Index Swapping Assignment Writeup

Amanda Leonti, Bi622

1

a) Generate per base call distribution of quality scores

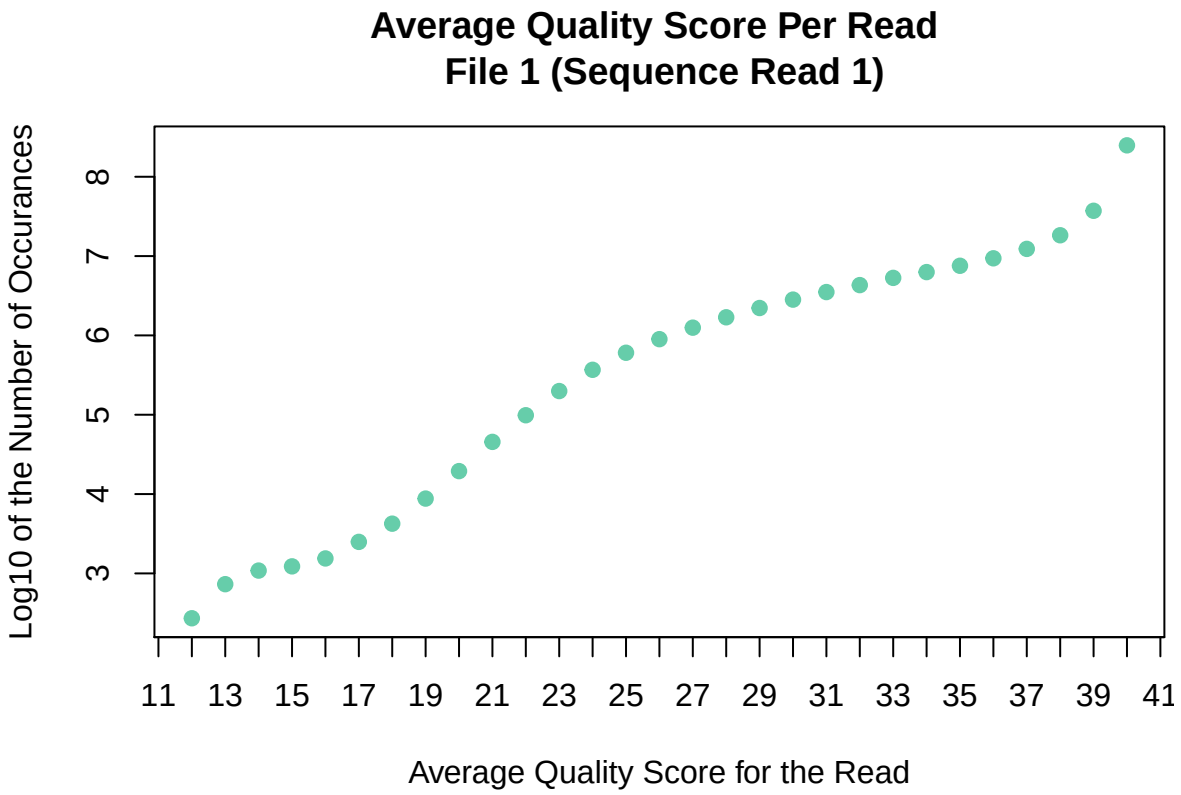
Generate a per nucleotide distribution as you did in part 1 of PS4 (in Leslie's AGAGTCCA class). Next, average the Quality scores for each read (for each of the four files) and plot frequency of the Quality Scores.

First, I need to read in my output files from the scripts I ran:

