Assignment 3

Group 28: Mona Behrouzian, Traye Lin, Maya Ansu

November 18, 2023

Q1. Sequencing technologies

Why are areas of the genome with high GC content are hard to sequence?

If a genome has high GC content (i.e. high amount of guanine and cytosine base pairs) then the melting temperature is high as well (due to forming more secondary structures that are very stable). This means it is harder to separate the 2 strands, thus it is harder to do PCR denaturation. If the strands don't separate, they won't be amplified and read in NGS. Additionally, high GC content in the entire genome influences fragment count, making these specific fragments underrepresented. Signal strength decreases as sequencing continues with high GC.

Source: Benjamini Y, Speed TP. Summarizing and correcting the GC content bias in high-throughput sequencing. Nucleic Acids Res. 2012 May;40(10):e72. doi: 10.1093/nar/gks001. Epub 2012 Feb 9. PMID: 22323520; PMCID: PMC3378858.

Q2. Global alignment exercise

Similar to the approach for Needleman–Wunsch algorithm, find the best global alignment between the two following sequences: ATTCGAC and ATCAC.

Please see merged PDF on final pages

Q3. Looking at the Metadata of an alignment (SAM) file

Q3.1. According to the header table in section 1.3 of the BAM/SAM document in the appendix, what do the SN and LN tags indicate?

The SN tags indicate the reference sequence name. They all must be distinct. The LN tags indicate the reference sequence length. It has a range of [1, (2e31) - 1]

Q3.2. What is the length of the X chromosome, in bp, for our alignment?

The length of the X chromosome is 171031299

```
x_length <- RNAseq$V3[RNAseq$V2 == "SN:X"]
x_length
```

```
## [1] "LN:171031299"
```

Q4. Looking at the Reads of an alignment (SAM) file

Q4.1. How many reads are there in this BAM file?

There are 146,346 reads in this BAM file.

```
# load the reads into an R dataframe. Each row contains one read.
sam <- read.csv("single_cell_RNA_seq_bam.sam", sep = "\t", header = FALSE,
    comment.char = "@", col.names = paste0("V", seq_len(30)), fill = TRUE)
sam <- sam[paste0("V", seq_len(11))]

# dim(sam) head(sam)
print(nrow(sam))
```

```
## [1] 146346
```

Q4.2.1 Which column of your dataframe should you look at to find the chromosome to which the read was aligned?

Column 3 shows the "RNAME" which would show us the chromosome to which the read was aligned.

4.2.2 To which BAM data field does the dataframe column "V11" correspond?

Column 11 corresponds to the "QUAL" tag, or the base quality, in ASCII format.

```
# Print out the 10th row of the dataframe to look at the format of a
# read.
sam[10, ]
```

```
##                                          V1 V2 V3      V4  V5  V6 V7 V8 V9
## 10 NS500668:199:HV73CBGX2:1:11203:20546:3351 16  1 3365976 255 58M  *  0  0
##                                                                    V10
## 10 AATCAAAAAGGGGGCTGTCAGTAGGATGATATAAGATATAGATGTAGTTTATCTCCTA
##                                                                    V11
## 10 EEEEEEEEEEA//AAAAEEEEEE/AEEAEAEEEEEEEEEEEEEAAEE///EEEAAA6A
```

```
# Compare it to the mandatory BAM fields table in section 1.4 of the
# SAM/BAM documentation in the appendix.  The order of columns in the
# bam file have been preserved in the dataframe.
```

Q4.3. How many reads in this file align to chromosome X?

There are 5,999 reads in this file that align to chromosome X.

```
Xchrom <- which(sam$V3 == "X")  #this is giving the indices of elements that satisfy the condition
print(length(Xchrom))
```

```
## [1] 5999
```

```
# head(Xchrom)
```

Q4.4. What is the mean base quality (BQ) for reads aligning to chromosome X?

32.7 is the mean BQ for reads aligning to chrom X.

```
Xchrom <- sam[sam$V3 == "X", ]

bq_int_x <- sapply(Xchrom$V11, utf8ToInt) - 33

print(mean(bq_int_x))
```

```
## [1] 32.72349
```

Q4.5. Plot the distribution of BQs across all bases and reads as a boxplot. Comment on your observation.

