Andrew Gordon

CMSC423 Project 1 Writeup

Dr. Patro

3/27/20

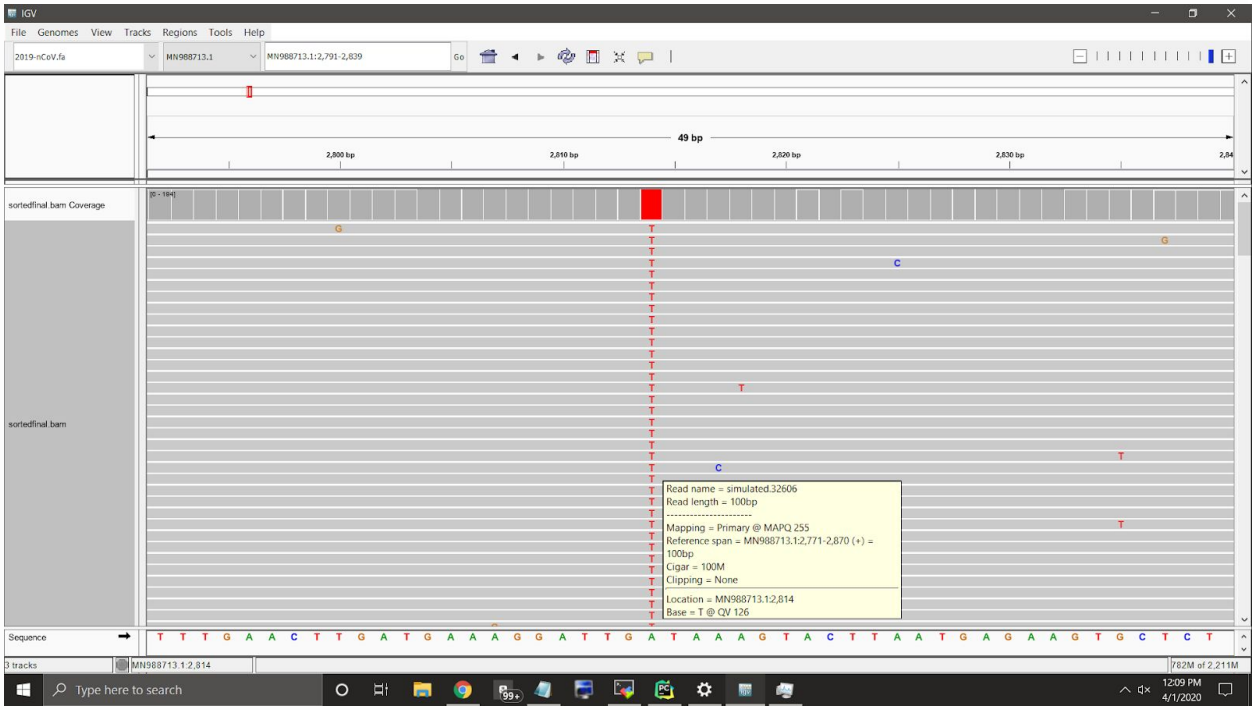
Detailing My Findings

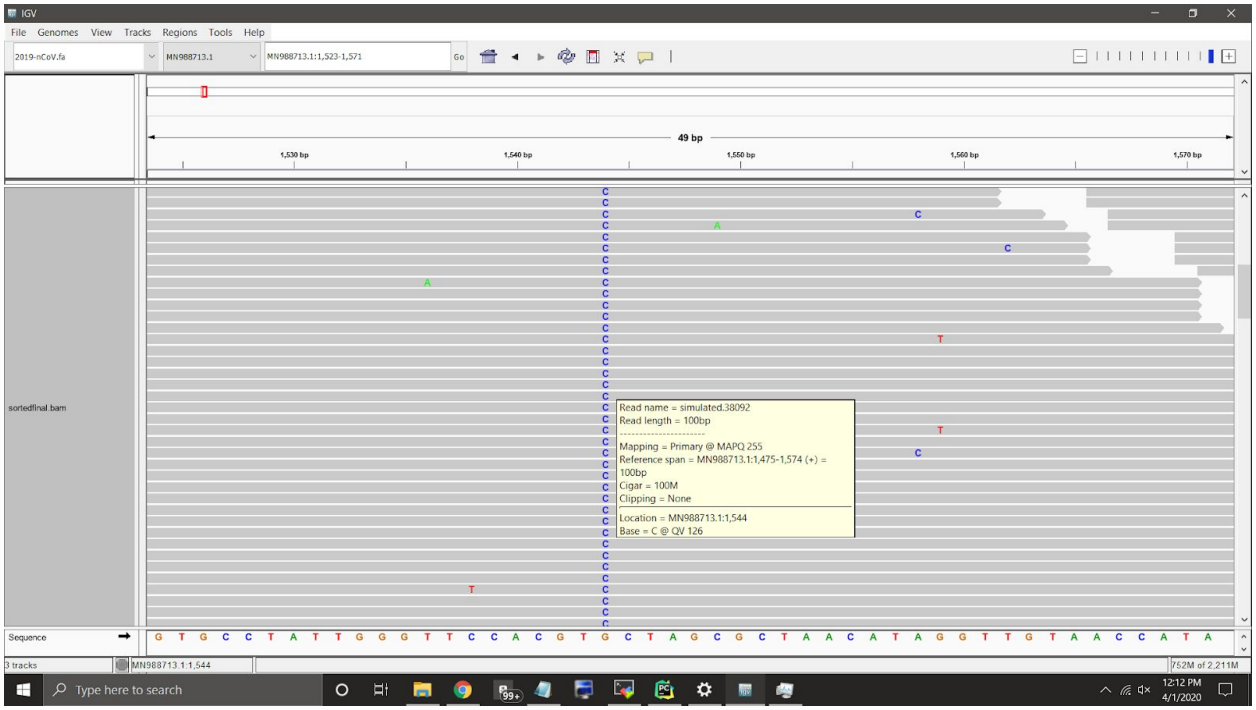
