**Wednesday July 20 2016**

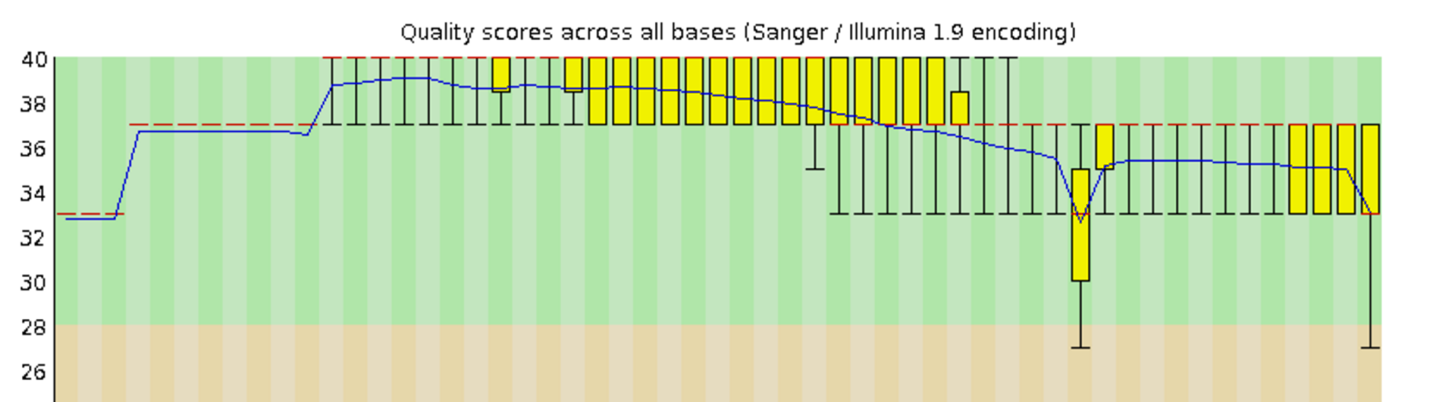
1. Obtaining quality information for reads for Decodon and Rhinanthus projects
   1. Using fastqc
      1. I ran this on the command line:
         1. for f in \*fastq.gz; do fastqc $f -o ../fastqc ; done &

-note this assumes you already have a folder called fastqc that is in the same directory as the folder of your raw reads

* + 1. I also made a custom shell script to run fastqc on all files:
       1. This script is housed: /home/hpc3461/shell\_scripts/automate\_fastqc.sh
       2. To make executable:
          1. Make sure the following line is the first line of the script !#/bin/bash
          2. Export PATH=$PATH:/appropriate/directory (typically you want $HOME/bin)
          3. Make it executable:

chmod u+x scriptname

* 1. Using my own custom script.
     1. This was basically practice at coding and provides a bit of information regarding the illumina phred scores and how they convert to quality scores.
     2. The Q scores are defined as a property that is logarithmically related to the base calling error probability P.
        1. Q=-10log10P
        2. For example if Phred assigns Q score of 30 this is equivalent to the probability of an incorrect base call 1 in 1000 times.

There seems to be something odd with the qual scores of the reads from Rhinanthus 

NEXT STEPS:

1. Assess whether adaptors sequence needs to be trimmed. I think it might already be removed.

**HOW TO RUN STACKS:**

I worked on this starting Thurs Jul 21. After speaking with Anna T, she suggested there are problem with running stacks when one has variable barcode lengths. After much reading and playing with stacks I discovered this in fact is not an issue. As of stacks 1.22 variable length barcodes are now supported and "will automatically be trimmed to keep stacks a uniform length with the variable barcode lengths”. Currently on the server Stacks 1.35 is running (type which process\_radtags to get version)

Here is the link to the update: Stacks 1.22 - Dec 08, 2014

<http://catchenlab.life.illinois.edu/stacks/changelog.php>

**Setting up Stacks:**

1. -First you need to make the folder for raw\_reads, barcodes and samples. I took a .txt file (saved in excel) and then replaced the weird ^M character using the following:

