Name: Dien Mach
UNIX (GCP) Username: ethanmach1998

Metagenomics Midterm Notes

About Project:

All files are stored on Google Cloud in the 'AS41073481-dmach1' project within the 'metagenomics' instance.

NOTE: My username on the on the Google Cloud instance is 'ethanmach1998'

Data Gathering:

For the metagenomic sample, I downloaded and used the HMP mock community reads obtained using Roche 454 sequencing technology. The sample is an artificially generated dataset with an even amount of DNA used for each organism. The FASTQ and FASTA files were downloaded from the Dropbox link provided and transferred onto the Google Cloud Instance at:

/home/ethanmach1998/project/samples/hmp_mock_454_even/SRR072233.fastq
/home/ethanmach1998/project/samples/hmp_mock_454_even/SRR072233.fasta

Referencing the table of DNA compositions and reference genomes for the HMP mock community datasets, I chose 10 different bacterial genomes for reference [1]. I downloaded data for the bacteria (shown in the table below) from the NCBI website and uploaded the data onto:

/home/ethanmach1998/project/hmmc_refgenomes/{taxon_id}

Organism name & repository number	Taxon ID	Contig/Scaffold Count	GenBank Gene Count
Acinetobacter baumannii ATCC 17978	470	2/2	3839
Staphylococcus epidermidis ATCC 12228	1282	2/2	2354
Streptococcus pneumoniae ATCC BAA-334	170187	1/1	2292
Deinococcus radiodurans DSM 20539	243230	4/4	3305
Lactobacillus gasseri DSM 20243	324831	26	1786
Actinomyces odontolyticus ATCC 17982	411466	4/2	2214
Methanobrevibacter smithii ATCC 35061	420247	1/1	1837
Bacteroides vulgatus ATCC 8482	435590	1/1	4183
Enterococcus faecalis ATCC 47077	474186	75/NA	2584
Bacillus cereus ATCC 10987	2026187	239/222	5807

Figure 1: Table of all the reference genomes downloaded from the NCBI website.

Installing Packages and Dependencies

Firstly, I downloaded and ran the Anaconda installer package on the 'metagenomics' instance, following instructions from its official documentation [3]. Next, referencing the Bioconda installation page, I adjusted the Anaconda parameters to enable and install Bioconda [4]. Afterwards, I installed SnakeMake and BowTie2 via the conda package manager [5,6].

NOTE: 'bowtie2_env' conda environment was created using:

```
$ source ~/anaconda3/bin/activate  # activate base Conda base environment
$ mamba create --name bowtie2_env bowtie2
```

NOTE: 'snakemake' conda environment was created / installed using [6]:

```
$ conda install -n base -c conda-forge mamba
$ conda activate base
$ mamba create -c conda-forge -c bioconda -n snakemake snakemake

# activate 'snakemake' conda environment via:
# conda activate snakemake
```

For the BWA alignment algorithm, I referenced the public repo on GitHub and installed via [7]:

```
$ git clone https://github.com/lh3/bwa.git
$ cd bwa; make
```

Lastly, I also installed SAMtools following the instructions on their website and added its installation path to my bash $PATH variable [8].

```
$ cd samtools-1.x    # and similarly for bcftools and htslib
$ ./configure --prefix=/where/to/install
$ make
$ make install
```

Alignment to Reference Genomes

NOTE: The home directory (~/) refers to /home/ethanmach1998/ and the project directory refers to /home/ethanmach1998/project

Running contig_combiner.py

The two alignment algorithms I chose to use were BowTie2 and BWA. However, before running the alignment algorithms, I wrote a Python script called 'contig_combiner.py' where users input fragmented and unfragmented reference genome FASTA files, outputting a new FASTA file which links contigs with an 'XXXXX' linker chain of nucleotides. After standardizing all reference genome FASTA files with the contig_combiner.py script and combining all contigs / scaffolds (if any) in each genome, I proceeded with the alignment:

```
$ ~/project/scripts/contig_combiner.py {ref_genome}.fna
```

The output is '{ref_genome}_combined.fna', which I will now refer to as {combined_ref_genome}.

