TRACE MICROBIAL SEQUENCES NGS INSTRUMENTS

Akeem Wells
School of Data Science
University of Virginia
Charlottesville, VA
ajw3rg@virginia.edu

Matthew Sachs
School of Data Science
University of Virginia
Charlottesville, VA
mds9b@virginia.edu

Aubrey Brockmiller
School of Data Science
University of Virginia
Charlottesville, VA
alb3cb@virginia.edu

ABSTRACT

"Next Generation Sequencing (NGS) generates millions of reads per library prepared. Of these sequences, between ten to thirty percent of the total reads are not mapped to a reference genome. These unmapped sequences could be a result of differences between the reference genome and sequence reads, low quality sequences or an introduction of exogenous nucleic acids. Identifying a reliable method to identify trace microbial contaminating sequences and the instruments generating the sequences could distinguish between the introduction of exogenous nucleic acids from laboratory practices or true microbial reads. Investigating unmapped reads within next generation sequencing data will provide additional information regarding the source of the trace microbial reads. It is of interest to the Sponsor to determine if unmapped sequences are true microbial reads or exogenous contamination and if there are clustering trends specific to the instruments generating the sequencing data."

1 Introduction

Our capstone project aims to see if there is a relationship between the metadata – location of the instrument, the instrument, or the processing technician – and the contamination present in the run. Sourcing contamination to a particular instrument or location can provide insight to reduce future contamination events or reduce the impact that contamination has on downstream analysis. If the association between the metadata and the sequence is sufficiently strong, the sequencing results could even be used to fill in missing metadata.

2 Background

2.1 Literature Review

Contamination in DNA sequencing data, despite the tireless efforts of laboratory staff to maintain a clean environment, is inevitable. Weyrich et al. performed a study of by collecting 144 negative and/or blank samples in various labs for a five year span. The sequenced samples, although should have been blank, were all contaminated with some form of bacteria or unknown contaminant[1]. Unknown contaminants pose a additional challenge - if a contaminant cannot be identified it is difficult to remove it from your sample. Jurasz, Pawłowski, and Perlejewski, take a deep dive into these contaminants and note "contamination cannot be totally avoided; in particular, the issue of reagent contamination should always be addressed with high priority." [2] This suggests current sequencing files are riddled with known and unknown contaminants. For our project, we aim to find a correlation between contaminant sequences and the metadata. We must find a method to include both the known and unknown contaminants.

"From Trash To Treasure: Detecting Unexpected Contamination In Unmapped NGS Sequences" proposes a contamination identification tool called DecontaMiner[3]. While the intricacies of their BLAST algorithm are detailed in another paper, they do spend time to explain their pipeline: 1) take FASTQ, BAM, or FASTA input files, 2) convert to FASTA, 3) use their BLAST algorithm to map to a consolidated database of bacteria, virus, and fungi genomes, 4) group samples into "Low quality", "Ambiguous", and "Valid", and 5) provide visualizations and scoring metrics. They then go on to evaluate the efficacy of their pipeline by comparing to 3 widely used alternative solutions. Unsurprisingly, DecontaMiner exceeds the benchmarks. Since their tool is publicly available, we tried using it ourselves and discovered that, while it may exceed the benchmarks, it is still inordinately slow due to data processing volumes. Ultimately we do not use this tool, but the process outlined by the group provides a framework for our own project.

An additional reading, "A Systematic Sequencing-Based Approach for Microbial Contaminant Detection and Functional Inference" highlights an ETL pipeline with an integrated computational approach aimed at identifying unmapped sequences[4]. The entire pipeline consists of 9 steps: 1) assess quality of input by removing adapters and trimming reads, 2) map to the reference genome, 3) remove reference-mapped reads, 4) mask low-complexity sequences in unmapped reads, 5) align masked sequences to a microbial genome of the same genus, 6) compute single and multi hit metrics for microbial mappings, 7) penalize multi-hit mappings using an exponential function, 8) calculate z-scores to assess chance occurrence of unique hits, and 9) calculate microbe-mapped reads per million reference genome-mapped reads. The reading was instrumental in expanding our project scope beyond ETL and into the realm of data inference.

