Tools for Evaluating Quality of Sequence Read Archive Accessions in NCBI Database

Maxwell Fleming*, Udit Sharma†
* Johns Hopkins University mjfleming99@gmail.com
† Johns Hopkins Universiy ushama@gmail.com
Department of Computer Science, Johns Hopkins Univerisity
Baltimore, Maryland

***Abstract*—Currently many researchers and professionals who work in Bioinfomatics rely on data from the National Center for Biotechnology Information (NCBI). A common problem that people using the Sequence Read Archive (SRA) from NCBI face is low quality and mislabeled data. To help with this problem, we designed two tools that allow researchers to evaluate the quality of data from SRA and visualize the results of alignments. This paper will cover the development of these two tools: an alignment visualizer and an assay classifier.**

***Index Terms*—SRA, NCBI, Bowtie2, Data Visualization, Machine Learning.**

I. Introduction

The Sequence Read Archive makes sequence data publicly available which helps the research community by making experiments reproducible and allows for new discoveries by comparing data sets. The SRA databases are easy to contribute to and contain many of sequencing reads. However due to the ease of entry, many SRA accessions are mislabeled, or have other processing applied to them (such as bar coding) and this information is not correctly documented in the NCBI databases. This leads to a lack of public trust in the data, resulting in a under use of a valuable public resource.

To help motivate researchers to use the NCBI and reinforce public trust in the data, we created two analysis tools. The first tool is the data visualization tool, which provides graphs displaying various metrics about an SRA accession aligned with an index by Bowtie2. Our second tool is the assay classifier; which predicts the sequencing assay used to generate the reads in a SRA accession.

We believe that by providing quick and easy to use tools to researchers, we will increase the use of and trust in public NCBI data.

II. Prior Work

This section discusses the biotechnology resources we used, and how we used them.

A. National Center for Biotechnology Information

NCBI (National Center for Biotechnology Information) is a database founded in 1988 and is a subsidiary of the United States National Library of Medicine. NCBI provides us with the SRAs (Sequence Read Accessions) that we analyze.

B. Bowtie2

Bowtie2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. Bowtie2 was created in 2012 by Ben Langmead and Steven Salzberg at University of Maryland. We use Bowtie2 to align SRAs from NCBI.

C. SRAdb

SRAdb is a tool created by Jack Zhu and Sean Davis that makes accessing the metadata associated with the NCBI SRA database more feasible. We used SRAdb to generate the lists of accessions we used in our classification project.

D. BT2-UI

BT2-UI (Bowtie2 User Interface) is a user interface made by Rone Charles and Ben Langmead to educate people how to use Bowtie2 and is where we contributed our data visualization and classification tools.

E. NCLS

NCLS (Nested Containment List) is a Python library created by Alexander Alekseyenko and Christopher Lee that allows Python users to create a nested containment list. A nested containment list is a data structure for interval overlap queries. We use NCLS to determine what percent of the reads in a accession are expressed genes.

III. Methods and Software

A. Data Visualization

This section will discusses the work flow that we undertook when creating the ”Visualization” tab on the BT2-UI. We had to learn how to use Bowtie2, how to analyze a SAM (Sequence Alignment Map), how to create useful plots our of the data we generated from the SAM, how to render these plots on a server, and how to use Bowtie2’s input options.

1) SAM Parser: The first thing we handled was parsing the SAM file, the output of Bowtie2. We started by handling simple Bowtie2 stderr output and plotting alignment frequencies. Then, we learned the SAM format and created our own SAM parser. Initially we tried to parse the entire SAM file at once. Later we realized that this was inefficient with both time and space, and we transitioned to a parser that reads each line individually.

2) Plot Generation: After successfully parsing SAM files and Bowtie2 output, the next step was to create useful graphs from the data. We started using Matplotlib to plot our extracted data and we figured out useful metrics to plot. Then we transitioned to the Plotly library for generating interactive JavaScript graphs. Our final implementation uses Plotly's R library.

