Small Group Exercise #7:

Variant calling from transcriptomic data

1. Explore the documentation for the BCFtools here. Follow the instructions to load the bcftools module into your session on the ASC. Note, if you logout and back on again, you will need to re-load the module. If you want to know which modules are currently loaded, run `module list`.

2. Create a new directory for this exercise called `L_luymesi_trans_variation` and copy the three (3) files from the `Lamellibrachia_luymesi_transcriptome_variant_calling` folder in the `/home/shared/biobootcamp/data` directory to your new directory using `cp` and a wildcard (i.e., `*`). Now, move into your `L_luymesi_trans_variation` directory to continue with the exercise.

    i. Assess the contig assembly in `Lamellibrachia_luymesi_sub1M_NON_NORM_TRI_05_2015.fasta` using `get_fasta_stats.pl` and the `-T` option (what does this option do to the output?). Record the summary statistics in the table below.

Contig statistics:	Value (before filtering):	After filtering
Number of contigs		
Total bp		
Shortest		
Longest		
Average length		
Average GC%		
Non-ACGT bases		

    ii. The exercise will focus on contigs **>2,000** bp in size, so those need to be isolated. Run `select_contigs.pl -help` and read over the information under USAGE: (near the top) to 1) identify the required option flag and 2) structure the command. When ready, execute the command and save the size filtered contigs to a file called `Lamellibrachia_luymesi_2000bp_plus.fasta`

iii. Assess the contigs in `Lamellibrachia_luymesi_2000bp_plus.fasta` using the `get_fasta_stats.pl -T`. Note whether (and how) the summary statistics differ from (2a) above by adding them to the table in the column **After filtering**.

3. Next, use the `module load` command to make bowtie2, samtools, and picard available interactively in your workspace.

4. Now, we will make two index files for our reference fasta file. For this step, we will use picard and samtools.

   i. **Dictionary index file (picard)**: Note the java file we need and how we specify it's location on the asc. You can read more about this sequence analysis package in here.

   ```
   java -Xms2g -Xmx4g -jar /opt/asn/apps/picard_1.79/picard-tools-1.79/CreateSequenceDictionary.jar REFERENCE=Lamellibrachia_luymesi_2000bp_plus.fasta OUTPUT=Lamellibrachia_luymesi_2000bp_plus.dict
   ```

   ii. **Fai index file (samtools)**:

   ```
   samtools faidx Lamellibrachia_luymesi_2000bp_plus.fasta
   ```

5. Execute `bowtie2-build` and examine the usage and options. Now rerun `bowtie2-build` to create an index using `Lamellibrachia_luymesi_transcriptomic_variants_index` as the *bt2_index_base* and **Lamellibrachia_luymesi_2000bp_plus.fasta** as the *reference_in*. No additional options are needed.

6. Now run the following command (and elsewhere, document what this command is accomplishing by running `bowtie2 -h` and `samtools view` to dissect the options being used):

```
bowtie2 \
  -p 2 \
  --sensitive-local \
  -x Lamellibrachia_luymesi_transcriptomic_variants_index  \
  -1 Lamellibrachia_luymesi_transcriptomic_sub1M_L001_R1_001.fastq \
  -2 Lamellibrachia_luymesi_transcriptomic_sub1M_L001_R2_001.fastq | \
  samtools view -bS - > Lamellibrachia_luymesi_transcriptomic_variants.bam
```

This will take a bit of time to run, so review the command to understand all the options used and take a minute to review what you've done so far.

7. Once the above command is done, a summary is printed to the screen. Note the overall

alignment rate, which is quite low. Why might this be the case?

8. Next, use `samtools sort` on the **Lamellibrachia_luymesi_transcriptomic_variants.bam** file and output the result to a file called *Lamellibrachia_luymesi_transcriptomic_variants.sorted* (why are you doing this?).

9. Next, use `samtools index` on the *Lamellibrachia_luymesi_transcriptomic_variants.sorted.bam* file (again, why are you doing this?).

