

Supplementary Material:

Nangura spinosa deriving genotypes scripts

Nicholas Bail

2024-06-11

Contents

Background	2
Study system	2
Information on the sequencing run	2
Intent	2
Procedure	3
Preparing the input files	3
Create directories	3
Add read files	3
Visually assessing the sequencing data	3
The read structure:	3
Inspect a read:	3
Cleaning and demultiplexing the reads	4
Process radtags	4
Job resources	5
Parameter testing	5
Background	5
Stacks	6
vcftools	7
‘Bad Apples’ to remove.	9
r80 method	9
The samples that are replicates	12
Tiger	12
Inferring error rates per parameter set	12
Exporting Datasets from Stacks	13
Dataset 1	13

Samples to be removed	15
Finallising Dataset 1	16
Dataset 2	16
Dataset 3	17
References	19
Packages	19

Background

Study system

The Nangur Spiny Skink (*Nangura spinosa*) is a Critically Endangered lizard found in South East Queensland. It is restricted to two populations: ‘Oakview’ which consists of 100s of individuals as of last count and ‘Nangur’ which consists of 10s of individuals. Based on current knowledge these two populations have been isolated due to separation of habitat since before European settlement, but it is not known if they show any level of reproductive isolation. Habitat degradation, weeds, and feral animals are leading to declines in their population - especially in the smaller Nangur population. A captive population (from Oakview) has been established by Queensland Parks, and they hope to reintroduce individuals to the wild at a new site as an insurance population. So far, genetics has not informed these efforts.

Individuals from both populations and the captive population have been sampled by Queensland Parks officers (including Dan Ferguson) and they have funded sequencing of these individuals. In addition, Conrad Hoskin has contributed older samples from both natural population to assess changes in genetic diversity over time, these come from samples used for Borsboom et al (2010).

Prior to sequencing, I subsampled the tissue samples and sent a small amount to AGRF for DNA extraction. There were some issues with these samples. In particular, Oakview05 had degraded DNA. I persisted with the sequencing of these samples, but I need to be aware that they may bring issues.

Information on the sequencing run

These samples were sequenced using AGRF Genotyping-By-Sequencing method. In this method, samples are double digested, and then ligated with 48 unique barcoded adapters and up to 4 (2 were used) sequencing indexes. The library undergoes size selection and an indexing PCR. The size selection chosen was the 62bp narrow selection (selects fragments between 280-342 bp) which is aimed at having increased depth though with fewer tags. These sequences underwent single-end sequencing rather than paired-end sequencing.

The restriction enzymes used were selected using a standard establishment, and were **PstI** and **HpyCH4IV**. We expect the following restriction enzyme overhangs:

- **PstI**: 5' TGCAG 3'
- **HpyCH4IV**: 5' ACGT 3'

Intent

The samples have been sequenced at AGRF (sequencing funded by QLD Government). The data is RADseq data using AGRF's Genotyping-By-Sequencing method. I will undergo a variety of analyses to assess kinship, population structure, genetic diversity, effective population size, inbreeding, demographic history, etc. In

order to do these analyses, I need a dataset of 1000s of SNPs derived from the raw sequencing data I have been provided.

This R Markdown is intended to record the most important scripts of deriving genotypes with STACKS for *N. spinosa*, and then the final filtering to generate Dataset 1, 2, and 3. Guidance comes from Rochette & Catchen (2017), Paris (2017), Bresadola et al. (2020), and Cerca et al., (2021).

I used a containerised version of Stacks version 2.62 (Catchen et al., 2011; Catchen et al., 2013) container on the JCU High Performance Computer.

Procedure

Preparing the input files

Create directories

On the HPC, I created the following directory to house my analysis

```
proj_spinosa/  
  deriving_genotypes/  
    top/  
      alignments/  
      cleaned/  
      genome/  
      info/  
      raw/  
      stacks.denovo/  
      stacks.ref/  
      tests.denovo/  
      tests.ref/
```

Add read files

Within the **raw/** sub directory, I created **lane2/** in which my read files (.fastq.gz files) went.

Metadata can be found in 12377_CAGRF22039951_DemultiplexingSampleSheet_PstI_HpyCH4IV_Run1126V_Libraries_1A_

Visually assessing the sequencing data

The read structure:

The samples are ligated with 2 NGS adaptors, and two multiplexing barcodes. The structure of each read should be the following:

multiplexing barcode common restriction site overhang (PstI) genomic fragment 280 - 342 bp
long rare restriction site NGS adaptor 2.