By outlining previously successful pipelines, the readings instructed our own pipeline construction. However, our project is unique in its approach to downstream analysis. The provided resources attempted to identify the contaminants by mapping them to a contaminant reference genome whereas our project will leave the sequences in their nucleic form and cluster solely on the nucleotide order. The length of these sequences will be defined based on downstream optimization. This step required a fair amount of experimentation to optimize. One paper suggests a package, K-mer–length Iterative Selection for UNbiased Ecophylogenomics (KITSUNE), "for assessing the empirically optimal k-mer length of any given set of genomes of interest for phylogenomic analysis via a three-step approach based on (1) cumulative relative entropy (CRE), (2) average number of common features (ACF), and (3) observed common features (OCF)."[5] This package was not appropriate for our applications, so we were forced to optimize the k-mer length through other means.

2.2 Data Description

All sequencing data being utilized for this project is publicly available via national databases. This data is typically accessed from the National Center for Biotechnology Information (NCBI) via the Sequence Read Archive (SRA). The Python module Pysradb is specifically built for 1) calling metadata for relevant lines of inquiry and 2) using this metadata to download desired sequence data. The metadata call worked quite fluidly within the module, but the metadata being pulled did not provide the accurate download link. The help resources at NCBI were not helpful in resolving the discrepancy. However, we later discovered these files are also available via the European Nucleotide Archive (ENA). The files are a bit easier to access via the ENA. Our sponsor provided five sequencing projects (via an identifier known as "accession number") containing a total of 50 sequencing output files to be processed for this project. These sequencing files are available in the widely used FASTQ format. FASTQ files are text based files that contain both the sequencing output as well as quality scores associated with the run.

A FASTQ file normally uses four lines per sequence:

- Line 1 begins with a '@' character and is followed by a sequence identifier and an optional description (like a FASTA title line). We later found that the usual metadata contained in this line is often stripped out before being uploaded to the ENA database. Therefore, we are only able to group our project files by location the sequencing occured. We are making the assumption all runs were done on the same or similar instruments.
- Line 2 is the raw sequence letters.
- Line 3 begins with a '+' character and is optionally followed by the same sequence identifier (and any description) again.
- Line 4 encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence.

This pattern can be repeated hundreds of times depending on the amount of sequences targeted within the run. This results in a range of file sizes (400Mb - 2Gb).

For our project, one 'unit' can be defined as one project that was run on a single instrument. The metadata for the run is what gives each unit its uniqueness. The hundreds of thousands of sequences contained in the file are all collected data points for the run. Because of this, our full data set will have millions of data points.

After mapping occurs(discussed in methods), the output SAM file contains a vast amount of information about each sequence and if/how it was mapped to the reference genome. There are eleven mandatory fields in the SAM file. This file is exceptionally large; therefore, it is typically converted to a BAM file, which is the binary form of SAM. Our pipeline parses down the BAM file to a manageable size by obtaining the sequences that are unmapped to the human reference genome; notably, partially unmapped sequences are excluded from consideration. These fully unmapped sequences are the target for our analysis.

3 Methods

3.1 Packages and Programs

As previously discussed, our data will be accessed through an API hosted by the ENA database. Once acquired, Python has a host of bioinformatics packages that we will utilize to unpack and evaluate these files. Some of these packages and programs include:

Biopython – an open-source package built for biological sequencing data. This package is built to understand file types such as the FASTQ and SAM/BAM files.

- HiSTAT2 – an open-source program utilized to map sequencing runs to reference genomes. This program will ingest the FASTQ file and output a SAM file.
- SAMtools – this is a set of tools designed to manage SAM and BAM file. This includes reading, parsing, and converting SAM to BAM files.
- Pysam - a python module used to read and manipulate mapped short read sequence data stored in SAM/BAM files

3.2 Data Filtering and Splitting

Our goal was to ingest 50 FASTQ files from five projects. Some FASTQ files were not usable after being compared to the reference data via HISTAT2 because they had little to no contaminant sequences. After mapping to the reference genome, we only retained and used projects with more then 17K contaminant sequences.

After identifying the usable projects, the sequences were aggregated into one dataframe and split 80/20 for training and testing.

3.3 Pipeline

A visualization of the constructed pipeline can be seen below in Figure 1.