:%s/^M/\r/g

to make the weird ^M character: hit ctrl-v and enter to make ^M

1. Run process radtags:

These are the first attempts at getting stacsk to work. They were wrong and have subsequently been deleted. I worked on this Friday July 22 2016. (To remove directories that are not empty use rm –rf)

process\_radtags -p ./raw\_reads/ -b ./barcodes/barcodes.txt -o ./samples/ -e pstI -e mspI -c -q -r -i gzfastq

**Run 2:**

process\_radtags -f ../raw\_reads/\* -b ./barcodes/8bp\_barcode.txt -o ./samples/ --renz\_1 pstI --renz\_2 mspI -c -q -r -i gzfastq –D&

**Final Run: This is the “good run”**

hpc3461@swlogin1$ process\_radtags -p ../raw\_reads/ -o ./samples/ -b ./barcodes/barcodes.txt --renz\_1 pstI --renz\_2 mspI -E phred33 -r -c -q -i fastq&

**Notes on stacks:**

**Google User group posts that are useful:**

Variable length barcodes:

<https://groups.google.com/forum/embed/#!topic/stacks-users/37Fq_ucVwRE>

<https://groups.google.com/forum/#!topic/stacks-users/HFzGbDANyTI>

Processing reads:

Trim length

<https://groups.google.com/forum/#!searchin/stacks-users/trim/stacks-users/Kijc_-D3aro/0GdgT4_IyVwJ>

Check the length of reads:

for i in \*.fq; do OUTPUT="$(cat "$i"| grep '^[ACTG]' | awk '{print length}'| sort -nr | uniq -c )" ; echo "$OUTPUT $i" >> read\_length\_count\_by\_sample2.txt; done&

**I found one sample actually has a strange read length:**

1 140

1589253 92

1 91

1 86 2014NkSML2605.fq

Let’s print out the short and long reads to see what is wrong with those:

This command pulls out reads that length does not equal 92 and prints the line number, length and read. Note that you have to +1 and \*4 -4 to get the line number in the original fq file.

grep '^[ACGT]' 2014NkSML2605.fq | awk 'length!=92 {print FNR "\t" length "\t" $0}'

119 86 TGCAGGTGGCGTTCCTGTGAAGATGTGTGCGGCCTGAGCTGAATATTTAGTCTGTGTGTATGAGGTCGGTGGCTCCTTGTAGAGCT

277 91 TGCAGGAACTGCACCAGCCTCACCGACCTATCCACCAAAGGGTCTCCGTAATCGCCTTTAACGTAAACCTTGGGCAGCTTGTAAACGCGAC

1385172 140 TGCAGCAGTAAGATGAGATGATCCATTCATATATATATATAAATCATTTTGTTTGTTTGTTCGAAAAAATGGTAATCATATATATACACAACFFFFFFFFBBBFFFBBBFFFFFFBBBB<BBBFBFFBBBFBBBBFFFBF

Seems there is a glitch when removing some reads. It seems it wither didn’t remove this read or there is some error in the original fastq file.

1.quality is a bit low on some bp at 33

awk 'NR>475 && NR <480' 2014NkSML2605.fq

@1\_1105\_16916\_2801\_1

TGCAGGTGGCGTTCCTGTGAAGATGTGTGCGGCCTGAGCTGAATATTTAGTCTGTGTGTATGAGGTCGGTGGCTCCTTGTAGAGCT

+

FFF

2, quality ok but more 33 bp towards end

awk 'NR>1105 && NR<1111' 2014NkSML2605.fq

@1\_1105\_15687\_3973\_1

TGCAGGTACATAAACTGCTTGAATAGAAGTTATGGACCCTTCTTTGGTAGAAGTTATTCTTTCTTGTAAAGAGCCCATTTCGGTACTCAGGG

+

FFFFFFIBFFIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIFIIIIIFIIIIIIIIIIIIIIFFIIIIFFFFFFFFFFFFFFFFFF

3.

