Papaya SNP calling from WGRS pipeline

Leela Manoharan & Ido Bar

21 August 2025

## Experimental Design

Fresh leaf samples were collected from XX genotypes of papaya (*Carica papaya*) from Rocky Top farm in North Queensland in 2023-24 and were stored in liquid Nitrogen and were stored in -80°C until DNA extraction. DNA was extracted from the leaf samples using XXX kit (check with Josh if it was Qiagen DNeasy or another kit). The extracted DNA was shipped in dry ice to the State Agricultural Biotechnology Centre ([SABC](https://www.murdoch.edu.au/research/sabc)) at Murdoch University (led by Prof. Rajeev Varshney). DNA libraries were prepared for whole-genome resequencing (WGRS) on an MGI DNBSEQ-T7, producing 150 bp paired-end reads.

### Samples used in this study

sabc\_metadata <- read\_tsv('data/papaya\_wgrs\_Murdoch\_files.txt')   
sabc\_meta\_sum <- sabc\_metadata %>%   
 mutate(genotype=sub("\\.E2.+", "", filename)) %>%   
 group\_by(genotype) %>%   
 summarise(total\_size\_mb = sum(size)/1e6, file\_num=n())  
write\_xlsx(sabc\_meta\_sum, "data/papaya\_WGRS\_SRA\_metadata.xlsx", sheet = "SABC\_WGRS\_sum",  
 overwritesheet=TRUE)  
  
sra\_seq\_data <- read\_excel("data/Cpapaya\_PRJNA727683\_sra\_wgs.xlsx", sheet = 'Cpapaya\_PRJNA727683\_sra\_wgs') %>%   
 clean\_names() %>% mutate(sample\_title = gsub("\\s+", "", sample\_title))  
  
sra\_sample\_data <- read\_excel("data/Cpapaya\_PRJNA727683\_sra\_wgs.xlsx", sheet = 'Accession\_table') %>%   
 clean\_names() %>%   
 mutate(cultivar\_name=sub("‘\*(.+)'", "\\1", cultivar\_name),  
 name = gsub("\\s+", "", name))  
  
# sra\_seq\_data$sample\_title[!sra\_seq\_data$sample\_title %in% sra\_sample\_data$name]  
  
sra\_combined <- sra\_seq\_data %>%   
 select(name=sample\_title, total\_size\_mb, sra\_accession) %>%   
 left\_join(sra\_sample\_data)  
  
#sra\_combined %>% filter(is.na(sex))  
# sra\_combined %>%   
# mutate(sra\_accession = glue::glue("[{sra\_accession}](https://www.ncbi.nlm.nih.gov/sra/{sra\_accession})"),  
# sra\_accession = map(sra\_accession, gt::md),  
# total\_size\_mb = comma(total\_size\_mb)) %>%  
# clean\_names(case="title", abbreviations="SRA") %>%   
# gt()  
  
  
# sra\_combined %>%   
# mutate(sra\_accession = glue::glue("<a href=https://www.ncbi.nlm.nih.gov/sra/{sra\_accession}>{sra\_accession}</a>")  
# # total\_size\_mb = comma(total\_size\_mb),   
# # sra\_accession = map(sra\_accession, gt::md)  
# ) %>%  
# clean\_names(case="title", abbreviations="SRA") %>%  
# formattable(.,  
# list(  
# `Total Size Mb` = color\_bar(customGreen)  
# ))  
  
# sra\_combined %>%   
# mutate(sra\_accession = glue::glue("[{sra\_accession}](https://www.ncbi.nlm.nih.gov/sra/{sra\_accession})"),  
# sra\_accession = map(sra\_accession, gt::md)) %>%   
# # total\_size\_mb = comma(total\_size\_mb)) %>%  
# clean\_names(case="title", abbreviations="SRA") %>%   
# reactable(.,  
# columns = list(  
# "Total Size Mb" = colDef(  
# cell = data\_bars(.,  
# fill\_color = viridis(5),  
# background = "lightgrey",  
# text\_position = "inside-end",  
# max\_value = 15000)  
# )  
# ))

## Aims

1. Assess the genetic diversity of papaya genotypes in the Australian collection

* Align the sequencing reads to a reference genome (Solo Sunset or Holland\_5)
* Call variants (SNPs)
* Infer phylogeny and diversity

1. Identify variants associated with productivity and fruit quality traits

* Assemble a pangenome (or use the one from Murdoch)
* Align WGRS reads to the pangenome using vg-giraffe
* Call variants
* Perform genome wide association study (GWAS) for a range of traits

## Methods

### Overview of Analysis Pipeline

1. Obtain sequencing data from public repositories (NCBI SRA) or from SABC
2. Data pre-processing:
   1. Quality check
   2. Adaptor trimming
   3. Post-trim quality check(?)
3. Mapping reads to a reference genome
4. Reads deduplication and read group addition
5. Variant calling and filtration
6. Population genetics analysis (diversity and phylogeny)

Sequencing data processing, mapping and variant calling were performed on the *QCIF Bunya* (using Slurm scheduler, see [documentation](https://github.com/UQ-RCC/hpc-docs/tree/main)) and in [GalaxyAU](https://usegalaxy.org.au).

### Setup compute environment

Install needed software in a conda environment on the HPC cluster (we will install a [Miniforge distribution](https://github.com/conda-forge/miniforge), which has mamba already installed - see [mamba docs](https://mamba.readthedocs.io/en/latest/installation/mamba-installation.html)).

alias start\_interactive\_job='salloc --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 --mem=50G --job-name=interactive-job --time=05:00:00 --partition=general --account=a\_agri\_genomics srun --export=ALL,PATH,TERM,HOME,LANG --pty /bin/bash -l'  
start\_interactive\_job  
  
mkdir -p ~/bin  
  
# Prepare a general array Slurm script  
echo '#!/bin/bash --login  
#SBATCH --nodes=1  
#SBATCH --ntasks=1  
#SBATCH --output=%x.%A.%a.log'"  
#SBATCH --account=a\_agri\_genomics  
#SBATCH --partition=general  
  
set -Eeo pipefail  
source ~/.bashrc  
conda activate \$CONDA\_NAME  
cd \$SLURM\_SUBMIT\_DIR  
gawk -v ARRAY\_IND=\$SLURM\_ARRAY\_TASK\_ID 'NR==ARRAY\_IND' \$CMDS\_FILE | bash" > ~/bin/array.slurm  
  
echo '#!/bin/bash --login  
#SBATCH --nodes=1  
#SBATCH --ntasks=1  
#SBATCH --output=%x.%j.log'"  
#SBATCH --account=a\_agri\_genomics  
#SBATCH --partition=general  
  
set -Eeo pipefail  
source ~/.bashrc  
conda activate \$CONDA\_NAME  
cd \$SLURM\_SUBMIT\_DIR  
bash \$CMDS\_FILE" > ~/bin/serial\_jobs\_run.slurm  
  
# Prepare a parallel jobs Slurm script  
echo '  
cat $CMDS\_FILE | parallel' | cat <(head -n -1 ~/bin/serial\_jobs\_run.slurm) - > ~/bin/parallel\_jobs\_run.slurm  
  
  
# download and install miniforge conda and environment  
JOBNAME="install-conda"  
NCORES=6  
MEM=32  
WALLTIME=5:00:00  
CONDA\_NAME="genomics"  
echo "wget \"https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-$(uname)-$(uname -m).sh\"  
bash Miniforge3-$(uname)-$(uname -m).sh  
# accept defaults and let conda initialise  
# initialise conda  
source ~/.bashrc  
# add channels and set priorities  
conda config --add channels conda-forge  
conda config --append channels bioconda  
  
# install extra packages to the base environment  
mamba install -n base libgcc gnutls libuuid readline cmake git tmux libgfortran parallel mamba gawk pigz rename genozip autoconf sshpass gh perl-text-csv  
# install snippy (need to fix internet connection to gowonda2 - use patched netcheck in ~/bin)  
# source ~/.proxy  
  
mamba create -n $CONDA\_NAME sra-tools bcbio-gff libgd xorg-libxpm \  
 libpng libjpeg-turbo jpeg snpsift biobambam bwa-mem2 sambamba \  
 libtiff qualimap multiqc bbmap fastp freebayes bedops \  
 entrez-direct parallel-fastq-dump mosdepth  
# Clean extra space  
# conda update -n base conda  
conda clean -y --all" > $JOBNAME.cmds  
  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$JOBNAME.cmds,CONDA\_NAME=base ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")

### Data preparation

Create a folder for our processing ($RUN\_DIR) and prepare (create accessory index files) the reference genome (Holland\_5 - PAPHI003, Haplotype 2).

