pool-seq pipeline - Stanford et al.

brenna.cm.stanford[1]

[1]University of Calgary

brenna.cm.stanford

MAY 08, 2023

Protocol status: Working
We use this protocol and it's working

Created: Jul 28, 2022

Last Modified: May 08, 2023

PROTOCOL integer ID: 67838

ABSTRACT

Determining cryptic species and diversity in at-risk species is necessary for the understanding and conservation of biodiversity. The endangered Banff Springs Snail, *Physella johnsoni*, inhabits seven highly specialized thermal springs in Banff National Park, Alberta, Canada. However, it has been difficult to reconcile its species status to the much more common *Physella gyrina* using ecology, morphology and genetics. Here we used pooled whole-genome sequencing to characterize genomic variation and structure among five populations of *P. johnsoni* and three geographical proximate *P. gyrina* populations. By comparing over two million single nucleotide polymorphisms, we detected substantial genetic distance (pairwise F_{ST} of 0.27 to 0.44) between *P. johnsoni* and *P. gyrina*, indicative of unique gene pools. Genetic clusters among populations were found for both species, with up to 10% for *P. johnsoni*and 30% for *P. gyrina* of genetic variation being explained by population structure. *P. johnsoni* was found to have lower genetic diversity compared to *P. gyrina*, however, no patterns of were observed between genetic diversity and population minimums. Our results confirm that designation of *P. johnsoni* as an endangered species is warranted and that both *P. johnsoni*and *P. gyrina* exhibit microgeographic population genomic structure suggestive of rapid local adaptation and/or genetic drift within environments. This study showcases the utility of genomics to resolve patterns of cryptic species and diversity for effective conservation management. Future studies on the functional genomic diversity of *P. johnsoni* populations are needed to test for the possible role of selection within this thermal spring environment.

Project information and background

1 Project information

Below is the pipeline I used to analyze my pool-seq data (Stanford 2019). For my project, I analyzed five populations of *Physella johnsoni* and three populations of *Physella gyrina* of 20 to 40 individuals per population. These were sequenced by Genome Quebec on the Illumina HiSeq X, aiming for about 40x coverage (determined to be almost double that in actuality). Each

population consisted of half a lane worth of data (total of four lanes). I included the SLURMs because it gives an idea of how long each thing took to run for my files.

1.1 Helpful SLURM commands

sbatch SLURM_name #will submit job to queue

squeue -u your_user_name #will give you the job's status

scancel job_ID_number #will cancel the job

acct -j job_ID_number --format=JobID,JobName,MaxRSS,Elapsed #Gives JobID (kind of redundant), the name of the job, the memory and the time it took.

Trimming and cleaning reads

2 Convert BCL to Fastq

Can skip this step if sequencing company has returned fastq files and you are happy with the barcode mismatch parameters they used.

Manual: https://support.illumina.com/content/dam/illumina-support/documents/documentation/software_documentation/bcl2fastq/bcl2fastq2_guide_15051736_v2.pdf

tiles: these are the tiles that your data is on (because there will likely be other sequence info)

sample-sheet: need to get this from Genome Quebec. Just info about your sequences

-r -p -w : reading, processing and writing thread count. I set them all to 32.

use-bases-mask Y151,I6n2,Y151: this is specific about the sequencer - the Y151 is because it is PE 150 reads

```
#!/bin/bash
# -----------------------------------------------------------------
# bcl2fastq for L001
# -----------------------------------------------------------------

#SBATCH --job-name=bcl2fastq.1
#SBATCH --account=def-account
#SBATCH --cpus-per-task=8
#SBATCH --time=0-03:00
#SBATCH --mem=20G

# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
```

```
echo "Starting run at: `date`"
# ------------------------------------------------------------

module load bcl2fastq2/2.20

bcl2fastq\
--runfolder-dir /path/to/BCLfiles/
--output-dir /path/to/00_raw_data \
--tiles s_1 \
--sample-sheet /path/to/BCLfiles/SampleSheet.1.csv \
 --create-fastq-for-index-reads \
-r 8 -p 8 -w 8 \
--barcode-mismatches 0 --use-bases-mask Y151,I6n2,Y151

# ------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# ------------------------------------------------------------
```

