Analysis of Variants Called by BCFtools and Snippy using Nextflow

I created a Nextflow script *main.nf* that has two key usage methods. The first method of usage is validation of variant callers by simulating mutations on a fasta file and then creating approximately 30x depth paired end reads. The mutated reads are then aligned, ran through snippy and BCFtools, and have their variants compared with a truth vcf. The second main usage takes two fastq short read files and a reference genome. These reads are aligned and variants are called, and then the two vcfs are merged. Both methods output summary statistics.

Validation Code

To run the pipeline in validation mode usage should follow:

```
nextflow run main.nf --reference [reference file] --prefix
[intermediate file name] --outdir [out directory] --simulate_reads
true
```

Simulate reads must be set to true otherwise the user will be required to provide fastq files. Reference file (.fasta) is always required for usage.

Validation mode applies nextflow process mutate_and_simulate, which takes the input reference genome and creates 300 SNPs and 20 InDels. This is accomplished using *simulate_reads.py* in the scripts/ folder. This script contains functions for insertions, deletions, and SNPs. It also contains a function to ensure mutations don't overwrite or overlap. This is accomplished through casing. The reference genome is converted to uppercase. Then snps and indels are added as lowercase. When the location for a mutation is randomly determined, the script first checks to ensure there are no lowercase characters in the range of the InDel or SNP. Mutations are logged in a *mutations.csv* file. mutate_and_simulate then simulates 33x depth paired end reads (100bp long with a step size of 6. These reads are saved as fastq1.fq and fastq2.fq respectively.

After fastq files are made the two modes of the pipeline converge to undergo alignment and variant calling. Alignment stores a bam and bam index (.bai) file. Variants are called using BCFtools and Snippy. Only the VCF files are preserved from variant caller output.

The two methods diverge again if a truth file (*mutations.csv*) is present. This is done inside of the *compare_vcs.py* script in the scripts/ folder. *compare_vcs.py* merges BCFtools and Snippy output with the truth vcf when it is available. SNPs are merged by matching position, ref and alt columns directly, only allowing exact matches.

Due to different VCF representations of InDels, InDel matching is less precise. InDel prefixes and suffixes are determined first. This is accomplished in the insertions_match and deletions_match functions. These functions create an a prefix and suffix that is the length of the longer sequence (alt for insertions, ref for deletions) minus the shorter sequence (ref for insertions, alt for deletions). The prefixes and suffixes are then compared by the indel_chunks_match function to ensure they overlap. The script also produces summary statistics for the comparison.

Validation Code Results

I ran the validation code on the two given fasta files through the following usages:

```
nextflow run main.nf --reference data/NC_037282.1.fasta --prefix
NC_037282 --outdir ~/griffin/task2/task2_nextflow/NC_037282_results --
simulate_reads true
```

```
nextflow run main.nf --reference data/EcoliK12-MG1655.fasta --prefix
EcoliK12-MG1655 --outdir ~/griffin/task2/task2_nextflow/EcoliK12-
MG1655_results --simulate_reads true
```

The summary statistics (*NC_037282_compare_summary.txt*) for NC_037282 reveal clear limitations of my method for merging VCF. To assess the success of both variant callers, the nextflow pipeline outputs true positives, false positives (identified by variant caller when it shouldn't have been) and false negatives (not identified by variant caller). Both tools identify 320 total SNPs and InDels, matching the expected number. BCFtools has 8 false positives and false negatives. Snippy has 4 false positives and false negatives. Further investigation of the merged vcf (*NC_037282_compare.txt)* reveals that this is likely due to genomic repeats or differences in variant representation methods that are undetected by the script.

The E.coliK12-MG1665 has similar results, however this time the variant callers do not identify the expected 320 mutations. BCF tools identified 313 true positives, and 4 false positives, failing to identify at least 3 mutations. Snippy results were similar with 312 true positives and 5 false positives.

Overall, this suggests that both variant callers are quite accurate at assessing reads with high quality (near perfect) reads.

