Computer Exercise 2: Genome Sequencing

Introduction to Bioinformatics (MVE510), Autumn 2024

Group Member: Houshi He

Swedish Social Security Number: 20011114-4838

December 1, 2024

Contents

1 Questions and Answers

1.1 Question 1

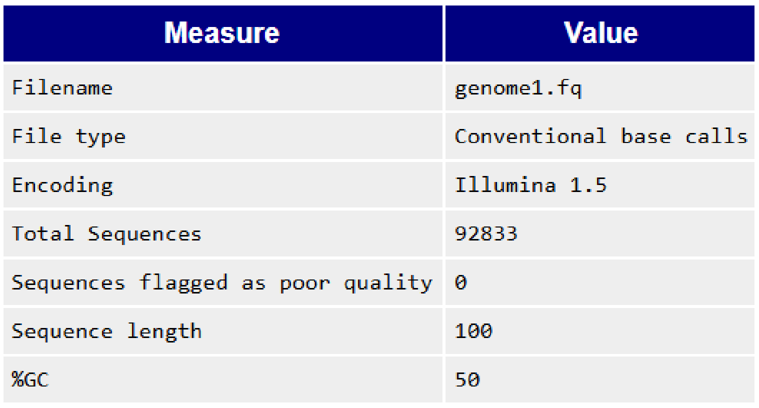
Problem:
1. Is the file a proper FASTQ file? Can you identify the different parts?

Answer:
1. Yes, it is a proper FASTQ file. Each record consists of: the identifier of the read, nucleotide sequence, and the quality of the corresponding base in the sequence.

1.2 Question 2

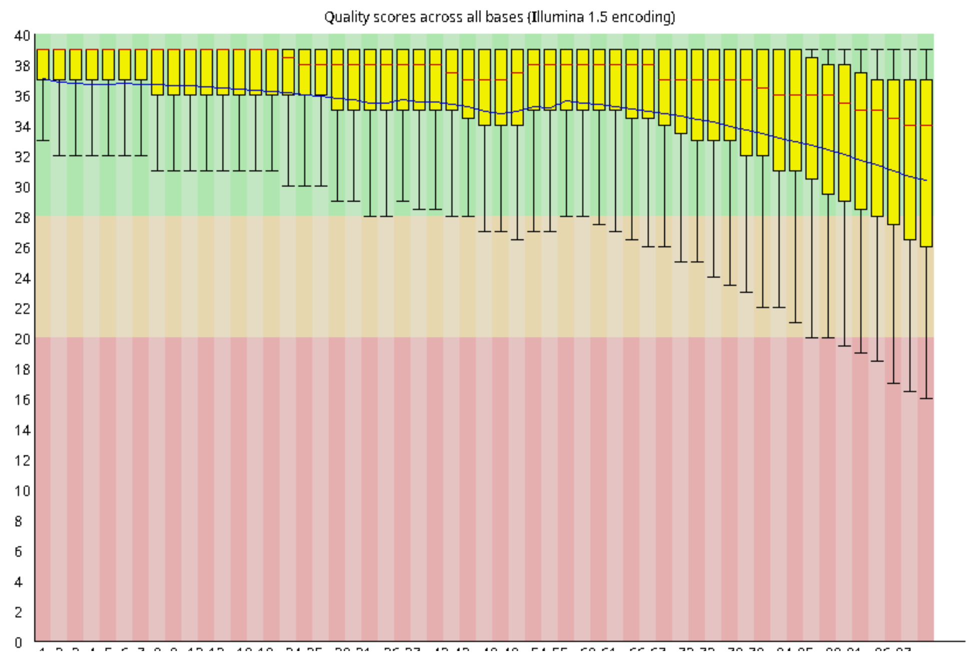
Problem:
1. How many sequences did the file of genome1 contain? How are the quality scores distributed over the reads? What is the G/C content of the reads?
2. How does the quality compare between the samples? Which sample has the worst quality? Are there any other differences?
3. Process all three genomes. Then rerun fastqc on the filtered data files. Do you see any differences? How many sequences were removed from each of the files? Did the read length change?