# download the SunUp genome from SharePoint  
rclone copy -P Papaya\_genomics:Genomic\_Resources/SunUp\_SunSet\_genomes\_Nature\_paper/SunUp /scratch/project/adna/Papaya/SunUp\_reference\_genome  
# start an interactive job on Bunya  
alias start\_interactive\_job='salloc --nodes=1 --ntasks-per-node=1 --cpus-per-task=10 --mem=50G --job-name=interactive --time=05:00:00 --partition=general --account=a\_agri\_genomics srun --export=PATH,TERM,HOME,LANG --pty /bin/bash -l'  
start\_interactive\_job  
WORK\_DIR="/scratch/project/adna/leela"  
  
CONDA\_NAME="genomics"  
conda activate $CONDA\_NAME  
# Prepare the commands  
REFERENCE="/scratch/project/adna/Papaya/SunUp\_reference\_genome/GWHBFSC00000000" # Holland\_5 genome from Murdoch on Bunya HPC  
cd $WORK\_DIR  
# pigz -cd $REF\_DIR/ArME14.fasta.gz > $GENOME.fa  
pigz -cd $REFERENCE.gff.gz > $REFERENCE.gff3  
gff2bed < $REFERENCE.gff3 > $REFERENCE.bed  
zcat $REFERENCE.genome.fasta.gz | bgzip -c -I $REFERENCE.fasta.gz.gzi -@ $SLURM\_CPUS\_PER\_TASK > $REFERENCE.fasta.gz && samtools faidx $REFERENCE.fasta.gz && bwa-mem2 index $REFERENCE.fasta.gz   
  
# zcat $REFERENCE.fasta.gz | bgzip -c > PAPHI003\_hap2.fasta.gz  
# samtools faidx --fai-idx PAPHI003\_hap2.fasta.fai --gzi-idx PAPHI003\_hap2.fasta.gz.gzi PAPHI003\_hap2.fasta.gz

Download the raw papaya whole-genome resequencing files (.fastq.gz) from NCBI SRA (WGS libraries from [PRJNA727683](https://www.ncbi.nlm.nih.gov/bioproject/PRJNA727683/)). Reference genomes of cultivars Solo SunUp (GMO for PRSV resistance, accession [GWHBFSC00000000](https://ngdc.cncb.ac.cn/gwh/Assembly/23161/show), corresponding to NCBI accession [GCA\_021527605.1](https://www.ncbi.nlm.nih.gov/datasets/genome/GCA_021527605.1/), but including gene annotation) and Sunset (accession [GWHBFSD00000000](https://ngdc.cncb.ac.cn/gwh/Assembly/23162/show), corresponding to NCBI [GCA\_022788785.1](https://www.ncbi.nlm.nih.gov/datasets/genome/GCA_022788785.1/), but including gene annotation; Yue *et al.* (2022)) were downloaded from the [Genome Warehouse](https://ngdc.cncb.ac.cn/gwh).

# create a working directory  
WORK\_DIR="/scratch/project/adna/leela"  
mkdir -p $WORK\_DIR  
cd $WORK\_DIR  
# download Papaya sequencing files from SRA  
JOBNAME="download-papaya-SRA"  
NCORES=6  
MEM=32  
WALLTIME=5:00:00  
CONDA\_NAME="genomics"  
conda activate $CONDA\_NAME  
# get WGS information from the SRA   
esearch -db sra -query PRJNA727683 | efetch -format runinfo > Cpapaya\_PRJNA727683\_SRA\_runinfo.csv  
cat Cpapaya\_PRJNA727683\_SRA\_runinfo.csv | gawk -F "," '$13=="WGS"' > Cpapaya\_PRJNA727683\_SRA\_WGS\_runinfo.csv  
  
# cat Cpapaya\_PRJNA727683\_SRA\_WGS\_runinfo.csv | cut -f 1 -d "," | parallel --dry-run "prefetch {}; parallel-fastq-dump --split-3 -t \$SLURM\_CPUS\_PER\_TASK --sra-id {}/{}.sra --gzip -O {}" > $JOBNAME.cmds  
  
# submit to the cluster  
# ARRAY\_ID=$(sbatch -a 1-$(cat $JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")

### Mapping to the reference genome

Reads were mapped to the *C. papaya* SunUp reference genome assembled and annotated by the Fujian Agriculture and Forestry University (Yue *et al.* 2022), using bwa-mem2 v2.2.1 (Vasimuddin *et al.* 2019). The alignment files were then coordinate-sorted and PCR duplicates were marked using the bamsormadup command from BioBamBam2 v2.0.183 (Tischler and Leonard 2014).  
Mapping quality and coverage were assessed with Qualimap v.2.2.2-dev (Okonechnikov *et al.* 2016) and Mosdepth v0.3.10 (Pedersen and Quinlan 2018) and were consolidated along with quality-trimming measures into a single, interactive report for each batch using MultiQC v1.21 (Ewels *et al.* 2016).

# setup workspace  
CONDA\_NAME="genomics"   
WORK\_DIR="/scratch/project/adna/leela"  
# save the run date  
DATE=$(date +%d\_%m\_%Y)  
RUN\_DIR="$WORK\_DIR/FB\_SNP\_calling\_${DATE}"  
# BAM\_DIR="$RUN\_DIR/aligned\_reads"  
mkdir -p $RUN\_DIR && cd $RUN\_DIR  
NCORES=12  
MEM=96  
WALLTIME="5:00:00"  
RGPM="Illumina NovaSeq 6000"  
RGPL="Illumina"  
JOBNAME="SRA-fastp-bwa"  
  
