Using the GBS Analysis Pipeline Tutorial
Cornell CBSU/IGD GBS Bioinformatics Workshop

Step 0: 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/lab/doc/wrkshp_CBSUlab_access.pdf

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 1: Creating Working Directory Structure

Data files shared by all participants are stored in the directory /local_data/GBS. Personal data files are stored in the directory /workdir. You will need to create your own subdirectory (username) and directory structure in /workdir before the exercise starts, and put the results from all commands in that directory.

NOTE FOR RUNNING COMMAND LINE PROGRAMS: The system will often finish typing long filenames if you type part of the name and press “tab”. This is called “tab auto-completion” and it works for the commands in this workshop. Please use it, it makes things much easier and avoids typos. (e.g. 01 “tab” in the first step)

Use this command to create the directory structure:

01_create_run_folders.sh

That script creates these files and directories.

/workdir/<username>
/workdir/<username>/01_RawSequence
/workdir/<username>/01_RawSequence/10244719_81PVTABXX_s_2_qseq.txt
/workdir/<username>/01_RawSequence/10244719_81PVTABXX_s_3_qseq.txt
/workdir/<username>/04_TOPM
/workdir/<username>/02_TagCounts
/workdir/<username>/02_TagCounts/02_MergedTagCounts
/workdir/<username>/02_TagCounts/01_IndividualTagCounts
/workdir/<username>/02_TagCounts/03_TagCountToFastq
/workdir/<username>/03_SAM
/workdir/<username>/06_HapMap
/workdir/<username>/06_HapMap/05_ImputedSNPs
/workdir/<username>/06_HapMap/02_MergeDupSNPs
Change to your working directory with this command:

```bash
cd /workdir/<username>
```

The GBS analysis pipeline is an extension to the program TASSEL; therefore, GBS commands are executed as TASSEL Pipeline plugins. The shell scripts you’ll be executing today use those plugins in the following way. For more details see these documents:

- [http://www.maizegenetics.net/tassel/docs/TasselPipelineCLI.pdf](http://www.maizegenetics.net/tassel/docs/TasselPipelineCLI.pdf)
- [http://www.maizegenetics.net/tassel/docs/TasselPipelineGBS.pdf](http://www.maizegenetics.net/tassel/docs/TasselPipelineGBS.pdf)

**Command line options to run plugins**

```bash
run_pipeline.pl -fork1 -PluginName -options -endPlugin -runfork1
```

**It can be run through following command line**

```bash
run_pipeline.pl -configFile config.xml
```

**Example command line**

```bash
/programs/tassel/run_pipeline.pl -Xmx5g -configFile xmls/02_01_QseqToTagCount.xml 2>&1 | tee -a logs/02_01_QseqToTagCount.log &
```

**Step 2.1: Count GBS Tags**

First, we identify GBS sequence tags in all QSEQ (.qseq) files from the project. In order to find useful GBS tags, the software needs to know which DNA barcodes and restriction sites were used to create the library. In this tutorial, the enzyme (ApeKI) and “key file” (rice.key) associates DNA barcodes with sample names. The software prints all barcodes found in the key file, and searches all QSEQ files in
GOAL: Main goal of this plugin is to capture all unique sequence tags separately from each sequence (QSEQ/FASTQ) file.

NOTE: Illumina base calling software version 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”. In future versions of TASSEL these two plugins will be merged into something like -SeqToTagCountPlugin for simplicity of operation.

Step 2.2: Merge Tag Count Files and Convert to FASTQ

QSEQ files are too large to fit in memory. They are analyzed one at a time, and the results merged. The following command merges files created in the previous step. The option “-c5” specifies that GBS tags appearing more than 5 times should be accepted. This helps reduce the large number of singleton reads produced by sequencing errors. The resulting file will be named:
/workdir/<username>/02_TagCounts/02_MergedTagCounts/*.cnt

02_02_run_mergetagcounts.sh

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<TasselPipeline>
  <fork1>
    <MergeMultipleTagCountPlugin>
      <i>02_TagCounts/01_IndividualTagCounts</i>
      <o>02_TagCounts/02_MergedTagCounts/Workshop_Rice_mergedtagcounts.cnt</o>
      <c>5</c>
    </MergeMultipleTagCountPlugin>
  </fork1>
  <runfork1/>
</TasselPipeline>
GOAL: Unique sequence tags from each separate sequence file will be merged together and produces master tag counts file having unique tags produced at least for 5 times across all sequence files.

To align the unique GBS tags to the reference genome using an external aligner (bowtie2 in this example), we need to convert them to FASTQ format. Execute this command to produce /workdir/<username>/02_TagCounts/03_TagCountToFastq/Workshop_Rice_20130915.fq

02_03_run_tagcounttofastq.sh

GOAL: Convert master tag count file to FASTQ for preparation to execute the Aligner.

Step 3: Align GBS Tags to Reference Genome

Once unique GBS tags have been found, they are aligned to a physical position in the reference genome. The following command aligns to the O. sativa reference genome, using Bowtie2, and outputs results (SAM format) to the file: /workdir/<username>/03_SAM/Workshop_Rice_20130915.sam

03_run_alignwithbowtie2.sh

GOAL: Tags will be aligned to reference genome in SAM format.

Step 4: Convert SAM to Tags on Physical Map (TOPM)

From the previous step, the SAM formatted file is converted to our alignment format (TOPM). Results are written to /workdir/<username>/04_TOPM/Workshop_Rice_20130915.topm

04_run_createTOPM.sh
GOAL: SAM format is converted to Tags On Physical Map (TOPM) format can be used further for SNP calling.