3 Concatenate files together

```
cat XX_Pool_L001_R1.fq.gz XX_Pool_L002_R1.fq.gz > XX_Pool_R1.fq.gz
```

4 FastQC - check the quality of your raw sequencing reads

```
#!/bin/bash
# ------------------------------------------------------------
# FastQC for  pool sites
# ------------------------------------------------------------

#SBATCH --job-name=fastqc
#SBATCH --account=def-account
#SBATCH --cpus-per-task=1
#SBATCH --time=0-15:00
#SBATCH --mem=5G

# ------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# ------------------------------------------------------------

module load fastqc/0.11.5

for i in /path/to/fastqfiles/*fq.gz
do
      fastqc -o /path/to/where/you/want/reports $i
```

```
done
# ----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# ----------------------------------------------------------------
```

5 Trimmomatic - cleaning and filtering low-quality reads and removing adaptors

phred : can be 64. Depends on your sequencing.

threads : change to the number of threads you have available on your cluster. We used 16, half of a node, but I think we could have used more. Just make sure to match this to what you are asking for in your SLURM.

ILLUMINACLIP : path to the adaptor file you made.
2 : seed mismatches
30 : is how accurate the match between the two adaptor ligated reads must be
10 : SimpleClip Threshold

CROP : we were having an issue in some of our reads where Trimmomatic just wasn't detecting the repetitive sequences in the last 15 nucleotides. Hence the hard crop to trim the last 15 nucleotides off each read.

LEADING : trim the leading nucleotides if they fall under Q5

TRAILING : trim the trailing nucleotides if they fall under Q5

SLIDINGWINDOW :
5:20 : look at 5 bases at a time and trim if the average Q is less than 20

MINLEN : only keep reads that are minimally # bp

```
#!/bin/bash
# ----------------------------------------------------------------
# Trim cat files
# ----------------------------------------------------------------
#SBATCH --job-name=XX_trim
#SBATCH --account=def-srogers
#SBATCH --cpus-per-task=16
#SBATCH --time=0-20:00
#SBATCH --mem=15G
#SBATCH --array=1-8
# ----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# ----------------------------------------------------------------
```

```
module load trimmomatic/0.36

cd /path/to/fastqfiles

filenameR1=`ls -1 *_R1.fastq.gz | tail -n +${SLURM_ARRAY_TASK_ID} | head-1`
filenameR2=`ls -1 *_R2.fastq.gz | tail -n +${SLURM_ARRAY_TASK_ID} | head-1`

filenameR1.out=${filenameR1::-4} #adjust for the length you need to remove
filenameR2.out=${filenameR2::-4} #adjust for the length you need to remove

java -jar $EBROOTTRIMMOMATIC/trimmomatic-0.36.jar PE -phred33 -threads 16 -trimlog logfile \
$filenameR1 $filenameR2 \
${filenameR1.out} _P_qtrim.fq ${filenameR1.out}_U_qtrim.fq ${filenameR2.out} _P_qtrim.fq
${filenameR2.out} _U_qtrim.fq \
ILLUMINACLIP:/path/to/Adaptors/TruSeq3-PE-all.fa:2:30:10 CROP:135 LEADING:5 TRAILING:5
SLIDINGWINDOW:5:20 MINLEN:100

# ------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ------------------------------------------------------------
```

Contamination removal (skip if high quality reference geno...

6 The steps below allow you to filter your sequences against databases of different potential contaminants. I didn't find it removed many sequences. I made the databases a variety of ways as I found out one way wouldn't work for every group of organisms.

6.1 Databases for Archaea and Algae (Charophyceae, Chlorophyta, Cryptophyta, Eustigmatophyceae and Klebsormidiophyceae)

Create a database from NCBI of the sequences you would like to remove. You need to get the GI list
from NCBI (as per http://johnstantongeddes.org/aptranscriptome/2013/12/31/notes.html or https://www.biostars.org/p/6528/). You will need to install the newest version of NCBI BLAST+ (https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=Download).

I have heard that moving forward NCBI will be switching to using taxid to create databases rather than GI list but upon publishing this GI list were still being used.

In this example, I am using Archaea. Key thing is that the nt database and your GI list must exist
in the same directory. It will be messy, which is why I put the "X" in front of the files I was

making so
that all of them were at the bottom.