NOTE: All {combined_ref_genome} and {ref_genome} files are stored as:

```
{combined_ref_genome} =
        ~/project/hmmc_refgenomes/{taxon_id}/{taxon_id}_combined.fna
{ref_genome} =
        ~/project/hmmc_refgenomes/{taxon_id}/{taxon_id}.fna
```

Running BowTie2

The general workflow for running BowTie2 is to activate the 'bowtie2_env' conda environment, build the BowTie2 index files for each reference genome, and then run the actual BowTie2 algorithm. The BASH commands are shown below with comments and relevant parameters explained:

```
# activating bowtie2_env conda environment
$ source ~/anaconda3/bin/activate
$ conda activate bowtie2_env

# building BowTie2 indices for one reference genome
$ bowtie2-build {combined_ref_genome} {bowtie2_index_base_name}
```

NOTE: {bowtie2_index_base_name} represents the base name to which BowTie2-build will create several indices files with the same base name

eg:

```
$ bowtie2-build ~/hmmc_refgenomes/1282/1282_combined.fna
~/hmmc_refgenomes/1282/1282_index
```

example output:
~/project/hmmc_refgenomes/1282/1282_index.1.bt2
~/project/hmmc_refgenomes/1282/1282_index.2.bt2
~/project/hmmc_refgenomes/1282/1282_index.3.bt2
~/project/hmmc_refgenomes/1282/1282_index.4.bt2
~/project/hmmc_refgenomes/1282/1282_index.rev.1.bt2
~/project/hmmc_refgenomes/1282/1282_index.rev.2.bt2

```
# bowtie2 alignment
$ bowtie2 -x {bowtie2_index_base_name} -U {reads}.fastq  -S {SAM output file name}
```
-x: reference genome index base name
-U: unpaired sample reads input file
-S: SAM output file name

All resulting index files and SAM files are stored in: ~/project/hmmc_refgenomes/{taxon_id}/

Running BWA Mem:

To run BWA, users need to build index files for each reference genome, and then align via the appropriate BWA algorithm. BWA has several alignment algorithms to choose from in the installed package; however, BWA mem was most appropriate for the 454 sequencing reads based on the instructions [7].

BWA is quite simple to use and has fewer parameters than BowTie2.

```
# build bwa index files
$ bwa index {combined_ref_genome}
```

This command outputs a series of index files in .amb, .ann, .bwt, .pac, .sa format.
eg: "1282_combined.fna.amb"

Then, to run the alignment:

```
# run bwa mem alignment
$ bwa mem {combined_ref_genome} {reads}.fastq > {SAM output file name}
```
NOTE: the index files must be in the working directory when running the command

All BWA index files and resulting SAM files are stored in:
~/project/hmmc_refgenomes/{taxon_id}/

Filtering SAM Reads:
Both BWA mem and BowTie2 produced SAM alignment files. To filter out unmapped reads in the SAM files and make the outputs more concise, the SAMtools package was used. More specifically, I used the 'samtools view' command.

```
# filter out unmapped reads
```

$ samtools view -h -F 4 {input_alignment_file} > {output_alignment_file}

samtools view: the input {input_alignment_file} is default SAM but can also be BAM or CRAM; the standard output is SAM format with no header

-h: includes header in the output file

-F 4: filters out unmapped reads in the input files

Justification for alignment methods:

I originally chose to use FR-HIT and BowTie2. I chose these two algorithms because of the literature students were provided in week 6. BowTie2 was easy enough to use, more updated than BowTie, and produced a SAM file. Meanwhile, FR-HIT provided a .psl format which could be converted to SAM format via psl2sam.pl in the SAMtools package.

