Dhanan Shahdadpuri
IS 497
Final Project Document

PHASE 1

- <u>Dataset selection</u>

For the final project, I want to work with Illumina DNA sequencing data from CRISPR knockout screens. The raw data will be in fastq format and consist of several million different reads or spots, with each being around 20 base pairs or nucleotides in length. Below is a small subset of sequencing data in fastq format:

```
@SRR12455213.1.1 D00430:285:CBWVLANXX:1:1108:2150:2218 length=20
GCGCCACCGGATCCACCAGC
+SRR12455213.1.1 D00430:285:CBWVLANXX:1:1108:2150:2218 length=20
CCCBGGGGGGGGGGGGGFGG
@SRR12455213.2.1 D00430:285:CBWVLANXX:1:1108:2245:2227 length=20
CCACCAGTGACCAACACCCA
+SRR12455213.2.1 D00430:285:CBWVLANXX:1:1108:2245:2227 length=20
CCCBGGEG1DEDD0CF<>FG
@SRR12455213.3.1 D00430:285:CBWVLANXX:1:1108:2933:2162 length=20
AGAGGCAGCGGATAGCTCTG
+SRR12455213.3.1 D00430:285:CBWVLANXX:1:1108:2933:2162 length=20
@BCBGGGGCGGGGGGGGGGG
@SRR12455213.4.1 D00430:285:CBWVLANXX:1:1108:2933:2221 length=20
GCGGGAATACAAGATGCCTG
+SRR12455213.4.1 D00430:285:CBWVLANXX:1:1108:2933:2221 length=20
CCCCGBGGGGEGGGGGFGGG
```

I will describe the data in more detail in the “Dataset description” section. Each read corresponds to that of a gene’s guide RNA from the gene library used to perform the knockout screen. My goal is to align the raw data to the knockout library, count how many times each guide RNA showed up in the sample and then perform analysis on that read count file to calculate fold change, precision-recall data and Bayes factors. Ultimately, the goal would be to find out what genes are responsible for the phenotypic trait being studied. For this project, since I will only be carrying out the computational portion, I will only be focusing on the data processing and analysis. I will not actually be generating my own data and will instead use publicly available data from the NCBI website (https://www.ncbi.nlm.nih.gov/gds/?term=).

- Problem formulation

Currently, there are many methods to process Illumina sequencing data for CRISPR screening data. However, the steps are typically not easy to find and it seems like every researcher or project has their own approach. My goal is to simplify this so that the data processing can be completed easily by running a single script that contains all of the necessary commands. Additionally, I will include all documentation and references for the individual tools involved in the processing. Because I am currently working on this for my research project, I have already established and developed a working pipeline. This project will help me organize the project help other researchers learn and understand how to process raw sequencing data.

- Dataset description

The data will come from NCBI. I have not selected the exact dataset that I will use for this project but I know what I will be looking for. This could be a possible dataset:

https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE128210

Moffat, Kevin R Brown Barbara Mair Martin Soste Jason. "CRISPR screens are feasible in TP53 wild-type cells." *Mol Syst Biol* (2019).

The reason this data is so interesting is that it comes from a CRISPR screen. CRISPR/Cas9 is a genome editing tool that allows scientists and researchers to add, alter or remove parts of a DNA sequence within cells. This tool has tremendous potential and has already resulted in many experiments and paper publications. This particular paper wants to find out whether or not CRISPR screens are viable in human cells with intact p53 (a tumor suppressing protein) signaling. The reason is that another paper (referenced in the above linked paper) had concluded that the CRISPR/Cas9 would induce DNA damage that would activate the p53 pathway and lead to cell cycle arrest. The authors of the paper I obtained the data from were able compare their observations with the other author's (Haapaniemi et al, 2018). Overall, CRISPR screening data can be very useful to find what genes are essential to protein pathways. CRISPR/Cas9 has 2 elements that help it carry out its function of altering DNA. The enzyme cas9 cuts the DNA at a specific location so that DNA can be added or removed from that location. A guide RNA or gRNA is a piece of RNA which is around 20 base pairs long which is complementary to the target DNA sequence in the genome. Essentially, the guide RNA will show the Cas9 enzyme where to make the cuts in the DNA. When the DNA is damaged, the cell will try to repair itself and introduce a mutation or change of the DNA sequence. This page linked below explains the process in more detail:

https://www.yourgenome.org/facts/what-is-crispr-cas9

The tools I will use to process the data are bowtie and BAGEL. Bowtie is an aligner which will be used to map the raw reads from the sample to the gRNA sequences in the knockout library. The gRNAs I mentioned above are compiled into a library. The library will contain both the gene targeted by the gRNA as well as the gRNA sequence. The aligner will go through the millions of individual reads from the sample and try to match each read with its gRNA sequence. The output is a file in which the sequences are mapped to the gRNAs and respective genes. I will then use a python script to clean this data up, tally each read (to count the number of respective gRNAs in the sample) and sort it by gene name in alphabetical order and by read counts in descending order. This will be the read count file and will include data from multiple samples and a particular cell line. The experiment performs the knockout screen on various cell lines but to keep this simple, I will only be using a few samples from the same cell line. To analyze screen performance, BAGEL will be used. BAGEL is a statistical tool used to analyze knockout screens. It uses its own algorithm that normalizes the read count file, calculates fold changes, bayes factors and precision recall data. It does so by using training datasets of known essential “CEGv2.txt” and nonessential genes “NEGv1.txt”. If the genes are essential to the pathway, they should be enriched the most.

The documentation section below highlights the tools needed to carry out the steps mentioned above. Conda is a package manager, Bowtie is an alignment tool and BAGEL is an analysis tool. Additionally, will need to install Python to be able to run some of the scripts. Most of the commands will be run on the command line but simplified into a single shell script to facilitate the work.

Documentation:

-Bowtie: http://bowtie-bio.sourceforge.net/index.shtml

-BAGEL: https://sourceforge.net/p/bagel-for-knockout-screens/wiki/Home/

-Conda: https://docs.conda.io/en/latest/

-Python: https://www.python.org/downloads/

- Project organization

This is format is subject to change. I used this paper as a reference:
https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424

```
>master
        >README.md
                ~ file to explain the project and the commands needed to carry out the project
        >doc
                > scientific papers/references
        >data
                ~ all of the data files will be in here: raw data and files needed to carry out
                scripts/analysis
        >src
                ~ all of the code will be in here including a shell script to run all of the commands
                at once as well as the individual python scripts
        >results
                > tabulated results
                > plots
        >bin
                ~ binaries to download tools needed for processing/analysis
```

- Project repository

https://github.com/dshahd2/is497_final_project

PHASE 2

CONDA

This pipeline can only be run from a command line-like terminal whether that be on Mac/Linux or a command shell on Windows. For this tutorial, installation for Mac and Linux will be prioritized, however, the links include instructions to download the same tools on Windows. Conda is a package manager that will create a virtual environment and be used to install the required packages in that environment. To install conda, first download the binaries for the installer for your operating system from https://docs.conda.io/en/latest/miniconda.html. From this point, follow the instructions for your operating system from https://conda.io/projects/conda/en/latest/user-guide/install/index.html.

Once you have the installer downloaded, you may run the following command in the terminal window to install Miniconda (one of conda's two dependencies – Miniconda takes up less space than Anaconda).

For mac:

```
bash Miniconda3-latest-MacOSX-x86_64.sh
```

For linux:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

Once these are installed, follow the prompts on the screen, then close and reopen terminal and test the installation by running the command "conda list". This will show all of the installed packages that come with Miniconda and will prove a successful installation.

At this point, you will now be able to use conda to install the necessary tools for the data processing.

SRA TOOLKIT

The SRA toolkit is needed to download the raw data from the NCBI project repository. To install the SRA toolkit, download the binaries for your operating system from https://github.com/ncbi/sra-tools/wiki/02.-Installing-SRA-Toolkit.

For Linux/Ubuntu:

```
wget --output-document sratoolkit.tar.gz http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-
ubuntu64.tar.gz
```

For Mac:

```
curl --output sratoolkit.tar.gz http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-mac64.tar.gz
```

From here, extract the file contents:

```
tar -vxzf sratoolkit.tar.gz
```

For convenience (and to show you where the binaries are) append the path to the binaries to your PATH environment variable:

```
export PATH=$PATH:$PWD/sratoolkit.2.4.0-1.mac64/bin
```

Verify that the binaries are found by the shell:

```
which fastq-dump
```

The output should look something like this:

```
/Users/JoeUser/sratoolkit.2.4.0-1.mac64/bin/fastq-dump
```

Next, you will need to configure the tool. To do this, run the command:

```
vdb-config -i
```

You will see a screen where you operate the buttons by pressing the letter highlighted in red, or by pressing the tab-key until the wanted button is reached and then pressing the space- or the enter-key.