Discovered variants:

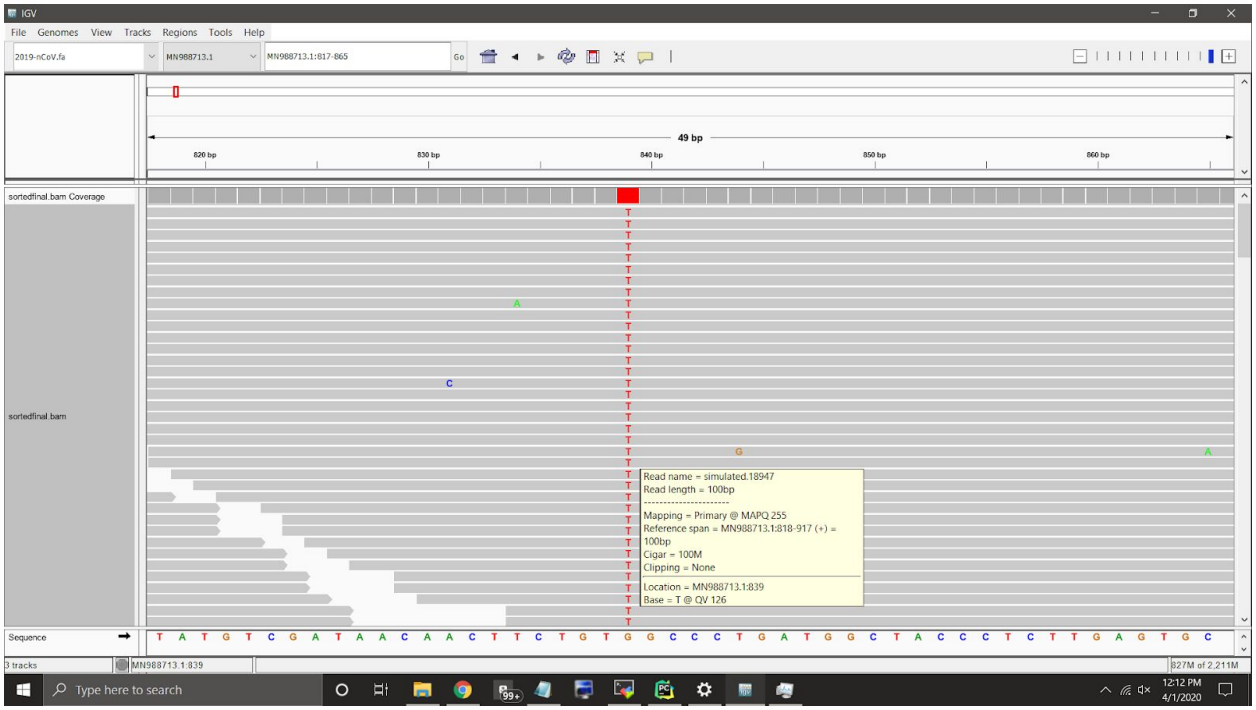
What variants did you discover? What is the evidence you found for these variants?

