**MVE510 - Introduction to Bioinformatics**
**Computer Exercise 2**

Yankun Xu (940630-0237)
yankun@student.chalmers.se

Duygu Ayalp (931006-C701)
ayalp@student.chalmers.se

---

# Step 1

## Exercise 1

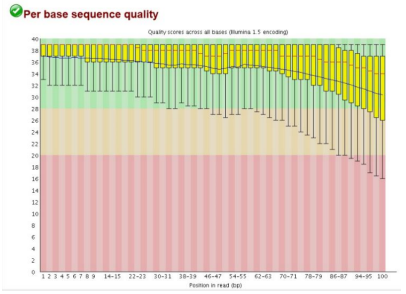
The generated file shown in figure 1 represents a proper FASTQ-file since it contains the read name in the first line (red arrow), the read sequence (blue arrow) followed by a “+” and the sequence quality score (grey arrow). The quality score per base of the sequence is encoded with distinct characters. [1]


Figure 1. FASTQ-file of genome sequence of sample 1

## Exercise 2

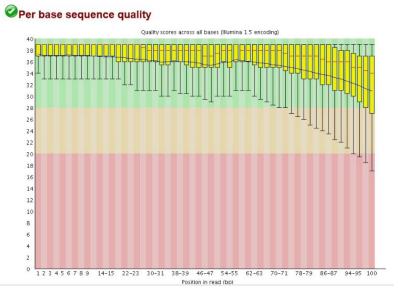
The sequence quality of the data is represented in the FastQC Report (figure 2). For the first sample, 92 833 sequences in total were processed. The quality score per base is uneven distributed over the reads. Initially, the quality score is very high but after 70 to 71 bp the quality of the read becomes lower. The overall G/C-content of the reads is 50 % which is relatively medium. However, a warning for the per sequence G/C-content is shown in the summary, which is an indicator for a (slightly) unusual data quality. As shown in the second plot, the peak for the GC count per read is higher than the theoretical distribution.

| 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 |



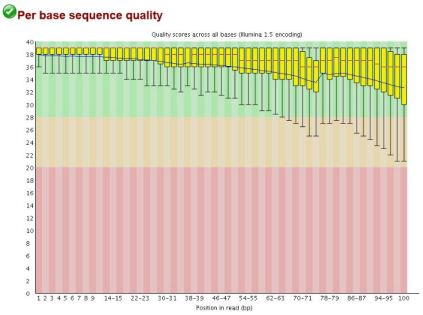

2(a). Original data of genome1

| Measure | Value |
|---|---|
| Filename | genome1.filtered.fq |
| File type | Conventional base calls |
| Encoding | Illumina 1.5 |
| Total Sequences | 72989 |
| Sequences flagged as poor quality | 0 |
| Sequence length | 100 |
| %GC | 50 |



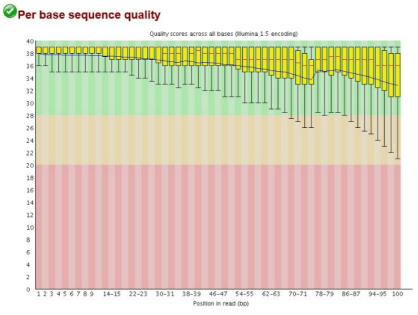

2(b). Filtered data of genome1

| Measure | Value |
|---|---|
| Filename | genome2.fq |
| File type | Conventional base calls |
| Encoding | Illumina 1.5 |
| Total Sequences | 116042 |
| Sequences flagged as poor quality | 0 |
| Sequence length | 100 |
| %GC | 50 |



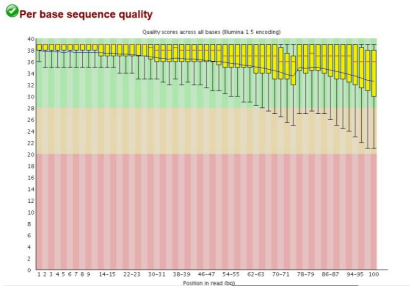

2(c). Original data of genome2

| Measure | Value |
|---|---|
| Filename | genome2.filtered.fq |
| File type | Conventional base calls |
| Encoding | Illumina 1.5 |
| Total Sequences | 110161 |
| Sequences flagged as poor quality | 0 |
| Sequence length | 100 |
| %GC | 50 |



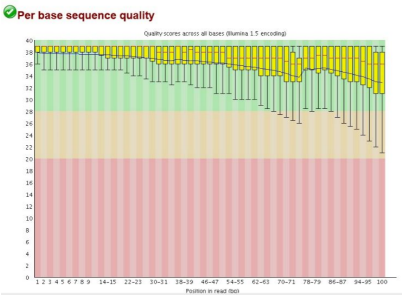

2(d). Filtered data of genome2

| Measure | Value |
|---|---|
| Filename | genome3.fq |
| File type | Conventional base calls |
| Encoding | Illumina 1.5 |
| Total Sequences | 58021 |
| Sequences flagged as poor quality | 0 |
| Sequence length | 100 |
| %GC | 50 |