Original reads that got filtered incorrectly:

1.this read was cut to 86 bp and the line of qualities seems to be cut between 2 lines in the .fq file

@SN916:365:C8PF9ACXX:1:1105:16916:2801 1:N:0:1

GAATTCATGCAGGTGGCGTTCCTGTGAAGATGTGTGCGGCCTGAGCTGAATATTTAGTCTGTGTGTATGAGGTCGGTGGCTCCTTGTAGAGCTTCGCATG

+

BBBFFFFFFFFFFFFIIIFIIIIIFIIIIIIIIIIIIIIIIIIFIIIIBFIFIIIIIIIIIFFFFFFFFFFFBBBBFBFFFFFFFFFFFFFBBFBFFFFB

2.

TUESDAY:

I resubmit the process radtags just incase the issue with the one sample file was some writing error on the server. The output is in stacks\_run5\_jul2016/

process\_radtags -p ../raw\_reads/ -o ./samples/ -b ./barcodes/barcodes.txt --renz\_1 pstI --renz\_2 mspI -E phred33 -r -c -q -i fastq&

After rerunning stacks the third error where the quality scores were concatenated to the read resolved, but there are 2 reads with 86 and 91 for length. I think I should just remove these.

**Tuesday July 26, 2016**

Rather than worrying about removing two reads, I am going to process with the pipeline in stacks.

1. Denovo\_map.pl:
   1. I am using denovo\_map.pl for population genomic analysis. For sets of populatoins it Is recommended to supply a population “map” just a file that species which samples belong to which populations. In the population analysis all individuals are loaded into the catalog. The pipeline finishes by running a program called populatoins which calculates genomic statistics like heterozygosity pi and Fis. Can also get pairwise Fst. Populations can be re-executed after the pipeline runs with a variety of command line options to filter the data.
   2. De\_novo\_map.pl takes in Fastq or fasta file formats. Compressed or uncompressed. It will first execute ustacks to assemble loci in each individual de novo calling SNPs in each assembled locus. Then it executes cstacks to build the catalog followed by sstacks to match all samples against the catalog.

How to make population map for stacks:

The file needs to look like this: the pops can be numeric or string.

% more popmap

indv\_01 fw

indv\_02 fw

indv\_03 fw

indv\_04 oc

indv\_05 oc

indv\_06 oc

I wrote a little python script “~/python\_scripts/parsing\_pop\_map\_for\_stacks.py” that takes the barcodes.txt file and uses that to make a file of the indiv names and pop names.

!. wrote shell script to run ustacks:

vi ustacks\_script.sh

1. Submit job using
   1. qsub ustacks\_script.sh
   2. checked job using qsub –j 1645242

**Wed July 27, 2016**

Had some minor bugs in the .sh script. I fixed these (took a silly amount of time to find typos.)

submit the job: 1645451

Checked and it is processing the reads.

It might be better to just run the wrapper, but I didn’t see an example for the population analysis with the denovo wrapper. Also, Anna T warned me that we have limited memory on server so it might be best to run stacks in chunks. I also want to know what each step in the pipeline does so I’m running them separately.

\*\*\*This afternoon/evening I found a mistake. It seems that the same sample was sent twice. Sample “2014NkSML2605” was sent as sample #12 and sample #60. See the “Rhinathus\_Set\_form\_sequence\_en.xls" that is housed here:

/Users/Maggie/Dropbox/Dave Chris Maggie/Laval

Something is wrong after processing reads I have 95 samples not the 96 I am expecting:

cut -f 2 barcodes.txt | sort | uniq

make list of barcoded samples:

less ../barcodes/barcodes.txt |cut -f 2| sort |uniq >barcoded\_samples.temp

make list of processed files;

ls \*.fq > samples\_list\_processed

remove .fq in vi

%s/.fq//g

Because we want to keep the two identical samples separate I changed the name of the first duplicated sample to 2014NkSML2605TGCGA. Basically I stuck the barcode to the end of the sample name.