Pipeline Code

The pipeline method doesn't create simulated reads and therefore reads and a reference must both be specified by the user. This method works the same as the validation mode, except now there is no truth genome to compare to. Instead, BCFtools and Snippy VCFs are directly compared in the *compare_vcs.py* script. Furthermore, since there is not truth VCF, false positives and negatives are not known, so the summary output contains the number of matched variants and unique variants. Usage of pipeline mode should follow:

```
nextflow run main.nf --reference [reference file] --fastq1
[fq1.fastq.gz] --fastq2 [fq2.fastq.gz] –prefix [intermediate file
name] --outdir [out directory]
```

Simulate reads is automatically set to false and therefore should not be included.

Pipeline Code Results

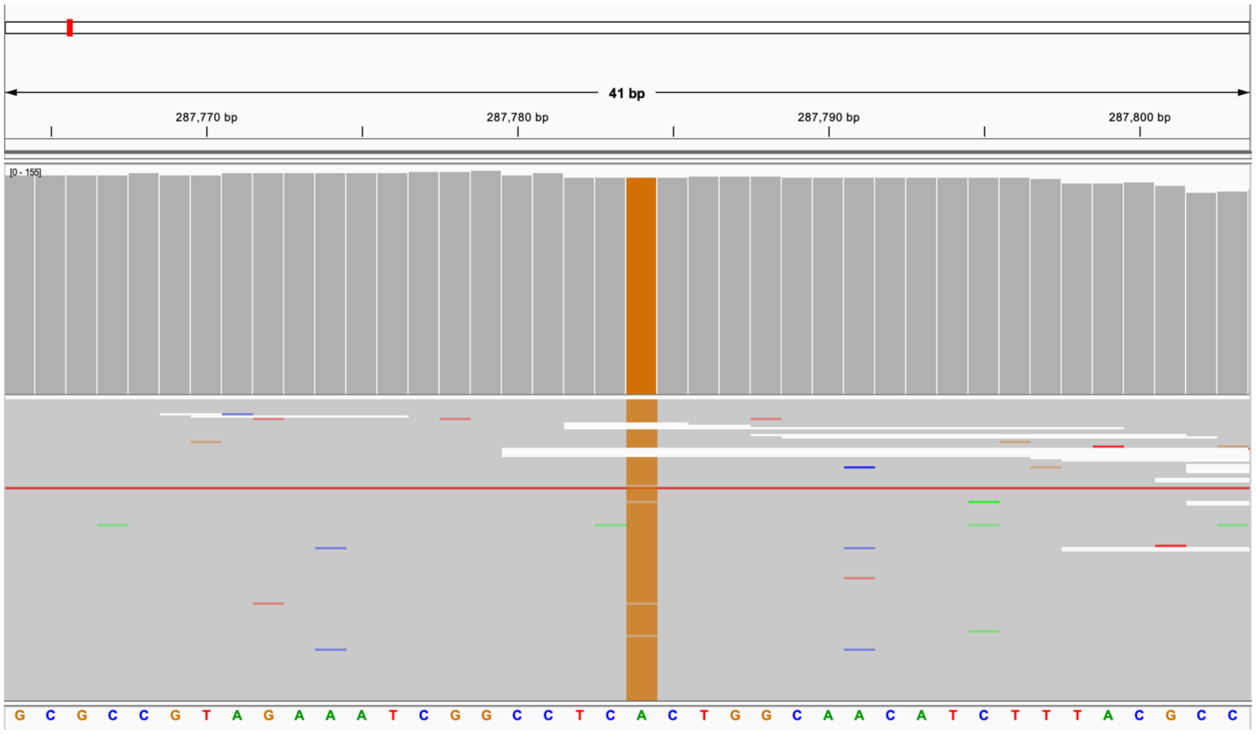
I ran the pipeline on the provided SRR25083113 fastq files. Looking at this output in NIH's Sequence Read Archive allowed me to determine that these files come from the Escherichia coli O157:H7 Strain. I used E. coli O157:H7 Sakai reference genome for alignment.

```
nextflow run main.nf   --reference data/ecoli_O157H7_sakai.fasta   --
fastq1 ~/griffin/task2/task2_nextflow/data/SRR25083113_1.fastq.gz   --
fastq2 ~/griffin/task2/task2_nextflow/data/SRR25083113_2.fastq.gz   --
prefix SRR25083113   --outdir
~/griffin/task2/task2_nextflow/SRR25083113
```

Running the pipeline on unsimulated reads produced surprising results. Snippy and BCFtools identified a shared 784 variants. BCFtools identified 920 unique variants, far more than BCFtools which only identified 7. Furthermore, the vast majority of these variants were SNPs in similar regions of the genome.

