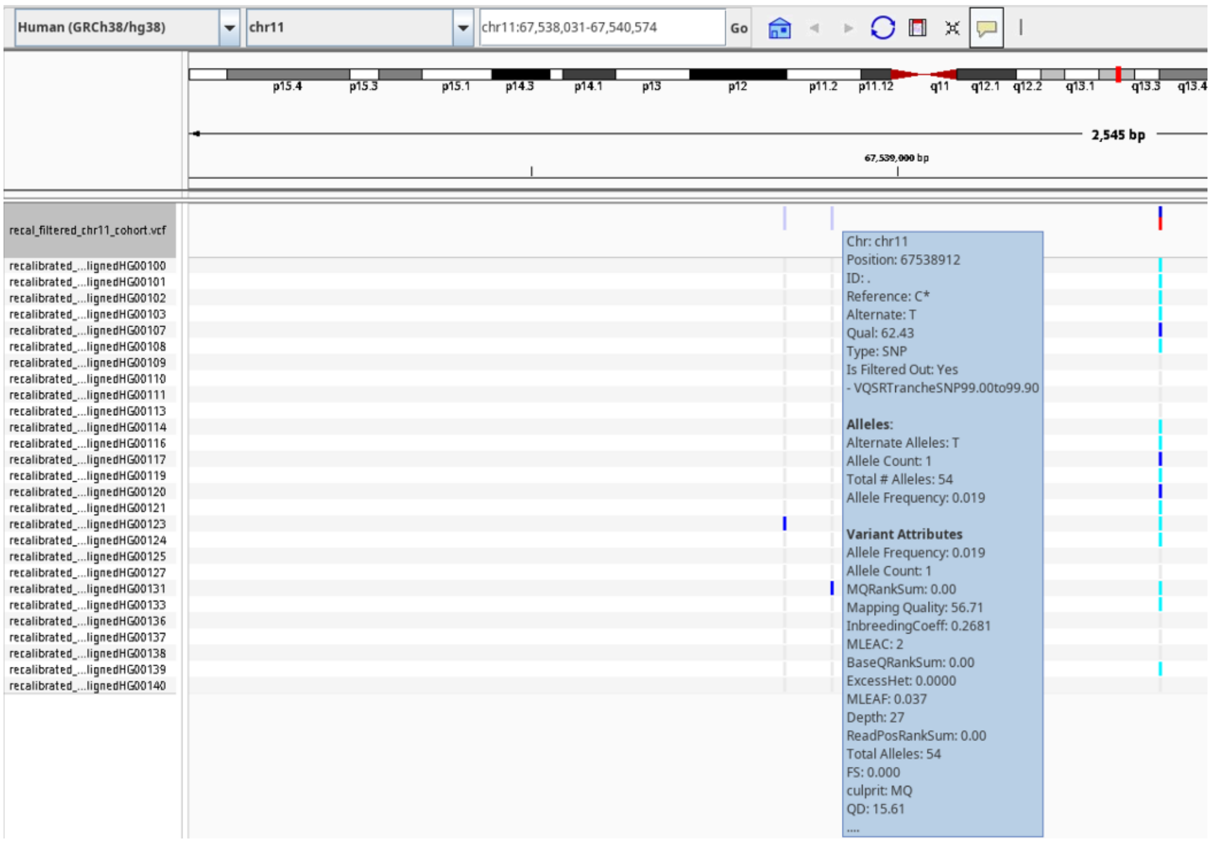
3.1.5 Variant Quality Score Recalibration Results

Using the Integrative Genomics Viewer (IGV), it becomes apparent that, following the application of VQSR (Variant Quality Score Recalibration) to the cohort variant data, there are two additional tags which were not present earlier. These tags are:

- '`Is Filtered Out`', which indicates with a 'yes' or 'no' whether a variant site has failed to meet the criteria set by VQSR filtering
- '`Culprit`', which identifies the primary factor that influenced the filtering decision

For instance, Figure 5 illustrates a variant site at position chr11:67538912 being filtered out due to its Mapping Quality (MQ) score.

Initially, a total of 346,200 SNP and INDEL variants were identified. Following the filtering process, 955 variants were excluded, resulting in 345,245 variants remaining.