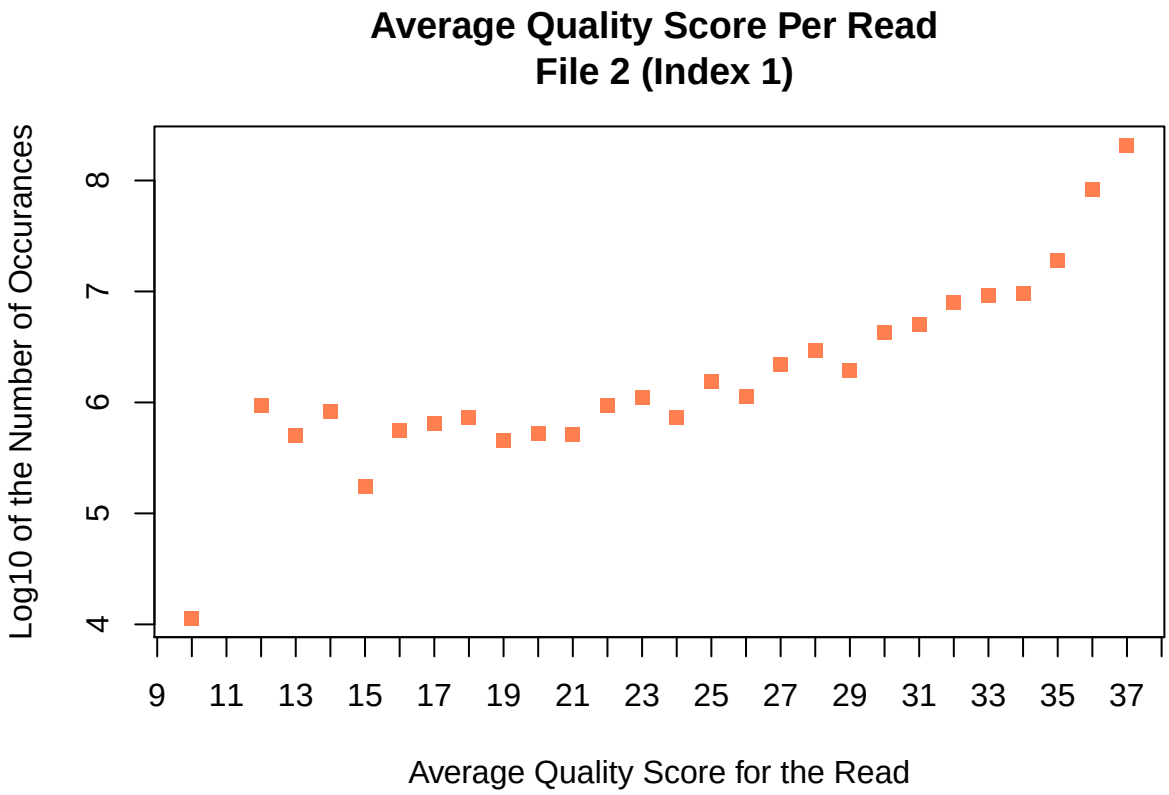
```
setwd("/Users/amanda/Bi622_2017/IndexSwapping/files_from_talapas")

# average Q-score for the entire read
r1_readavgs <- read.table("R1_ReadAvgs.tsv", header=T, sep = "\t" )
r2_readavgs <- read.table("R2_ReadAvgs.tsv", header=T, sep = "\t" )
r3_readavgs <- read.table("R3_ReadAvgs.tsv", header=T, sep = "\t" )
r4_readavgs <- read.table("R4_ReadAvgs.tsv", header=T, sep = "\t" )


# scp aleonti@talapas-ln1.uoregon.edu:/home/aleonti/indexhopping/output_files/qscore_distributions/R3_q

# average Q-score per position
r1_qdist <- read.table("R1_qscore_distrib.tsv", header=T, sep = "\t" )
colnames(r1_qdist) <- c("Basepair_Position", "Average_QScore")

r2_qdist <- read.delim("R2_qscore_distrib.tsv", header=T, sep = "\t" )
colnames(r2_qdist) <- c("Basepair_Position", "Average_QScore")

r3_qdist <- read.table("R3_qscore_distrib.tsv", header=T, sep = "\t" )
colnames(r3_qdist) <- c("Basepair_Position", "Average_QScore")

r4_qdist <- read.table("R4_qscore_distrib.tsv", header=T, sep = "\t" )
colnames(r4_qdist) <- c("Basepair_Position", "Average_QScore")
```

Then I can make my plots:

```
# Sequence 1 Read Means
#par(mfrow=c(2,2))

plot(r1_readavgs$Quality.Score, log10(r1_readavgs$Frequency.of.Occurance),
     main = "Average Quality Score Per Read \n File 1 (Sequence Read 1)",
     xlab = "Average Quality Score for the Read",
     ylab = "Log10 of the Number of Occurances",
     col = "aquamarine3",
     cex = 1,
     pch = 19,
     xaxt="n")

axis(1, at=c(0:42), labels = c(0:42))
```

