Group Project #2: Mapping project
Team: Shun Sambongi, Megan McGhie, Lucas Pinto

Intro:

Next generation sequencing has proven revolutionary in bioinformatics, which allows for high accuracy aligning from small reads given the fact that there is a high coverage of reads. A high coverage contingency poses a burden when trying to compare genomes of two individuals. Also, most next-gen aligning algorithms of rely on high coverage to account for variants, but to build variant callers based on few reads and comparing against another genome requires a slightly different approach.

We take on this challenge here in computational biology here at BYU. A given set of unique reads we developed a tool which aligns this to the genome sequence given, and we also incorporate a robust mechanism which accounts for variants which may randomly occur in this data.

Method of mapping and handling errors:

First we take all reads and split them into kmers. Second we slide these kmers across every index in the reference. On each iteration of the sliding portion we increment the index value and the offset which allows us to calculate a support for each index while sliding. This support allows us to keep track of a location where each read may align with higher accuracy while allowing for variance in the sequence, whereas basic aligners operate as exact string matches. Once we've done all these iterations to the reference we select the index where the support is maximized.

In summary this is a mapping algorithm which *also* incorporates variance.

Analysis:

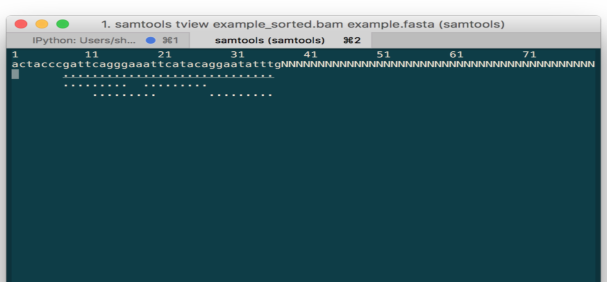
a. Our model

1. Example data set

I. The only parameter we use during mapping is differing kmer sizes.
II. We asses our mapping quality based on # of errors on the sam files in IGV
III. The reason these metrics validate the results is because with less errors, the better the alignment.

--additional deliverables:

A screenshot of the whole view using samtools:



2. RNA-seq

I. The only parameter we use during mapping is differing kmer sizes.
II. We asses our mapping quality based on # of errors on the sam files in IGV
III. The reason these metrics validate the results is because with less errors, the better the alignment.

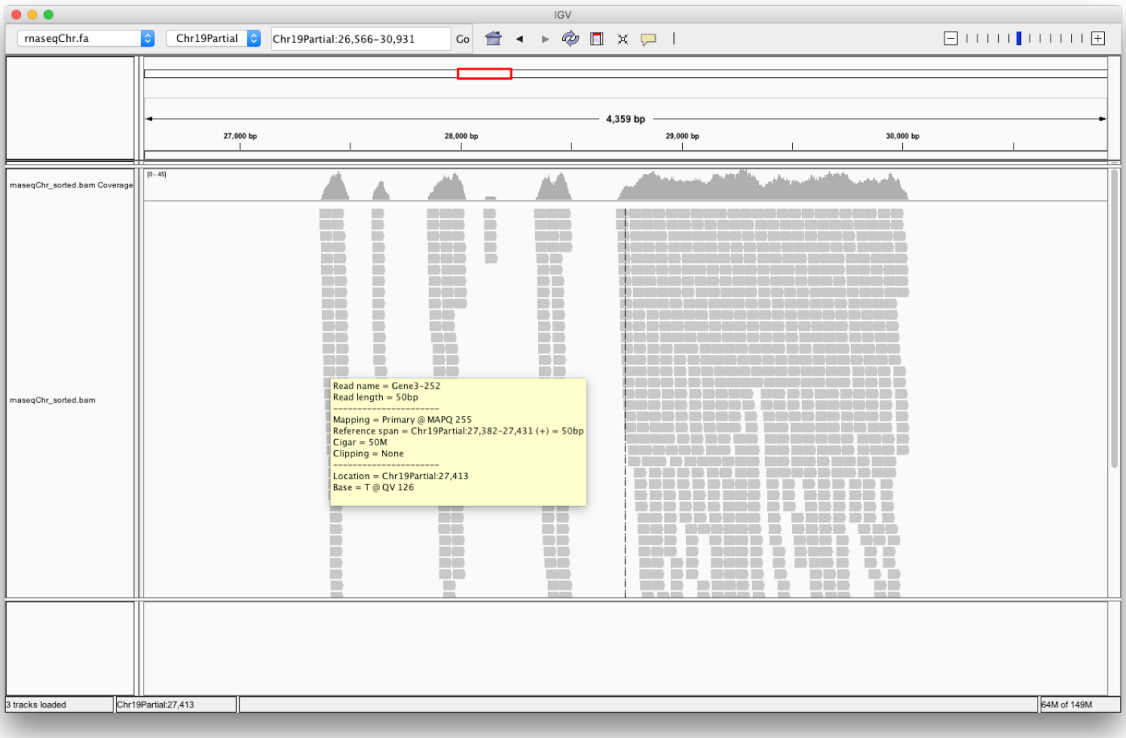
We found that kmer sizes of 70-80 gave us similar but the best results.

