GNA 5012

Capstone Assessment

Name: Mudra Lad

Student ID: 32858396

3rd November, 2025

Part 1

Code

FastQC

Hide

```
#Make directory for fastqc output
mkdir -p ~/Assessment_GNA5012/fastqc_output

#Running Fastqc on the files
fastqc ~/Assessment_GNA5012/SRR075204_1.fastq.gz ~/Assessment_GNA5012/SRR075204_2.fas
tq.gz -o ~/Assessment_GNA5012/fastqc_output
```

FastQC Report

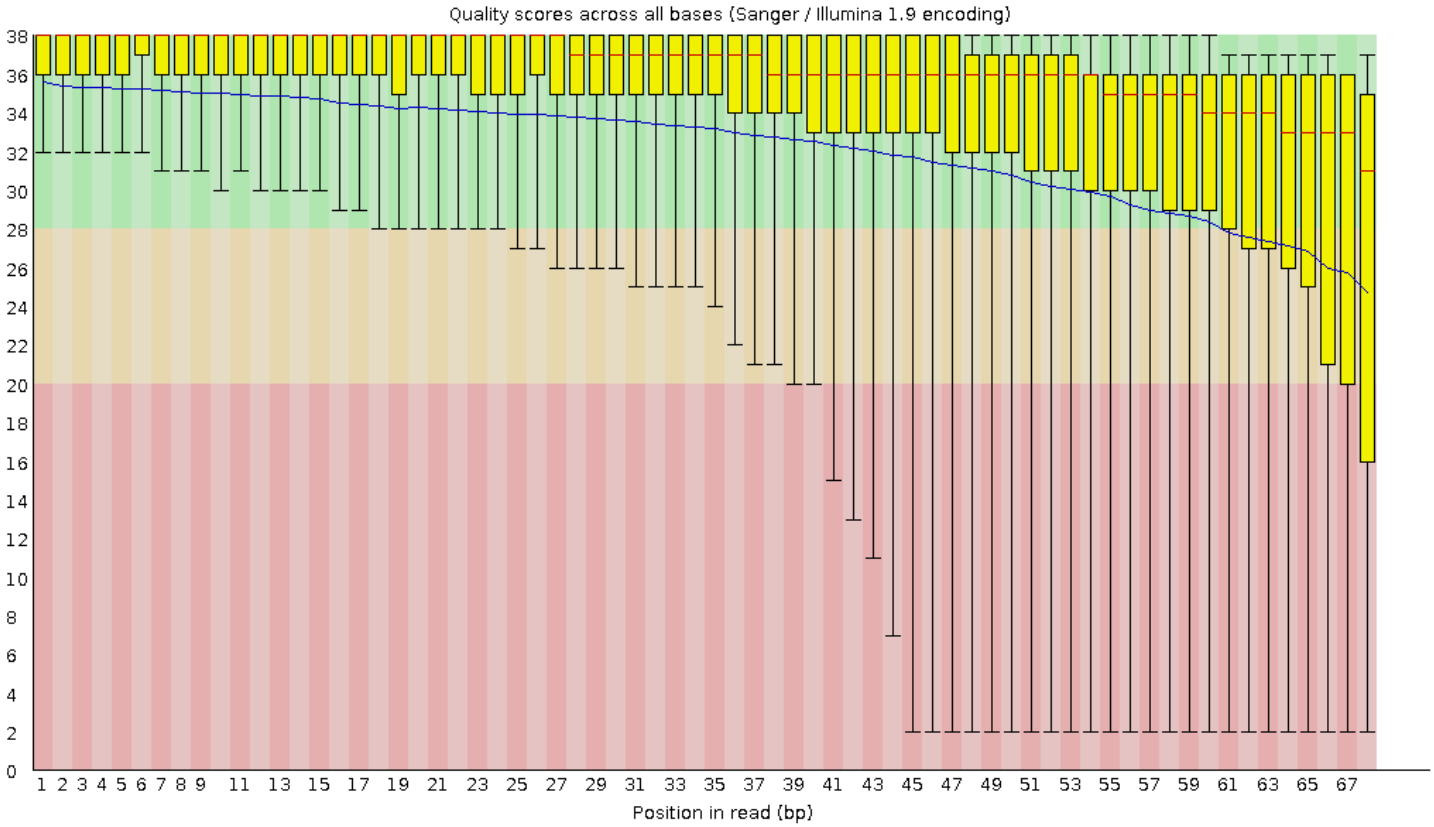
Fri 10 Oct 2025
SRR075204_1.fastq.gz

Summary

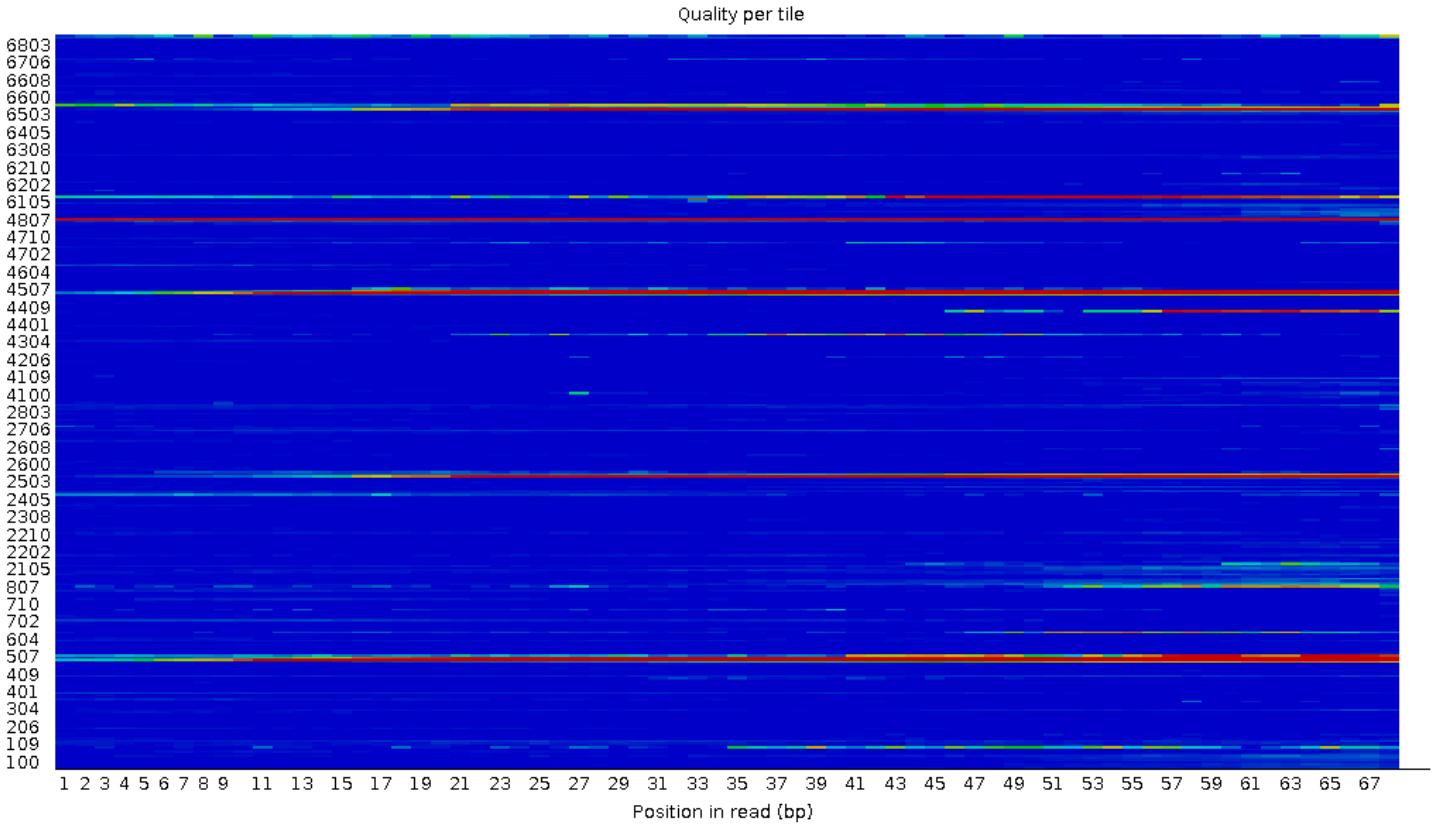
Basic Statistics

Measure	Value
Filename	SRR075204_1.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	782707
Total Bases	53.2 Mbp
Sequences flagged as poor quality	0
Sequence length	68
%GC	45

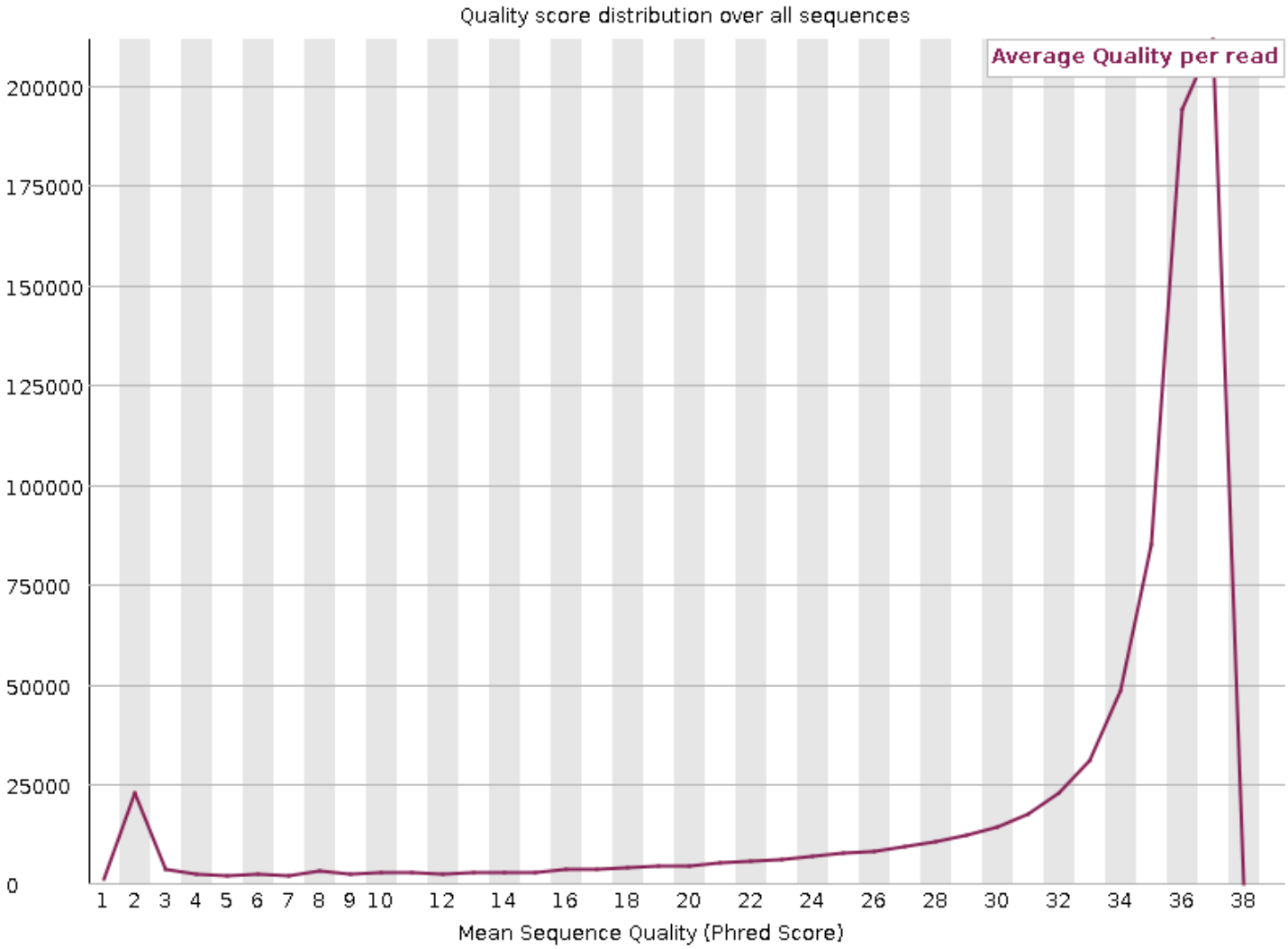
Per base sequence quality



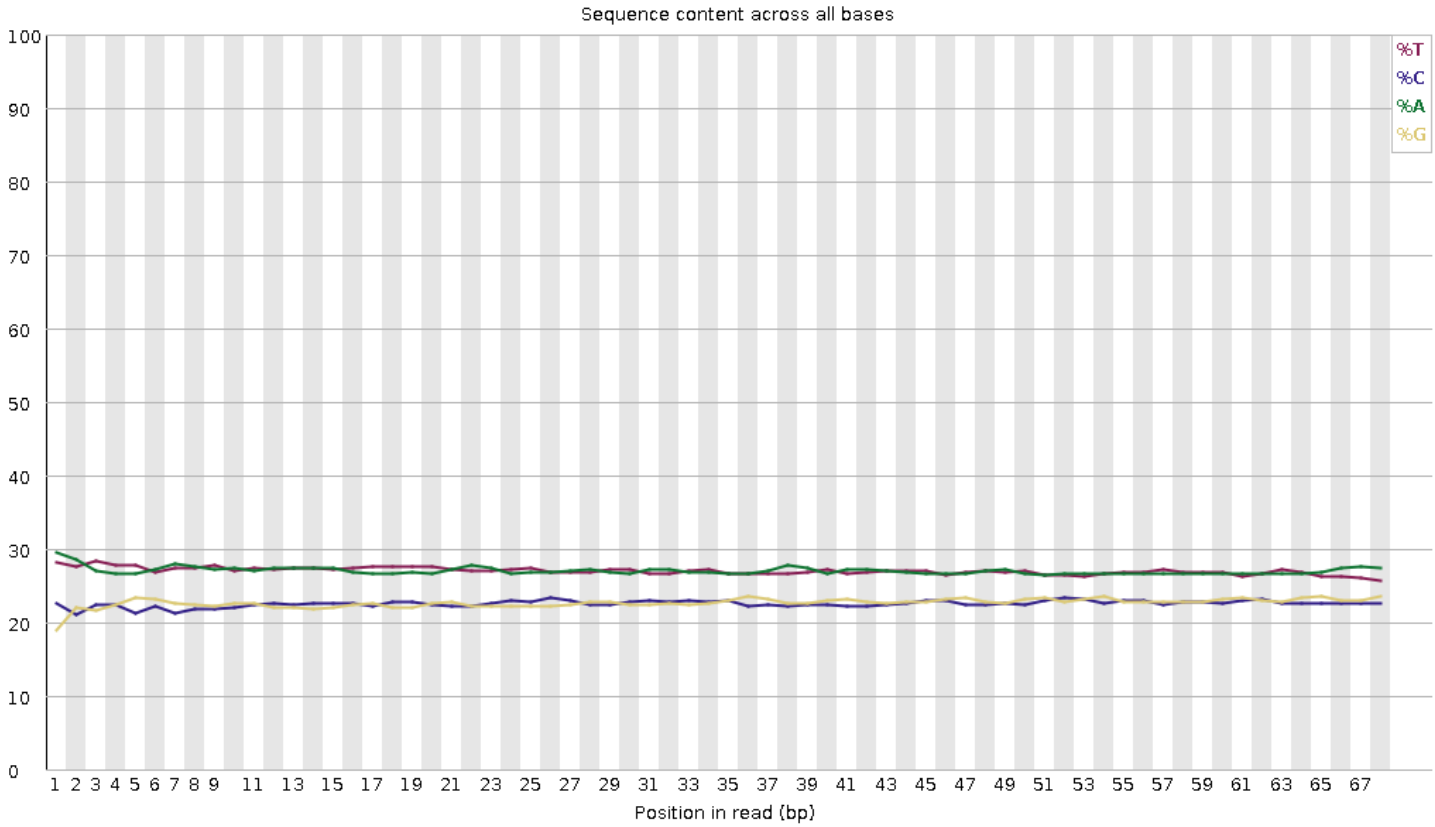
Per tile sequence quality


Quality per tile
6803
6706
6608
6600
6503
6405
6308
6210
6202
6105
4807
4710
4702
4604
4507
4409
4401
4304
4206
4109
4100
2803
2706
2608
2600
2503
2405
2308
2210
2202
2105
807
710
702
604
507
409
401
304
206
109
100
1 2 3 4 5 6 7 8 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67
Position in read (bp)

