1. Running Bowtie2

- Reminder to take notes, share info here: [Etherpad]
- For this part of the tutorial, we're each going to work with a single long scaffold from the Red Siskin assembly.
  - They are here: `/data/genomics/dikowr/SMSC/assembly_scaffolds`
  - Copy one of these to a new directory (PopGen) in your `/pool/genomics/` space.

- First we'll need to index the fasta file before we can map raw data to it. Create a job file for this:

  - **module**: `bioinformatics/bowtie2`
  - **command**:\` `bowtie2-build --threads $NSLOTS <SCAFFOLD.fasta> NAME`
  - Hint: it doesn't need much RAM or time.

- Then we can start mapping the reads to our scaffolds.

  - There are 9 individuals plus the one from which we are assembling the genome.
  - The read data are here: `/data/genomics/dikowr/SMSC/resequence_data`
  - Don't copy it to your space.
  - Create job files to map each individual's reads to your scaffold.
  - **module**: `bioinformatics/bowtie2`
  - Here is an example **command** (this will need editing to make it work for your data):
    `bowtie2 --very-sensitive -N 1 -I 100 -X 600 -x siskin -p $NSLOTS --phred33 --rg-id "MB-12866" --rg SM:"MB-12866" --rg PL:"ILLUMINA" -`
  - Check out the output file using `head`

2. Manipulating the Bowtie2 output

- In order to get the Bowtie2 outputs ready to go for variant calling, we'll have to do some file manipulation.
- First, we'll have to convert our sam file output to bam format.
- Create a job file to do this.
  - **module**: `bioinformatics/samtools/1.6`
  - **command**: `samtools view -b <YOUR_OUTPUT.sam> > <YOUR_OUTPUT.bam>`
  - Try `head` on the output bam file to see what happens.

- Next we'll need to sort the bam file.
  - **module**: `bioinformatics/samtools/1.6`
  - **command**: `samtools sort <YOUR_BAM.bam> -o <YOUR_BAM_sorted.bam>`

3. Mark Duplicates with picard-tools

- For future steps, we will need an sequence dictionary and fasta index.
- Create a job file to create a fasta index on your scaffold:
  - **module**: `module load bioinformatics/samtools/1.6`
  - **command**: `samtools faidx <YOUR_SCAFFOLD.fasta>`

- Create a job file to create a sequence dictionary:
  - **module**: `module load bioinformatics/picard-tools/2.5.0`
  - **command**: `runpicard CreateSequenceDictionary R=<YOUR_SCAFFOLD.fasta> O=<YOUR_SCAFFOLD.dict>`

- Now we can create a job file to mark duplicates:
  - **module**: `module load bioinformatics/picard-tools/2.5.0`
  - **command**: `runpicard MarkDuplicates I=<YOUR_SORTED_BAM.bam> O=<YOUR_SORTED_BAM_marked.bam> M=marked_dup_metrics_SAMPLE_NAME.txt`

4. Realign indels before calling variants

- Now we'll look for regions with indels in our bam files and realign them.
- Create a job file to use GATK RealignerTargetCreator:
  - **module**: `module load bioinformatics/gatk/3.7`
  - **command**:
    `rungatk -T RealignerTargetCreator -R <YOUR_SCAFFOLD.fasta> -I <YOUR_SORTED_BAM_marked.bam> -o <YOUR_SORTED_BAM_marked.bam>.list`

- Create a job file to use GATK IndelRealigner:
  - **module**: `module load bioinformatics/gatk/3.7`
  - **command**:
    `rungatk -T IndelRealigner -R <YOUR_SCAFFOLD.fasta> -I <YOUR_SORTED_BAM_marked.bam> -o indels-<YOUR_SORTED_BAM_marked.bam> -targetInte`

- Index the marked bam files:
  - **module**: `module load bioinformatics/samtools/1.6`
  - **command**: `samtools index <YOUR_SORTED_BAM_marked.bam>`

5. Call variants with GATK

- First, we will run GATK HaplotypeCaller on each bam file individually:
  - **module**: `module load bioinformatics/gatk/3.7`
  - **command**:
    `rungatk -T HaplotypeCaller --emitRefConfidence GVCF --variant_index_type LINEAR --variant_index_parameter 128000 -R <YOUR_SCAFFOLD.fa`
  - This job might need to be run on himem.

- Now we'll run GATK GenotypeGVCFs across all vcf files:
  - **module**: `module load bioinformatics/gatk/3.7`
  - **command**:
    `rungatk -T GenotypeGVCFs -variant_index_type LINEAR -variant_index_parameter 128000 -R <YOUR_SCAFFOLD.fasta> -V HERE_LIST_ALL_YOUR_VC`
  - This job might need to be run on himem.

- Use `head` to look at the VCF file.