--additional deliverables:

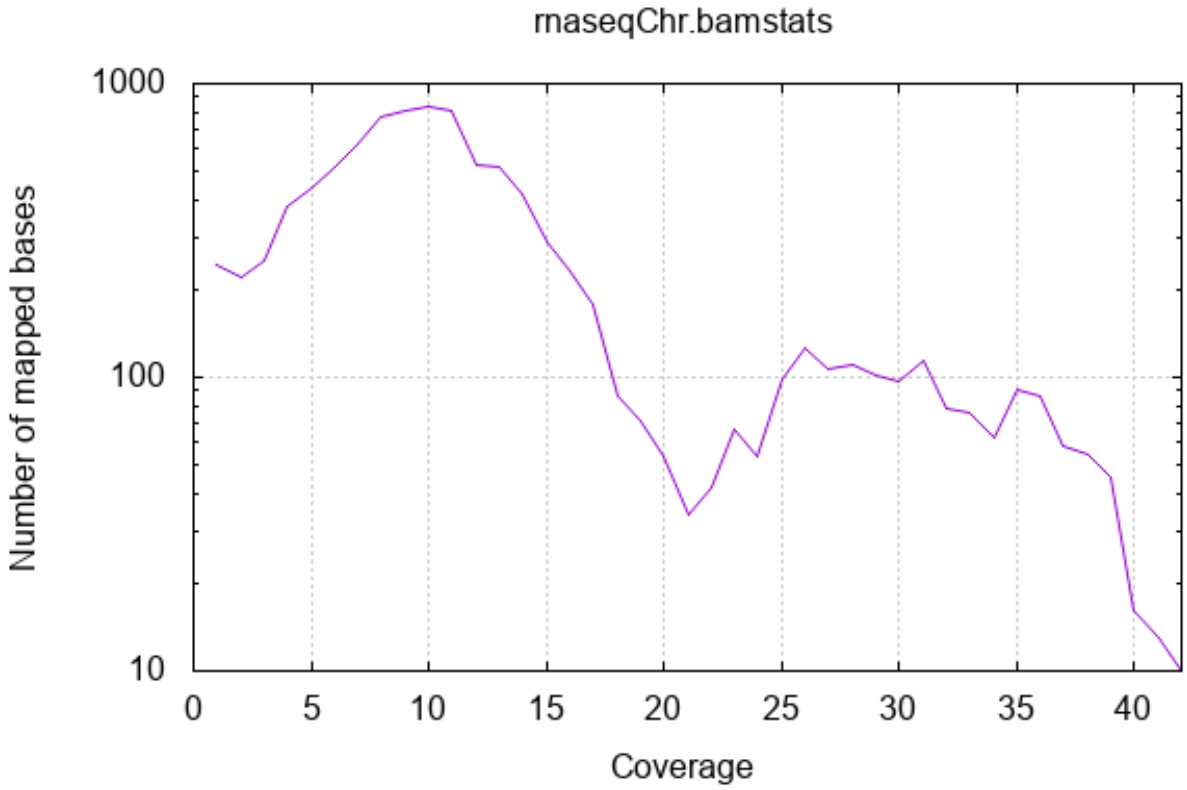
a. The region with highest depth or expression