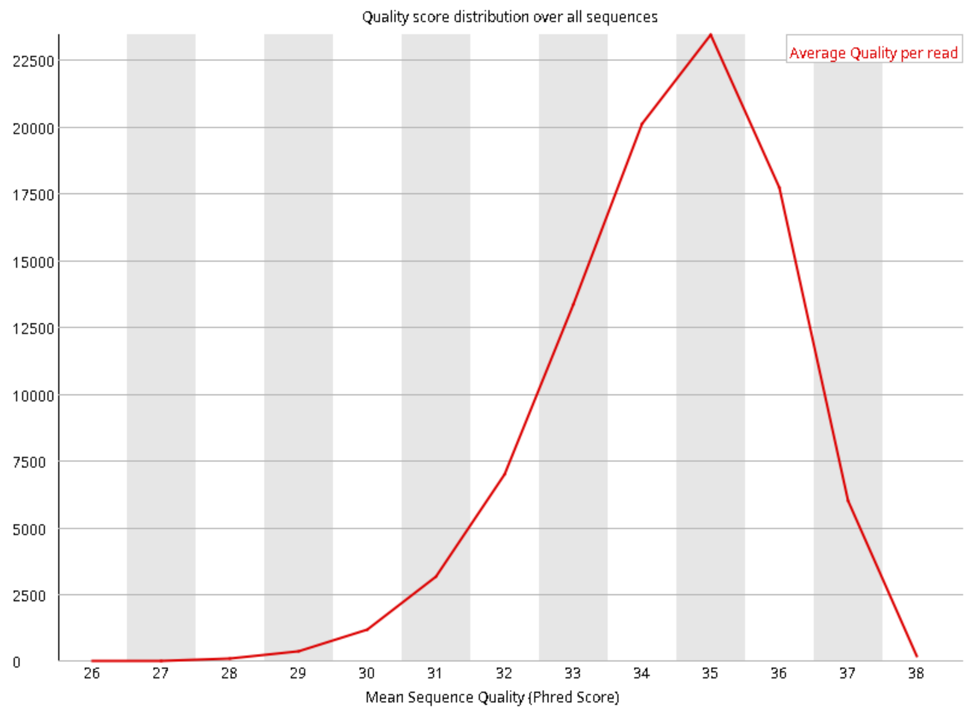
Answer:
1. The file contains 92,833 sequences. The quality scores are primarily distributed between 33 and 37, with a peak around 35. The G/C content of the reads is 50%.



Measure	Value
Filename	genome1.fq
File type	Conventional base calls
Encoding	Illumina 1.5
Total Sequences	92833
Sequences flagged as poor quality	0
Sequence length	100
%GC	50

2. The quality of sample1 is slightly lower compared to the other samples, with mean quality scores concentrated around 35, while the others are around 36. Additionally, sample1 shows a sharp drop in quality towards the end of the reads. Therefore, sample1 has the worst quality.





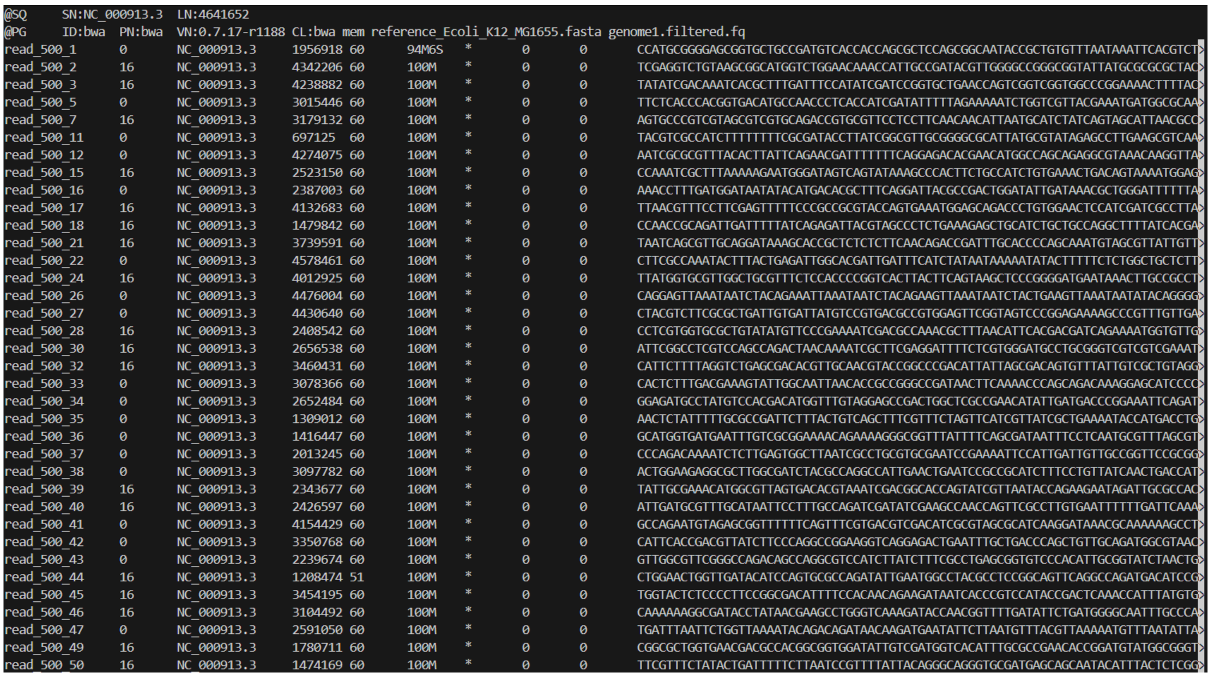
3. Yes, there are differences. For genome1, 19,844 sequences were removed (from 92,833 to 72,989). For genome2, 5,881 sequences were removed (from 116,042 to 110,161). For genome3, 2,852 sequences were removed (from 58,021 to 55,169). The sequence length remains unchanged at 100.