3) UI Page: We created a separate tab for the "Visualization" feature. This tab lets a user select a Bowtie2 index, input an SRA accession and pick Bowtie2 run options. It outputs various graphs displaying information about the accession, and allows the user download both the SAM file and the data we generated about the accession.

4) Bowtie2 options: We implemented several of the running options available to Bowtie2. We allow the user to choose between Phred 33, Phred 64, and Integer Quality values. We also allow the user to trim base pairs from the $5'$ and $3'$ ends which removes barcoding. We also allow the user to pick paired-end alignment options. This includes limiting the minimum and maximum fragment length, suppressing unpaired and discordant alignments, and making mates not concordant when the alignments contain each other or overlap.

B. Classifier

For the classifier, our goal was to be able to distinguish between three different assay types: Whole Genome Sequencing (WGS), small RNA-Seq (sRNA) and mRNA-Seq (mRNA) given the Bowtie2 output for an accession.

1) Gathering Accessions: The first step towards implementing a classifier was to gather many SRA accessions. Instead of gathering SRA accessions by hand from the archive, we decided to search for a database that we could query. We experimented with Sean Davis' Big Query Table, but it did not have all the functionality we required. Instead, we used Jack Zhu and Sean Davis' earlier implementation, SRAdb. We learned how to use RSQLite and gathered 10,000 random accessions for each of WGS, sRNA and mRNA. However, most of these accessions ended up being protected, so we could not access them. In the end, we gathered 1,672 WGS, 5,720 sRNA and 3,420 mRNA accessions which we could actually analyze.

2) Data Generator: For each accession, we needed an efficient way to run Bowtie2 and generate the SAM file. There were a few concerns. First, we wanted to get reads from random places in the accession so that the data was representative, and not influenced by any sorting the initial publishers may have used. Second, we wanted to use threading to speed up the process. Finally, we wanted enough reads that we had some information on the accession, but not so many that running Bowtie2 was slow. After some experimentation, we settled on 4 threads, each processing 2500 reads in random spots over the accession.

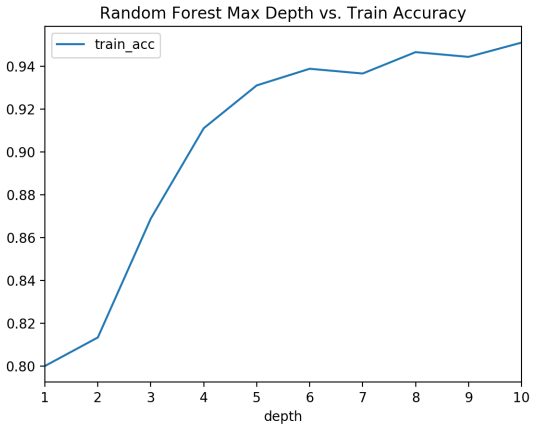
3) Parsing SAM file: Originally, we relied on our naive implementation of SAM parsing from the visualizer. But we soon realized this was much too slow and used far too much memory. We switched from attempting to process the whole file at once to parsing line by line, cutting the run time by 10 seconds for each accession.

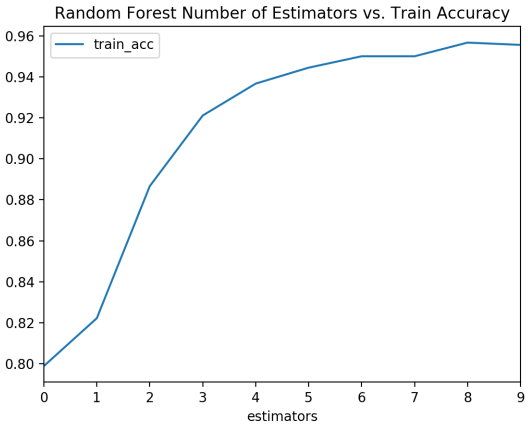
4) Gene Annotation Metric: Gene Annotation Percent, the percent of reads in a sequence that correspond to genes, ended up being our most difficult feature to generate, yet also most informative. We created a gene annotation tree using a GTF (gene transfer format) file which contained the expressed genes in the human genome, and the NCLS data structure discusses earlier. Using these two tools, we could determine if a read corresponds to a gene.