I am rerunning the process\_reads again using this new list of samples and barcodes.

|  |
| --- |
| process\_radtags -p ../raw\_reads/ -o ./samples/ -b ./barcodes/barcodes\_2014NkSML2605T\_separated.txt --renz\_1 pstI --renz\_2 mspI -E phred33 -r -c -q -i fastq& |

Interestingly, this is the same sample that had a few reads whose length !=92

Strangely, making sure the samples are separate now changes the previous results of finding strange read length for this sample 2014NkSML2605. Now all demultiplexed samples have reads of length 92 bp.

Running ustacks:

I have the scripts to process the stacks pipeline in this folder:

/home/hpc3461/rhinanthus/stacks\_pipeline

Run ustacks\_script.sh:

qsub ustacks\_script.sh

Job ID: 1647053

The output of ustacks is comprised of 3 files per sample (.alleles.tsv, .tags.tsv and .snps.tsv) and is located in folder called “stacks”. Error files and process updates are output to “ustacks.err”. This output stats like :

Min depth of coverage to create a stack: 3

Max distance allowed between stacks: 4

Max distance allowed to align secondary reads: 6

Max number of stacks allowed per de novo locus: 3

Deleveraging algorithm: enabled

Removal algorithm: enabled

Model type: SNP

Alpha significance level for model: 0.05

Parsing ../stacks\_run6\_jul2016/samples/2012HBLo01.fq

….

0 reads contained uncalled nucleotides that were modified.

Mean coverage depth is 64; Std Dev: 757.749 Max: 128820

Coverage mean: 64; stdev: 757.749

Deleveraging trigger: 822; Removal trigger: 1579

Calculating distance for removing repetitive stacks.

Distance allowed between stacks: 1

Using a k-mer length of 45

Number of kmers per sequence: 48

Minimum number of k-mers to define a match: 3

Removing repetitive stacks.

Removed 4414 stacks.

74897 stacks remain for merging.

Calculating distance between stacks...

Distance allowed between stacks: 4

Using a k-mer length of 17

Number of kmers per sequence: 76

Minimum number of k-mers to define a match: 8

Merging stacks, maximum allowed distance: 4 nucleotide(s)

74897 stacks merged into 57321 stacks; deleveraged 736 stacks; removed 1720 stacks.

Mean merged coverage depth is 88.1136; Std Dev: 981.743; Max: 139248

Merging remainder radtags

598368 remainder sequences left to merge.

Distance allowed between stacks: 6

Using a k-mer length of 13

Number of kmers per sequence: 80

Minimum number of k-mers to define a match: 2

Ustacks: takes as input short-read sequences (the demultiplexed samples) and align them into exactly-matching stacks. Comparing the stacks it will form a set of loci and detect SNPs at each locus using max likelihood. -1Hohenlohe PA, Bassham S, Etter PD, Stiffler N, Johnson EA, Cresko WA. 2010.*Population genomics of parallel adaptation in threespine stickleback using sequenced RAD tags.* **PLoS Genetics**, [6(2):e1000862](http://dx.doi.org/doi:10.1371/journal.pgen.1000862).

See this part of manual for explanation of ustacks:

### Heading:What do the fields mean in *Stacks* output files? outputhttp://catchenlab.life.illinois.edu/stacks/manual/index.php#files

I’m now waiting for ustacks to finish running. Once that is done I can run cstacks.

I’m confused about some of the output of ustacks. In the alleles file the haplotypes are given. It seems strange to me that some loci have multiple haplotypes reported. I’m not sure if these are loci that are then recorded as unknown or what the consequences of that are.

I must check whether the “strange” loci with multiple haplotypes is also called as an unknown genotype at that locus.

Then run cstacks.

Notes on what ustacks did:

Ustacks constructs loci and alleles for individual samples. It does so in 2 steps. First, using exact matches it creates loci that are 92 bp in length. Why it constructs stacks to this length I don’t understand. It splits the reads into primary and secondary reads. Primary reads that are placed into stacks in the first pass, secondary reads are not discarded just set aside temporarily. The minimum stack depth parameter (-m default is set to 3 in ustacks) and I believe it only refers to depth of primary reads to initialize a locus. It’s recommended to increase this parameter if one has high sequencing depth (although what high is is not defined) or if one has high sequencing error. If there is sufficiently high sequencing error that yields identical reads then it is likely we will se convergent sequencing of PCR errors (errors that occur independently at the same nucleotide position in the same read).