2(e). Original data of genome3

| Measure | Value |
|---|---|
| Filename | genome3.filtered.fq |
| File type | Conventional base calls |
| Encoding | Illumina 1.5 |
| Total Sequences | 55169 |
| Sequences flagged as poor quality | 0 |
| Sequence length | 100 |
| %GC | 50 |




2(f). Filtered data of genome3

Figure 2. FastQC-report for three genomes

The distribution of the quality scores between the samples is relatively comparable. However, for sample 2 and 3 the interquartile range of the quality scores are higher than 28 whereas the interquartile range in the last positions of the read of sample 1 are below 28. Considering this fact, sample 1 has the worst quality. Sample 2 contains 116 042 sequences while sample 3 contains only 58 021 sequences in total. For all samples, the overall G/C-content is 50 %.

Using the “fastq quality filter” tool reads with low quality were filtered. This led to higher quality scores in the latter positions in the read for each sample. For instance, in sample 2 and 3 the interquartile range of the quality score for the last position (100 bp) is 32 after filtering  instead of 30. Furthermore, the total of sequences for each sample is reduced by filtering. 19844 sequences were removed from sample 1, 5881 sequences were removed from sample 2 and 2852 sequences were removed from sample 3. The read length remained the same for each sample.

## Exercise 3

With the help of the BWA software the reads were aligned to a reference genome (*E.Coli* strain: K12 MG1655). The resulting SAM-file for sample 1 consisting of the header section (beginning with “@” and several columns for the alignment information. The columns contain information about the sequence of the reads, the quality score and alignment coordinates.

In general, it is difficult to interpret the given information due to the amount of the information and the complexity which is partly caused by encoding information.

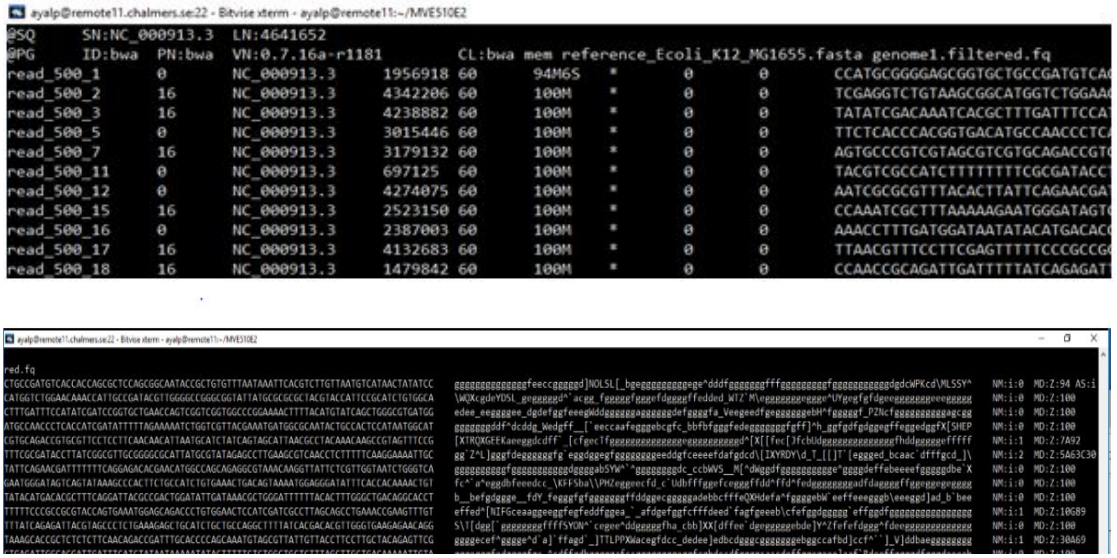



Figure 3. Part of the SAM-file for sample 1

## Exercise 4

After converting the SAM-file into a BAM-file the results of the alignment were visualized with the help of the IGV software. Figure 4 shows the results as coverage plots for the distinct

samples. For each sample, an accumulation of aligned reads at particular positions of the reference genome can be observed, which is reflected by the coverage track. Comparing the coverage bar plots of the three samples along the reference genome, sample 2 seems to have the highest overall coverage, followed by sample 3. By zooming individual base mismatches indicated by different colours according to the base become visible. An example of sequence errors in sample 2 is shown in figure 5. The reference sequence is shown in the lower panel. [2]