1. Enable the "Remote Access" option on the Main screen.
2. In the "Cache" tab, enable "local file-caching" and set the "Location of user-repository".

   a) The repository directory needs to be set to an empty folder. This is the folder where prefetch will deposit the files.
3. Go to cloud provider tab and accept to "report cloud instance identity".

The cloud instance identity only reports back in what cloud (AWS v GCP) you are working so you can access data for free.

Now that the toolkit is set up, you can test the tool by running the following command:

```
fastq-dump --stdout SRR390728 | head -n 8
```

The output should look like this:

```
@SRR390728.1 1 length=72
CATTCTTCACGTAGTTCTCGAGCCTTGGTTTTCAGCGATGGAGAATGACTTTGACAAGCTGAGAGAAGN
TNC
```

```
+SRR390728.1 1 length=72
;;;;;;;;;;;;;;;;;;;;;;;;;;;9;;665142;;;;;;;;;;;;;;;;;;;;;;;;;;;;;96&&&&(
@SRR390728.2 2 length=72
AAGTAGGTCTCGTCTGTGTTTTCTACGAGCTTGTGTTCCAGCTGACCCACTCCCTGGGTGGGGGGACTG
GGT
+SRR390728.2 2 length=72
;;;;;;;;;;;;;;;;;4;;;;3;393.1+4&&5&&;;;;;;;;;;;;;;;;;;;;;<9;<;;;;;464262
```

BOWTIE

Bowtie is responsible for the alignment portion of the data processing. To install Bowtie (https://anaconda.org/bioconda/bowtie), run the following command on either Mac or Linux:

```
conda install –c bioconda bowtie
```

To ensure it has been installed properly, run the command:

```
bowtie
```

This will show all of the parameters and functions for Bowtie.

PYTHON

Python will be installed as a part of miniconda, however, there are several packages or modules that are essential to the pipeline that must be downloaded. Here are the following commands to run for the installation of the different modules:

```
conda install pandas

conda install numpy

conda install pip

pip install click

pip install scipy

brew install pkg-config

pip install matplotlib
```

You will also need a text editor such as Atom to open and make changes to the python script that is included in the project folder. Python can also be run from the command line. Here are instructions how to do so https://docs.python.org/3/using/cmdline.html. The specific commands needed for this pipeline will be specified further in the document.

BAGEL

BAGEL will be used to analyze the intermediate data made from running the Python script (readcounts file) and used to produce visualizations. **The files needed to run BAGEL for this pipeline are included in the project file under "raw_data"**. However, to install the entirety of the BAGEL project, you can do so from https://github.com/hart-lab/bagel.

Now, we are ready to run through the entire pipeline. Before starting, it is highly encouraged to deposit all data (including raw data) into the same folder so that when the script "data_script.sh" is run, the commands will not give an error as all of the files will be in the same directory.

First, download the raw data. This raw data is from Brown et. Al (2019), https://pubmed.ncbi.nlm.nih.gov/31464370/. The raw data from this project was made publicly available and deposited on the NCBI website. From this page, https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE128210, you can download all of the projects supplementary files. These include readcounts files for the different libraries that the project used as well as the sgRNA libraries used. For this pipeline, the necessary data that we need will be included in the project folder under "supplementary_data". In summary, we will be using GSE128210_PACE002_RPE1-hTERT_WT-TKOv3_readcounts.txt as the read counts file to compare to. We will also use GSE128210_TKOv3-Human-Library.txt as the library for Bowtie alignments. To reformat the given library to one that we can use for the Bowtie step, we can run the following command:

```
awk '{print ">"$2"\n"$1}' GSE128210_TKOv3-Human-Library.txt > TKOv3.fasta
```

This reformatted library will also be provided under "raw_data".

The actual raw data (sequences) can be found on the same page as the supplementary data. For this pipeline, we will use the first two samples (RPE1_hTERT_Cas9_s1_T0, which we will rename to "T0.fastq" and RPE1_hTERT_Cas9_s1_T18A, "T18A.fastq"). To download these, click on the following links:

For T0, https://www.ncbi.nlm.nih.gov/sra?term=SRX5512138.

For T18A, https://www.ncbi.nlm.nih.gov/sra?term=SRX5512139.

On these pages, you will find the run information for both samples (SRR8718296 and SRR8718297).

The next step is to download the sequences using the SRA toolkit.