We may want to increase the value of –m from 3 to some higher number and see how the number of loci changes. (Note I did so on Thurs July 28, 2016 see notes below for that day)

After the initial loci are created the second stage of this algorithm tried to match putative alleles together into a locus. The distance allowed between stacks parameter represents the number of nucleotides that may be different between two stacks in order to merge them. The default (-M 2). If set too low then some loci will fail to merge. So what should be one locus will appear as 2. On the flip side if it is set too high then too many repetitive sequences will be chained together. It’s recommended to use trial and error to see how many polymorphic loci one can construct. A selfer has neutral genome wide diversity of somewhere between 7 in 1000 or 1 in 100 bp differences. So I think the parameter value here of 2 should be about right. Depending on the level of diversity we see we may consider increasing this to parameter. I could probably figure out what I expect the range of snps should be in a 92 bp stack.

I wanted to find out how many loci were built for each sample:

In the .snps.tsv file the 3rd column in the locus id. So I pull out the last line of the file and pull out the 3rd column of the last line and print that out along with the file name.

for f in \*.snps.tsv; do tail -1 $f |cut -f 3 && echo -n " " $f " "; done

This send the counts of loci and file name to a new file:

for f in \*.snps.tsv; do tail -1 $f |cut -f 3 |tr '\n' " " >>loci\_counts\_by\_sample.txt && echo $f " " >>loci\_counts\_by\_sample.txt; done

Wrote a quick little python script to get min and max loci counts:

file=open("loci\_counts\_by\_sample.txt", 'r')

my\_list=[]

for line in file:

sline=line.split()

my\_list.append(int(sline[0])

my\_list.sort()

min(my\_list)

max(my\_list)

Min loci=16120

Max loci= 84493

**Thursday July 28, 2016**

CSTACKS: merges alleles together to construct a catalog

In this folder: /home/hpc3461/rhinanthus/stacks\_pipeline

I submit: qsub cstacks\_script.sh

I made sure to update the sample list within the script to account for the new name of that duplicated sample. Now I’m waiting for this to run. I also need to figure out how the next step uses the population info because the rstacks program doesn’t seem to take the population list as an input.

Redo-Ustacks:

Because it is recommended to increase min read depth when one is suspicious of high sequencing error or when one has high depth I changed the ustacks script so that the minimum number of reads is changed from –m 3 to –m 6. I can then compare the number of loci between the two runs to try to determine which is better.

Your job 1647731 ("ustacks\_script.sh") has been submitted

Notes on what Cstacks does and it’s ouput:

Cstacks merges alleles from different samples together. The output is a

For each sample a .matches.tsv file is created. This shows how sample locus matches to a catalog locus.

**Monday August 1, 2016**

Update from Dave about sample 2014NkSML2605.:

|  |
| --- |
| Hi Maggie,  Looks like you're right, it was sent twice, my mistake! I am very confident about the sample IDs being correct, so I looked into the Good\_Samples\_19April2016.xlsx file, and it is also listed in there twice. My best guess is that it was used as a replacement during extractions, despite having already been done. One final check would be to check the microtubes in the -80 freezer; they are samples 12 and 60 (Both in box 1).  Sorry about the mistake, might be interesting to see if they come out very similar down stream. A kind of double check if we need a silver lining! |

Summary of what’s been done to date:

1. process\_radtags: output is located in /home/hpc3461/rhinanthus/ stacks\_run6\_jul2016
2. ustascks:
   1. script “/home/hpc3461/rhinanthus/stacks\_pipeline /ustacks\_script.sh”
   2. output:
3. cstacks:
   1. script: /home/hpc3461/rhinanthus/stacks\_pipeline /
   2. output: 2 outputs
      1. /home/hpc3461/rhinanthus/stacks\_pipeline/stacks