/path/to/ncbi_directory/ncbi-blast-2.7.1+/bin/blastdb_aliastool -db nt -gilist Archaea.gi
-dbtype nucl -out X_nt_archaea -title "database for Archaea"

Once you've created the .nal file, you need to convert the database to .fasta file. It's small so I ran it in the SLURM.

```
#!/bin/bash
# -----------------------------------------------------------------
# NCBI to fasta
# -----------------------------------------------------------------
#SBATCH --job-name=Charo_fasta
#SBATCH --account=def-srogers
#SBATCH --cpus-per-task=1
#SBATCH --time=0-05:00
#SBATCH --mem=1G
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------

module load perl/5.22.4

/path/to/ncbi/ncbi-blast-2.7.1+/bin/blastdbcmd -entry all
-db X_nt_archaea -out Archaea.fasta

# -----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

6.2 Creating the database for threespine stickleback and human

Threespine stickleback downloaded from:
https://datadryad.org/resource/doi:10.5061/dryad.h7h32 (Peichel et al. 2017)
Human genome was downloaded from:
ftp://ftp.ncbi.nih.gov/genomes/H_sapiens/Assembled_chromosomes/seq/hs_ref_GRCh38.p12_ch.
I used the tutorial provided by DeconSeq to access the human genome.

```
#Download sequence data
for i in {1..22} X Y MT;
do wget
ftp://ftp.ncbi.nih.gov/genomes/H_sapiens/Assembled_chromosomes/seq/hs_ref_GRCh38.p12_chr$i.fa.gz;
done
```

```
#Extracting and joining data
for i in {1..22} X Y MT; do gzip -dvc hs_ref_GRCh38.p12_chr$i.fa.gz
>>hs_ref_GRCh38_p12.fa; rm hs_ref_GRCh38.p12_chr$i.fa.gz;
done
```

6.3 Creation of the Bacterial database

I downloaded all of the full bacterial genomes off of NCBI (I think there was about 10,000 of them?) and all assembly levels for bacteria that have been found in the thermal springs onto a computer (many Gb). I then used Globus to put them on to Cedar. https://www.ncbi.nlm.nih.gov/assembly/?term=bacteria. You will need to unzip any files that are zipped (including your query sequences) because DeconSeq can't use zipped files.

Can use:

```
for file in *.gz #loop through all files with this file extension
do
gunzip $file #unzip them
done #when it's finished, stop.
```

This will had to be done for all of the bacterial genomes. It took 10+ hours so I would consider submitting it as a SLURM.

Then I concatenated the bacterial genomes together, using:

```
#!/bin/bash
# -----------------------------------------------------------------
# cat FG NCBI database
# -----------------------------------------------------------------
#SBATCH --job-name=cat_bac
#SBATCH --account=def-account
#SBATCH --cpus-per-task=16
#SBATCH --time=0-10:00
#SBATCH --mem=1G
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------

find . -name '*.fna' -print0 | xargs -0 cat > bacteria_genomes.fa

# -----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

6.4 Splitting sequences by long repeats of ambiguous base N (this is from the DeconSeq manual

Below is all for the human genome because that is what is listed in the manual but I did this for all of the databases.

```
cat hs_ref_GRCh38_p12.fa | perl -p -e 's/N\n/N/' |
perl -p -e 's/^N+//;s/N+$//;s/N{200,}/\n>split\n/'
>hs_ref_GRCh38_p12_split.fa; rm hs_ref_GRCh38_p12.fa
```

6.5 Filtering databases

Need to download and install PRINSEQ - can be found at
https://sourceforge.net/projects/prinseq/files/
This step needs a SLURM because it will run out of memory otherwise

```
#!/bin/bash
# -----------------------------------------------------------------
# Filtering sequences - PRINSEQ
# -----------------------------------------------------------------
#SBATCH --job-name=human_prinseq
#SBATCH --account=def-account
#SBATCH --cpus-per-task=1
#SBATCH --time=0-0:10
#SBATCH --mem=10G
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------

module load perl/5.22.4

perl /path/to/prinseq-lite-0.20.4/prinseq-lite.pl -log -verbose
-fasta hs_ref_GRCh38_p12_split.fa -min_len 200 -ns_max_p 10 -derep 12345
-out_good hs_ref_GRCh38_p12_split_prinseq -seq_id hs_ref_GRCh38_p12_
-rm_header -out_bad null

# -----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

6.6 Index the databases

For the bacterial database (because it is over 200Gb) you will need to first split it into managable chunks before BWA can use them. Can use fasta splitter (http://kirill-kryukov.com/study/tools/fasta-splitter/).

The files need to be under 3Gb each, so split it to as many chunks as you need. I did 100.