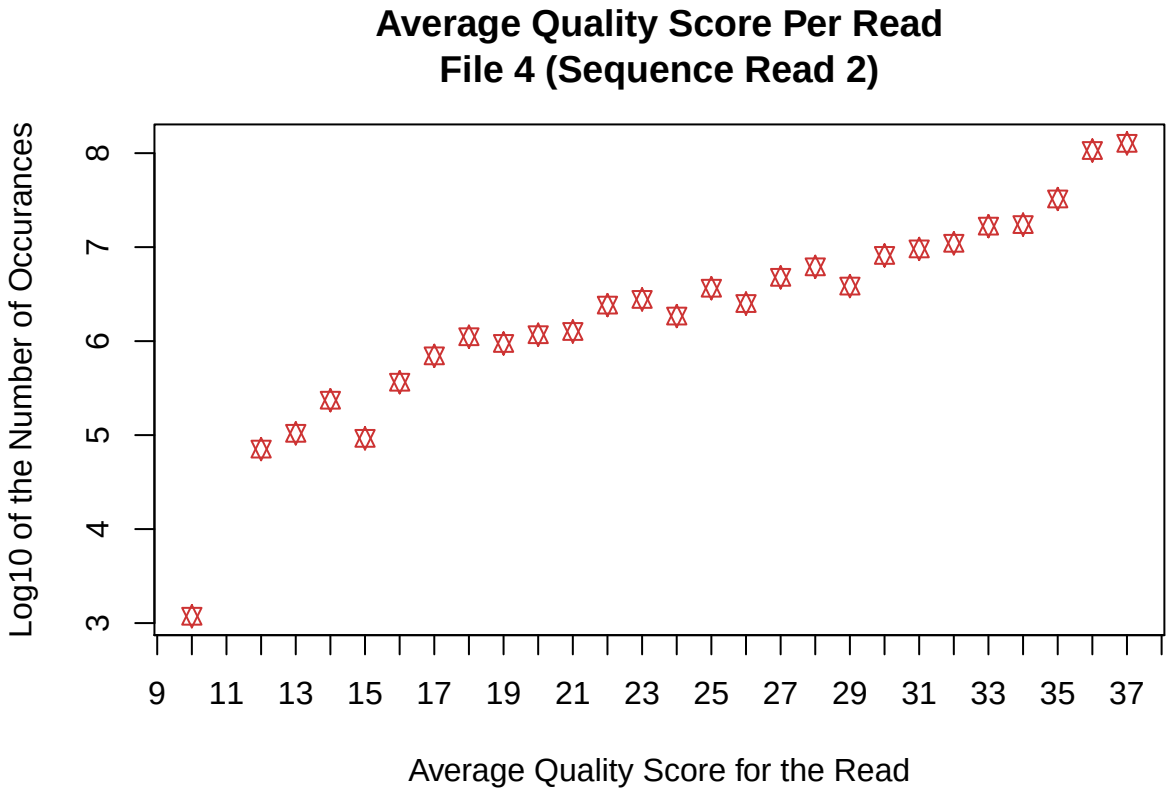



```
# Index 1 Read Means}
plot(r2_readavgs$Quality.Score, log10(r2_readavgs$Frequency.of.Occurance),
     main = "Average Quality Score Per Read \n File 2 (Index 1)",
     xlab = "Average Quality Score for the Read",
     ylab = "Log10 of the Number of Occurances",
     col = "coral",
     cex = 1,
     pch = 15,
     xaxt="n")

axis(1, at=c(0:42), labels = c(0:42))
```



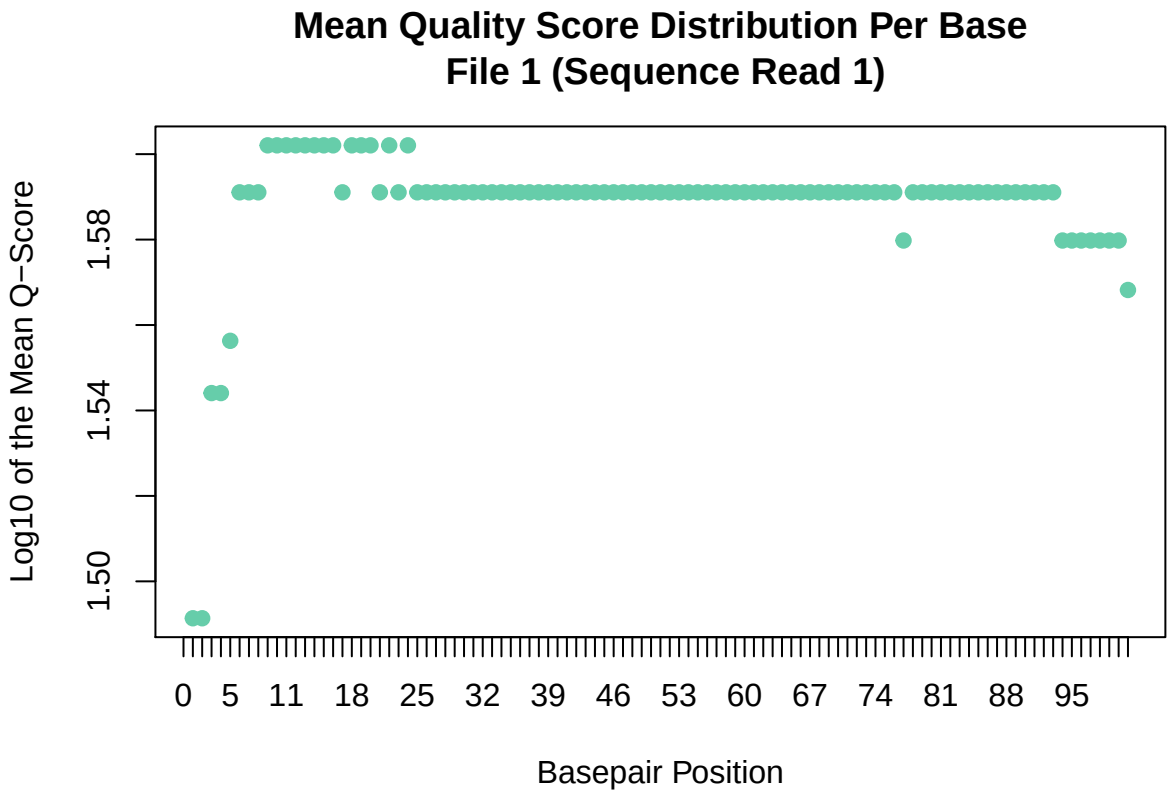

```
# Index 2 Read Means
plot(r3_readavgs$Quality.Score, log10(r3_readavgs$Frequency.of.Occurance),
     main = "Average Quality Score Per Read \n File 3 (Index 2)",
     xlab = "Average Quality Score for the Read",
     ylab = "Log10 of the Number of Occurances",
     col = "cadetblue3",
     cex = 1,
     pch = 17,
     xaxt="n")

axis(1, at=c(0:42), labels = c(0:42))
```


```
# Sequence 2 Read Means}
plot(r3_readavgs$Quality.Score, log10(r3_readavgs$Frequency.of.Occurance),
     main = "Average Quality Score Per Read\nFile 4 (Sequence Read 2)",
     xlab = "Average Quality Score for the Read",
     ylab = "Log10 of the Number of Occurances",
     col = "brown3",
     cex = 1,
     pch = 11,
     xaxt="n")

axis(1, at=c(0:42), labels = c(0:42))
```