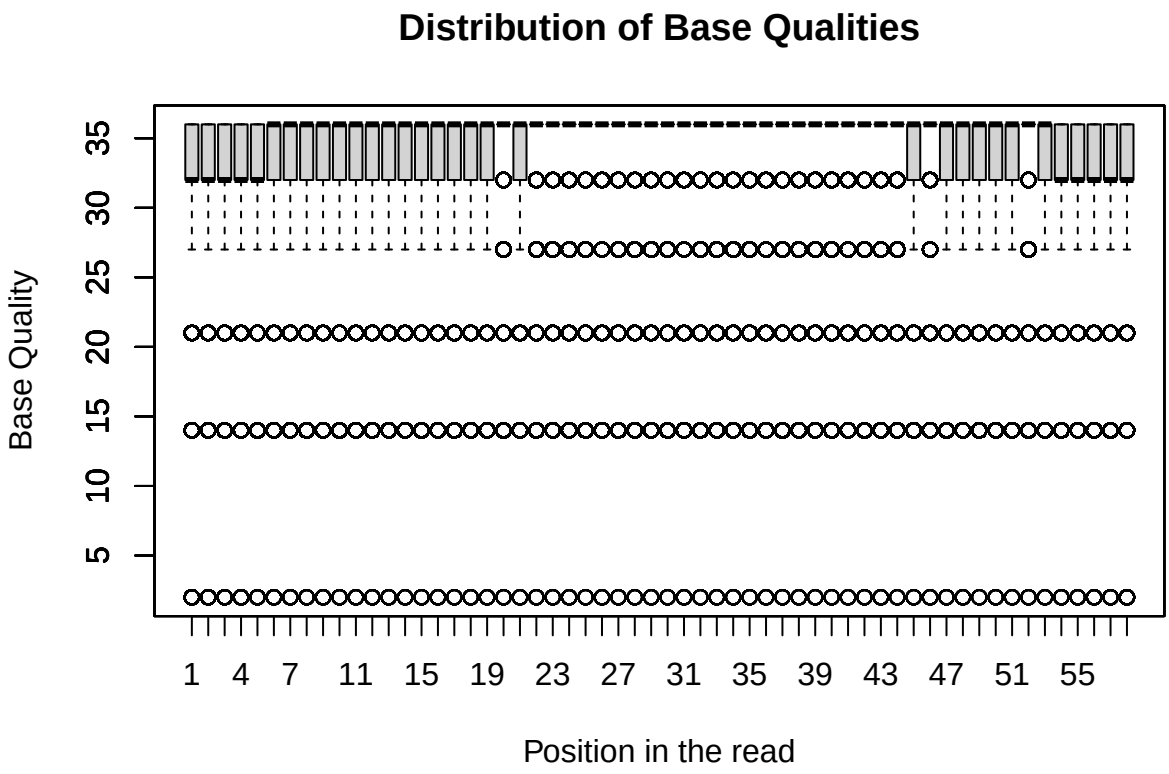
We observe that there is a good per base sequence quality, throughout all the positions. This shows us that we likely do not need to trim any of the data

```
bq_int <- sapply(sam$V11, utf8ToInt) - 33
bq_int <- t(bq_int)

# dim(bq_int)

# We got help on our code from the TA! (Tina) :)
boxplot(bq_int[, 1:10], at = 1:10, xlim = c(1, 58), xaxt = "n", main = "Distribution of Base Qualities"
    xlab = "Position in the read", ylab = "Base Quality")
boxplot(bq_int[, 11:20], at = 11:20, add = TRUE, xaxt = "n")
boxplot(bq_int[, 21:30], at = 21:30, add = TRUE, xaxt = "n")
boxplot(bq_int[, 31:40], at = 31:40, add = TRUE, xaxt = "n")
boxplot(bq_int[, 41:50], at = 41:50, add = TRUE, xaxt = "n")
boxplot(bq_int[, 51:58], at = 51:58, add = TRUE, xaxt = "n")

axis(1:58, at = 1:58)
```



Q4.6. Referring to section 1.4 of the SAM/BAM documentation, what column contains the leftmost

mapping position of the reads? *Column 4, with the POS tag.*

Q4.7. How many reads have their leftmost mapping position aligned within these coordinates?

119 reads.

```
reads_indices <- which(sam$V3 == "9" & sam$V4 >= 40801273 & sam$V4 <= 40805199)
reads_pos <- sam[reads_indices, 4]

length(reads_pos)
```

```
## [1] 119
```

Q4.8. How many reads have mapping quality less than 50?