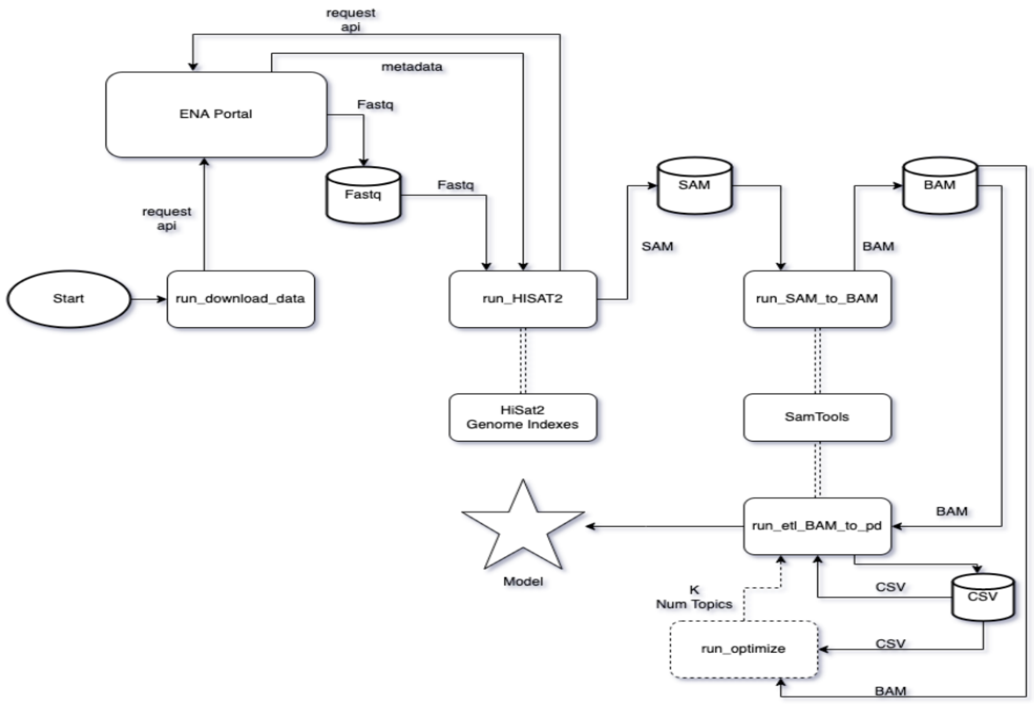

Figure 1: Graphical visualization of the overall project pipeline.

Beginning at the start circle on the left, our pipeline begins by downloading the raw FASTQ files. This can be done from your own instrument or local device, but for our purposes we follow the arrow upwards to the ENA Portal to access our FASTQ files. The ENA portal provides the raw files and associated metadata. These files are stored locally for later ingestion. However, it should be noted after the files are mapped in the new step, they are no longer needed and can be removed to save space.

Next following the arrows downward and to the right, the raw FASTQ files are ingested into HISTAT2. HISTAT2 is a program that compares the raw project files to the known reference genome indexes stored in HISTAT2. The program compares each sequence line in the FASTQ file to the reference. It returns a file known as a SAM file. This is a large file that contains a vast amount of information about the FASTQ file. The SAM file shows which sequence lines match a reference and which ones are unmapped. As a general rule, we would expect <30% of sequences in the original file to be unmapped.

In the pipeline, the rectangles indicating the SamTools are utilized to convert the SAM file to a BAM file. The BAM file is a binary form of the SAM file. Since the SAM file is so cumbersome, this allows easier storage and loading of the file.

Moving downward in the graphic, the BAM file is fed into a function to convert the BAM file to a pandas dataframe. This dataframe only contains the unmapped sequences lines from the original FASTQ file.

Once the unmapped lines are identified, we must construct a k-mer representation of the unmapped reads. The k-mer algorithm incrementally steps through the sequence, extracting tokens of k, until a token of length k can no longer be produced. See Figure 2 for a visualization of this process. These k-mers are saved in a dataframe to be fed into the modeling.

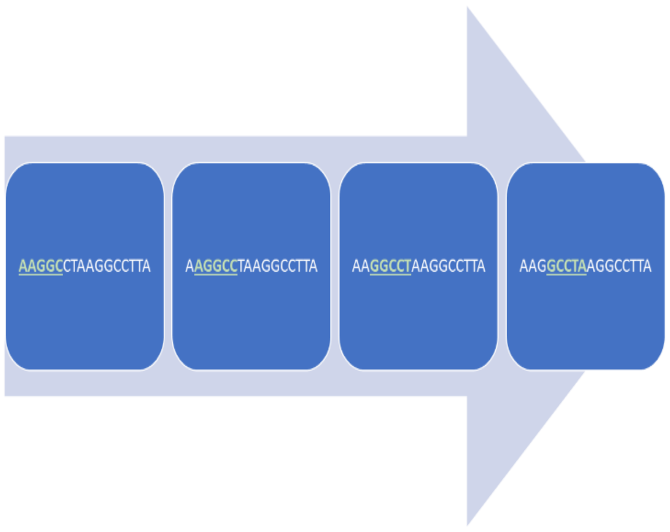

Figure 2: Visual representation of the k-mer algorithm constructing the k-mer sequences on a sliding scale.

