**Bioinformatics Pipeline for dRAD tag Data**

**Software you will Need**

Stacks (<http://creskolab.uoregon.edu/stacks/>)

CAP3 (<http://seq.cs.iastate.edu/cap3.html>)

BWA (<http://bio-bwa.sourceforge.net/>)

SAMtools (<http://samtools.sourceforge.net/>)

Stacks is currently available as a module on pegasus, but NOT pegasus2

CAP3 needs to be installed on your computer

BWA and SAMtools are available as modules on both pegasus and pegasus2

**Recommended Software to Run**

FastQC (<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>)

Trimmomatic (http://www.usadellab.org/cms/?page=trimmomatic)

**Other Resources**

Basic Unix Commands: <http://mally.stanford.edu/~sr/computing/basic-unix.html>

Vi (you will use vi to edit code on the cluster): <http://www.cs.colostate.edu/helpdocs/vi.html>

**Starting the Pipeline:**

Green text represents input

Blue text represents output

Purple text (followed by a $ sign) is a location, and NOT part of a command

**Steps 1-4: Create the “Consensus Reference” sequences based on the Tags**

**Steps 5-7: Align the individual rad tag data back to the “Reference” genome**

**Steps 8-11: Call the SNPs for each individual and merge across all individuals**

**1.** Use process\_radtags from the Stacks package to sort the raw data into individual files based on barcode. Create a directory for the output (Pools\_Dir) and a file with a list of the barcode sequences (5bp.pools). Then, specify the restriction enzyme used to cut by the barcode site (-e ecoRI), the type of quality score used in the raw data file (this will depend on the machine used to sequence the data, in this case it is –E phred33). Retain reads with a single error in the barcode or cut-site, and correct the error (-r); remove reads where the quality threshold drops below the cut-off (-q 10) in a given window size (-w .11). This will also remove reads with a low quality mate (for paired end data).

Single End Example:

process\_radtags -f raw\_data.txt -o Pools\_Dir/ -b 5bp.pools -e ecoRI -E phred33 -r -q -w .11

Paired End Example:

process\_radtags -1 raw\_data\_1.txt -2 raw\_data\_2.txt -o Pools\_Dir/ -b 5bp.pools -e ecoRI -E phred33 -r -q -w .11

- Often, the reads from the med school will have the first base of the barcode cut off for most of the reads, so you will need to run process\_radtags once with the 5bp barcode file, and then again with the 4bp barcode file. The two files have to be run separately, since process\_radtags will not accept barcodes of different lengths. After doing this, you need to merge the sorted files together for each individual, and then trim all reads to be the same length (the example below is trimming all reads to be 94 bp):

 Pools\_Dir$ perl mergeRename.pl barcode.pools SE/PE

 Pools\_Dir$ perl trimFastq.pl Individual\_1.fq Individual\_1\_t.fq 94

**2.** After trimming ALL of the individual fastq files to be the same length, use ustacks (part of Stacks) to find all unique and nearly unique (99% identical) reads for EACH individual (I run this on the LSF cluster). We will specify the file type (-t fastq), the individual fastq file to process (-f Indiv.fq), the directory for the output file (-o Output\_Dir), and a database ID (-i 1) (Note that we will not actually use this, but the program will not run without an id specified). We will require a minimum depth of 1 (-m 1), and allow only 1 mismatch between reads to call a stack (-M 1 –N1). Finally, we will specify that we want to use 15 processors (-p 15).

ustacks -t fastq -f Indiv\_1\_t.fq -o Output\_Dir/ -i 1 -m 1 -M 1 -N 1 -p 15

 - Save only the tags.tsv file (we can ignore the snps.tsv and alleles.tsv files since we are just working on making a consensus sequence right now; we will call SNPs with a different program later), and process them such that you save the read depth for each consensus tag and convert the tag file into fasta format. Run the perl script that does this IN THE FOLDER WITH THE TSV FILES!