Inspect a read:

I use the following code to inspect some reads, and it looks like it is single-end with an index barcode and an inline barcode. So I will use the call **-inline_index** to specify this barcode combination.

```
# Inspect:
zcat 12377-1A_HGNMFDRX2_CTTGTA_L002_R1.fastq.gz | head -n 20
```

Cleaning and demultiplexing the reads

Process radtags

I will create the following script, which I will call `process_radtags_spinosa.sh` and I will run it on the HPC as a batch job.

In addition to running this script, I need to rename the raw files. They currently are named... `12377-1A_HGNMFDRX2_CTTGTA_L002_R1.fastq.gz` and `12377-1B_HGNMFDRX2_ACAGTG_L002_R1.fastq.gz`. But they need to be named `12377-1A_HGNMFDRX2_CTTGTA_L002_R1_001.fastq.gz` and `12377-1B_HGNMFDRX2_ACAGTG_L002_R1_001.fastq.gz`. They are only renamed in the raw directory in the HPC.

```
#!/bin/bash
# Shebang

#PBS -j oe
# Merge the standard output and error output.

#PBS -N process_radtags
# Name the job

#PBS -l select=1:ncpus=2:mem=6gb
# Decide on for computation grunt

#PBS -l walltime=6:00:00
# How long will it take

#PBS -m ae
# Send an email

#PBS -M name@domain.email
# Send the email to me

# END PBS SCRIPT

# USE APPTAINERS
cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

# DIRECTORIES

# Move into the cleaned/ directory.

cd ~/projects/proj_spinosa/deriving_genotypes/top/cleaned/

# Set up the directory where the raw files are
```

```

raw_dir=../raw/lane2/

# Set up the directory where the barcodes file is
barcodes_file=../info/barcodes.lane2.tsv

# LOAD module

module load stacks

# RUN process_radtags
stacks process_radtags -p $raw_dir -b $barcodes_file -o ./ -r -c -q \
--inline_index --renz_1 pstI --renz_2 hpyCH4IV

```

Arguments:

- *-P* recognises that it is paired end sequencing products
- *-p* input directory
- *-b* barcodes file directory
- *-o* output directory
- *-r*, *-c*, *-q* these arguments clean the data, correct the barcodes, and correct the restriction enzyme cutsites.
- *--inline_index* as described previously, this argument specifies the type of combinatorial barcoding. Make sure you check the data and know what you are dealing with before running this function.
- *--renz_1*, *--renz_2* specify what restriction enzymes are used.

Walltime:I have allowed for 6 hours based on previous experience.

Memory:I have allowed for 6 gb based on previous experience with *L. lorica*.

Job resources

- CPUs requested: 1 CPUs.
- CPUs used: 99% of 1 CPU.
- Wall time requested: 6:00:00
- CPU time: 3:03:03
- Total wall time: 03:03:29
- Memory requested: 6 gb
- Physical memory used: 116288 kb = 0.116288 gb
- Virtual memory used: 1404824 kb = 1.404824 gb

Parameter testing

Background

The basic process by which STACKs derives genotypes (following demultiplexing) is as follows:

ustacks

1. Identical reads within an individual sample are combined into putative alleles (called ‘stacks’).
2. Similar stacks within an individual sample are combined into putative loci.

cstacks

3. Similar loci are combined across individuals into a catalogue of population-wide loci.

Three of the parameters involved in this process are very important.

- **m** - the minimum number of identical reads required to form a stack.
- **M** - the maximum distance between two stacks required to join them as two alleles of the same locus.
- **n** - the maximum distance allowed between loci across individuals required to join them as the same locus.
- A fourth parameter: **N** - controls the re-incorporation of reads that were discarded in the first step, but it is not generally recommended to modify it from its default value except if modifying m to high values).*

Setting these values too low or too high can introduce error or reduce the amount of polymorphic loci (valuable biological information). Hence, it is important to explore the possible values for these parameters to determine what is ideal. I will base my exploration on Rochette and Catchen (2017) and Paris et al., (2017), as well as the STACKS manual. To determine what is optimum, I will modify the parameters within a particular range of values and monitor two things: (1) the number of loci shared by 80% of the samples in my test set; and, (2) the distribution of the number of SNPs per locus. I seek to assess whether a certain parameter maximises the former, and stabilises the later.