5) UI Page: We created a separate tab for the "Classification" feature. This tab lets a user input a SRA accession, and we process the accession and run it thought our machine learning model and display our prediction. In addition to this, we provide the user with interactive plots that allow for users to compare their accession to the 10,812 accessions that we have pre-processed.

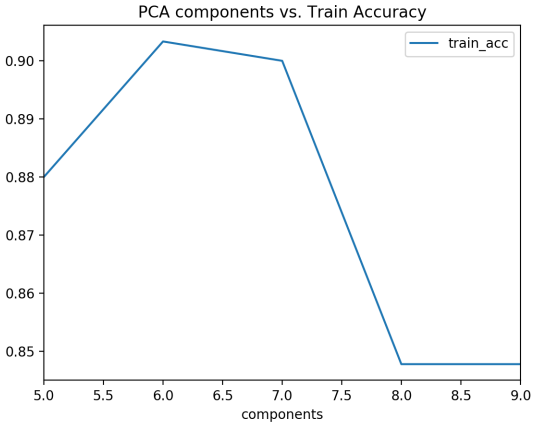
6) Models: Of the 1,672 WGS, 5,720 sRNA and 3,420 mRNA accessions, we randomly selected 1,500 accessions of each for training and testing, so our model would train on an even distribution of all three assay types. We split this data into 80% train and 20% test. We used 10 fold cross validation for training instead of creating another development split. We implemented two models, Random Forest and K nearest neighbors.

1) Random Forest. For the random forest, we tuned two hyper parameters: max depth and number of learners. First, we tracked cross validation accuracy for 10, 20, 40, 50, 60, 70, 80, 90 and 100 learners while keeping max depth held at 2. Then, we set number of learners to 80 and tracked cross validation accuracy for a max depth of 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10. The results are plotted below. We chose a random forest with 80 learners and a max depth of 3, since both these values come right before the elbow in the graphs plotted before, meaning they most likely are not overfit and will not change performance too much when tested on real data. Our final random forest with 80 learners and a max depth of 3 got a test accuracy of 91%.





2) K nearest neighbors. We experimented with naive KNN as well as KNN after principal component analysis. Naive KNN got a test accuracy of 89%. For PCA (principal component analysis) we experimented with 5, 6, 7, 8, 9 and 10 components, with 6 performing the best, and achieving a test accuracy of 90%.



IV. Results

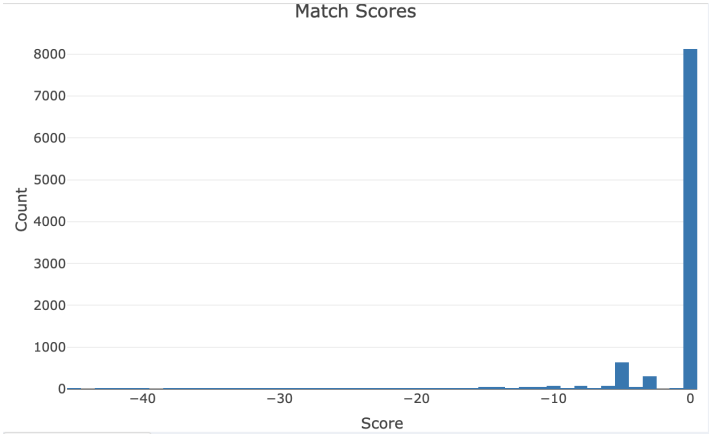
A. Data Visualization

Our Visual Analysis tool allows users to quickly visualize information about a randomly sampled SRA. The random sampling will ensure that any data ordering that the initial publishers may have used is not a factor in our analysis. We can successfully build interactive graphs that allows users to analyze the quality of their accession.