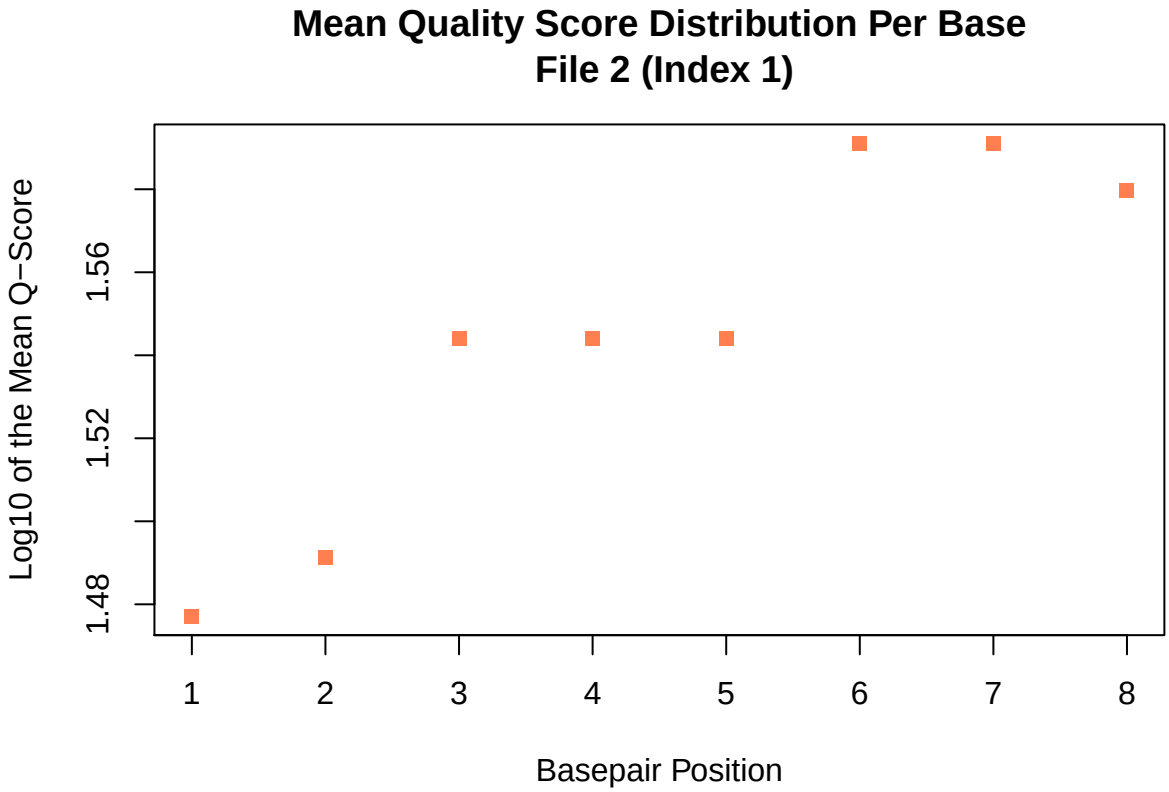


And more plots:

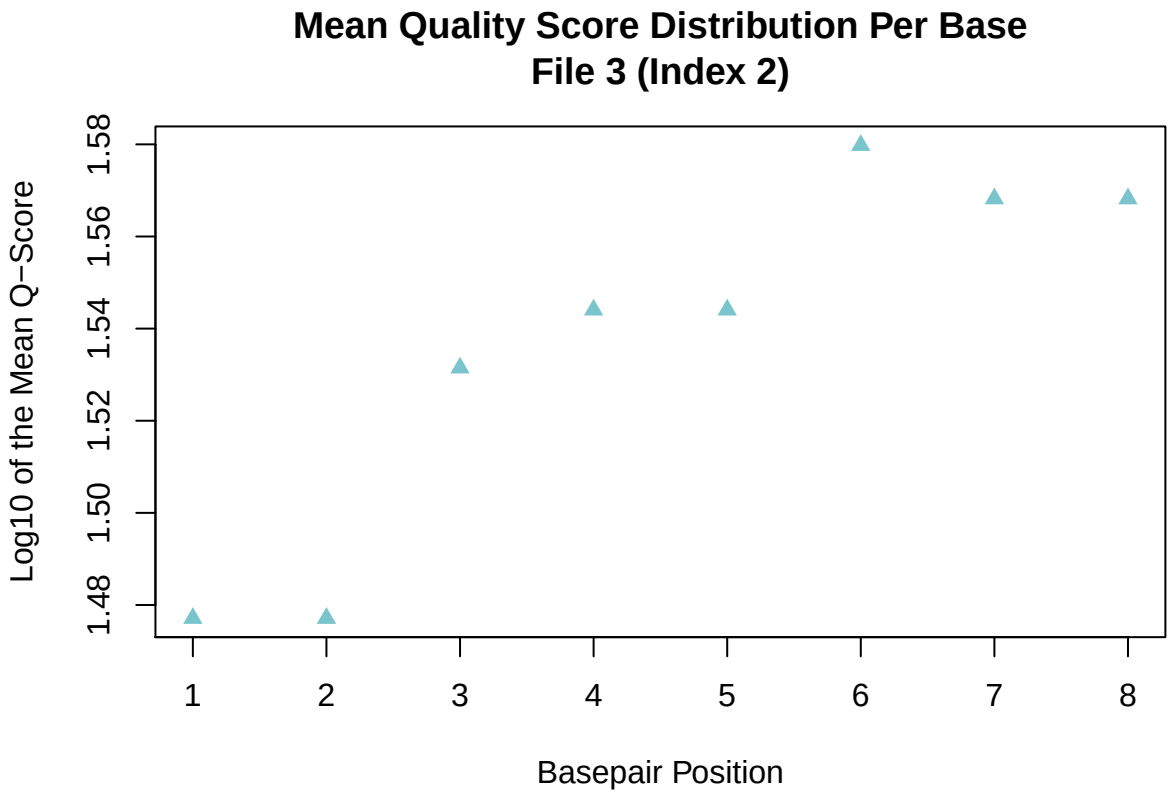
```
# Sequence 1 Q-Score Distribution
#par(mfrow=c(2,2))

plot(r1_qdist$Basepair_Position, log10(r1_qdist$Average_QScore),
     main = "Mean Quality Score Distribution Per Base\n File 1 (Sequence Read 1)",
     xlab = "Basepair Position",
     ylab = "Log10 of the Mean Q-Score",
     col = "aquamarine3",
     cex = 1,
     pch = 19,
     xaxt="n")

axis(1, at=c(0:101), labels = c(0:101))
```