Unfortunately, even though I could convert the psl output from FR-HIT to SAM, I came across many issues trying to calculate the necessary MD and header lines in the SAM file. As a result, I abandoned FR-HIT and used BWA instead. BWA mem is fairly new, easy to run, and also produces a SAM file with appropriate header and MD lines.

The Sequence Alignment Map (SAM) file is a text-based format which stores information about sequences aligned to a reference sequence. The header section starts with lines that have the '@' symbol at the beginning followed by the alignment sections with 11 fields such as QNAME, FLAG, RNAME, POS, MAPQ, CIGAR, and more [9]. The POS field, CIGAR line, MD line, and header lines are most important in the plotting script I use as they are used to store various fields and perform several calculations which will be discussed further below.

Quick Comparison of Statistics

When using both alignment methods, BowTie2 contained the runtime as well as the alignment rate printed to the console. Meanwhile, BWA only outputs the runtime. Comparing the two runtimes of the two algorithms, BowTie2 took 240 seconds while bwa mem took 472 seconds when aligning to the reference genome for taxon id: 470. When comparing the runtimes for alignment against Taxon ID 1282, the runtime for BowTie2 vs. BWA was 148 seconds vs. 409.23 seconds.

For the BowTie2 alignment rates for Taxon ID 470 and 1282, the alignment rate was 11.51% and 3.92% respectively.

As BowTie2 ran faster and provided more descriptive statistics upon output, I will be mainly discussing BowTie2 in the analyses.

Generation of Fragment Recruitment Plots (FRPs) Using Alignment Data

To plot FRPs using alignment data, I wrote a script called 'sam_to_frp.py' found in
~/project/scripts/sam_to_frp.py

The script is reusable and takes a SAM file input, parses the SAM file, and uses matplotlib's plotly library to generate an FRP. The Python script is composed of two main functions: parse_sam and generate_frp. parse_sam takes the inputted SAM file, stores the reference genome name, calculates the percent identity using the CIGAR line and the MD line, and stores the starting position of each alignment. Since the function calculates the matches, mismatches, and deletions for percent identity, the input SAM file must have the CIGAR, MD, and header lines. Moreover, percent identity was calculated as:

percent identity = (actual matches) / ((matches) + (mismatches) + (deletions))

Afterward, a tuple is returned containing a list of starting positions where matches occur, a list of percent identities calculated, the reference genome name ([pos_list], [perc_identity_list], [ref_genome].

The second function 'generate_frp' takes the tuple data from the first function and stores various fields of the file name, creating a simple FRP via matplotlib.plotly, and saving the output plot as a .png file.

The script can be run like:

$ ~/project/scripts/sam_to_frp.py {SAM input}

The output is created in the same directory as the {SAM input}

Snakemake Workflow and Diagram:

One challenge I faced when doing this project was the amount of commands I had to use to properly align the reads to the reference genomes, then generate the FRP plot. As a result, I learned to use Snakemake, which is a workflow engine that allows for bioinformatics workflows or pipelines to be built using Python-like language [10]. Essentially, the workflow engine pipes together multiple Snakemake rules allowing for dependencies to be made. As a result, instead of typing the multiple commands numerous times (as shown above in the previous sections) for different reference genomes, users can write one command for each reference genome instead.

I wrote a bash script called 'main.sh' which contains all 10 snakemake command runs which aligns the metagenomic reads to all 10 reference genomes using both BWA mem and BowTie2, outputting a total of 20 FRPs. For example, the general snakemake line command is:
$ snakemake
"/home/ethanmach1998/project/hmmc_refgenomes/{taxon_id}/{taxon_id}_bwa_frp.png"
--cores all

--cores: This parameter just indicates how many computer cores you want to pass to the workflow engine, with the minimum value being 1 and the maximum being 'all'

Here, we can see that the snakemake command can simplify bash commands to simply typing the downstream product of the workflow.