**Step 5: Match Tags to Samples (Taxa)**

To match GBS tags to the samples in which they were found, we search the QSEQ files again, this time creating a Tags by Taxa (TBT) matrix indicating how often each GBS tag occurs in each sample taxon. These files use HDF5 format for more efficient storage, ease of manipulation and the ability to handle very large data sets.

Execute this command to match tags from /workdir/<username>/01_RawSequence to their sample (taxon) producing: /workdir/<username>/05_TBT/01_IndividualTBT/TBT_Node1_20130915.h5

```bash
05_01_run_seqtotbthdf5.sh
```

GOAL: Tags present in master tag count file were matched across all taxa from all sequence files called as Tags By Taxa (TBT) file. This is a matrix produced for all tags by taxa but in hdf5 format.

Execute this command to merge all TBT matrices producing:

```
/workdir/<username>/05_TBT/02_MergedTBT/mergeTBTTHDF5_20130915.h5
```

```bash
05_02_run_mergetbthdf5.sh
```
**GOAL:** If multiple TBT files are produced, use this plugin to merge all TBT files

Execute this command to merge taxa with same sample prep id to produce:
/workdir/<username>/05_TBT/03_MergedTaxaTBT/mergeTBT HDF5_mergedtaxa_20130915.h5

```
05_03_run_mergetaxatbthdf5.sh
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<TasselPipeline>
  <fork1>
    <ModifyTBT HDF5Plugin>
      <o>05_TBT/03_MergedTaxaTBT/mergeTBT HDF5.h5</o>
      <c></c>
    </ModifyTBT HDF5Plugin>
    <runfork1/>
  </fork1>
</TasselPipeline>
```

**GOAL:** Same taxa can be present in multiple sequence files. These taxa are merged in this step.

Execute this command to pivot the TBT from previous step in preparation for SNP calling to produce:
/workdir/<username>/05_TBT/04_PivotMergedTaxaTBT/mergeTBT HDF5_mergedtaxa_pivot_20130915.h5

```
05_04_run_pivotmergetaxatbthdf5.sh
```

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<TasselPipeline>
  <fork1>
    <ModifyTBT HDF5Plugin>
      <o>05_TBT/03_MergedTaxaTBT/mergeTBT HDF5.h5</o>
      <p>05_TBT/04_PivotMergedTaxaTBT/mergeTBT HDF5_pivot.h5</p>
      <c></c>
    </ModifyTBT HDF5Plugin>
    <runfork1/>
  </fork1>
</TasselPipeline>
```

**GOAL:** This plugin pivots the TBT file and prepares it for SNP calling.

**Step 6.1: Call SNPs and Write Hapmap File**

Now that physical positions and tax are known for the GBS tags, those that map to the same location in different lines are compared to identify SNPs. The results will be output as Hapmap format genotype files named below where # is chromosome.
/workdir/<username>/06_HapMap/01_UnfilteredSNPs/04_PivotMergedTaxaTBT.chr#.hmp.txt.gz

```
06_01_run_callsnps.sh
```
GOAL: Tags mapped in the same physical location on reference genome were aligned together to identify SNPs. These SNPs are tabulated in Hapmap format. TASSEL can use these hapmap files directly for further analysis.

Step 6.2: Merge Duplicate SNPs from Opposite Strands

The following command merges duplicate SNPs that occur in overlapping tags from opposite strands producing:
/workdir/<username>/06_HapMap/02_MergeDupSNPs/<username>_MERGEDUPSNPS_GBS_Workshop_Rice_Build_September_2012_RC-1_chr#.hmp.txt.gz(where# = Chromosome).

06_02_run_mergedupsnps.sh
Step 6.3: Filter Hapmap

At this point, SNPs have been called (and duplicates merged) and the genotypes can be loaded into TASSEL and analyzed. However, it is usually a good idea to do some filtering first. The following command will filter out taxa with poor sequencing coverage and SNPs that are not in LD with a neighboring SNP or that have low call rates across taxa. The filtering parameters should reflect your experimental design (the settings here are for biparental recombinant inbred lines).

/workdir/<username>/06_HapMap/03_HapMapFilteredSNPs/<username>_HAPMAPFILTERED_GBS_Workshop_Rice_Build_September_2012_RC-1_chr#.hmp.txt.gz(where# = Chromosome).

06_03_run_hapmapfilters.sh

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<TasselPipeline>
  <fork1>
    <GBSHapMapFiltersPlugin>
      <hmp>06_HapMap/02_MergeDupSNPs/
        MERGEDUPSNPS_GBS_Rice_chr+.hmp.txt.gz</hmp>
      <o>06_HapMap/03_HapMapFilteredSNPs/
        HAPMAPFILTERED_GBS_Rice_chr+.hmp.txt.gz</o>
      <mnSCov>0.2</mnSCov>
      <mnTCov>0.05</mnTCov>
      <mnMAF>0</mnMAF>
      <mxMAF>1</mxMAF>
      <hLD></hLD>
      <mnR2>0.01</mnR2>
      <mnBonP>0.01</mnBonP>
      <sC>1</sC>
      <eC>12</eC>
    </GBSHapMapFiltersPlugin>
  </fork1>
  <runfork1/>
</TasselPipeline>

GOAL: Low quality SNPs can be removed based on their coverage and LD.