Prior to running the r80 method, I need to identify ‘bad apples’ (Cerca et al., 2021), individuals that have lower read depths or call rates. I will run the full population map for each of Oakview and Nangur separately (including both old and recent samples) in the denovo_map pipeline in stacks. I am running the following parameters: M=2 n=2, M=2 n=3, M=3 n=2, M=3 n=3, M=3 n=4, M=4 n=3, M=4 n=4, M=4 n=5.

Stacks

Note that I manually created a new pop map file for all of the Oakview samples and all of the Nangur samples.

There is a total of 16 scripts to run, each of which looks broadly like this:

```
#!/bin/bash
# Shebang

#PBS -j oe
# Merge the standard output and error output.

#PBS -N Oakview.denovo_M2n2
# Name the job

#PBS -l select=1:ncpus=8:mem=40gb
# Decide on memory and cpus tht is allocated to each parallel job.

#PBS -l walltime=24:00:00
# How long each job is allowed to take
```

```

#PBS -m ae
# Send an email

#PBS -M name@domain.email
# Send the email to me

# End PBS script

# USE APPTAINERS
cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

# Begin job script

# move to directory
cd ~/projects/proj_spinosa/deriving_genotypes/top/stacks.denovo/

# save standard environmental variables
popmap=./info/Oakview_popmap.tsv

reads_dir=./cleaned

M=2
n=2

# load the most recent working version of stacks
module load stacks/2.62

# Choose directory to work in (based on parameters)
out_dir=stacks.M$M.n$n/Oakview

# Create log file
log_file=$out_dir/denovo_map_full.M$M.n$n.oe

# run the program
denovo_map.pl --samples $reads_dir --popmap $popmap -o $out_dir \
-M $M -n $n -m 3 -T 8 &> $log_file

```

Here is an example of the computation resources used by one of these runs:

```

PBS Job Id: 2360723.jobmgr1 Job Name: Oakview.denovo_M4n3 Execution terminated Exit_status=0
resources_used.cputpercent=689 resources_used.cput=14:18:27 resources_used.mem=10880796kb re-
sources_used.ncpus=8 resources_used.vmem=12849896kb resources_used.walltime=07:37:15

```

vcftools

I will now export all of the results to vcftools. I will then assess a few important summary statistics (especially missingness) exclude some individuals and run the Tiger analysis for each population.

See the optimisation folder for the list of scripts used for this analysis.

I first used `spinosa_stacks2vcf.sh` to convert each parameter combination to vcf format.

PBS Job Id: 2361039.jobmgr1 Job Name: spinosa_stacks2vcf Execution terminated Exit_status=0
resources_used.cpus=109 resources_used.cput=00:23:17 resources_used.mem=363044kb re-
sources_used.ncpus=8 resources_used.vmem=2195420kb resources_used.walltime=00:22:32

I then copied all of the `populations.snps.vcf` files into the `optimisation/0_stacks2vcf/` file (within top) and renamed them with their parameter combo and their population. E.g.,

```
cp ../../stacks.denovo/stacks.M4.n5/Nangur/populations.snps.vcf stacks.M4.n5.Nangur.snps.vcf
```

I now want to export per individual statistics for call rate and read depth and individual heterozygosity from vcftools to look for bad apples. I will also export per site statistics for call rate and read depth.

I did that using script that looked like this (but for each parameter combination):

```
# Find directory and initiate software

cd projects/proj_spinosa/deriving_genotypes/top/optimisation/1_vcftools/

module load vcftools/0.1.16
# Please add the section for singularities
# if you are submitting this script as a job.

# Set path

input=stacks.M2.n2.Nangur.snps.vcf

vcf_name=spinosa_M2n2_Nangur

# Filter for biallelic data only

vcftools --vcf $input --min-alleles 2 --max-alleles 2 --recode --out $vcf_name

# SUMMARISE VCF

vcftools --vcf $vcf_name.recode.vcf --depth --out $vcf_name.summary

vcftools --vcf $vcf_name.recode.vcf --site-mean-depth --out $vcf_name.summary

vcftools --vcf $vcf_name.recode.vcf --missing-indv --out $vcf_name.summary

vcftools --vcf $vcf_name.recode.vcf --missing-site --out $vcf_name.summary

vcftools --vcf $vcf_name.recode.vcf --het --out $vcf_name.summary
```

I plotted all of the results output by that script (for each of the 5 parameter combinations).