10. Lastly, we will use picard to add read groups and index the file for gatk compatibility.

First type (takes several minutes):

```
java \
  -jar /opt/asn/apps/picard_1.79/picard-tools-1.79/AddOrReplaceReadGroups.jar \
  INPUT=Lamellibrachia_luymesi_transcriptomic_variants.sorted.bam \
  RGPL=Illumina \
  RGPU=lane1 \
  RGLB=sub1M \
  RGSM=L_Luymesi \
  RGID=luymesi \
  OUTPUT=Lamellibrachia_luymesi_transcriptomic_variants.sorted.wHeader.bam
```

Then type:

```
java \
  -Xms2g \
  -Xmx4g \
  -jar /opt/asn/apps/picard_1.79/picard-tools-1.79/BuildBamIndex.jar \
  INPUT=Lamellibrachia_luymesi_transcriptomic_variants.sorted.wHeader.bam \
  OUTPUT=Lamellibrachia_luymesi_transcriptomic_variants.sorted.wHeader.bam.bai \
  VALIDATION_STRINGENCY=SILENT
```

11. We are ready to start our variant calling. First we will use ***GATK*** to create a VCF file of all potential variants:

    i. For this step, we will submit the job to the ASC queue. As done in previous exercises, copy `/home/shared/biobootcamp/data/example_ASC_queue_scripts/GATK_example.sh` to your current directory. Submit it to the ASC queue system using the directions at the bottom of the script.

    ii. While this job runs, explore and read more about the options of GATK here. Note, we are

using the older versin of GATK, not the latest one, which has many more dependencies.

iii. At this point, we will work directly with the VCF file in various ways. `less -S Lamellibrachia_luymesi_transcriptomic_variants.vcf` and look it over. Note how information in the file is structured, which should be familiar from our lecture on variant calling and the VCF standard format.

iv. Run the following command and look at the output:
`wc -l Lamellibrachia_luymesi_transcriptomic_variants.vcf && grep "#" Lamellibrachia_luymesi_transcriptomic_variants.vcf | wc -l && grep -v "#" Lamellibrachia_luymesi_transcriptomic_variants.vcf | wc -l`

What does this command summarize from your VCF file (note that the `&&` means *"proceed to the next command if the previous command DID NOT produce an error"*)?

For reference, consult: `less -S Lamellibrachia_luymesi_transcriptomic_variants.vcf`

or try the following separately: `grep "#" Lamellibrachia_luymesi_transcriptomic_variants.vcf` and `grep -v "#" Lamellibrachia_luymesi_transcriptomic_variants.vcf` without the `| wc -l` to see the output.

12. Now, let's generate some summary statistics for our variants:

i. `bcftools stats Lamellibrachia_luymesi_transcriptomic_variants.vcf > Lamellibrachia_luymesi_transcriptomic_variants_ALL.stats`

ii. We can now generate some graphics to summarize our statistics, with all of them being placed in their own specific directory: `plot-vcfstats_bbc -p ALL_variants_stats_output_dir/ Lamellibrachia_luymesi_transcriptomic_variants_ALL.stats`. This will use the local `plot-vcfstats` located in the bootcamp bin directory, which should already be in your PATH.

iii. Change directory into `ALL_variants_stats_output_dir/` and long list the contents of the directory. Many of the files will be familiar by their extensions. If you are curious of what's in the ***.dat** file, feel free to explore using `cat` or `less`.

iv. Now you will download some of the PDF files to your laptop for viewing. Connect to the ASC with file transfer software of your choice or using `scp` as previously. Look in `ALL_variants_stats_output_dir/` for the file summary.pdf and download to a directory on your computer called `ALL_variants_stats_output`. Open the file and interpret the information. Does it make sense to you in the context of molecular biology and evolution?