27366 - 30027

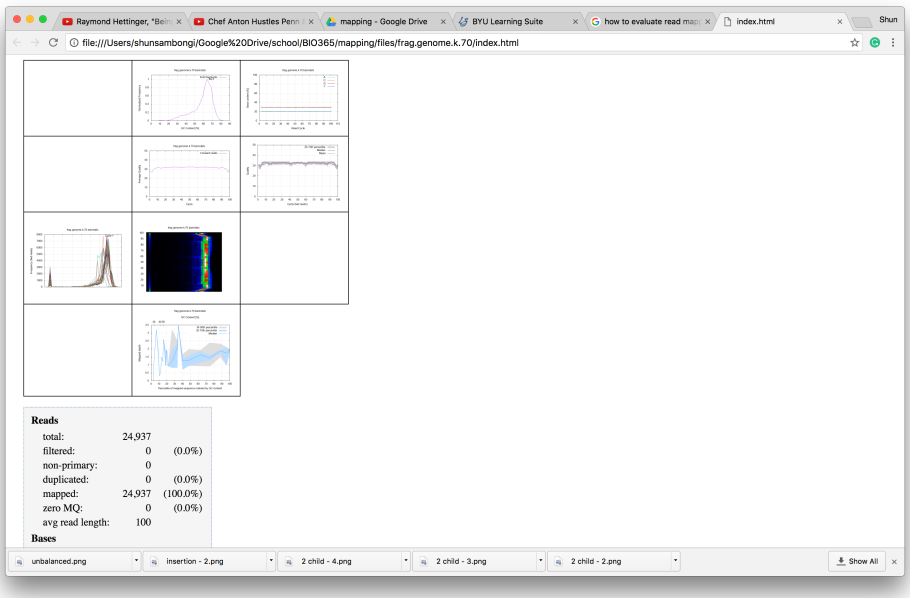
b. Screenshot of depth throughout the chromosome



coverage X mapped regions:



more analysis:



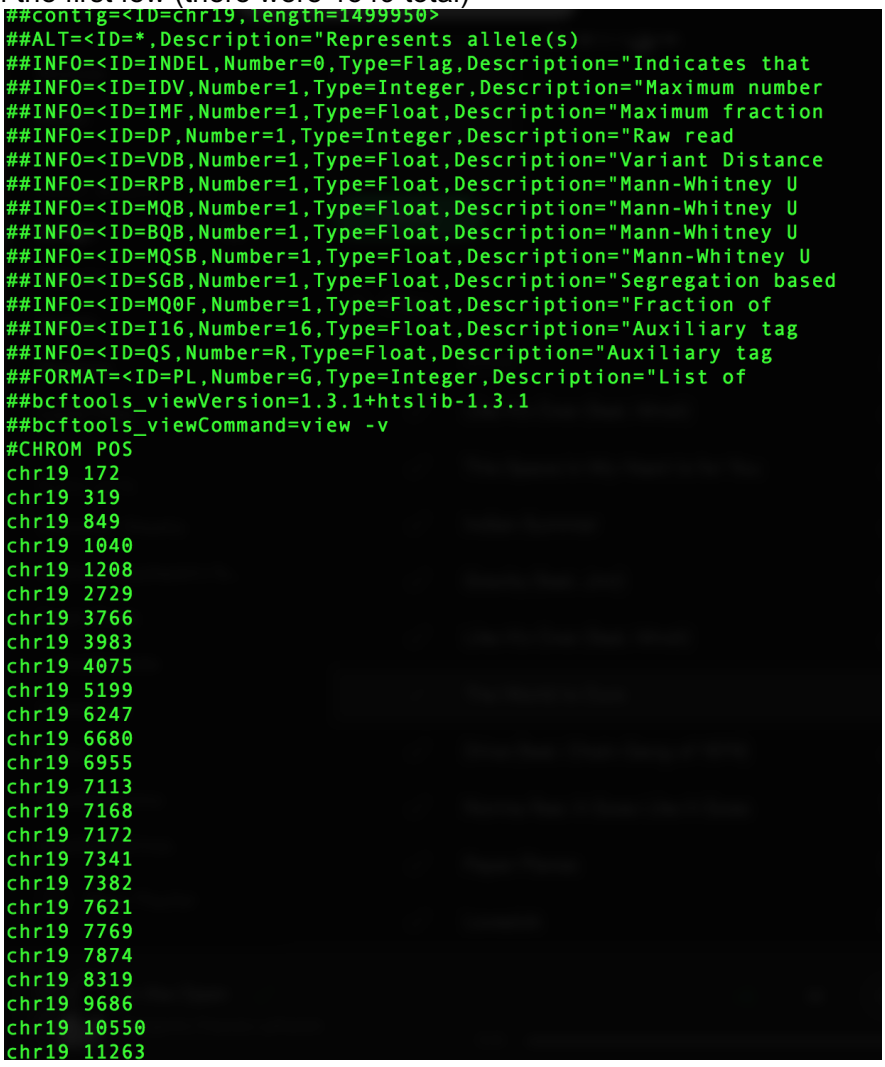
3. Shotgun data

I. Once again, the only parameter we use during mapping is differing kmer sizes.
Best kmer size: 70-80
II. We asses our mapping quality based on # of errors on the sam files in IGV
Number of errors:
III. The reason these metrics validate the results is because with less errors, the better the alignment.