An example of the figures output by this:

```
M2n2_Nangur.perindv_missingness
```

‘Bad Apples’ to remove.

The ‘bad apples’ were identified as Nangur05, Oakview05, and BSU0454. These individuals were removed from this analysis.

r80 method

I ran the full dataset through the r80 method, which provided a recommended parameter value for M and n for my dataset. As in the previous step, I tested M=2 to M=6. I excluded all of the bad apples from the dataset (Nangur05, Oakview05, and BSU0454) by editing the popmap that was used.

I used the following two scripts:

```
#!/bin/bash
# Shebang

#PBS -j oe
# Merge the standard output and error output.

#PBS -N Stacks.denovo_M2
# Name the job

#PBS -l select=1:ncpus=8:mem=30gb
# Decide on memory and cpus tht is allocated to each parallel job.

#PBS -l walltime=60:00:00
# How long each job is allowed to take

#PBS -m ae
# Send an email

#PBS -M name@domain.email
# Send the email to me

# End PBS script

# USE APPTAINERS
cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

# Begin job script

# move to directory
cd ~/projects/proj_spinosa/deriving_genotypes/top/stacks.denovo/

# save standard environmental variables
popmap=../info/popmap_Final.tsv

reads_dir=../cleaned
```

```

M=2
n=$M

# load the most recent working version of stacks
module load stacks/2.62

# Choose directory to work in (based on parameters)
out_dir=stacks.M$M.n$n

# Create log file
log_file=$out_dir/denovo_map_full.M$M.n$n.oe

# run the program

denovo_map.pl --samples $reads_dir --popmap $popmap -o $out_dir \
-M $M -n $n -m 3 -T 8 &> $log_file

# Note here that I use populations to filter out loci that aren't
# found in 80% of the population.

populations -P $out_dir -R 0.80 --max-obs-het 0.6 --write-single-snp \
--vcf &> $log_file

```

```

#!/bin/bash
# Shebang

#PBS -j oe
# Merge the standard output and error output.

#PBS -N r80stacks-dist-extract
# Name the job

#PBS -l select=1:ncpus=1:mem=2gb
# Decide on memory and cpus tht is allocated to each parallel job.
# This is based on each populations run taking 300000, and there being 23 runs.
# Only one cpu it will only go in series because it is small.

#PBS -l walltime=1:00:00
# How long each job is allowed to take. This is based on each run of
# populations taking 5 minutes with some buffer.

#PBS -m ae
# Send an email

#PBS -M name@domain.email
# Send the email to me

# End PBS script

# USE APPTAINERS
cd $PBS_O_WORKDIR

```

```

shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

# Begin job script

# move to main directory I will work from
cd ~/projects/proj_spinosa/deriving_genotypes/top/stacks.denovo/

# Create a directory to place the output in
mkdir r80distribs

# load the most recent working version of stacks
module load stacks/2.62

# Start a for loop that runs through each parameter combo and
# extracts the distributions from the populations.log.distribs file.
# Run through 8 iterations
for i in {2..6}
do

# Make the variables for M.
let M=$i

# Say the variables
echo "This time, the variables are..."
echo $M

# LOCI PER SAMPLE
# Extract the file and put it into a .txt file for the first value of n
stacks-dist-extract stacks.M${M}.n${M}/populations.log.distribs \
loci_per_sample &> r80distribs/loci_per_sample_M${M}n${M}.txt

# VARIANT SITES PER SAMPLE
# Extract the file and put it into a .txt file for the first value of n
stacks-dist-extract stacks.M${M}.n${M}/populations.log.distribs \
variant_sites_per_sample &> r80distribs/variant_sites_per_sample_M${M}n${M}.txt

# SNPS PER LOCUS
# Extract the file and put it into a .txt file for the first value of n
stacks-dist-extract stacks.M${M}.n${M}/populations.log.distribs \
snps_per_loc_postfilters &> r80distribs/snps_per_loc_postfilters_M${M}n${M}.txt

done

```

The samples that are replicates

Note that these samples are replicates of the same individuals:

- BSU0531 and BSU0449
- BSU0535 and BSU0448

Tiger

It is possible to infer error rates using individual replicates. Following Bresadola et al., (2020) I use the software Tiger to determine the value of the M and n parameters provide the lowest error rates.