To help visualize the workflow, I drew a diagram below to help visualize the workflow and rule dependencies (Figure 2). The snakefile containing all the rules and shells commands used is found in ~/project/scripts/Snakefile

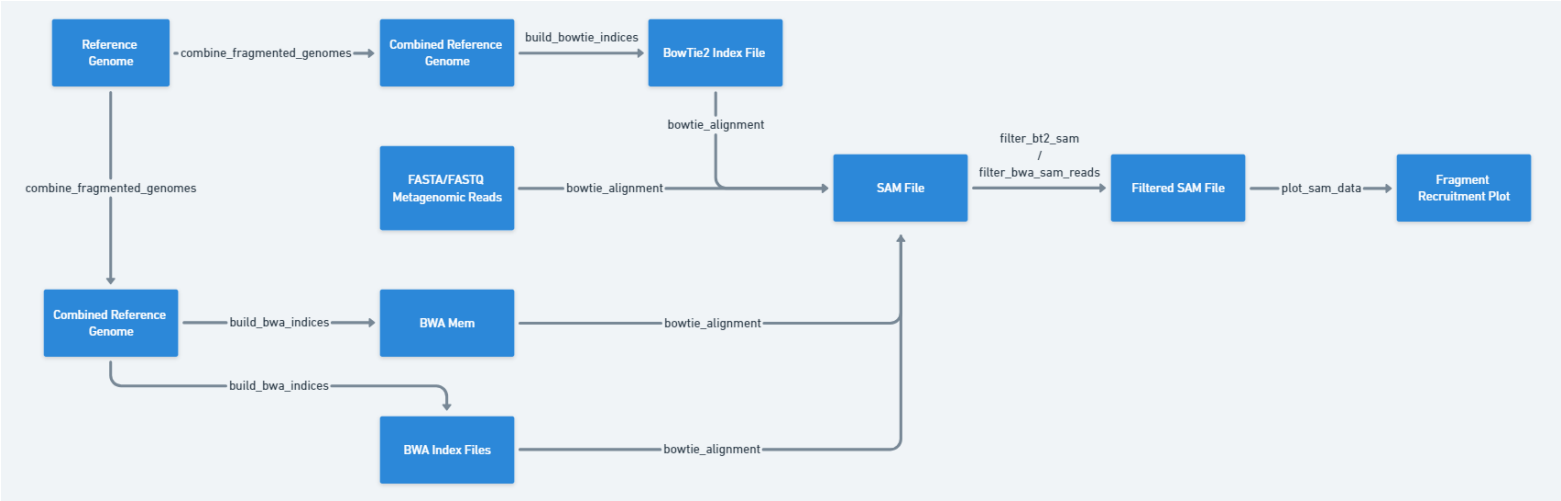

Figure 2: Snakemake Workflow Diagram demonstrating the Directed Acyclic Graph pipeline.

Analysis

Which of your reference genomes is most represented in your sample and what percentage of the genome is covered by your reads?

- Below, I have included the outputs of all 10 reference genomes aligned using BowTie2's algorithm. The outputs show which percentage of each genome is covered by the reads by the overall alignment rate.
- The reference genome most represented is Taxon ID 243230 which belongs to *Deinococcus radiodurans DSM 20539* with a 35.24% overall alignment rate.

1282
51705 (3.73%) aligned exactly 1 time
2668 (0.19%) aligned >1 times
3.92% overall alignment rate

170187
37066 (2.67%) aligned exactly 1 time
2087 (0.15%) aligned >1 times
2.82% overall alignment rate

470
154319 (11.13%) aligned exactly 1 time
5295 (0.38%) aligned >1 times
11.51% overall alignment rate

243230
469245 (33.85%) aligned exactly 1 time
19302 (1.39%) aligned >1 times
35.24% overall alignment rate

324831
665 (0.05%) aligned exactly 1 time
0 (0.00%) aligned >1 times
0.05% overall alignment rate

411466
31989 (2.31%) aligned exactly 1 time
522 (0.04%) aligned >1 times
2.35% overall alignment rate