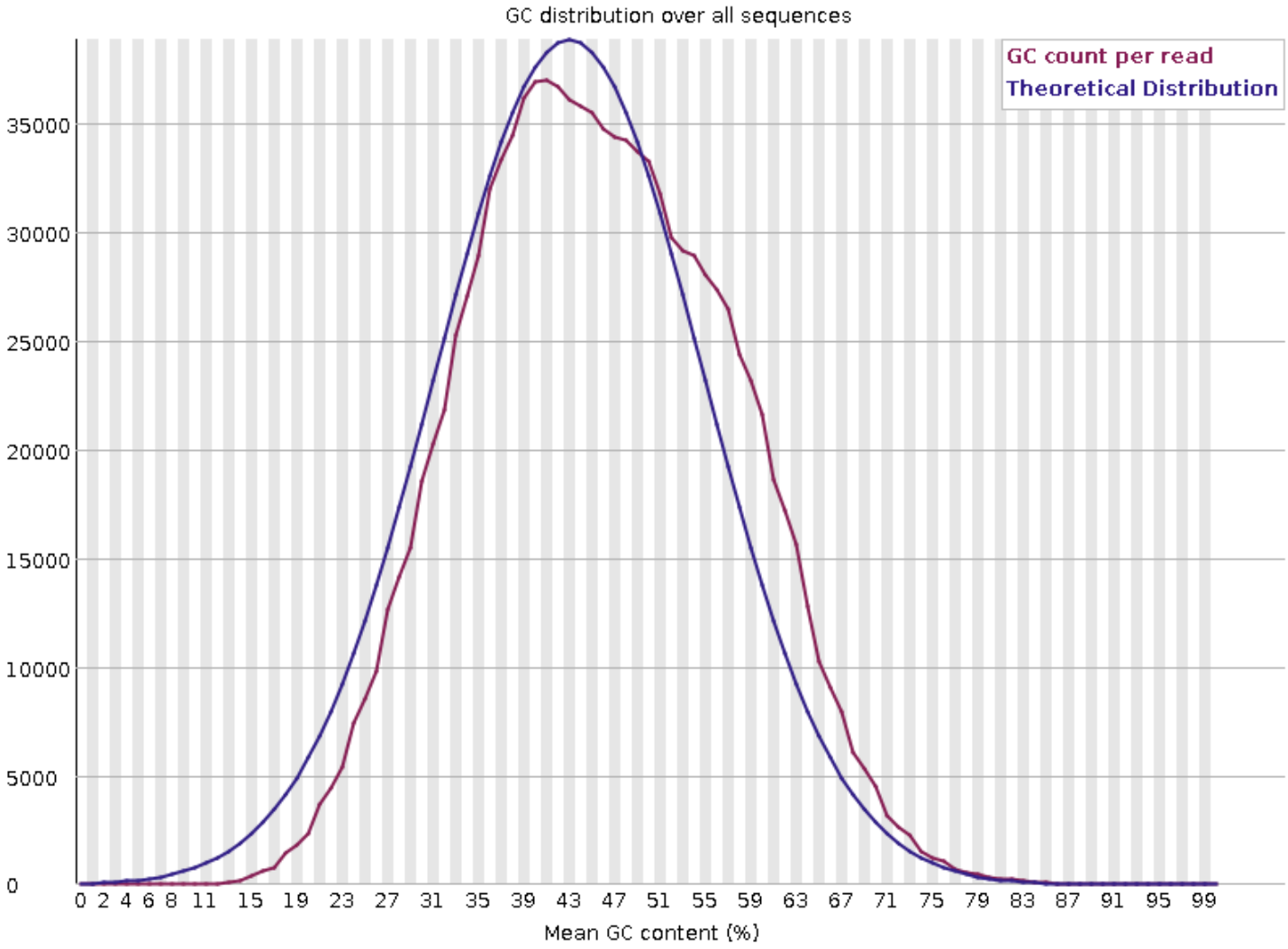
Per sequence quality scores



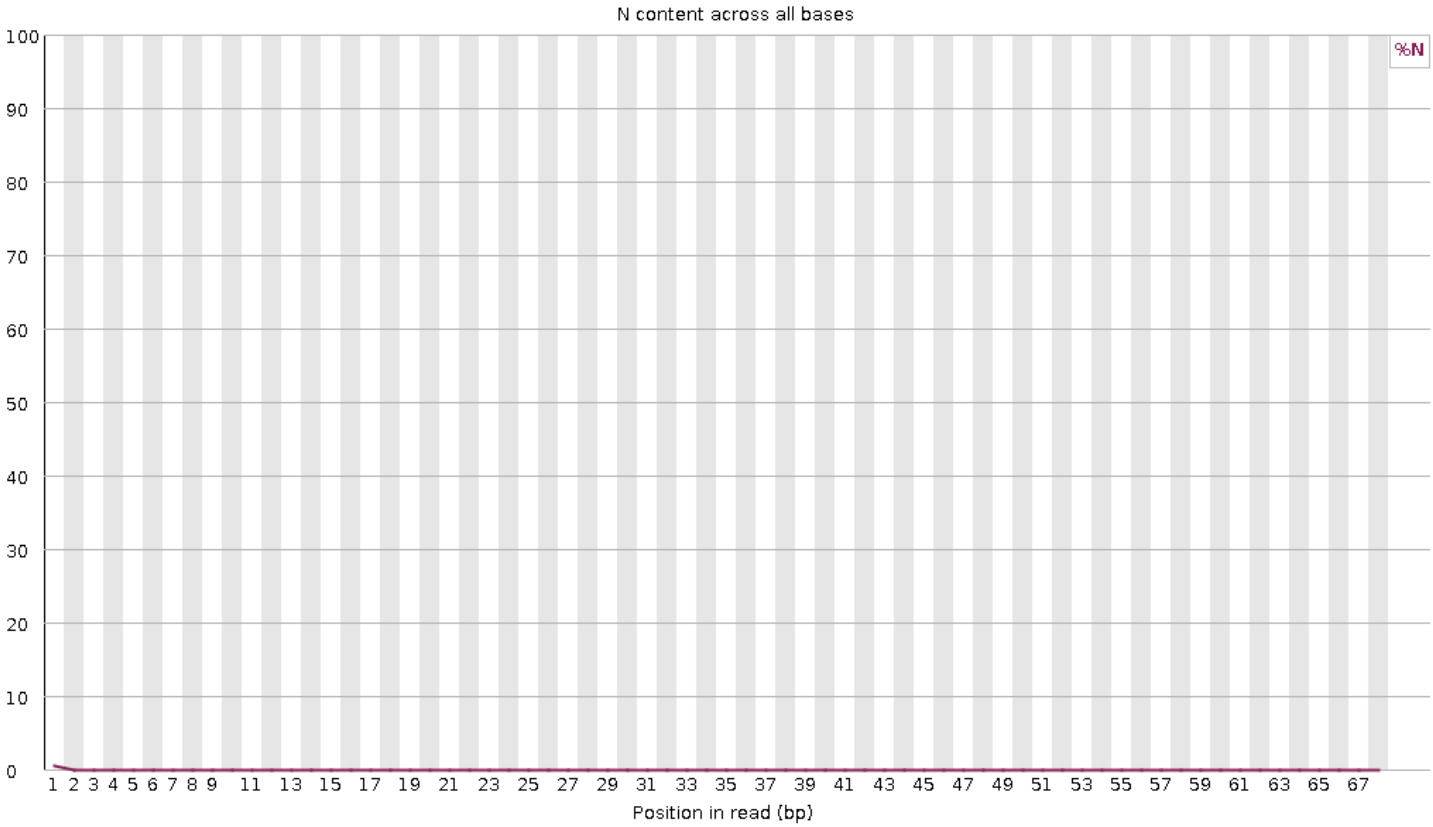
Per base sequence content



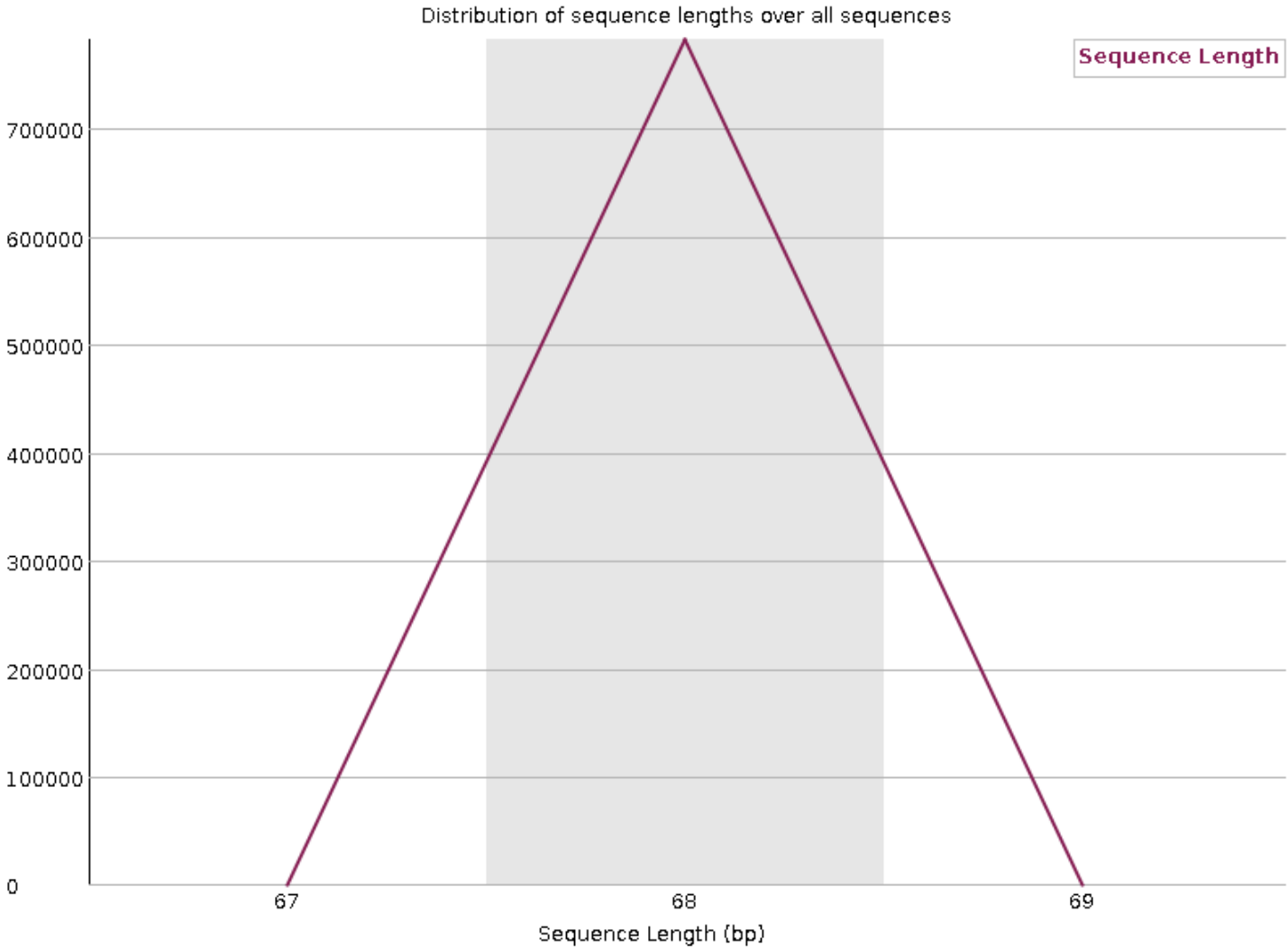
Per sequence GC content



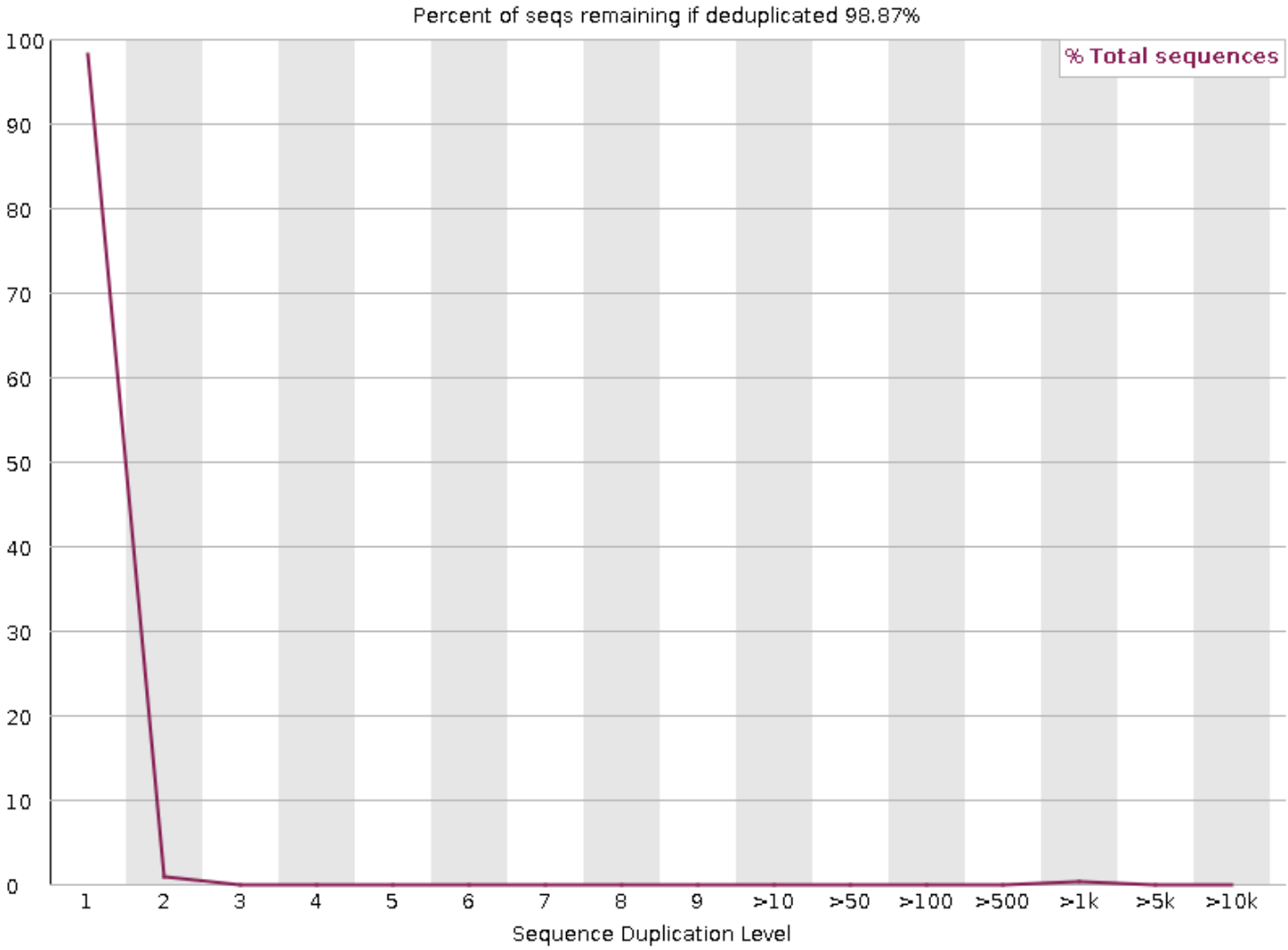
Per base N content



Sequence Length Distribution



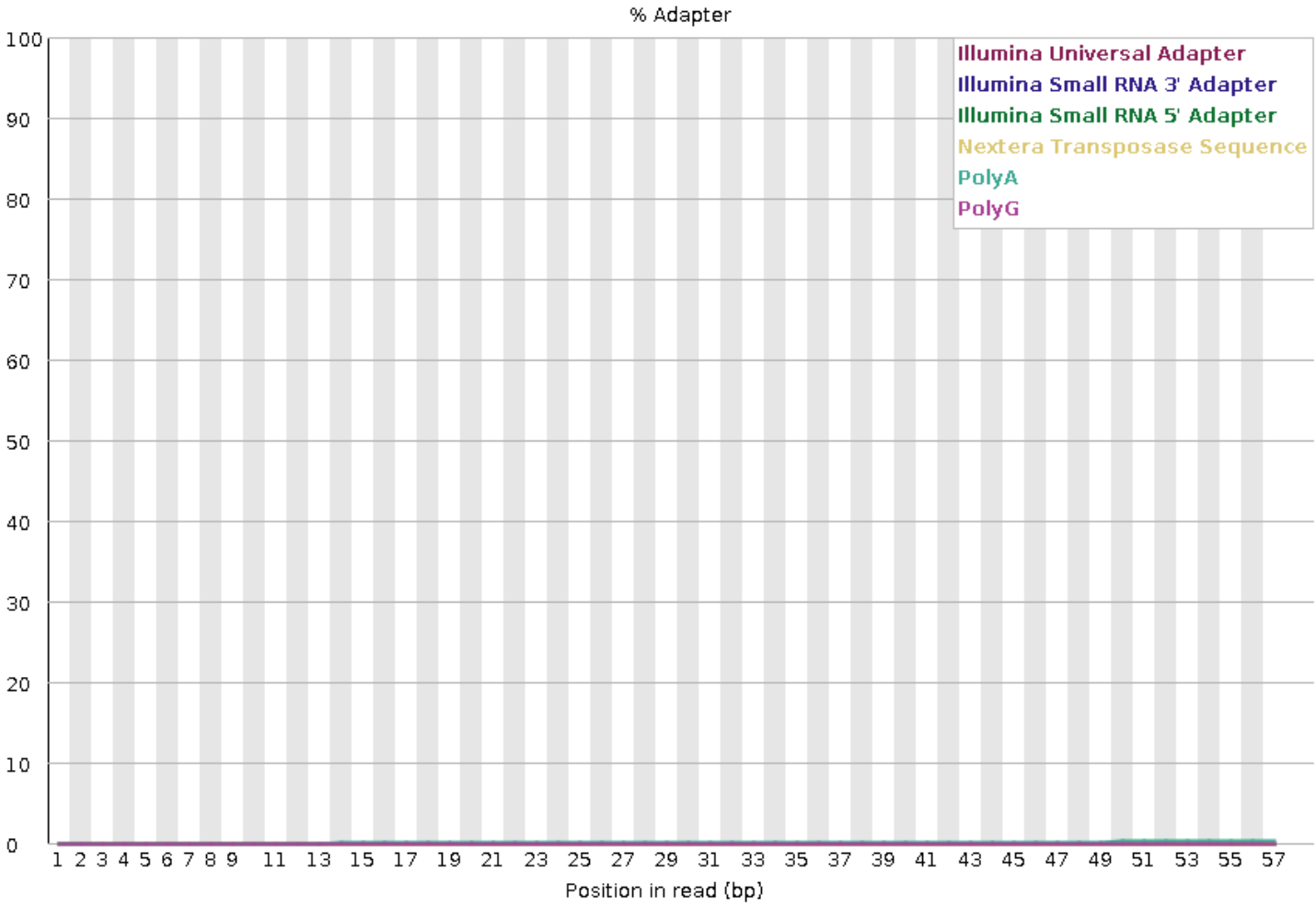
Sequence Duplication Levels



Overrepresented sequences

Sequence	Count	Percentage	Possible Source
GATCGGAAGAGCACACGTCTGAACTCCAGTCACTCGGAATGATCTCGTAT	1903	0.24313057120991635	TruSeq Adapter, Index 10 (97% over 36bp)
GATCGGAAGAGCACACGTCTGAACTCCAGTCACTCGGAATGATATCGTAT	1242	0.1586800680203448	TruSeq Adapter, Index 10 (97% over 36bp)

Adapter Content



Produced by FastQC (version 0.12.1)