1.3 Question 3

Problem:
Give an example of two different values contained in the columns FLAG, RNAME, POS, and MAPQ in your SAM file and explain their meaning.

Answer:
FLAG: The values 0 and 16 represent different read orientations. 0 means the read is

mapped in a proper pair, while 16 indicates that the read is in reverse orientation.
RNAME: Both values in the RNAME column are NC_000913.3, indicating that all reads are aligned to the same reference genome.
POS: The values 1956918 and 4342206 represent different positions on the reference genome where the reads are aligned.
MAPQ: The values 60 and 51 represent the mapping quality score of the reads. A 60 indicates a high-confidence alignment, meaning the read is mapped with very low probability of error. 51 indicates a slightly lower mapping quality.

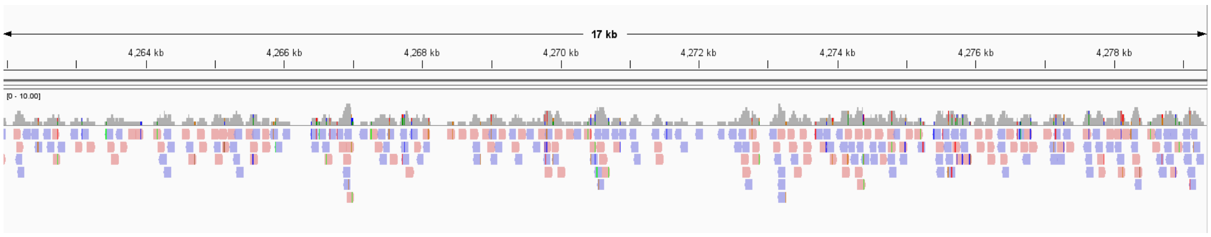


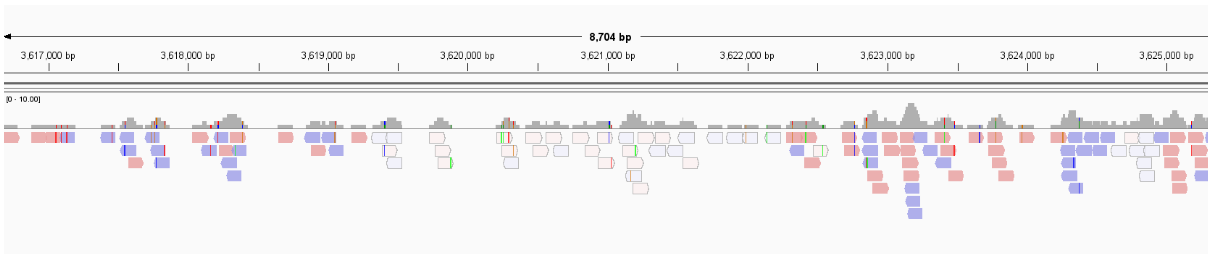
```
@SQ     SN:NC_000913.3  LN:4641652
@PG     ID:bwa  PN:bwa  VN:0.7.17-r1188 CL:bwa mem reference_Ecoli_K12_MG1655.fasta genome1.filtered.fq
read_500_1      0       NC_000913.3     1956918 60      94M6S   *       0       0       CCATGCGGGGAGCGGTGCTGCCGATGTCACCACCAGCGCTCCAGCGGCAATACCGCTGTGTTTAATAAATTCACGTCT>
read_500_2      16      NC_000913.3     4342206 60      100M    *       0       0       TCGAGGTCTGTAAGCGGCATGGTCTGGAACAAACCATTGCCGATACGTTGGGGCCGGGCGGTATTATGCGCGCGCTAC>
read_500_3      16      NC_000913.3     4238882 60      100M    *       0       0       TATATCGACAAATCACGCTTTGATTTCCATATCGATCCGGTGCTGAACCAGTCGGTCGGTGGCCCGGAAAACTTTTAC>
read_500_5      0       NC_000913.3     3015446 60      100M    *       0       0       TTCTCACCCACGGTGACATGCCAACCCTCACCATCGATATTTTTAGAAAAATCTGGTCGTTACGAAATGATGGCGCAA>
read_500_7      16      NC_000913.3     3179132 60      100M    *       0       0       AGTGCCCGTCGTAGCGTCGTGCAGACCGTGCGTTCCTCCTTCAACAACATTAATGCATCTATCAGTAGCATTAACGCC>
read_500_11     0       NC_000913.3     697125  60      100M    *       0       0       TACGTCGCCATCTTTTTTTTCGCGATACCTTATCGGCGTTGCGGGGCGCATTATGCGTATAGAGCCTTGAAGCGTCAA>
read_500_12     0       NC_000913.3     4274075 60      100M    *       0       0       AATCGCGCGTTTACACTTATTCAGAACGATTTTTTTCAGGAGACACGAACATGGCCAGCAGAGGCGTAAACAAGGTTA>
read_500_15     16      NC_000913.3     2523150 60      100M    *       0       0       CCAAATCGCTTTAAAAAGAATGGGATAGTCAGTATAAAGCCCACTTCTGCCATCTGTGAAACTGACAGTAAAATGGAG>
read_500_16     0       NC_000913.3     2387003 60      100M    *       0       0       AAACCTTTGATGGATAATATACATGACACGCTTTCAGGATTACGCCGACTGGATATTGATAAACGCTGGGATTTTTTA>
read_500_17     16      NC_000913.3     4132683 60      100M    *       0       0       TTAACGTTTCCTTCGAGTTTTTCCCGCCGCGTACCAGTGAAATGGAGCAGACCCTGTGGAACTCCATCGATCGCCTTA>
read_500_18     16      NC_000913.3     1479842 60      100M    *       0       0       CCAACCGCAGATTGATTTTTATCAGAGATTACGTAGCCCTCTGAAAGAGCTGCATCTGCTGCCAGGCTTTTATCACGA>
read_500_21     16      NC_000913.3     3739591 60      100M    *       0       0       TAATCAGCGTTGCAGGATAAAGCACCGCTCTCTCTTCAACAGACCGATTTGCACCCCAGCAAATGTAGCGTTATTGTT>
read_500_22     0       NC_000913.3     4578461 60      100M    *       0       0       CTTCGCCAAATACTTTACTGAGATTGGCACGATTGATTTCATCTATAATAAAAATATACTTTTTCTCTGGCTGCTCTT>
read_500_24     16      NC_000913.3     4012925 60      100M    *       0       0       TTATGGTGCGTTGGCTGCGTTTCTCCACCCCGGTCACTTACTTCAGTAAGCTCCCGGGGATGAATAAACTTGCCGCCT>
read_500_26     0       NC_000913.3     4476004 60      100M    *       0       0       CAGGAGTTAAATAATCTACAGAAATTAAATAATCTACAGAAGTTAAATAATCTACTGAAGTTAAATAATATACAGGGG>