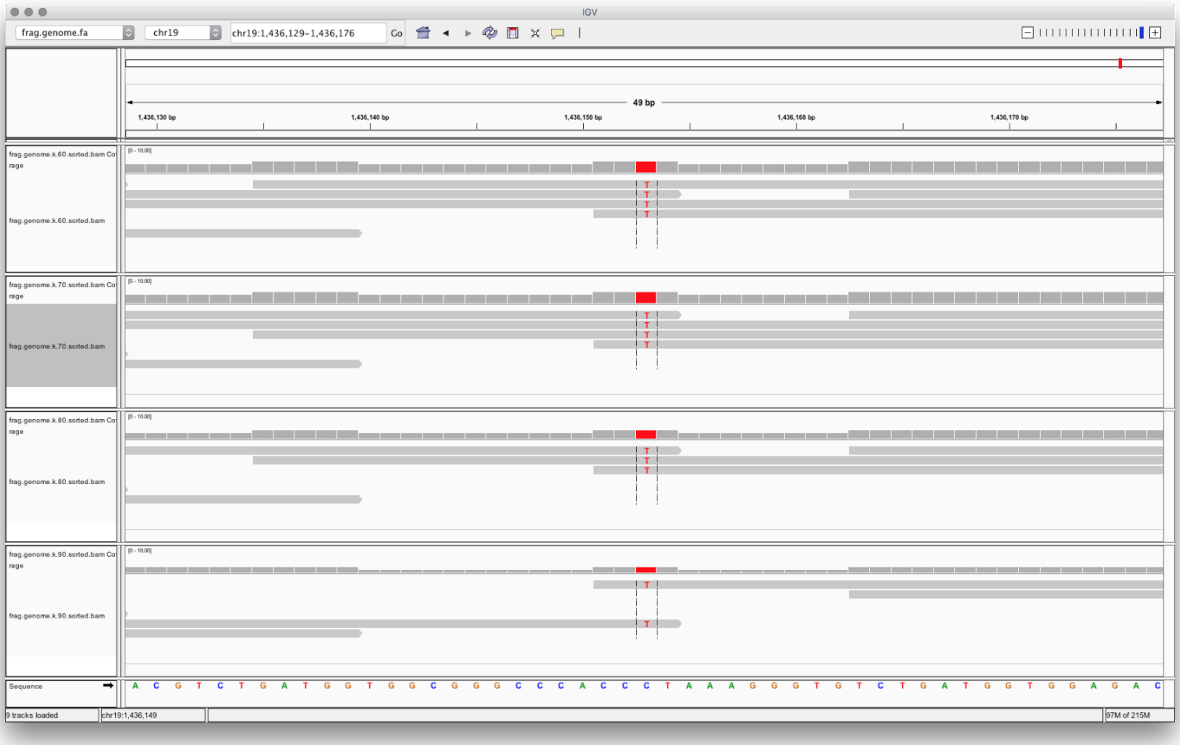
Additional deliverables:

a. Locations of SNVs we found

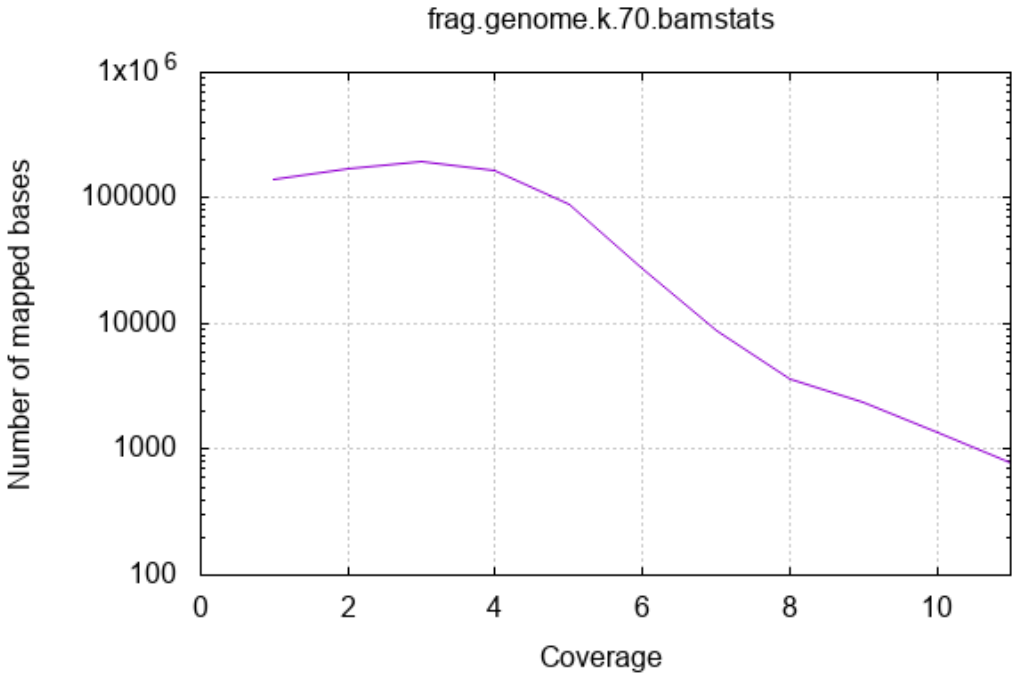
Screen shot of the first few (there were 1549 total)