FastQC Report

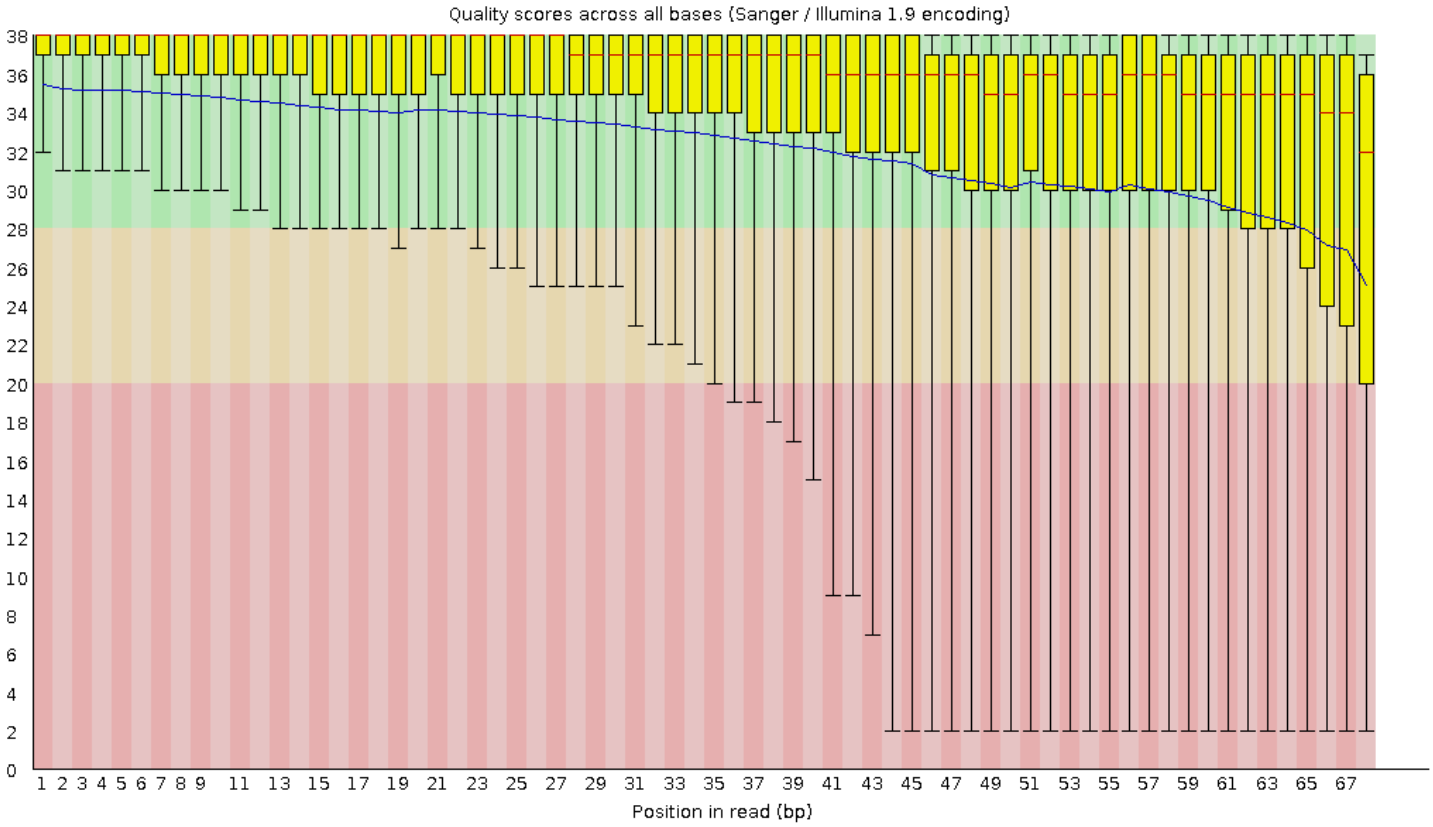
Fri 10 Oct 2025
SRR075204_2.fastq.gz

Summary

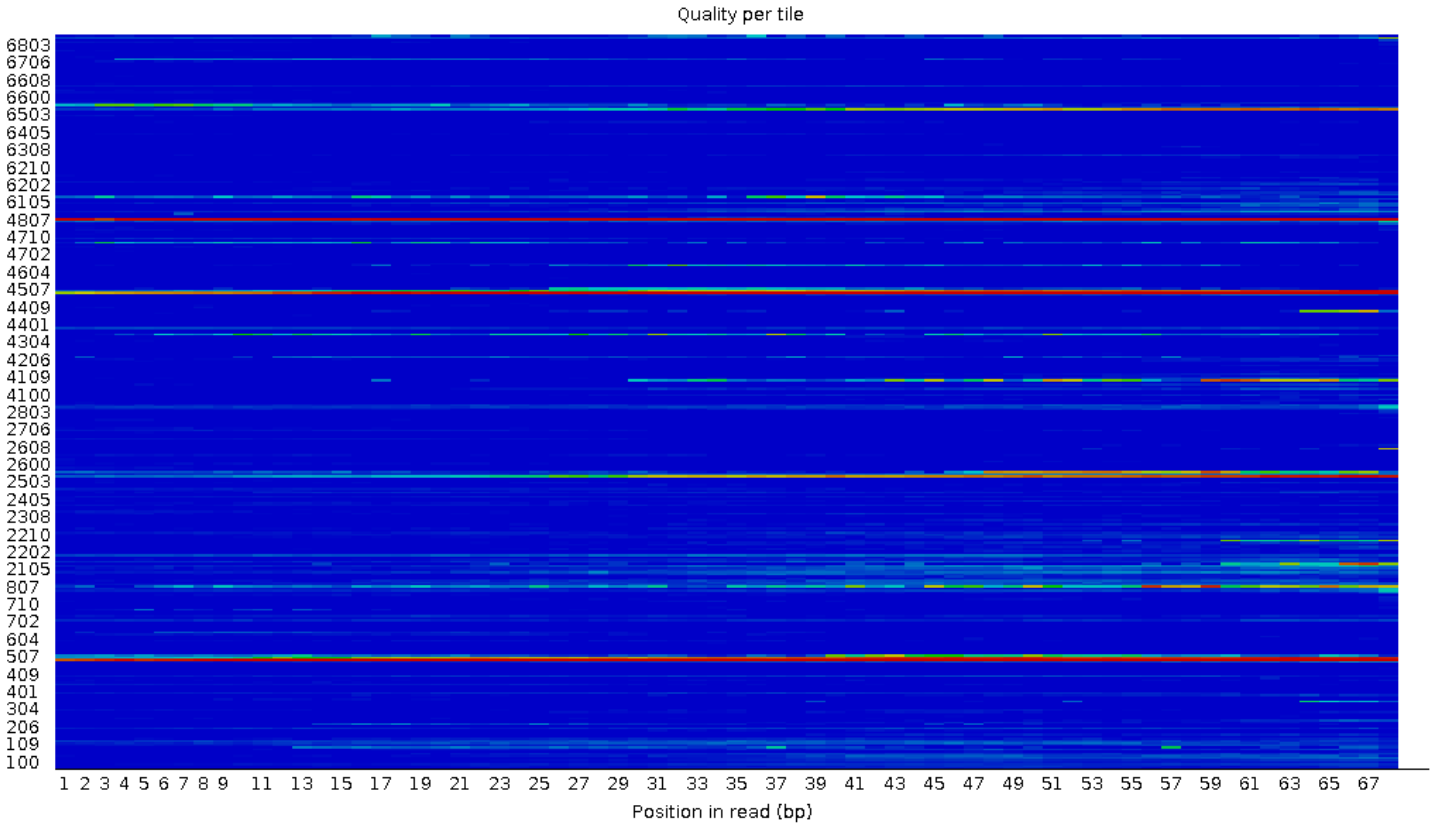
Basic Statistics

Measure	Value
Filename	SRR075204_2.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	782707
Total Bases	53.2 Mbp
Sequences flagged as poor quality	0
Sequence length	68
%GC	45