Tags\_Dir$ perl get\_stacks\_depth.pl

 - Now filter out reads with excessive depth (since these might be PCR duplicates instead of distinct tag sites). The example below is setting a minimum depth of 1, and a max depth of 50. Do this for EACH individual file:

Tags\_Dir$ perl filter\_by\_depth.pl Indiv\_1\_stacks.fasta 1 50 Indiv\_1\_filter.fasta

**3.** For EACH “uniqued” and depth-filtered fasta file, run CAP3 with 90% identity to identify and remove any possible paralogs within the individual files.

cap3 Indiv\_1\_filter.fasta -p 90 -o 90

 - We want to save ONLY the reads without multiple hits; these are in the cap3.singlets files. We will run a script IN THE FOLDER WITH THESE FILES to process the singlets files into correctly formatted fasta files:

Singlets\_Dir$ perl fixFasta.pl

**4.** Create the consensus “reference” sequence. Start by combining all of the “fixed” individual fasta files into 1 file using the unix command: cat:

cat \*.fixed.fasta >Combined.fasta

 - Run ustacks again to get the necessary tsv files for creating the catalog, then run cstacks on the ustacks output. For cstacks, we will specify a batch number to be incorporated into the names of all cstacks output files (-b 604), an output directory for the cstacks files (-o Output\_Dir), a maximum of 10 mismatches between tags (-n 10), the prefix for all of the ustacks output file (this can include the path; -s Ustacks\_prefix), the database ID number from the ustacks run (-S 601) and we will use 45 processors (-p 45):

ustacks -t fasta -f Combined.fasta -o /Output/ -i 101 -m 1 -M 10 -p 15

 (This will take about 24 hours-best to run on the cluster)

cstacks -b 604 -o /Output\_Dir/ -n 10 -s Ustacks\_prefix -S 101 -p 45

 (This should run pretty quickly, but still best on cluster)

 - Again, we will save only the tags file from the cstacks output (and not the SNPs or alleles files. The final step is to convert the tags file into a fasta file:

perl tags2fasta.pl catalog.tags.tsv Consensus.fasta

**5.** Align the individual pooled data (created in Step 1) to the new consensus “reference” sequence using BWA, a common and widely used alignment program for next-gen data. This is best done on the cluster so that all individuals can be aligned simultaneously. Here is an example of the steps for bwa (note that indexing of the reference only needs to be done ONCE).

-Single-End example:

bwa index -p ref\_prefix -a is Consensus.fasta

bwa aln -n 10 ref\_prefix Indiv\_1.fq >Indiv\_1.sai

bwa samse ref\_prefix Indiv\_1.sai Indiv\_1.fq >Indiv\_1.sam

-Paired-End example

bwa index -p ref\_prefix -a is Consensus.fasta

bwa aln -n 10 ref\_prefix Indiv\_1.fq\_1 >Indiv\_1.sai\_1

bwa aln -n 10 ref\_prefix Indiv\_1.fq\_2 >Indiv\_1.sai\_2

bwa sampe ref\_prefix Indiv\_1.sai\_1 Indiv\_1.sai\_2 Indiv\_1.fq\_1 Indiv\_1.fq\_2 >Indiv\_1.sam

**6.** Use samtools to process the SAM format files and get the mapped reads as well as the site-by-site pileup files. Note that the reference sequence only needs to be indexed once!

samtools faidx Ref.fasta

samtools view –bt Ref.fasta.fai Indiv\_1.sam >Indiv\_1.bam

samtools sort Indiv\_1.bam Indiv\_1\_sorted

samtools mpileup -f Ref.fasta Indiv\_1\_sorted.bam >Indiv\_1.pileup

samtools view -F4 Indiv\_1\_sorted.bam >Indiv\_1\_mapped.sam

**7.** Create a merged file that contains coverage information for every possible site in the reference “genome” using all of the pileup files (Note that this step can be a bit slow, since files are being merged one at a time).

perl makeRefpileup.pl Ref.fasta Ref.pileup

perl run\_covMerge.pl Ref.pileup /Population\_Pileup\_Dir/

 - Do this separately for each population to make it easier to keep track of them

 - Check the path to covMerge.pl in the run\_covMerge.pl script!