420427
7232 (0.52%) aligned exactly 1 time
141 (0.01%) aligned >1 times
0.53% overall alignment rate

435590
107435 (7.75%) aligned exactly 1 time
6072 (0.44%) aligned >1 times
8.19% overall alignment rate

474186
16883 (1.22%) aligned exactly 1 time
27 (0.00%) aligned >1 times
1.22% overall alignment rate

2026187
10395 (0.75%) aligned exactly 1 time
50 (0.00%) aligned >1 times
0.75% overall alignment rate

Comparison of FRPs

Since TaxonID 243230 and 470 *(Acinetobacter baumannii ATCC 17978)* had the highest alignment rates, I will show only their FRP graphs for brevity purposes. The other 16 FRPs .png files can be found in ~/projects/frp

When comparing the number of reads recruited shown in Figure 3 and 4, there is a noticeable difference. BowTie2 has a much higher density of reads as the graph compared to BWA, as shown in blue. As a result, The FRPs indicate that BowTie2 recruits more reads in a shorter time. However, a study by Li suggests that BWA-MEM is more accurate than BowTie2, as shown in their Figure 1 [11]. Moreover, the same study claims that the runtime of BWA-MEM is as similar or up to 6x faster than BowTie2 [11]. However, my results were different as BowTie2 aligned in a faster runtime compared to BWA MEM.