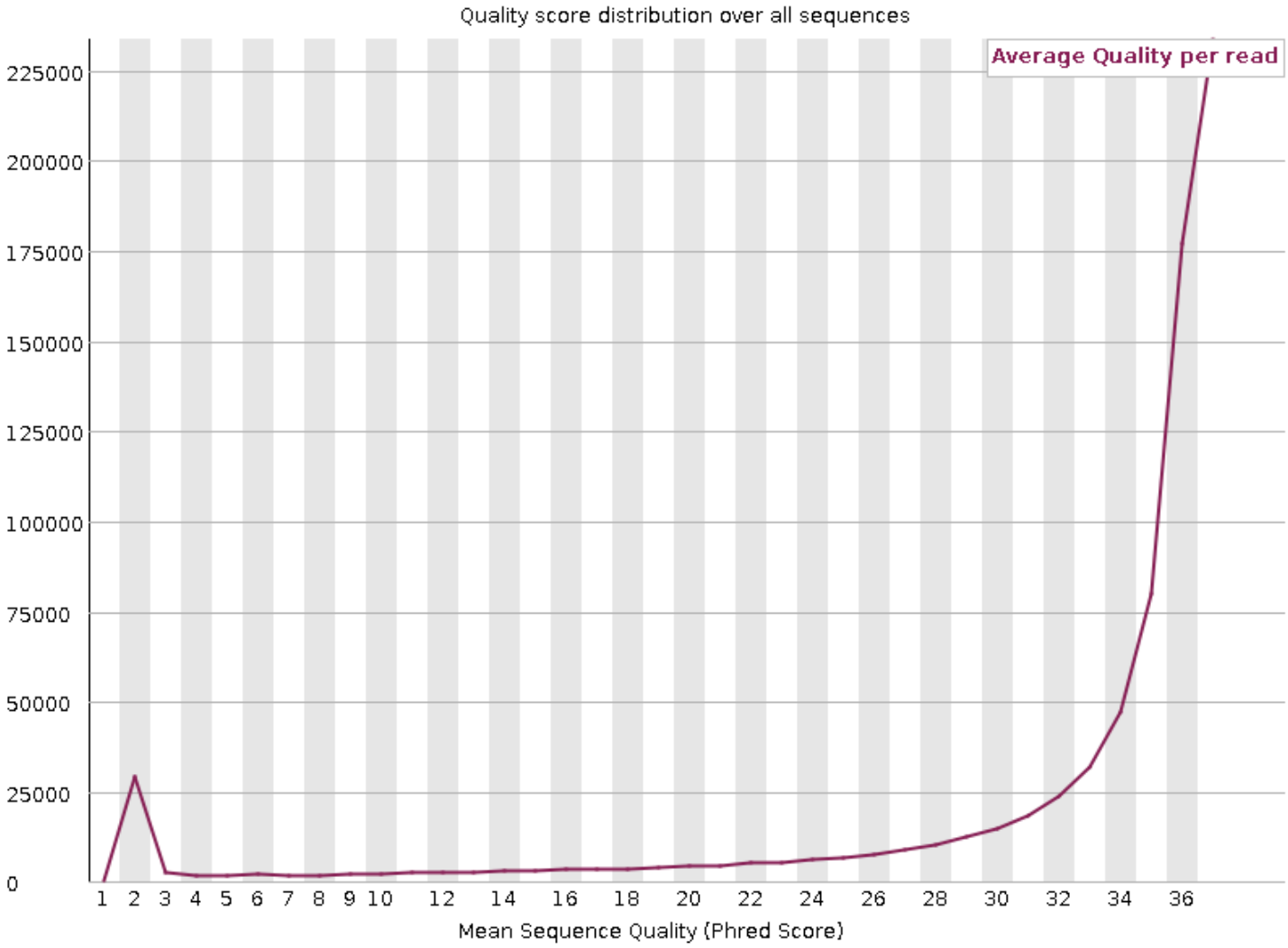
Per base sequence quality



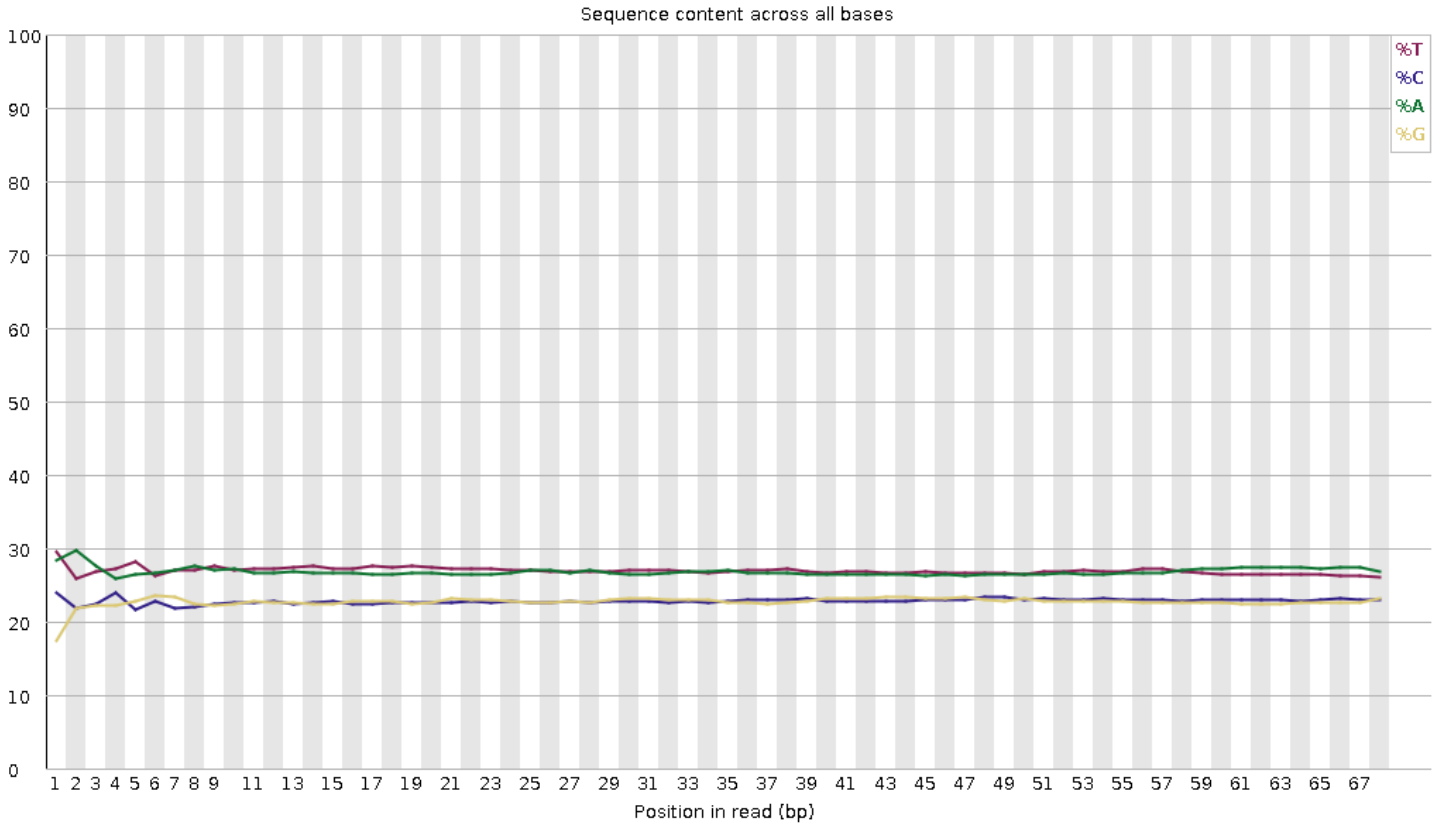
Per tile sequence quality



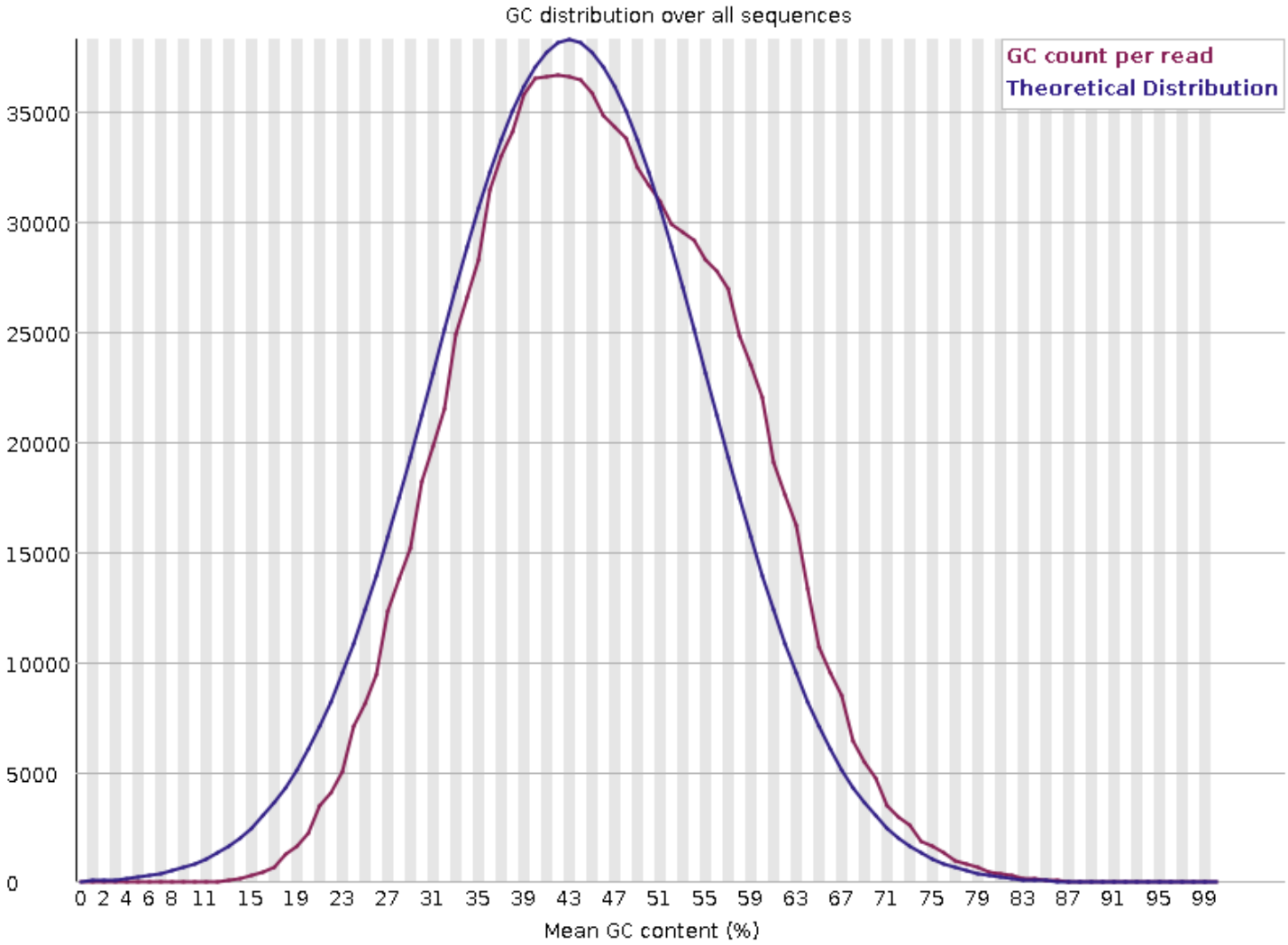
Per sequence quality scores



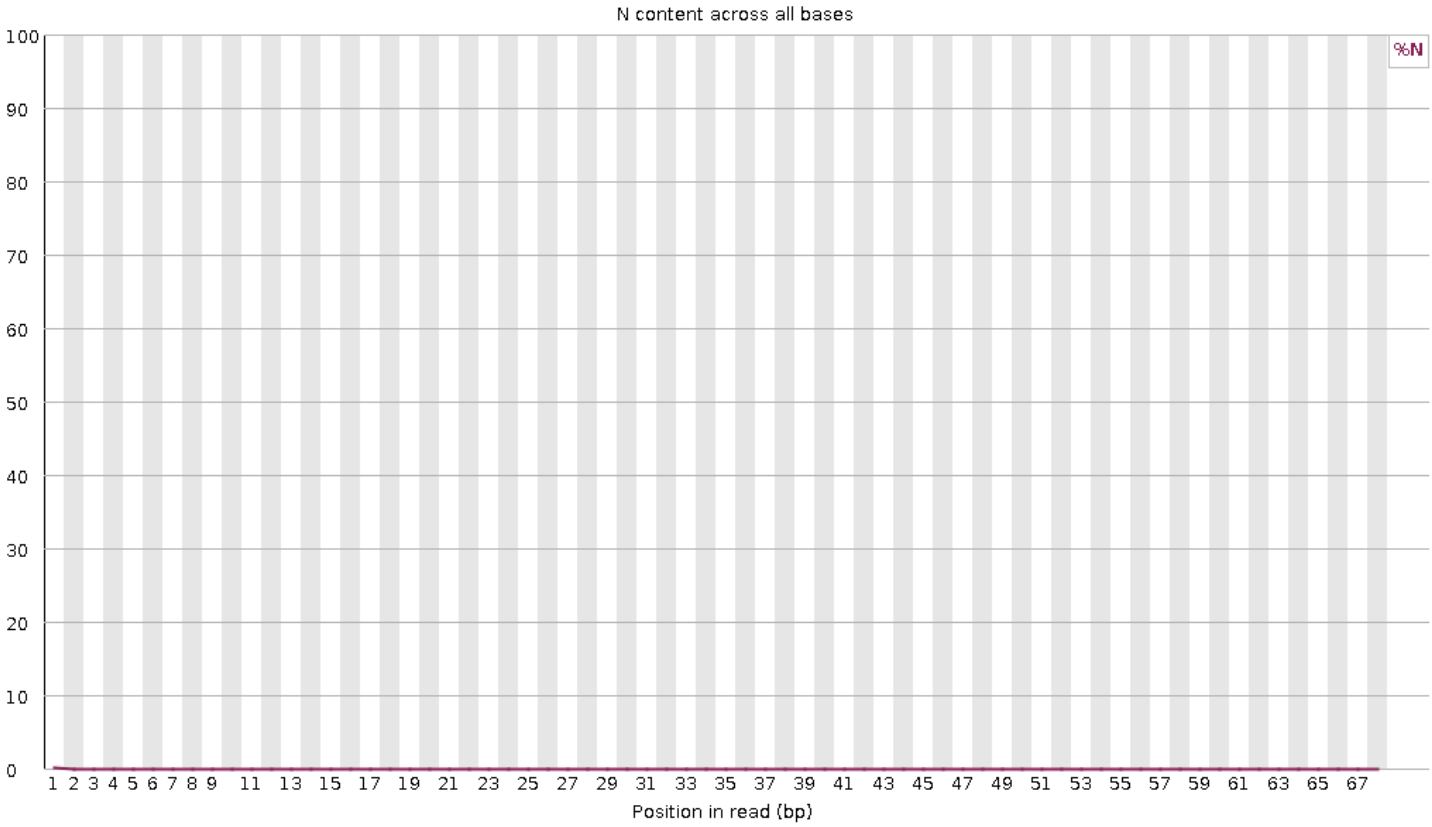
Per base sequence content



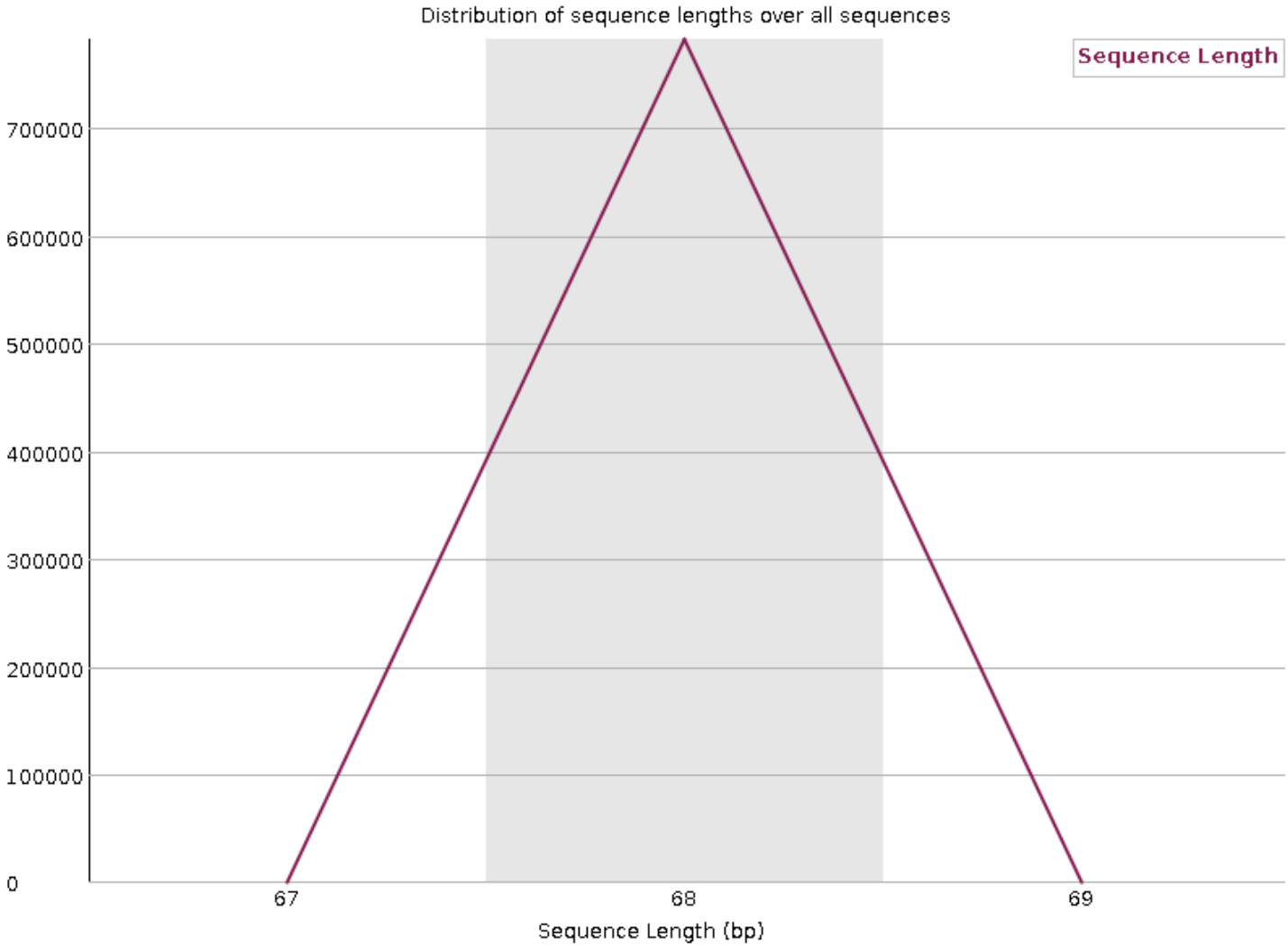
Per sequence GC content



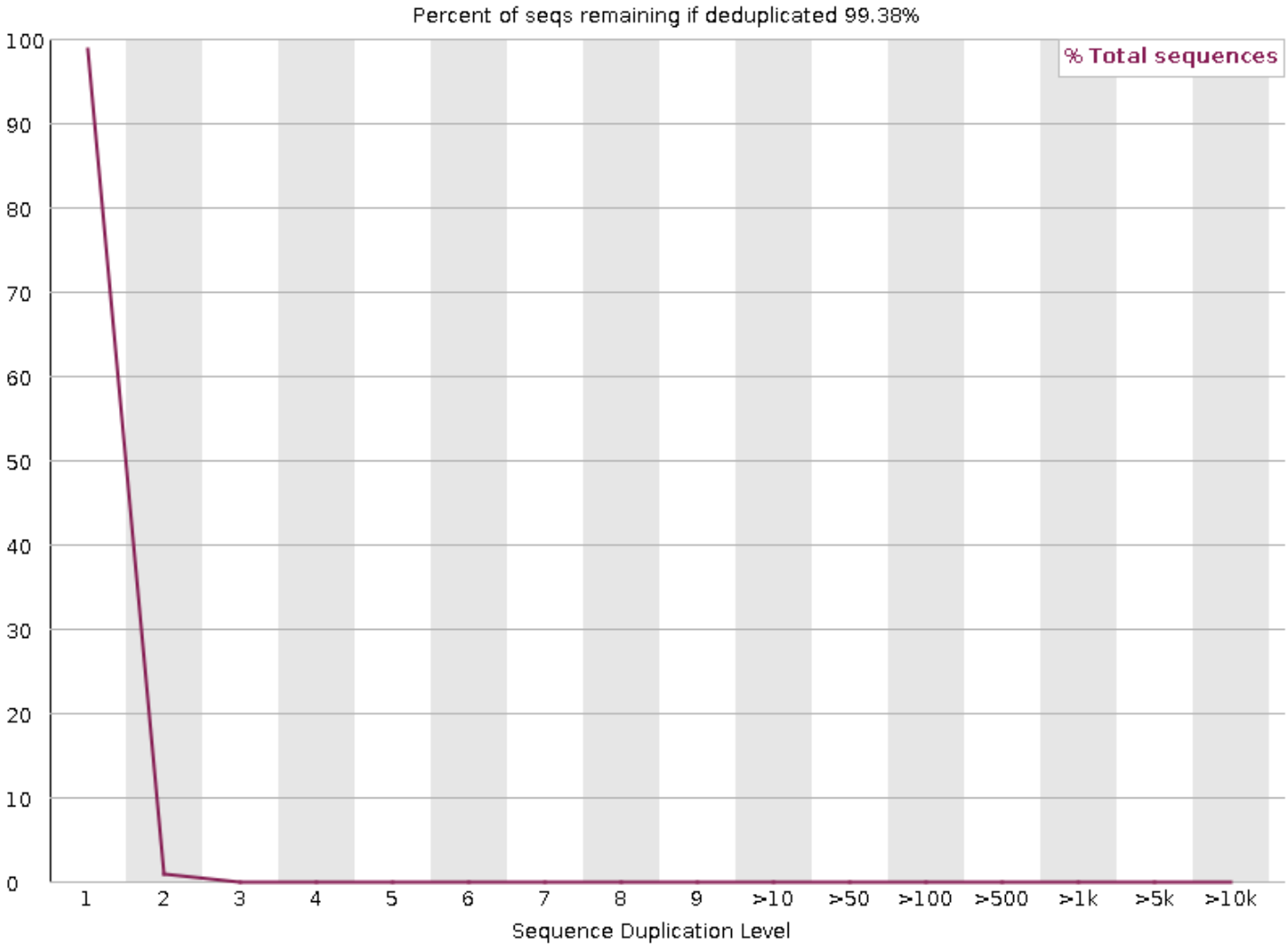
Per base N content



Sequence Length Distribution



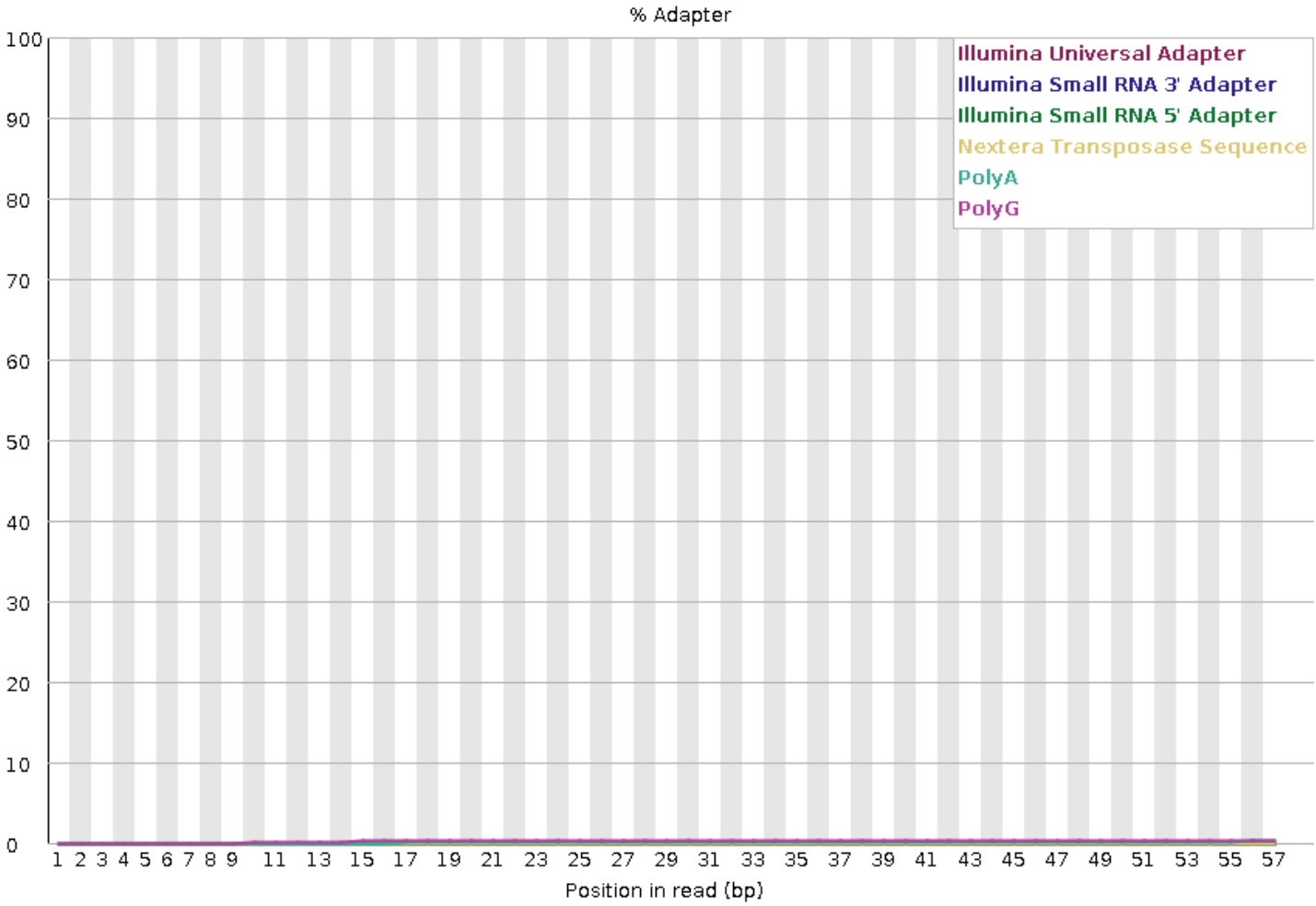
Sequence Duplication Levels



Overrepresented sequences

No overrepresented sequences

Adapter Content



Produced by FastQC (version 0.12.1)

Part 2

Code

1. Read trimming

Hide

```
cd Assessment_GNA5012/Part2/
fastp \
  -i Part2/SRR075204_1.fastq.gz \
  -I Part2/SRR075204_2.fastq.gz \
  -o Part2/SRR075204_trimmed_R1.fastq.gz \
  -O Part2/SRR075204_trimmed_R2.fastq.gz \
  --detect_adapter_for_pe \
  --qualified_quality_phred 20 \
  --length_required 50 \
  --html Part2/SRR075204_fastp_report.html \
```

2. Sequence alignment

Hide

```
#Copying over the reference genome fasta
cp -r /apps/data/GNA5012/week3/reference ~/Assessment_GNA5012/Part2/

bwa mem -t 4 \
  reference/Homo_sapiens.GRCh38.dna.primary_assembly.fa \
  SRR075204_trimmed_R1.fastq.gz \
  SRR075204_trimmed_R2.fastq.gz \
  > SRR075204_aligned.sam

#Count number of aligned reads
grep -v '^@' SRR075204_aligned.sam | wc -l
#Answer: 1336667
```

3. Sorting/Indexing Aligned files

Hide

```
# Convert SAM to BAM
samtools view -bS SRR075204_aligned.sam \
  > SRR075204_aligned.bam

#sort the BAM file
samtools sort SRR075204_aligned.bam \
  -o SRR075204_aligned.sorted.bam
#index the BAM file
samtools index SRR075204_aligned.sorted.bam
```

4. Marking and removing duplicates

Hide

```
#mark duplicates
gatk MarkDuplicates \
  -I SRR075204_aligned.sorted.bam \
  -O SRR075204_aligned.marked.bam \
  -M SRR075204_marked_metrics.txt

#sorting
samtools sort SRR075204_aligned.marked.bam \
  -o SRR075204_aligned.marked.sorted.bam

#indexing
samtools index SRR075204_aligned.marked.sorted.bam
```

5. Variant calling

Hide

```
#variant calling
INPUT=~/Assessment_GNA5012/Part2
SAMPLE=SRR075204
OUTPUT=~/Assessment_GNA5012/Part2/strelka_output
REF=~/Assessment_GNA5012/Part2/reference/Homo_sapiens.GRCh38.dna.primary_assembly.fa

configureStrelkaGermlineWorkflow.py \
  --bam ${INPUT}/${SAMPLE}_aligned.marked.sorted.bam \
  --referenceFasta $REF \
  --runDir ${OUTPUT}/${SAMPLE} \
  --exome

${OUTPUT}/${SAMPLE}/runWorkflow.py -m local -j 4

#running the workflow
cd strelka_output/SRR075204
python2 runWorkflow.py -m local -j 4

#counting
zcat ~/Assessment_GNA5012/Part2/strelka_output/SRR075204/results/variants/variants.vc
f.gz \
  | grep -v "^#" \
  | wc -l
#Answer:94549
```

6. Variant filtering

Hide

```
#filtering variant file
VCF_DIR=~/Assessment_GNA5012/Part2/strelka_output/SRR075204/results/variants
SAMPLE=SRR075204
VCF_FILE=${VCF_DIR}/variants.vcf.gz

bcftools view -i 'FMT/DP>=5 & FMT/GQ>=10' $VCF_FILE \
  -o ${VCF_DIR}/${SAMPLE}_filtered_variants.vcf \
  -O v

#compression
bgzip -c SRR075204_variants.vcf > SRR075204_variants.vcf.gz
bcftools index SRR075204_variants.vcf.gz
```

7. Identifying variant types.

Hide

```
#multiallelic sites
bcftools view --min-alleles 3 *.vcf.gz \
  | grep -v "^#" \
  | wc -l
#Answer:8

#biallelic
bcftools view --min-alleles 2 --max-alleles 2 *.vcf.gz\
  | grep -v "^#" \
  | wc -l
#Answer:1377

#Split multiallelic sites to biallelelic sites
bcftools norm -m- *.vcf.gz

#find number of snps
bcftools view -v snps *.vcf.gz | grep -v "^#" | wc -l
#Answer:1385

#find number of insertions and deletions
bcftools view -v indels *.vcf.gz | grep -v "^#" | wc -l
#Answer:0
```

8. Creating a table

Hide

```
# Create a data frame summarizing the results
results <- data.frame(
  Sample_ID = "NA19655",
  Category = c(
    "Multiallelic sites",
    "Biallelic sites",
    "SNPs",
    "Insertions/Deletions"
  ),
  Count = c(8, 1377, 1385, 0)
)

# Print the table
print(results)
```

Sample_ID <chr>	Category <chr>	Count <dbl>
NA19655	Multiallelic sites	8
NA19655	Biallelic sites	1377
NA19655	SNPs	1385
NA19655	Insertions/Deletions	0

4 rows

Part 3

Code

1. Extract biallelic monomorphic snps from Chromosomes 1 and 10 for all the samples.

Hide

```
bcftools view -m2 -M2 -v snps chr1.vcf.gz -Oz -o chr1_biallelic_snps.vcf.gz
bcftools view -m2 -M2 -v snps chr10.vcf.gz -Oz -o chr10_biallelic_snps.vcf.gz

#minor allele filtering at 0.01 using bcftools in order to reduce file sizes
bcftools view -i 'MAF>=0.01' -Oz -o chr1_biallelic_snps_maf01.vcf.gz chr1_biallelic_s
nps.vcf.gz
bcftools view -i 'MAF>=0.01' -Oz -o chr10_biallelic_snps_maf01.vcf.gz chr10_biallelic
_snps.vcf.gz
```

2. VCF to GDS conversion

Hide

```
#snpgdsVCF2GDS("chr1_biallelic_snps_maf01.vcf.gz", "chr1.gds", method="biallelic.onl
y")
#snpgdsVCF2GDS("chr10_biallelic_snps_maf01.vcf.gz", "chr10.gds", method="biallelic.on
ly")
```

3. Opening the GDS file

Hide

```
library(SNPRelate)
genofile1 <- snpgdsOpen("chr1.gds")
genofile10 <- snpgdsOpen("chr10.gds")
```

4. Running PCA on both GDS files

Hide

```
#pca1 <- snpgdsPCA(genofile1,maf=0.05,num.thread=6)
#pca10 <- snpgdsPCA(genofile10,maf=0.05,num.thread=6)

# Save PCA results to RDS
saveRDS(pca1, file = "pca1.rds")
saveRDS(pca10, file = "pca10.rds")
```

5. Extracting PCA results

Hide

```
#Close the files after use
snpgdsClose(genofile1)
snpgdsClose(genofile10)

#Load PCA results from RDS
pca1 <- readRDS("pca1.rds")
pca10 <- readRDS("pca10.rds")

#Extract key PCA results
# Sample IDs
sample.id1 <- pca1$sample.id
sample.id10 <- pca10$sample.id

# First two principal components
pc1_chr1 <- pca1$eigenvect[, 1]   # PC1
pc2_chr1 <- pca1$eigenvect[, 2]   # PC2

pc1_chr10 <- pca10$eigenvect[, 1]
pc2_chr10 <- pca10$eigenvect[, 2]
```

6. Create data frames for PCA results

Hide

```
# Chromosome 1
pca_chr1.df <- data.frame(
  Sample = sample.id1,
  PC1 = pca1$eigenvect[, 1],
  PC2 = pca1$eigenvect[, 2]
)

# Chromosome 10
pca_chr10.df <- data.frame(
  Sample = sample.id10,
  PC1 = pca10$eigenvect[, 1],
  PC2 = pca10$eigenvect[, 2]
)
```