3.4 K-mer Optimization and Filtering

An optimization of k-mer needed to be established. With a k too large, the k-mer sequences would become too specific and unable to occur more than once in the dataframe. With a k-mer too small, the sequences would not be specific enough, and would not provide a way to cluster contaminants to the centers. Documentation suggested a k-mer length of 10 to 20 basepairs is long enough to indicate contamination such as bacterial, viral, or fungal. We chose to go to the middle of these values and set k = 15.

After running the k-mer algorithm (Figure 2), a similarity matrix was conducted to visualize the raw similarities of the projects. Figures 3 and 4 show the Jaccard and Cosine similarities, respectively.

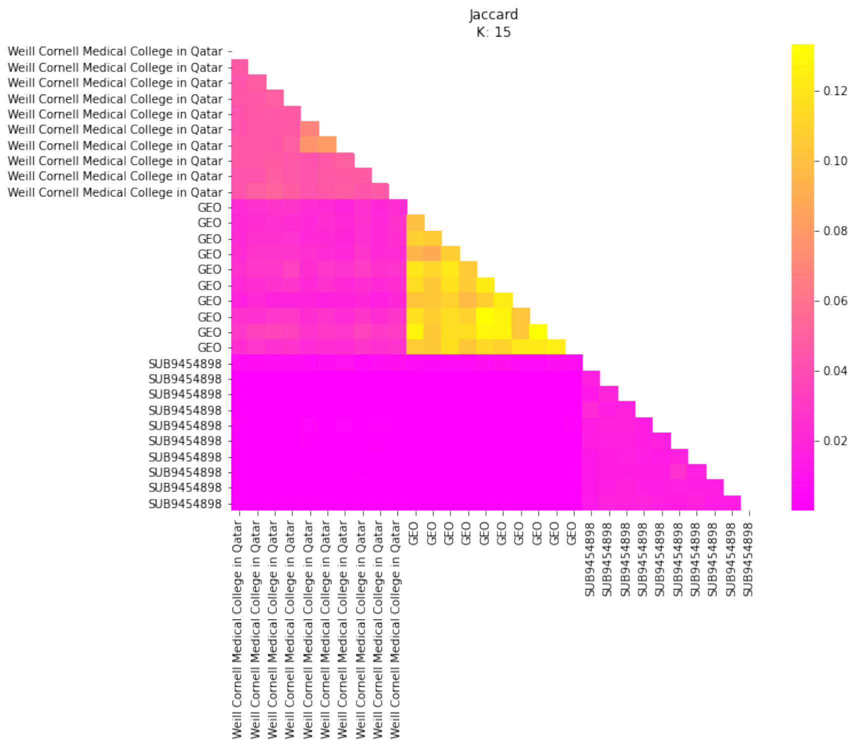

Figure 3: Jaccard Similarity Matrix

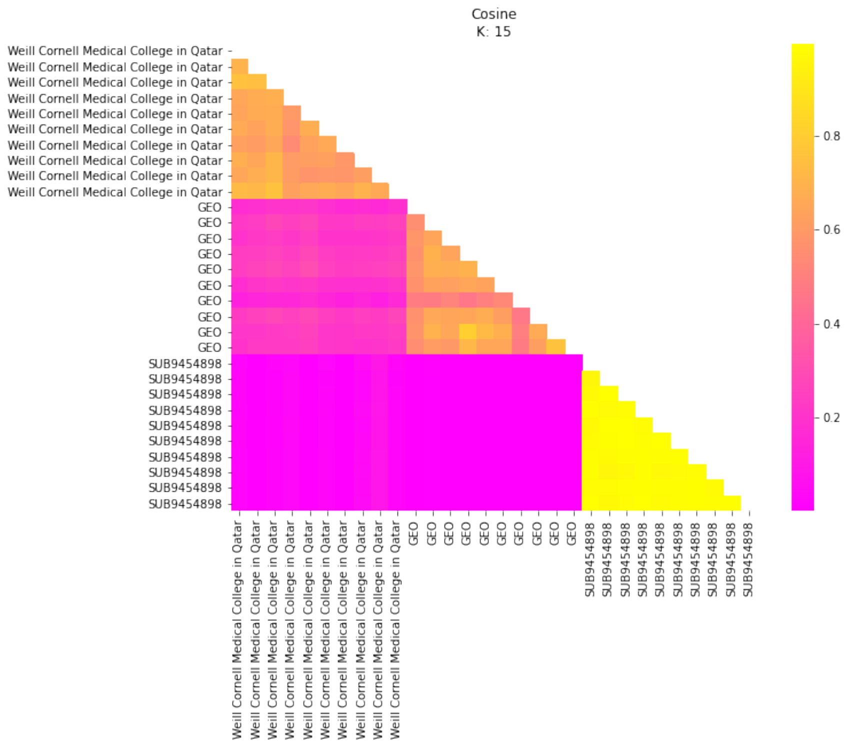

Figure 4: Cosine Similarity Matrix

Some k-mers will be distinct for certain centers, and help with clustering, some k-mers may occur in equal ratios across all projects. To reduce the amount of unnecessary k-mer sequences, we calculated a chi-squared test between the k-mer counts in each project. If the resulting chi-squared was statistically significant between any of the projects, that k-mer saved for further topic modeling. If the k-mer was not statistically significant, it was dropped. The remaining k-mers are saved as a working library and used to construct the corpus to be fed into the topic model.

3.5 Topic Modeling