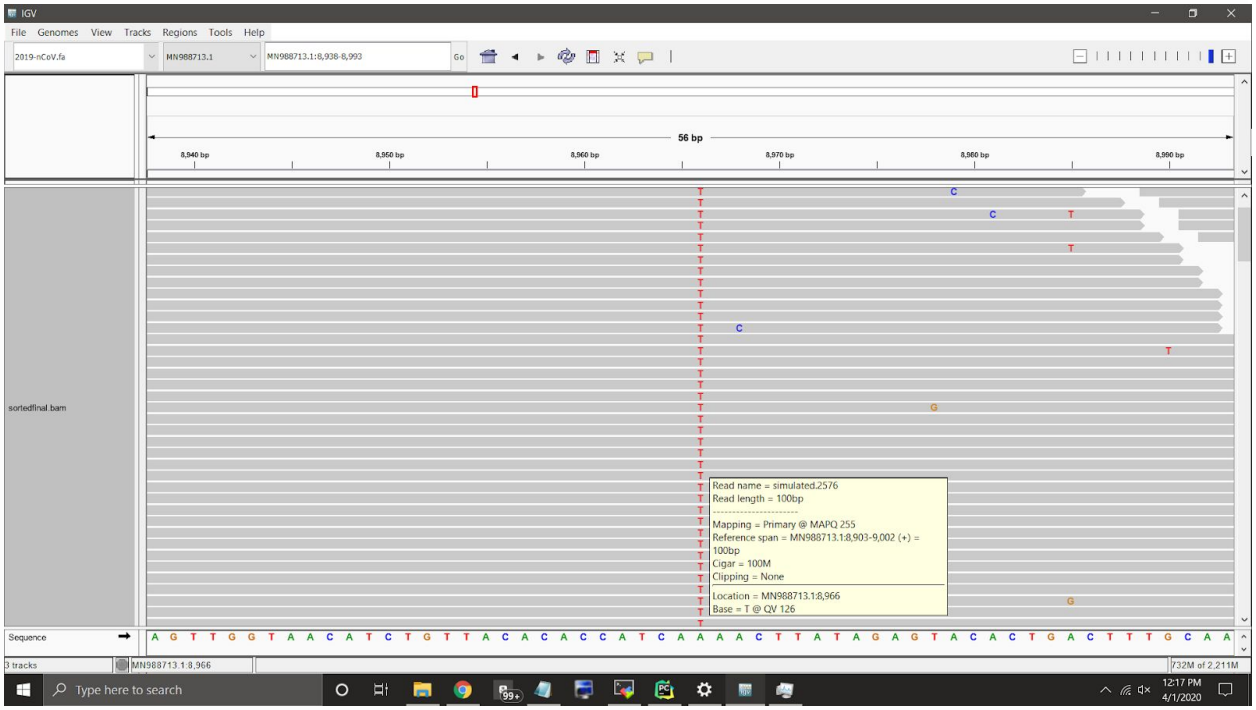
In the project description, it was said that there are 8 variants encoded through these readings. To my understanding, a variant is a place where the reads consistently differ from the genome, whether that's in mismatches, insertions, deletions, etc. Through the use of IGV, I was able to find some areas where these differences occurred.