read_500_27     0       NC_000913.3     4430640 60      100M    *       0       0       CTACGTCTTCGCGCTGATTGTGATTATGTCCGTGACGCCGTGGAGTTCGGTAGTCCCGGAGAAAAGCCCGTTTGTTGA>
read_500_28     16      NC_000913.3     2408542 60      100M    *       0       0       CCTCGTGGTGCGCTGTATATGTTCCCGAAAATCGACGCCAAACGCTTTAACATTCACGACGATCAGAAAATGGTGTTG>
read_500_30     16      NC_000913.3     2656538 60      100M    *       0       0       ATTCGGCCTCGTCCAGCCAGACTAACAAAATCGCTTCGAGGATTTTCTCGTGGGATGCCTGCGGGTCGTCGTCGAAAT>
read_500_32     16      NC_000913.3     3460431 60      100M    *       0       0       CATTCTTTTAGGTCTGAGCGACACGTTGCAACGTACCGGCCCGACATTATTAGCGACAGTGTTTATTGTCGCTGTAGG>
read_500_33     0       NC_000913.3     3078366 60      100M    *       0       0       CACTCTTTGACGAAAGTATTGGCAATTAACACCGCCGGGCCGATAACTTCAAAACCCAGCAGACAAAGGAGCATCCCC>
read_500_34     0       NC_000913.3     2652484 60      100M    *       0       0       GGAGATGCCTATGTCCACGACATGGTTTGTAGGAGCCGACTGGCTCGCCGAACATATTGATGACCCGGAAATTCAGAT>
read_500_35     0       NC_000913.3     1309012 60      100M    *       0       0       AACTCTATTTTTGCGCCGATTCTTTACTGTCAGCTTTCGTTTCTAGTTCATCGTTATCGCTGAAAATACCATGACCTG>
read_500_36     0       NC_000913.3     1416447 60      100M    *       0       0       GCATGGTGATGAATTTGTCGCGGAAAACAGAAAAGGGCGGTTTATTTTCAGCGATAATTTCCTCAATGCGTTTAGCGT>
read_500_37     0       NC_000913.3     2013245 60      100M    *       0       0       CCCAGACAAAATCTCTTGAGTGGCTTAATCGCCTGCGTGCGAATCCGAAAATTCCATTGATTGTTGCCGGTTCCGCGG>
read_500_38     0       NC_000913.3     3097782 60      100M    *       0       0       ACTGGAAGAGGCGCTTGGCGATCTACGCCAGGCCATTGAACTGAATCCGCCGCATCTTTCCTGTTATCAACTGACCAT>
read_500_39     16      NC_000913.3     2343677 60      100M    *       0       0       TATTGCGAAACATGGCGTTAGTGACACGTAAATCGACGGCACCAGTATCGTTAATACCAGAAGAATAGATTGCGCCAC>
read_500_40     16      NC_000913.3     2426597 60      100M    *       0       0       ATTGATGCGTTTGCATAATTCCTTTGCCAGATCGATATCGAAGCCAACCAGTTCGCCTTGTGAATTTTTTGATTCAAA>
read_500_41     0       NC_000913.3     4154429 60      100M    *       0       0       GCCAGAATGTAGAGCGGTTTTTTCAGTTTCGTGACGTCGACATCGCGTAGCGCATCAAGGATAAACGCAAAAAAGCCT>
read_500_42     0       NC_000913.3     3350768 60      100M    *       0       0       CATTCACCGACGTTATCTTCCCAGGCCGGAAGGTCAGGAGACTGAATTTGCTGACCCAGCTGTTGCAGATGGCGTAAC>
read_500_43     0       NC_000913.3     2239674 60      100M    *       0       0       GTTGGCGTTCGGGCCAGACAGCCAGGCGTCCATCTTATCTTTCGCCTGAGCGGTGTCCCACATTGCGGTATCTAACTG>
read_500_44     16      NC_000913.3     1208474 51      100M    *       0       0       CTGGAACTGGTTGATACATCCAGTGCGCCAGATATTGAATGGCCTACGCCTCCGGCAGTTCAGGCCAGATGACATCCG>
read_500_45     16      NC_000913.3     3454195 60      100M    *       0       0       TGGTACTCTCCCCTTCCGGCGACATTTTCCACAACAGAAGATAATCACCCGTCCATACCGACTCAAACCATTTATGTG>
read_500_46     16      NC_000913.3     3104492 60      100M    *       0       0       CAAAAAAGGCGATACCTATAACGAAGCCTGGGTCAAAGATACCAACGGTTTTGATATTCTGATGGGGCAATTTGCCCA>
read_500_47     0       NC_000913.3     2591050 60      100M    *       0       0       TGATTTAATTCTGGTTAAAATACAGACAGATAACAAGATGAATATTCTTAATGTTTACGTTAAAAATGTTTAATATTA>
read_500_49     16      NC_000913.3     1780711 60      100M    *       0       0       CGGCGCTGGTGAACGACGCCACGGCGGTGGATATTGTCGATGGTCACATTTGCGCCGAACACCGGATGTATGGCGGGT>
read_500_50     16      NC_000913.3     1474169 60      100M    *       0       0       TTCGTTTCTATACTGATTTTTCTTAATCCGTTTTATTACAGGGCAGGGTGCGATGAGCAGCAATACATTTACTCTCGG>
```