- The last ‘.cov’ file made should contain the combined info. for all individuals in the population.

paste Pop\_1.cov Pop\_2.cov >All\_Pops.cov

perl cleanup\_cov.pl All\_Pops.cov Final\_Cleaned.cov

- To determine the best coverage cut-offs to use, you can check the empirical distribution of coverages with the following script:

perl getCov\_dist.pl Final\_Cleaned.cov CoverageFreqs.txt

**8.** Now we will use the vcfutils and bcfutils tools in the SAMtools package to call SNPs for EACH individual. Use a quality threshold of 20 (-Q 20), and depth cut-off of 50 (-D 50), and a minimum of 1 alternate allele (-a 1). Be sure to check the path to samtools in the perl script! It is easiest to use a script that will run on an entire population directory at once:

perl runVCFutils.pl /Population\_BAM\_Dir/ Ref.fasta

- This script is running the following samtools commands:

samtools mpileup -uD –f Ref.fasta $bamfile | bcftools view -bvcg - >$bcffile

bcftools view $bcffile | vcfutils.pl varFilter -Q 20 -D50 -a 1 >$vcffile

**9.** Convert the VCF format files into “genotype” files, with homozygous and heterozygous SNP calls. These files will be made individually, but you should run the script within the directory with ALL of the VCF files:

Population\_VCF$ perl ../Scripts/vcf2genotype.pl

**10.** Determine all possible SNP sites, and then get a collective genotypes file (now with a single genotype call per individual) for all individuals at every site. Note that filterSNPs.pl is set to check for a minimum read depth of 5X, and replace genotype calls with “N” if coverage is too low. To change these settings, go to line 98 in the script, and edit where noted:

perl getSNPpositions.pl All.sitelist #Indivs /VCF\_Dir1/ /VCF\_Dir2/ … /VCF\_DirN/

perl ../Scripts/filterSNPs.pl All.sitelist All.genotypes

**11.** Finally, merge the collective genotypes file with the merged coverage file from step 7, to fill in reference base information for individuals with X’s:

sort -n All.genotypes >All\_sorted.genotypes

sort -n Final\_Cleaned.cov >All\_Pops\_sorted.cov

perl mergeSNPcov.pl All\_sorted.genotypes All\_sorted.cov All.snps #NumInd

**Running Analyses Described in the Manuscript:**

**12. Population Differentiation:**

-Calculate Minor Allele Frequencies:

perl snps2maf.pl All.snps All.maf Pop1,Pop2,…,PopN

-Categorize SNPs as Shared or Private Polymorphisms (in the paper we did this before filtering, to find the right coverage threshold, and then after filtering for final results):

perl SNP\_Categories\_Uni.pl All.maf All.cats All.counts

-Calculate Proportion Shared as Function of Pop-level:

perl shared\_v\_cov2.pl input.counts output.txt

**-**Filter based on coverage and minor allele frequency:

perl filterMAF.pl All.maf Filtered.maf MinCov MinMAF NumPops

**-**Polarize SNPs (determine if they are ancestral or derived). This script is written for the data used in the paper, and will need to be edited to work with another data set:

 -Line 22: change headers to desired population names

 -Lines 33 & 37: Currently set to assume Outgroup is in Column 5 of the MAF file; change accordingly (notice in perl, numbering starts at 0, not 1!)

 - Lines 36 & 40: Change column numbers to match populations in data set

perl getDerived.pl Filtered.maf Derived.maf

**-**Calculate the Derived Frequency Spectrum (again, this script will need to be modified to accommodate different populations):

 - Line 19: currently 3 hashes defined; 1 per pop, add or remove hashes as needed

 - Lines 100, 103, 106: Check that the column number is correct for populations in MAF file

 - Line 116: Edit to print out the right populations

perl AFS.pl Derived.maf AFS\_bins.txt

**-**Calculate site-by-site Fst:

perl FST\_by\_MAF\_Uni.pl <infile.maf> <outfile.fst> <npops>

**-**Run Fst permutation test. Check paths to scripts within permuteFst.pl!

perl permuteFst.pl <input.snps> <output.perm> <num\_iterations> <coverage\_cutoff> <MAF\_minimum>

**-**Split the SNP file into different populations or subsets. This script was used to get within population Fst, but could really be used to get a subset of the .snps file for any reason. In the below example, the output will be a new snps file with individuals 1-7 from population 1 in group 1, and individuals 8-10 from population 1 in group 2. The code “0.0” refers to “pop1.indiv1.” To get individual 4 from population 3, use the code “2.3”. The first number is pop, the number after the decimal is individual. Note that Perl number starts at 0, not 1! After subsetting the snps file, the same scripts for calculating MAF and Fst were re-run to get the new results.