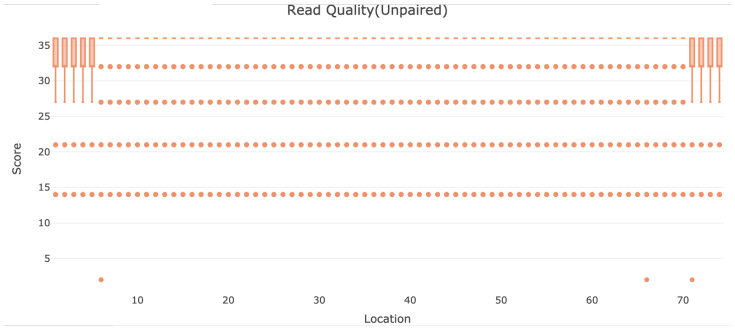
As discussed in the software and methods section, we allow for many of the runtime options that Bowtie2 supports in our UI.

We create six plots that help inspect the data. We have included examples of all the plots. These plots were made with the accession SRS2471517 using 10,000 reads and the GRCh38 reference genome.

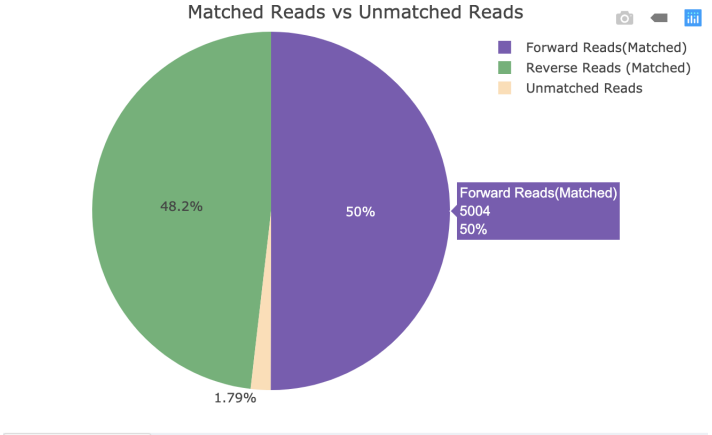
The first plot we provide is a histogram displaying the match scores. The match score is an alignment score that quantifies how similar the read sequences are to the reference sequence they aligned to. The higher the scores, the more similar they are.



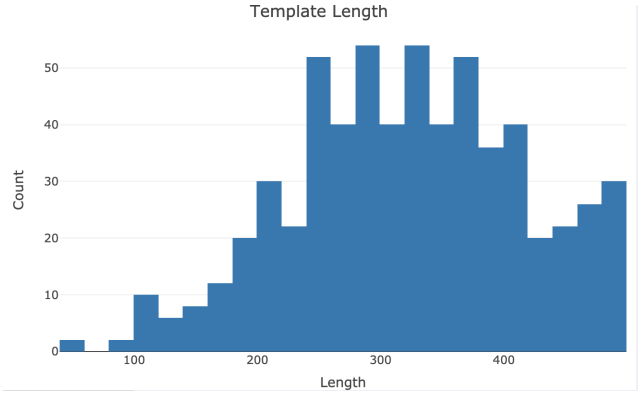
The second plot we provide is a read quality box and whisker plot. This plot is displaying the Phred quality score of all the bases at each index for the unpaired reads. We also provide the same plot for paired forward reads and paired reverse reads.