7. Merge PCA dataframe with Population information file

Hide

```
library(readr)
pop_data <- read_tsv("Populations.tsv")
# Merge using matching columns
pca_chr1_merged <- merge(pca_chr1.df, pop_data, by.x = "Sample", by.y = "Sample nam
e")
pca_chr10_merged <- merge(pca_chr10.df, pop_data, by.x = "Sample", by.y = "Sample nam
e")

#Check merged data
head(pca_chr1_merged)
```

	Sample <chr>	PC1 <dbl>	PC2 <dbl>	Sex <chr>	Biosample ID <chr>	Population code <chr>	Population nar <chr>
1	HG00096	-0.010725200	-0.02232879	male	SAME123268	GBR	British
2	HG00097	-0.010116423	-0.02463782	female	SAME123267	GBR	British
3	HG00099	-0.010864126	-0.02409722	female	SAME123271	GBR	British
4	HG00100	-0.010688910	-0.02297233	female	SAME125154	GBR	British
5	HG00101	-0.011125493	-0.02453220	male	SAME125153	GBR	British
6	HG00102	-0.009851153	-0.02307373	female	SAME123945	GBR	British

6 rows | 1-8 of 11 columns

Hide

```
head(pca_chr10_merged)
```

	Sample <chr>	PC1 <dbl>	PC2 <dbl>	Sex <chr>	Biosample ID <chr>	Population code <chr>	Population nar <chr>
1	HG00096	-0.010758033	-0.02252491	male	SAME123268	GBR	British
2	HG00097	-0.010993712	-0.02511726	female	SAME123267	GBR	British
3	HG00099	-0.008630289	-0.02280042	female	SAME123271	GBR	British
4	HG00100	-0.009862659	-0.02468313	female	SAME125154	GBR	British
5	HG00101	-0.010740820	-0.02344574	male	SAME125153	GBR	British
6	HG00102	-0.011863574	-0.02157427	female	SAME123945	GBR	British

6 rows | 1-8 of 11 columns

8. Get unique populations and superpopluations to use for downstream plotting.

Hide

```
# Get unique populations
unique_pops <- unique(pop_data$"Population code")

# View them
print(unique_pops)
```

```
 [1] "GBR"     "CHS"     "FIN"     "PUR"     "CLM"     "CDX"     "PEL"     "PJL"
"KHV"     "IBS"     "ACB"     "GWD"     "ESN"     "BEB"     "MSL"     "ITU"     "STU"
"CEU"
[19] "YRI"     "JPT"     "CHB"     "LWK"     "MXL"     "ASW"     "TSI"     "GIH"
"IBS,MSL"
```

Hide

```
# Get unique superpopulations
unique_superpops <- unique(pop_data$"Superpopulation code")

# View them
print(unique_superpops)
```

```
[1] "EUR"     "EAS"     "AMR"     "SAS"     "AFR"     "EUR,AFR"
```

9. Create subsets according to superpopulations and assign them to a chromosome specific data frame

Hide

```
# --- Chromosome 1 ---
pca_chr1_AFR <- subset(pca_chr1_merged, `Superpopulation code` == "AFR")
pca_chr1_EUR <- subset(pca_chr1_merged, `Superpopulation code` == "EUR")
pca_chr1_EAS <- subset(pca_chr1_merged, `Superpopulation code` == "EAS")
pca_chr1_AMR <- subset(pca_chr1_merged, `Superpopulation code` == "AMR")
pca_chr1_SAS <- subset(pca_chr1_merged, `Superpopulation code` == "SAS")

# --- Chromosome 10 ---
pca_chr10_AFR <- subset(pca_chr10_merged, `Superpopulation code` == "AFR")
pca_chr10_EUR <- subset(pca_chr10_merged, `Superpopulation code` == "EUR")
pca_chr10_EAS <- subset(pca_chr10_merged, `Superpopulation code` == "EAS")
pca_chr10_AMR <- subset(pca_chr10_merged, `Superpopulation code` == "AMR")
pca_chr10_SAS <- subset(pca_chr10_merged, `Superpopulation code` == "SAS")

# List of Chromosome 1 data frames
chr1_dfs <- list(
  AFR = pca_chr1_AFR,
  EUR = pca_chr1_EUR,
  EAS = pca_chr1_EAS,
  AMR = pca_chr1_AMR,
  SAS = pca_chr1_SAS
)

# List of Chromosome 10 data frames
chr10_dfs <- list(
  AFR = pca_chr10_AFR,
  EUR = pca_chr10_EUR,
  EAS = pca_chr10_EAS,
  AMR = pca_chr10_AMR,
  SAS = pca_chr10_SAS
)
```

10. Creating the figure legend specifications and color specifications for the GGplot for all 26 chromsomes altogether.

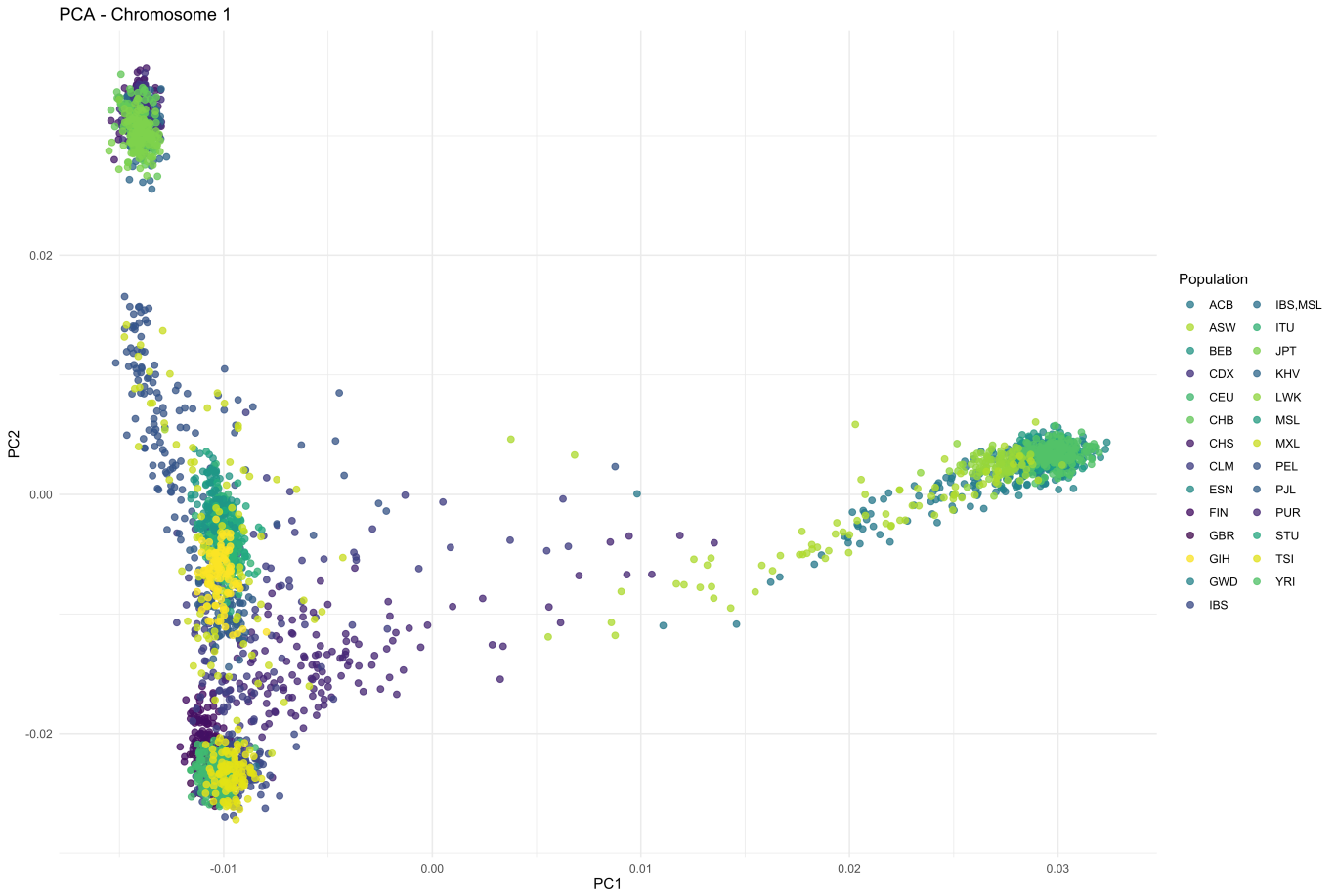
Hide

```
library(viridisLite)
library(stringr)  # for str_wrap

# Define color palette based on all populations
all_pops <- unique(c(pca_chr1_merged$`Population code`, pca_chr10_merged$`Population
code`))
pop_colors <- setNames(viridis(length(all_pops)), all_pops)

# Function to create caption for all populations
create_full_caption <- function(pop_codes, chromosome_number) {
  sp_data <- pop_data[pop_data$`Population code` %in% pop_codes, ]

  # Arrange populations in the order of pop_codes
  sp_data <- sp_data[match(pop_codes, sp_data$`Population code`), ]

  pop_list <- paste0(sp_data$`Population name`, " (", sp_data$`Population code`, ")",
collapse = ", ")

  legend_text <- paste0(
    "This PCA plot shows the variation present in Chromosome ", chromosome_number,
    ". The populations included are: ", pop_list, "."
  )

  # Wrap long caption so it doesn't get cut off
  str_wrap(legend_text, width = 200)
}
```

11. Plot for PCA results for chromosome 1 and 10 for all 26 populations.

Hide

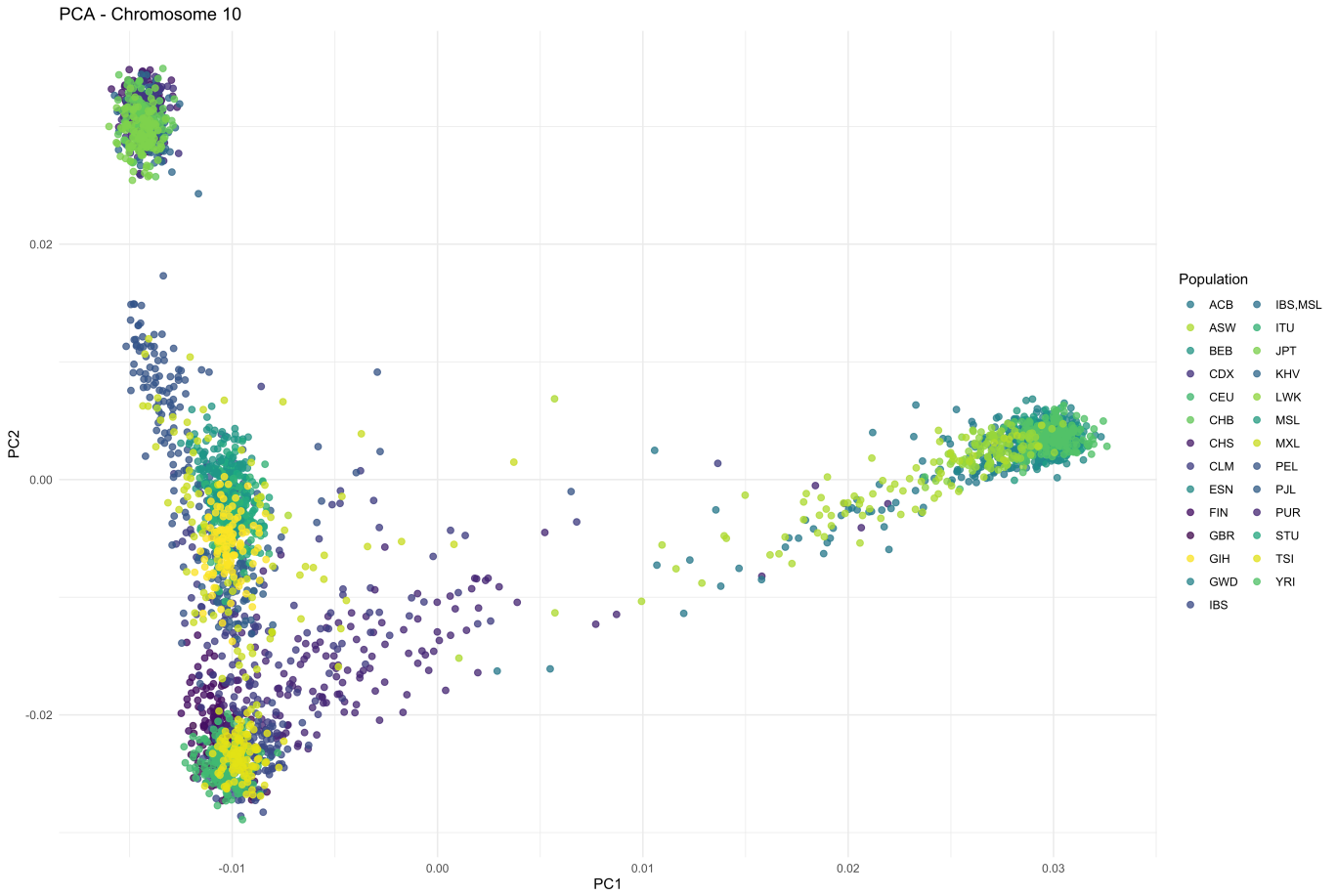
```
library(ggplot2)
#PCA plot for Chromosome 1
caption_chr1 <- create_full_caption(all_pops, chromosome_number = 1)

ggplot(pca_chr1_merged, aes(x = PC1, y = PC2, color = `Population code`)) +
  geom_point(size = 2, alpha = 0.8) +
  scale_color_manual(values = pop_colors) +
  labs(title = "PCA – Chromosome 1",
       x = "PC1",
       y = "PC2",
       color = "Population",
       caption = caption_chr1) +
  theme_minimal() +
  theme(plot.caption.position = "plot",
        plot.caption = element_text(hjust = 0, face = "italic", size = 9),

        # Add extra margin around plot to prevent clipping
        plot.margin = margin(t = 15, r = 10, b = 10, l = 10))
```



This PCA plot shows the variation present in Chromosome 1. The populations included are: British (GBR), Finnish (FIN), Southern Han Chinese (CHS), Puerto Rican (PUR), Dai Chinese (CDX), Colombian (CLM), Iberian (IBS), Peruvian (PEL), Punjabi (PJL), Kinh Vietnamese (KHV), Iberian,Mende (IBS,MSL), African Caribbean (ACB), Gambian Mandinka (GWD), Esan (ESN), Bengali (BEB), Mende (MSL), Tamil (STU), Telugu (ITU), CEPH (CEU), Yoruba (YRI), Han Chinese (CHB), Japanese (JPT), Luhya (LWK), African Ancestry SW (ASW), Mexican Ancestry (MXL), Toscani (TSI), Gujarati (GIH).

Hide

```
#PCA plot for Chromosome 10
caption_chr10 <- create_full_caption(all_pops, chromosome_number = 10)

ggplot(pca_chr10_merged, aes(x = PC1, y = PC2, color = `Population code`)) +
  geom_point(size = 2, alpha = 0.8) +
  scale_color_manual(values = pop_colors) +
  labs(title = "PCA - Chromosome 10",
       x = "PC1",
       y = "PC2",
       color = "Population",
       caption = caption_chr10) +
  theme_minimal() +
  theme(plot.caption.position = "plot",
        plot.caption = element_text(hjust = 0, face = "italic", size = 9),

        # Add extra margin around plot to prevent clipping
        plot.margin = margin(t = 15, r = 10, b = 10, l = 10))
```



This PCA plot shows the variation present in Chromosome 10. The populations included are: British (GBR), Finnish (FIN), Southern Han Chinese (CHS), Puerto Rican (PUR), Dai Chinese (CDX), Colombian (CLM), Iberian (IBS), Peruvian (PEL), Punjabi (PJL), Kinh Vietnamese (KHV), Iberian,Mende (IBS,MSL), African Caribbean (ACB), Gambian Mandinka (GWD), Esan (ESN), Bengali (BEB), Mende (MSL), Tamil (STU), Telugu (ITU), CEPH (CEU), Yoruba (YRI), Han Chinese (CHB), Japanese (JPT), Luhya (LWK), African Ancestry SW (ASW), Mexican Ancestry (MXL), Toscani (TSI), Gujarati (GIH).

12. Creating the figure legend specifications and color specifications for the GGplots divided across subpopulations.

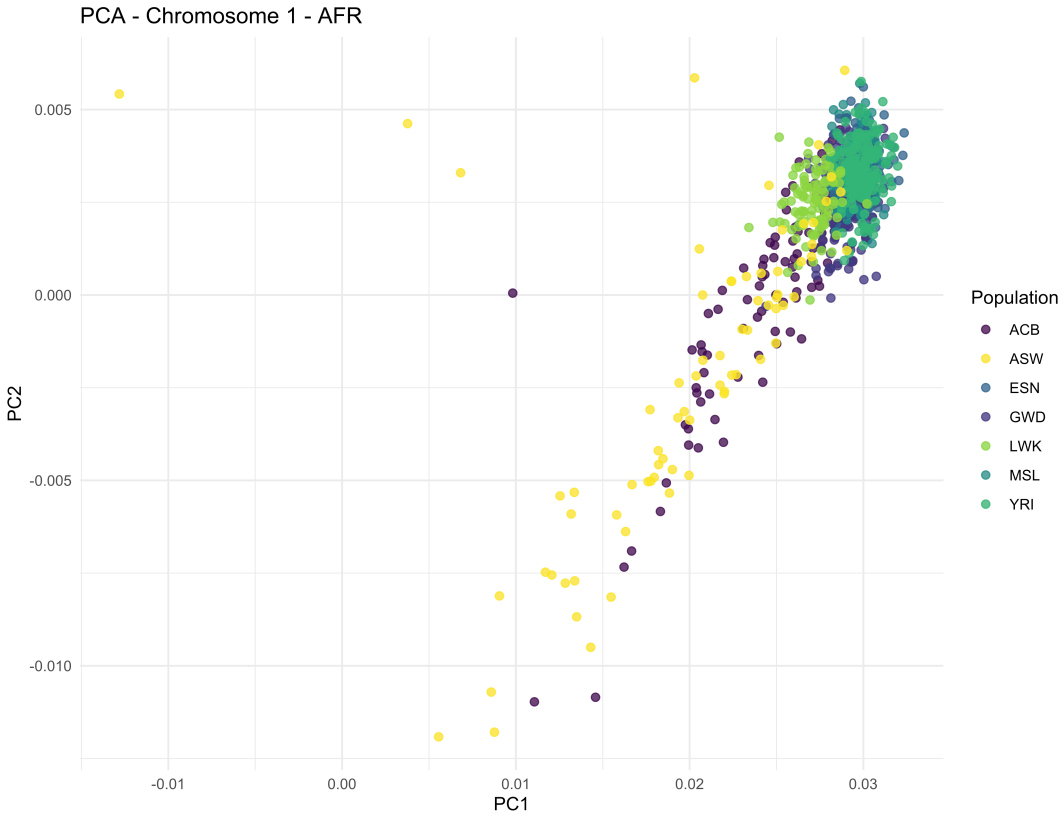
Hide

```
library (stringr)
create_legend_text <- function(superpop_code, chromosome_number, pops_in_order) {
  sp_data <- pop_data[pop_data$`Superpopulation code` == superpop_code, ]
  # Arrange in the same order as the plot
  sp_data <- sp_data[match(pops_in_order, sp_data$`Population code`), ]

  pop_list <- paste0(sp_data$`Population name`, " (", sp_data$`Population code`, ")",
collapse = ", ")

  legend_text <- paste0(
    "This PCA plot shows the variation present in Chromosome ", chromosome_number,
    " of the ", sp_data$`Superpopulation name`[1], " superpopulation (", superpop_cod
e, "). ",
    "This superpopulation includes the populations: ", pop_list, "."
  )

  legend_text <- str_wrap(legend_text, width = 100)

  return(legend_text)
}
```

