########## The Stacks Pipeline ##########

###Written/compiled by Jermaine Mahguib
###Graduate Student, Iowa State University
###Spring 2018

###The following is a comprehensive summary of the Stacks pipeline for use on GBS SNP data to accomplish two main things:

#1. To use replicate sample data to optimize a core set of Stacks parameters, and determine the utility of secondary options within the pipeline

#2. To process whole data sets using the pipeline and the identified optimal parameter settings.

###The information in this summary is broken up into two SECTIONS. The first section will introduce each component program of the Stacks pipeline by showing, for each program, an example script for running it and a piece-by-piece breakdown of each option of the script. The second section will outline which programs to use, and in what order, to accomplish #1. and #2. above.

SECTION ONE

#***Disclaimer: while some of the options shown as part of each example script are necessary for running the script, not all of them are; which options you utilize will depend on what you want to accomplish in running a particular program. Info on each component of the pipeline can also be reviewed here at http://catchenlab.life.illinois.edu/stacks/; scroll down the page to the

"Pipeline components" section for a list of links which will each direct you to a page providing more detailed information on each component of the pipeline***#

###Stacks program: process_radtags

###"This program examines raw reads from an Illumina sequencing run and first, checks that the barcode and the RAD cutsite are intact, and demultiplexes the data. If there are errors in the barcode or the RAD site within a certain allowance process_radtags can correct them. Second, it slides a window down the length of the read and checks the average quality score within the window. If the score drops below 90% probability of being correct (a raw phred score of 10), the read is discarded. This allows for some seqeuncing errors while elimating reads where the sequence is degrading as it is being sequenced. By default the sliding window is 15% of the length of the read, but can be specified on the command line (the threshold and window size can be adjusted)." {Quoted directly from http://catchenlab.life.illinois.edu/stacks/comp/process_radtags.php}

###Note: process_radtags must be run on each group of samples by Index used, from each population/location that samples were taken

###Example script for running process_radtags as it should be entered into the terminal:

```
process_radtags -p ./jmahguib/Raw_GBS_Data/TrkNbr-1083_I12/ -o ./jmahguib/Stacks_Process_RadTags_outputs/TrkNbr-1083_I12/ -b ./jmahguib/Barcodes/Barcodes_I12.txt -e pstI -E phred33 -r -c -q -D -t 90
```

###Breakdown of the components of the above script for running Process_Radtags (in the order that the components of the script appear, starting in this case with the 'process_radtags' component, then the '-p ./jmahguib/Raw_GBS_Data/TrkNbr-1083_I12/' component, etc.), with annotations:

process_radtags
###Command to initiate the process_radtags program within the Stacks module

-p ./jmahguib/Raw_GBS_Data/TrkNbr-1083_I12/

```
###Instructs the program to pair the files within the specified directory

-o ./jmahguib/Stacks_Process_RadTags_outputs/TrkNbr-1083_I12/
###Tells the program to put output files into the specified directory

-b ./jmahguib/Barcodes/Barcodes_I12.txt
###Specifies a file path to the directory containing the 'Barcodes' file for this
   particular run of the program; 'Barcodes' files need to be constructed in advance
   by hand and made available for use by process_radtags (in other words, 'Barcodes'
   file must be uploaded to the Speedy server)

-e pstI
###Specifies the restriction enzyme that was used to generate the data

-E phred33
###Tells the program how to encode the quality scores, based on how sequencing was
   carried out ('phred33' [Illumina 1.8+, Sanger, default] or 'phred64' [Illumina 1.3
   - 1.5])

-r
###Instructs the program to "rescue barcodes and RAD-Tags"

-c
###Instructs the program to "clean data" and "remove any read with an uncalled base"

-q
###Option to "discard reads with low quality scores"

-D
###Tells the program to "capture discarded reads to a file"

-t 90
###Specifies to the program to "truncate final read length" to the value given
```

###Stacks program: ustacks

###"The unique stacks program will take as input a set of short-read sequences and align them into exactly-matching stacks (or putative alleles). Comparing the stacks it will form a set of putative loci and detect SNPs at each locus using a maximum likelihood framework." {Quoted directly from http://catchenlab.life.illinois.edu/stacks/comp/ustacks.php}

###Example script for running ustacks as it should be entered into the terminal:

```
#!/bin/bash
files="WM10.fq WM12.fq WM14.fq WM16.fq WM18.fq WM1.fq WM21.fq WM23.fq WM2.fq WM4.fq WM6.fq WM8.fq
WM11.fq WM13.fq WM15.fq WM17.fq WM19.fq WM20.fq WM22.fq WM24.fq WM3.fq WM5.fq WM7.fq WM9.fq"
#
i=1
for file in $files
do
ustacks -i $i -f /work/LAS/kjroe-lab/Cyprogenia_Dromus_practice_238/${file} -m 3 -M 2 --
max_locus_stacks 3 --model_type snp --alpha 0.05 -d -o ./jmahguib/UStacks_outputs/
Cg_practice_full_dataset -t fastq -p 15
let "i+=1";
done
```