61,527 reads.

```
mapq_less_50 <- which(sam$V5 < 50)
length(mapq_less_50)
```

```
## [1] 61527
```

Q4.9. What is the mean mapping quality of the reads which have mapping quality less than 50?

0.2418125 is the mean mapping quality.

```
reads_mapq_less_50 <- sam$V5[mapq_less_50]
mean(reads_mapq_less_50)
```

```
## [1] 0.2418125
```

Q4.10. (bonus): The genome of the mouse used in this experiment has been edited to include the

DNA sequence for the protein 'tdTomato', which is a fluorophore. Count the number of reads which align to the tdTomato sequence. Assuming that these reads are accurate, would you expect this cell to emit fluorescently? What might be the purpose of modifying a genome to include a fluorophore? Hint: Think about studying cell populations under a microscope. *asdf*

Q5. Investigating the Variants

```
vcf_con <- file("RNA_seq_annotated_variants.vcf", open = "r")
vcf_file <- readLines(vcf_con)
close(vcf_con)
vcf <- data.frame(vcf_file)
header <- vcf[grepl("##", vcf$vcf_file), ]
# factor(header)
variants <- read.csv("RNA_seq_annotated_variants.vcf", skip = length(header),
    header = TRUE, sep = "\t")
```

Q5.1 For the first variant (row) in the dataframe, what is the reference allele base at the site, and what is the alternative allele called by Strelka?

G and A

```
ref.allele <- variants$REF[1]
alt.allele <- variants$ALT[1]

cat("Reference Allele:", ref.allele, "\n")
```

```
## Reference Allele: G
```

```
cat("Alternative Allele:", alt.allele)
```

```
## Alternative Allele: A
```

Q5.2. Write code to obtain the entirety of the ANN info value contents from the INFO field for the first variant.

```
info_field <- as.character(variants$INFO[1])
info_variables <- strsplit(info_field, ";")[[1]]
ann_info <- info_variables[grep("ANN=", info_variables)]
ann_info
```

```
## [1] "ANN=A|intron_variant|MODIFIER|Sulf1|ENSMUSG00000016918|transcript|ENSMUST00000088585.9|protein_
```

Q5.3. Based on the ANN value of the first variant, what does the 'Annotation' field tell us about this variant?

The field tells us that this is an intronic variant.

```
# need to split up the annotation with commas
strsplit(as.character(ann_info), ",")
```

```
## [[1]]
## [1] "ANN=A|intron_variant|MODIFIER|Sulf1|ENSMUSG00000016918|transcript|ENSMUST00000088585.9|protein_
## [2] "A|intron_variant|MODIFIER|Sulf1|ENSMUSG00000016918|transcript|ENSMUST00000177608.7|protein_codi
## [3] "A|intron_variant|MODIFIER|Sulf1|ENSMUSG00000016918|transcript|ENSMUST00000180062.7|protein_codi
## [4] "A|intron_variant|MODIFIER|Sulf1|ENSMUSG00000016918|transcript|ENSMUST00000186051.6|protein_codi
## [5] "A|intron_variant|MODIFIER|Sulf1|ENSMUSG00000016918|transcript|ENSMUST00000187376.6|retained_int
## [6] "A|intron_variant|MODIFIER|Sulf1|ENSMUSG00000016918|transcript|ENSMUST00000186405.6|protein_codi
## [7] "A|intron_variant|MODIFIER|Sulf1|ENSMUSG00000016918|transcript|ENSMUST00000189541.6|protein_codi
```

Q5.4. Perform the parsing done in Q5.1-3 again on variant line 683. What gene would this variant affect?