```
#!/bin/bash
```

```
# -----------------------------------------------------------------
# FASTA splitter
# -----------------------------------------------------------------
#SBATCH --job-name=fasta_split_bac
#SBATCH --account=def-account
#SBATCH --cpus-per-task=16
#SBATCH --time=0-10:00
#SBATCH --mem=100G
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------

module load perl/5.22.4

perl /pathto/fasta-splitter.pl --n-parts 100 Bacteria_FG_split_prinseq.fasta

# -----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

Next step is to index the databases You must use the BWA provided in the DeconSeq package!
The newer BWA reads the files incorrectly for this and will only produce 5 of the 8 outfiles necessary. Cedar is a 64 bit Linux system, so use bwa64.

Modified from script kindly provided by Dr. Stefan Dennenmoser

```
#!/bin/bash
# -----------------------------------------------------------------
# BWA
# -----------------------------------------------------------------
#SBATCH --job-name=index_bac
#SBATCH --ntasks=1
#SBATCH --account=def-srogers
#SBATCH --time=0-10:00
#SBATCH --mem-per-cpu=20G
#SBATCH --array=1-100
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------

cd /path/to/Bacterial_Database

filename=`ls -1 *.fasta* | tail -n +${SLURM_ARRAY_TASK_ID} | head -1`
```

```
filename2=${filename::-6} #the filename without the .fasta part (-6 letters)

/path/to/deconseq-standalone-0.4.3/bwa64 index -p $filename2 -a bwtsw $filename
>bwa.log 2>&1

# -----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

6.7 Configure the DeconSeq file

You will have to go into the installed DeconSeq directory and set up the configure file DeconSeqConfig.pm. You will just need to change the database directory and out directory and the what files are accessed in the “use constant DBS”.

6.8 Split the query sequences

DeconSeq is unable to handle big datasets (e.g. 50+ Gb files that I was using)
Will need to use fastq splitter (http://kirill-kryukov.com/study/tools/fastq-splitter/)
I don’t think I ran this in a SLURM and just used the console. If it fails. . . put it in a SLURM.

```
perl /path/to/fastq-splitter.pl --n-parts 50 --check XX_R1.fq
```

6.9 Running Deconseq

For DeconSeq you need to choose the identity (-i) which is the percent match between your query sequence and the database and the coverage (-c) which is the amount of the sequence aligns.
I went with the parameters they used in their paper and based on what I had seen other people do, which was 94% identity (-i 94) and 90% to 95% coverage (-c 90 or -c 95).

Submit this as a batch job.

```
#!/bin/bash
# -----------------------------------------------------------------
# Deconseq
# -----------------------------------------------------------------
#SBATCH --job-name=XX_decon
#SBATCH --ntasks=1
#SBATCH --account=def-srogers
#SBATCH --time=3-00:00
#SBATCH --mem-per-cpu=7500MB
#SBATCH --array=1-50
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------
```

```
module load perl/5.22.4

#put all of the split files from one pop and read direction in its own directory
cd /path/to/trimmed_seq/XX_R1_split
filename=`ls -1 *.fq* | tail -n +${SLURM_ARRAY_TASK_ID} | head -1`
perl /path/to/deconseq-standalone-0.4.3/deconseq.pl -i 94 -c 90 -f
/path/to/trimmed_seq/XX_R1_split/$filename -dbs hsref
-out_dir /path/to/deconseq_out/XX_R1 -id $filename

# ------------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# ------------------------------------------------------------------
```

6.10 Creating paired files again

DeconSeq was never designed for paired-end sequencing. Therefore it will process each read direction separately and this causes sequences to be removed in one direction and not the other.
Firstly, concatenate the 50 (or more or less depending on what you split your in files to) of clean files. I also did this for the .cont files because I wanted to keep them and see how many were removed.

Then you can use this person's script found at: https://github.com/linsalrob/fastq-pair
Step 1: Clone or download - copy URL
Step 2: In Cedar or ARC or whatever cluster type: git clone -should see a directory called "fastq-pair"
Step 3: gcc fastq-pair/*.c -o fastq_pair
Step 4: Should see an exectuable script called "fastq_pair"

Running it is super simple.
path/to/fastq_pair -t VALUE path/to/file_R1.fastq path/to/file_R2.fastq

Where VALUE is roughly the number of sequences in your file. This is the setting of the hash table size. For some reason I don't know why, I couldn't get this to run as a SLURM. It would make the outfiles (if I gave it over 50Gb) but it would run for far longer than it runs in the terminal and just never finish. I ended up running it in ARC's console because Cedar doesn't have enough resources allocated to their console.

It makes four output files. R1 paired and unpaired and R2 paired and unpaired. Then you will need to zip the files back up

Align reads - BWA (Burrows-Wheeler Aligner)

7 Aligning sequences to the reference genome for *Physella acuta* (GenBank RDRX00000000) (Schultz et al. 2020)

7.1 Index the reference genome