###Breakdown of the components of the above script for running ustacks, with annotations:

```
#!/bin/bash
files="WM10.fq WM12.fq WM14.fq WM16.fq WM18.fq WM1.fq WM21.fq WM23.fq WM2.fq WM4.fq
WM6.fq WM8.fq WM11.fq WM13.fq WM15.fq WM17.fq WM19.fq WM20.fq WM22.fq WM24.fq WM3.fq
WM5.fq WM7.fq WM9.fq"
#
i=1
for file in $files
do

###'files=' is used to list the file names you want the program to read and process
```

```
ustacks
###Command to initiate the ustacks program within the Stacks module

-i $i
###Specifies a "unique integer ID to identify this sample;" this is related to the bit
   of code employed above (from "#!/bin/bash" to "do") and basically tells the program
   to run on the first sample in the "files=" list, then on the second sample, etc...

-f ./jmahguib/Stacks_Process_RadTags_outputs/TrkNbr-1083_UStacks_input/${file}
###This specifies the file path input files for this program; again, the way this
   option is set up here, with that "${file}" at the end of the file path, is related
   to that bit of code above that's used to tell the program to run through the input
   files in the "files=" list one after the other

-m 3
###Specifies to the program the "minimum depth of coverage required to create a stack"
   (the default setting is 2)

-M 2
###This setting specifies the "maximum distance (in nucleotides) allowed between
   stacks" (the default setting is 2)

--max_locus_stacks
###This option specifies the "maximum number of stacks at a single de novo locus" (the
   default setting is 3; if no number is explicitly specified for the option, then the
   default is used)

--model_type snp
###Sets the SNP-calling model for the analysis, either the 'snp' model (as shown in
   this example) or the 'bounded' model

--alpha
###Option to specify the "chi square significance level required to call a
   heterozygote or homozygote, either 0.1, 0.05 (default), 0.01, or 0.001" (if no
   number is explicitly specified for the option, then the default is used)
```

```
-d
###Tells the program to enable the "deleveraging algorithm, used for resolving over
   merged tags"

-o ./jmahguib/UStacks_outputs/
###Specifies a directory for the program to write its output to

-t fastq
###Species the file type of the input files to be used for running the program
   (supported types: fasta, fastq, gzfasta, or gzfastq)

-p 15
###Option to "enable parallel execution" with a specified number of computational
   threads

let "i+=1";
###When you want to add additional data to continue a UStacks analysis, change "i+= "
   to the next number in the output folder; for example, if you analyze 24 samples the
   first analysis, and you want to add another 24, change "i+=1" to "i+=25", and if
   you wanted to add another 24 after that, change to "i+=49"
```

###Stacks program: cstacks

###"A catalog can be built from any set of samples processed by the ustacks program. It will create a set of consensus loci, merging alleles together." {Quoted directly from http://catchenlab.life.illinois.edu/stacks/comp/cstacks.php}

###Cstacks builds a reference catalogue that is used later in the pipeline to match your samples to; while all of your samples for a particular study can be used to build this catalogue, it is not necessary to do so. You can build your catalogue using just a portion of your samples, in which case you would want to select a few samples per population (and pick samples with the largest file sizes, or in other words, samples that have the most information); you would also want all populations in your study represented evenly. The cstacks program can take a huge amount

of time to complete if you run it with a large number of samples, so if you want to save time it is a valid option to use fewer samples for this step in the pipeline. A useful command for looking at ustacks output files listed from largest to smallest file size is 'ls -S -lh'; file size can be used here as a proxy for selecting which samples have the most reads.

###Example script for running cstacks as it should be entered into the terminal:

```
cstacks -b 1 -n 3 -p 15 -o ./jmahguib/CStacks_TGCP_10_outputs -s ./jmahguib/UStacks_outputs/WM2 -
s ./jmahguib/UStacks_outputs/WM14 -s ./jmahguib/UStacks_outputs/WM20 -s ./jmahguib/
UStacks_outputs/WM24 -s ./jmahguib/UStacks_outputs/WM12 -s ./jmahguib/UStacks_outputs/WM22 -s ./
jmahguib/UStacks_outputs/WM23 -s ./jmahguib/UStacks_outputs/WM11 -s ./jmahguib/UStacks_outputs/
WM19 -s ./jmahguib/UStacks_outputs/WM16
```

###Breakdown of the components of the above script for running cstacks, with annotations:

cstacks
###Command to initiate the cstacks program within the Stacks module

-b 1
###Specifies a "database/batch ID for this catalog" (the default ID is 1; you're going to want to think about using an ID code specific to each population/location)

-n 3
###This option tells the program the "number of mismatches allowed between sample loci when build the catalog" (the default setting is 1)

-p 15
###Option to "enable parallel execution" with a specified number of computational threads

-o ./jmahguib/CStacks_TGCP_10_outputs
###This is where you want cstacks to put it's output files into

-s ./jmahguib/UStacks_outputs/WM2
###This is describing the file path to your ustacks output files, which cstacks needs

```
   to read in order to build a catalog; you need this command for each sample you're
   going to use in the analysis, as shown below

-s ./jmahguib/UStacks_outputs/WM14
-s ./jmahguib/UStacks_outputs/WM20
-s ./jmahguib/UStacks_outputs/WM24
-s ./jmahguib/UStacks_outputs/WM12
-s ./jmahguib/UStacks_outputs/WM22
-s ./jmahguib/UStacks_outputs/WM23
-s ./jmahguib/UStacks_outputs/WM11
-s ./jmahguib/UStacks_outputs/WM19
-s ./jmahguib/UStacks_outputs/WM16
```

###Stacks program: sstacks

###"Sets of stacks, i.e. putative loci, constructed by the ustacks program can be searched against a catalog produced by cstacks. [...] all samples in the population [are] matched against the catalog with sstacks." {Quoted directly from http://catchenlab.life.illinois.edu/stacks/comp/sstacks.php}

###Note: sstacks requires output files from both ustacks AND cstacks, so you should put output files from ustacks and cstacks into the same directory before proceeding!

###Example script for running sstacks as it should be entered into the terminal:

```
#!/bin/bash
files="WM1 WM2 WM3 WM4 WM5 WM6 WM7 WM8 WM9 WM10 WM11 WM12 WM13 WM14 WM15 WM16 WM17 WM18 WM19 WM20 WM21 WM22 WM23 WM24"
#
for file in $files
do
sstacks -b 1 -c ./jmahguib/UStacks_outputs/batch_1 -s ./jmahguib/UStacks_outputs/$file -o ./jmahguib/UStacks_outputs -p 15
done
```