# fields in SRA csv file: ID:{1},SM:{30},PL:{19},PM:{20}  
cat $WORK\_DIR/Cpapaya\_PRJNA727683\_SRA\_WGS\_runinfo.csv | parallel --csv --dry-run " prefetch {1} && cd {1} && parallel-fastq-dump --split-3 -t \$SLURM\_CPUS\_PER\_TASK --sra-id {1}.sra --gzip && fastp -i {1}\_1.fastq.gz -I {1}\_2.fastq.gz --detect\_adapter\_for\_pe -c -l 30 -p -w \$SLURM\_CPUS\_PER\_TASK -z 7 -o {1}\_R1.trimmed.fastq.gz -O {1}\_R2.trimmed.fastq.gz -j {1}.fastp.json -h {1}.fastp.html && bwa-mem2 mem -R \"@RG\tID:{1}\tSM:{30}\tLB:{30}\tPL:{19}\tPM:{20}\" -t \$[SLURM\_CPUS\_PER\_TASK - 2] $REFERENCE.fasta.gz {1}\_R1.trimmed.fastq.gz {1}\_R2.trimmed.fastq.gz | bamsormadup tmpfile=\$TMPDIR/bamsormadup\_\$(hostname)\_\$SLURM\_ARRAY\_JOB\_ID inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] indexfilename={1}.dedup.rg.csorted.bam.bai > {1}.dedup.rg.csorted.bam && mkdir -p {1}\_bamqc && mosdepth -t \$SLURM\_CPUS\_PER\_TASK -x -n {1}\_bamqc/{1} {1}.dedup.rg.csorted.bam; unset DISPLAY; qualimap bamqc -bam {1}.dedup.rg.csorted.bam --java-mem-size=32G -c -gff $REFERENCE.bed -outdir {1}\_bamqc && rm {1}.sra {1}\_[12].fastq.gz" > $RUN\_DIR/$JOBNAME.cmds  
  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l)%10 --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4)  
  
# find if any jobs failed  
FAILED\_TASKS=$(sacct -n -X -j $ARRAY\_ID -o state%20,jobid%20 | grep -v COMPLETED | gawk '{print $2}' | cut -d"\_" -f2 | paste -sd,)  
# Change job directives (if needed)  
# MEM=64  
WALLTIME="10:00:00"  
# retry failed jobs  
sbatch -a $FAILED\_TASKS --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm

### Calling variants (using Freebayes)

We used Freebayes *v1.3.5* (Garrison and Marth 2012) to assign variant probability scores and call variants. Notice that we used diploid mode (-p 2) and added the flag --genotype-qualities to be able to filter the resulting vcf file based on genotype qualities (GQ).

WORK\_DIR="/scratch/project/adna/A\_rabiei/Murdoch\_WGRS"  
RUN\_DIR="$WORK\_DIR/SNP\_calling\_04\_04\_2025"  
FQ\_DIR="$WORK\_DIR/fungaldata"  
GENOME="$WORK\_DIR/A\_rabiei\_TECAN\_2022/ref\_genome/ArME14\_v2\_CCDM"  
CONDA\_NAME=genomics  
  
BAM\_DIR=$RUN\_DIR/aligned\_reads  
  
# bring in BAM files (if not prepared in the previous step)  
# select isolates  
ISOLATES="Ar0020|Ar0023|Ar0212|AR0242|AR0052|AR0210|AR0022|AR0128"  
find /scratch/project/adna/A\_rabiei/AGRF\_gatk\_13\_03\_2025/aligned\_reads -maxdepth 1 -name "\*.rg.csorted.bam\*" | egrep -i $ISOLATES | parallel ln -s {} $BAM\_DIR/  
ln -s /scratch/project/adna/A\_rabiei/Murdoch\_WGRS/GATK\_Murdoch\_WGRS\_04\_04\_2025/aligned\_reads/\*.rg.csorted.bam\* $BAM\_DIR/  
  
# Distributed freebayes (each node runs freebayes-parallel on one contig)  
# download script  
aria2c -c -x5 -d ~/bin https://raw.githubusercontent.com/freebayes/freebayes/master/scripts/split\_ref\_by\_bai\_datasize.py   
chmod +x ~/bin/split\_ref\_by\_bai\_datasize.py  
mamba install -y -n $CONDA\_NAME numpy scipy  
  
# start\_interactive\_job  
conda activate $CONDA\_NAME  
# fix library dependencies  
find $CONDA\_PREFIX -name "libtabixpp.so\*" | parallel ln -s {} {.}.0  
# ln -s $CONDA\_PREFIX/lib/libtabixpp.so.1 $CONDA\_PREFIX/lib/libtabixpp.so.0  
# split each contig/chromosome to smaller 1e6 bits  
# prepare BAM files  
JOBNAME="prep\_bams"  
NCORES=2  
MEM=16  
WALLTIME="1:00:00"  
# submit it as a Slurm job  
echo "~/bin/split\_ref\_by\_bai\_datasize.py -s 1e6 -r $GENOME.fa.fai $(ls -1S $BAM\_DIR/\*.dedup.rg.csorted.bam | tail -n1) > $RUN\_DIR/ArME14\_target\_1e6\_regions\_chr.tsv" > $JOBNAME.cmds  
# submit the job   
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm  
  
JOBNAME="Murdoch\_FB\_diploid"  
NCORES=6  
MEM=32  
WALLTIME="1:00:00"  
# RUN\_DIR=$WORK\_DIR/SNP\_calling\_24\_01\_2025  
PLOIDY=2  
MIN\_DP=7  
# prepare commands  
BAM\_FILES=$( find $BAM\_DIR -maxdepth 1 -name "\*.rg.csorted.bam" | xargs )  
cut -f1 $GENOME.fa.fai | parallel --dry-run "freebayes-parallel <(grep '{}' $RUN\_DIR/ArME14\_target\_1e6\_regions\_chr.tsv | gawk '{printf \"%s:%s-%s\n\", \$1, \$2, \$3}') \$SLURM\_CPUS\_PER\_TASK -f $GENOME.fa --genotype-qualities -g 100000 -C $MIN\_DP -p $PLOIDY $BAM\_FILES > $RUN\_DIR/FB\_array\_output/{}.combined.vcf" > $RUN\_DIR/$JOBNAME.cmds  
mkdir -p $RUN\_DIR/FB\_array\_output  
# exit interactive job  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4)  
  
# merge variants  
ls -1 $RUN\_DIR/FB\_array\_output/ArME14\_ctg\_\*.combined.vcf | parallel --dry-run "printf \"{}\t%s\t%s\t%s\t%s\n\" \$(cat {} | grep -c -v '^#') \$(cat {} | SnpSift filter \"( GEN[?].DP > $MIN\_DP ) & ( GEN[?].GT != './.' )\" | gawk '{if (\$0 ~ /^#/ || (length(\$4)==1 && length(\$5)==1)); print \$0}' | grep -c -v '^#') \$(cat {} | SnpSift filter \"( GEN[?].DP > $MIN\_DP ) & ( GEN[?].GT != './.' ) & ( QUAL > 20 )\" | gawk '{if (\$0 ~ /^#/ || (length(\$4)==1 && length(\$5)==1)); print \$0}' | grep -c -v '^#') \$(cat {} | SnpSift filter \"( GEN[?].DP > $MIN\_DP ) & ( GEN[?].GT != './.' ) & ( QUAL > 30 )\" | gawk '{if (\$0 ~ /^#/ || (length(\$4)==1 && length(\$5)==1)); print \$0}' | grep -c -v '^#') > {}.stats" > ${RUN\_DIR}/freebayes-stats.cmds  
JOBNAME="freebayes-merge"  
echo "cat ${RUN\_DIR}/freebayes-stats.cmds | parallel && cat $RUN\_DIR/FB\_array\_output/ArME14\_ctg\_\*.combined.vcf | vcffirstheader | vcfstreamsort -w 1000 | vcfuniq > A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf && bgzip A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf && tabix A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf.gz" > $JOBNAME.cmds  
# submit job to cluster  
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm  
# combaine stats  
cat <(printf "file\ttotal\_snps\tDP${MIN\_DP}\_filtered\_snps\tQUAL20\_filtered\_snps\tQUAL30\_filtered\_snps\n") <(cat $RUN\_DIR/FB\_array\_output/ArME14\_ctg\_\*.vcf.stats) > A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid\_vcf\_stats\_$(date +%d\_%m\_%Y).txt

#### Create and filter final variant file

The variants that were called individually in each chromosome were merged into a single vcf file, which was gzipped, indexed. Variants were filtered using a combination of commands from SnpSift *v5.1d* (Ruden *et al.* 2012), BCFtools *v1.17* (Li 2011; Danecek *et al.* 2021) and VCFtools *v0.1.16* (Danecek *et al.* 2011), based on their total loci depth and quality, keeping only bi-allelic polymorphic SNP loci with a depth of at least 10 and not more than 100,000 reads covering the locus and a minimum Quality of 30 (10<DP<100000 & QUAL>30, based on EDA). In addition, each isolate’s genotype call was reset (recoded as missing, or ./.) if it had read depth (DP<10) or called as a heterozygote. Variant statistics were generated by BCFtools pre and post filter.