```
#!/bin/bash
# -----------------------------------------------------------------
# BWA
# -----------------------------------------------------------------
#SBATCH --job-name=ref_index
#SBATCH --account=def-account
#SBATCH --cpus-per-task=1
#SBATCH --time=0-04:00
#SBATCH --mem=5G
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------

module load nixpkgs/16.09
module load gcc/5.4.0
module load intel/2016.4
module load intel/2017.1
module load bwa/0.7.17

bwa index -p reference_genome /path/to/reference/reference_genome.fa

# -----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

7.2 Align to reference

Details about parameters can be found here: http://bio-bwa.sourceforge.net/bwa.shtml

-M: mark shorter split hits as secondary (necessary for using the file in Picard downstream)
-t 16: number of threads. Match with SLURM
-R: complete read group header line
XX: need to change XX to whatever site you are currently working on.

The reference genome needs to be put in without the “.fa” because it will be using all of the indexes that are in the directory too.

```
#!/bin/bash
# -----------------------------------------------------------------
# sam
# -----------------------------------------------------------------
```

```
#SBATCH --job-name=XX_align
14
#SBATCH --account=def-account
#SBATCH --cpus-per-task=16
#SBATCH --time=0-05:00
#SBATCH --mem=10G
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------

module load nixpkgs/16.09
module load gcc/5.4.0
module load intel/2016.4
module load intel/2017.1
module load bwa/0.7.17

bwa mem -M -t 16 -R '@RG\tID:XX\tLB:XX\tSM:XX\tPL:ILLUMINA'
/path/to/ref/reference_genome
path/to/XX_R1.fq.gz path/to/XX_R2.fq.gz > XX_out.sam

# -----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

Filter mapped reads - Samtools

8

8.1 Sort the .sam file into a .bam by chromosome number

There are two options for sorting but it needs to be sorted by chromosome for downstream applications.

-@: this argument is where you set the number of threads
-T: this argument is where you set the indentifier - change to which ever site you are currently working on
-o: write to this outfile

```
#!/bin/bash
# -----------------------------------------------------------------
# Samtools Sort
# -----------------------------------------------------------------
```

```
#SBATCH --job-name=XX.bam
#SBATCH --account=def-account
#SBATCH --cpus-per-task=16
#SBATCH --time=0-04:00
#SBATCH --mem=5G
# -----------------------------------------------------------------
echo "Current working directory: `pwd`"
echo "Starting run at: `date`"
# -----------------------------------------------------------------
module load nixpkgs/16.09
module load gcc/5.4.0
module load intel/2016.4
module load samtools/1.5

samtools sort -@ 16 -T XX -o XX.bam XX_out.sam

# -----------------------------------------------------------------
echo "Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

8.2 Remove duplicates

```
#!/bin/bash
# -----------------------------------------------------------------
# Picard no dups
# -----------------------------------------------------------------
#SBATCH --job-name=nodups
#SBATCH --account=def-account
#SBATCH --cpus-per-task=6
#SBATCH --time=0-20:00
#SBATCH --mem=100G
#SBATCH --array=1-8
# -----------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# -----------------------------------------------------------------

module load StdEnv/2020
module load picard/2.26.3

inputdir=/path/to/bam/file/directory

cd $inputdir

filename=`ls -1 *.bam | tail -n +${SLURM_ARRAY_TASK_ID} | head -1`
filename2=${filename::-4}
```

```
java -Xmx50g -XX:ParallelGCThreads=5 -jar $EBROOTPICARD/picard.jar MarkDuplicates \
I=$filename \
O=/path/to/outfile/directory/${filename2}_nodup.bam \
M=/path/to/outfile/directory/${filename2}_nodup.txt \
REMOVE_DUPLICATES=TRUE \
MAX_FILE_HANDLES_FOR_READ_ENDS_MAP=900 MAX_RECORDS_IN_RAM=50000
ASSUME_SORT_ORDER=coordinate SORTING_COLLECTION_SIZE_RATIO=0.1

# ------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ------------------------------------------------------------
```