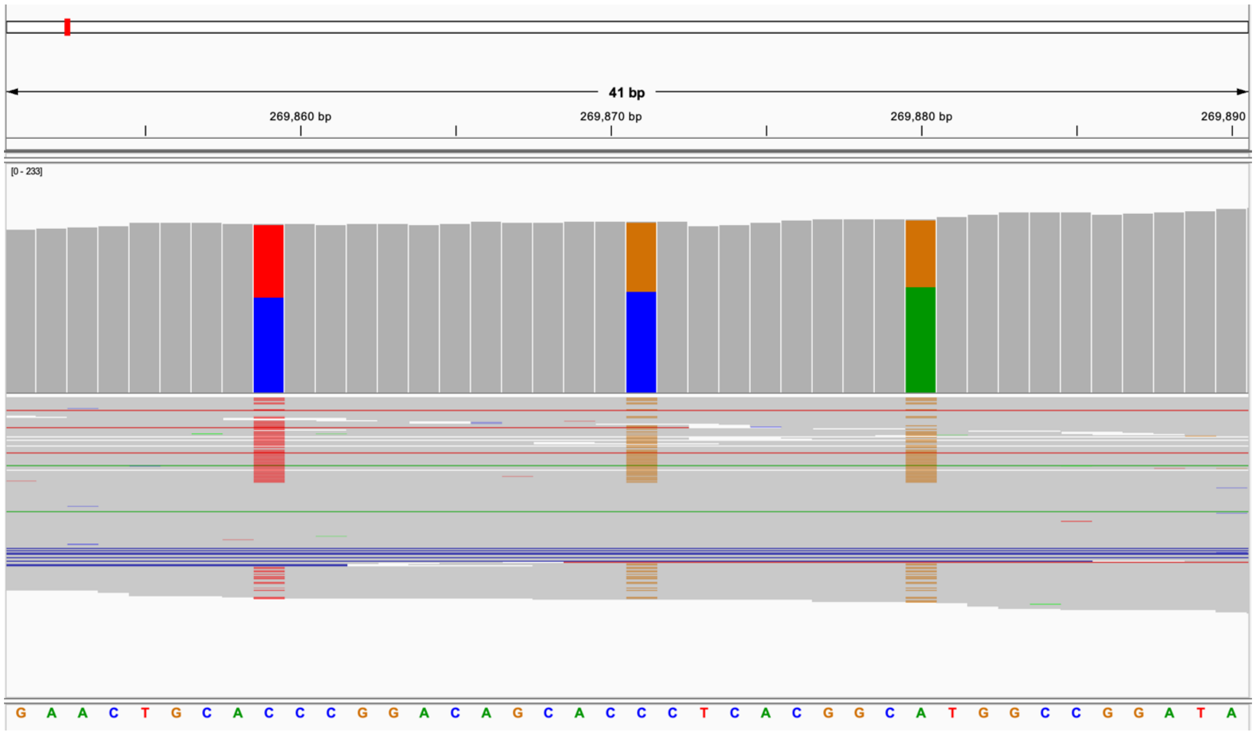
Further investigation reveals that Snippy has stricter filtering requirements for what is considered a variant. When opening the SRR25083113 reference and bam files in IGV we can see this visually.

Figure 1: Example Variant called by both Snippy and VCF tools.



41 bp
287,770 bp
287,780 bp
287,790 bp
287,800 bp
[0 - 155]
G C G C C G T A G A A A T C G G C C T C A C T G G C A A C A T C T T T A C G C C

Figure 2: Example Variants only called by BCF tools.


41 bp
269,860 bp
269,870 bp
269,880 bp
269,890
[0 - 233]
G A A C T G C A C C C G G A C A G C A C C C T C A C G G C A T G G C C G G A T A

We see that in some reads the variant is not present while in others it is. This in turn makes combining the outputs of variant callers extremely valuable; Snippy output can be merged with BCFtools to provide the user more confidence in the results. Variants called by both programs are very likely to be true variants (true positives), where variants called by only one program require further filtering or investigation.

The nexflow pipeline, data, and results can be found on Climb at jovyan:~/griffin/task2/task2_nextflow.