The first four variants in the reads and the genome that I found were the single nucleotide variants (SNV) which occurred at positions 839, 1544, 2814 and 8966 in the genome. These are all areas where the base at the position in the genome is replaced with another base in the sample reads. Included below is evidence in the form of screenshots that these variants exist and are consistent through the read samples we ran alignments on. As you can see from all of the screenshots, the genome differed from the reads in one position, giving us a mismatch and a variant at those positions listed above. For position 839, we replace a G with a T, position 1544 we replace a G with a C, position 2814 we replace an A with a T and for position 8966 we replace an A with a T.

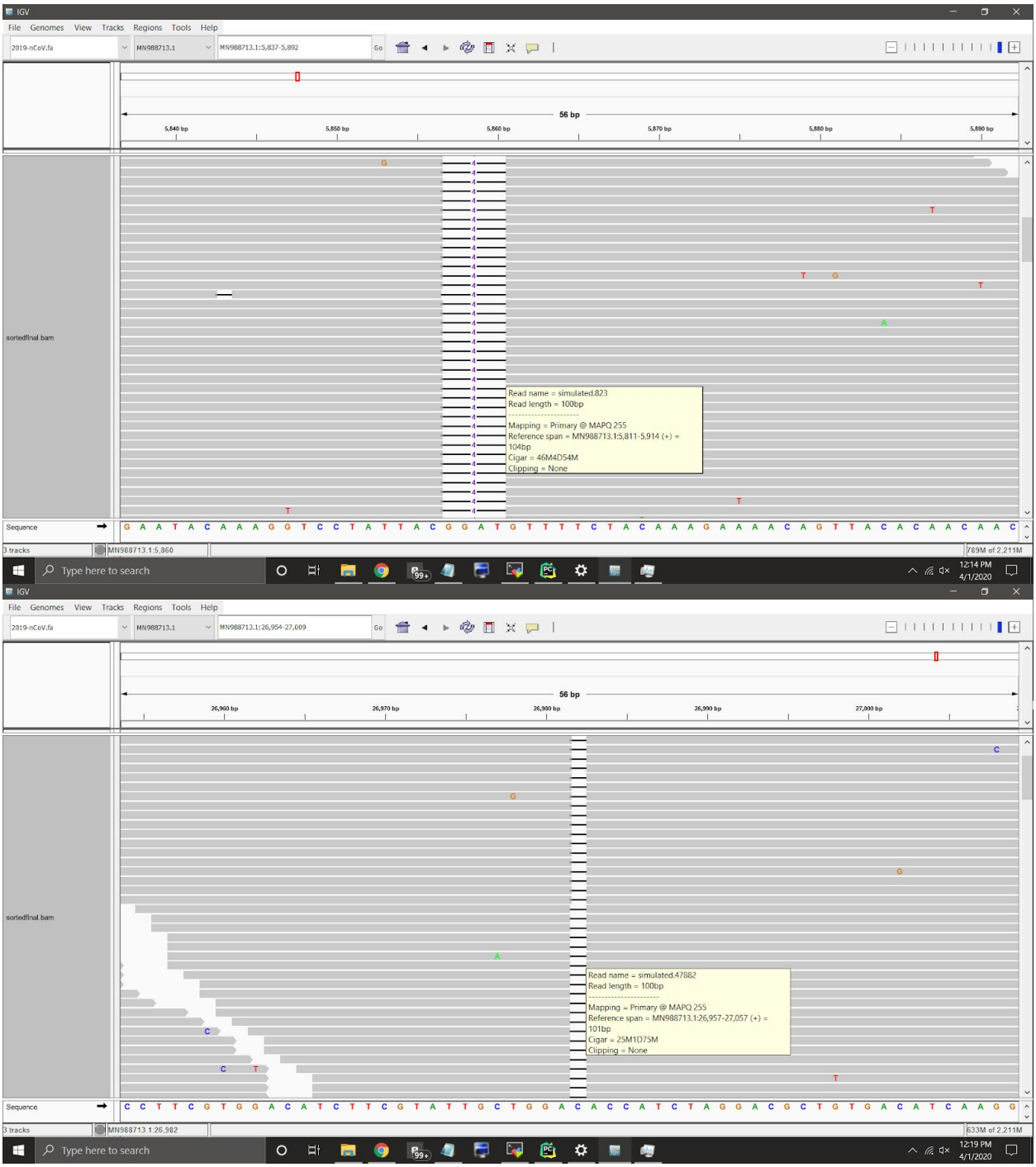

IGV
File Genomes View Tracks Regions Tools Help
2019-nCoV.fa
MN988713.1
MN988713.1:2,791-2,839
Go
49 bp
2,800 bp
2,810 bp
2,820 bp
2,830 bp
sortedfinal.bam Coverage
sortedfinal.bam
Read name = simulated.32606
Read length = 100bp
Mapping = Primary @ MAPQ 255
Reference span = MN988713.1:2,771-2,870 (+) = 100bp
Cigar = 100M
Clipping = None
Location = MN988713.1:2,814
Base = T @ QV 126
Sequence
T T T G A A C T T G A T G A A A G G A T T G A T A A A G T A C T T A A T G A G A A G T G C T C T
3 tracks
MN988713.1:2,814
782M of 2,211M
Type here to search
12:09 PM
4/1/2020


IGV
File Genomes View Tracks Regions Tools Help
2019-nCoV.fa
MN988713.1
MN988713.1:1,523-1,571
Go
49 bp
1,530 bp
1,540 bp
1,550 bp
1,560 bp
1,570 bp
sortedfinal.bam
Read name = simulated.38092
Read length = 100bp
Mapping = Primary @ MAPQ 255
Reference span = MN988713.1:1,475-1,574 (+) = 100bp
Cigar = 100M
Clipping = None
Location = MN988713.1:1,544
Base = C @ QV 126
Sequence
G T G C C T A T T G G G T T C C A C G T G C T A G C G C T A A C A T A G G T T G T A A C C A T A
3 tracks
MN988713.1:1,544
752M of 2,211M
Type here to search
12:12 PM
4/1/2020