1.4 Question 4

Problem:
Are the reads organized in any particular way? Can you say anything about the coverage? Remember that we only work with 10% of the total data. Do you see any sequencing errors?

Answer:
The reads are spread out evenly, with a similar number on the forward and reverse strands. The coverage is also even, with reads covering gene areas, coding regions, and non-coding regions. There are some sequencing errors, shown by mismatches and MAPQ of 0, which means the reads may not be accurate in some places.



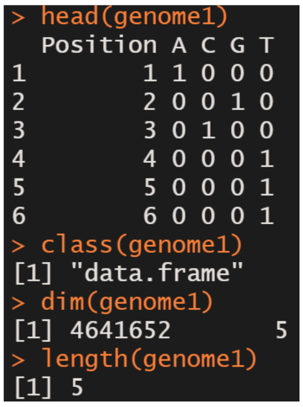


1.5 Question 5

Problem:
Familiarize yourself with these objects. How are they organized?

Answer:
We can familiarize with the data structure using the following code. Similarly, we can apply the same process to genome2.rdata, genome3.rdata, and reference.

```r
# Code for Question 5
load("genome1.rdata")
load("genome2.rdata")
load("genome3.rdata")

ls()

head(genome1)
class(genome1)
dim(genome1)
length(genome1)

genome1.subset=genome1[1:1000,]
ref.subset=reference[1:1000]
```



```
> head(genome1)
  Position A C G T
1        1 1 0 0 0
2        2 0 0 1 0
3        3 0 1 0 0
4        4 0 0 0 1
5        5 0 0 0 1
6        6 0 0 0 1
> class(genome1)
[1] "data.frame"
> dim(genome1)
[1] 4641652       5
> length(genome1)
[1] 5
```