Inferring error rates per parameter set I will now run these through Tiger to try infer error rates for each parameter set.

vcf filtering

First I need to filter the vcf files from each parameter set to only include these individuals.

That uses this code in each folders (note that M2n2 will change):

```
vcftools --vcf populations.snps.vcf --indv BSU0531 --indv BSU0449 --indv BSU0535 --indv BSU0448 --recode --out indReps_M6n6
```

tiger setup I need a text file that looks like this

```
sample group
BSU0531 1
BSU0449 1
BSU0535 2
BSU0448 2
```

Then I used a scripts that looked like this:

```
#!/bin/bash
# Shebang

#PBS -j oe
# Merge the standard output and error output.

#PBS -N spinosa_tiger-M2n2
# Name the job

#PBS -l select=1:ncpus=1:mem=5gb
# Decide on memory and cpus that is allocated to each parallel job.

#PBS -l walltime=50:00:00
# How long each job is allowed to take

#PBS -m ae
# Send an email

#PBS -M name@domain.email
```

```

# Send the email to me

# End PBS script

# USE APPTAINERS
cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

# Begin job script

# EXPORT STACKS FILES AS VCF

# move to directory
cd ~/projects/proj_spinosa/deriving_genotypes/top/optimisation/2_tiger

# load the most recent working version of stacks
module load tiger

# Convert from stacks to vcftools format using populations module
# for each relevant parameter set

path=../../stacks.denovo/stacks.M2.n2/indReps_M2n2.recode.vcf

tiger task=estimateIndReps vcf=${path} groups=indReps_groups.txt groupCol=2

```

The resources used looked like this:

PBS Job Id: 2363540.jobmgr1 Job Name: spinosa_tiger-M4n4 Execution terminated Exit_status=0 resources_used.cpupercent=99 resources_used.cput=00:03:48 resources_used.mem=26748kb resources_used.ncpus=1 resources_used.vmem=1217252kb resources_used.walltime=00:03:48

I output the results into R to investigate:

- Calculated error rate for each read depth (overall, homs, hets)
- Calculated numbers of calls for each read depth
- Calculated error rate for each parameter set (averaged across all read depths)
- Calculated error rate for each parameter set (averaged across useful read depths)

Exporting Datasets from Stacks

Based on the outcomes of the above (r80 method, Tiger method, Bad Apple method), I decided to use the Stacks parameters $m = 3$, $M = 2$, $n = 2$. Usefully, the method used to optimise these parameters meant that the populations module of Stacks had already generated the SNP dataset as a VCF file.

Dataset 1

The process leading to Dataset 1 started with the output of the r80 method (using the Stack parameters: $M = 2$, $n = 2$). The script that generated this is already provided above.

Preliminary analyses used vcftools version 0.1.16 (Danecek et al., 2011) and R to explore key parameters of the vcf file. Then the following script was run producing four ‘child’ datasets...

```
#!/bin/bash
# Shebang

#PBS -j oe
# Merge the standard output and error output.

#PBS -N spinosa_vcf_filter1
# Name the job

#PBS -l select=1:ncpus=1:mem=1gb
# Decide on memory and cpus tht is allocated to each parallel job.

#PBS -l walltime=05:00:00
# How long each job is allowed to take

#PBS -m ae
# Send an email

#PBS -M name@domain.email
# Send the email to me

# End PBS script

# USE APPTAINERS
cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

# Begin job script 1

# SETUP WORKING DIRECTORY

cd ~/projects/proj_spinosa/spinosa_analysis/1_vcf_filtering/

# USE APPTAINERS
cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

# LOAD SOFTWARE

# load the most recent working version of vcftools
module load vcftools/0.1.16
```

```

# SETUP PATH

IN_FILE=../O_data/NS_M2n2m3_parent.vcf

# APPLY FILTERS

# Child Alpha:
# Call rate 80% (i.e., no change)
# Read depth 10 - 60
# Minor allele count 1 (i.e., no change)

vcftools --vcf $IN_FILE --min-meanDP 10 --max-meanDP 60 --recode \
--out NS_M2n2m3_ChildAlpha

# Child Beta:
# Call rate 80% (i.e., no change)
# Read depth 10 - 60
# Minor allele count 3 (minimum)

vcftools --vcf $IN_FILE --min-meanDP 10 --max-meanDP 60 --mac 3 --recode \
--out NS_M2n2m3_ChildBeta

# Child Gamma:
# Call rate 100% (minimum)
# Read depth 10 - 60
# Minor allele count 1 (i.e., no change)

vcftools --vcf $IN_FILE --max-missing 1 --min-meanDP 10 --max-meanDP 60 \
--recode --out NS_M2n2m3_ChildGamma

# Child Delta:
# Call rate 100% (minimum)
# Read depth 10 - 60
# Minor allele count 3 (minimum)

vcftools --vcf $IN_FILE --max-missing 1 --min-meanDP 10 --max-meanDP 60 \
--mac 3 --recode --out NS_M2n2m3_ChildDelta