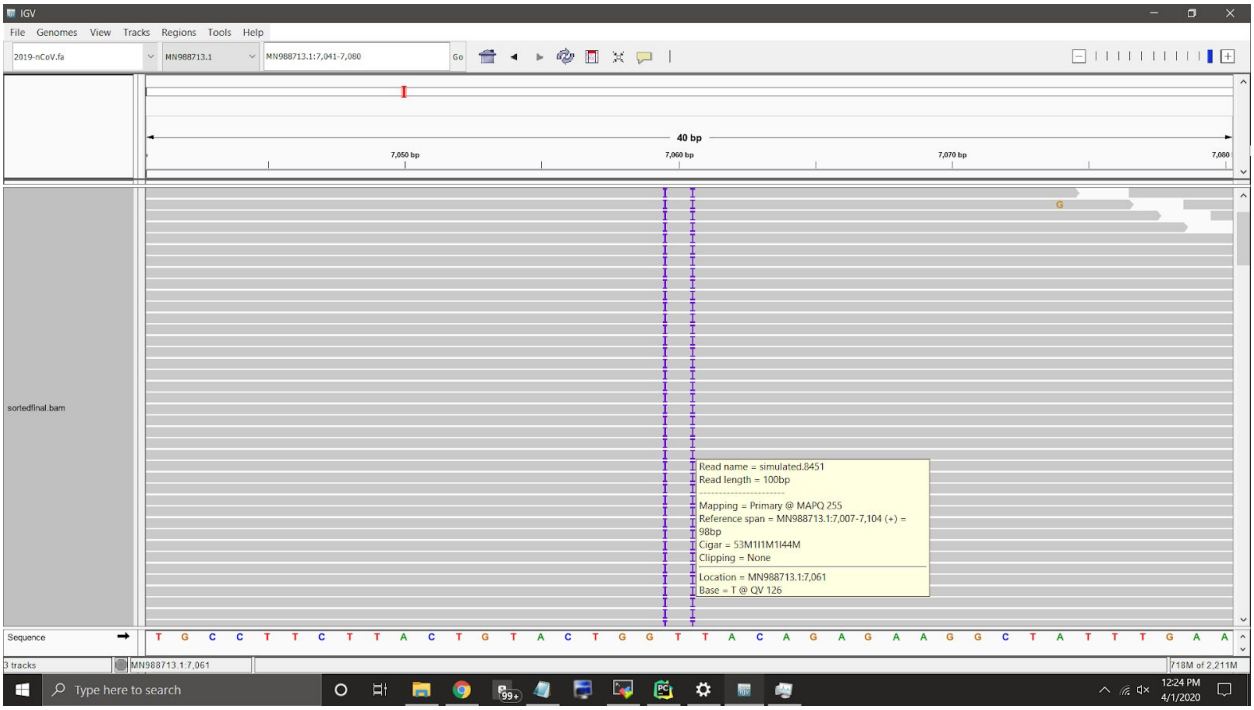

IGV
File Genomes View Tracks Regions Tools Help
2019-nCoV.fa
MN988713.1
MN988713.1:817-865
Go
49 bp
820 bp
830 bp
840 bp
850 bp
860 bp
sortedfinal.bam Coverage
sortedfinal.bam
Read name = simulated.18947
Read length = 100bp
Mapping = Primary @ MAPQ 255
Reference span = MN988713.1:818-917 (+) = 100bp
Cigar = 100M
Clipping = None
Location = MN988713.1:839
Base = T @ QV 126
Sequence
T A T G T C G A T A A C A A C T T C T G T G G C C C T G A T G G C T A C C C T C T T G A G T G C
3 tracks
MN988713.1:839
827M of 2.211M
Type here to search
12:12 PM
4/1/2020