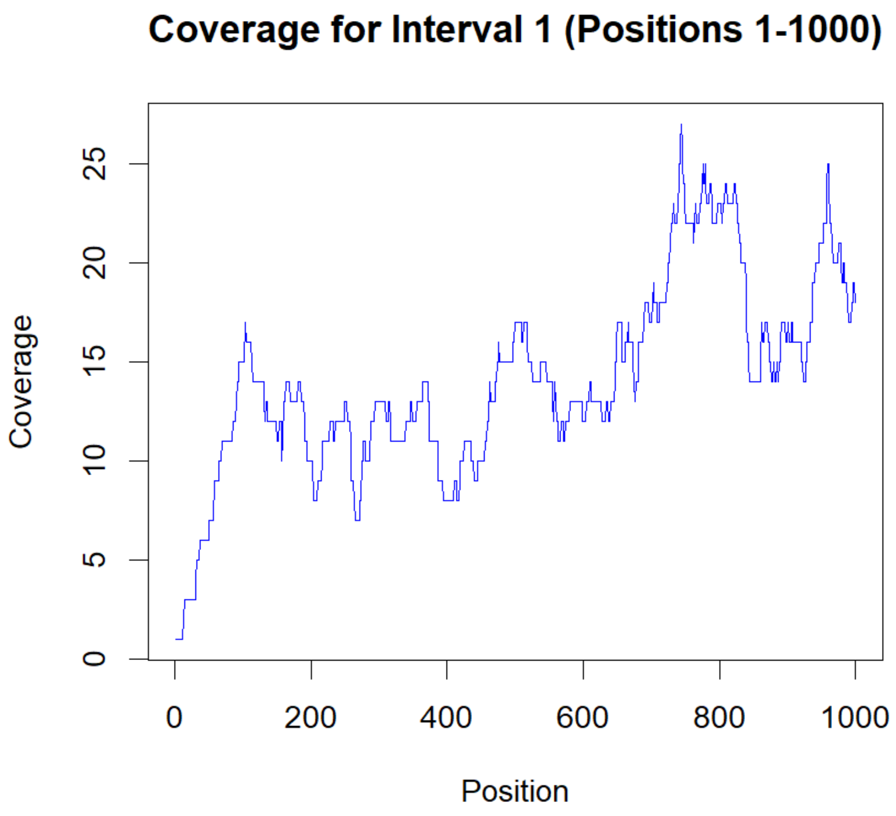
1.6 Question 6

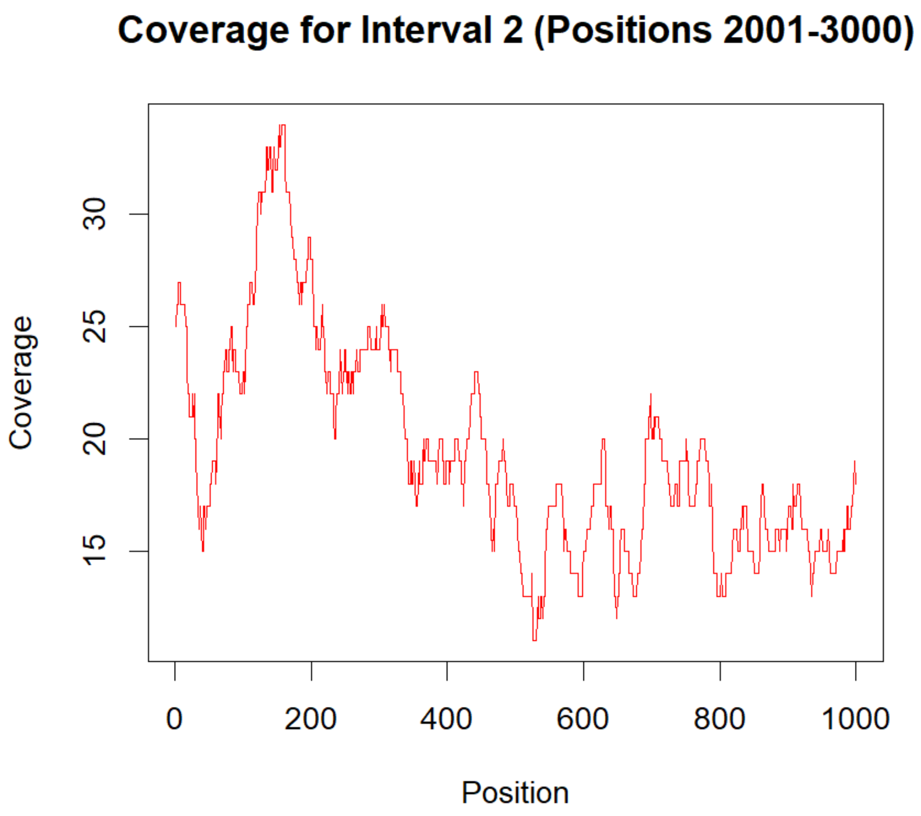
Problem: Calculate also the mean coverage over the entire genome. Is the coverage varying? Why? What is the maximum coverage? Choose two different intervals of 1,000 positions in length and plot the coverage for those regions. Why is it good to have a high coverage?