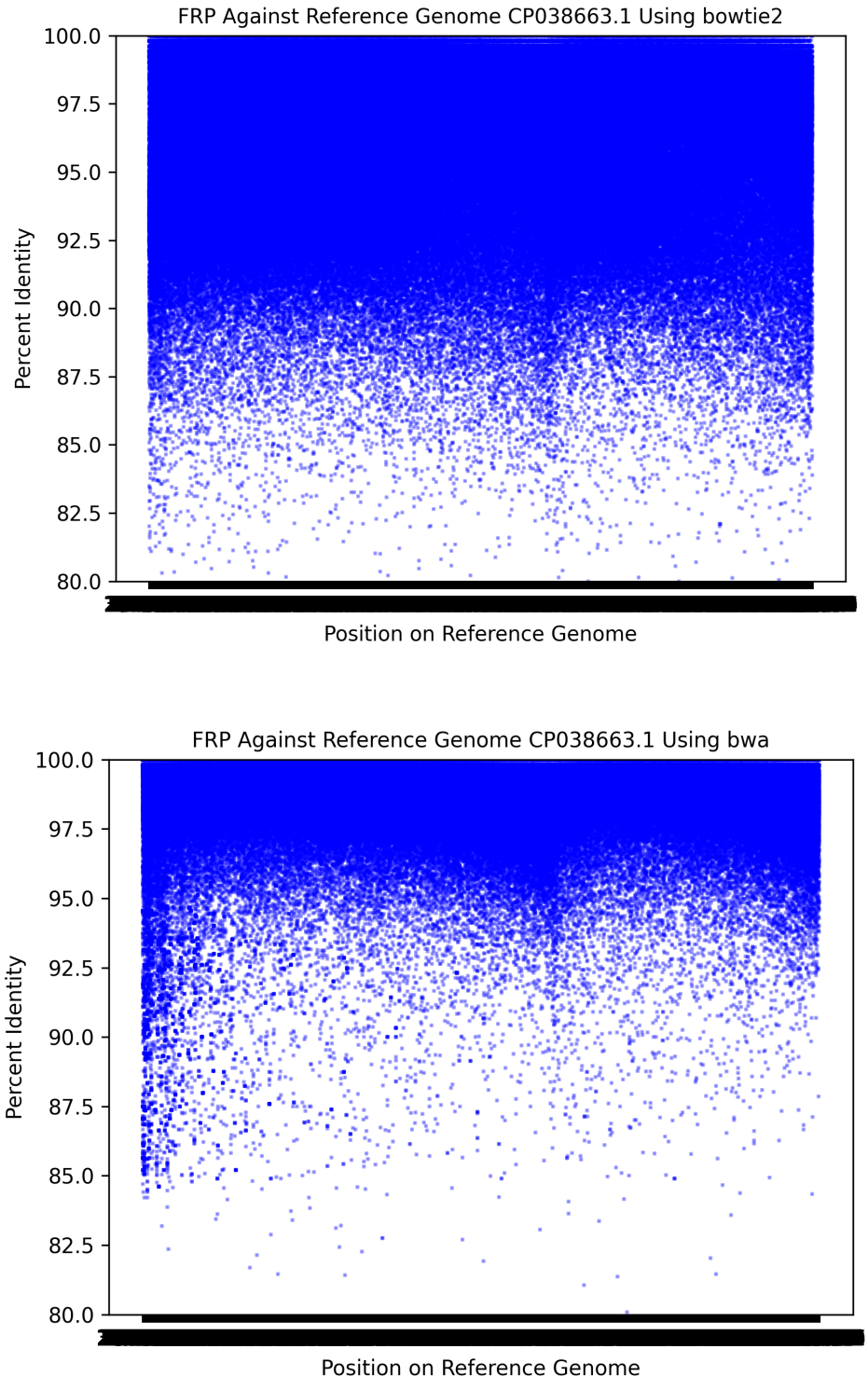

Figure 3: FRPs belonging to TaxonID 243230 using BowTie2 (above) and BWA (below)