###Breakdown of the components of the above script for running sstacks, with annotations:

```
#!/bin/bash
files="WM1 WM2 WM3 WM4 WM5 WM6 WM7 WM8 WM9 WM10 WM11 WM12 WM13 WM14 WM15 WM16 WM17
WM18 WM19 WM20 WM21 WM22 WM23 WM24"
#
for file in $files
do

###'files=' is used to list the file names you want the program to read and process


sstacks
###Command to initiate the sstacks program within the Stacks module

-b 1
###Specifies a "database/batch ID for this catalog" (the default ID is 1; you're going
   to want to match the ID used here to what you used in CStacks)

-c ./jmahguib/UStacks_outputs/batch_1
###Tells SStacks to read the CStacks output files

-s ./jmahguib/UStacks_outputs/$file
###Tells SStacks to read the UStacks sample output files

-o ./jmahguib/UStacks_outputs
###This is where you want SStacks to put the output files in

-p 15
###Option to "enable parallel execution" with a specified number of computational
   threads
```

###Stacks program: 'populations'

###"The populations program will analyze a population of individual samples computing a number of population genetics statistics as well as exporting a variety of standard output formats. A population map specifying which individuals belong to which population is submitted to the program and the program will then calculate population genetics statistics such as expected/observed heterozygosity, π, and FIS at each nucleotide position. The populations program will compare all populations pairwise to compute FST. If a set of data is reference aligned, then a kernel-smoothed FST will also be calculated. The populations program can also compute a number of haplotype-based population genetics statistics including haplotype diversity, ΦST, and FST'." {Quoted directly from http://catchenlab.life.illinois.edu/stacks/comp/populations.php}

###Example script for running the 'populations' program in Stacks, as it should be entered into the terminal:

```
populations -P ./jmahguib/RXCSStacks_outputs/Cg_practice_cat33_full_dataset_corr -O ./jmahguib/Stacks_Populations_outputs/Cg_practice_cat33_full_dataset -M ./jmahguib/PopMaps/Cyprogenia/popmap_cg_dr_232_ind.txt -b 1 -p 10 -r 0.85 -t 8 --phylip_var --write_single_snp
```

###Breakdown of the components of the above script for running populations, with annotations:

```
populations
###Command to initiate the populations program within the Stacks module

-P ./jmahguib/RXCSStacks_outputs/Cg_practice_cat33_full_dataset_corr
###Species the file path to the directory containing the Stacks files that will be
   used

-O ./jmahguib/Stacks_Populations_outputs/Cg_practice_cat33_full_dataset
###Specifies to the program what directory to put the output files into

-M ./jmahguib/PopMaps/Cyprogenia/popmap_cg_dr_232_ind.txt
###Path to a directory containing a population map (PopMap); a PopMap is a separate
   file you have to set up yourself (a plain text file will do) that specifies what
   group each of your samples belongs to, and should be tab delimited
```

```
        -b 1
        ###Designates a batch identification number that is used to label output files

        -p 10
        ###Designates a minimum number of "groups" a locus must be present in to process
           the locus; "groups" can be populations, shared geographical regions, species, etc.,
           depending on what's needed for your study, and are designated by you within the
           PopMap file you must manually create prior to use of the 'populations' program

        -r 0.85
        ###Specifies a minimum percentage of individuals in a population required to process a
           locus for that populations; this will be variable depending on your needs

        -t 8
        ###Thread count for running parallel sections of code

        --phylip_var
        ###Option for specifying to the program to include variable sites in the phylip output
           encoded using IUPAC notation

        --write_single_snp
        ###Option to restrict the analysis to only use the first SNP in each locus


###Example of the contents of a tab-delimited PopMap file used for running the 'populations'
program; the first column lists the name of each sample to be included in the analysis, and the
second column designates what "group" each sample belongs to (e.g. population, river basin,
geographical region, species, etc.):

WM1     TGCP
WM2     TGCP
WM3     TGCP
WM4     TGCP
WM5     TGCP
WM6     TGCP
```

```
WM7     TGCP
WM8     TGCP
WM9     TGCP
WM10    TGCP
WM61    BHP
WM62    BHP
WM63    BHP
WM64    BHP
WM65    BHP
WM66    BHP
WM67    BHP
WM68    BHP
WM69    BHP
WM70    BHP


###Example notes you should record (copy & paste) from the Terminal after each run of the
'populations' program (scroll up in the terminal window to find these info), for your later
reference:

        ###With '-r' set to 70%...

           Populating observed haplotypes for 24 samples, 514411 loci.
           Removing 404802 loci that did not pass sample/population constraints... retained
           109609 loci.
           89821 loci

        ###With '-r' set to 85%...

           Populating observed haplotypes for 24 samples, 514411 loci.
           Removing 449317 loci that did not pass sample/population constraints... retained
           65094 loci.
           52138 loci

        ###With '-r' set to 100%...
```

```
Populating observed haplotypes for 24 samples, 514411 loci.
Removing 480583 loci that did not pass sample/population constraints... retained
33828 loci.
26175 loci
```

SECTION TWO

###Stacks Parameter Optimization

###In the first part of this section, I outline a method for testing for optimal parameter settings in Stacks based on Mastretta-Yanes et al (2015). The "Stacks program: denovo_map" subsection below should be performed using a subset of samples from your study for which replicate GBS data was generated; in other words, choose a subset of samples from your study, extract DNA from each sample, and then perform your GBS protocol (from PCR to sequencing) TWICE on each sample, so that you end up with a set of data comprised of sample replicate pairs. You should generate replicate data for at least 8 individuals, netting you a replicate dataset of 16 samples (two samples from each individual), although doing more than 8 individuals would be better (the more individuals you use for this, the more accurate your estimates of SNP error rates will be down the line as you progress through your optimization process). Before proceeding, it is HIGHLY RECOMMENDED that you read through Mastretta-Yanes et al (2015) to gain a basic understanding of how you will be testing the different parameter sets (described below) based on SNP error rate using a custom R script (provided in the paper's Supplementary Information), as I only outline the details of this optimization process that are concerned with the use of Stacks and do not detail how to use the R script to estimate SNP error rates here.