Answer:
The mean coverage over the entire genome is 19.99. This means, on average, each position in the genome is covered by about 20 sequencing reads.
Yes, the coverage is varying. The variance in coverage is 25.35, which indicates there is a fluctuation in the number of reads that cover different positions. This could be due to uneven sequencing depth across the genome, or the presence of certain genomic regions more likely to be captured by reads.
The plots are below the text.
High coverage is good because it improves the accuracy of the sequencing results. More reads covering the same position reduce the chance of errors.

```r
# Code for Question 6
# Calculate the coverage for each position by summing A, C, G, T
   for each row
coverage <- apply(genome1[, -1], 1, sum)

# Calculate the mean coverage over the entire genome
mean_coverage <- mean(coverage)

# Print the mean coverage
cat("Mean coverage over the entire genome:", mean_coverage, "\n")

# Check if the coverage is varying by looking at the variance
coverage_variance <- var(coverage)
cat("Variance in coverage:", coverage_variance, "\n")

# Find the maximum coverage
max_coverage <- max(coverage)
cat("Maximum coverage:", max_coverage, "\n")

# Select two different intervals of coverage (positions 1-1000
   and 2001-3000)
interval1_coverage <- coverage[1:1000]
interval2_coverage <- coverage[2001:3000]

# Plot the coverage for the first interval
plot(interval1_coverage, type = "l", col = "blue",
     main = "Coverage for Interval 1 (Positions 1-1000)", xlab =
        "Position", ylab = "Coverage")

# Plot the coverage for the second interval
plot(interval2_coverage, type = "l", col = "red",
     main = "Coverage for Interval 2 (Positions 2001-3000)", xlab
        = "Position", ylab = "Coverage")
```

Listing 1: Code for Question 6

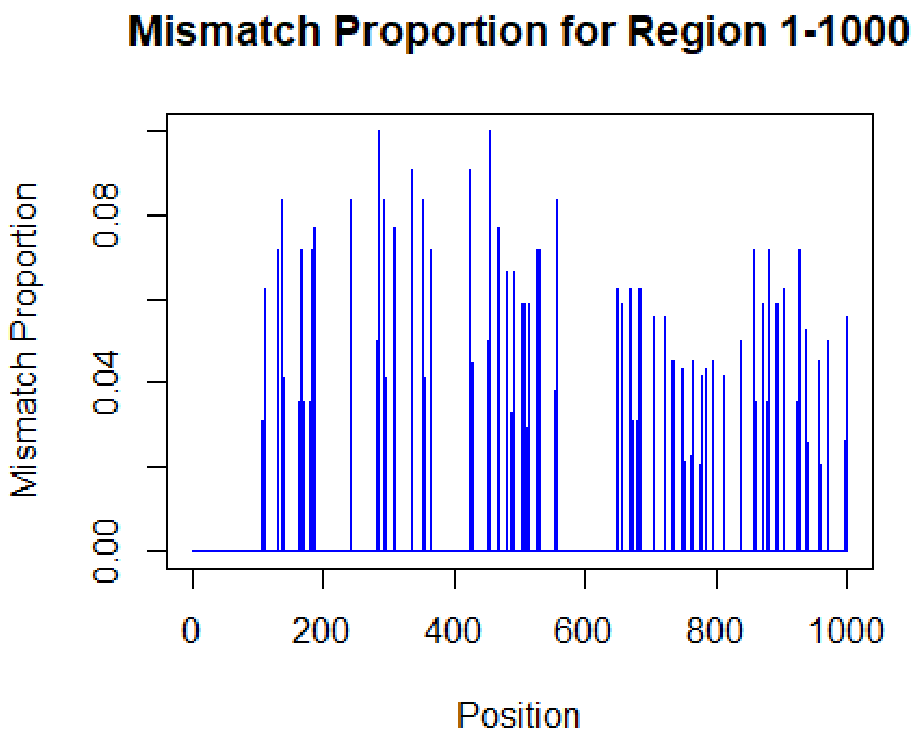

Coverage for Interval 1 (Positions 1-1000)
Coverage
0
5
10
15
20
25
0
200
400
600
800
1000
Position


Coverage for Interval 2 (Positions 2001-3000)
Coverage
15
20
25
30
0
200
400
600
800
1000
Position

1.7 Question 7

Problem:
How many positions have at least one read with a mismatch? Visualize the proportion of mismatching reads over a region covering 1,000 positions.