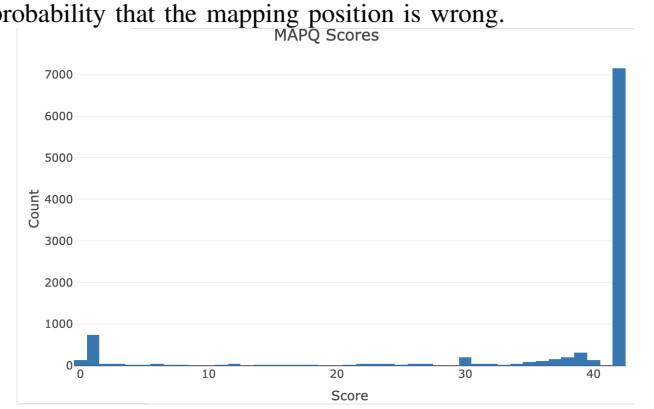
The third plot we provide is a matched vs. unmatched pie chart which represents the proportion of reads that were matched forwards, reversed or not at all.



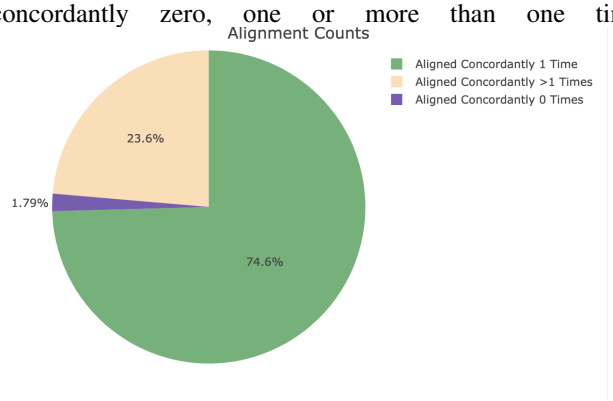
The fourth plot we provide is a histogram of the template lengths. This plot was run with the accession SRR10474462 instead of SRS2471517. The template length histogram represents the signed observed template lengths of all the reads.



The fifth plot we provide is a histogram of the MAPQ scores. The MAPQ score represents the Phred-scaled posterior probability that the mapping position is wrong.



The sixth plot we provide is a pie chart representing the proportion of reads that matched concordantly zero, one or more than one times.



1) Limitations: We struggled to get the rendering times lower. Some plots, mainly the read quality box plot, are representing thousands of data points in an interactive manner. It seems that Shiny and Plotly have a difficult time rendering these plots quickly. We tried several different techniques to render these quicker and have the interaction with them be smoother, however it still had lag that made interacting with larger quantity of reads difficult.

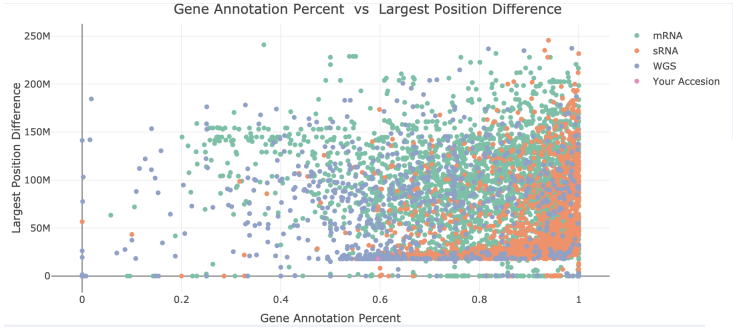
2) Comparison: In order to gauge the quality of our data visualizer, we compared our project to Iobio.io, a company focused on providing visualisation of genomic data, specifically their "BAM" tool. We provide similar plots such as forward/reverse stand matching pie chart, read lengths and mapping quality. However, we both have our unique plots. We provide a alignment summary and a alignment score plots, while they provide coverage distribution. We both can add more reads to our plots, however add Iobio.io can add reads without having to re-render. However, we provide options that they do not. We allow for the user to choose their input options, and we allow for users to enter the SRA accession while Iobio.io users have to upload their accessions.

B. Classification

1) Features: Our model uses many features that were gathered from SAM files and the gene annotation tree. We will explain each of the features that we used in our model.