CONDA\_NAME="genomics"  
conda activate $CONDA\_NAME  
# Recode genotypes as missing if below a certain threshold, such as genotyping quality or depth (GQ:DP)   
# filter only polymorphic bi-allelic SNPs, using QUAL>20, 7<DP<100000  
  
# filter Freebayes variants with SnpSift and vcftools (wipe any heterozygote genotype with DP<7 with bcftools)  
QUAL=30 # 30  
# MQ=30  
MAX\_DP=100000  
MIN\_DP=10  
IND\_DP=10  
JOBNAME="Murdoch-wgrs-fb-filter"  
  
echo "bcftools filter -S . -e \"GT=='het' | FMT/DP<$IND\_DP\" $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf.gz -O v | SnpSift filter \"( QUAL>=$QUAL ) & (DP<$MAX\_DP) & ( countRef()>=1 & countVariant()>=1 )\" | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz   
vcftools --gzvcf $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz --recode --recode-INFO-all --minQ $QUAL --remove-indels -c | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.snps.Q$QUAL.noHet.poly.recode.vcf.gz   
vcftools --gzvcf $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz --recode --recode-INFO-all --minQ $QUAL --keep-only-indels -c | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.indels.Q$QUAL.noHet.poly.recode.vcf.gz "> $RUN\_DIR/$JOBNAME.cmds  
  
NCORES=6  
MEM=32  
WALLTIME="1:00:00"  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
  
#bcftools filter -S . -e "GT=='het' | FMT/DP<$MIN\_DP" A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf.gz -O v | SnpSift filter "( QUAL>=$QUAL ) & ( DP<=$MAX\_DP ) & ( DP>=$MIN\_DP ) & ( countRef()>=1 & countVariant()>=1 )" | vcftools --vcf - --recode --recode-INFO-all --minQ $QUAL --max-missing 0.75 --remove-indels -c | bgzip -o A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.GT75.noRep.noHet.poly.recode.vcf.gz && tabix A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.GT75.noRep.noHet.poly.recode.vcf.gz  
# generate stats   
JOBNAME="bcftools\_stats"  
WALLTIME=2:00:00  
MEM=32  
NCORES=8  
find . -name "\*.vcf.gz" | parallel --dry-run "bcftools stats -s - {} > {.}.bcfstats.txt" > $JOBNAME.cmds  
# send to the cluster  
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/parallel\_jobs\_run.slurm

An in-house R script (estimate\_error\_rates\_vcf\_files.R) was used to estimate the error rates based on the presence of duplicated samples.

#### Call SNPs with GATK

The SNPs were also called from the WGRS using gatk *v4.3.0.0* (Van der Auwera *et al.* 2002; Heldenbrand *et al.* 2019) following [this tutorial](https://www.melbournebioinformatics.org.au/tutorials/tutorials/variant_calling_gatk1/variant_calling_gatk1/) from Melbourne Bioinformatics. GATK can use a reference variant file as a guide for Base Quality Score Recalibration (BQSR, highly recommended) and will only call variants in those locations, so we need to generate that “Gold Standard” reference. It can be done by either of these options:  
1. Use a set of variants called by CCDM from their assemblies (if it exists)  
2. Use the variants generated by Freebayes (or Snippy) for the 2020 WGS batch sequenced at AGRF (which might introduce a bias towards those pre-selected sites)  
3. Use GATK to call variants from the 2020 WGS batch sequenced at AGRF without BQSR, then use Variant Filtering (VQSR) to select a set of high-confidence variants and use them as the reference for subsequent recalibration and variant calling (in a bootstrap approach, as described in the [GATK BQSR documentation](https://gatk.broadinstitute.org/hc/en-us/articles/360035890531-Base-Quality-Score-Recalibration-BQSR))  
We chose the 3rd option, as follows:

CONDA\_NAME="genomics"  
# install tools  
mamba install -y -n $CONDA\_NAME bwa-mem2 bowtie2 biobambam sambamba qualimap multiqc fastp gatk4  
WORK\_DIR="/home/ibar/adna/A\_rabiei"  
REF\_DIR="/scratch/project/adna/A\_rabiei/A\_rabiei\_TECAN\_2022/ref\_genome"  
GENOME="$REF\_DIR/ArME14\_v2\_CCDM"  
  
# prepare working folder and reference genome  
DATE=`date +%d\_%m\_%Y`  
BATCH=AGRF  
RUN="${BATCH}\_gatk\_${DATE}" # day of run was 02\_02\_2019  
RUN\_DIR=$WORK\_DIR/${RUN}  
  
mkdir -p $RUN\_DIR/aligned\_reads $RUN\_DIR/$JOBNAME && cd $RUN\_DIR  
  
NCORES=2  
MEM=8  
WALLTIME="2:00:00"  
JOBNAME="gatk-prep-genome"  
# prepare genome  
echo "picard CreateSequenceDictionary R=$GENOME.fa O=$GENOME.dict; gatk IndexFeatureFile -I $REF\_VCF" > $RUN\_DIR/$JOBNAME.cmds  
# send to the cluster  
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm   
  
# prepare commands to run mapping and initial gatk  
# define variables   
NCORES=12  
MEM=64  
WALLTIME=2:00:00  
RGPM="NovaSeq"  
RGPL="ILLUMINA"  
RGPU="H3HGFDSX2"  
RGCN="AGRF"  
FQ\_DIR="/scratch/project/adna/A\_rabiei/AGRF\_snippy\_05\_03\_2025"  
JOBNAME="agrf-initial-gatk"  
  
# Create the bwa-mem2 commands to align all read pairs and create GVCF files  
find $FQ\_DIR -name "\*\_R1.trimmed.fastq.gz" | parallel -k --dry-run --rpl "{file2} s:\_R1:\_R2:" --rpl "{sample} s:.+\/(.+?)\_R1.+:\1:" "ALIGN\_DIR=$RUN\_DIR/aligned\_reads && bwa-mem2 mem -R \"@RG\tID:{sample}\tSM:{sample}\tLB:{sample}\tPU:$RGPU\tPL:$RGPL\tPM:$RGPM\tCN:$RGCN\" -t \$[SLURM\_CPUS\_PER\_TASK - 2] $GENOME.fa {} {file2} | bamsormadup tmpfile=\$TMPDIR/bamsormadup\_\$(hostname)\_\$SLURM\_ARRAY\_JOB\_ID inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] indexfilename=\$ALIGN\_DIR/{sample}.dedup.rg.csorted.bam.bai > \$ALIGN\_DIR/{sample}.dedup.rg.csorted.bam; gatk --java-options \"-Xmx7g\" HaplotypeCaller -I \$ALIGN\_DIR/{sample}.dedup.rg.csorted.bam -R $GENOME.fa -ERC GVCF -O $RUN\_DIR/$JOBNAME/{sample}.g.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4)  
  