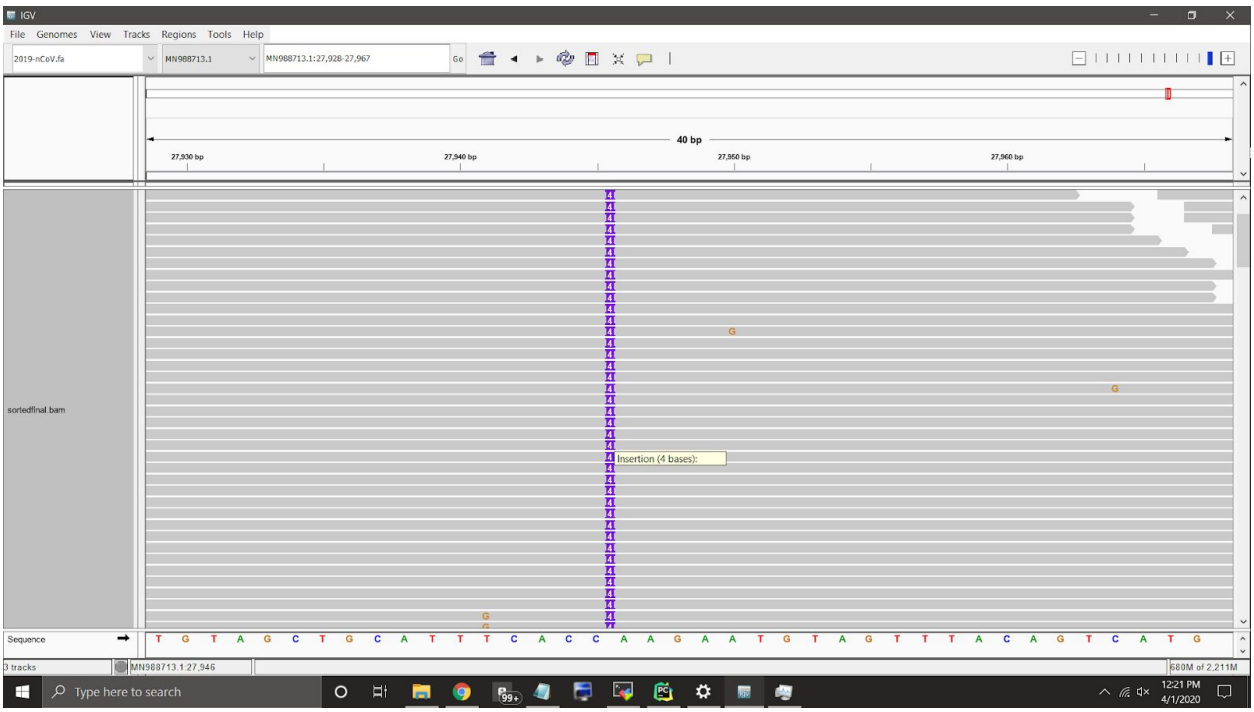

IGV
File Genomes View Tracks Regions Tools Help
2019-nCoV.fa
MN988713.1
MN988713.1:8,938-8,993
Go
56 bp
8,940 bp
8,950 bp
8,960 bp
8,970 bp
8,980 bp
8,990 bp
sortedfinal.bam
Read name = simulated.2576
Read length = 100bp
Mapping = Primary @ MAPQ 255
Reference span = MN988713.1:8,903-9,002 (+) = 100bp
Cigar = 100M
Clipping = None
Location = MN988713.1:8,966
Base = T @ QV 126
Sequence
A G T T G G T A A C A T C T G T T A C A C A C C A T C A A A A C T T A T A G A G T A C A C T G A C T T T G C A A
3 tracks
MN988713.1:8,966
732M of 2.211M
Type here to search
12:17 PM
4/1/2020

The next two variants that I found were deletions at positions 5856 (a deletion of four bases) and 26982 (a deletion of one base). As you can see below, the black lines represent a deletion of the bases from G to T, meaning that the genome has a base string of GGAT at that position that the reads did not. From the screenshot below that, you see a similar situation however it's a smaller deletion, meaning the genome had a variant of one base (C) at that position from the reads.



Lastly, I found two variants that related to insertions in the genome. These were found at positions 7060 and 27945 in the genome and screenshots of them can be seen below. For the first one, we can see that there is an insertion before (insertion of T) and after (insertion of A) the base T in the genome sequence. In the second screenshot (position 27,945), we have an insertion between bases C and A in the genome of the base string of length 4 (AGTG).





Overall, I found 8 different variants in the genome and the reads, four of which were SNVs, two of which were deletions and two which were insertions. By combing through the results in IGV, all of these variants stuck out pretty clearly, enabling me to quickly draw conclusions about the results of my index and alignment functions.

Design:

What specific design decisions did you make in your tool, and why?

When I started this project, I didn't really have a vision for what I wanted it to look like and how I wanted to get from point A to point B. My mindset was to keep chipping away at it starting with the index function and then getting into the nitty gritty details of the align function. After weeks of working, I finally got to the end of the alignment function and ran it to find out that it was going to take over five days to run the entire alignment. I thought this was a bug but after analyzing it further, it definitely wasn't.