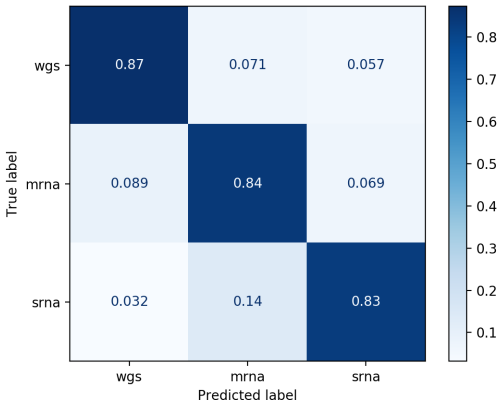
1) Gene Annotation Percent: This is the percent of reads in the SRA that matched to a expressed gene. This metric is very useful in separating WGS from mRNA and sRNA as WGS has a significantly lower gene expression rate than mRNA and sRNA.
2) Average Read Length: This feature measures how long each read is on average.
3) Read Frequency: The percent of reads from the accession that align.
4) Maximum Position Difference: Position difference measured how many bases were between where a read matched to the next matched read. The max position difference is the maximum of all the position differences calculated.
5) Minimum Position Difference: This is the minimum of all the position differences calculated.
6) Mean of Position Difference: After calculated all the position differences, we calculated the mean of the values. This acted as a measure of how tightly clustered the reads were.
7) Standard Deviation of Position Difference: After calculating all the positions differences, we calculated the standard deviation of position differences. Similarly to mean of position differences, this acted as a measure of how tightly clustered the reads were.
8) Number of Chromosomes: This measured the number of chromosomes that the reads matched to. This metric ended up being rather uninformative.
9) Percent of A, C, G and T: These features represent the distribution of the base pairs through all the reads we processed.

2) UI: In the user interface for classification we provide two metrics. The first metric is the result of the model, what we predicted to be the assay. The second metric is a visualization of how the accession the user provided fits into the data. We allow the user to plot against all of the features we used to classify the accession. The plot below is a plot of gene annotation percent vs. largest position difference. These are the features that our random forest determined we significant for classifying the assay.



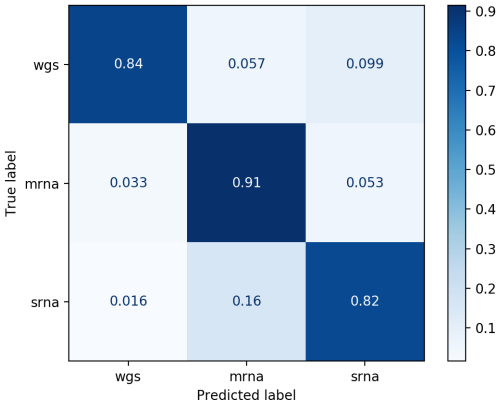
3) Models: We implemented two models, K nearest neighbors (KNN) with principal component analysis (PCA) and a random decision forest.

1) PCA KNN. Our PCA KNN model achieved a test accuracy of 90%, with the confusion matrix plotted below. For PCA, we reduced our 12 feature data down to 6 features. The high test accuracy supports the idea that assays are indeed separable by a model and that our features were well selected. As shown by the confusion matrix, the model does not seem to favor one assay type significantly more than any other.



2) Random Forest. Our Random Forest model achieved a test accuracy of 91%, with the confusion matrix plotted below. For hyper parameters, we settled on a max depth of 3 for each decision tree and 80 estimators. The Random Forest had the following feature importances (the higher the number, the more important the feature, important features are in bold):
Gene Annotation Percent: 0.18
Average Read Length: 0.07
Read Frequency: 0.24
Max Position Difference: 0.13
Min Position Difference: 0
Mean of Position Difference: 0.05
Standard Deviation of Position Difference: 0.14
Number of Chromosomes: 0.026
Percent A: 0.04
Percent C: 0.04
Percent G: 0.04
Percent T: 0.04

Again, the high test accuracy supports our hypothesis that the assays are separable. Like the KNN, the random forest does not seem to favor one assay more than any other, predicting each label about 80% of the time.