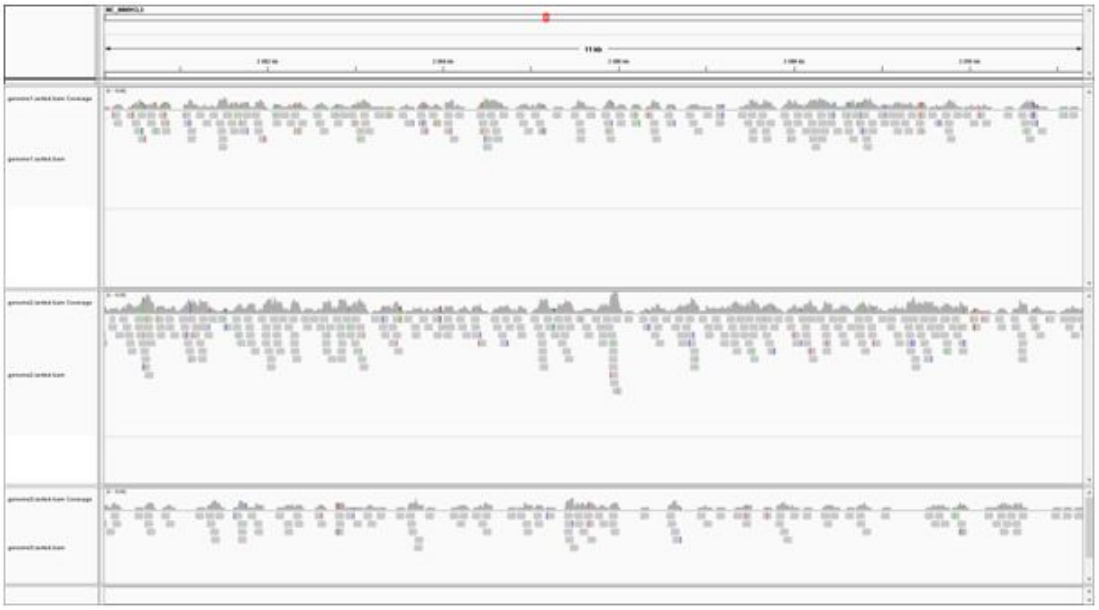


Figure 4. IGV results for sample 1, 2 and 3


Figure 5. Base mismatches in sample 2

# Step 2

## Exercise 6

The coverage varies so much. The coverage is the average number of times the sequenced genome is covered by reads, it depends on the experimental design, amount of sequencing data generated, quality of the sequencing data etc. And properties of reads, such as length, primer also determine coverage. There are less coverage in the beginning or ending part of genome, and the middle part of genome would have more coverages.

The mean coverage over the entire genome 1 is 20, and the maximum coverage is 51. The figure of region for first 1000 position is plotted as figure X. Higher coverage can lead to higher accuracy in identifying mutations.

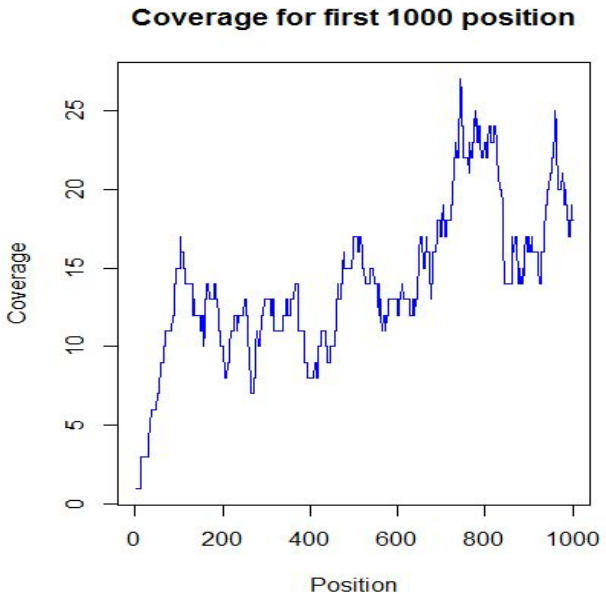



Figure X. The coverage plot for first 1000 position

## Exercise 8

We use hypothesis testing to test single nucleotide polymorphisms (SNPs). It is necessary for us to make assumption that each position follows the binomial distribution with same error rate even if it cannot exactly happen in nature. Our null hypothesis is $H_0 : p_i = p_{error}$ , if our observation make p-value too small, we will reject the null hypothesis and then accept the alternative hypothesis $H_1 : p_i > p_{error}$ . In this case, we use one-side testing which means we need set the parameter as “greater” in the function binom.test.

By the way, there are some positions with zero coverage, for these positions we set p-value as 1, and the code is shown in the appendix.