###Stacks program: denovo_map

###The denovo_maps program within the Stacks package allows you to run all the components of the Stacks pipeline in one command script; this is particularly useful for running multiple runs of

the pipeline while trying to test for the best parameter set to use for your full-dataset run of the Stacks pipeline. This is the program you will use to generate SNPs for your sample replicates (from across random populations of your total specimen sampling) using variations in the Stacks parameters "-m," "-M," "--max_locus_stacks," and "-n," keeping all parameters set to their default values while adjusting the values of only one of the parameters at a time. The following indicates the default values that will be used, as well as the value ranges you might want to test for each parameter:

-m (specifies the minimum number of identical raw reads required to create a stack)
Default value: 3
Range for testing: 2 - 10

-M (specifies number of mismatches allowed between loci when processing a single individual)
Default value: 2
Range for testing: 2 - 8

--max_locus_stacks (maximum number of stacks at a single de novo locus)
Default value: 3
Range for testing: 2 - 6

-n (specifies number of mismatches allowed between loci when building the catalog)
Default value: 0
Range for testing: 0 - 7

###An Example script for running denovo_map as it should be entered into the terminal is shown below. Before proceeding in the use of this script, it is CRITICAL that you read and understand the use of the '-b' option of the script; I can not stress enough how valuable understanding the '-b' option will be to you in this process, as it will allow you to differentiate outputs from the multiple runs of the denovo_map program you will be performing. Being able to differentiate these outputs will be necessary when you go to run the R script from Mastretta-Yanes et al (2015) to estimate SNP error rates for the outputs generated from each combination of the core parameter set...

```
denovo_map.pl -m 3 -M 2 -X "ustacks:--max_locus_stacks 3" -n 0 -T 15 -S -b 3230 -t -o ./jmahguib/
Stacks_Denovo_Map_outputs/Cg_practice_8replicates/ -X "populations:--plink --vcf" -s ./jmahguib/
```

```
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_09.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_09_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10_r.fq
```

###Breakdown of the components of the above script for running denovo_map, with annotations:

```
        denovo_map.pl
        ###Command to initiate the denovo_map program within the Stacks module

        -m 2
        ###Indicates the minimum number of identical raw sequence reads needed to create a
           stack

        -M 2
        ###Tells the program how many mismatches are allowed between loci when processing one
           individual specimen

        -X "ustacks:--max_locus_stacks 3"
        ###The '-X' option in denovo_map allows for a specific component of the Stacks
           pipeline to be given a specific value for one of its options; in the example script
           here, '-X' is being used to specify to the UStacks component of the pipeline
           (indicated by "ustacks: " where the parenthesis are required for the option to
```

```
   function) to set the option '--max_locus_stacks' to specified values. Notice that
   unlike the normal way of specifying options in programs by putting a dash in front
   of the signifying letter for the option (such as '-m'), when specifying options for
   a specific program within the '-X' option in denovo_map, you have to use two dashes
   before the option signifier (such as '--m')

-n 1
###Sets the number of mismatches allowed between loci when building the catalog of
   loci

-T 15
###Specifies the number of threads to use (I'm not really clear on what this means;
   this isn't a value I'll be playing around with)

-S
###Used to disable SQL data in the database (whatever that means)

-b 3230
###This sets the batch ID number for the dataset; the number you assign here will be
   tagged onto the output files for each run of the script. This is a useful tool for
   labeling your output files so that they don't override previous output files that
   were generated and saved to the same directory using program default names. In this
   example, I used the batch ID number '3230' to signify the run that tested parameter
   combination of '-m' (3), '-M' (2), '--max_locus_stacks' (3), and '-n' (0)

-t
###This option instructs the pipeline to remove/break up highly repetitive RAD-Tags
   (done so in the UStacks portion of the pipeline)

-o ./jmahguib/Denovo_Map_outputs/Cg_practice_8replicates/
###This option designates the directory that output files from the pipeline will be
   deposited into

-X "populations:--plink --vcf"
###Option for specifically instructing the Populations program of the Stacks pipeline
   to generate output files in both 'plink' and 'vcf' formats
```

```
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_09.fq
        ###This option describes the file path to your raw sequence data files, which
           denovo_map needs to read in order to run the pipeline. You need to add this option
           once for every input file you have for the pipeline, as shown below...

        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_09_r.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12_r.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15_r.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16_r.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06_r.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08_r.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09_r.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10.fq
        -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10_r.fq

###...run this script once for every combination of the core parameter set. In other words, run
this script with...

        '-m' set to 2
        '-M' set to default (2)
        '--max_locus_stacks' set to default (3)
        '-n' set to default (0)

###...then run the script again, with...

        '-m' set to 3
        '-M' set to default (2)
        '--max_locus_stacks' set to default (3)
        '-n' set to default (0)
```

###...then run it again and again, until you've varied '-m' all the way up to 10. Once you've finished varying '-m', run the script again, but this time vary '-M'...

'-m' set to default (3)
'-M' set to 3
'--max_locus_stacks' set to default (3)
'-n' set to default (0)

###...keep performing additional runs of the script until you've varied '-M' up to 8. Then repeat the whole process again, varying '--max_locus_stacks', and then again for varying '-n'. Once you have all of your output files from all of these different runs of denovo_map, you can then use each set of outputs (representing each combination of the core parameter set, which you've designated with unique batch ID numbers using the '-b' option) to run the R script from Mastretta-Yanes et al (2015) and estimate SNP error rates for your replicate samples based on what parameter set was used to process the data. For each run of the denovo_map script that you performed, you have to run the SNP error rate estimation R script on the output data. Additionally, each time that you run the R script, you need to identify the estimated SNP error rate for each sample within the output that is generated; record these SNP error rate estimations in an Excel spreadsheet, remembering to label everything accordingly so that you can keep track of what estimations belong to which samples, processed using which combination of the Stacks core parameters.