From there I scanned through my code using libraries such as cProfile and the debugger tool in Pycharm to find where a lot of time was being spent during execution. The first thing that I did was create a dictionary that stored all substrings from the genome that we had already visited at line 375. This makes it so that for each read we don't recompute things that we already have, potentially saving us a good deal of time.

Another part of my code that was killing time was the function that finds the interval of matches for each seed. Since this function is running five times for each read, that means it's running five million times so if it's slow then that's going to destroy efficiency. My original approach was to find the interval of the starting character and then loop through that entire range,

finding the first and last occurrences of the next character in the read and then adjusting the range accordingly, however this is extremely inefficient. I then remembered that I should be using the occurrence table to help me and was able to transform a linear operation into a constant lookup operation which drastically saved time as well (line 195).

Aside from those larger fixes, I had a lot of areas where I made small tweaks to my program that enabled my program to make a few less function calls here and there, however described above were the tweaks that allowed me to drop from over five days to run the entire program to right around a day and a half. This may not sound great, but I was definitely proud of it.

If you were going to start the project again from scratch, what, if anything, would you do differently?

If I were to start this project over again, I would do a lot differently. First of all, I would make sure I had a solid understanding of the end goal of the project and all of the intricacies that came along with it. For example, it would've been extremely helpful if I knew what a SAM file was, what a cigar string looked like and what an alignment looked like before I started writing code. That would've saved me a lot of time considering there were several occasions that I had to go back and scrap code because it didn't do what I needed it to.