perl subsetSNPs.pl All.snps Subset.snps 0.0,0.1,0.2,0.3,0.4,0.5,0.6 0.7,0.8,0.9

**-**Create a STRUCTURE input file

perl snps2structure.pl <input.snps> <output.structure> <pop\_list> <names.file> <min\_coverage> <min\_maf>

 pop\_list = A comma delimited list of numeric codes for populations, starting at 0, so ‘0,1,2’ refers to populations 1-3 in the .snps file

 names.file = A text file containing all individual names for every population. Each LINE must correspond to a single population, and consist of a comma-delimited list of individual names. Note that the naming format must meet STRUCTURE’s requirements. Below is an example for 3 populations with 4 individuals each:

 FO3048,FO3162,FO3171,FO3402

 FU0261,FU0262,FU0278,FU0286

 MA3607,MA3626,MA3656,MA3665

**-**Convert snps format to hapmap. The hapmap format is another text format, with rows as positions and columns as individuals. It is very similar to the .snps file, but is designed to be read by the program TASSEL (<http://www.maizegenetics.net/#!tassel/c17q9>), which has several useful features and analytical functions. Among other things, TASSEL will output in various file formats used by other software, including Plink.

perl snps2hmp.pl All.snps Out.hmp names.file minCov minMAF Haploid/Diploid

 names.file = The same kind of names file used for Structure script

 minCov = Population level coverage required for keeping a SNP (integer)

 minMAF = minimum overall MAF for including a SNP (given as decimal)

 Ploidy = Specify Haploid or Diploid (or just H or D); this affects the OUTPUT, not input. If you specify haploid, the output will have 2 lines per individual instead of 1, such that t here are no heterozygous SNP codes

**13. Species Tree**

-Create a nexus format file from the SNP data (2 steps). Step 1 gets a list of sites from the SNP file that meet the requirements of SNAPP (i.e. no missing data) The “names.file” is the same format described above. Step 2 converts those sites to Nexus format:

perl getSitelist.pl All.snps Filtered.list names.file

perl snps2nexus.pl Filtered.list Filtered.nex

-Sub-sample the nexus file (i.e. get a random set of individuals from each population to reduce computation time). The list of sample sizes should be a comma delimited list of integers; desired sizes should be given in the same order of the populations in the file (e.g. 3,3,3,4 means get 3 individuals each from pops 1,2, and 3, and take 4 individuals from pop 4):

perl subsample\_nex.pl Filtered.nex Sample.nex <List of Sample Sizes>

**14. ABBA-BABA test**

- Perform the ABBA-BABA test using the filtered list of SNPs with no missing data. This script will do the following: 1) randomly pick 4 individuals (1 from each population/species), 2) Determine if the site is informative, and classify the SNP as ABBA or BABA, 3) count the total number of ABBA and BABA SNPs and use this to calculate the D-statistic, 4) report a D-statistic for each sub-sampling run with bootstrap values (optional) to determine significance

perl ABBA\_BABA.pl Filtered.list Out.dstats Num\_subsamples Num\_Bootstrap

Num\_subsamples = The number of sub-sampling steps to perform

Num\_Bootstrap = The number of bootstrap reps to perform (if desire to calculate Z-score; this is not actually something we did in the manuscript, but has been described by others)

.

**15. Migration Analyses**

- Create the SNPs input format file specific to dadi. This script assumes the last population in the SNPs file can be treated as an outgroup, and will filter for SNPs fixed in this group. “PopList” is a comma delimited list of population names as you want them to appear in the dadi file.

perl snps2dadi.pl All.snps dadiSNPs.txt <PopList>

- Following the dadi manual instructions, make the frequency spectrum files as follows (as done in iPython). The remaining dadi analyses are performed in the Python scripts provided separately:

ipython -pylab

dd = dadi.Misc.make\_data\_dict('dadiSNPs.txt')

fs = dadi.Spectrum.from\_data\_dict(dd, ['POP1','POP2','POP3'], [24,24,24])

fs.to\_file("dadiSNPs.fs")

- Create an IMa2 format input file:

perl makeLoci.pl Filtered.list Ref.fasta IMa2.loci

perl loci2IM.pl IMa2.loci IM.infile <pops> <num.loci> <pop sizes> <seq.length> <mutation model> <inheritance scalar>

**-**Create a Migrate input file

perl im2migrate.pl IM.infile Migrate.infile (Locus format) OR

perl makeRADmigrate.pl All.snps Migrate.infile (“old” SNP format)

**-**To create LAMARC input files, it is possible to use TASSEL to export into Phylip format, which is readable by LAMARC