###After testing the above four parameters (-m, -M, -mls and -n) and determining which set results in the lowest average SNP error rate, there are four more options that should be added to the above denovo_map script and tested (using, of course, the optimal values for -m, -M, -mls and -n determined previously): --model_type ('snp' model vs 'bounded' model, implemented in UStacks; for 'bounded' model, option '--bound_high' should be included and varied); --min_maf (minimum minor allele frequency, implemented in Populations program); --write_single_snp (option in the Populations program to restrict data analysis to only the first SNP per locus); rxstacks (another program in the Stacks package that makes corrections to genotype and haplotype calls in individual samples based on data accumulated from a population-wide examination). However, unlike with -m, -M, -mls and -n, testing --model_type, --min_maf, --write_single_snp and the rxstacks corrections should be done one after the other, rather than in sets, and they should be tested in the order they're presented here:

--model_type ('snp' (default model) or 'bounded' model)
When testing 'bounded' model, use option --bound_high and vary its value...
Values for testing: 0.05, 0.1, 0.2 and 0.3

--min_maf (specifies a minimum minor allele frequency required to process a nucleotide site at a locus)
Value for testing: 0.05
This is a "with or without" test; don't need to test a range of values

rxstacks (makes corrections to genotype and haplotype calls in individual samples)
*From this point, you will discontinue using the denovo_map program when seeking to test utilization of rxstacks and the '--write_single_snp' option in the populations program, because rxstacks must be run after sstacks but before the populations program. Furthermore, after running rxstacks, you need to re-run cstacks and sstacks using the output files from rxstacks (from this point, I will refer to the post rxstacks runs of cstacks and sstacks as 'cstacks_corr' and 'sstacks_corr', where 'corr' denotes that the programs were run using the 'corrected' data from rxstacks. Testing use of the rxstacks program must be done by manually running each program in the Stacks pipeline because there's no option to run rxstacks and subsequently re-run cstacks and sstacks afterward within the denovo_map program.

--write_single_snp (restricts data analysis to only the first SNP per locus)
*Once you've finished determining whether it's better to use or not use the rxstacks corrections on your data, run the populations program (on either the 'corrected' replicate data or the original replicate data), and see what results in a lower SNP error rate. This is a "with or without" test; no specific values to test

###Example scripts for running denovo_map as it should be entered into the terminal for testing model type, minimum minor allele frequency, rxstacks correction and single SNP per locus calling (REMEMBER, these should be tested one after the other in the above listed sequence, and the resulting lowest SNP error rate in each test should inform the options you use for the next test):

```
#For testing model type...

denovo_map.pl -m 3 -M 2 -X "ustacks:--max_locus_stacks 3 --model_type snp" -n 3 -T 15 -S -b
323300 -t -o ./jmahguib/Stacks_Denovo_Map_outputs/Cg_practice_8replicates/ -X "populations:--
plink --vcf" -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_09.fq -s ./
jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_09_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10_r.fq

#...then...

denovo_map.pl -m 3 -M 2 -X "ustacks:--max_locus_stacks 3 --model_type bounded --bound_high 0.05"
-n 3 -T 15 -S -b 323305 -t -o ./jmahguib/Stacks_Denovo_Map_outputs/Cg_practice_8replicates/ -X
"populations:--plink --vcf" -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/
CLR_09.fq -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_09_r.fq -s ./
jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08.fq -s ./jmahguib/
```

```
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10_r.fq

#...remember to test the 'bounded' model multiple times, varying the values of the '--bound_high'
option (example: 0.05, 0.1, 0.2 and 0.3)


#For testing minimum minor allele frequency (remember to set the model type to whichever model
gave you the lowest SNP error rate in the previous test above)...

denovo_map.pl -m 3 -M 2 -X "ustacks:--max_locus_stacks 3 --model_type SNP" -n 3 -T 15 -S -b
32330005 -t -o ./jmahguib/Stacks_Denovo_Map_outputs/Cg_practice_8replicates/ -X "populations:--
min_maf 0.05 --plink --vcf" -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/
CLR_09.fq -s ./jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_09_r.fq -s ./
jmahguib/Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_12_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_15_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CLR_16_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_06_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_08_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_09_r.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10.fq -s ./jmahguib/
Stacks_Process_RadTags_outputs/Cg_practice_8replicates/CSP_10_r.fq


#For running the rxstacks correction (remember to set the model type to whichever model gave you
the lowest SNP error rate in the previous test above)...
```

```
rxstacks -b 1 -P ./kkim/cgdr_out_238/ -o ./kkim/cgdr_out_238_corr_snp/ --conf_lim 0.25 --
prune_haplo --model_type snp --lnl_lim -10.0 -t 8


        #Breakdown of the components of the above script for running rxstacks, with
        annotations:

                rxstacks
                #Command to initiate the rxstacks program within the Stacks module

                -b 1
                #Designates a batch identification number that is used to label output files

                -P ./kkim/cgdr_out_238/ -o ./kkim/cgdr_out_238_corr_snp/
                #Specifies the file path to the directory containing the ustacks, cstacks and
                   sstacks output files for the samples you want to run the corrections on

                --conf_lim 0.25
                #Sets a value (between 0.0 and 1.0; default value 0.75) that determines the
                   proportion of loci in a population that must be confounded relative to the
                   catalog locus

                --prune_haplo
                #An option that will remove non-biological haplotypes unlikely to occur in the
                   population

                --model_type snp
                #Sets the SNP-calling model for the analysis, either the 'snp' model (as shown
                   in this example) or the 'bounded' model

                --lnl_lim -10.0
                #Sets a minimum log likelihood required to keep a catalog locus

                -t 8
                #Specifies the thread count for running parallel sections of code
```

#For testing option to call first-SNP only (manually use the populations program for this test, and have it use as input whichever data set yielded a lower average SNP error rate when you tested use of the rxstacks corrections above; in other words, use as input either the rxstacks corrected data, or the non-corrected data, depending on which gave a lower SNP error rate)...

```
populations -P ./jmahguib/RXCSStacks_outputs/Cg_practice_cat33_full_dataset_corr -O ./jmahguib/Stacks_Populations_outputs/Cg_practice_cat33_full_dataset -M ./jmahguib/PopMaps/Cyprogenia/popmap_cg_dr_232_ind.txt -b 1 -p 10 -r 0.85 -t 8 --phylip_var --write_single_snp
```

#...remember to adjust the '-p' and '-r' options based on your specific needs for your study data

###Running the complete Stacks pipeline on your full dataset...