Rps19 gene. We can see it is this gene from the annotations printed below.

```
ANN_683 = unlist(strsplit(as.character(variants$INFO)[683], "ANN="))[2]
strsplit(as.character(ANN_683), ",")
```

```
## [[1]]
##  [1] "A|frameshift_variant&splice_acceptor_variant&splice_region_variant&intron_variant|HIGH|Rps19|E
##  [2] "A|frameshift_variant&splice_acceptor_variant&splice_region_variant&intron_variant|HIGH|Rps19|E
##  [3] "A|frameshift_variant&splice_acceptor_variant&splice_region_variant&intron_variant|HIGH|Rps19|E
##  [4] "A|frameshift_variant&splice_acceptor_variant&splice_region_variant&intron_variant|HIGH|Rps19|E
##  [5] "A|frameshift_variant&splice_acceptor_variant&splice_region_variant&intron_variant|HIGH|Rps19|E
##  [6] "A|frameshift_variant&splice_acceptor_variant&splice_region_variant&intron_variant|HIGH|Rps19|E
##  [7] "A|splice_acceptor_variant&3_prime_UTR_variant&intron_variant|HIGH|Rps19|ENSMUSG00000040952|tra
```

```
##  [8] "A|splice_acceptor_variant&splice_region_variant&intron_variant&non_coding_transcript_exon_varia
##  [9] "A|splice_region_variant|LOW|Rps19|ENSMUSG00000040952|transcript|ENSMUST00000129847.7|nonsense_
## [10] "A|downstream_gene_variant|MODIFIER|Rps19|ENSMUSG00000040952|transcript|ENSMUST00000138662.1|re
## [11] "A|non_coding_transcript_exon_variant|MODIFIER|Rps19|ENSMUSG00000040952|transcript|ENSMUST00000
## [12] "A|non_coding_transcript_variant|MODIFIER|Rps19|ENSMUSG00000040952|transcript|ENSMUST0000012984
```

Q5.5. Within the entire VCF file, how many HIGH impact variants we see in total?

4 in total

```
high_impacts <- grep("HIGH", variants$INFO)

length(high_impacts)
```

```
## [1] 4
```

Q5.6. What is a frameshift variant? Does it have a greater or lesser effect on the resultant protein than a missense variant? Why?

A frameshift variant is a type of mutation that involves the insertion or deletion of nucleotides that changes the reading frame of a DNA sequence. It often results in a premature STOP codon. A frameshift variant is more likely to have more severe consequences than the a missense variant. This is because a frameshift variant has the ability to affect the entire sequence downstream of the mutation, causing a shift in the entire amino sequence during protein translation. However, a missense variant involves the substitution of only one nucleotide, resulting in the change of a single amino acid in the protein sequence.

Q5.7. We can divide variants into two broad categories: intronic/intergenic and exonic. Count the number of potential intronic variants AND intergenic! What do you notice about the number of intronic variants (compared to overall number of variants)?

We observe that a majority of the variants are of the intronic/intergenic category.

```
intronic_variants <- grep("intron_variant", variants$INFO)
intergenic_variants <- grep("intergenic_region", variants$INFO)

cat("Number of intronic variants:", length(intronic_variants), "\n")
```

```
## Number of intronic variants: 476
```

```
cat("Number of intergenic variants:", length(intergenic_variants), "\n")
```

```
## Number of intergenic variants: 130
```

```
total_interon_and_intergenic_variants <- length(intronic_variants) + length(intergenic_variants)
cat("Total number of intronic/intergenic:", total_interon_and_intergenic_variants,
    "\n")
```

```
## Total number of intronic/intergenic: 606
```

```
cat("Total number of variants:", nrow(variants), "\n")
```

```
## Total number of variants: 836
```

Q5.8. List all the genes that have been affected by coding mutations and have high impact. What do you find that is interesting?

As printed below, the genes affected are: Nufip2, Tmem45a, Rpl11, n-R5s193-Spsb1.

```
coding_mutations <- grep("protein_coding", variants$INFO)
coding_mutations_and_high_impact <- grep("HIGH", variants$INFO[coding_mutations])

info_coding <- as.character(variants$INFO[coding_mutations_and_high_impact])
info_coding <- strsplit(info_coding, ";")

# printing off the annotation
for (i in 1:length(info_coding)) {
    print(info_coding[[i]][grep("^ANN=", info_coding[[i]])])
}
```

```
## [1] "ANN=G|intron_variant|MODIFIER|Nufip2|ENSMUSG00000037857|transcript|ENSMUST00000181023.1|protein
## [1] "ANN=G|intron_variant|MODIFIER|Tmem45a|ENSMUSG00000022754|transcript|ENSMUST00000023435.5|protei
## [1] "ANN=T|missense_variant|MODERATE|Rpl11|ENSMUSG00000059291|transcript|ENSMUST00000102536.10|prote
## [1] "ANN=T|intergenic_region|MODIFIER|n-R5s193-Spsb1|ENSMUSG00000064798-ENSMUSG00000039911|intergeni
```

Q5.9. (bonus): Using Strelka on our data, we can detect indels, but only to a limited extent. Most of the reads in our BAM file have read lengths around 60bp long. Why might this have consequences for the detection of insertions that are longer than 60bp?

