Table of Contents

I. GBS (Genotype By Sequencing) - With Reference

Attempts to find relevant SNPs by aligning DNA against a reference for the organism ***Required Files:*** *sequence data files for each chip (.fq), tray position/genotype sheet (.csv), barcode name/sequence sheet (.csv), reference genome files (.bt2)*

1. Backup any important data you have received to backup1 (in tar format)
2. Create Barcode.tab file, associating each sample name with it's barcode sequence
    - generated using two spreadsheets given from biologists
        - one should contain the sample name on each well position when they did their sequencing
            - a well is just a slot in the machine. Aka plate position
    - the other should contain the barcode assigned to each well
        - usually used the same barcode file, so this file can be reused
        - BC-Proton-PstI.xlsx
    - combine the two sheets to find the barcode associated with each sample id
    - all barcodes should be the same length
        - barcodes are usually suffixed with GAT. This gives a buffer so some can be added or removed to enforce same size
            - ensure that barcode file is in UTF8 format
            - sample barcode file: (sampleID, barcode)

1206	CTAAGGTAACGA
1207	TAAGGAGAACGA
WI	AAGAGGATTCGA

3. Split out chip fastQ files into seperate ones for each sample (barcode)
    - chip fastQ files contain giant lists of sequences, but we want to parse through and look at each sample individually
        - each chip represents a run. We split them out individually to avoid using a huge file, but maybe they could be joined first
    - use fastx_barcode_splitter to split out each barcode into it's own file for each chip

    - `$ cat ./Chip1.fastq | fastx_barcode_splitter.pl --bcfile barcode_file.tab --bol --prefix Chip1/ --suffix ".fq"`
        - prefix is used to specify which directory to save the output to. This direcotry will have to be created before the command is run
        - must be run once for each Chipi.fq file
    - if you have fastq.gz files, you can use gunzip instead of cat to uncompress in place
        - `$gunzip -c ./Chip1.fastq.gz | fastx_barcode_splitter.pl --bcfile barcode_file.tab --bol --prefix Chip1/ --suffix ".fq"`

4. Merge sample fastQ files from each chip, into combined sample files

    - go into each new ./Chipi directory, find .fq files with matching names, and concatenate them into a new merged directory

    - can use Craig's simple Merge script:

    ```
    ls $1 | while read FILE; do
            cat $1/"$FILE" $2/"$FILE" $3/"$FILE" >> $4/"$FILE"
      done
    ```

        - usage: `Merge.sh ./Chip1 ./Chip2 ./Chip3 ./Merged`
        - reads through all files in first folder, finds matches in others, and saves results in last folder
5. If not zipped, gzip up fastQ files to save space
    - `gzip *.fq`
6. Create new analysis folder, and symlink all the .fq.gz files into it
    - to keep things cleaner
    - analysis folder should be named with today's date
    - `cp -s ./Merged/*.fq.gz ./28Apr16_Analysis`
7. Create initial quality statistics file
    - use fastQC to generate quality reports
        - `/usr/bin/FastQC/fastqc -t 8 *.fq.gz -o ./original_quality`
    - use my stats_from_zip.py script to generate quality spreadsheet of the data
        - `python ./stats_from_zip.py ./original_quality > ./quality_sheets/original_quality.tab`
        - includes average/set length, average/std quality, total reads, range of length
8. copy BT_ST_Pipeline_SE.sh and TruSeq3-SE.fa into analysis folder
    - SE is for single ends, must use PE files for paired ends
    - BT_ST_Pipeline_SE.sh is found in the svn repo in data2/svn
    - TruSeq3 was part of the trimmomatic package. The same file is reused every time we do a GBS. You can find it in any old analysis folder