This output is using default parameters of ustacks

* + 1. /home/hpc3461/rhinanthus/stacks\_pipeline/stacks\_m\_6

This output is for a ustacks run which uses a higher number of minimum number of reads is changed from –m 3 to –m 6

There is a difference between the two ustack runs. With a higher minimum number of reads required there are fewer loci per sample. We lose approximately 30 of loci when we require there to be a minimum of 6 reads per loci per sample. So for now I am proceeding with the default parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| Sample ID | number of loci using default parameters | number of loci when minimum number of reads is 6 | Proportion of loci |
| 2012HBLo01.snps.tsv | 56085 | 40427 | 0.72 |
| 2012HBLo02.snps.tsv | 57321 | 36565 | 0.64 |
| 2012HBMd01.snps.tsv | 36616 | 25295 | 0.69 |
| 2012HBMd02.snps.tsv | 18960 | 15346 | 0.81 |
| 2014KdHi1103.snps.tsv | 28106 | 22996 | 0.82 |
| 2014KdHi1402.snps.tsv | 31346 | 25016 | 0.80 |
| 2014KdHi2604.snps.tsv | 18084 | 14812 | 0.82 |
| 2014KdLo1803b.snps.tsv | 20791 | 16549 | 0.80 |
| 2014KdLo2402b.snps.tsv | 17861 | 15070 | 0.84 |
| 2014KdLo2502c.snps.tsv | 32777 | 20911 | 0.64 |
| 2014KdMH0405.snps.tsv | 17631 | 14739 | 0.84 |
| 2014KdMH2305.snps.tsv | 19502 | 16273 | 0.83 |
| 2014KdMH3005.snps.tsv | 21134 | 17057 | 0.81 |
| 2014KdML0105.snps.tsv | 26945 | 20838 | 0.77 |
| 2014KdML0401.snps.tsv | 27326 | 20607 | 0.75 |
| 2014KdML2605.snps.tsv | 19852 | 16077 | 0.81 |
| 2014NkNHi0705.snps.tsv | 20128 | 16237 | 0.81 |
| 2014NkNHi1205.snps.tsv | 21415 | 17551 | 0.82 |
| 2014NkNHi1903.snps.tsv | 22789 | 17871 | 0.78 |
| 2014NkNLo0605.snps.tsv | 20617 | 16467 | 0.80 |
| 2014NkNLo0801.snps.tsv | 30875 | 26647 | 0.86 |
| 2014NkNLo2802.snps.tsv | 21076 | 17183 | 0.82 |
| 2014NkNMH1005.snps.tsv | 23960 | 17383 | 0.73 |
| 2014NkNMH1401.snps.tsv | 20781 | 16967 | 0.82 |
| 2014NkNMH1702b.snps.tsv | 20910 | 16457 | 0.79 |
| 2014NkNML0504.snps.tsv | 20836 | 16531 | 0.79 |
| 2014NkNML0702.snps.tsv | 22580 | 17134 | 0.76 |
| 2014NkNML0902.snps.tsv | 27229 | 20688 | 0.76 |
| 2014NkSHi0203.snps.tsv | 20576 | 16701 | 0.81 |
| 2014NkSHi0505.snps.tsv | 19752 | 16239 | 0.82 |
| 2014NkSHi2201.snps.tsv | 18332 | 15274 | 0.83 |
| 2014NkSLo1001c.snps.tsv | 20987 | 17026 | 0.81 |
| 2014NkSLo1702.snps.tsv | 21105 | 17059 | 0.81 |
| 2014NkSLo2604.snps.tsv | 21603 | 17093 | 0.79 |
| 2014NkSMH0104.snps.tsv | 21239 | 17454 | 0.82 |
| 2014NkSMH1801.snps.tsv | 23257 | 18023 | 0.77 |
| 2014NkSMH2901.snps.tsv | 19741 | 16040 | 0.81 |
| 2014NkSML2605.snps.tsv | 34448 | 21427 | 0.62 |
| 2014NkSML2605TGCGA.snps.tsv | 21143 | 17247 | 0.82 |
| 2014NkSML2901.snps.tsv | 20115 | 16516 | 0.82 |
| 2015APKHi08.snps.tsv | 24391 | 14905 | 0.61 |
