Using the GBS analysis pipeline to analyze sequence data.

Step 1. If one of the CBSU BioHPC Lab workstations was reserved for you, it will be listed on the workshop website:

http://cbsu.tc.cornell.edu/ww/1/Default.aspx?wid=13

Please visit this website for instructions on how to access and use the Lab workstations.

If you would like to continue using a BioHPC Lab workstation outside of the workshop, you will need to reserve it as described in the user guide at http://cbsu.tc.cornell.edu/lab/use.aspx .

Step 2. The analysis software is started by running the command /programs/tassel/run_pipeline.pl . Data files shared by all participants are stored in the directory /shared_data/GBS . Personal data files are stored in the directory /workdir . You will need to create your own subfolder in /workdir before the exercise starts, and run all commands from this directory. Create a subfolder (using your own user name) with the command:

```
mkdir  /workdir/username
```

Change your working directory to the new directory with the command:

```
cd  /workdir/username
```

And create some new folders to hold the output files from the pipeline:

```
mkdir qseq
mkdir tagCounts
mkdir mergedTagCounts
mkdir topm
mkdir tbt
mkdir mergedTBT
mkdir hapmap
mkdir hapmap/unfilt
mkdir hapmap/mergedSNPs
mkdir hapmap/filt
```

Finally create symbolic links to the raw sequence data files with the command:

```
ln -s  /local_data/GBS/*qseq.txt qseq/
```

NOTE: A single period specifies the current working directory. Also note that the system will often finish typing long filenames if you type half of the name and press “tab”.

The GBS analysis pipeline is an extension to the program TASSEL, and, as such, GBS commands are run as TASSEL plugins in the following format:

```
run_pipeline.pl -fork1 -PluginName --plugin-option -endPlugin -runfork1
```

Each step of the pipeline is specified with a "fork" command and a number, since TASSEL can run several processes at once, and split and recombine their results. The fork option is followed by the name of the plugin, and any plugin-specific options. If no plugin options are provided, the program will print a list of available options. -endPlugin signals the end of plugin-specific options, and -runfork1 then runs the specified plugin.

Step 3. To begin, we identify GBS sequence reads in all .qseq files from the project. In order to find authentic reads, the software needs to know which DNA barcodes and restriction sites were used to create the library. This is done by specifying the name of the enzyme used, and a "key file" that associates DNA barcodes with sample names.

```
/programs/tassel/run_pipeline.pl -fork1 -QseqToTagCountPlugin -i qseq -o tagCounts –k
/local_data/GBS/rice.key -e apeki -endPlugin -runfork1
```

NOTE: Illumina base calling software versions 1.8 and greater will produce sequence in FASTQ format rather than QSEQ format. These are handled identically to QSEQ data, except that the TASSEL plugin used is called "-FastqToTagCountPlugin".

The software will print all barcodes found in the key file, then begin searching all .qseq files in the data directory for those barcodes. Results will be stored in files with the extension .cnt .

Step 4. Since .qseq files are too large to fit in memory, they must be analyzed one at a time, and the results merged. The following command merges files in the specified input directory to the specified output file. The option "-c 5" specifies that only unique sequences appearing more than 5 times should be accepted. This helps cut down on the large number of singleton reads produced by sequencing errors.

```
/programs/tassel/run_pipeline.pl -fork1 -MergeMultipleTagCountPlugin -i tagCounts -o
mergedTagCounts/riceGBStags.cnt -c 5 -endPlugin -runfork1
```

To format the sequences for alignment, repeat the command, using a different file name and specifying FastQ format with the option "-t".

```
/programs/tassel/run_pipeline.pl -fork1 -MergeMultipleTagCountPlugin -i tagCounts -o
mergedTagCounts/riceGBStags.fastq -c 5 -t -endPlugin -runfork1
```

Results will be output as a file named "rice.fastq" in the mergedTagCounts directory.

Step 5. Once unique reads have been found, they are aligned to a physical position in the genome. The following command aligns to the O. sativa reference genome, using the BWA program, and outputs results to the file "riceGBStags.sai". **Before you run this command, change the directory to mergedTagCounts (**cd mergedTagCounts**).**

```
bwa aln -t 8 /local_data/GBS/rice.fa  riceGBStags.fastq.tmp > riceGBStags.sai
```

Step 6. BWA results need to be converted to SAM format for reading. Results are output to the file "rice.sam" in the current directory (mergedTagCounts) .

```
bwa samse /local_data/GBS/rice.fa  riceGBStags.sai  riceGBStags.fastq > riceGBStags.sam
```

Step 7. Finally, SAM format is converted to our own alignment format. Results are written to "riceGBS.topm.bin" in the "topm" directory. **Before you run this command, change the directory to the next higher level (cd ..).**

```
/programs/tassel/run_pipeline.pl -fork1 -SAMConverterPlugin
-i mergedTagCounts/riceGBStags.sam  -o topm/rice.topm.bin -endPlugin -runfork1
```

Step 8. To match reads to the lines in which they were found, we search the .qseq files again, this time creating a matrix of sequence reads by their lines of origin (tags by taxa, or TBT).

```
/programs/tassel/run_pipeline.pl -fork1 -QseqToTBTPlugin -i qseq -o tbt
-k /local_data/GBS/rice.key -e apeki -t mergedTagCounts/riceGBStags.cnt -endPlugin -runfork1
```

NOTE: Illumina base calling software versions 1.8 and greater will produce sequence in FASTQ format rather than QSEQ format. These are handled identically to QSEQ data, except that the TASSEL plugin used is called "-FastqToTBTPlugin".

Step 9. We merge the TBT files created in the previous step to the file "rice.tbt.bin" in the "mergedTBT" folder.

```
/programs/tassel/run_pipeline.pl -fork1 -MergeTagsByTaxaFilesPlugin -i tbt
-o mergedTBT/rice.tbt.bin -endPlugin -runfork1
```

Step 10. Now that physical positions and lines of origin are known, reads that map to the same location in different lines can be clustered and compared to identify SNPs or indels. The results will be output as HapMap format genotype files named mergedTBT.c#.hmp.txt (where # = chromosome).

```
/programs/tassel/run_pipeline.pl -fork1 -TagsToSNPByAlignmentMTPlugin -i
mergedTBT/rice.tbt.bin -m topm/rice.topm.bin -o hapmap/unfilt -s 1 -e 12 -mnF 0.9
-endPlugin -runfork1
```

Step 11. At this point, SNPs have been called and can be viewed and analyzed with TASSEL. Depending on your experimental design, you may want to apply one or more data filters to your results. For example, the following command is used to merge duplicate SNPs that occur in overlapping reads from opposite strands.

```
/programs/tassel/run_pipeline.pl -fork1 -MergeDuplicateSNPsPlugin -hmp
hapmap/unfilt/mergedTBT.c+.hmp.txt -o hapmap/mergedSNPs/rice.mergedSNPs.c+.hmp.txt
-callHets -misMat 0.1 -s 1 -e 12 -endPlugin -runfork1
```

The following command will filter out SNPs that are not in LD with a neighboring SNP, with low representation across lines, and from lines with poor sequencing coverage.

```
/programs/tassel/run_pipeline.pl -fork1 -GBSHapMapFiltersPlugin -hmp
hapmap/mergedSNPs/rice.mergedSNPs.c+.hmp.txt
-o hapmap/filt/rice.mergedSNPs.filt.c+.hmp.txt -hLD -mnTCov .05 -mnSCov .05 -sC 1
-eC 12 -endPlugin -runfork1
```

NOTE: Pre-computed files created for this workshop are stored in the directory /shared_data/GBS/output.