9. Run BT_ST_Pipeline on each .gz file
    - can use find command to do them all at once
        - `find . -name "*.fq.gz" -exec ./BT_ST_Pipeline_SE.sh {} \; > output_logs.txt`
    - will create a filder with outputs for each sample
    - pipeline does 6 things:
        a. trims data using trimmomatic
            - cuts off barcodes and bad data
            - aims for 70% survival rate
        b. creates fastQC report
        c. aligns reads against reference using Bowtie
        d. creates BAM file with sorted data, with duplicates removed
        e. creates an index of the BAM file
            - so that file can be viewed later in a viewer
        f. appends path of BAM file to bam_list.txt
            - done to all samples, so they can all be used as input into a single combined pileup
10. create a new spreadsheet for statistics, now that the data has been trimed/cleaned by the pipeline
    - the pipeline already created fastqc files, you just need to point the stats_from_zip script at the parent directory of them all
    - `python ./stats_from_zip.py ./ > ./quality_sheets/trimmed_quality.tab`
11. Use Samtools to create a combined pileup of all samples
    - mpileup takes all the sequences, and stacks them on top of each other. Represents information at each position on the chromosome
    - requires the path to a reference genome to stack against
    - uses bam_list.txt file generated by BT_ST_Pipeline to reference all bam files
    - `samtools mpileup -gf /path_to_ref/ref.fa -b ./bam_list.txt > all.raw.bcf`
        - generates binary file (bcf)
12. Use bcftools to convert the binary bcf file into a human readable vcf
    - `bcftools call -vcO z -o all.raw.vcf.gz all.raw.bcf`
        - the step can be combined with previous step using piping
        - `samtools mpileup -gf ./ref.fa -b ./bam_list.txt | bcftools call -vcO z -o all.raw.vcf.gz_`
13. Filter pileup data using vcftools to remove unnecessary data
    - `vcftools --gzvcf ./all.raw.vcf.gz --maf 0.1 --minDP 6 --max-missing 0.8 --recode --out 2May2016_MAF01_DP6_MM08`
        - filters out bad data and resaves it as a new vcf file
        - maf = minimum allele frequency
        - output file is named to make it obvious what was done to it
14. Turn filtered vcf file into human readable FilterCalls spreadsheet (aka VCF tab)
    - `cat 2May2016_MAF01_DP6_MM08.recode.vcf | vcf-to-tab > 2May2016_MAF01_DP6_MM08.tab`

- send this spreadsheet to someone. It is one of the desired outputs
- example output:

```
    <table>
  <tr><td>Marker</td><td>Chr</td><td>Pos</td><td>1206</td><td>1207</td></tr>
  <tr><td>TGAACv1_scaffold_1</td><td>1AS</td><td>11160</td><td>C/C</td><td>T/T</tc
   <tr><td>TGAACv1_scaffold_2</td><td>1AL</td><td>11162</td><td>A/A</td><td>C/C</t
```

II. UNEAK GBS

Another way to find markers, this time with no reference needed. ***Required Files:*** *sequence data files for each chip (.fq), tray position/genotype sheet (.csv), barcode name/sequence sheet (.csv)*

1. Backup any important data you have received to backup1 (in tar format)
2. gzip up the sequence data files for each chip
   - the pipeline expects fastq.gz files
   - `$ gzip ./*.fastq`
3. rename sequence files to follow format: CELL_LANE_fastq.gz
   - ex, C_1_fastq.gz
4. create a barcode 'key' file
   - associates the following data:
     - barcodes
       - sample (genotype) names
       - flowcell/lanes (chips)
       - plate positions
   - generated using two spreadsheets given from biologists
     - one should contain the sample name on each well position when they did their sequencing
       - a well is just a slot in the machine. Aka plate position
     - the other should contain the barcode assigned to each well
       - usually used the same barcode file, so this file can be reused
       - BC-Proton-PstI.xlsx
     - combine the two sheets to find the barcode associated with each sample id, along with plate positions
   - flowcell/lanes corelate to the sequence data files (Chip files, C1.fq)
     - If you want to extract the same genotypes from each sequence file, you may have to do some copy/pasting
   - sample barcode file: (sampleID, barcode)

```
  <table>
```

```
<tr><td>Flowcell</td><td>Lane</td><td>Barcode</td><td>Sample</td><td>PlateName<
<tr><td>C</td><td>1</td><td>CTAAGGTAAC</td><td>PI_189547</td><td>A</td><td>0</t
<tr><td>C</td><td>1</td><td>TAAGGAGAAC</td><td>PI_322734</td><td>B</td><td>1</t
<tr><td>C</td><td>2</td><td>CTAAGGTAAC</td><td>PI_189547</td><td>A</td><td>0</t
</table>
```

  - copy UNEAK_Pipeline_SE.sh and TruSeq3-SE.fa into analysis folder
    - UNEAK_Pipeline_SE.sh is found in the svn repo in data2/svn
      - might have some issues with trimming the raw data. I will have to update the repo soon
    - TruSeq3 was part of the trimmomatic package. The same file is reused every time we do a GBS. You can find it in any old analysis folder
  - run UNEAK_Pipeline_SE.sh

III. Associative Analysis

Attempts to find which genotypes are responsible for observed phenotype traits ***Required Files:*** *pileup data for sequences (.vfc), HapMap genotype sheet (.csv), phenotype trait sheet (.csv)*

1. Backup any important data you have received to backup1 (in tar format)
2. Create structure.str file from Hap Map (genotype) file
   - I created a script called structure_generator for this purpose
   - example structure:

Genotypes	TP158	TP188
C10046	2	3

3. Find the number of populations (K) in the data
   - use the structure software
   - want to run structure n iterations for each possible K value, to find the optinal
     - typically do ~10 iterations
       - can use Craig's MultiStructure script to run multiple runs in parallel
     - requires generating parameter files using Structure's front end
       - should be in your user directory. If not, Craig sent it as an email
         - use Structure Harvester website to find optimal K value
     - look at deltaK chart at bottom
     - can also use Craig's StructureChooseK.m script to generate similar charts
       - If speed is an issue, use fastStructure instead of regular structure
     - different implementation designed to run faster, but we had issues finding reliable K value
     - run multiple times for different K values to find best one:

        - `$ python ~/proj/fastStructure/ /structure.py –input=./structure.str –output=./results/structure_output --format=str -K 1`
    - to find optimal K:
        - `$ python ~/proj/fastStructure/chooseK.py –input=./results/structure_output`
4. Generate QMatrix.tab file
    - should be output by the structure command
    - represents the percent for each population for each Genotypes
    - i used space separated values, because that's what my sample data had, but I'm not sure if it matters
    - sample file:

<Covariate>		
<Trait>	Q1	Q2
C10046	0.961	0.039
C11006	0.946	0.054

    - assuming Structure was used rather than fastStructure:
        - Q Matrix is saved in each Kn-n_f file
        - can be extraced using my CreateQMatrix python script
5. Prepare genotype file
    - Hap Map = genotype file
    - just need to ensure that all genotype names match the names in the Q Matrix
6. Prepare phenotype trait file
    - the file for my analysis was called Biochemical composition 2013 - Abdullah.tsv
    - ensure all genotype names match other filters
    - add header information for Tassel (top two rows)
    - expected format:

<Phenotype>		
taxa	data	data
Name	Protein	Starch
C10046	18.04	63.72
C11006	17.75	12.01

7. Input Genotype, Phenotype, and QMatrix files into Tassel
    - to start the program:
        - `/usr/bin/tassel-5-standalone/start_tassel.pl &`
    - set genotype fils as a hapmap, the others can use best guess

8. Use TASSEL to do a GLM Analysis
    - filter genotype sites to remove bad data
        - I cut out sites with count < 50 (there were 90 sequences), and minimum frequency < 0.05
    - cut out last column in QMatrix
        - I was told this is necessary to keep the data linearally independant, otherwise it would break
            - may not be true
        - highlight Q matrix and filter > traits
    - join phenotype/(filtered)genotype/(filtered)QMatrix into single table
        - highlight all three, select Data>Intersect join
    - run GLM command on combined file
        - use 1000 permutations
        - results in two outputs:
            - Stats File
                - shows how associated each geneotype is with each trait (pvalue)
            - Genotypes file
                - shows how each genotype effects each trait
        - These files are important, save them in a results foler somewhere
9. Use TASSEL to do a MLM Analysis
    - create kinship file from filtered genotype file
        - highlight genotype, select Analysis>kinship
        - use scaled_IBS method
    - run MLM analysis with kinship file and combined file selected
        - combined file = intersection of genotype/phenotype/QMatrix
        - results in three files. Save these files along with the GLM files in results directory
10. Generate XLSX files with traits on their own sheets for each result file
    - previously, result files will be in plain text csv, with each trait stacked on top of each other. Biologists will want to view each trait independantly, in a single xlsx workspace
    - I created a script that will do this automatically, called CreateSheets .py
11. send 5 files to biologists

IV. RNA Seq

Used for measuring the expression level of genes. Looks at coding portions of genome, rather than the entire thing. See RNA_Seq_AnalysisMehod file in repo for Craig's description of the process ***Required Files:*** *Paired end sequence data (.fq), observed expression data for each genotype (.csv)*

1. Backup any important data you have received to backup1 (in tar format)

2. Copy RefList file into working directory

- points to the reference genomes for A,B,D genomes
- sample file:

  > A /mnt/data1/reference_genomes/Triticum_Aestivum/Triticum_aestivum.IWGSC1.0+popseq.30.dna.chromosome.A /mnt/data1/reference_genomes/Triticum_Aestivum/Triticum_aestivum.IWGSC1.0+popseq.30.chromosome.A.gff3 B /mnt/data1/reference_genomes/Triticum_Aestivum/Triticum_aestivum.IWGSC1.0+popseq.30.dna.chromosome.B /mnt/data1/reference_genomes/Triticum_Aestivum/Triticum_aestivum.IWGSC1.0+popseq.30.chromosome.B.gff3 D /mnt/data1/reference_genomes/Triticum_Aestivum/Triticum_aestivum.IWGSC1.0+popseq.30.dna.chromosome.D /mnt/data1/reference_genomes/Triticum_Aestivum/Triticum_aestivum.IWGSC1.0+popseq.30.chromosome.D.gff3

