Computer Exercise 2:
Genome sequencing

Introduction to Bioinformatics (MVE510)
Autumn, 2023

Introduction

This computer exercise focuses on genome sequencing. We will analyze three datasets, each representing a sequencing experiment of a bacterial genome. All the major steps in the analysis will be included, starting with quality assessment and preprocessing of the reads. After that, the reads will be mapped to a reference gene and visualized in a software called Integrative Genome Viewer (IGV). Then, the data will be loaded into R, where it will be analyzed and single nucleotide polymorphisms (SNPs) be identified.

An important aim of this exercise is to introduce you to Linux, which is the computer environment used to perform bioinformatics analysis. Since you may not have previously worked extensively in the Linux environment, the first part of the computer exercise comes with detailed directions on how to run commands in Linux. If you have issues or questions regarding Linux, don't hesitate to ask the assistants!

The computer exercise should be performed alone or in groups of a maximum of two students. The examination will be based on a written report describing the different steps you took, the generated figures, and your answers to the exercise questions. Please refer to the guidelines on the course homepage regarding how the report should be structured and submitted. The reports should be handed in through the course homepage in Canvas under Assignment latest December 3rd.

Background of the data

In this computer exercise, you play the role of an employee at the Swedish Center for Disease Control (Folkhälsomyndigheten). In a recent investigation of disease-causing bacteria, you encountered three suspicious bacterial isolates sampled from vegetables in a supermarket. The isolates were identified as *Escherichia coli* and you fear this may be the start of an outbreak. You therefore want to make sure that this form of *E. coli* is treatable with antibiotics. Resistance to antibiotics is often caused by mutations in the bacterial genome, so you order whole-genome sequencing of all three isolates. The sequencing was done using the Illumina sequencing platform at a read length of 100 nucleotides and resulted in approximately 1 million reads for each isolate.

Your ultimate aim is to decide whether any of the three isolates are resistant to antibiotics. This will be done by processing and analyzing the whole-genome sequencing data and interpreting the result. We will start from the very first step, with pre-processing the raw data, and end with identifying potential mutations and examining their biological impact.

Step 1 – Preprocessing and sequence alignment

The first part of the computer exercise will be done in Linux. This is the computer environment for which the vast majority of all methods and software used to handle and process sequencing data are developed. We will run these methods through the Linux shell, which is similar to the command line that is available in Windows.

You can log in to your account on the Linux system using the Windows software Bitwise. To log in, write **remote11.chalmers.se** in 'Host' in the 'Server' field and your CID under 'Username' in the 'Authentication' field. Once you have logged in, a shell is opened in which we will execute various bioinformatical methods (or click 'new terminal console' in the left menu). An SFTP window is also opened which you can use to transfer files between the Windows and Linux systems – you have a file storage area on both systems (or click 'New SFTP window' in the left menu). A list of basic commands that can be used in the shell can be found in Table 1 below.

Note that in this part of the exercise, we will only work with a small subset of the sequencing data (approximately 10% of the total number of reads). The reason for this is to minimize the waiting time and to be able to perform the analysis within the disk quota of your student accounts. However, we will use the full dataset in step 2 of the exercise.

Table 1: A list of important Linux commands

Linux command	Description
ls	Lists all files in the current directory
cd directory	Changes the working directory to 'directory'
Cd	Changes the working directory to the home directory
pwd	Prints the full path of the current working directory
rm file	Removes 'file'
rm –R directory	Removes directory and all its content (use with caution!)
less file	Shows 'file' on the screen (use 'q' to quit **less**)
nano file	Opens 'file' in the basic editor nano

Exercise 2.1: Downloading the sequence files to your Linux account

Log in to your Linux account. After successfully logging in to the server, you will be placed in your home directory. You can, at any point, move back to this directory by simply writing

$ cd

'cd' is short for 'change directory' and when no specific directory is given, it takes you to your home directory.

Create a directory 'MVE510E2' to store the data files used in the computer exercise. This is done using

$ **mkdir MVE510E2**

where 'mkdir' is short for 'make directory'. You can, if you want, use an alternative name for the directory or place it at any other location. Note, however, that the names of directories and

files in the Linux environment are case-sensitive, which means that 'mve510E2' and 'MVE510E2' are not identical.