Human (GRCh38/hg38)
chr11
chr11:67,538,031-67,540,574
Go
p15.4
p15.3
p15.1
p14.3
p14.1
p13
p12
p11.2
p11.12
q11
q12.1
q12.2
q13.1
q13.3
q13.4
2,545 bp
67,539,000 bp
recal_filtered_chr11_cohort.vcf
recalibrated_...lignedHG00100
recalibrated_...lignedHG00101
recalibrated_...lignedHG00102
recalibrated_...lignedHG00103
recalibrated_...lignedHG00107
recalibrated_...lignedHG00108
recalibrated_...lignedHG00109
recalibrated_...lignedHG00110
recalibrated_...lignedHG00111
recalibrated_...lignedHG00113
recalibrated_...lignedHG00114
recalibrated_...lignedHG00116
recalibrated_...lignedHG00117
recalibrated_...lignedHG00119
recalibrated_...lignedHG00120
recalibrated_...lignedHG00121
recalibrated_...lignedHG00123
recalibrated_...lignedHG00124
recalibrated_...lignedHG00125
recalibrated_...lignedHG00127
recalibrated_...lignedHG00131
recalibrated_...lignedHG00133
recalibrated_...lignedHG00136
recalibrated_...lignedHG00137
recalibrated_...lignedHG00138
recalibrated_...lignedHG00139
recalibrated_...lignedHG00140

```
Chr: chr11
Position: 67538912
ID: .
Reference: C*
Alternate: T
Qual: 62.43
Type: SNP
Is Filtered Out: Yes
- VQSRTrancheSNP99.00to99.90

Alleles:
Alternate Alleles: T
Allele Count: 1
Total # Alleles: 54
Allele Frequency: 0.019

Variant Attributes
Allele Frequency: 0.019
Allele Count: 1
MQRankSum: 0.00
Mapping Quality: 56.71
InbreedingCoeff: 0.2681
MLEAC: 2
BaseQRankSum: 0.00
ExcessHet: 0.0000
MLEAF: 0.037
Depth: 27
ReadPosRankSum: 0.00
Total Alleles: 54
FS: 0.000
culprit: MQ
QD: 15.61
....
```

Figure 5: Example of a VQSR Filtered Out Variation Site

3.2 Final Results

This section discusses the results produced following the successful execution of the GATK variant calling pipeline, complemented by the variant annotation process using `snpEff`.

3.2.1 Cohort Variation Summary Statistics

The results from ’`annot_metrics.csv`’ indicate that, on average, there is a variant every 391 bases across the 345,245 variant sites identified (Table 1). Of these, 89.3% are classified as SNPs, while the remaining 10.7% are INDELs. Notably, a significant 98.526% of the total variants are modifiers, predominantly affecting non-coding regions such as introns and regulatory elements, potentially influencing gene expression or splicing mechanisms (Table 2). A relatively minor portion of the variants, amounting to 0.758%, are responsible for premature transcription termination (Nonsense variations). The bulk of the variants are divided into two main types: Missense variations, constituting 55.504%, which alter amino acid sequences in proteins, and Silent variations, making up 46.739%, which do not affect the protein’s amino acid sequence (Table 3).

Chromosome	Length	Variants	Variants rate
11	135,086,622	345,245	391
Total	135,086,622	345,245	391

Table 1: Variants Rate Details

Type (alphabetical order)	Count	Percent
HIGH	328	0.039%
LOW	6,808	0.805%
MODERATE	5,330	0.63%
MODIFIER	833,452	98.526%

Table 2: Number of Effects by Impact

Type (alphabetical order)	Count	Percent
MISSENSE	5,200	52.504%
NONSENSE	75	0.757%
SILENT	4,629	46.739%

Table 3: Number of Effects by Functional Class

3.2.2 Detailed Count of Effects by Type and Genomic Region

A closer examination of the variation distribution across genomic regions, as shown in the results from '`annot_metrics.csv`', reveals that the majority of variations are located in introns (62.513%) and intergenic regions (21.26%). This pattern aligns with expectations, as these areas, being less subject to evolutionary pressure compared to coding regions, are more prone to accumulating small mutations (Table 4).