# find which jobs failed  
FAILED\_TASKS=$(sacct -n -X -j $ARRAY\_ID -o state%20,jobid%20 | grep -v COMPLETED | gawk '{print $2}' | cut -d"\_" -f2 | paste -sd,)  
MEM=64  
sbatch -a $FAILED\_TASKS --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm   
  
  
# step 3 - Combine GCVF files and call variants  
GCVF\_FILES=$(find $RUN\_DIR/agrf-initial-gatk -maxdepth 1 -name "\*.g.vcf.gz" | gawk '{printf " -V %s", $1}')  
JOBNAME="agrf-initial-gatk-combine-gvcf"  
mkdir -p $RUN\_DIR/output  
echo "gatk --java-options \"-Xmx7g\" CombineGVCFs -R $GENOME.fa $GCVF\_FILES -O $RUN\_DIR/output/AGRF2020\_cohort.g.vcf.gz  
gatk IndexFeatureFile -I $RUN\_DIR/output/AGRF2020\_cohort.g.vcf.gz  
gatk --java-options \"-Xmx7g\" GenotypeGVCFs -R $GENOME.fa -V $RUN\_DIR/output/AGRF2020\_cohort.g.vcf.gz -O $RUN\_DIR/output/AGRF2020\_cohort.gatk.vcf.gz"> $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm  
  
# step 4a - Filter variants  
QUAL=30 # 30  
MQ=40  
MAX\_DP=100000  
MIN\_DP=10  
IND\_DP=10  
  
JOBNAME="agrf-initial-gatk-filter-snps"  
echo "gatk SelectVariants -V $RUN\_DIR/output/AGRF2020\_cohort.gatk.vcf.gz -select-type SNP -O $RUN\_DIR/output/AGRF2020\_cohort.gatk.snps.vcf.gz   
gatk VariantFiltration -V $RUN\_DIR/output/AGRF2020\_cohort.gatk.snps.vcf.gz \  
 -filter \"QD < 2.0\" --filter-name \"QD2\" \  
 -filter \"QUAL < $QUAL.0\" --filter-name \"QUAL$QUAL\" \  
 -filter \"SOR > 3.0\" --filter-name \"SOR3\" \  
 -filter \"FS > 60.0\" --filter-name \"FS60\" \  
 -filter \"MQ < $MQ.0\" --filter-name \"MQ$MQ\" \  
 -filter \"MQRankSum < -12.5\" --filter-name \"MQRankSum-12.5\" \  
 -filter \"ReadPosRankSum < -8.0\" --filter-name \"ReadPosRankSum-8\" \  
 --genotype-filter-expression \"isHet == 1\" \  
 --genotype-filter-name \"isHetFilter\" \  
 --genotype-filter-expression \"DP < $IND\_DP.0\" \  
 --genotype-filter-name \"DP$IND\_DP\" \  
 --genotype-filter-expression \"GQ < $QUAL.0\" \  
 --genotype-filter-name \"DP$QUAL\" \  
 -O $RUN\_DIR/output/AGRF2020\_cohort.gatk.snps.filtered.vcf.gz  
gatk SelectVariants \  
 -V $RUN\_DIR/output/AGRF2020\_cohort.gatk.snps.filtered.vcf.gz \  
 --set-filtered-gt-to-nocall \  
 --max-fraction-filtered-genotypes 0.5 \  
 -O $RUN\_DIR/output/AGRF2020\_cohort.gatk.snps.gt\_filtered.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm  
  
# step 4b - Filter variants  
JOBNAME="agrf-initial-gatk-filter-indels"  
echo "gatk SelectVariants -V $RUN\_DIR/output/AGRF2020\_cohort.gatk.vcf.gz -select-type INDEL -O $RUN\_DIR/output/AGRF2020\_cohort.gatk.indels.vcf.gz   
gatk VariantFiltration \  
 -V $RUN\_DIR/output/AGRF2020\_cohort.gatk.indels.vcf.gz \  
 -filter \"QD < 2.0\" --filter-name \"QD2\" \  
 -filter \"QUAL < 30.0\" --filter-name \"QUAL30\" \  
 -filter \"FS > 200.0\" --filter-name \"FS200\" \  
 -filter \"ReadPosRankSum < -20.0\" --filter-name \"ReadPosRankSum-20\" \  
 --genotype-filter-expression \"isHet == 1\" \  
 --genotype-filter-name \"isHetFilter\" \  
 --genotype-filter-expression \"DP < 10.0\" \  
 --genotype-filter-name \"DP10\" \  
 --genotype-filter-expression \"GQ < 30.0\" \  
 --genotype-filter-name \"DP30\" \  
 -O $RUN\_DIR/output/AGRF2020\_cohort.gatk.indels.filtered.vcf.gz  
gatk SelectVariants \  
 -V $RUN\_DIR/output/AGRF2020\_cohort.gatk.indels.filtered.vcf.gz \  
 --set-filtered-gt-to-nocall \  
 --max-fraction-filtered-genotypes 0.5 \  
 -O $RUN\_DIR/output/AGRF2020\_cohort.gatk.indels.gt\_filtered.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm  
  
# Step 5 - Combine VCFs  
  
JOBNAME="agrf-initial-gatk-combine-vcfs"  
echo "picard MergeVcfs I=$RUN\_DIR/output/AGRF2020\_cohort.gatk.snps.gt\_filtered.vcf.gz I=$RUN\_DIR/output/AGRF2020\_cohort.gatk.indels.gt\_filtered.vcf.gz O=$RUN\_DIR/output/AGRF2020\_cohort.gatk.gt\_filtered.combined.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm

Now we can run the full GATK pipeline.

WORK\_DIR="/home/ibar/adna/A\_rabiei"  
RUN\_DIR="$WORK\_DIR/AGRF\_gatk\_13\_03\_2025"  
CONDA\_NAME="genomics"  
REF\_DIR="/scratch/project/adna/A\_rabiei/A\_rabiei\_TECAN\_2022/ref\_genome"  
GENOME="$REF\_DIR/ArME14\_v2\_CCDM"  
REF\_VCF="$RUN\_DIR/output/AGRF2020\_cohort.gatk.gt\_filtered.combined.vcf.gz"  
BAM\_DIR="$RUN\_DIR/aligned\_reads"  
JOBNAME="agrf-gatk-bqsr"  
  
mkdir -p $RUN\_DIR/bqsr && cd $RUN\_DIR  
  
# step 1 - Build BQSR model and apply BQSR  
find $BAM\_DIR -maxdepth 1 -name "\*.rg.csorted.bam" -size +1M | parallel --dry-run "gatk --java-options \"-Xmx7g\" BaseRecalibrator -I {} -R $GENOME.fa --known-sites $REF\_VCF -O $RUN\_DIR/bqsr/{/.}.recal\_data.table; gatk --java-options \"-Xmx7g\" ApplyBQSR -I {} -R $GENOME.fa --bqsr-recal-file $RUN\_DIR/bqsr/{/.}.recal\_data.table -O bqsr/{/.}.bqsr.bam" > $RUN\_DIR/$JOBNAME.cmds  
  
  
# submit to the cluster  
sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm  
  
  
NCORES=2  
MEM=16  
WALLTIME="2:00:00"  
JOBNAME="agrf-gatk-gvcf"  
mkdir -p $RUN\_DIR/output2  
# step 2 - Create GCVF files per sample  
find $RUN\_DIR/bqsr -maxdepth 1 -name "\*.rg.csorted.bqsr.bam" -size +1M | parallel --dry-run --rpl "{sample} s:.+\/(.+?).dedup.rg.csorted.bqsr.bam:\1:" "gatk --java-options \"-Xmx7g\" HaplotypeCaller -I {} -R $GENOME.fa -ERC GVCF -O output/{sample}.g.vcf.gz"> $RUN\_DIR/$JOBNAME.cmds  
  