Enter the new directory by typing

$ **cd MVE510E2**

Next, download the sequence data for the computer exercise using the command 'wget', which retrieves files available from the web. The files for this exercise are located at a local server here at Chalmers and to download them, type

$ wget http://bioinformatics.math.chalmers.se/courses/MVE510/genome1.fq.gz

After the download is complete, the file will be placed in your current working directory. The downloaded file is compressed with **gzip** to save space on the hard drive. To decompress the file, type

$ gunzip genome1.fq.gz

This will generate a new file called **genome1.fq**. To view the content of this file, type

$ less genome1.fq

Note that you quit less by typing 'q'. Is the file a proper FASTQ file? Can you identify the different parts?

Download and repeat this procedure for the two other genomes (**genome2.fq.gz** and **genome3.fq.gz**).

Exercise 2.2: Preprocessing of the data

Our next aim is to assess the quality of the downloaded sequence data. This will be done using software called **fastqc**, which calculates statistics about reads from next-generation sequencing, including curves of the quality score. This software is run directly on a FASTQ file and summarizes the quality of the data. To run **fastqc**, type

$ fastqc genome1.fq

The results from **fastqc** will be saved in compressed directory a named genome1.fastq.zip in the same directory (type 'ls' to list all the files). Transfer the file to your Windows account, unzip it, and open the file 'fastqc_report.html' in your browser. How many sequences did the file contain? How are the quality scores distributed over the reads? What is the G/C content of the reads?

Repeat the quality control for the other two files (genome2.fq and genome3.fq) and examine the results. How does the quality compare between the samples? Which sample has the worst quality? Are there any other differences?

To remove nucleotides with too low quality, the data need to be processed. For this computer exercise, this will be done using the toolkit **fastx**, which removes reads or parts of reads based on their quality scores. The software takes a FASTQ file, a set of parameters, and generates a

new FASTQ file containing only the reads that passed the pre-defined quality cut-off. **fastx**, similar to **fastqc**, runs in the Linux environment.

From the **fastx** toolkit, we will use **fastq_quality_filter** to filter the reads. This command takes two parameters: the quality score threshold and the minimum percent of nucleotides that need to be above this score threshold. Thus, any read that does not satisfy these parameters will be removed from the file (i.e. has more than the 'minimum percent of nucleotides' with a quality score less than the threshold). We will use a quality score threshold of 30 and set the minimum percent of nucleotides to 80, i.e.

$ fastq_quality_filter -i genome1.fq -o genome1.filtered.fq -q 30 -p 80 -Q64

Here, the option flag '-i' specifies the input file and '-o' the output file. The flag '-Q64' sets the encoding of the quality scores used for the FASTQ files we are using in this exercise.

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?

Optional: Rerun the preprocessing steps using different parameters and view the results by running **fastqc**. What happens if you use more strict or less strict values?

Exercise 2.3: Mapping the reads to a reference

To compare the genomes from the three samples, each of them needs to be aligned to a reference. In this exercise, we will use a wild-type reference of *E. coli* called strain K12 MG1655. The sequence of the reference can be downloaded using **wget** by typing

$ wget http://bioinformatics.math.chalmers.se/courses/MVE510/reference_Ecoli_K12_MG1655.fasta

Alignment of the read against the reference will be done using the tool BWA (Burrows-Wheeler Alignment). BWA works in two stages, by first finding seeds using suffix arrays and the Burrow-Wheeler transform. The read is then extended around the seed using the Smith-Waterman algorithm. To find the seeds, we first must calculate the index, consisting of the BWT and the corresponding suffix array for the reference genome. This can be done by typing

$ bwa index reference_Ecoli_K12_MG1655.fasta

After the index has been created, BWA can align reads to the reference. Mapping of the reads to the reference is done by proving BWA with the reference and the file with the sequence reads

$ bwa mem reference_Ecoli_K12_MG1655.fasta genome1.filtered.fq > genome1.sam

The option 'mem' indicates which specific BWA algorithm to use (BWA contains other algorithms not covered by this course). Typing '> genome1.sam' at the end of the call tells BWA to save the output to the file 'genome1.sam'. This file will be in SAM format, which stands for 'sequence alignment'. Even though SAM files are rather complex, they can be

viewed in text format using e.g. **less**. Note that, depending on the screen size, some rows may overflow to the following rows, so you may try to type the following to increase readability

$ less -S genome1.sam

Note that you quit less by typing 'q'. The SAM file is a tab-delimited file that starts with a header. Each row corresponds to an aligned read and each column contains specific information of the match to the reference genome.