8.3 Filter for quality

```
#!/bin/bash
# ------------------------------------------------------------
# Samtools q20
# ------------------------------------------------------------
#SBATCH --job-name=q20
#SBATCH --account=def-srogers
#SBATCH --cpus-per-task=8
#SBATCH --time=0-20:00
#SBATCH --mem=20G
#SBATCH --array=1-8
# ------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# ------------------------------------------------------------

module load StdEnv/2020
module load gcc/9.3.0
module load samtools/1.13

inputdir=/path/to/no_dup/file/directory

cd $inputdir

filename=`ls -1 *.bam | tail -n +${SLURM_ARRAY_TASK_ID} | head -1`
filename2=${filename::-6}

samtools view -@ 8 -q20 $filename \
-o /path/to/q20/file/directory/${filename2}_q20.bam

# ------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ------------------------------------------------------------
```

8.4 Filter any reads without a pair

```
#!/bin/bash
# ----------------------------------------------------------------
# Samtools f2
# ----------------------------------------------------------------
#SBATCH --job-name=f2
#SBATCH --account=def-account
#SBATCH --cpus-per-task=8
#SBATCH --time=0-20:00
#SBATCH --mem=20G
#SBATCH --array=1-8
# ----------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# ----------------------------------------------------------------

module load StdEnv/2020
module load gcc/9.3.0
module load samtools/1.13

inputdir=/path/to/q_20/file/directory

cd $inputdir

filename=`ls -1 *.bam | tail -n +${SLURM_ARRAY_TASK_ID} | head -1`
filename2=${filename::-12}

samtools view -@ 8 -f2 $filename \
-o /path/to/f2/file/directory/${filename2}_q20.bam/${filename2}_f2.bam

# ----------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ----------------------------------------------------------------
```

Validate reads - Picard

9 Validate files before moving forward

```
#!/bin/bash
# ----------------------------------------------------------------
# Picard validate
# ----------------------------------------------------------------
#SBATCH --job-name=vali
#SBATCH --account=def-account
```

```
#SBATCH --cpus-per-task=6
#SBATCH --time=0-03:00
#SBATCH --mem=100G
#SBATCH --array=1-8
# ------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# ------------------------------------------------------------

module load StdEnv/2020
module load picard/2.26.3

inputdir=/path/to/f2/file/directory

cd $inputdir

filename=`ls -1 *.bam | tail -n +${SLURM_ARRAY_TASK_ID} | head -1`

java -Xmx50g -XX:ParallelGCThreads=5 -jar $EBROOTPICARD/picard.jar ValidateSamFile \
I=$filename\
MODE=SUMMARY \
MAX_OPEN_TEMP_FILES=1000 \
MAX_RECORDS_IN_RAM=50000

# ------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ------------------------------------------------------------
```

Create mpileup file type for downstream software - Samtoo...

10 Need to create a mpileup file for downstream analysis. Need an mpileup for all pools and one for each pool individually.

10.1 All pools

```
#!/bin/bash
# ------------------------------------------------------------
# Samtools mpileup
# ------------------------------------------------------------
#SBATCH --job-name=allpools_mpileup
#SBATCH --account=def-account
#SBATCH --cpus-per-task=16
```

```
#SBATCH --time=0-06:00
#SBATCH --mem=1G
# ------------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# ------------------------------------------------------------------

module load StdEnv/2020
module load gcc/9.3.0
module load samtools/1.13

cd /path/to/f2/file/directory

samtools mpileup -B XX_f2.bam YY_f2.bam ZZ_f2.bam \
-o all_pools_ordered.mpileup

# ------------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ------------------------------------------------------------------
```

10.2 Individual files for each pool

```
#!/bin/bash
# ------------------------------------------------------------------
# Samtools individual mpileups
# ------------------------------------------------------------------
#SBATCH --job-name=ind_mpileups
#SBATCH --account=def-account
#SBATCH --cpus-per-task=8
#SBATCH --time=0-06:00
#SBATCH --mem=1G
#SBATCH --array=1-8
# ------------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# ------------------------------------------------------------------

module load StdEnv/2020
module load gcc/9.3.0
module load samtools/1.13

inputdir=/path/to/f2/file/directory

cd $inputdir

filename=`ls -1 *.bam | tail -n +${SLURM_ARRAY_TASK_ID} | head -1`
filename2=${filename::-12}
```

```
samtools mpileup -B $filename -o ${filename2}.mpileup

# ----------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ----------------------------------------------------------------
```

Pairwise FST - Poolfstat

11 Calculating pairwise FST between the pools. If you have a high quality reference genome Poolfstat can do other analyses as well.