```

Samples to be removed The four ‘Child’ VCF files were then explored using vcftools version 0.1.16 (Danecek et al., 2011). Individuals in these files were explored using exports of vcftools: kinship, genetic distance, observed and expected heterozygosities. The following samples were then removed from all ‘Child’ datasets after careful examination. Note that some individuals were already removed.

- BSU0433 - probable contamination by BSU0455 (lab notes indicated possible contamination during subsampling and high kinship between individuals that were from separate populations but close on the plate)
- BSU0455 - strangely low inbreeding coefficient (negative) due to higher than expected heterozygosity, possibly due to contamination.
- Nangur01-04 and Oakview01-04 as they are older samples and technically not part of the current population. They will be included in Dataset , but not in Dataset 1 or 2.
- One of each duplicates. Picked randomly but kept consistent:

- From BSU0448 and BSU0535 (I chose to keep only BSU0535).
- From BSU0449 and BSU0531 (I chose to keep only BSU0531).

Finallising Dataset 1 All of the four ‘Child’ vcf files were ran through many of the analyses used in the final methodology to investigate any impacts of the differing call rates, and minor allele counts. When I was satisfied that they wouldn’t make a huge difference, ‘Child Delta’ was chosen to be Dataset 1.

Dataset 2

Schmidt et al. (2021) demonstrated that robust estimates of heterozygosity required the inclusion of all sites on all loci (‘genome-wide estimates’), not just SNPs on polymorphic loci (‘SNP estimates’). For this reason, a second specialised dataset was generated from the data for analysing genetic diversity metrics. So that monomorphic loci would be subject to filters, Dataset 2 was filtered in Stacks rather than VCFtools.

Note that I have included the list of excluded loci which have been filtered by read depth. These loci can only be identified by first producing this dataset WITHOUT removing those loci, and then generating the list of loci to be removed in vcftools.

A new script that looks like this was used to perform this filtering:

```
#!/bin/bash
# Shebang

#PBS -j oe
# Merge the standard output and error output.

#PBS -N autosomal_He
# Name the job

#PBS -l select=1:ncpus=1:mem=5gb
# Decide on memory and cpus tht is allocated to each parallel job.
# This is based on each populations run taking 300000, and there being 23 runs.
# Only one cpu it will only go in series because it is small.

#PBS -l walltime=5:00:00
# How long each job is allowed to take. This is based on each run of
# populations taking 5 minutes with some buffer.

#PBS -m ae
# Send an email

#PBS -M name@domain.email
# Send the email to me

# End PBS script

# USE APPTAINERS
cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"
```

```

# Begin job script

# move to main directory I will work from
cd ~/projects/proj_spinosa/deriving_genotypes/top/stacks.denovo/

# load the most recent working version of stacks
module load stacks/2.62

# folder to access files from
stacks_dir=stacks.M2.n2

# create a new directory to use
mkdir $stacks_dir/ChildGamma_cleanPop/depthFiltered

# Where to put the results
out_dir=$stacks_dir/ChildGamma_cleanPop/depthFiltered

# Create log file
log_file=$out_dir/ChildGamma_cleanPop.oe

# run the populations to reflect all filtering settings
# Note that I have included the list of excluded loci which have been filtered
# by read depth. These loci can only be identified by first producing this
# dataset WITHOUT removing those loci.

populations -P $stacks_dir --popmap ../info/popmap_He_cleanPop.tsv -O $out_dir \
-R 1 --max-obs-het 0.6 -B \
../info/excludedLoci.autosomal_ChildGamma_cleanPop.tsv --vcf-all &> $log_file

```

Importantly, the outputs of this were examined so that the number of invariant loci was known. These are automatically filtered by vcftools, despite being included in the initially included in the vcffile. However, once the number is known, they can be removed to shrink the size of the file.

