Deliverable 2C: Working with Illumina sequencing reads

Log onto a computing node before beginning this exercise.

- Perform the operations below first using the week4 sequence files.
- Then repeat the operations using the Day data files in the shared folder.

You've downloaded and uncompressed a set of sequencing files into their own directory. The original file ended in tar.gz. Now the various files that resulted from uncompressing end in:

Write a shell script that:

- unzips each file but leaves the original copy intact

```
For filename in *.fastq.gz
Do

Base=$(basename $filename.fastq.gz)
Echo ${base}

gunzip -c -d ${base}.fastq.gz > ${base}.fastq
```

Now the directory has two sets of files, each with a different extension. Make a new directory and move all the unzipped files to it.

```
mv *.fastq unzipped_files/
```

But wouldn't it have been more efficient to include this step in the original shell script? Let's try that. First, remove the directory that contains the unzipped files.

```
Rmdir unzipped_files
```

Then go back to the shell script and amend it to include commands to:

- make a new directory
- place the unzipped files there

```
for filename in *.tiny.fastq.gz
do

base=$(basename $filename .tiny.fastq.gz)
echo ${base}

gunzip -c -d ${base}.tiny.fastq.gz > ${base}.fastq
mkdir fastq
 mv *.fastq fastq

done
```

Use the `grep` command to determine the number of reads in one unzipped file

```
grep -c '>' SRR6805881.fastq
```

then in all the unzipped files, printing to the screen

```
grep -c '>' *.fastq
```

then in all the unzipped files, printing to a new file.

```
grep -c '>' *.fastq > all_reads.txt
```

Use head to examine the start of each read and see if there is an obvious adapter sequence to trim.
The start of each read has "TGCAG" as the adapter sequence. They are all the same!

Now you want to generate quality control reports for each of the files, using fastqc/0.11.9. This package requires that you first load the module OpenJDK/19.0.1, and the command operates on zipped files (ending in fastq.gz).
First test out the operation on only one file, using the command line.

Now try performing this task on all the files, in two different ways-- using a shell script and using a bash script.

Shell script:

```
module load OpenJDK/19.0.1
module load fastqc/0.11.9

for filename in *.tiny.fastq.gz
do

base=$(basename $filename .tiny.fastq.gz)
echo ${base}

fastqc ${base}.tiny.fastq.gz

done
```

Bash script:

```
 #!/bin/bash
#SBATCH --partition=short
#SBATCH --job-name=qr
#SBATCH --time=24:00:00
```

```
#SBATCH --nodes=1
#SBATCH --cpus-per-task=2
#SBATCH --mem=256G
#SBATCH --output=%j.output
#SBATCH --error=%j.error

module load OpenJDK/19.0.1
module load fastqc/0.11.9

for filename in *.tiny.fastq.gz
do
base=$(basename $filename .tiny.fastq.gz)
echo ${base}

fastqc ${base}.tiny.fastq.gz
done
```

After generating the new files, make a separate directory and put the .html files into it (you can try adding this step directly to your script from above, if you like).

Open one of the .html files (you'll need to do this through the OOD, using a browser to view it).

Watch this video on interpreting the report.

All Unit 2 work is due by Mar 10.

Each deliverable is complete when you have:

- answered each question
- saved a terminal session or screenshots demonstrating your performance of the commands (on Discovery or posted on GitHub)
- indicated in a "TOC" (table of contents) file where your work is found (GitHub repo or specific path/name_of_file on the cluster).