Topic modeling is a type of statistical modeling for discovering groups that occur in a collection of documents. Latent Dirichlet Allocation (LDA) is an example of topic model and is used to classify text in a document to a particular topic. It builds a topic per document model and words per topic model, modeled as Dirichlet distributions.

For our application, each k-mer is treated as a word in a sequence file, or document. Our goal is to assign a topic to the k-mer groups that can then cluster back to the sequencing center. Once the k-mers are filtered for statistical signifcance, we utilized LDA to cluster the k-mers into topics.

4 Results (in progress)

4.1 Topic Modeling

The resulting topics can be visualized in Figure 5. Thus far, the topic model is constructed on a default size of 16. However, once additional data can be ingested we will work to optimize the number of topics for better results.

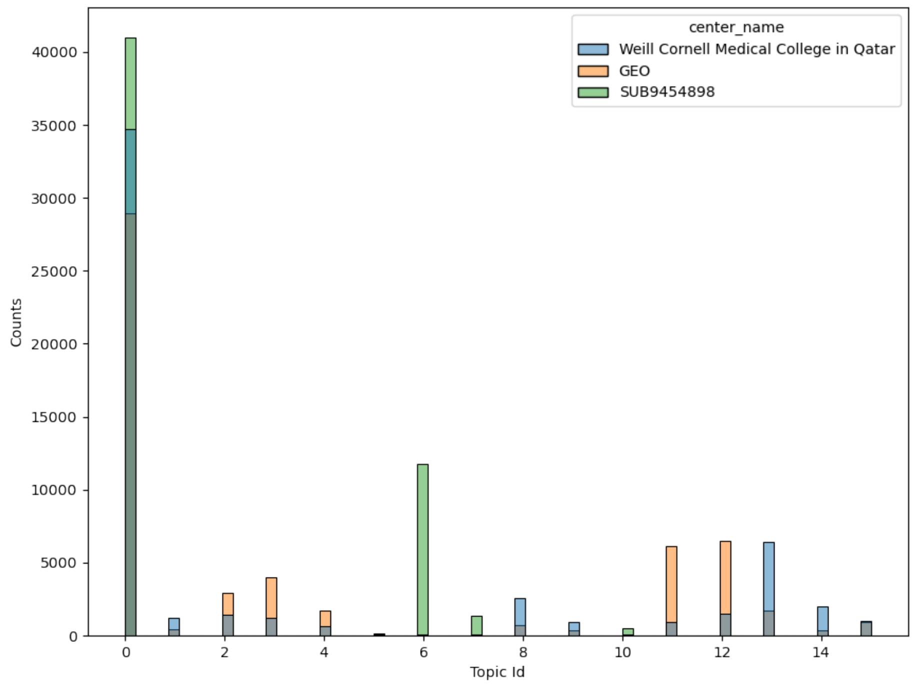

Figure 5: Visualizing topic models by color bands. Each color represents a different center.

Currently, the confusion matrix and accurancy is calculated by a count. Each k-mer is assigned a topic is is most likely to belong to. Each topic is assigned a center. The topic center is the center with the highest probability in that topic. We are considering adding in a weighting for topics with high probabilkties of distingushing centers. Again, this process will be optimized when we are able to process more files. The current confusion matrix and accuracy scores can be seen in Figure 6. Our current coherency score is -0.5739849413040531.

