Data gathering

Before we get into everything, let's make sure we have all the data needed for this part of the module. We will need to have the whole genome sequencing data, the index for error correction, and some pre-generated assemblies and BUSCO results (just to keep things moving). If you haven't done so already, please navigate to scratch (cd /scratch) and we'll create the necessary folders and get data moved over.

```
cd /scratch
iget -rPT /iplant/home/shared/Botany2020NMGWorkshop/Genome_assembly/sorted_bam
iget -rPT /iplant/home/shared/Botany2020NMGWorkshop/Genome_assembly/SNP_calling
iget -rPT /iplant/home/shared/Botany2020NMGWorkshop/raw_data/Ugibba/transcriptome
```

Now that you have a genome assembly, even a draft version, there are many things that you can do with this. One is to use it in population genetic studies as a reference to call SNPs using a RAD-Seq, Hyb-Seq, or whole genome resequencing approaches. Because we are not generating any new data here, we are going to be using some RNA-Seq data to call SNPs using the Stacks pipeline. For nonRAD-Seq data other programs such as GATK, FreeBayes, or Samtools might be better to use but they can be more difficult to learn in a short-time frame and can take a lot longer to run. However, with a reference approach, all these methods share an initial step, mapping the cleaned reads of interest to the reference genome.

Before we get started, this is just a recap of a few of the linux commands that you may need to use in this tutorial:

```
#make new directory
mkdir

#change directory
cd

#check the use of processors
top

#check amount of RAM available
free -m

#check the amount of storage left on your partition
df

#check the size of folders in a directory
du -sh *
```

SNP Calling

Before read mapping, the first thing to do is download the data of interest and/or make sure the fastq files you have are cleaned and ready to be used. To download raw sequencing data, one of the easiest methods is to use the ENA database and wget and then to rename the file to something that is easier to read (the data has already been downloaded for the workshop, but here is an example of what you would do on your own) –

```
wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR106/046/SRR10676746/SRR10676746.fastq.gz
mv SRR10676746.fastq.gz Ugibba_bladderR1.fastq.gz
```

To clean up the data, either your own or downloaded data sets, the easiest and fastest way is using the program fastp. The parameters here are for single end reads input, output, Illumina adapater sequence, minimum quality score, minimum length to keep, and number of threads. This step has already been done on the data we will use, so no need to do it again here.

```
fastp -I Read_R1.fastq.gz -o Read_cleaned_R1.fastq.gz -z 4 –
adapter_sequence=AGATCGGAAGAGCACACGTCTGAACTCCAGTCA –
adapter_sequence_r2=AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGT -q 20 –length_required
100 –thread 8
```

Now that you have cleaned reads ready to go, we will be using the program BWA to map the reads to the reference, specifically using the bwa mem function. To save some time, we will use a pruned genome file that only contains four chromosomes instead of the full assembly. The first step is to index the reference fasta file to make it easier to navigate and locate read locations:

```
cd SNP_calling
scp /scratch/completed_assemblies/Ugibba_pruned_assembly.fasta .

bwa index Ugibba_pruned_assembly.fasta
```

With the reference now indexed, one at a time we will map the reads from the RNA-Seq data sets to the reference. When using multiple samples, it is advisable to use read group information to make it easier to filter or subset the data later on. The read group information consists of four different types of information:

Read group information
ID: is unique identifier of the samples, for now doing the sample name and the barcode info
SM: is the sample name
PL: is the sequencing equipment, in almost all cases this will be Illumina
PU: is the run identifier, the lane, followed by the specific barcode of the sample
LB: is the library count or other relevant information

The mapping script below calls the program, enforces read group information, specifies the reference fasta file, the sequencing data, and then the output.

```
bwa mem -t 6 -R "@RG\tID:bladderR1\tSM:Rep1\tPL:HiSeq\tPU:HTNMKDSXX\tLB:RNA-Seq"
Ugibba_pruned_assembly.fasta /scratch/transcriptome/Ugibba_bladderR1.fastq.gz >
Ugibba_bladderR1.sam
```

The output file is a SAM file. To save memory requirements, we will convert this to a binary format and then sort with Samtools. You can do this individually for each file, but to make it easier we have a script set up to run through each of RNA-Seq data files, map to the reference, convert to binary, and then sort. At the end all of your sorted BAM files can be found in a new folder “sorted_bam”. To run this script, just type:

```
BWA_Uggiba.sh
```

With the BAM files for all the samples, we are ready to start calling SNPs. One additional piece of information that we need to supply is a population map for Stacks. This lets the program know what samples to use/expect, and if any are found in the same population. This doesn’t impact the actual SNP calling but will impact the exporting of some result files. The population map looks like this:

```
Ugibba_bladderR1      bladder
Ugibba_rhizoidR1      rhizoid
```