13. Note that at this point, we still need to filter false positives from our potential variants. Variant filtering is an area of active development and is not easy. The annotations produced by variant callers provide only indirect hints about quality of the calls and an approach that worked for one dataset may not work for another. This is where you will need to experiment to determine what is best for the dataset at hand.

    i. Let's use `bcftools filter` to do the filtering, so run the command to view its options. A good measure of the callset quality is often **ts**/**tv**, the ratio of transitions and transversions (which of these classes of mutations do you expect to be more common?). Other useful metrics are sequencing **depth** (DP, where bigger than twice the average depth indicates problematic regions and is often enriched for artifacts) and proximity to **indels** (since these may represent areas of questionable mapping and alignment and creating artifacts in variation).

    ii. The base command that we start with is: `bcftools filter -e'DP<10 || %QUAL<15 || DP>30' -g4 -sLowQual Lamellibrachia_luymesi_transcriptomic_variants.vcf` (and consult `bcftools filter` to understand the options being utilized). What is the output of this command (which should look familiar)? How does it differ from the output of `cat Lamellibrachia_luymesi_transcriptomic_variants.vcf`?

    iii. Rerun the command in (13.ii) above BUT now change values for DP, QUAL and/or -g in each iteration. To see the impact of the changes on the level of variant filtering, you might 1) want to only make a single change at a time in the command and 2) add the following to the end of the command `| grep -v "#" | grep PASS | wc -l` (what's this addition to the command doing? For example, you could change *PASS* to *LowQual* to quantify what is being filtered out).

14. Once you are happy with your level of filtering, its time to analyze the results.

    i. Redirect the output to a new VCF file: `bcftools filter -e'DP<XX || %QUAL<XX || DP>XX' -gX Lamellibrachia_luymesi_transcriptomic_variants.vcf > Lamellibrachia_luymesi_transcriptomic_variants_FILTERED.vcf` (NOTE: the X's in this command should be replaced with your chosen values from (13c) above and that the option `-sLowQual` has been removed).

    ii. Create the summary statistics and graphics as in #12 above, adding "**_FILTERED**" to file names so that they can be distinguished from your previous analyses.

    iii. Connect to the ASC with Cyberduck and download the file `summary.pdf` for the filtered analysis to a directory on the Desktop of your laptop called `FILTERED_variants_stats_output`.

    iv. Compare the `summary.pdf` files from the **ALL** and **FILTERED** analyses side-by-side on

your laptop. How do they differ from each other and (more importantly) why do they differ?

15. You can use `samtools tview` to view the BAM file and reference FASTA; looking for the variants that you have identified:
`samtools tview Lamellibrachia_luymesi_transcriptomic_variants.bam.sorted.bam Lamellibrachia_luymesi_2000bp_plus.fasta`. In samtools' tview mode, you get a simple but graphical representation of the read alignments; pressing `?` will bring up a help menu on navigating and a key regarding the colors. On Friday, you will learn some additional GUI based genome visualization tools.

16. You can instantly navigate to a particular variant by adding an additional option:

    ```
    samtools tview -p
    Lamellibrachia_luymesi_sub1M_NON_NORM_TRI_05_2015_c3462_g1_i1:930
    Lamellibrachia_luymesi_transcriptomic_variants.bam.sorted.bam
    Lamellibrachia_luymesi_2000bp_plus.fasta
    ```

    where `-p` is CHROM:POS (which is CHROMosome:POSition separated by a colon). To identify additional SNPs you might want to look at: `grep -v "##" Lamellibrachia_luymesi_transcriptomic_variants_FILTERED.vcf | awk 'OFS="\t"{print $1,$2,$4,$5}'` and replacing the CHROM:POS in the above with specific output from the first and second fields from this command.

This short exercise should serve as an example of how the tools you are learning can be used to parse and filter data and how the presence or absence of flags can change the behavior of a program. Lastly and as you have seen, variant filtering can be difficult: not being stringent enough can lead to false positives while filtering at too high stringency could exclude real variants. The best practice is to experiment with options and filtering levels and erring on the side of caution.

Some studies incorporate empirical validation, but this can be costly given the number of variants. For example if you identify 100,000 SNPs (a modest size), and you want to validate 1%, that requires you to validate 1000 SNPs, which can be costly and time intensive (especially using traditional methods such as Sanger sequencing).

More recently, researchers validate SNP calls by using different variant calling softwares or different types of genomic data (e.g. high

coverage exome data), where a larger fraction of the genome can be validated. These approaches allow you to better estimate the true and false positive rate in your dataset and make adjustments to your filtering accordingly.