Number of Effects by Type			Number of Effects by Region		
Type (alphabetical order)	Count	Percent	Type (alphabetical order)	Count	Percent
3_prime_UTR_variant	12,879	1.519%	DOWNSTREAM	53,208	6.29%
5_prime_UTR_premature_start_codon_gain_variant	332	0.039%	EXON	13,929	1.647%
5_prime_UTR_variant	2,037	0.24%	INTERGENIC	180,306	21.315%
conservative_inframe_insertion	12	0.001%	INTRON	528,297	62.453%
conservative_inframe_deletion	50	0.006%	SPLICE_SITE_ACCEPTOR	98	0.012%
disruptive_inframe_insertion	8	0.001%	SPLICE_SITE_DONOR	39	0.005%
disruptive_inframe_deletion	69	0.008%	SPLICE_SITE_REGION	1,975	0.233%
downstream_gene_variant	53,208	6.274%	TRANSCRIPT	2	0%
frameshift_variant	107	0.013%	UPSTREAM	52,816	6.244%
intergenic_region	180,306	21.26%	UTR_3_PRIME	12,879	1.522%
intragenic_variant	1	0%	UTR_5_PRIME	2,369	0.28%
intron_variant	530,168	62.513%			
missense_variant	5,191	0.612%			
non_coding_transcript_exon_variant	3,950	0.466%			
non_coding_transcript_variant	1	0%			
splice_acceptor_variant	104	0.012%			
splice_donor_variant	44	0.005%			
splice_region_variant	2,092	0.247%			
start_lost	5	0.001%			
stop_gained	75	0.009%			
stop_lost	4	0%			
stop_retained_variant	7	0.001%			
synonymous_variant	4,622	0.545%			
upstream_gene_variant	52,816	6.228%			

Table 4: Number of Effects by Type and Region

3.2.3 SNP Specific Statistics

Regarding the SNP Ts/Tv ratio, which evaluates the proportion of transitions (Ts) to transversions (Tv) in genetic variation data:

- Transitions, which involve the substitution of a purine base for another purine (A to G) or a pyrimidine for another pyrimidine (C to T), constitute the majority of SNPs observed in the cohort, totaling 283,612
- Transversions, characterized by the replacement of a purine with a pyrimidine or vice versa, represent a smaller portion of the SNPs, amounting to 135,013

- The resulting Ts/Tv ratio is 2.100627

	A	C	G	T
A	0	10,907	45,379	11,122
C	23,278	0	12,104	51,502
G	51,476	12,100	0	23,246
T	11,104	45,229	10,901	0

Table 5: Base Changes (SNPs)

3.2.4 Count of Variants per Chromosome Position

Figure 6 illustrates the distribution of variants along the positions on chromosome 11. A specific region on chromosome 11 (chr11: 51,000,000 - 53,000,000) is observed to have no detected variants. However, it is crucial to consider that this lack of variants might be attributed to several factors, such as the impact of the variant quality score recalibration filter or the relatively small sample size of 27 individuals used in this analysis. Consequently, drawing conclusions about this region being under intense evolutionary pressure based solely on this limited dataset would be premature.