## Exercise 9

We build for-loop function to test three genome dataset, the code is shown in the appendix.

The position 1239179 and 1037292 in genome1, 2339173 in genome2, 3279936, 3385977, 3548496 in genome3 has too small p-values, we believe that mutations happen in these position. Normally 5% significant level is set to test, but in this case, we set 1e-04 or less as cut-off p-value, because we only pick up all mismatch position as mutation, if we set higher p-value, we will consider mutations happening in some positions without mutations, or with different genotypes in allele. In addition, the highest number of significant SNPs happens in genome3 which has three mutation position.

## Exercise 10

In genome1, we found nucleotides on position 1239179 and 1037292, then we go to the website NCBI GenBank database for Escherichia coli to investigate more. For position 1239179, we look into geneID 945754. The original codon on position 1239179 is CAC, but mutation make this codon become TAC, and this mutation happens on 101st (floor[(1239179 - 1238879)/3] + 1) codon of this gene. For position 1037292, we look into geneID 945572 , the mutation happening on 544th(floor[(1037292-1036749)/3]+1) codon transfer GAC to AAC.

Repeating the operations for genome2 and genome3, we will get following table.

| | Genome1 | | Genome2 | Genome3 | | |
|---|---|---|---|---|---|---|
| Position | 1037292 | 1239179 | 2339179 | 3279936 | 3385977 | 3548496 |
| GeneID | 945572 | 945754 | 946614 | 947637 | 947830 | 947923 |
| Gene symbol | hyaF | dadX | gyrA | kbaZ | yhcO | malQ |
| Position | 544 | 301 | 2387 | 1023 | 121 | 511 |
| No. Codon | 182nd | 101st | 796th | 341st | 41st | 171st |
| Strand | 5’ → 3’ | 5’ → 3’ | 5’ → 3’ | 5’ → 3’ | 3’ →5’ | 3’ → 5’ |
| Direction | Forward | Forward | Reverse | Forward | Reverse | Reverse |
| Origin Codon | GAC | CAC | GGT | GCG | GAA | CTG |
| Alternative | AAC | TAC | GCT | GCA | GAT | CTC |

To be honest, it’s not easy to search relevant information on a very opening source platform, fortunately we got something. The mutation on gyrA gene would confer resistance to

nalidixic acid, and point mutations in gyrA would lead to quinolone resistance acquisition. So we believe that the isolate 2 is supposed to be resistant to antibiotics. [3][4]

# Reference


1. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3346182/
3. https://www.ncbi.nlm.nih.gov/pubmed/23089747
4. https://www.ncbi.nlm.nih.gov/pubmed/24306130

# Appendix

### Code in R language

```
load("genome1.rdata")
load("genome2.rdata")
load("genome3.rdata")

# Ex5 Ex6
genome1.subset = genome1[1:1000,]
cov.set = apply(genome1.subset[,2:5], 1, sum)
ref.subset = reference[1:1000]
genome1.subset = cbind(genome1.subset,cov.set)

cov = apply(genome1[,2:5], 1, sum)
genome1 = cbind(genome1,cov)
mean.coverage = mean(genome1[,6],2) # 20
max.coverage = max(genome1[,6],2)

plot(genome1.subset[1:1000,1],genome1.subset[1:1000,6],"l",col="blue",
    main="Coverage for first 1000 position",
    xlab = "Position", ylab = "Coverage")

# Ex7

genome1.subset$matches <- apply(genome1.subset, 1, function(row)
row[ref.subset[row[1]]])

# Ex8

calculate.pvalue = function(genome,perror){
  genome.length = nrow(genome)
  pvalue = vector(length=genome.length, mode = "double")
  for (i in 1:genome.length) {
    if (genome[i,6] == 0) {
      pvalue[i] = 1
    } else {
      pvalue[i] =
        binom.test(genome[i,6]-genome[i,7], genome[i,6], p = perror,
                   alternative = "greater")$p.value
    }
  }
  return(pvalue)

}
```

```
pval = calculate.pvalue(genome1.subset,0.05)


# Ex9
genome1.subset = cbind(genome1.subset,pval)
genome1.subset.order=order(genome1.subset[,8], decreasing=FALSE)
genome1.subset.sorted = genome1.subset[genome1.subset.order,]

# combine into one function
sort.pval = function(genome){
  coverage = apply(genome[,2:5], 1, sum)
  genome = cbind(genome,coverage)
  genome$matches <- apply(genome, 1, function(row) row[reference[row[1]]])
  pval = calculate.pvalue(genome,0.05)
  genome = cbind(genome,pval)
  genome.order=order(genome[,8], decreasing=FALSE)
  genome.pval.sorted = genome[genome.order,]
  return(genome.pval.sorted)
}
```