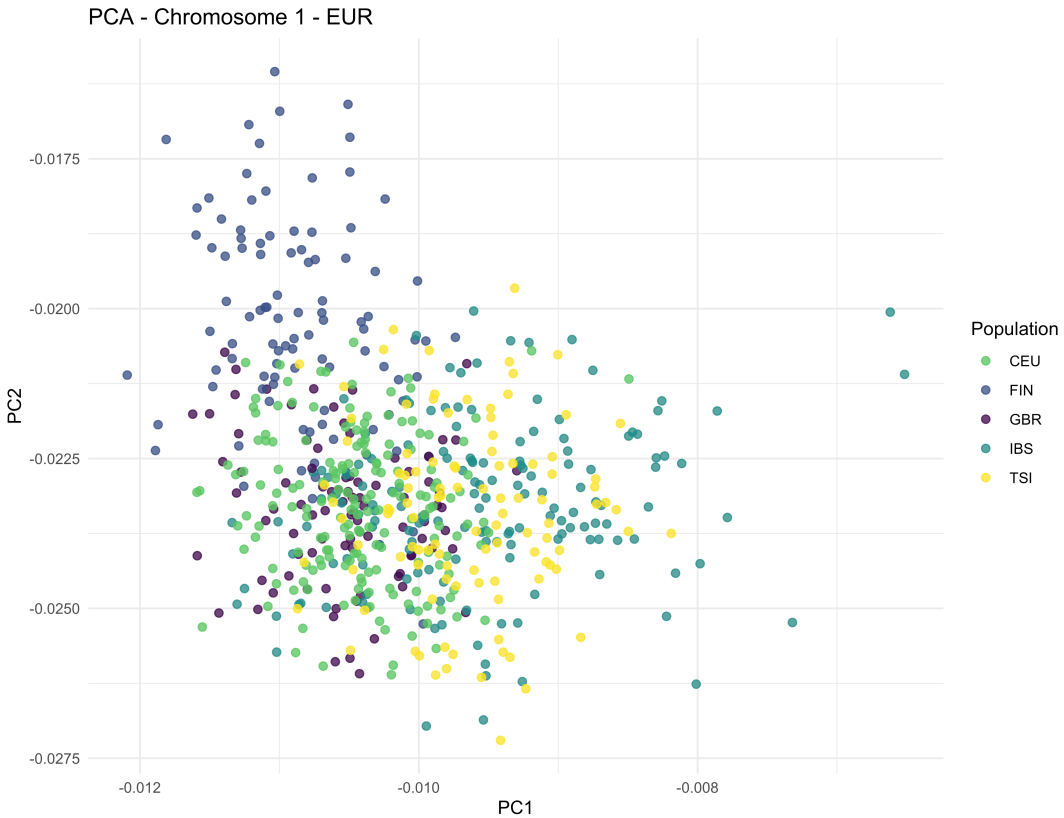
13. Plotting for PCA results for chromosome 1 and 10 for all 26 populations divided by superpopulations.

Hide

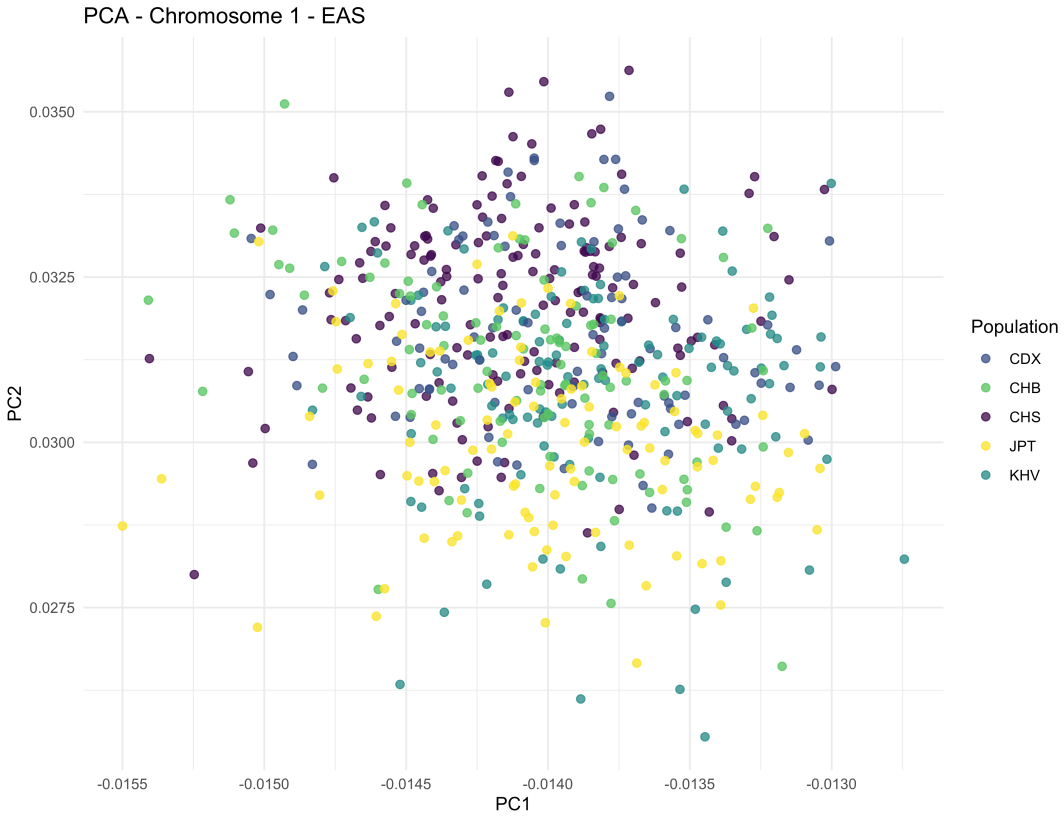
```
library(ggplot2)
library(viridisLite)
#Chromosome 1 plots
for(pop in names(chr1_dfs)){
  df <- chr1_dfs[[pop]]

  # Define population order and colors
  pops_in_group <- unique(df$`Population code`)
  pop_colors <- setNames(viridis(length(pops_in_group)), pops_in_group)

  # Generate caption
  legend_text <- create_legend_text(pop, chromosome_number = 1, pops_in_order = pops_
in_group)

  # Plot
  p <- ggplot(df, aes(x = PC1, y = PC2, color = `Population code`)) +
    geom_point(size = 2, alpha = 0.8) +
    scale_color_manual(values = pop_colors) +
    labs(title = paste("PCA - Chromosome 1 -", pop),
         x = "PC1",
         y = "PC2",
         color = "Population",
         caption = legend_text) +
    theme_minimal() +
    theme(plot.caption = element_text(hjust = 0,size = 10),

          # Add extra margin around plot to prevent clipping
          plot.margin = margin(t = 15, r = 10, b = 10, l = 10))

  print(p)
}
```



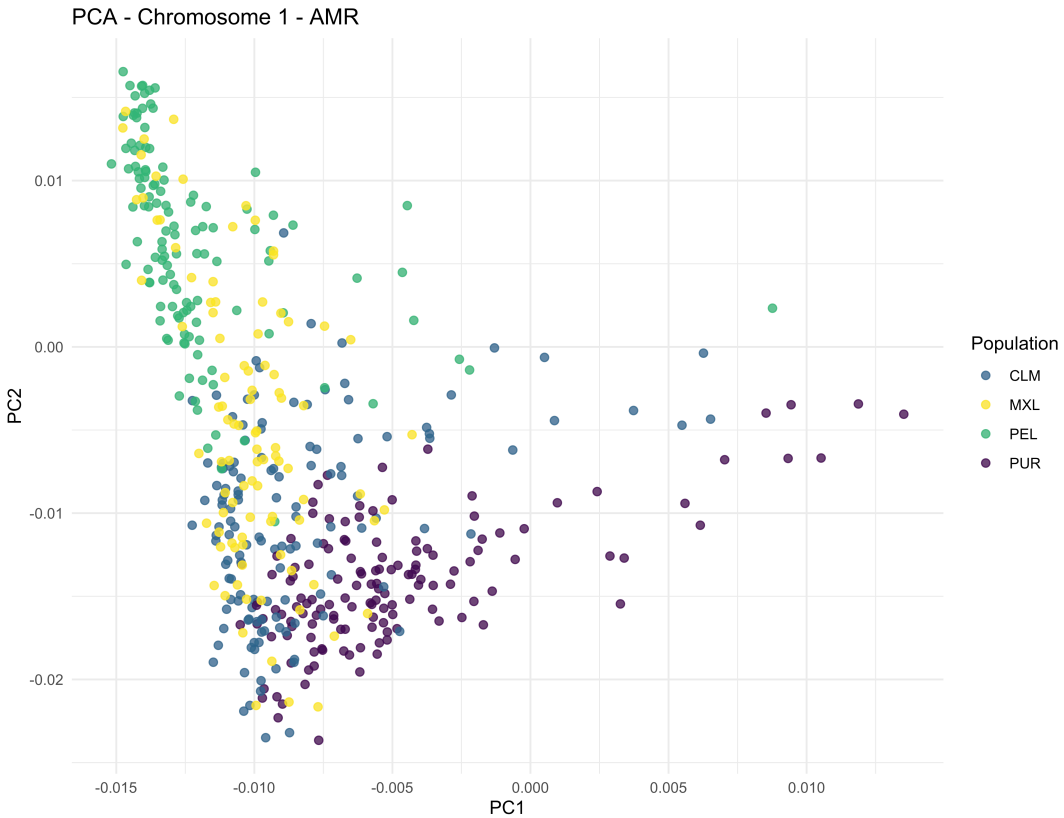
This PCA plot shows the variation present in Chromosome 1 of the African Ancestry superpopulation (AFR). This superpopulation includes the populations: African Caribbean (ACB), Gambian Mandinka (GWD), Esan (ESN), Mende (MSL), Yoruba (YRI), Luhya (LWK), African Ancestry SW (ASW).



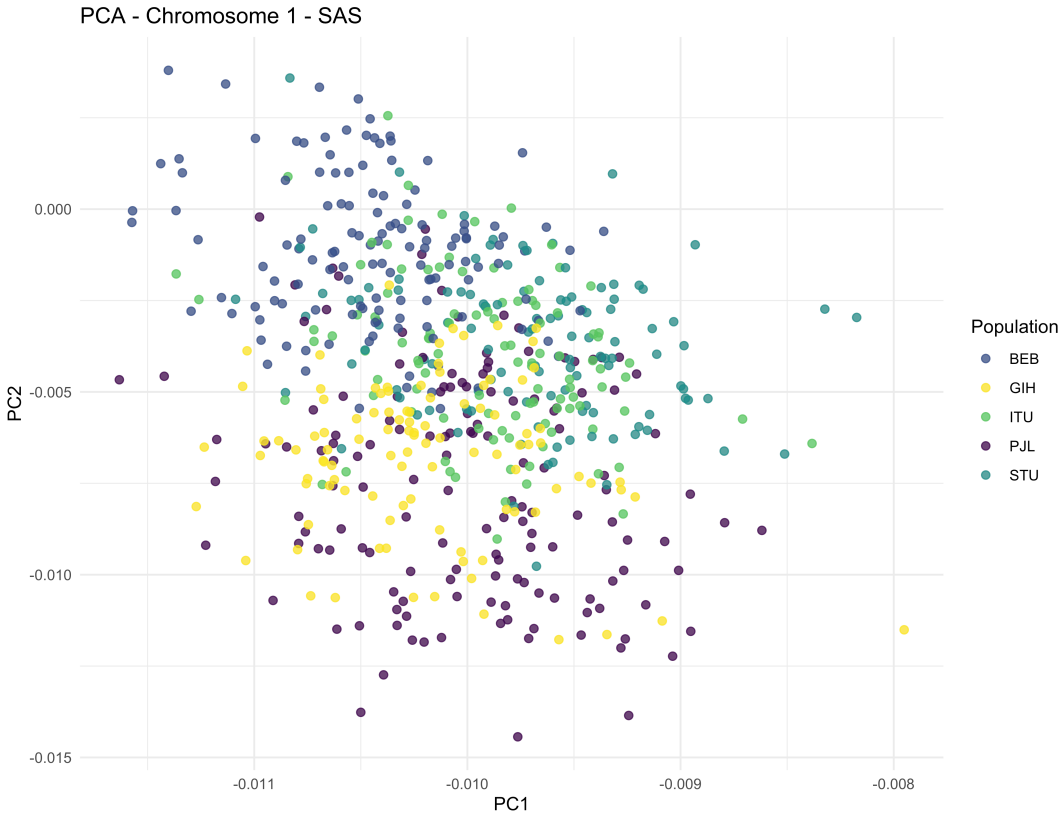
This PCA plot shows the variation present in Chromosome 1 of the European Ancestry superpopulation (EUR). This superpopulation includes the populations: British (GBR), Finnish (FIN), Iberian (IBS), CEPH (CEU), Toscani (TSI).



This PCA plot shows the variation present in Chromosome 1 of the East Asian Ancestry superpopulation (EAS). This superpopulation includes the populations: Southern Han Chinese (CHS), Dai Chinese (CDX), Kinh Vietnamese (KHV), Han Chinese (CHB), Japanese (JPT).



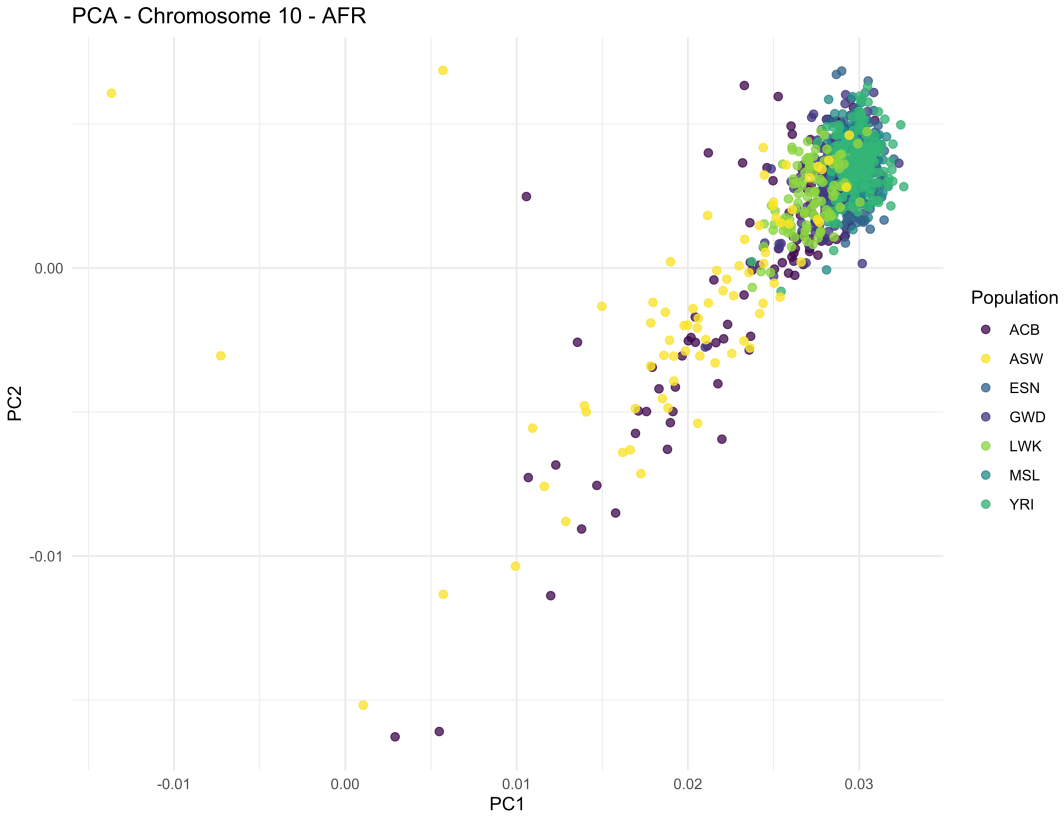
This PCA plot shows the variation present in Chromosome 1 of the American Ancestry superpopulation (AMR). This superpopulation includes the populations: Puerto Rican (PUR), Colombian (CLM), Peruvian (PEL), Mexican Ancestry (MXL).



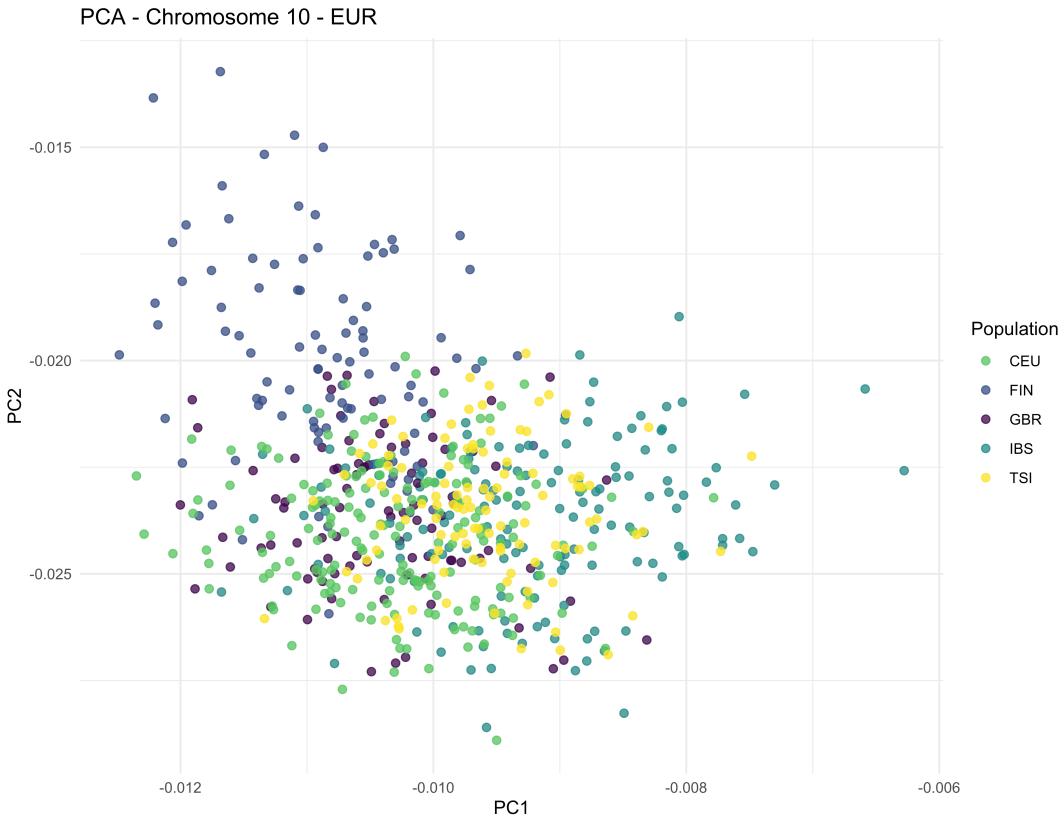
This PCA plot shows the variation present in Chromosome 1 of the South Asian Ancestry superpopulation (SAS). This superpopulation includes the populations: Punjabi (PJL), Bengali (BEB), Tamil (STU), Telugu (ITU), Gujarati (GIH).