  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")  
  
JOBNAME="gatk-combine-gvcf"  
# step 3 - Combine GCVF files and call variants  
GCVF\_FILES=$(find $RUN\_DIR/output -maxdepth 1 -name "AR\*.g.vcf.gz" | gawk '{printf " -V %s", $1}')  
echo "gatk --java-options \"-Xmx7g\" CombineGVCFs -R $GENOME.fa $GCVF\_FILES -O $RUN\_DIR/output2/AGRF2020\_cohort.g.vcf.gz  
gatk IndexFeatureFile -I $RUN\_DIR/output2/AGRF2020\_cohort.g.vcf.gz  
gatk --java-options \"-Xmx7g\" GenotypeGVCFs -R $GENOME.fa -V $RUN\_DIR/output2/AGRF2020\_cohort.g.vcf.gz -O $RUN\_DIR/output2/AGRF2020\_cohort.gatk.vcf.gz"> $RUN\_DIR/$JOBNAME.cmds  
  
# submit to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f4 -d " ")  
  
# step 4a - Filter variants (maybe ExcessHet > 10)  
QUAL=30 # 30  
MQ=40  
GQ=30  
MAX\_DP=100000  
MIN\_DP=10  
IND\_DP=10  
JOBNAME="agrf-gatk-filter-snps"  
echo "gatk SelectVariants -V $RUN\_DIR/output2/AGRF2020\_cohort.gatk.vcf.gz -select-type SNP -O $RUN\_DIR/output2/AGRF2020\_cohort.gatk.snps.vcf.gz   
gatk VariantFiltration -V $RUN\_DIR/output2/AGRF2020\_cohort.gatk.snps.vcf.gz \  
 -filter \"QD < 2.0\" --filter-name \"QD2\" \  
 -filter \"QUAL < $QUAL.0\" --filter-name \"QUAL$QUAL\" \  
 -filter \"SOR > 3.0\" --filter-name \"SOR3\" \  
 -filter \"FS > 60.0\" --filter-name \"FS60\" \  
 -filter \"MQ < $MQ.0\" --filter-name \"MQ$MQ\" \  
 -filter \"MQRankSum < -12.5\" --filter-name \"MQRankSum-12.5\" \  
 -filter \"ReadPosRankSum < -8.0\" --filter-name \"ReadPosRankSum-8\" \  
 --genotype-filter-expression \"isHet == 1\" \  
 --genotype-filter-name \"isHetFilter\" \  
 --genotype-filter-expression \"DP < $IND\_DP.0\" \  
 --genotype-filter-name \"DP$IND\_DP\" \  
 --genotype-filter-expression \"GQ < $GQ.0\" \  
 --genotype-filter-name \"GQ$GQ\" \  
 -O $RUN\_DIR/output2/AGRF2020\_cohort.gatk.snps.filtered.vcf.gz  
gatk SelectVariants \  
 -V $RUN\_DIR/output2/AGRF2020\_cohort.gatk.snps.filtered.vcf.gz \  
 --set-filtered-gt-to-nocall \  
 --max-fraction-filtered-genotypes 0.5 \  
 --exclude-filtered true \  
 -O $RUN\_DIR/output2/AGRF2020\_cohort.gatk.snps.gt\_filtered.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm| cut -f4 -d " ")  
  
# step 4b - Filter variants  
JOBNAME="agrf-gatk-filter-indels"  
echo "gatk SelectVariants -V $RUN\_DIR/output2/AGRF2020\_cohort.gatk.vcf.gz -select-type INDEL -O $RUN\_DIR/output2/AGRF2020\_cohort.gatk.indels.vcf.gz   
gatk VariantFiltration \  
 -V $RUN\_DIR/output2/AGRF2020\_cohort.gatk.indels.vcf.gz \  
 -filter \"QD < 2.0\" --filter-name \"QD2\" \  
 -filter \"QUAL < $QUAL.0\" --filter-name \"QUAL$QUAL\" \  
 -filter \"FS > 200.0\" --filter-name \"FS200\" \  
 -filter \"ReadPosRankSum < -20.0\" --filter-name \"ReadPosRankSum-20\" \  
 --genotype-filter-expression \"isHet == 1\" \  
 --genotype-filter-name \"isHetFilter\" \  
 --genotype-filter-expression \"DP < $IND\_DP.0\" \  
 --genotype-filter-name \"DP$IND\_DP\" \  
 --genotype-filter-expression \"GQ < $GQ.0\" \  
 --genotype-filter-name \"GQ$GQ\" \  
 -O $RUN\_DIR/output2/AGRF2020\_cohort.gatk.indels.filtered.vcf.gz  
gatk SelectVariants \  
 -V $RUN\_DIR/output2/AGRF2020\_cohort.gatk.indels.filtered.vcf.gz \  
 --set-filtered-gt-to-nocall \  
 --max-fraction-filtered-genotypes 0.5 \  
 -O $RUN\_DIR/output2/AGRF2020\_cohort.gatk.indels.gt\_filtered.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f4 -d " ")  
  
# Step 5 - Combine VCFs  
  
JOBNAME="agrf-gatk-combine-vcfs"  
echo "picard MergeVcfs I=$RUN\_DIR/output2/AGRF2020\_cohort.gatk.snps.gt\_filtered.vcf.gz I=$RUN\_DIR/output2/AGRF2020\_cohort.gatk.indels.gt\_filtered.vcf.gz O=$RUN\_DIR/output2/AGRF2020\_cohort.gatk.gt\_filtered.combined.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f4 -d " ")  
  
# clean output folders  
rm -rf $RUN\_DIR/output $RUN\_DIR/aligned\_reads $RUN\_DIR/bqsr

CONDA\_NAME="genomics"  
# install tools  
mamba install -y -n $CONDA\_NAME bwa-mem2 bowtie2 biobambam sambamba qualimap multiqc fastp gatk4  
  
# create working directory  
WORK\_DIR="/scratch/project/adna/A\_rabiei/Murdoch\_WGRS"  
RUN\_DIR="$WORK\_DIR/GATK\_Murdoch\_WGRS\_04\_04\_2025"  
mkdir -p $RUN\_DIR/bqsr $RUN\_DIR/gvcf $RUN\_DIR/output && cd $RUN\_DIR  
# Set variables  
GENOME="$REF\_DIR/ArME14\_v2\_CCDM"  
AGRF\_GATK\_DIR="/scratch/project/adna/A\_rabiei/AGRF\_gatk\_13\_03\_2025"  
REF\_VCF="$AGRF\_GATK\_DIR/output/AGRF2020\_cohort.gatk.gt\_filtered.combined.vcf.gz"  
BAM\_DIR="$AGRF\_GATK\_DIR/aligned\_reads"  
JOBNAME="agrf-gatk-bqsr-gvcf"  
  