Dataset 3

The process of generating Dataset3 is identical to the process of generating Dataset2, except that the popmap includes individuals Nangur01-04 and Oakview01-04. Later, it was important to check that read depth was not correlated with heterozygosity, as these older samples were less well preserved. Fortunately, there was no such correlation and their read depths were fine.

```

#!/bin/bash
# Shebang

#PBS -j oe
# Merge the standard output and error output.

#PBS -N autosomal_He
# Name the job

#PBS -l select=1:ncpus=1:mem=5gb
# Decide on memory and cpus tht is allocated to each parallel job.

```

```

# This is based on each populations run taking 300000, and there being 23 runs.
# Only one cpu it will only go in series because it is small.

#PBS -l walltime=5:00:00
# How long each job is allowed to take. This is based on each run of populations
# taking 5 minutes with some buffer.

#PBS -m ae
# Send an email

#PBS -M name@domain.email
# Send the email to me

# End PBS script

# USE APPTAINERS
cd $PBS_O_WORKDIR
shopt -s expand_aliases
source /etc/profile.d/modules.sh
echo "Job identifier is $PBS_JOBID"
echo "Working directory is $PBS_O_WORKDIR"

# Begin job script

# move to main directory I will work from
cd ~/projects/proj_spinosa/deriving_genotypes/top/stacks.denovo/

# load the most recent working version of stacks
module load stacks/2.62

# folder to access files from
stacks_dir=stacks.M2.n2

# create a new directory to use
mkdir $stacks_dir/ChildGamma_cleanPop_oldPop/depthFiltered

# Where to put the results
out_dir=$stacks_dir/ChildGamma_cleanPop_oldPop/depthFiltered

# Create log file
log_file=$out_dir/ChildGamma_cleanPop_oldPop.oe

# run the populations to reflect all filtering settings
# Note that I have included the list of excluded loci which have been filtered
# by read depth. These loci can only be identified by first producing this
# dataset WITHOUT removing those loci.

populations -P $stacks_dir --popmap \
../info/popmap_autosomal_cleanPop_oldPop.tsv -O $out_dir -R 1 \
--max-obs-het 0.6 -B ../info/excludedLoci.autosomal_ChildGamma_cleanPop.tsv \

```

```
--vcf-all &> $log_file
```

References

This reference list only includes references and packages that are cited in this R Markdown document. For a full reference list, please see the main text.

- Borsboom, A. C., Couper, P. J., Amey, A., & Hoskin, C. J. (2010). Distribution and population genetic structure of the critically endangered skink *Nangura spinosa*, and the implications for management. *Australian Journal of Zoology*, 58(6), 369. <https://doi.org/10.1071/ZO10070>
- Bresadola, L., Link, V., Buerkle, C. A., Lexer, C., & Wegmann, D. (2020). Estimating and accounting for genotyping errors in RAD-seq experiments. *Molecular Ecology Resources*, 20(4), 856–870. <https://doi.org/10.1111/1755-0998.13153>
- Catchen, J., Hohenlohe, P. A., Bassham, S., Amores, A., & Cresko, W. A. (2013). Stacks: an analysis tool set for population genomics. *Molecular Ecology*, 22(11), 3124–3140. <https://doi.org/10.1111/mec.12354>
- Catchen, J. M., Amores, A., Hohenlohe, P., Cresko, W., & Postlethwait, J. H. (2011). Stacks: Building and Genotyping Loci De Novo From Short-Read Sequences. *G3 Genes|Genomes|Genetics*, 1(3). <https://doi.org/10.1534/g3.111.000240>
- Cerca, J., Maurstad, M. F., Rochette, N. C., Rivera-Colón, A. G., Rayamajhi, N., Catchen, J. M., & Struck, T. H. (2021). Removing the bad apples: A simple bioinformatic method to improve loci-recovery in de novo RADseq data for non-model organisms. *Methods in Ecology and Evolution*, 12(5), 805–817. <https://doi.org/10.1111/2041-210X.13562>
- Danecek, P., Auton, A., Abecasis, G., Albers, C. A., Banks, E., DePristo, M. A., Handsaker, R. E., Lunter, G., Marth, G. T., Sherry, S. T., McVean, G., & Durbin, R. (2011). The variant call format and VCFtools. *Bioinformatics*, 27(15), 2156–2158. <https://doi.org/10.1093/bioinformatics/btr330>
- Paris, J. R., Stevens, J. R., & Catchen, J. M. (2017). Lost in parameter space: a road map for stacks. *Methods in Ecology and Evolution*, 8(10), 1360–1373. <https://doi.org/10.1111/2041-210X.12775>
- Rochette, N. C., & Catchen, J. M. (2017). Deriving genotypes from RAD-seq short-read data using Stacks. *Nature Protocols*, 12(12), 2640–2659. <https://doi.org/10.1038/nprot.2017.123>
- Schmidt, T. L., Jasper, M., Weeks, A. R., & Hoffmann, A. A. (2021). Unbiased population heterozygosity estimates from genome-wide sequence data. *Methods in Ecology and Evolution*, 12(10), 1888–1898. <https://doi.org/10.1111/2041-210X.13659>