Hide

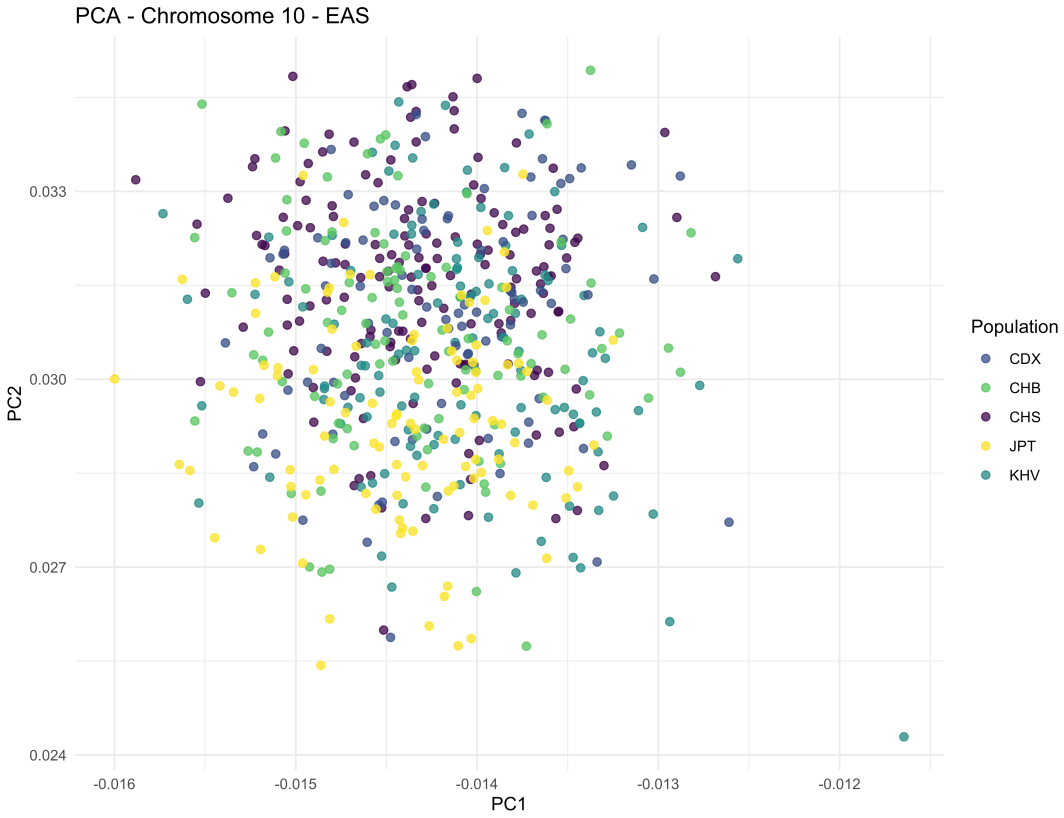
```
#Chromosome 10 plots
for(pop in names(chr10_dfs)){
  df <- chr10_dfs[[pop]]

  pops_in_group <- unique(df$`Population code`)
  pop_colors <- setNames(viridis(length(pops_in_group)), pops_in_group)

  legend_text <- create_legend_text(pop, chromosome_number = 10, pops_in_order = pops_in_group)

  p <- ggplot(df, aes(x = PC1, y = PC2, color = `Population code`)) +
    geom_point(size = 2, alpha = 0.8) +
    scale_color_manual(values = pop_colors) +
    labs(title = paste("PCA - Chromosome 10 -", pop),
         x = "PC1",
         y = "PC2",
         color = "Population",
         caption = legend_text) +
    theme_minimal() +
    theme(plot.caption = element_text(hjust = 0, size = 10),

          # Add extra margin around plot to prevent clipping
          plot.margin = margin(t = 15, r = 10, b = 10, l = 10))

  print(p)
}
```



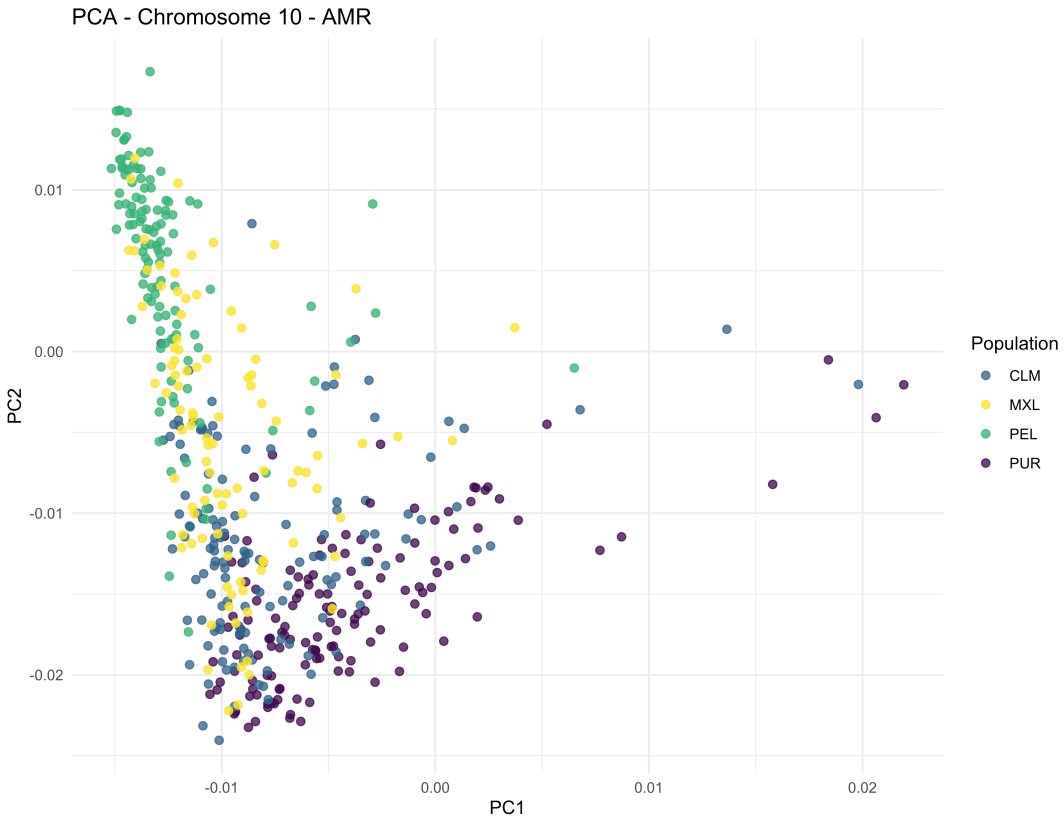
This PCA plot shows the variation present in Chromosome 10 of the African Ancestry superpopulation (AFR). This superpopulation includes the populations: African Caribbean (ACB), Gambian Mandinka (GWD), Esan (ESN), Mende (MSL), Yoruba (YRI), Luhya (LWK), African Ancestry SW (ASW).



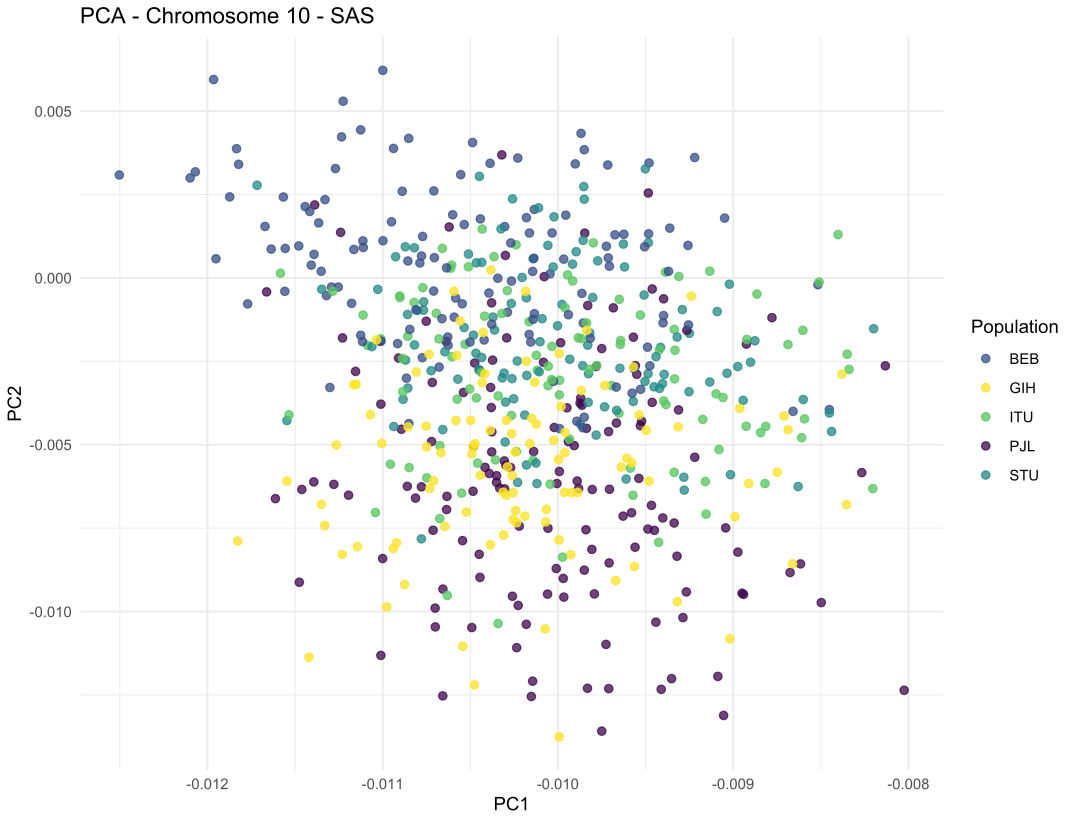
This PCA plot shows the variation present in Chromosome 10 of the European Ancestry superpopulation (EUR). This superpopulation includes the populations: British (GBR), Finnish (FIN), Iberian (IBS), CEPH (CEU), Toscani (TSI).



This PCA plot shows the variation present in Chromosome 10 of the East Asian Ancestry superpopulation (EAS). This superpopulation includes the populations: Southern Han Chinese (CHS), Dai Chinese (CDX), Kinh Vietnamese (KHV), Han Chinese (CHB), Japanese (JPT).



This PCA plot shows the variation present in Chromosome 10 of the American Ancestry superpopulation (AMR). This superpopulation includes the populations: Puerto Rican (PUR), Colombian (CLM), Peruvian (PEL), Mexican Ancestry (MXL).



This PCA plot shows the variation present in Chromosome 10 of the South Asian Ancestry superpopulation (SAS). This superpopulation includes the populations: Punjabi (PJL), Bengali (BEB), Tamil (STU), Telugu (ITU), Gujarati (GIH).

Part 4

Code

1. Obtain sample ID text file from Populations.tsv file

Hide

```
# Read the Populations.tsv file
pop_data <- read.table("Populations.tsv", header = TRUE, sep = "\t", stringsAsFactors = FALSE)

# Filter for PJL and extract the sample IDs
Sample_IDs <- pop_data$Sample.name[pop_data$'Population.code' == "PJL"]

# Write to file
writeLines(Sample_IDs, "sample_ids.txt")
```

2. Extract biallelic monomorphic snps for all the samples in the assigned focal population (PJL)

Hide

```
# Path to sample list
sample_file="$HOME/Assessment_GNA5012/R2/sample_ids.txt"

for chr in {1..22}; do
    vcf_in="CCDG_14151_B01_GRM_WGS_2020-08-05_chr${chr}.filtered.shapeit2-duohmm-phased.vcf.gz"
    vcf_out="$HOME/Assessment_GNA5012/R2/chr${chr}_filtered_samples.vcf.gz"

    # Filter VCF for the samples in the file
    bcftools view -S "$sample_file" \
                  -v snps \
                  -m2 -M2 \
                  -Ou "$vcf_in" | \
    bcftools view -c 1 -Oz -o "$vcf_out"

    # Index the output
    bcftools index "$vcf_out"
done

# Go to the directory with filtered VCFs
cd $HOME/Assessment_GNA5012/R2

# Combine all chromosomes into a single VCF
bcftools concat -Oz \
    chr{1..22}_filtered_samples.vcf.gz \
    -o combined_filtered_samples.vcf.gz

# Index the combined VCF
bcftools index combined_filtered_samples.vcf.gz
```

3. Convert VCF to GDS

Hide

```
library (SNPRelate)
#snpgdsVCF2GDS("focal_samples.vcf.gz", "focal.gds")
```

4. Load the GDs file and undertake PCA, save results to RDS.

Hide

```
genofile_focal <- snpgdsOpen("focal.gds")
pca_focal <- snpgdsPCA(genofile_focal,num.thread=6)
```

```
Principal Component Analysis (PCA) on genotypes:
Excluding 0 SNP on non-autosomes
Excluding 182,196 SNPs (monomorphic: TRUE, MAF: NaN, missing rate: NaN)
    # of samples: 146
    # of SNPs: 14,407,566
    using 6 threads
    # of principal components: 32
PCA:    the sum of all selected genotypes (0,1,2) = 3532691990
CPU capabilities:
Mon Nov  3 20:06:01 2025    (internal increment: 3364)

[..................................................]  0%, ETC: ---
[==================================================] 100%, completed, 59s
Mon Nov  3 20:07:00 2025    Begin (eigenvalues and eigenvectors)
Mon Nov  3 20:07:00 2025    Done.
```

Hide

```
# Save PCA results to RDS
saveRDS(pca_focal, file = "pca_focal.rds")

# Close the files after use
snpgdsClose(genofile_focal)
```

5. Extract PCA results and create a dataframe.

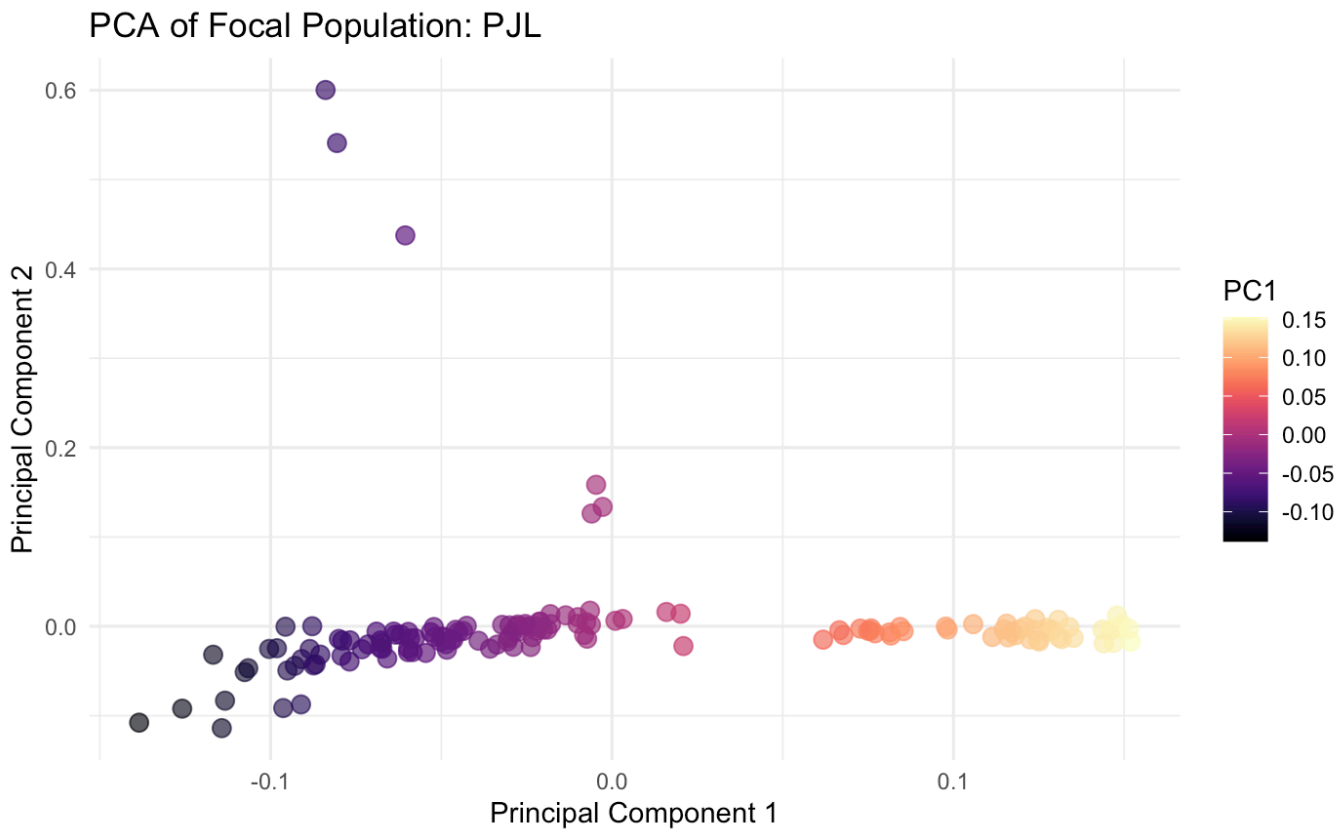
Hide

```
pca_focal <- readRDS("pca_focal.rds")

pca_focal.df <- data.frame(
  Sample = pca_focal$sample.id,
  PC1 = pca_focal$eigenvect[, 1],
  PC2 = pca_focal$eigenvect[, 2]
)
```

6. Create a GGplot for the variation observed in the focal population.

Hide

```
library(ggplot2)
library(viridis)
ggplot(pca_focal.df, aes(x = PC1, y = PC2, color = PC1)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_viridis(option = "magma") +
  theme_minimal() +
  labs(
    title = "PCA of Focal Population: PJL",
    x = "Principal Component 1",
    y = "Principal Component 2",
    color = "PC1"
  )
```



7. Subset the populations dataframe from Part 3 to only include chromosome 1 and 10 for the samples in the focal populations.