4) Limitations: There are a few limitations to our models. First of all, we can only classify the three assay types: WGS, sRNA and mRNA. Second, we can only classify these assay types for human reads. Third, our models were trained on SAM files with exactly 10,000 reads. Furthermore, although we achieve good test accuracy and the data looks separable by inspection, we only trained on 1,500 accessions of each assay type, which is not enough for us to confidently claim that our model is proficient.

V. Conclusion

A. What We Learned

1) Technical: Here is an abbreviated list of the most important technical knowledge we learned through this project:

1) R/Shiny
2) Plotly
3) Docker
4) Singularity
5) SQLite
6) Sci-kit learn
7) Mult-thread processes
8) Interval Trees
9) Feature design
10) Random Forests
11) Principal Component Analysis / K Nearest Neighbors
12) Git (pull requests, branching, history, resets, merging)
13) Technical Communication
14) Division of labor

This list could go on and on, but these are the things we decided had the biggest impact on us.

2) Non Technical: Little victories are important. The most important thing in this long term project was the little victories. During the time periods when we could consistently finish at least one small improvement, it seemed that anything was possible. A good example of this was throughout the summer and early fall semester. However, when we stopped making small iterations and tried to tackle too large a chunk, work

could stagnate for up to 4 weeks at a time. This happened when we were struggling to actually generate the SAM files for the accessions we had gathered. In the end, the solution was simple: use singularity and speed up parsing. However, because we tried to tackle too big a problem all at once it took much longer than it should have. We did not realize parsing was the major time consumer, we spent too much time iterating on Dockerfiles, and we tried to use Google cloud instances instead of the MARCC cluster. Of course, these roadblocks also happened because we of our inexperience with the relevant technologies, but taking the time upfront to be more explicit about the issues and breaking them down into smaller chunks would have sped up the process.

The best way to learn a new technology is to do something with it. At almost every turn of this project we were presented with a new technology, framework, technique or theory we needed to learn in order to make progress. New approaches are intimidating and it was hard to see the way forward when we felt so inexperienced, but at the end of the day when we sat down and experimented and struggled with new concepts we learned a lot. Online tutorials and articles were invaluable resources, but the only way to really learn a new technology was sitting down and actually working with it.

You are never done. As we made more progress, new features, ideas, techniques or limitations came to light. Even after working for the better part of eight months, it feels as though both the classifier and the visualization are somewhat incomplete. On the flip side, we also had to define or accept reasonable goals, otherwise we would never make any progress.

Nothing is easy, and the best things are the hardest. Even the smallest, most straightforward iterations of the project seemed to bring with them unforeseen complications. At the same time, the most fun and rewarding parts of the project were when we conquered some part that we had been struggling with for a long time.

Acknowledgments

We are extremely grateful to Professor Ben Langmead and Rone Charles for supporting, teaching and guiding us throughout this project. This has been an invaluable experience that would not be possible without their hard work and willingness to support us throughout the entire process. It was inspiring to be able to work with such dedicated professionals, and without a doubt this has been a formative experience of our college careers.

We would also like to thank all of those that created the tools that enabled us to do our project.

Source Code

All work done for the UI is available at github.com/usharma6/bt2-ui

All work done for the classifier is avalible at github.com/mfleming99/DNAAssayClassifier

Work Sited

Langmead B, Salzberg S. Fast gapped-read alignment with Bowtie 2. Nature Methods. 2012, 9:357-359.

Zhu Y, Stephens RM, Meltzer PS, Davis SR (2013). “SRAdb: query and use public next-generation sequencing data from within R.” BMC bioinformatics, 14(1), 19.

Alexander V. Alekseyenko, Christopher J. Lee; Nested Containment List (NCList): a new algorithm for accelerating interval query of genome alignment and interval databases, Bioinformatics, Volume 23, Issue 11, 1 June 2007, Pages 1386–1393, https://doi.org/10.1093/bioinformatics/btl647