Packages

```
citation('devtools')
```

```
## To cite package 'devtools' in publications use:
##
## Wickham H, Hester J, Chang W, Bryan J (2022). _devtools: Tools to
## Make Developing R Packages Easier_. R package version 2.4.5,
## <https://CRAN.R-project.org/package=devtools>.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {devtools: Tools to Make Developing R Packages Easier},
##   author = {Hadley Wickham and Jim Hester and Winston Chang and Jennifer Bryan},
```

```
##   year = {2022},
##   note = {R package version 2.4.5},
##   url = {https://CRAN.R-project.org/package=devtools},
## }
```

```
citation('tidyverse')
```

```
## To cite package 'tidyverse' in publications use:
```

```
##
```

```
## Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R,
## Golemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller
## E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V,
## Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to
## the tidyverse." _Journal of Open Source Software_, *4*(43), 1686.
## doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.
```

```
##
```

```
## A BibTeX entry for LaTeX users is
```

```
##
```

```
## @Article{,
##   title = {Welcome to the {tidyverse}},
##   author = {Hadley Wickham and Mara Averick and Jennifer Bryan and Winston Chang and Lucy D'Agostini
##   year = {2019},
##   journal = {Journal of Open Source Software},
##   volume = {4},
##   number = {43},
##   pages = {1686},
##   doi = {10.21105/joss.01686},
## }
```

```
citation('knitr')
```

```
## To cite package 'knitr' in publications use:
```

```
##
```

```
## Xie Y (2024). _knitr: A General-Purpose Package for Dynamic Report
## Generation in R_. R package version 1.49, <https://yihui.org/knitr/>.
```

```
##
```

```
## Yihui Xie (2015) Dynamic Documents with R and knitr. 2nd edition.
## Chapman and Hall/CRC. ISBN 978-1498716963
```

```
##
```

```
## Yihui Xie (2014) knitr: A Comprehensive Tool for Reproducible
## Research in R. In Victoria Stodden, Friedrich Leisch and Roger D.
## Peng, editors, Implementing Reproducible Computational Research.
## Chapman and Hall/CRC. ISBN 978-1466561595
```

```
##
```

```
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```

```
citation('kableExtra')
```

```
## To cite package 'kableExtra' in publications use:
```

```
##
```

```
## Zhu H (2024). _kableExtra: Construct Complex Table with 'kable' and
## Pipe Syntax_. R package version 1.4.0,
## <https://CRAN.R-project.org/package=kableExtra>.
##
## A BibTeX entry for LaTeX users is
##
## @Manual{,
##   title = {kableExtra: Construct Complex Table with 'kable' and Pipe Syntax},
##   author = {Hao Zhu},
##   year = {2024},
##   note = {R package version 1.4.0},
##   url = {https://CRAN.R-project.org/package=kableExtra},
## }
```

```
citation('tinytex')
```

```
## To cite package 'tinytex' in publications use:
##
## Xie Y (2025). _tinytex: Helper Functions to Install and Maintain TeX
## Live, and Compile LaTeX Documents_. R package version 0.55,
## <https://github.com/rstudio/tinytex>.
##
## Xie Y (2019). "TinyTeX: A lightweight, cross-platform, and
## easy-to-maintain LaTeX distribution based on TeX Live." _TUGboat_,
## *40*(1), 30-32. <https://tug.org/TUGboat/Contents/contents40-1.html>.
##
## To see these entries in BibTeX format, use 'print(<citation>,
## bibtex=TRUE)', 'toBibtex(.)', or set
## 'options(citation.bibtex.max=999)'.
```