###After you have determined what the optimal values are of the core Stacks parameters (-m, -M, -mls and -n) and the additional options (--model_type, --min_maf, rxstacks, and --write_single_snp) for your data, you will be ready to run the entire Stacks pipeline manually (meaning run each component of the pipeline individually, rather than using denovo_map) on your full dataset. Below I will outline each program you should run (in the appropriate order they should be run), providing again an example script for each, as well as relevant notes indicating considerations that should be kept in mind as you progress through the pipeline.

###1. process_radtags

###Run this program on all of your raw GBS data to demultiplex it. Remember to change the file paths in the script to lead to your own directories, as well as adjust any of the options as you need them to be...

```
process_radtags -p ./jmahguib/Raw_GBS_Data/TrkNbr-1083_I12/ -o ./jmahguib/Stacks_Process_RadTags_outputs/TrkNbr-1083_I12/ -b ./jmahguib/Barcodes/Barcodes_I12.txt -e pstI -E phred33 -r -c -q -D -t 90
```

###2. ustacks

###Once all your data has been demultiplexed, run the ustacks program using all the process_radtags output files as input. Remember to list all of your specimen sample files as ".fq" in the 'files=' portion of the script. Also, notice that the '-m', '-M', '--max_locus_stacks' and '-model_type' options can be set in this program; make sure to set these to whatever values you found were optimal for your data in terms of reducing SNP error rate, as was determined in the Stacks parameter optimization process performed previously...

```
#!/bin/bash
files="WM10.fq WM12.fq WM14.fq WM16.fq WM18.fq WM1.fq WM21.fq WM23.fq WM2.fq WM4.fq WM6.fq WM8.fq WM11.fq WM13.fq WM15.fq WM17.fq WM19.fq WM20.fq WM22.fq WM24.fq WM3.fq WM5.fq WM7.fq WM9.fq"
#
i=1
for file in $files
do
ustacks -i $i -f /work/LAS/kjroe-lab/Cyprogenia_Dromus_practice_238/${file} -m 3 -M 2 --max_locus_stacks 3 --model_type snp --alpha 0.05 -d -o ./jmahguib/UStacks_outputs/Cg_practice_full_dataset -t fastq -p 15
let "i+=1";
done
```

###3. cstacks

###Create a catalog based on a portion of your sample data, as discussed in the cstacks portion of SECTION ONE above. Notice that the '-n' option is available in this program; make sure you set this to whatever value you found was optimal during your Stacks parameter optimization process performed previously. Cstacks uses the output files from ustacks as input files...

```
cstacks -b 1 -n 3 -p 15 -o ./jmahguib/CStacks_TGCP_10_outputs -s ./jmahguib/UStacks_outputs/WM2 -s ./jmahguib/UStacks_outputs/WM14 -s ./jmahguib/UStacks_outputs/WM20 -s ./jmahguib/UStacks_outputs/WM24 -s ./jmahguib/UStacks_outputs/WM12 -s ./jmahguib/UStacks_outputs/WM22 -s ./jmahguib/UStacks_outputs/WM23 -s ./jmahguib/UStacks_outputs/WM11 -s ./jmahguib/UStacks_outputs/WM19 -s ./jmahguib/UStacks_outputs/WM16
```

###4. sstacks

###Search your individual sample data (ustacks outputs) against the catalog you built (cstacks outputs) using this program. Remember to list all of your sample names in the 'files=' portion of the script; no file extensions are needed here, only the file names. Sstacks requires output files from both ustacks and cstacks in order to run; sstacks can be directed to those output files in different ways depending on how you decide to organize your Stacks directories. For example, below I have the sstacks script set to read my cstacks output files from one directory, and my ustacks output files from another, using the '-c' and '-s' options respectively. Or you can have '-c' and '-s' lead to the same directory if you stored your ustacks and cstacks outputs together. Another approach is to store all your ustacks and cstacks outputs in one directory and use the '-P' option to direct the program to that directory, although I have not tried this myself; I suspect that using '-P' would invalidate the need for the first block of script where you use 'files=' to designate your file names. In any case, as was previously stated above, it's worth it to think about how you want to organize your pipeline directories before getting started, in a way that best fits how you think and will allow you to understand your workflow...

```
#!/bin/bash
files="WM1 WM2 WM3 WM4 WM5 WM6 WM7 WM8 WM9 WM10 WM11 WM12 WM13 WM14 WM15 WM16 WM17 WM18 WM19 WM20
WM21 WM22 WM23 WM24"
#
for file in $files
do
sstacks -b 1 -c ./jmahguib/CStacks_outputs/Cg_practice_cat33_for_full_dataset/batch_1 -s ./
jmahguib/UStacks_outputs/New_Dir_System/Cg_practice_full_dataset/$file -o ./jmahguib/
SStacks_outputs/Cg_practice_full_dataset -p 15
done
```

###5. rxstacks

###If you determined previously that using the rxstacks corrections did NOT reduce the average SNP error rate for your data, then you can skip step #5 here (and the following steps #6 and #7) and move on to step #8 below. If, however, you determined that using the rxstacks corrections DID reduce the average SNP error rate for your data, then you will want to continue reading from

here. Rxstacks will require the output files from ustacks, cstacks AND sstacks that you have generated, so you will need to direct the program to a directory containing all of these output files together, using the '-P' option. However it is that you ultimately decided to organize your directories for the pipeline, I highly recommend that you create a separate directory for receiving the output files from rxstacks, rather than outputting them into the same directory as your other Stacks output files, otherwise the rxstacks output files will override your previous ustacks output files as they are generated...

```
rxstacks -b 1 -P ./jmahguib/UCSStacks_outputs/Cg_practice_full_dataset -o ./jmahguib/
RXStacks_outputs/Cg_practice_full_dataset --conf_lim 0.25 --prune_haplo --model_type snp --
lnl_lim -10.0 -t 8
```