Answer:
Number of positions with at least one mismatch: 427643
The plots are below the text.



```r
# Code for Question 7
load("genome1.rdata")
load("genome2.rdata")
load("genome3.rdata")

genome1.subset=genome1[1:1000,]
ref.subset=reference[1:1000]

genome1.length <- nrow(genome1)
mismatches <- vector(length = genome1.length)
coverage <- vector(length = genome1.length)

for (pos in 1:genome1.length) {
  ref_base <- reference[pos]

  genome_bases <- names(genome1[pos, -1])[genome1[pos, -1] > 0]
  genome_counts <- genome1[pos, -1][genome1[pos, -1] > 0]

  coverage[pos] <- sum(genome_counts)

  mismatches[pos] <- sum(genome_counts[genome_bases != ref_base])
}

proportion_mismatches <- mismatches / coverage

region_proportion <- proportion_mismatches[1:1000]

plot(region_proportion, type = "l", col = "blue",
     main = "Mismatch Proportion for Region 1-1000",
     xlab = "Position", ylab = "Mismatch Proportion")

positions_with_mismatches <- sum(mismatches > 0)
print(paste("Number of positions with at least one mismatch:",
   positions_with_mismatches))
```

Listing 2: Code for Question 7

1.8 Question 8

Problem:
1. Under the assumption that H_0 is true, Y_i can be shown to follow a binomial distribution with parameters p_error and N_i. Why? What assumptions are necessary for this to be true?
2. What would you say is a suitable value for p_error considering that we are working with Illumina data? Can you see any reason for the setting p_error to be larger than the average error rate?

Answer:
1. Each nucleotide read at position is an independent event. The probability of a sequencing error is constant. The total number of reads at position i is N_i.
Errors occur randomly and independently across reads. The probability of a sequencing error is consistent. The outcome of one read does not influence the outcome of another read at the same position.
2. p_error could be chosen as 0.01 (1%), which represents a conservative upper limit of the average error rate for Illumina sequencing.
Because low-quality regions or reads might have a higher error rate, and setting p_error slightly higher creates a more conservative null hypothesis. Also, if the observed mismatch is still significantly higher than p_error, then it provides stronger evidence of a true mutation rather than random sequencing errors.

```r
# Code for Question 8
calculate_p_value <- function(y_i, N_i, p_error) {

  if (N_i == 0) {
    return(NA)
  }

  result <- binom.test(y_i, N_i, p_error, alternative = "greater"
    )
  return(result$p.value)
}
```

Listing 3: Code for Question 8

1.9 Question 9

Problem:
Are there any positions that show evidence of mutation? What is a good p-value cut-off for selecting significant positions? Is there any risk of setting the p-value cut-off to high? Repeat the analysis for all three genomes. Which of the genomes has the highest number of significant SNPs?

Answer:
Yes, there are positions in all three genomes where the p-values are below the threshold of 0.05, which means significant mismatches compared to the reference genome. I choose 0.05, which means there's less than a 5% chance the observed mismatch is due to random error. Yes, this will increase the likelihood of false positives. Genome1 has the highest number of significant SNPs. If we choose 0.05 as the p-value cut-off, genome1 has 20785 significant SNPs, genome2 has 6290, and genome3 has 2710.

```r
# Code for Question 9
load("genome1.rdata")
load("genome2.rdata")
load("genome3.rdata")

genome1.subset=genome1[1:5000,]
ref.subset=reference[1:5000]

calculate_p_value <- function(y_i, N_i, p_error) {

  if (N_i == 0) {
    return(NA)
  }

  result <- binom.test(y_i, N_i, p_error, alternative = "greater"
    )
  return(result$p.value)
}

#For the code of genome2 and genome3, need to change the name of
   the variable.
genome1.length <- nrow(genome1)
mismatches <- vector(length = genome1.length)
```

```r
coverage <- vector(length = genome1.length)
p_values <- vector(length = genome1.length)

p_error <- 0.01

for (pos in 1:genome1.length) {
  ref_base <- reference[pos]

  genome_bases <- names(genome1[pos, -1])[genome1[pos, -1] > 0]
  genome_counts <- genome1[pos, -1][genome1[pos, -1] > 0]

  coverage[pos] <- sum(genome_counts)

  mismatches[pos] <- sum(genome_counts[genome_bases != ref_base])

  p_values[pos] <- calculate_p_value(mismatches[pos], coverage[
     pos], p_error)
}

significant_positions <- which(p_values < 0.05)

significant_data <- data.frame(
  Position = significant_positions,
  P_value = p_values[significant_positions]
)

write.csv(significant_data, file = "genome1.csv", row.names =
   FALSE)
```

Listing 4: Code for Question 9

1.10 Question 10

Problem:
Do any of the isolates carry mutations that make them resistant to an antibiotic? Answer this question by examining where the three most significant SNPs in each isolate are located.

Answer:
For genome1, the three positions with the smallest p-values are 1239179, 1037292, and 797291. For genome2, the three positions with the smallest p-values are 2339173, 2220911, and 3521659. For genome3, the three positions with the smallest p-values are 3279936, 3385977, and 3548496. Through the NCBI GenBank database, it was found that the mutation at position 1239179 (genome1) affects alanine racemase 2, which may lead to antibiotic resistance, and the mutation at position 2339173 (genome2) may cause a structural change in the gyrA protein, leading to antibiotic resistance to quinolone antibiotics in the bacteria.