Aside from that, I wish I would have mapped out the structure of my project more and had a method of testing my results before actually coding. As I mention below, I was blindly coding a majority of this project and I had no idea if what I was doing was right. Due to this, once I finished a function I would just throw a couple short strings into it and if it looked right, I

just assumed it was. When we're getting into strings with thousands of characters being aligned with a million strings, a lot changes and every possible case in your code is tested, often breaking it. If from the start I mapped out the functions I needed to write and what their output should look like, I'd be able to implement one by one and test them to ensure correctness. From there, I could begin to integrate them together and continue testing them to ensure they were working together properly.

Outside of coding issues, I found that I had a big issue with time. My laptop is pretty old and doesn't run like it used to. I've never had a project before that really tested the capabilities of my laptop's processing power until this. The first time that I ran the alignment algorithm on reads.fa, I estimated based on a few calculations that it would take my program 128 hours to align all one million reads in the file. I've never had a project where it took more than thirty seconds to run the program so this was a shock to me. If I could go back, I would've been more cognizant of time complexity from the start instead of having to go back at the end and find where potential bottlenecks may occur. As I mentioned above, I was able to implement conditions that first enabled me to get the time from 128 hours to 97 hours and then from there I was able to make some more changes that got my time down to 46 hours to run the program, resulting in a close to 300 percent speed increase.

Lastly, I probably would've borrowed my sister's mac and done this project there if I could go back. As I mention below I had countless issues due to the fact that I'm on a Windows machine that easily could've been avoided, saving tens of hours and making this project a lot less of a headache for me.

Roadblocks:

What part of the project did you find the most difficult?

I'd say that this was the most challenging project I've had during my time at the University of Maryland. The content itself wasn't too hard to grasp, however the implementation was complex and there was a steep learning curve to the software needed for the project.

My biggest struggles in this project came from the fact that I'm a Windows user and all of the software/libraries seemed to be catered towards mac/linux users. I spent a lot of time (I'd say over ten hours) figuring out alternatives to the software and libraries recommended and then from there spent a lot more time figuring out how those libraries compare to the recommended ones. In terms of total time spent on this project, this is where a great deal of my time was spent.

Aside from library and software issues, understanding the file types was definitely a difficult task. We had to learn what FASTA, SAM and BAM files were and how to properly format them which doesn't sound too hard, however was tedious in practice. For the SAM file alone we had to decode what all of the variables were and manually write them to a file. From there, we had to figure out how to format a cigar string using the results of our dynamic programming function which definitely took some time to understand.

I think the hardest part of this project was the lack of direction and clarity I had throughout completing it. A majority of the time that I was working on implementing the code, I had no idea what my end goal was and if I was even remotely on the right path. I understood the general idea of this project but I knew it was a lot more complex than the summary I had in my head and that proved to be true. Every time that I felt like I made a step forward in progress, it seemed like there were now five to ten more things that needed to be fixed.

Where did you get stuck and how did you (hopefully) overcome this difficulty?

Most of the times I was stuck during this project had to do with trying to find alternatives to the tools provided in the documentation (I had to figure out simplesam, picard, etc.). My first thought was to look on Google for answers to how to use these programs, however it turned out that this wasn't helpful. Most of the answers to questions about picard were answered by people saying "just use samtools" (which I couldn't because I have Windows and spent nearly two hours setting up a WSL which in the end didn't work) so then I turned to Piazza. It seemed like others were having similar struggles and we were able to come up with solutions together with the help of Dr. Patro.

The area in implementation that I got stuck at was writing to the sam file and then converting it to a bam file. Picard seems to have an extremely rigid SAM parser and it caused me A LOT of issues. The biggest problem I had with it was that it wasn't recognizing certain parts of my cigar string. For example, if I had the string "90M3D1M2D4M" then it would give me an error and say my cigar is only 95 encodings long and needs to be 100 (even though it clearly totals up to 100 encodings). I realized eventually that the deletions "D" weren't being interpreted correctly.