###6. cstacks_corr

###After running the rxstacks corrections on your data, you will need to re-run cstacks AND sstacks on the output files from rxstacks. The rxstacks output files will essentially look like your ustacks output files (in terms of how many files you will have generated and how they are labeled), except that the data they contain will be different because it has undergone corrections. The reason step #6 here has been titled "cstacks_corr" is to designate this run of cstacks as being performed on the 'corrected' data (hence the '_corr') outputted from rxstacks. This does not change the script you will use to run cstacks in this step, other than using different file paths to lead to different directories; as was recommended above in step #5, I would consider making new directories for storing your 'cstack_corr' and 'sstack_corr' output files, rather than storing them in the same directories as your initial runs of cstacks and sstacks. Lastly, recall that the number of samples used in this program is largely up to you; you don't have to include all of your samples, but you don't want to include too few samples either. You will want to include at least a few samples from every population across your total dataset, and make sure that the samples you select from each population are your largest file-size samples as was discussed in the cstacks portion of SECTION ONE above. Also, you should use the same samples in both steps #3 and #6 (even though different samples are shown in the example scripts here)...

```
cstacks -b 1 -n 3 -p 15 -o ./jmahguib/CStacks_corr_outputs/
Cg_practice_cat33_for_full_dataset_corr -s ./jmahguib/RXStacks_outputs/Cg_practice_full_dataset/
CST_02 -s ./jmahguib/RXStacks_outputs/Cg_practice_full_dataset/CST_10 -s ./jmahguib/
```

```
RXStacks_outputs/Cg_practice_full_dataset/CST_04 -s ./jmahguib/RXStacks_outputs/
Cg_practice_full_dataset/CGE_08 -s ./jmahguib/RXStacks_outputs/Cg_practice_full_dataset/CGE_09 -
s ./jmahguib/RXStacks_outputs/Cg_practice_full_dataset/CGE_05 -s ./jmahguib/RXStacks_outputs/
Cg_practice_full_dataset/CCB_02
```

###7. sstacks_corr

###Now you will perform your second run of the sstacks program, except this time on your 'corrected' output files. To run this 'sstacks_corr' step, sstacks will require the output files from both rxstacks and 'cstacks_corr', so adjust your file paths in the script accordingly. And as in the previous two steps, it would be a good idea to output your 'sstacks_corr' files to a different directory than that of your previous Stacks outputs from steps #2 through #4 above. Also note that I used a different set of samples in the example script below than I did in the example script in step #4; I did this only to show that it is ok to have your 'files=' list written on multiple lines rather than all on one line, as long as all the sample names remain within the " " and each sample name is delimited by at least one space on each side. But in practice, the samples in steps #4 and #7 would be the same samples (aside from that the samples used here in step #7 have been 'corrected')...

```
#!/bin/bash
files="CBR_02 CBW_16 CLR_16 CSF_01 CSP_08 CST_01
CBR_03 CBW_17 CLR_17 CSF_02 CSP_09 CST_02
CBR_05 CBW_18 CLR_18 CSF_03 CSP_10 CST_03
CBR_06 CBW_19 CLR_19 CSF_04 CSP_11 CST_04
CBR_07 CCB_01 CLR_20 CSF_05 CSP_12 CST_05
CBR_08 CCB_02 CLR_21 CSF_06 CSP_13 CST_06
CBR_09 CCF_01 CLR_22 CSF_07 CSP_14 CST_07
CBR_10 CCF_02 CLR_23 CSF_08 CSP_15 CST_08
CBR_11 CCF_03 CLR_24 CSF_09 CSP_16 CST_09
CBR_12 CCF_04 CLR_25 CSF_10 CSP_17 CST_10
CBR_13 CCF_05 CLRd_09 CSF_11 CSP_18 UAUC076
CBR_14 CCW_01 CLRd_10 CSF_12 CSPd_04 UAUC1446
CBR_15 CGE_01 CLRd_11 CSF_13 CSPd_05 UAUC1499
CBR_16 CGE_02 CLRd_12 CSF_14 CSPd_06 UAUC1500
CBR_17 CGE_03 CLRd_13 CSF_15 CSPd_07 UAUC1535
```

```
CBR_18 CGE_04 CLRd_14 CSF_16 CSPd_08 UAUC1536
CBR_19 CGE_05 CLRd_15 CSF_17 CSPd_09 UAUC1647
CBR_20 CGE_06 CLRd_16 CSF_18 CSPd_10 UAUC1650"
#
for file in $files
do
sstacks -b 1 -c ./jmahguib/CStacks_corr_outputs/Cg_practice_cat33_for_full_dataset_corr/batch_1 -
s ./jmahguib/RXStacks_outputs/Cg_practice_full_dataset/$file -o ./jmahguib/SStacks_corr_outputs/
Cg_practice_full_dataset_corr -p 15
done


###8. 'populations' (for sample selection)

###Once you have completed the pipeline up to this point, you will be ready to analyze your
processed data using the 'populations' program. Initially, you will want to perform a run of the
'populations' program for the purpose of generating an output file that you can then use to
perform an additional sample selection step before re-running the 'populations' program a second
time to generate specific output file types for downstream analyses. The 'populations' program
takes the output files from ustacks, cstacks and sstacks as inputs, or in the case that you
determined that use of the rxstacks corrections on your data was appropriate, you would use the
output files from rxstacks, 'cstacks_corr' and 'sstacks_corr' as inputs for the 'populations'
program. Note that you will want to include the '--write_single_snp' option here if you
previously determined that its use resulted in a lower SNP error rate for your data.
Additionally, note that you can control which loci are processed in this analysis based on what
portion of the samples they are shared across, using the '-p' parameter. In the example script
below, 232 samples were included in the analysis and each sample was designated as its own
"group" in the PopMap file that the '-M' option directs the program to; therefore, setting '-p'
to 162 (as in the example below) tells the 'populations' program to process only loci shared by
70% of the samples included in the analysis (162/232 = 0.7 or 70%)...

populations -P ./kkim/cgdr_snp_out_238_corr/ -O ./kkim/cgdr_snp_out_238_corr_232_ind_p160_SinSNP
-M ./kkim/popmap/popmap_cg_dr_232_ind.txt -b 1 -p 162 -t 8 --phylip_var --write_single_snp

###...the content of the resulting '.phylip' output file will be a series of consensus sequences,
each sequence being comprised of all the SNPs recovered from each individual included in the
```