# prepare AGRF batch (original isolates)  
# select isolates  
ISOLATES="Ar0020|Ar0023|Ar0212|AR0242|AR0052|AR0210|AR0022|AR0128"  
find $BAM\_DIR -maxdepth 1 -name "\*.rg.csorted.bam" -size +1M | egrep -i $ISOLATES | parallel --dry-run --rpl "{sample} s:.+\/(.+?).dedup.rg.csorted.bam:\1:" "gatk --java-options \"-Xmx7g\" BaseRecalibrator -I {} -R $GENOME.fa --known-sites $REF\_VCF -O $RUN\_DIR/bqsr/{/.}.recal\_data.table; gatk --java-options \"-Xmx7g\" ApplyBQSR -I {} -R $GENOME.fa --bqsr-recal-file $RUN\_DIR/bqsr/{/.}.recal\_data.table -O bqsr/{/.}.bqsr.bam; gatk --java-options \"-Xmx7g\" HaplotypeCaller -I $RUN\_DIR/bqsr/{/.}.bqsr.bam -R $GENOME.fa -ERC GVCF -O $RUN\_DIR/gvcf/{sample}.g.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
  
NCORES=2  
MEM=8  
WALLTIME="3:00:00"  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")  
  
  
# assign variables  
RGPM="DNBseq\_T7"  
RGPL="MGI"  
RGPU="E250038400"  
RGCN="MU\_SABC"  
FQ\_DIR="$WORK\_DIR/fungaldata"  
ln -s $WORK\_DIR/fungal/\*.fq.gz $FQ\_DIR/  
BAM\_DIR="$RUN\_DIR/aligned\_reads"  
mkdir -p $BAM\_DIR $FQ\_DIR/trimmed\_reads/QC  
JOBNAME="Murdoch-fastp-bwa"  
  
  
find $FQ\_DIR/\*.R1.fq.gz | parallel -k --dry-run --rpl "{file2} s:.R1:.R2:; uq()" --rpl "{sample} s:.+\/(.+?).R1.fq.gz:\1:" "fastp -i {} -I {file2} --detect\_adapter\_for\_pe -c -l 30 -p -w \$SLURM\_CPUS\_PER\_TASK -z 7 -o $FQ\_DIR/trimmed\_reads/{sample}\_R1.trimmed.fastq.gz -O $FQ\_DIR/trimmed\_reads/{sample}\_R2.trimmed.fastq.gz -j $FQ\_DIR/trimmed\_reads/QC/{sample}.fastp.json && bwa-mem2 mem -R \"@RG\tID:{sample}\tSM:{sample}\tLB:{sample}\tPU:$RGPU\tPL:$RGPL\tPM:$RGPM\tCN:$RGCN\" -t \$[SLURM\_CPUS\_PER\_TASK - 2] $GENOME.fa $FQ\_DIR/trimmed\_reads/{sample}\_R1.trimmed.fastq.gz $FQ\_DIR/trimmed\_reads/{sample}\_R2.trimmed.fastq.gz | bamsormadup tmpfile=\$TMPDIR/bamsormadup\_\$(hostname)\_\$SLURM\_ARRAY\_JOB\_ID inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] indexfilename=$BAM\_DIR/{sample}.dedup.rg.csorted.bam.bai > $BAM\_DIR/{sample}.dedup.rg.csorted.bam" > $RUN\_DIR/$JOBNAME.cmds  
  
NCORES=12  
MEM=96  
WALLTIME="5:00:00"  
  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")  
  
# step 0 - assess coverage of files  
JOBNAME="Murdoch-gatk-coverage"  
mkdir -p $RUN\_DIR/coverage  
  
find $BAM\_DIR -maxdepth 1 -name "\*.rg.csorted.bam" -size +1M | parallel --dry-run --rpl "{sample} s:.+\/(.+?).dedup.rg.csorted.bam:\1:" "samtools idxstats {} | gawk -vreadlen=150 '{len += \$2; nreads += \$3} END {cov=nreads \* readlen / len; printf \"x%s\n\", cov}' > $RUN\_DIR/coverage/{sample}.coverage" > $RUN\_DIR/$JOBNAME.cmds  
  
NCORES=12  
MEM=64  
WALLTIME="3:00:00"  
  
# submit the job array to the cluster  
sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/parallel\_jobs\_run.slurm  
# summarise coverage  
find $RUN\_DIR/coverage -maxdepth 1 -name "\*.coverage" | parallel "printf \"{}\t %s\n\" \$(cat {})" > coverage\_summary.txt  
  
  
# step 1 - Build BQSR model and apply BQSR  
JOBNAME="Murdoch-gatk-bqsr-gvcf"  
  
find $BAM\_DIR -maxdepth 1 -name "\*.rg.csorted.bam" -size +1M | parallel --dry-run --rpl "{sample} s:.+\/(.+?).dedup.rg.csorted.bam:\1:" "gatk --java-options \"-Xmx7g\" BaseRecalibrator -I {} -R $GENOME.fa --known-sites $REF\_VCF -O $RUN\_DIR/bqsr/{/.}.recal\_data.table; gatk --java-options \"-Xmx7g\" ApplyBQSR -I {} -R $GENOME.fa --bqsr-recal-file $RUN\_DIR/bqsr/{/.}.recal\_data.table -O $RUN\_DIR/bqsr/{/.}.bqsr.bam; gatk --java-options \"-Xmx7g\" HaplotypeCaller -I $RUN\_DIR/bqsr/{/.}.bqsr.bam -R $GENOME.fa -ERC GVCF -O $RUN\_DIR/gvcf/{sample}.g.vcf.gz" > $RUN\_DIR/$JOBNAME.cmds  
  
NCORES=4  
MEM=12  
WALLTIME="3:00:00"  
  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")  
  
# step 3 - Combine GCVF files, call and filter variants  
QUAL=30  
MQ=30  
DP=10  
MAX\_DP=100000  
JOBNAME="Murdoch-gatk-combine-vcf"  
GCVF\_FILES=$(find $RUN\_DIR/gvcf -maxdepth 1 -name "\*.g.vcf.gz" | gawk '{printf " -V %s", $1}')  
echo "gatk --java-options \"-Xmx7g\" CombineGVCFs -R $GENOME.fa $GCVF\_FILES -O $RUN\_DIR/gvcf/Murdoch\_WGRS\_cohort.g.vcf.gz  
gatk IndexFeatureFile -I $RUN\_DIR/gvcf/Murdoch\_WGRS\_cohort.g.vcf.gz  
gatk --java-options \"-Xmx7g\" GenotypeGVCFs -R $GENOME.fa -V $RUN\_DIR/gvcf/Murdoch\_WGRS\_cohort.g.vcf.gz -O $RUN\_DIR/output/Murdoch\_WGRS\_cohort.gatk.vcf.gz"> $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
  
# step 4 - Filter variants  
JOBNAME="Murdoch-gatk-vcf-filter"  
echo "bcftools filter -S . -e \"GT=='het' | FMT/DP<$DP\" $RUN\_DIR/output/Murdoch\_WGRS\_cohort.gatk.vcf.gz -O v | SnpSift filter \"( QUAL>=$QUAL ) & ( MQ>=$MQ ) & ( DP<=$MAX\_DP ) & ( countRef()>=1 & countVariant()>=1 )\" | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/output/Murdoch\_WGRS\_cohort.gatk.gt\_filtered.vcf.gz   
vcftools --gzvcf $RUN\_DIR/output/Murdoch\_WGRS\_cohort.gatk.gt\_filtered.vcf.gz --recode --recode-INFO-all --minQ $QUAL --remove-indels -c | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/output/Murdoch\_WGRS\_cohort.gatk.snps.gt\_filtered.vcf.gz   
vcftools --gzvcf $RUN\_DIR/output/Murdoch\_WGRS\_cohort.gatk.gt\_filtered.vcf.gz --recode --recode-INFO-all --minQ $QUAL --keep-only-indels -c | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/output/Murdoch\_WGRS\_cohort.gatk.indels.gt\_filtered.vcf.gz "> $RUN\_DIR/$JOBNAME.cmds  
# submit to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
  
# Step 5 - generate stats   
JOBNAME="bcftools\_stats"  
WALLTIME=2:00:00  
MEM=8  
NCORES=4  
find . -name "\*.vcf.gz" | parallel --dry-run "bcftools stats -s - {} > {.}.bcfstats.txt" > $JOBNAME.cmds  
# send to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/parallel\_jobs\_run.slurm | cut -f 4 -d " ")  
  
# clean output folders  
rm -r $RUN\_DIR/output/Ar\*.g.vcf.gz\* $RUN\_DIR/bqsr $RUN\_DIR/aligned\_reads

#### MutilQC

Quality metrics were collected from the raw read QC, alignment and variant calling steps and were consolidated into a single, interactive report for each batch using MultiQC v1.21 (Ewels *et al.* 2016).