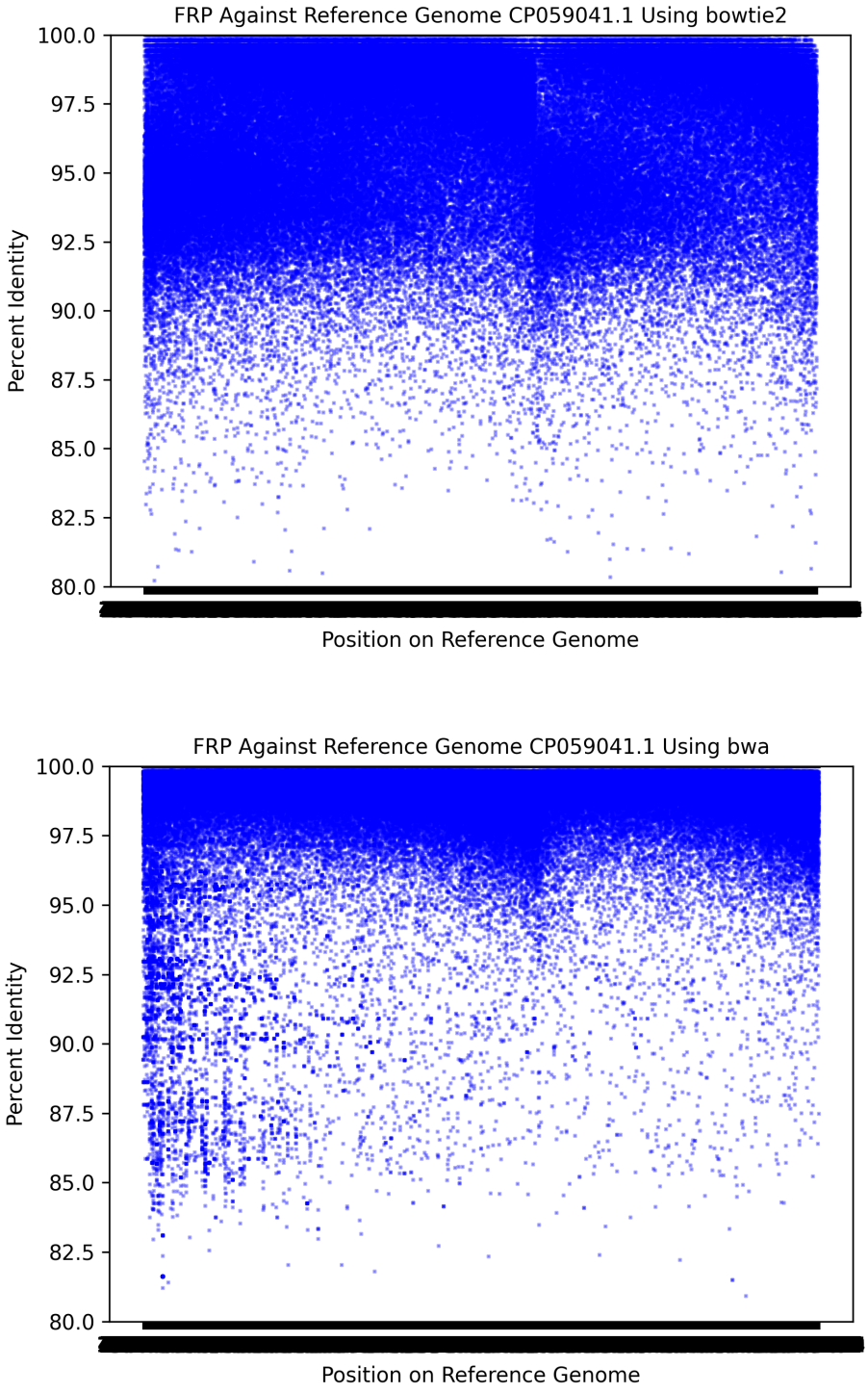

Figure 4: FRPs belonging to TaxonID 470 using BowTie2 (above) and BWA (below)

What did I learn?

From this midterm project, I learned how to use various new tools and packages including Matplotlib, shell scripting, installing dependencies, Google Cloud platform, Snakemake, BowTie2, and BWA mem. Running each tool was not difficult but learning how to use them was. I learned how to use each tool through official documentation, literature, YouTube, and many Google searches.

Moreover, I learned how to manipulate reference genome FASTA files and align metagenomic FASTQ next-gen sequencing reads to them. Through BowTie2, BWA-Mem, and Matplotlib, I was able to extract valuable information from the metagenomic sample. For example, the information determined which reference genomes were more or less represented by my sample. Moreover, the alignment rate was determined by the BowTie2 algorithm.

Though, if I had more time, I would write another Python script to compare the amount of reads from BWA-mem's output SAM file. That way, I can compare the actual number of reads versus the outputted recruited reads from BowTie2. Moreover, another challenge I faced was plotting the FRP via Matplotlib. Data visualization is new to me and I had issues trying to figure out how to label the x-axis with the position numbers. By default, MatPlotLib zoomed in on the most dense cluster of data but is not representative of the entire reference genome. As a result, a huge obstacle was finding a viable, standardized strategy for labeling the position numbers of the x axis of the graph for each reference genome.

Works cited:

[1] http://downloads.hmpdacc.org/data/HMMC/HMPRP_sT1-Mock.pdf

[2] https://www.dropbox.com/sh/a4dwo0q27qr6fig/AAAeYWc0k8BP9Ay5UAhhGtH5a/hmp_mock_community_project/samples/hmp_mock_454_even?dl=0&preview=SRR072233.fastq&subfolder_nav_tracking=1

[3] https://docs.anaconda.com/free/anaconda/install/linux/

[4] https://bioconda.github.io/

[5] http://bioconda.github.io/recipes/bowtie2/README.html

[6] https://snakemake.readthedocs.io/en/stable/getting_started/installation.html

[7] https://github.com/lh3/bwa

[8] https://www.htslib.org/download/

[9] https://en.wikipedia.org/wiki/SAM_(file_format)

[10] Johannes, K., & Sven, R. (2012, August 20). Snakemake—a scalable bioinformatics workflow engine. OUP Academic. https://academic.oup.com/bioinformatics/article/28/19/2520/290322

[11] Li H. (2013) Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997v2 [q-bio.GN].