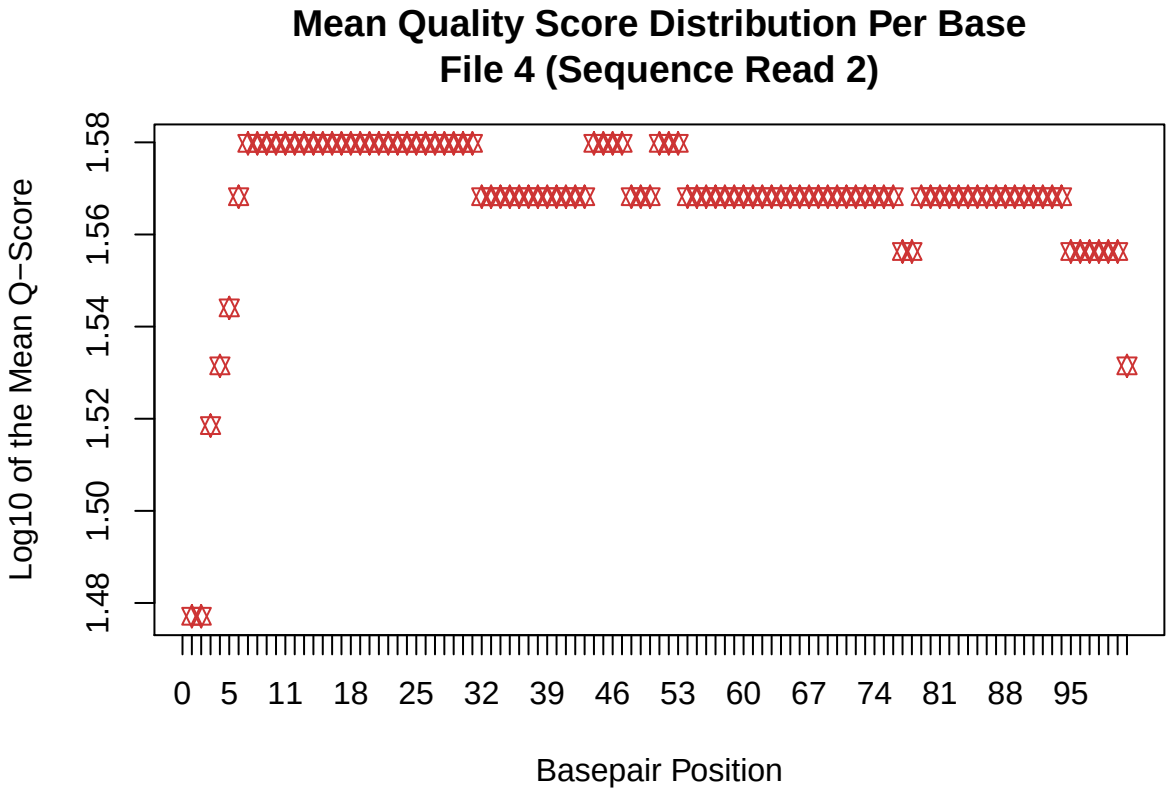

```r
# Index 1 Q-Score Distribution
plot(r2_qdist$Basepair_Position, log10(r2_qdist$Average_QScore),
     main = "Mean Quality Score Distribution Per Base\n File 2 (Index 1)",
     xlab = "Basepair Position",
     ylab = "Log10 of the Mean Q-Score",
     col = "coral",
     cex = 1,
     pch = 15,
     xaxt="n")

axis(1, at=c(0:8), labels = c(0:8))
```




```
# Index 2 Q-Score Distribuion
plot(r3_qdist$Basepair_Position, log10(r3_qdist$Average_QScore),
     main = "Mean Quality Score Distribution Per Base\nFile 3 (Index 2)",
     xlab = "Basepair Position",
     ylab = "Log10 of the Mean Q-Score",
     col = "cadetblue3",
     cex = 1,
     pch = 17,
     xaxt="n")

axis(1, at=c(0:8), labels = c(0:8))
```




```
# Sequence 2 Q-Score Distribution
plot(r4_qdist$Basepair_Position, log10(r4_qdist$Average_QScore),
     main = "Mean Quality Score Distribution Per Base\nFile 4 (Sequence Read 2)",
     xlab = "Basepair Position",
     ylab = "Log10 of the Mean Q-Score",
     col = "brown3",
     cex = 1,
     pch = 11,
     xaxt="n")

axis(1, at=c(0:101), labels = c(0:101))
```



b) Choosing a Quality Score Cutoff

What is a good quality score cutoff for index reads and pairs to utilize for sample identification and downstream analysis, respectively?

The more frequent average quality score per read for all 4 files is around 35 to 40 (that's where there is a very clear increase in the frequencies). To avoid filtering out the reads with this average score, I think a good cutoff value would be 30 - this conclusion is based on my own data. In addition, Illumina actually uses this value and it appears frequently in their user guides and documentation as the "Q30" value. According to Illumina, a Q score of 30 indicates that Phred assigns a Q score of 30 (Q30) to a base, this is equivalent to the probability of an incorrect base call 1 in 1000 times (the probability of the call being correct is 99.9%). It's an industry standard and is used by Illumina themselves, so it makes sense to filter our data with that value in mind.

This is quoted directly from an Illumina document regarding Q-Scores:

"When se- quencing quality reaches Q30, virtually all of the reads will be perfect, having zero errors and ambiguities. This is why Q30 is considered a benchmark for quality in next-generation sequencing."

c) Undetermined Indexes

How many indexes have Undetermined (N) base calls? (Utilize your command line tool knowledge. Summit the command you used. CHALLENGE: use a one line command)

I made a dictionary of indexes containing the letter N (or any unrecognized index that was NOT associated with one of our libraries OR did not match any combination of the indexes we used), and I incremented that dictionary every time such an index was encountered. I then printed the dictionary (as a table of keys and

values) to a .tsv file, and used this the command to see how many *unique* items containing Ns were in that file:

```
cat N_indexes.txt | grep "N" | wc -l
```

There are 2840 *unique* indexes containing undetermined base calls (that passed my minimum quality score of 30 and made it to the dictionary).