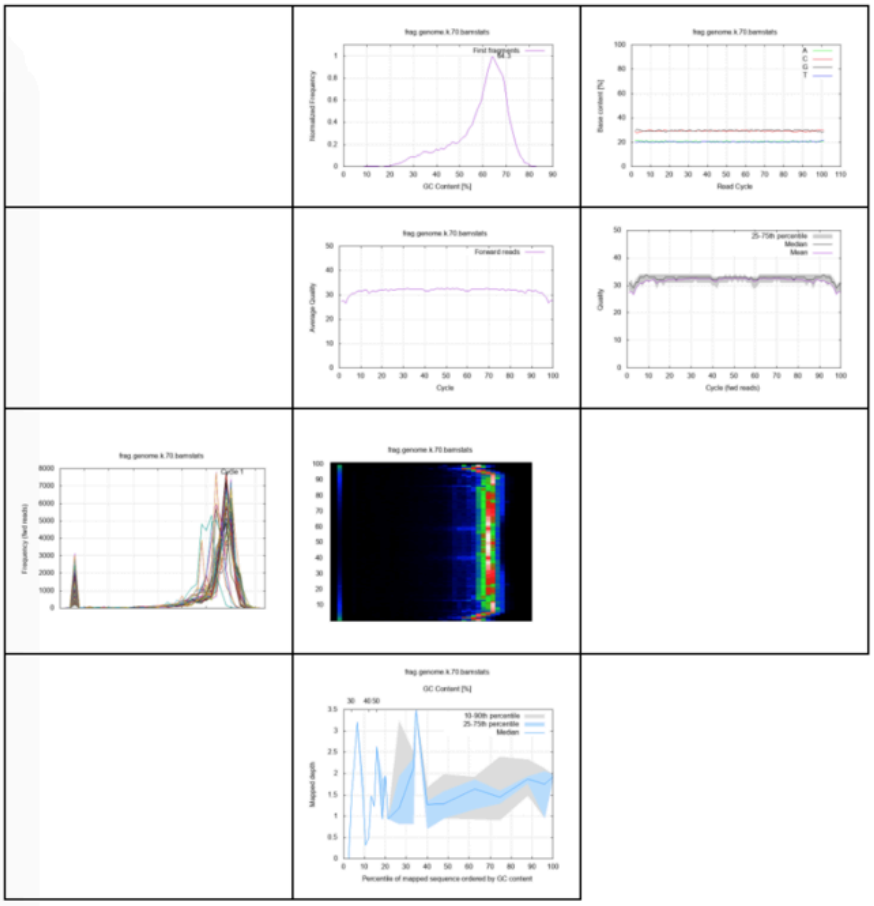
b. Screen shot of one of these SNVs



coverage X mapped regions:



more analysis:



Reads

total:	24,937	
filtered:	0	(0.0%)
non-primary:	0	
duplicated:	0	(0.0%)
mapped:	24,937	(100.0%)
zero MQ:	0	(0.0%)
avg read length:	100	

Bases

total:	2,493,700	(100.0%)
mapped:	2,493,700	
error rate:	0.00%	

b. bowtie

1. RNA-seq data

100% percent aligning accuracy.
# of error reads: 0

2. Shotgun data

61.52% aligning accuracy.

Improvements:

There are not too many areas for leverage aside from the number of mismatches (which we did) and the possibility of there being multiple indices where the support value is tied (which we did not do.

I'm not sure there is a straightforward way to break a tie between two equal support values so I think the idea would be to just find the minimum kmer size which results in one unique max value for all supports for all kmers across the reference sequence.

What we did, as explained above is took whatever max value was returned and used that value as mapping index. How we evaluated our performance was by [].

Recommendations:

I don't really have any recommendations I think it's a cool project. Maybe one slight critique is to show some practical examples of this at work with current variant callers etc. A lot of the stuff we go over in class is really cool, if there was a reference paper (this is semi-recent) that explains this method and shows its *practicality* in awesome challenges I think it makes it that much more cool.