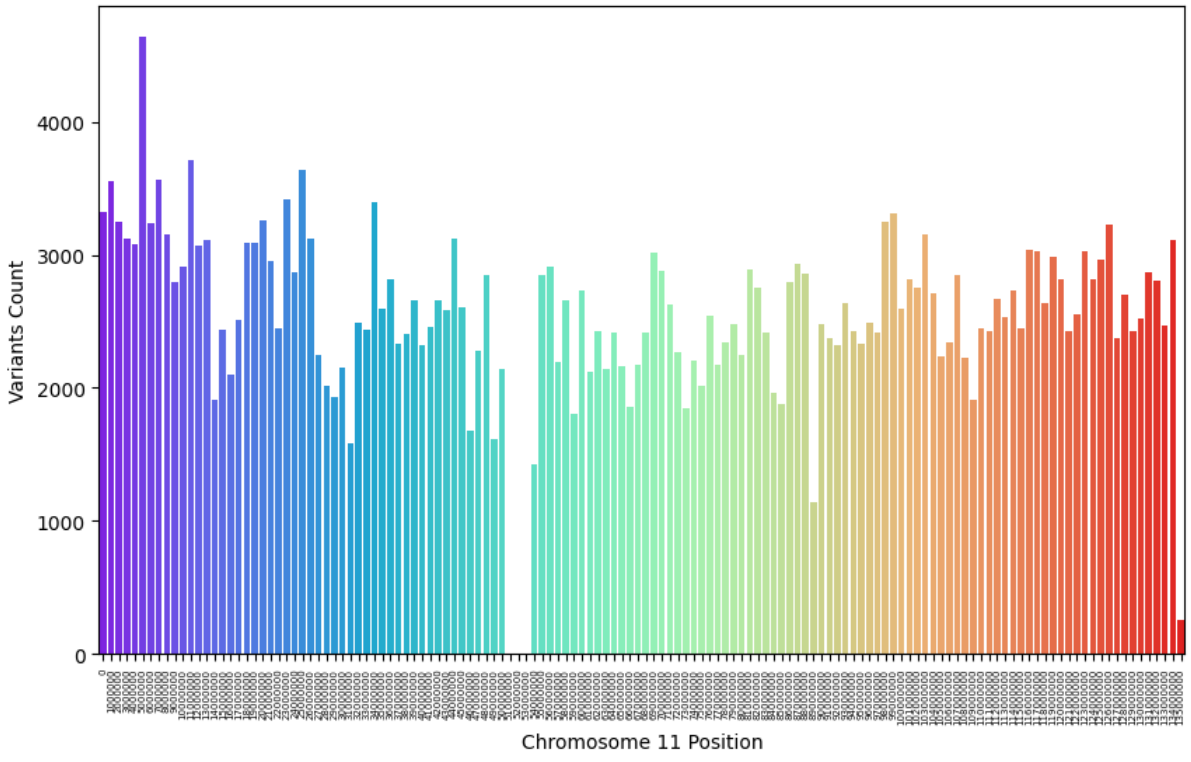


Figure 6: Count of Variants per Chromosome Position

3.2.5 Per Sample Homozygous and Heterozygous Variant Counts

Figure 7 displays the counts of homozygous and heterozygous genotypes across the 15 relevant samples, of which only 11 existed. Heterozygous genotypes are characterized by the presence of both reference and alternative alleles at a given locus. In contrast, homozygous genotypes occur when the same allele is present on both DNA strands. This data is crucial for exploring genetic diversity within the sample population and for pinpointing potential genetic markers that may be associated with specific characteristics or diseases. Generally, homozygous genotypes, especially when linked with an alternative allele that influences the protein product, tend to have a more pronounced effect compared to heterozygous ones.