| 2015APKHi23.snps.tsv | 26568 | 17168 | 0.65 |
| 2015APRMd09.snps.tsv | 23022 | 16067 | 0.70 |
| 2015APRMd20.snps.tsv | 22594 | 16307 | 0.72 |
| 2015KdHi0302b.snps.tsv | 20428 | 16656 | 0.82 |
| 2015KdHi0703.snps.tsv | 19887 | 16354 | 0.82 |
| 2015KdHi0905.snps.tsv | 22083 | 17730 | 0.80 |
| 2015KdLo1501b.snps.tsv | 21373 | 16930 | 0.79 |
| 2015KdLo2004d.snps.tsv | 26576 | 19522 | 0.73 |
| 2015KdLo2702c.snps.tsv | 21183 | 17222 | 0.81 |
| 2015KdMH07A04.snps.tsv | 26863 | 20446 | 0.76 |
| 2015KdMH1104.snps.tsv | 18042 | 15203 | 0.84 |
| 2015KdMH2204.snps.tsv | 21483 | 16484 | 0.77 |
| 2015KdML0801.snps.tsv | 20349 | 15896 | 0.78 |
| 2015KdML0905.snps.tsv | 33639 | 23858 | 0.71 |
| 2015KdML2905.snps.tsv | 20512 | 16542 | 0.81 |
| 2015KobHi01.snps.tsv | 20623 | 16443 | 0.80 |
| 2015KobHi14.snps.tsv | 18210 | 15198 | 0.83 |
| 2015NkNHi0904.snps.tsv | 21026 | 17077 | 0.81 |
| 2015NkNHi1205.snps.tsv | 26132 | 18194 | 0.70 |
| 2015NkNHi2403.snps.tsv | 25062 | 19867 | 0.79 |
| 2015NkNLo0302.snps.tsv | 33497 | 27640 | 0.83 |
| 2015NkNLo2103b.snps.tsv | 22477 | 17087 | 0.76 |
| 2015NkNLo2805.snps.tsv | 28128 | 20461 | 0.73 |
| 2015NkNMH0202.snps.tsv | 22607 | 18112 | 0.80 |
| 2015NkNMH1001.snps.tsv | 26362 | 19914 | 0.76 |
| 2015NkNMH1801b.snps.tsv | 21737 | 17319 | 0.80 |
| 2015NkNML0902b.snps.tsv | 21236 | 16567 | 0.78 |
| 2015NkNML1402.snps.tsv | 29126 | 20336 | 0.70 |
| 2015NkNML2105.snps.tsv | 30012 | 21629 | 0.72 |
| 2015NkSHi0705c.snps.tsv | 21349 | 16509 | 0.77 |
| 2015NkSHi1502.snps.tsv | 22264 | 17513 | 0.79 |
| 2015NkSHi2201b.snps.tsv | 22610 | 17073 | 0.76 |
| 2015NkSLo3702.snps.tsv | 22337 | 17565 | 0.79 |
| 2015NkSLo3802b.snps.tsv | 22860 | 16909 | 0.74 |
| 2015NkSLo4001.snps.tsv | 26796 | 19855 | 0.74 |
| 2015NkSMH0302.snps.tsv | 84493 | 51654 | 0.61 |
| 2015NkSMH1203.snps.tsv | 26950 | 18262 | 0.68 |
| 2015-NkS-MH-20-5.snps.tsv | 42256 | 25578 | 0.61 |
| 2015NkSML1301b.snps.tsv | 52184 | 27195 | 0.52 |
| 2015NkSML2703b.snps.tsv | 22093 | 17106 | 0.77 |
| 2015NkSML3001b.snps.tsv | 23374 | 17541 | 0.75 |
| 2015PhoMH20.snps.tsv | 19061 | 15440 | 0.81 |
| 2015PhoMH21.snps.tsv | 16120 | 13727 | 0.85 |
| 2015SPHi04.snps.tsv | 23249 | 17395 | 0.75 |
| 2015SPHi17.snps.tsv | 35682 | 21910 | 0.61 |
| 2015SPLo15.snps.tsv | 30300 | 22489 | 0.74 |
| 2015SPLo16.snps.tsv | 24538 | 19578 | 0.80 |
| 2015SPMd03.snps.tsv | 20593 | 16542 | 0.80 |
| 2015SPMd23.snps.tsv | 20315 | 16287 | 0.80 |
| 2015SSHi13.snps.tsv | 35253 | 23251 | 0.66 |
| 2015SSHi19.snps.tsv | 19302 | 15655 | 0.81 |
| 2015SSLo02.snps.tsv | 21174 | 16473 | 0.78 |
| 2015SSLo09.snps.tsv | 25053 | 19397 | 0.77 |
| 2015SSMH22.snps.tsv | 23055 | 18122 | 0.79 |
| 2015SSML06.snps.tsv | 29048 | 17641 | 0.61 |
|  |  | Min | 0.52 |
|  |  | Max | 0.86 |
|  |  | Avg | 0.77 |