11.1 Create sync file from mpileup

Need to use PoPoolation 2 to do this - https://sourceforge.net/p/popoolation2/wiki/Main/

```
#!/bin/bash
# ----------------------------------------------------------------
# mpileup to sync
# ----------------------------------------------------------------
#SBATCH --job-name=mpileup_sync
#SBATCH --account=def-account
#SBATCH --cpus-per-task=8
#SBATCH --time=0-02:00
#SBATCH --mem=15G
# ----------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# ----------------------------------------------------------------

module load StdEnv/2020
module load java/17.0.2

java -Xmx10g -jar /path/to/popoolation2_1201/mpileup2sync.jar \
--input /path/to/mpileup/file/all_pools_ordered.mpileup \
--output ./all_pools_ordered.sync \
--fastq-type sanger --min-qual 20 --threads 8

# ----------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ----------------------------------------------------------------
```

11.2 Pairwise FST using Poolfstat

This is a R script that I ran on the cluster

R script (poolfstat.R)

```
library(poolfstat)

pool_file <- popsync2pooldata((sync.file="./all_pools_ordered.sync"),
poolsizes =c(40,20,40,39,40,40,40,40),
poolnames = c("XX","YY","ZZ"),
min.rc = 1, min.cov.per.pool = 20, max.cov.per.pool = 200, min.maf = 0.05, noindel = TRUE)

save(pool_file, file = "filename.RData")

PW_fst <- compute.pairwiseFST(pool_file, method = "Anova", min.cov.per.pool = 20,
max.cov.per.pool = 200, min.maf = 0.05, output.snp.values = TRUE)

save(PW_fst, file = "PW_fst.RData")

write.csv(PW_fst@PairwiseFSTmatrix, file = "PW_fst.csv")
```

SLURM

```
#!/bin/bash
# ----------------------------------------------------------------
# poolfstat calcs
# ----------------------------------------------------------------
#SBATCH --job-name=poolfstat
#SBATCH --account=def-account
#SBATCH --cpus-per-task=1
#SBATCH --time=0-05:00
#SBATCH --mem=15G
# ----------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# ----------------------------------------------------------------

module load StdEnv/2020
module load r/4.2.1

Rscript --vanilla --verbose poolfstat.R > slurm-${SLURM_JOBID}.Rout 2>&1

# append logfile to this scripts logfile
cat slurm-${SLURM_JOBID}.Rout >> slurm-${SLURM_JOBID}.out

# remove Rout log
```

```
rm slurm-${SLURM_JOBID}.Rout

# ------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# ------------------------------------------------------------
```

11.3 Heatmap in R

```
library(ComplexHeatmap)
library(circlize)

FST <- read.csv("PW_fst.csv")
str(FST)
FST[1:5,1:5]

row.names(FST) < - FST$X
FST[1:5,1:5]

FST < - acuta.FST[,-1]
FST[1:5,1:5]

FST.m <- as.matrix(acuta.FST)

col_fst_grey <- colorRamp2(c(0, 0.25, 0.5), c("grey99", "grey78", "grey65"))

tiff("FST.heatmap_with_num_no_colour.tiff", width = 11, height = 8.5, units = 'cm', res=300)
Heatmap(FST.m, col = col_fst_acuta_grey, heatmap_legend_param = list(title = "Pairwise Fst",
at = c(0, 0.1, 0.2, 0.3, 0.4, 0.5), labels = c("0.00","0.10","0.20","0.30","0.40","0.50")), cell_fun =
function(j, i, x, y, width, height, fill) {
    grid.text(sprintf("%.2f", acuta.FST.m[i, j]), x, y, gp = gpar(fontsize = 5))
})
dev.off()
```