Give an example of two different values contained in the columns FLAG, RNAME, POS, and MAPQ in your SAM file and explain their meaning. An overview of the SAM file format can be found on Wikipedia and more detailed descriptions at https://samtools.github.io/hts-specs/SAMv1.pdf.

Exercise 2.4: Viewing the results in Integrative Genome Viewer (IGV)

Integrative Genome Viewer (IGV) is software for visualizing results from read mapping. IGV is installed in the Windows system. Before you transfer our SAM files to Windows systems, they need to be converted to a format that IGV can read. This is done in three steps using the tool **samtools**

$ samtools view -b genome1.sam > genome1.bam
$ samtools sort genome1.bam > genome1.sorted.bam
$ samtools index genome1.sorted.bam

This converts the SAM file into a sorted and indexed BAM file. An indexed BAM file is organized in a much more efficient way to make it faster to read. As a consequence, the BAM file is no longer in a readable text format (the 'B' in BAM stands for 'binary').

Transfer the resulting files **genome1.sorted.bam** and **genome1.sorted.bam.bai** to your Windows account and start IGV. Before you load the files into IGV, you need to load an annotation file corresponding to the reference genome we used. The annotation files are available at http://bioinformatics.math.chalmers.se/courses/MVE510/K12.genome.zip (it can be downloaded using a web browser). Save **K12.genome.zip** on your Windows account in the same place as the previous files and unzip it. In IGV, first, load the annotation file **K12.genome** by selecting Genomes->Load Genome Form File. Then, load your alignment by selecting File->Load from File and then select **genome1.sorted.bam.**

Describe what you see. Note that you need to zoom in before any information is shown (using the '+' in the top right corner). 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?

Step 2 – Identification of mutations

In this part of the exercise, we will focus on identifying mutations present in the three bacterial isolates. This will require analysis of each position in the three genomes. For this analysis, we will use Rstudio/R on the Windows platform. The alignment done in the previous

step of the computer exercise was done on a subset to ensure small files and fast computations (and thus, as little waiting time as possible).

Exercise 2.5: Loading the data into R

In this part of the exercise, we will work with the complete dataset. This data is available in R data files generated directly from the SAM files produced by BWA (identical to what you did previously, but using the complete data). Use a web browser to download the datafiles 'genome1.rdata', 'genome2.rdata', and 'genome3.rdata' from http://bioinformatics.math.chalmers.se/courses/MVE510/ to a directory in your Windows account. Start R/RStudio and change its working directory to the place where you downloaded the files. You can then load them using

> load("genome1.rdata")

Repeat the process for the two other data files. Once loaded, you will get four new objects called 'genome1', 'genome2', 'genome3', and 'reference' (you can check this by using **ls()**). Familiarize yourself with these objects. How are they organized? Note that the objects are quite large, so the functions **class**, **dim**, **length**, and **head** may be useful here.

An important tip for this part of the computer exercise

The three data objects are large which means that some of the calculations may take some time. This is a very common complication when working with big and complex data. A *strong* recommendation is, therefore, to create a subset of the first genome which you can use to test the code you write. For example, write

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

to store the first 1000 positions in of the first genome in the variable **genome1.subset** and the first 1000 positions of the reference in **ref.subset**. Processing **genome1.subset** will be much faster than the full genome since it only contains the first 1000 positions. This makes it suitable to verify that the code that you are writing is working properly. Once your code works, you can use it on the complete data.

Exercise 2.6: Coverage

The *coverage* is the total number of sequence reads covering a specific genomic position. For genome1, use the function **apply** together with **sum** to calculate the coverage for each position. 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 questions using the full dataset of at least one of the genomes.