NEXT STEPs:

1. Sstacks: sstacks links up catalog locus to the within sample locus. The output file is for each sample

I thought I submit sstacks last week but I don’t have a record for it so I’m submitting the process now.

Your job 1652416 ("sstacks\_script.sh") has been submitted

In this folder:

/home/hpc3461/rhinanthus/stacks\_pipeline

command

qsub sstacks\_script.sh

1. rxstacks\_cstacks\_script.sh.

In this folder: /home/hpc3461/rhinanthus/stacks\_pipeline

Command

qsub rxstacks\_cstacks\_script.sh

Rxstacks applies 4 corrections to the earlier SNP calls:

1. It reevaluates SNP calls that were not significant in each individual. It uses the catalog to look at alleles for a particular nucleotide position and filter out putative sequencing errors from the individual under examination. It them recalls the nucleotide position as a heterozygotes or homozygote using the bounded SNP model.
2. Applies a log likelihood filtering. When a stack is assembled a log likelihood is assigned to that stack by ustacks. This removes loci that have log likelihood below the threshold. Loci with poor log likelihoods have low coverage, high sequencing error or both. Good likelihoods are close to zero and bad are highly negative. What is highly negative I don’t know.
3. For each catalog in the locus individuals are expected to have a single matching locus. When multiple loci from an individual match a catalog locus are called “confounded”. These loci are typically comprised of repetive sequence. These are removed from the analysis.
4. Haplotype pruning.

3. Submit populations for the population file with the years.

In folder:

/home/hpc3461/rhinanthus/stacks\_pipeline

Command:

qsub stacks\_populations\_script.sh

Tuesday August 2, 2016

Minor typo in populations script:

Submit populations for the population file with the years.

In folder:

/home/hpc3461/rhinanthus/stacks\_pipeline

Command:

qsub stacks\_populations\_script.sh

Fudge:

1. Ustacks is find because I did not pass list of files
2. Cstacks\_script fixed, checked that sample number is correct
   1. head -n 110 cstacks\_script.sh | tail -96 |wc –l
3. fixed sstacks-removed this script because it was part of the cstacks script as well.
4. rxstacks, was good didn’t need fixing.
   1. head -n 110 rxstacks\_cstacks\_script.sh | tail -n +15 | uniq -c
   2. head -n 110 rxstacks\_cstacks\_script.sh | tail -n +15 | wc -l

Just to be on the safe side I’m redoing the following

Cstacks+ sstacks combined script:submit 1150am tues

Rxstacks: submit at 530pm

Populations

Once these are done… finish pipeline for the ustacks samples where 6 reads were required in order to call a sample locus. The populations output for the default parameters seems to have a high number of loci that are rejected because they are not biallelic.