Nucleotide diversity - PoPoolation 1

12 Calculating nucleotide diversity for each pool using the same regions across pools.

12.1 Nucleotide diversity of each individual pool

Repeat for each pool (can do as a batch job)

```
#!/bin/sh
#------------------------------------------------------------
# Pi
```

```
# -----------------------------------------------------------------
#SBATCH --job-name=XX_pi
#SBATCH --account=def-account
#SBATCH --cpus-per-task=1
#SBATCH --time=0-15:00
#SBATCH --mem=500MB
# -----------------------------------------------------------------
echo"Current working directory: `pwd`"
echo"Starting run at: `date`"
# -----------------------------------------------------------------

module load StdEnv/2020
module load perl/5.30.2

cd /path/to/ind_mpileup/file/directory

perl /path/to/popoolation_1.2.2/Variance-sliding.pl \
--measure pi --input XX.mpileup --output XX_pi.file --snp-output XX_pi.snps \
--fastq-type sanger --pool-size 20 --min-count 4 --min-coverage 20 \
--max-coverage 200 --window-size 250 --step-size 250

# -----------------------------------------------------------------
echo"Job finished with exit code $? at: `date`"
# -----------------------------------------------------------------
```

12.2

```
XX_pi_file <- read.delim("XX_pi.file", header = FALSE)

col_names <- c("chr", "position", "Num.of.SNPs", "frac.of.cov", "pi")
colnames(XX_pi_file) <- col_names

colnames(XX_pi_file)[5] <- c("XX_pi")

colnames(XX_pi_file)[3] <- c("XX_num_snps")

XX_pi_file$ID <- paste(XX_pi_file$chr, XX_pi$position, sep = '_')

XX_pi_file$XX_pi <- as.numeric(XX_pi_file$XX_pi)

#Repeat for all pools

library(tidyverse)
pi_all <- (list(XX_pi_file, YY_pi_file, ZZ_pi_file) %>% reduce(inner_join, by='ID'))

library(dplyr)
pi_all.2 <- pi_all %>% select(-contains(c("frac","position","chr")))

pi_all.2 <- pi_acuta_all.2 %>% relocate(ID, .before = XX_num_snps)
```

```
head(pi_all.2)

pi_all.2_no.na <- na.omit(pi_all.2) #goes from > 2 million sections to 366,091 windows

XX_avg_pi <- mean(pi_all.2$XX_pi, na.rm=T)
XX_avg_pi.no.na <- mean(pi_all.2_no.na$XX_pi, na.rm=T)
XX_num_snps <- sum(pi_all.2_no.na$XX_num_snps, na.rm=T)
```

12.3 t test between species

```
avg.pi.t.test <- t.test(john$Average_pi_from_shared_windows,
gyrina$Average_pi_from_shared_windows, var.equal=TRUE)

tiff("Avg_pi_div_no_colour.tiff", width = 11, height = 8.5, units = 'cm', res=600)
ggplot(pi_snp_acuta, aes(x =Species, y =Average_pi_from_shared_windows)) +
geom_boxplot(aes(fill =Species)) + geom_jitter(shape=16, position=position_jitter(0.1)) +
theme(legend.position="none") + scale_fill_manual(values=c("grey", "grey")) + theme_classic()
+ labs(x="Species", y = "Average nucleotide diversity") + theme(legend.position="none") +
theme(axis.text.y =element_text(colour="black", size=10))
dev.off()
```

12.4 Linear regression between population minimum and nucleotide diversity

```
pi_min_pop <- read.csv("genetic_div_min_pop.csv") #I created this file in excel using the
above info and known pop minimum counts

plot(pi_min_pop$Average_pi_from_shared_windows, pi_min_pop_J$Recorded_min) #no trend

lm_pi <- lm(pi_min_pop_J$Average_pi_from_shared_windows ~ pi_min_pop_J$Recorded_min,
pi_min_pop)

confint(lm_pi, "pi_min_pop$Recorded_min", level=0.95)

library(ggpubr)
tiff("Pop_min_vs_pi_J_no_colour.tiff", width = 6, height = 5.4, units = 'in', res=300)
ggplot(pi_min_pop_J, aes(x=Recorded_min, y=Average_pi_from_shared_windows, label=Site))
+ geom_point(colour="black") + geom_text(nudge_x = 20)  + theme_classic() + labs(x =
"Lowest recorded population minimum (1996 to 2017)", y="Average pi across shared
windows") + theme(legend.position="none") + theme(axis.text.y =element_text(colour="black",
size=10), axis.text.x = element_text(colour="black", size=10)) + geom_smooth(method='lm',
color = "black", size = 0.5, se=TRUE) + stat_cor(label.y = 0.0042) + scale_y_continuous(labels =
scales::number_format(accuracy = 0.0001)) + scale_x_continuous(breaks=c(0, 50, 100, 150,
200, 250, 300, 350, 400), limits = c(0, 450))
dev.off()
```