We will use the ref_map.pl wrapper to invoke all the necessary components of Stacks. See their online manual for a better idea of what each step is doing. First we make a new directory for all the output files, then specify the folder where the BAM files to use are, the population map, and then the output directory:

```
mkdir ref_SNPs
scp /
/opt/stacks-2.53/scripts/ref_map.pl --samples /scratch/sorted_bam --popmap
population_map.txt -o ref_SNPs/ -T 6
```

Once the run finish, we will need to export the results to a VCF file using the Stacks function populations:

```
/opt/stacks-2.53/populations --batch_size 1 -P ref_SNPs/ -M population_map.txt -t 6 --
ordered_export --vcf
```

The exported file is unfiltered and ordered based on chromosome and position. Stacks does implement some filtering capacity, but those methods are a bit limited. To better filter the SNP data set, we will use VCFtools. We will need to install the program first since it wasn’t installed previously. First navigate to your scratch directory and follow the steps below:

```
cd /scratch
mkdir Installed_programs
```

```
cd Installed_programs
git clone https://github.com/vcftools/vcftools.git

./autogen.sh
./configure --prefix=/scratch/Installed_programs/vcftools
make
make install

/scratch/Installed_programs/vcftools/bin/vcftools --help
```

to filter at loci with more than 40% missing data, keep only sites with a coverage between 3-100, keep a minor allele frequency of at least 0.05, and then recode the header information after filtering.

```
cd /scratch/SNP_calling/ ref_SNPs
/scratch/Installed_programs/vcftools/bin/vcftools --vcf populations.snps.vcf --max-missing 0.6 --min-meanDP 3 --max-meanDP 100 --maf 0.05 --mac 3 --recode --recode-INFO-all --out Ugibba_SNPs_filtered
```

```
#initial pass throwing out all SNPs that are missing 60%
/scratch/Installed_programs/vcftools/bin/vcftools --vcf populations.snps.vcf --max-missing 0.4 --min-alleles 2 --max-alleles 2 --recode --recode-INFO-all --out Ugibba_first_pass

#gives missing proportion of loci for each individual
/scratch/Installed_programs/vcftools/bin/vcftools --vcf Ugibba_first_pass.recode.vcf --missing-indv

#average depth for each individual
/scratch/Installed_programs/vcftools/bin/vcftools --vcf Ugibba_first_pass.recode.vcf --depth

#observed and expected heterozygosity
/scratch/Installed_programs/vcftools/bin/vcftools --vcf Ugibba_first_pass.recode.vcf --het

#create a list of individuals with at least 50% missing data
awk '$5 > 0.75' out.imiss | cut -f1 > lowDP50.indv

/scratch/Installed_programs/vcftools/bin/vcftools --vcf Ugibba_first_pass.recode.vcf --max-missing 0.4  --remove lowDP50.indv --recode --recode-INFO-all --out Ugibba_filtered_SNPs

#gives missing proportion of loci for each individual
/scratch/Installed_programs/vcftools/bin/vcftools --vcf Ugibba_filtered_SNPs.recode.vcf --missing-indv
```

```
#average depth for each individual
/scratch/Installed_programs/vcftools/bin/vcftools --vcf Ugibba_filtered_SNPs.recode.vcf --depth

#observed and expected heterozygosity
/scratch/Installed_programs/vcftools/bin/vcftools --vcf Ugibba_filtered_SNPs.recode.vcf –het
```

Before reading this file back into Stacks to output our final files we need to adjust the chromosome names. This normally won't need to be done, but it will make some of the downstream analyses easier. To do this will use sed for find and replace, followed by writing a new file:

```
sed 's/ENA|CM007989|CM007989.1/Chr1/g' Ugibba_filtered_SNPs.recode.vcf | sed 's/ENA|CM007990|CM007990.1/Chr2/g' | sed 's/ENA|CM007991|CM007991.1/Chr3/g' | sed 's/ENA|CM007992|CM007992.1/Chr4/g' > Ugibba_filtered_SNPs_renamed.recode.vcf
```

We will now read the filtered SNP file back into Stacks to produce multiple files that can be used for downstream analyses:

```
mkdir ref_final_data_files
/opt/stacks-2.53/populations  --batch_size 1 -V Ugibba_filtered_SNPs_renamed.recode.vcf -O ref_final_data_files/ -M population_map.txt -t 6 --ordered-export --fstats --vcf --treemix --plink --structure --radpainter --genepop --fasta_samples
```

Principal Component Analysis

An easy downstream analysis that can be done to start looking at the population genetics of your samples is a Principal Component Analysis, or PCA. For this, we will be using R and specifically the R package 'SNPRelate'. The inputs for this are the filtered VCF file, a population map telling how the populations are related, and a pop file that lists in order the populations in the population map. The pop map for the PCA looks similar to that as the Stacks population map:

```
sample.id	pop
Ugibba_bladderR1	bladder
Ugibba_rhizoidR1	rhizoid
```

And the pop file is just the population column with no header:

```
bladder
rhizoid
bladder
```

Calling the appropriate files is all included in the R script, as well as code for plotting PC1 vs PC2, and PC2 vs PC3. To run the script, type:

```
Rscript SNPRelate.R
```