First, find the path to where you installed the SRA toolkit, specifically, the fastq-dump module. You can do so by doing:

```
which fastq-dump
```

Copy the output from this command. Next, cd to the folder you want to save the sequences to. Then run the following command, where you paste the output from "which fastq-dump", and then add "-I --split-files <accession number>". It should look something like this:

```
/Users/path_to/sratoolkit.2.10.8-mac64/bin/fastq-dump -I --split-files SRR8718296

/Users/path_to/sratoolkit.2.10.8-mac64/bin/fastq-dump -I --split-files SRR8718297
```

Due to the size of these files, it may take some time to download. Once downloaded, rename the files to "T0.fastq" for SRR8718296 and "T18A.fastq" for SRR8718297.

From here, most of the work is done. Simply run the shell script, "data_script.sh". This will build the library index, carry out the alignments, tally the readcounts and produce BAGEL calculated outputs. At this point, everything except for the visualizations will be produced.

VISUALIZATIONS

For visualizations, it is recommended to run the commands outside of the data_script. By doing so, you can alter the labels, titles, or add other data to the plots. For this particular pipeline, here is how to go about carrying out the visualizations. In the command line (and still in the same directory), run the following:

```
>python
>import pandas as pd
>import numpy as np
>import matplotlib.pyplot as plt
>from pylab import*
>pr_data = pd.read_table('hap1-t18.pr', index_col=0)
>pr_data.info()
>figure(figsize(4,4))
>plot( pr_data.Recall, pr_data.Precision, linewidth=2)
>xlim(0,1.01)
>ylim(0,1.02)
>xlabel('Recall')
```

```
>ylabel('Precision (1-FDR)')
>title('Precision-Recall Plot')
>show()
```

The “>” indicates a new line. This will produce the precision recall plot.

For the ideal Bayes factors distribution, run the following:

```
>figure(figsize(4,4))
>pr_data.hist('BF', bins=50, range=(-100,100))
>xlabel('Bayes Factor')
>ylabel('Number of Genes')
>show()
```

Testing:

To test results, you can compare readcounts.merged.txt to the one provided in the “results” folder. You can also compare the files generated by BAGEL (including visualizations) to the files in “results”.

Rationale behind approach:

Processing CRISPR sequencing data can be done in several ways. The most common is via the use of MAGeCK. This tool allows us to process the data (from alignment to readcount generation to screen analysis performance) with one singular command. While this seems great, it has its downsides. One of which is that the software does not account for mismatches in the alignment process. Because of this, if there are any misalignments between the raw sequencing data and the sgRNA sequences in the library, they will not be tallied or counted for the readcounts generation. This may lead to inaccurate analysis or data, depending on how many reads are not accounted for. Due to this, I have streamlined a pipeline that does account for mismatches. While a little more robust, my pipeline can generate more accurate results which will improve the quality of the screen when analyzed by the BAGEL software. The plots can help to visualize the screen performance.

The outputs produced by this pipeline can help to further understand what genes are responsible for specific protein pathways. By knowing essential genes, we can use (or not use) them to develop drugs or therapies to combat harmful pathways through the use of CRISPR gene editing. This has tremendous potential.

Citations:

Brown KR, Mair B, Soste M, Moffat J. CRISPR screens are feasible in TP53 wild-type cells. Mol Syst Biol. 2019 Aug;15(8):e8679. doi: 10.15252/msb.20188679. PMID: 31464370; PMCID: PMC6686785.

Hart, Traver & Moffat, Jason. (2016). BAGEL: A computational framework for identifying essential genes from pooled library screens. BMC Bioinformatics. 17. 10.1186/s12859-016-1015-8.

Hart T, Tong AHY, Chan K, et al. Evaluation and Design of Genome-Wide CRISPR/SpCas9 Knockout Screens. *G3 (Bethesda)*. 2017;7(8):2719-2727. Published 2017 Aug 7. doi:10.1534/g3.117.041277

Ran, F., Hsu, P., Wright, J. *et al.* Genome engineering using the CRISPR-Cas9 system. *Nat Protoc* **8,** 2281–2308 (2013). https://doi.org/10.1038/nprot.2013.143

Yau EH, Rana TM. Next-Generation Sequencing of Genome-Wide CRISPR Screens. Methods Mol Biol. (2018);1712:203-216. doi: 10.1007/978-1-4939-7514-3_13. PMID: 29224076; PMCID: PMC6089254.