Due to the average read length being around 60bp long, it may be difficult to detect insertion longer than 60bp. This is because the variant caller may not have enough information to identify and characterize long insertions. Additionally, if the alignment algorithm relies on short reads it may be difficult to accurately align longer insertion to a reference genome.

Q5.10. In the form of a boxplot, plot the distribution of the VAFs across all the variants. How many variants have VAF > 5%? How many of those variants (the ones with >5% VAF) are in coding regions?

How many variants have VAF > 5%?
ANSWER: 422

How many of those variants (the ones with >5% VAF) are in coding regions? ANSWER: 377

```
# Note, we got lots of help from the TA (Tina) :)
VAF <- c()
```

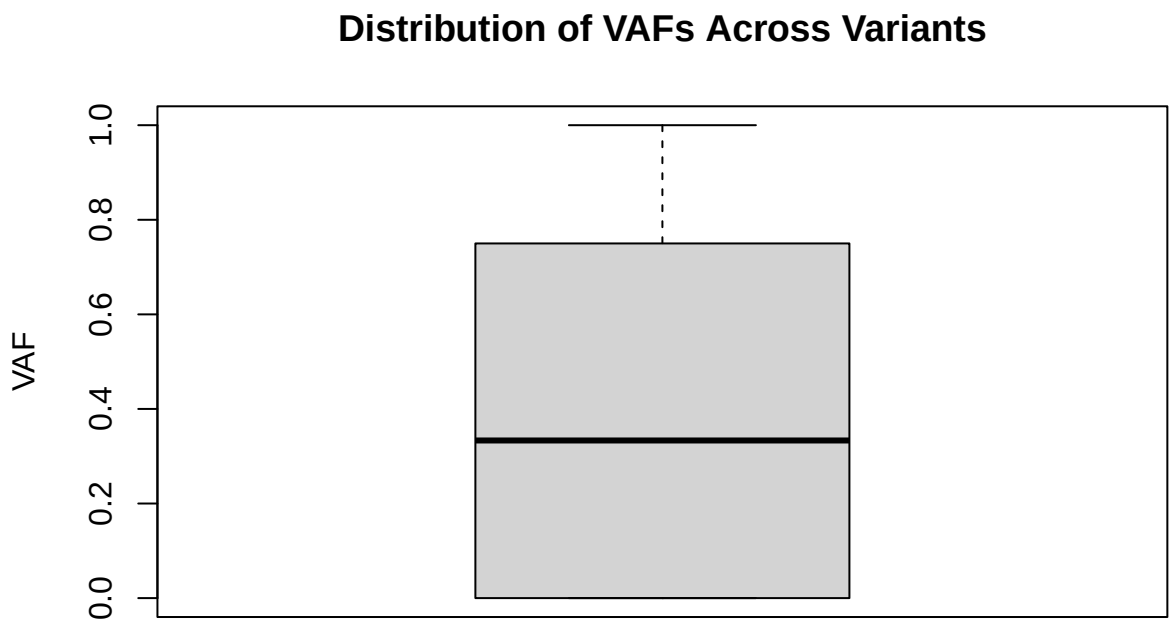
```r
for (i in 1:nrow(variants)) {
    # split by : for format and ws20171223...  unlist both of those
    format_info <- unlist(strsplit(as.character(variants$FORMAT[i]), ":"))
    ws_info <- unlist(strsplit(as.character(variants$ws20171223_MPs_tomatoMuscle8wkQuiescent201[i]),
        ":"))

    # use the AD index to pull the pair from ws_info
    AD_pair <- ws_info[grep("AD", format_info)[1]]

    # split the pair by ,
    AD_split <- as.integer(unlist(strsplit(as.character(AD_pair), ",")))

    # then do the math x/(x+y) (this is a %) store it in VAF
    VAF[i] <- AD_split[1]/(AD_split[1] + AD_split[2])
}

boxplot(VAF, main = "Distribution of VAFs Across Variants", ylab = "VAF")
```




```r
VAF_greater5 <- which(VAF >= 0.05)

cat("How many variants have VAF > 5%?", length(VAF_greater5), "\n")
```

```
## How many variants have VAF > 5%? 422
```

```
cat("How many of those variants (the ones with >5% VAF) are in coding regions?",
    length(VAF_greater5[grep("protein_coding", variants$INFO[VAF_greater5])]))
```

```
## How many of those variants (the ones with >5% VAF) are in coding regions? 377
```