But to do the same on the actual, unfiltered data, I used this command:

```
awk '(NR % 4 == 2)' 1294_S1_L008_R2_001.fastq | grep "N" | wc -l
```

There are 3976613 reads containing Ns in index 1 (in the Q30 filtered data).

```
awk '(NR % 4 == 2)' 1294_S1_L008_R3_001.fastq | grep "N" | wc -l
```

There are 3328051 reads containing Ns in index 2 (in the Q30 filtered data).

d) What do the averaged Quality Scores across the reads tell you? Interpret your data specifically.

The averaged scores across each read give you a decent estimate of how accurately each read throughout the sequencing run was called, and allows you to make some basic assumptions about the run itself. They're generally fairly high (which is very apparent from the plots, moreso than the output tables), so it appears that the run was of relatively high quality. I took a look at the lowest average read Q-score for each run below:

```
min(r1_readavgs$Quality.Score)
```

```
## [1] 12
```

```
min(r2_readavgs$Quality.Score)
```

```
## [1] 10
```

```
min(r3_readavgs$Quality.Score)
```

```
## [1] 10
```

```
min(r4_readavgs$Quality.Score)
```

```
## [1] 11
```

It doesn't appear as if any of the files have entire reads with an average score below 10, so every read had at least some high-scoring base calls. However, the lowest scores (somewhere between 10 and 15) are still fairly low, so the run was not as perfect as it could have been. A read average in that range is far below the minimum cutoff I used in my run (Q value of 30).

2. ——————————————————————————————————————

Write a program to de-multiplex the samples and document index swapping and

number of reads retained per sample.

a) How many reads are retained for each expected index pair? What is the percentage?

*Note: I performed all these calculations on my Q30 filtered data, not the raw, unfiltered reads.

```
# read in file containing the content of the dictionary I
# created to store occurances of reads with correct indexes in the file:
correct_indexes <- read.table("correct_indexes.txt", header=F, sep="\t")
colnames(correct_indexes) <- c("Index_Pair", "c_Reads")
```

```
# sum all of the reads associated with correct indexes:
correctsum <- sum(correct_indexes$c_Reads)

# calculate the percentage out of the total reads for the run:
correct_indexes$Percentage <- ((correct_indexes$c_Reads/363246735)*100)

all_indexes <- read.table("all_possindexes.txt", header=F, sep="\t")
colnames(all_indexes) <- c("Index_Pair", "ai_Reads")

# total number of reads for the entire run, including undesired index pairs, is 363246735
all_indexes$Percentage <- (all_indexes$ai_Reads/363246735)*100
```

The total number of reads assigned to correct indexes is `correctsum`.

The percentage of total reads is in the "Percentage" column in the tables below:

```
r  correct_indexes
```