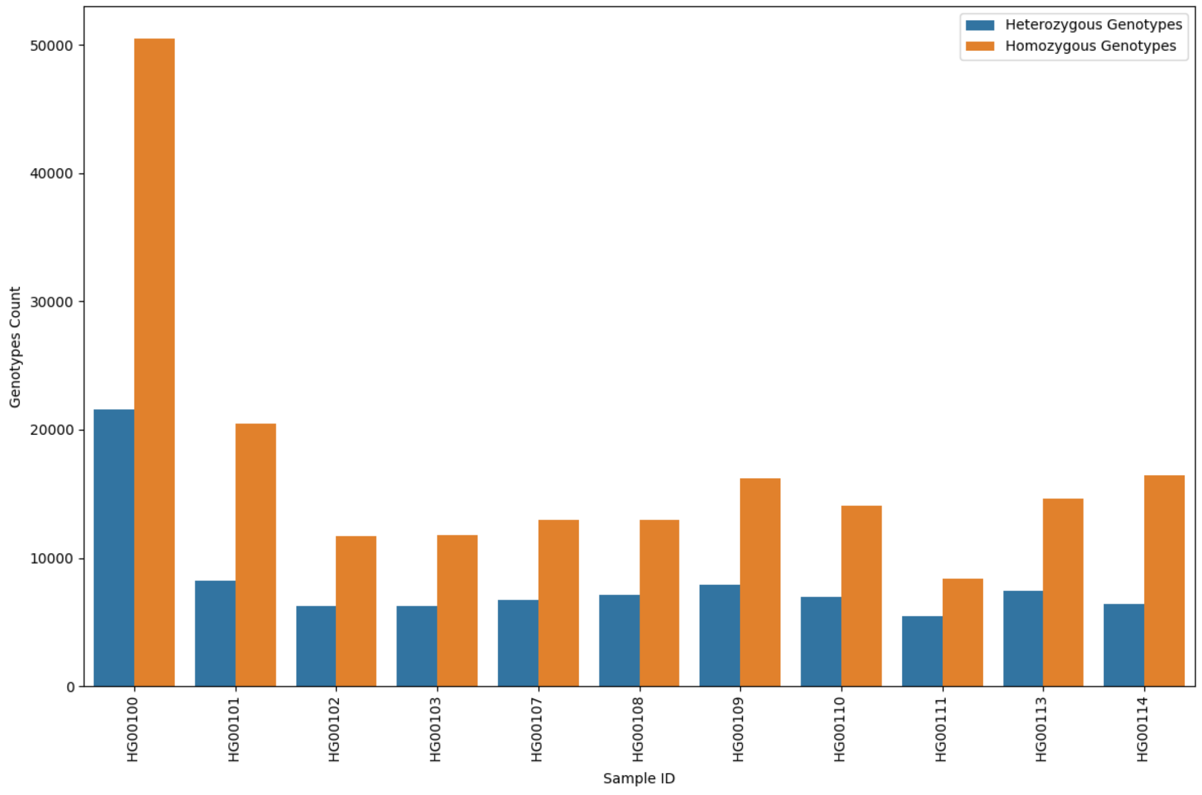


Figure 7: Per Sample Homozygous and Heterozygous Genotypes

References

[1] Chromosome 11-Related Diseases. `https://medlineplus.gov/genetics/chromosome/11/`.

[2] The International Genome Sample Resource. `https://www.internationalgenome.org/`.

[3] The international hapmap project. *Nature*, 426(6968):789–796, December 2003.

[4] Pablo Cingolani, Adrian Platts, Le Lily Wang, Melissa Coon, Tung Nguyen, Luan Wang, Susan J. Land, Xiangyi Lu, and Douglas M. Ruden. A program for annotating and predicting the effects of single nucleotide polymorphisms, snpeff: Snps in the genome of drosophila melanogaster strain w1118; iso-2; iso-3. *Fly*, 6(2):80–92, April 2012.

[5] Petr Danecek, James K Bonfield, Jennifer Liddle, John Marshall, Valeriu Ohan, Martin O Pollard, Andrew Whitwham, Thomas Keane, Shane A McCarthy, Robert M Davies, and Heng Li. Twelve years of samtools and bcftools. *GigaScience*, 10(2), January 2021.

[6] Mark A DePristo, Eric Banks, Ryan Poplin, Kiran V Garimella, Jared R Maguire, Christopher Hartl, Anthony A Philippakis, Guillermo del Angel, Manuel A Rivas, Matt Hanna, Aaron McKenna, Tim J Fennell, Andrew M Kernytsky, Andrey Y Sivachenko, Kristian Cibulskis, Stacey B Gabriel, David Altshuler, and Mark J Daly. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature Genetics*, 43(5):491–498, April 2011.

[7] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows–wheeler transform. *Bioinformatics*, 25(14):1754–1760, May 2009.

[8] Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A. DePristo. The genome analysis toolkit: A mapreduce framework for analyzing next-generation dna sequencing data. *Genome Research*, 20(9):1297–1303, July 2010.

[9] Geraldine A. Van der Auwera, Mauricio O. Carneiro, Christopher Hartl, Ryan Poplin, Guillermo del Angel, Ami Levy-Moonshine, Tadeusz Jordan, Khalid Shakir, David Roazen, Joel Thibault, Eric Banks, Kiran V. Garimella, David Altshuler, Stacey Gabriel, and Mark A. DePristo. From fastq data to high-confidence variant calls: The genome analysis toolkit best practices pipeline. *Current Protocols in Bioinformatics*, 43(1), October 2013.