BME 590 Homework #1

In this homework, the objectives are to:

1. *Become familiar with nucleotide sequencing data formats*
2. *Become comfortable using command line tools for nucleotide sequencing data*
3. *Create first shell script. Become comfortable generating and running scripts that perform sequencing data manipulation*

Next generation sequencing (NGS) platforms often produces sequencing data in FASTQ format. FASTQ is an extended version of FASTA, which is a text-based format representing nucleotide sequence data. Basic structures of FASTA and FASTQ files are shown in Figure 1 and Figure 2, respectively.

FASTA files (Figure 1) represent each sequence in 2 parts:

- Header starts with ">" and contains description of the sequence.
- Next line(s) contain(s) the raw sequence data. Note that this section may be composed of either a single long line or multiple short lines.

Figure 1. A sample of the FASTA file. Image adapted from Hosseini *et al. Information* (2016).

FASTQ files (Figure 2) usually use 4 lines per sequence:

- Line 1 contains the identifier of the sequence and starts with "@"
- Line 2 is the raw sequence
- Line 3 is starts with "+", optionally followed by description (or sometimes reprinted identifier)
- Line 4 shows Phred quality score for the raw sequence from line 2

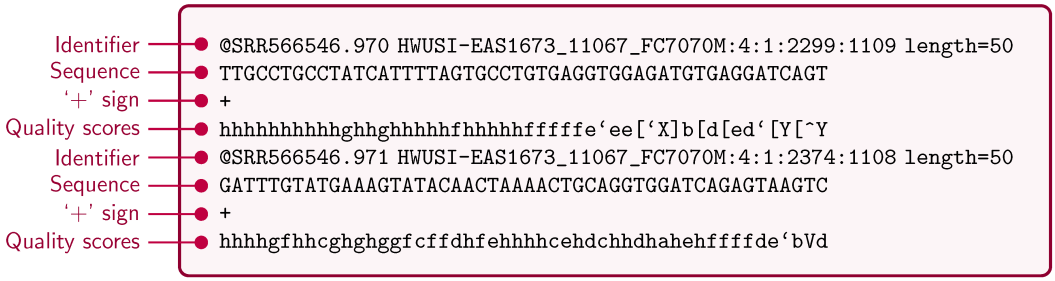

Figure 2. A sample of the FASTQ file. Image adapted from Hosseini *et al. Information* (2016).

The provided sequencing data are adapted from Diabimmune (https://pubs.broadinstitute.org/diabimmune). This study was published in Science Translational Medicine in 2016 (Yassour M, *et al.* 2016) and the pdf copy of this work can be found in Sakai.

The provided FASTA file has 16S ribosomal RNA (rRNA) sequencing data and the FASTQ file contains shotgun metagenomic sequencing data. These two sequencing methods are widely used in microbiome research to understand microbial composition and diversity in samples. Briefly, because 16S rRNA gene contains highly variable regions that are unique to a species, these regions can serve as identifiers of specific bacterial species. Shotgun metagenomic sequencing is also often used for analyzing the composition of microbiome. However, unlike 16S rRNA sequencing, which requires an amplification of the 16S rRNA region, shotgun metagenomic sequencing reads the entire genomes of all organisms present in the sample. Shotgun metagenomic sequencing data can provide a useful insight on genetic compositions within the samples. In this paper, specifically, the 16S rRNA data is used for understanding microbial composition in the samples, whereas the metagenomic data is analyzed for detecting strain-level variation and antibiotic resistance gene reservoir.

For more information on how these technologies work and what their differences are, refer to: https://blog.genohub.com/2018/04/12/16s-sequencing-vs-shotgun-metagenomics-which-one-to-use-when-it-comes-to-microbiome-studie/

Please include ALL commands/codes you use to get the answers. Be sure to include the printed outputs when appropriate.

1. Download the FASTA and FASTQ data from Sakai
   Note: .gz extension denotes that these files are compressed
   Note: .fna is a FASTA file for nucleotides data

2. Briefly state the objective and main findings of this study (Yassour M, *et al.* 2016) using 2-3 sentences.

3. When DNA sequencing data is really large (which is the case a lot of times), it is often preferable to work with the zipped version of the files. Look the first 10 lines (header) of the zipped FASTQ file in a human-readable mode – without unzipping it.
   Hint: use "gzcat" command; try looking it up on Google to learn what this function does

4. If your dataset is very large, it may be useful to extract a small subset of the original data to test out your code to save time and effort. Extract the first 100 lines of the FASTQ file and save it as **smallFQ.fastq**, without unzipping it.
   Hint: use "gzcat" command; try looking it up on Google to learn what this function does

5. Count the number of sequences (or reads) in the FASTQ file. Note that each sequence's information is shown over 4 lines (Figure 2). You may choose to unzip the file first if you want, but it is not necessary to get the answer.

6. What is the size of the zipped FASTA file?

7. Unzip the FASTA file

8. What is the size of the unzipped FASTA files?

9. Count the number of sequences in the FASTA file

10. What is the content of line 4321 of the FASTA file?

11. How many of the 16S rRNA sequencing reads in the FASTA file are from *Bacteroides fragilis*? Hint: Try to look for "CGTAAAATTGCAGTTGA", which is a 16S gene-specific sequence for *Bacteroides fragilis* (Okabe *et al* 2007)

12. The current FASTA file stores the raw sequence spanning over multiple lines for readability. For instance, lines 2 - 6 of the given FASTA file compose one sequence. However, single line sequences are easier to manipulate in downstream data analysis. Write a code to convert this multiline FASTA file into a single line FASTA file and save this new single line FASTA file named as singleFA.fasta. Briefly explain what each part of your code does.

13. How many rows are there in the new single line FASTA file singleFA.fasta?

14. Run the code below to import an example 1000 genomes variant data file and unzip it.

    ```
    $ wget
    ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/pilot_data/release/2010_07/trio/snps/trio.2010_06.ychr.sites.vcf.gz
    ```

15. Take a look at the first few lines of the vcf file. What is the reference of this data?

16. How many entries have quality scores higher than 50?

17. How many entries are there between positions 10,000,000 and 20,000,000?

Shell Scripting

1. First log into Duke Compute Cluster (DCC) by typing below in your terminal:

   ```
   $ ssh YOUR_NETID@dcc-slogin-01.oit.duke.edu
   ```
   NOTE: YOUR_NETID should be replaced with your own netid

2. Once you confirm that you are in your home directory, let us now generate a simple shell script named **myscript.sh**. Follow the below steps:

   ```
   $ echo '#!/bin/sh' > myscript.sh
   $ echo 'echo Hello World!' >> myscript.sh
   ```

3. Run the code below, which prints the file path of your new script. What is your output?

   ```
   $ readlink -f myscript.sh
   ```

4. Who has the permission to read this script? Who can write in this script? Who can execute this script?

5. Change the permission setting so that everyone can execute this script. Check if the permission has successfully modified. What is printed as the output?

6. Now execute this script. What is printed?

7. Write a new shell script named **read_counter.sh** that can count the number of reads from an input vcf file. We will use the same vcf file that we used in previous problems.
   (1) First, we need to download the vcf file to our DCC home directory. Run the command from Question 14 from above in your DCC home directory.
   (2) Unzip the file
   (3) Write a new shell script named **read_counter.sh** to count the number of the reads from our vcf file.
   Hint: Note that all the reads in this vcf file are from Chromosome Y. You may want to use this information when counting the number of lines

   Make sure you show all of your code that you used to write and execute the script for printing the number of reads. Verify your script by comparing the output with your answer to Question 9 above.