3. Use RNA-Seq_Pipeline_PE.sh on paired end files to create a transcriptome assembly GTF file for the genome

   - `./RNA-Seq_Pipeline_PE.sh ./Paired_End_1.fq.gz ./Paired_End_2.fq.gz`
     - This command does 5 things:
     - Trims out bad quality data using Trimmomatic
       - aims for 70% survival rate
       - barcodes are cut off on both ends because data is paired end
     - FastQC analysis
       - generates quality reports for data
     - Aligns reads to references using Tophat
       - uses multiple references specified in reference file (./RefList)
         - done separetly because tophat can only handle small references at this time
         - each line represents one of the chromosomes of the reference genome
       - returns accepted_hits.bam that is needed later
     - Merges alignments into a GTF file using Cufflinks
       - will create a gtf file for each alignment
     - Merges all GTF files into a combined GTF for each genome using Cuffmerge
       - will create GTF file for each genome in ./RefList

4. Sort each resulting alignment from tophat (accepted_hits.bam) using Samtools
   - must be done for each accepted_hits file. Should probably be moved to the script
   - accepted_hits files should have been generated by tophat
   - `samtools sort -n ./foldername/A/acccepted_hits.bam ./foldername/A/accepted_hits.sort`

5. count the reads in each alignment using htseq-count, and the merged GTF file for the associated genome
   - again, must be done for each output file, and should probably be moved to the main script
   - `htseq-count -q -f bam -r name -s no ./foldername/A/accepted_hits.sort.bam ./merged_asm_A/merged.gtf > ./foldername/A/htseq_count`
6. create a Map file for use in DESeq2
   - should be given data. Tells you wich genotypes are associated with which traits
   - sample map:

sampleName	fileName	DAF	Stats
250_D0_A	250_D0_A.htc	D0	resistant
66_D3_A	66_D3_A.htc	D3	resistant
123_D0_A	234_D0_A.htc	D0	Susceptible

7. Use R to run DESeq2 to analyze the data
   - Depends on what we are looking for. Will need to be adapted for each analysis
   - another helpful link
   - sample R code saved in dropbox

Vesper MATLAB Analysis

Given CLS data, remove the noise and isolate the peaks. ***Required Files:*** *CLS sample data file (.dat), CLS rawdata file (.dat)*

1. open MATLAB 2015a through the virtual computer lab
   - use the version under the Common U of S folder, not Comp Sci. It has the packages we require
2. copy over Craig's Vesper analysis MATLAB scripts onto your U of S account, so they can be accessed through the Virtual Computer Lab
   - Required Files: peakfit.m, remove_noise_f.m, vesper_Analyze_data.m
   - files can be found in the svn repo on the server, under vesper_analysis
   - copy them over to your cabinet directory in the "My Files" channel on Paws. The can be accessed through the T drive on the Virtual Computer Lab
3. copy over the sample and the raw data files
   - sample data file contains information about the plate reads
   - raw data file contains a bunch of rows of reads
4. Run vesper_Analyze_data.m
   - program will ask for sample and raw data files as input as it runs

---

Phylogenic Analysis

Attempts to determine howclosely related two genotypes are on an evolutionary scale

- creates a herirachy or a tree showing evolutionary relationships
- use Mega or Ugene

Linkage Analysis

Used to find out howmarkers are laid out on the chromosomes. Divides markers into groups, and attempts to find distances between them. ***Requirements:*** *filtercalls vcf file (.tab)*

1. Create FilterCalls VCF tab spreadsheet using GBS process
   - sample input (tab separated):

#CHROM	POS	REF	1206	1207	P1_W1	P2_LII
TGCA_scaffold_021167	11160	1AS	C/C	T/T	C/C	T/T

2. edit the data so that there are only two parents (no replicants)
3. use the OneMap_File_Generator_f2 .py script to create a OneMap input file from the sheet
   - `python OneMap_File_Generator_f2.py "./2May16_FilterCalls.tab" "ri self" "P1_WI" "P2_LJII"`
   - may throw away some bad data. This is expected
     - can change threshold by changing the parameter to the CountMissing function
   - sample output (space separated):A

data	type	ri	self
94	776	0	
*TGACv1_569471	B	A	B

4. send data file to OneMap using Onemap_Linkage.R
   - `./Onemap_Linkage.R --f2 ./2May16_FilterCalls.tab.onemap --lod 3 --maxrf 0.5`
     - lod = log of distance
     - maxrf = max recombinant frequency
   - adjust parameters until the groups are somewhat even (we don't want the majority in one giant group)
     - can use --test to make it only show groups, but not run full analysis for testing
5. results in .png and .map files for each groups

- tries to estimate the relative positions of markers on chromosomes
- results in forced and safe files. Safe limits to more likly data, forced shows all linkages
- send .map files to biologists to process