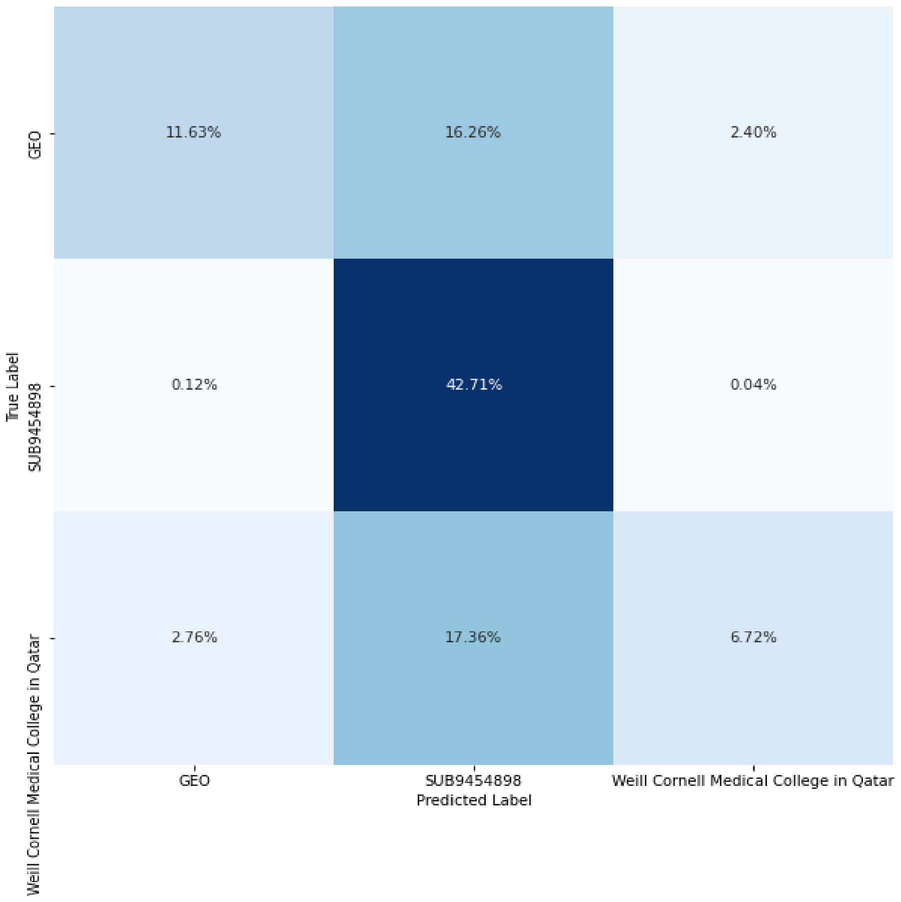

Figure 6: Current confusion matrix

5 Preliminary Discussion and Future Work

Thus far, we have built a pipeline that includes the following: importing the FASTQ files, mapping the files to the reference genome using HiSTAT2, converting the output SAM file to a BAM file, reading the BAM file, converted the BAM output to k-mers, filtered the k-mers for statistical signficance, and applied topic modeling for clustering. The pipeline building process has been informed by our twice weekly meeting cadence, our continued research of relevant open-source bioinformatics software and packages, and expanding domain knowledge in the field of. Due to the large size of the data files, we have set up processing support utilizing the Rivanna system. Through the Rivanna system, we have been able to read in multiple projects via an API. The pipeline is built to read in as many files as user defined.

For the remainder of the semester, we will continue to ingest FASTQ files to reach our file goal of 50 FASTQ files. As needed, we will continue to fine tune the parameters to improve our topic modeling and clustering as well as explore better classification based on the topic assignments.

References

[1] Laura Weyrich et al. Laboratory contamination over time during low-biomass sample analysis. In *Molecular Ecology Resources*, 2019.

[2] Henryk Jurasz et al. Contamination issue in viral metagenomics: Problems, solutions, and clinical perspectives. In *frontiers in Microbiology*, 2021.

[3] Mara Sangiovanni et al. From trash to treasure: Detecting unexpected contamination in unmapped ngs data. In *BMC Bioinformatics*, 2019.

[4] Sung-Joon Park et al. A systematic sequencing-based approach for microbial contaminant detection and functional inference. In *BMC Biology*, 2022.

[5] N Pornputtapong et al. Kitsune: A tool for identifying empirically optimal k-mer length for alignment-free phylogenomic analysis. In *Frontiers in bioengineering and biotechnology*, 2020.