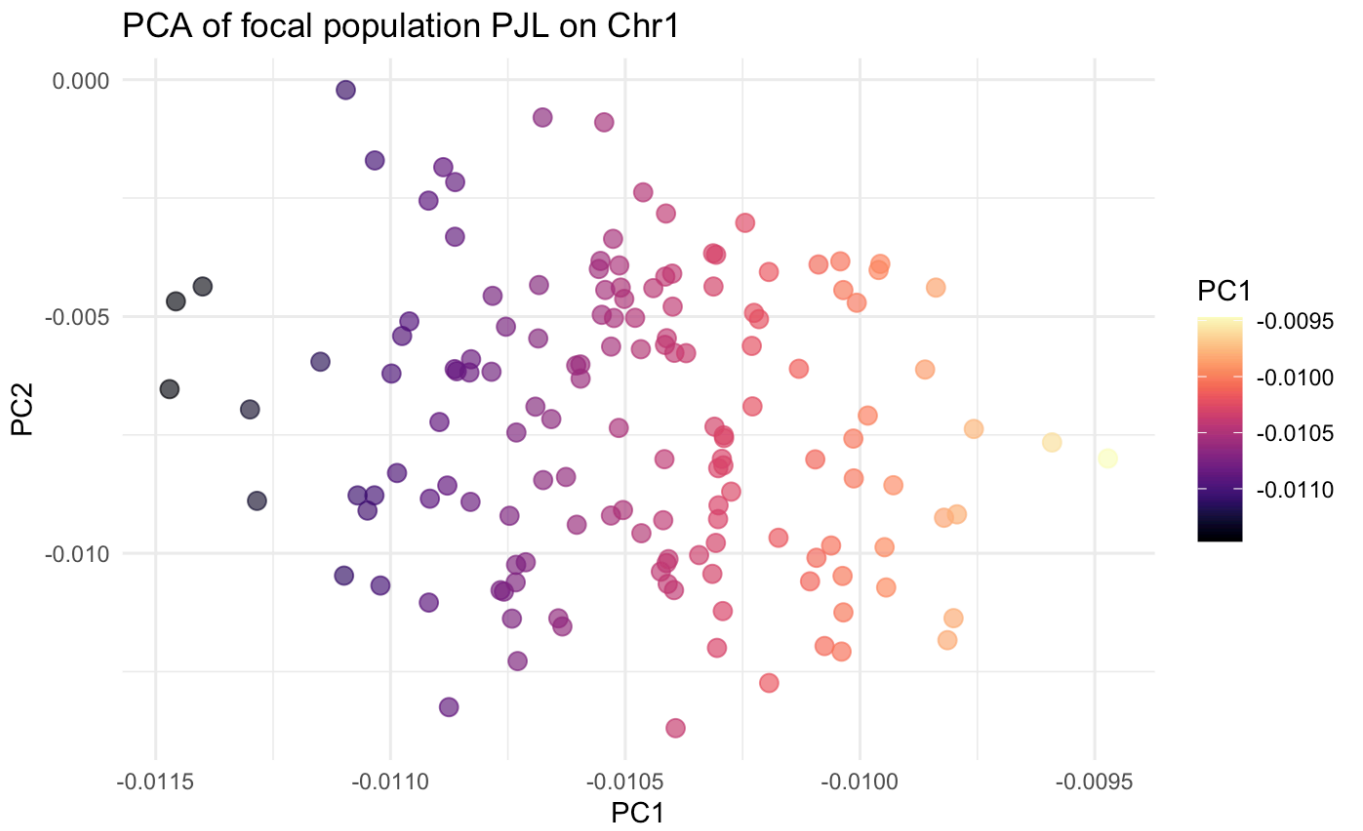
Hide

```
# Read your sample IDs from the text file
sample_ids <- readLines("sample_ids.txt")

# Subset PCA dataframes to only include those samples
focal_chr1 <- subset(pca_chr1.df, Sample %in% sample_ids)[, c("Sample", "PC1", "PC2")]
focal_chr10 <- subset(pca_chr10.df, Sample %in% sample_ids)[, c("Sample", "PC1", "PC2")]
```

8. Create a GGplot for only Chromosomes 1 and 10 for the focal samples extracted from the total population samples.

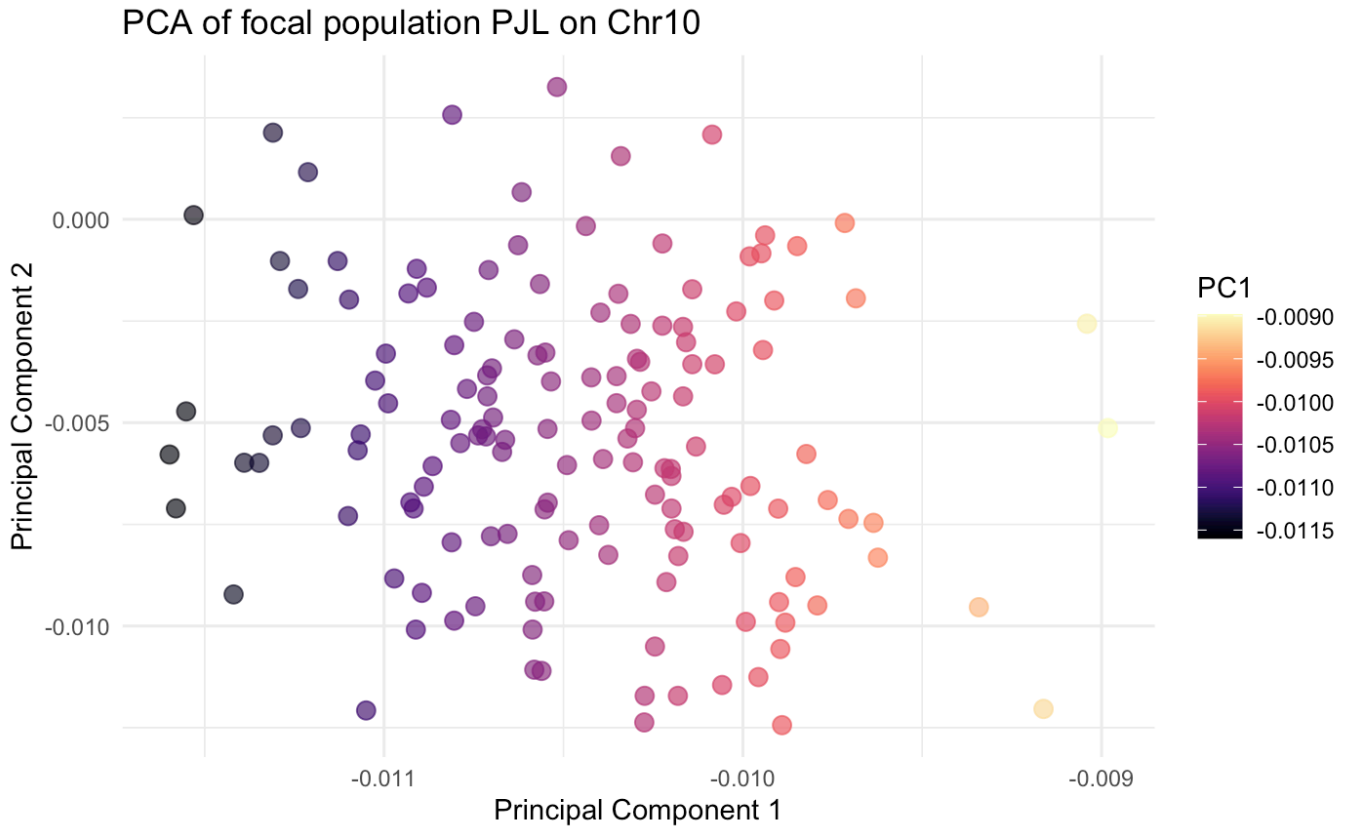
Hide

```
# Load libraries
library(viridis)
library(ggplot2)

# Plot PCA for Chr1
ggplot(focal_chr1, aes(x = PC1, y = PC2, color = PC1)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_viridis(option = "magma") +
  theme_minimal() +
  labs(title = "PCA of focal population PJL on Chr1", x = "PC1", y = "PC2", color = "PC1")
```



Hide

```
# Plot PCA for Chr10
ggplot(focal_chr10, aes(x = PC1, y = PC2, color = PC1)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_viridis(option = "magma") +
  theme_minimal() +
  labs(title = "PCA of focal population PJL on Chr10",
       x = "Principal Component 1",
       y = "Principal Component 2",
       color = "PC1")
```



Part 5

Code

1. Running snpgdsIBDKING to estimate kinship values.

Hide

```
#Open GDS file
focalgenofile <- snpgdsOpen("focal.gds")

# Calculate KING-robust kinship for all samples
king_focal <- snpgdsIBDKING(
  gdsobj = focalgenofile,
  sample.id = NULL,       # NULL means all samples
  snp.id = NULL,          # NULL means use all SNPs
  autosome.only = TRUE,   # exclude X/Y chromosomes
  maf = 0.01,             # filter low MAF
  missing.rate = 0.05,    # only use SNPs with missing rate <= 0.05
  type = "KING-robust",   # robust to population stratification
)
```

```
IBD analysis (KING method of moment) on genotypes:
Excluding 0 SNP on non-autosomes
Excluding 4,613,550 SNPs (monomorphic: TRUE, MAF: 0.01, missing rate: 0.05)
    # of samples: 146
    # of SNPs: 9,976,212
    using 1 thread
No family is specified, and all individuals are treated as singletons.
Relationship inference in the presence of population stratification.
KING IBD:    the sum of all selected genotypes (0,1,2) = 2253710464
CPU capabilities:
Sun Nov  2 23:08:15 2025    (internal increment: 65536)

[..................................................]  0%, ETC: ---
[==================================================] 100%, completed, 14s
Sun Nov  2 23:08:29 2025    Done.
```

Hide

```
#Close the files after use
snpgdsClose(focalgenofile)
```

2. Creating a matrix with row and column names corresponding to sample ID

Hide

```
samples <- read.table("sample_ids.txt", stringsAsFactors = FALSE)$V1
kin_mat <- as.matrix(king_focal$kinship)
rownames(kin_mat) <- samples
colnames(kin_mat) <- samples
```

3. Filtering the kinship matrix to include all values about 0.05 and removing self and duplicate pairs.

Hide

```
filtered_focal_df <- which(kin_mat > 0.05, arr.ind = TRUE)
filtered_focal_df <- data.frame(
  sample1 = rownames(kin_mat)[filtered_focal_df[,1]],
  sample2 = colnames(kin_mat)[filtered_focal_df[,2]],
  kinship = kin_mat[filtered_focal_df]
)

# Remove self-pairs and duplicate pairs
filtered_focal_df <- filtered_focal_df[filtered_focal_df$sample1 != filtered_focal_df$sample2, ]
filtered_focal_df <- filtered_focal_df[!duplicated(t(apply(filtered_focal_df[, c("sample1", "sample2")], 1, sor
t))), ]

nrow (filtered_focal_df)
```

```
[1] 85
```

Hide

```
head(filtered_focal_df)
```

	sample1 <chr>	sample2 <chr>	kinship <dbl>
4	HG02495	HG02493	0.2485288
6	HG02491	HG02492	0.2530563
7	HG02490	HG02492	0.2394506
9	HG02650	HG02649	0.2403393
11	HG02601	HG02602	0.2526316
12	HG02600	HG02602	0.2521799

6 rows

4. Creating a trio table that matches up the kinship information with the sample information table.

Hide

```
# Read the sample info (space-separated)
sample_info <- read.table("SampleInformation.tsv", header = FALSE, sep = "", stringsAsFactors = FALSE)

# Assign column names
colnames(sample_info) <- c("FamilyID", "SampleID", "FatherID", "MotherID", "Sex", "Population", "Superpopulatio
n")

# Filter only the PJL population
sample_info_pjl <- subset(sample_info, Population == "PJL")

# Keep only rows where both parents are listed (non-zero)
trios_pjl <- subset(sample_info_pjl, FatherID != "0" & MotherID != "0")

# Make a simple table
trio_table_pjl <- trios_pjl[, c("SampleID", "FatherID", "MotherID")]
colnames(trio_table_pjl) <- c("Child", "Father", "Mother")

# Save trio table to CSV
write.csv(trio_table_pjl, "trios_PJL.csv", row.names = FALSE)

# View first few rows
head(trio_table_pjl)
```

	Child <chr>	Father <chr>	Mother <chr>
1220	HG02492	HG02490	HG02491
1223	HG02495	HG02493	HG02494
1278	HG02602	HG02600	HG02601
1281	HG02605	HG02603	HG02604
1308	HG02650	HG02648	HG02649
1311	HG02653	HG02651	HG02652

6 rows

Part 6

Code

1. Extract, filter for variants with quality depth >30 and merge HG02602 (child), HG02601(mother), and HG02600 (father) high variant quality VCF.

Hide

```
#Extraction

for chr in {1..22}; do
    vcf_in="CCDG_14151_B01_GRM_WGS_2020-08-05_chr${chr}.filtered.shapeit2-duohmm-phased.vcf.gz"
    vcf_out="$HOME/Assessment_GNA5012/R6/HG02600/chr${chr}_HG02600.vcf.gz"
    bcftools view -s HG02600 \
        -Ou "$vcf_in" | \
    bcftools view -Oz -o "$vcf_out"
    bcftools index "$vcf_out"
done

# Combine all chromosomes into a single VCF and index
bcftools concat -Oz \
    chr{1..22}_HG02600.vcf.gz \
    -o HG02600_merged.vcf.gz
bcftools index HG02600_merged.vcf.gz

for chr in {1..22}; do
    vcf_in="CCDG_14151_B01_GRM_WGS_2020-08-05_chr${chr}.filtered.shapeit2-duohmm-phased.vcf.gz"
    vcf_out="$HOME/Assessment_GNA5012/R6/HG02601/chr${chr}_HG02601.vcf.gz"

    bcftools view -s HG02601 \
                -Ou "$vcf_in" | \
    bcftools view -Oz -o "$vcf_out"
    bcftools index "$vcf_out"
done

# Combine all chromosomes into a single VCF and index
bcftools concat -Oz \
    chr{1..22}_HG02601.vcf.gz \
    -o HG02601_merged.vcf.gz
bcftools index HG02601_merged.vcf.gz

for chr in {1..22}; do
    vcf_in="CCDG_14151_B01_GRM_WGS_2020-08-05_chr${chr}.filtered.shapeit2-duohmm-phased.vcf.gz"
    vcf_out="$HOME/Assessment_GNA5012/R6/HG02602/chr${chr}_HG02602.vcf.gz"

    bcftools view -s HG02602 \
                -Ou "$vcf_in" | \
    bcftools view -Oz -o "$vcf_out"
    bcftools index "$vcf_out"
done

# Combine all chromosomes into a single VCF and index
bcftools concat -Oz \
    chr{1..22}_HG02602.vcf.gz \
    -o HG02602_merged.vcf.gz
bcftools index HG02602_merged.vcf.gz

#Quality filtering
bcftools filter -i 'INFO/QD>30' HG02602_merged.vcf.gz -Oz -o HG02602_QD30.vcf.gz
bcftools index HG02602_QD30.vcf.gz

bcftools filter -i 'INFO/QD>30' HG02601_merged.vcf.gz -Oz -o HG02601_QD30.vcf.gz (for all 3)
bcftools index HG02601_QD30.vcf.gz

bcftools filter -i 'INFO/QD>30' HG02600_merged.vcf.gz -Oz -o HG02600_QD30.vcf.gz (for all 3)
bcftools index HG02600_QD30.vcf.gz

#Merging
bcftools merge -Oz -o trio_merged.vcf.gz HG02602_QD30.vcf.gz HG02601_QD30.vcf.gz HG02600_QD30.vcf.gz
```

2. Set input and output files

Hide

```
#library (VariantAnnotation)

# Input and output files
vcf_file <- "trio_merged.vcf.gz"
rds_file <- "trio_merged.vcf.rds"

# Read the merged trio VCF
vcf <- readVcf(vcf_file, genome = "hg19")
```

3. Extract genotype data perform filtering to identify de novo variants

Hide

```
# Extract genotype data
geno_data <- geno(vcf)$GT  # Genotype matrix: variants x samples

# Step 1: Extract genotypes as vectors
child_gt  <- as.vector(geno_data[, "HG02602"])
mother_gt <- as.vector(geno_data[, "HG02601"])
father_gt <- as.vector(geno_data[, "HG02600"])

# Step 2: Remove variants with any missing genotypes as we are not confident regarding them
non_missing <- !(child_gt == "./." | mother_gt == "./." | father_gt == "./.")
child_gt  <- child_gt[non_missing]
mother_gt <- mother_gt[non_missing]
father_gt <- father_gt[non_missing]
geno_non_missing <- geno_data[non_missing, ]

# Step 3: Remove parental heterozygotes / non-unique variants
valid_parents <- !(mother_gt %in% c("0/1","1/0","1/1","0|1","1|0","1|1") |
                   father_gt %in% c("0/1","1/0","1/1","0|1","1|0","1|1"))

child_gt  <- child_gt[valid_parents]
mother_gt <- mother_gt[valid_parents]
father_gt <- father_gt[valid_parents]
geno_trio_filtered <- geno_non_missing[valid_parents, ]

nrow(geno_trio_filtered)
```

```
[1] 234
```

Hide

```
head(geno_trio_filtered)
```

```
                                                                      HG02602 HG02601 HG02600
1:11160078:CATTTATTT:C;1:11160079:C:CATTTATTTA;1:11160075:C:CATTTATTT   "0|1"   "0|2"   "0|3"
1:11488639:CTG:C;1:11488640:C:CTGTGTG                                   "1|0"   "0|2"   "0|2"
1:15789666:A:ATGTGTG;1:15789669:A:ATGTGTGTGT;1:15789664:A:ATGTGTGTG     "0|1"   "2|0"   "3|0"
1:17449694:TAATAAATAA:T;1:17449696:TAATAAATAA:T;1:17449698:TAATAAATAA:T "1|0"   "0|2"   "0|3"
1:26082705:GCTCTCT:G;1:26082707:GCT:G;1:26082704:G:GCT                  "0|1"   "2|0"   "3|0"
1:28092166:AAAT:A;1:28092168:A:AAATAATAAT;1:28092164:AAATAAT:A          "0|1"   "2|0"   "0|3"
```

4. Create a dataframe containing the de novo variants that can ve exported as a TSV

Hide

```
# Subset the VCF to the filtered variants
vcf_filtered <- vcf[rownames(geno_trio_filtered)]

# Create a data frame for export
tsv_df <- data.frame(
  CHROM = as.character(seqnames(rowRanges(vcf_filtered))),
  POS   = start(rowRanges(vcf_filtered)),
  REF   = as.character(ref(vcf_filtered)),
  ALT   = sapply(alt(vcf_filtered), function(x) paste0(x, collapse=",")),
  CHILD_GT  = child_gt,
  MOTHER_GT = mother_gt,
  FATHER_GT = father_gt
)

tsv_df$VARIANT_TYPE <- ifelse(
  nchar(tsv_df$REF) == 1 & nchar(tsv_df$ALT) == 1, "SNV",
  "INDEL/CNV"
)

# Save final TSV
write.table(tsv_df, file = "geno_trio_filtered_with_type.tsv",
            sep = "\t", quote = FALSE, row.names = FALSE)

nrow(tsv_df)
```

```
[1] 234
```

Hide

```
head(tsv_df)
```

CH... <chr>	POS <int>	REF <chr>	ALT <chr>	CHILD_... <chr>
1 chr1	11160075	CATTTATTT	C,CATTTATTTATTTATTTATTTATTT,CATTTATTTATTTATTT	0|1
2 chr1	11488637	CTG	C,CTGTGTGTG	1|0
3 chr1	15789664	A	ATGTGTG,ATGTGTGTGTGTG,ATGTGTGTG	0|1
4 chr1	17449693	TAATAAATAAATAAATAAATA	T,TAATA,TAATAAATA	1|0
5 chr1	26082703	GCTCTCT	G,GCTCT,GCTCTCTCT	0|1
6 chr1	28092164	AAATAAT	AAAT,AAATAATAATAATAAT,A	0|1

6 rows | 1-6 of 8 columns

Hide

```
NA
```

Hide

```
#The genotypes of the child and parents are not visible in the pdf due to spacing issues, hence I have created an
other dataframe that hides the REF and ALT columns, so the genotypes can be viewed.

tsv_df_concise <- data.frame(
  CHROM = as.character(seqnames(rowRanges(vcf_filtered))),
  POS   = start(rowRanges(vcf_filtered)),
  CHILD_GT  = child_gt,
  MOTHER_GT = mother_gt,
  FATHER_GT = father_gt
)

tsv_df_concise$VARIANT_TYPE <- ifelse(
  nchar(tsv_df$REF) == 1 & nchar(tsv_df$ALT) == 1, "SNV",
  "INDEL/CNV"
)

nrow(tsv_df_concise)
```

```
[1] 234
```

Hide

```
head(tsv_df_concise)
```

	CHROM <chr>	POS <int>	CHILD_GT <chr>	MOTHER_GT <chr>	FATHER_GT <chr>	VARIANT_TYPE <chr>
1	chr1	11160075	0|1	0|2	0|3	INDEL/CNV
2	chr1	11488637	1|0	0|2	0|2	INDEL/CNV
3	chr1	15789664	0|1	2|0	3|0	INDEL/CNV
4	chr1	17449693	1|0	0|2	0|3	INDEL/CNV
5	chr1	26082703	0|1	2|0	3|0	INDEL/CNV
6	chr1	28092164	0|1	2|0	0|3	INDEL/CNV

6 rows