analysis. Each consensus sequence is a concatenation of only the SNPs found in its respective sampled individual, and contain no other non-SNP nucleotide positions. The contents of this output file can be copied into Excel, where formulas can be used to count the number of undetermined SNP positions in each consensus sequence (the number of 'N' positions in the sequence). Recall that the '--phylip_var' option specifies the inclusion of variable sites in the '.phylip' output file which are encoded using IUPAC notation; positions in each sequence that are designated 'N' are undetermined positions for that sequence relative to all other sequences. In other words, 'N' positions represent missing data. The reason you would want to count how many 'N' positions constitute each sequence is so that you could determine what percentage of each sequence is comprised of 'N's and then select out samples that have too much missing data (determined at your discretion). For example, you might decide that you only want to utilize sequences with less than 40% missing data, because samples with more than 40% missing data are not useful to your study; you would now be able to identify which samples meet this criteria and eliminate the samples that do not.

###9. 'populations' (for generating input files for downstream pop. genomic analyses)

###Once you have determined which samples you want to keep and which samples you want to eliminate, you can re-run the 'populations' program to generate a variety of input file formats for downstream analyses. The easiest way to direct the 'populations' program to only process the samples you've elected to keep, and ignore the samples you've determined have too much missing data, is by opening up your PopMap file, manually removing the samples you no longer want to use, and saving it as a new PopMap file for use in this additional run of the 'populations' program. Notice that in the example script below, several options have been added ('--plink', '--structure', '--genepop' and '--vcf') to instruct the program to write it's output files in additional file formats for your later use as input files for different programs used in downstream analyses. Also note that for this run of the 'populations' program, you will want to edit your PopMap file such that the second column assigns each individual (listed in the first column) to "groups" that are relevant to your study, rather than assigning each individual to its own unique group as was done for use in step #8 above. This will then alter your use of the '-p' parameter, and will likely also prompt you to want to include and make use of the '-r' parameter as well, as shown in the example below...

```
populations -P ./kkim/cgdr_snp_out_238_corr/ -O ./kkim/cgdr_corr_snp_156_4species -M ./kkim/
popmap/popmap_cg_dr_156_4species.txt -b 1 -k -p 2 -r 0.60 -t 16 --min_maf 0.05 --plink --
```

```
structure --genepop --vcf --phylip_var --write_single_snp

###...recall that the '-p' and '-r' parameters function as follows...

        -p 2
        ###Designates a minimum number of "groups" a locus must be present in to process
           the locus; "groups" can be populations, shared geographical regions, species, etc.,
           depending on what's needed for your study, and are designated by you within the
           PopMap file you must manually create prior to use of the 'populations' program

        -r 0.60
        ###Specifies a minimum percentage of individuals in a population required to process a
           locus for that population; this will vary depending on your study needs


###10. 'populations' (for generating input files for downstream phylogenomic analyses)

###Lastly, if you are interested in using your data for some phylogenomic analyses, the .phylip
output file format from 'populations' can be used for this. Both steps #8 and #9 above provide
example scripts that utilize the '--phylip_var' option to instruct the program to output
a .phylip file, but below I have outlined a scheme for generating a set of .phylip output files
(to be done after you complete step #8 and decide which samples you're going to keep) for use in
running multiple phylogenomic analyses on subsets of your data in order to test how varying
degrees of "missing data" affects the topology of your phylogenomic trees. The four scripts shown
below are the same except for the values designated for the '-p' option. This script was used to
generate .phylip output files for a dataset containing 156 individuals, and the investigator
wanted to see how excluding loci not present within a certain percentage of individuals would
affect the tree outcome; to do this, they set '-p 78' to instruct the program to only process
loci found in at least 50% of samples in the dataset (78/156 = 0.5 or 50%). They then re-ran the
'populations' program with '-p 94' to process loci present in at least 60% of the samples (94/156
= 0.6 or 60%), and so on, etc. Like in step #9 above, you will need to direct the 'populations'
program to a PopMap file (using the '-M' option); however, unlike in step #9 (where you altered
the PopMap such that each sample was designated as part of some "group" relavent to your study),
here you will want to use a PopMap file where each sample is designated as its own group, like
the PopMap file you used to accomplish step #8 above (except minus the samples you filtered out
in that step)...
```

```
###Process only loci shared by 50% of selected samples...

populations -P ./kkim/cgdr_snp_out_238_corr/ -O ./kkim/cgdr_corr_snp_156_ind_p78_SinSNP -M ./
kkim/popmap/popmap_cg_dr_156_ind.txt -b 1 -p 78 -t 8 --phylip_var --write_single_snp


###Process only loci shared by 60% of selected samples...

populations -P ./kkim/cgdr_snp_out_238_corr/ -O ./kkim/cgdr_corr_snp_156_ind_p94_SinSNP -M ./
kkim/popmap/popmap_cg_dr_156_ind.txt -b 1 -p 94 -t 8 --phylip_var --write_single_snp


###Process only loci shared by 70% of selected samples...

populations -P ./kkim/cgdr_snp_out_238_corr/ -O ./kkim/cgdr_corr_snp_156_ind_p109_SinSNP -M ./
kkim/popmap/popmap_cg_dr_156_ind.txt -b 1 -p 109 -t 8 --phylip_var --write_single_snp


###Process only loci shared by 80% of selected samples...

populations -P ./kkim/cgdr_snp_out_238_corr/ -O ./kkim/cgdr_corr_snp_156_ind_p125_SinSNP -M ./
kkim/popmap/popmap_cg_dr_156_ind.txt -b 1 -p 125 -t 8 --phylip_var --write_single_snp




#####And with that, you have completed a basic run of the Stacks pipeline; congratulations! Now
you get to realize that all the work you just went through was simply to prime you for whatever
extensive suite of downstream analyses you need to perform for your study (hurray...). Now go
figure out what you want to do with your shiny, newly processed data. Like a motivational poster
of a kitten dangling from a clothesline would tell you, "hang in there!"
```

#####The end ^_^