Exercise 2.7: Error rate

Sequencing errors are common, even after the data has been quality-assessed. Calculate, for each position, the proportion of reads that do not match the reference in **genome1**. How many positions have at least one read with a mismatch? Visualize the proportion of mismatching reads over a region covering 1,000 positions.

Hint: This exercise can be efficiently solved by a **for-loop** looping over the number of positions in the genome. Remember that the **for-loop** in R has the following syntax

```
# Calculate the length of genome1
genome1.length=nrow(genome1)

# Allocate an empty vector for matches
matches=vector(length=genome1.length)

# Syntax of for-loop in R
for (pos in 1:genome1.length){
  # Get the number of reads that have the same nucleotide
  # in position 'pos' as the reference
  matches[pos]=genome1[pos,reference[pos]]
}
```

The code above will, for each position, get the number of reads that have the same nucleotide as the reference. Note that this is only possible due to the column names of genome1 – for each position, the column with the same name as the reference nucleotide will be used (use **colnames** on the **genome1** data object to see what they are).

Exercise 2.8: A test for single nucleotide polymorphisms

Next, we aim to identify the position with mutations, i.e. true differences between the sequenced genomes and the reference that are not caused by sequencing errors. This will be done using a statistical test. The test will assume that sequencing errors appear randomly and independently between reads and nucleotides. Let the random variable Y_i be the number of mismatches at position i, i.e. the number of reads that do not have the same nucleotide as the reference. Furthermore, let N_i be the coverage at position i and p_i be the probability for observing a nucleotide different from the reference at position i. For positions where there are no mutations, we expect p_i to be small describing only the sequencing errors. For positions where we have a mutation, p_i is instead expected to be large (close to 1 since bacteria are haploid, i.e. only carrying one copy of the chromosome). We will therefore use a test to compare the following null hypothesis against the alternative hypothesis

$$H_0: p_i = p_{\text{error}}$$

$$H_1: p_i > p_{\text{error}}$$

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?

We can then, based on our observed data, y_i, test H_0 against H_1 by calculating

$$\text{p-value} = \text{Prob}(Y_i \geq y_i) = \sum_{j=y_i}^{N_i} p_{Y_j}(j).$$

This test is called a binomial test and is implemented in R in the function **binom.test**. Use the R help to read about the **binom.test** function and the parameters it requires. Then, implement an R-function that takes the number of mismatches and the coverage at a specific position and returns a p-value according to the test above. 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 setting p_{error} to be larger than the average error rate?

Hint: Make sure that your function can handle a position that has a coverage of zero (since such positions are present in the data). Since these positions are not covered by any sequence read, they contain no information. An **if**-statement at the beginning of your function can be used to handle such positions.

Exercise 2.9: Screening the genome for SNPs

Apply the function you wrote in the previous exercise to each position in the genome. Again, this can easily be solved using a **for-loop**. Remember that you need to save the output from your test function, i.e. the p-values, in a vector. After you are done, use **order** to identify the most significant p-values. 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?

Note that this analysis may take some time (~10 minutes per genome) so make sure that it works properly before you apply it to the full genome (this may also be a good time for a coffee break).

Exercise 2.10: Interpretation of the SNPs – what do they mean?

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. This can be done by first looking up the reference genome in the NCBI GenBank database at https://www.ncbi.nlm.nih.gov/nuccore/NC_000913.3. This page shows the full annotation of the reference genome we have used. Once you have located what gene that is located at the position of the mutation you can click on the corresponding GeneID (the line that says '/db_xref=GeneID:XXXXX'). On the new page, you will find a genome browser (under 'Genomic regions, transcripts, and products') showing the annotations. By zooming in to 100%, you can view both the DNA and the corresponding amino acid sequence. The 'Find' search bar lets find specific positions. The easiest way to get the codon where the mutation is located is to calculate the difference between the starting position of the gene and the position of the mutation and adjust for that each codon has a length of three nucleotides. Be aware that these pages contain a lot of information so take your time.

Once you know about the gene, the codon that is mutated, and its alternative codon, use PubMed (https://www.ncbi.nlm.nih.gov/pubmed) to search for relevant literature. Can you conclude that any of the isolates may be resistant to antibiotics? Which isolate and what kind of antibiotics?