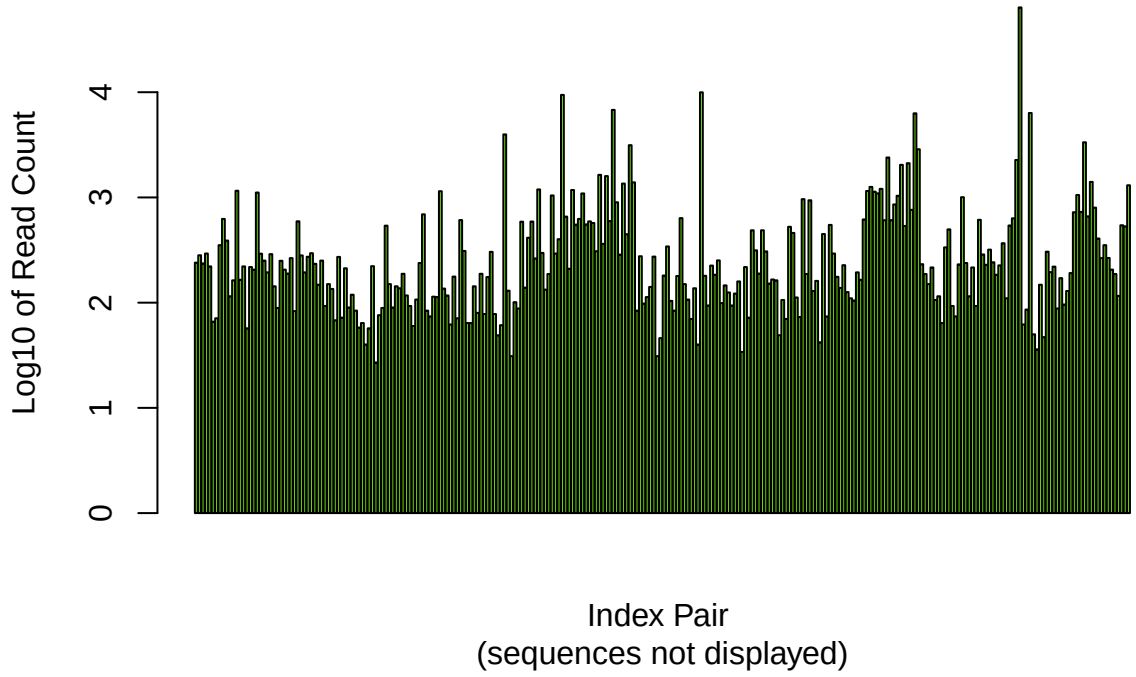
```
##           Index_Pair  c_Reads Percentage  ## 1  GTAGCGTA-GTAGCGTA  6174514  1.6998127
## 2  CGATCGAT-CGATCGAT  4570947  1.2583587  ## 3  GATCAAGG-GATCAAGG  5218247  1.4365572
## 4  AACAGCGA-AACAGCGA  6729364  1.8525601  ## 5  TAGCCATG-TAGCCATG  7606728  2.0940940
## 6  CGGTAATC-CGGTAATC  2601935  0.7162996  ## 7  CTCTGGAT-CTCTGGAT 26021640  7.1636267
## 8  TACCGGAT-TACCGGAT 50777370 13.9787547  ## 9  CTAGCTCA-CTAGCTCA 14094594  3.8801709
## 10 CACTTCAC-CACTTCAC  2848998  0.7843148  ## 11 GCTACTCT-GCTACTCT  4663663  1.2838830
## 12 ACGATCAG-ACGATCAG  6630015  1.8252098  ## 13 TATGGCAC-TATGGCAC  8184795  2.2532329
## 14 TGTTCCGT-TGTTCCGT 12443269  3.4255694  ## 15 GTCCTAAG-GTCCTAAG  6737119  1.8546950
## 16 TCGACAAG-TCGACAAG  2816275  0.7753063  ## 17 TCTTCGAC-TCTTCGAC 31529912  8.6800263
## 18 ATCATGCG-ATCATGCG  7915487  2.1790938  ## 19 ATCGTGGT-ATCGTGGT  5342253  1.4706954
## 20 TCGAGAGT-TCGAGAGT  7912633  2.1783081  ## 21 TCGGATTC-TCGGATTC  3346538  0.9212851
## 22 GATCTTGC-GATCTTGC  2947142  0.8113334  ## 23 AGAGTCCA-AGAGTCCA  8717275  2.3998220
## 24 AGGATAGC-AGGATAGC  6626287  1.8241835
```

b) Index-Swapped Read Counts

Every possible index combination (OTHER than the ones we intentionally assigned to our samples) experiences some degree of index swapping, so 276 indexes were swapped.

c) Create a distribution of swapped indexes. What does this histogram tell you/what is your interpretation of this data?

```
swapped_combos <- read.table("dashed_combos_swapping.txt", sep = ",")
swapped_combos <- as.data.frame(t(swapped_combos))
colnames(swapped_combos) <- c("Index_Pair")

# data frame containing
merged_swaps <- merge(all_indexes, swapped_combos, by = "Index_Pair")

total_swaps <- sum(merged_swaps$ai_Reads)

barplot(height = log10(merged_swaps$ai_Reads),
        main = "Counts of Reads Containing a Swapped Index",
        axes = TRUE, axisnames = TRUE,
        ylab = "Log10 of Read Count",
        xlab = "Index Pair \n(sequences not displayed)",
        col = "chartreuse"
        )
```




```
help(barplot)
```

There were this many reads with swapped indexes:

```
total_swaps
```

```
## [1] 208899
```

My interpretation of this is that some barcodees experienced substantially more swapping than others. Some reads had a relatively small amount on index swapping; however, one particular swapped pair had an extremely large amount of reads (over 50,000 reads!). These indexes didn't have an especially small Levenstein Distance so I don't think that factored in to why this occurred so much more than the others. It's possible this sample just got more reads because of a normalization error, and subsequently also experienced more index swapping just because there was so much more of that sample. Interestingly, every possible index combination had some reads, so quite a large amount of swapping occured. It's still a fairly small percentage of the total number of reads, but it's still a little jarring to see just how much index swapping occurred in our run. This definitely confirms other researcher's concerns about swapping on the HiSeq 4000, and it gives me a much better understanding of the degree to which this happens and its implications for data analysis (this whole process would have been quicker and cleaner if we had less index swapping!).