NCORES=8  
MEM=32  
WALLTIME="10:00:00"  
JOBNAME="Multiqc\_Murdoch\_WGRS"  
# multiqc report  
MULTIQC\_JOB=QC\_$(date +%d\_%m\_%Y)  
# submit it as a Slurm job  
echo "multiqc --interactive --force -i $MULTIQC\_JOB -o $MULTIQC\_JOB ." > $JOBNAME.cmds  
# submit the job   
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " " )  
# Done!  
  
# Copy html files to SharePoint  
rclone copy -P --exclude "\*\*.html" $RUN\_DIR GRDC\_rabiei:General/Projects/Hayley\_PhD/Genetics/Murdoch\_WGRS/SNP\_calling\_24\_01\_2025  
# Copy html files to SharePoint  
rclone copy -P --ignore-checksum --ignore-size --include "\*\*.html" $RUN\_DIR GRDC\_rabiei:General/Projects/Hayley\_PhD/Genetics/Murdoch\_WGRS/SNP\_calling\_24\_01\_2025

### Assessing the Sequencing Depth Effect on Clustering

* What depth of sequencing is required to achieve good (accurate) clustering of the samples? (how do we measure this)

To do this we can use the same set of samples (and controls) used to determine the optimal pipeline, but resample the reads (x5, x10, x20, x30 coverage), which is equivalent to 700,000, 1,400,000, 2,800,000, 4,200,000 PE 150bp reads. It would be easiest to do it on the aligned BAM files (this can be done with BBtools’ reformat.sh or seqtk, see this [BioStar thread](https://www.biostars.org/p/9563190/#9563191))

### Comparison of Alignment and SNP calling pipelines

We will test different alignment tools and parameters (BWA-MEM2, Bowtie2) with different variant callers (GATK, FreeBayes, Clair3) and a complete Nextflow pipeline implementing those (Sarek) to assess and identify the optimal SNP calling pipeline that will strike a good balance between sensitive SNP discovery and the lowest error rates (as measured by heterozygote calls and inconsistencies between technical replicates [AR0039, AR0052, AR0242] and low-coverage sampled [Ar23640, Ar23497, Ar23483]). To compare the VCF files we will use [RTG tools](https://github.com/RealTimeGenomics/rtg-tools).(Cleary *et al.* 2015). An example for this approach was described for another haploid fungi, *Candida auris*, as described by Li *et al.* (2023).

#### Compare alignment tools

We will test the following alignment options:  
1. BWA-MEM2 with -B ranging from 4:6 and -O 6/7/8 (which includes default -B 4 -O 6).  
2. Bowtie2 in local alignment mode: --local-fast, --local-sensitive (default), --local--very-sensitive.  
3. Same as above, but in global alignment mode: --global-sensitive, etc.  
Doing each step in parallel takes up too much space on the cluster. A better approach would be to align the reads and call variants with the different callers for each combination and then remove the bam files.

WORK\_DIR="/scratch/ibar/A\_rabiei"  
WORK\_DIR="/scratch/project/adna/A\_rabiei"  
  
# GENOME="$WORK\_DIR/A\_rabiei\_TECAN\_2022/ref\_genome/ArME14\_v2\_CCDM"  
CONDA\_NAME="genomics"  
# install tools  
mamba install -y -n $CONDA\_NAME bwa-mem2 bowtie2 biobambam sambamba qualimap multiqc fastp  
# assign variables  
REF\_DIR="/scratch/project/adna/A\_rabiei/A\_rabiei\_TECAN\_2022/ref\_genome"  
GENOME="$REF\_DIR/ArME14\_v2\_CCDM"  
# build index for bowtie2  
# conda activate $CONDA\_NAME  
# or run with a container (need to login to a compute node)  
start\_debug\_interactive\_job  
apptainer exec $NXF\_APPTAINER\_CACHEDIR/depot.galaxyproject.org-singularity-bowtie2-2.4.4--py39hbb4e92a\_0.img bowtie2-build $GENOME.fa $GENOME  
  
MURDOCH\_DIR="/scratch/project/adna/A\_rabiei/Murdoch\_WGRS/fungal"  
AGRF\_DIR="/scratch/project/adna/A\_rabiei/AGRF\_CAGRF20114478"  
# create a working folder  
RUN\_DIR="$WORK\_DIR/SNP\_calling\_comparison"  
FQ\_DIR="$RUN\_DIR/data"  
mkdir -p $RUN\_DIR/aligned\_reads $RUN\_DIR/trimmed\_reads/QC $FQ\_DIR && cd $RUN\_DIR  
  
# select isolates  
ISOLATES="Ar23401\_|Ar0020|Ar0023|Ar0212|AR0242|AR0052|AR0210|AR0022|AR0128|Ar23640|Ar23653\_|Ar23497|Ar23483"  
  
# Create symlinks to Murdoch reads  
ls -1 $MURDOCH\_DIR/\*.fq.gz | egrep -i $ISOLATES | parallel ln -s {} $FQ\_DIR/  
# Create symlinks to AGRF reads  
ls -1 $AGRF\_DIR/\*.fastq.gz | egrep -i $ISOLATES | parallel ln -s {} $FQ\_DIR/  
  
# standardise names  
# AR0070\_H3HGFDSX2\_CTGACTCTAC-TGACGGCCGT\_L002\_R1.fastq.gz  
cd $FQ\_DIR  
rename -v 's|.+\/(.+?)\_H3HGFDSX2\_[A-Z\-]+\_L002\_(R[0-9]).fastq.gz|\1.\2.fq.gz|' \*.fastq.gz  
  
cd $RUN\_DIR  
  
NCORES=12  
MEM=64  
WALLTIME="2:00:00"  
JOBNAME="process\_reads"  
# create commands  
find $FQ\_DIR/\*.R1.fq.gz | parallel -k --dry-run --rpl "{file2} s:.R1:.R2:; uq()" --rpl "{sample} s:.+\/(.+?).R1.fq.gz:\1:" "fastp -i {} -I {file2} --detect\_adapter\_for\_pe -c -l 30 -p -w \$SLURM\_CPUS\_PER\_TASK -z 7 -o $RUN\_DIR/trimmed\_reads/{sample}\_R1.trimmed.fastq.gz -O $RUN\_DIR/trimmed\_reads/{sample}\_R2.trimmed.fastq.gz -j $RUN\_DIR/trimmed\_reads/QC/{sample}.fastp.json" > $JOBNAME.cmds  
  
# submit to the cluster  
sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm  
  
  
  
NCORES=12  
MEM=64  
WALLTIME="1:00:00"  
JOBNAME="bwa-mem-align-AGRF"  
# Create the bwa-mem2 commands to align the reads to the genome.  
parallel -k --dry-run --rpl "{file2} s:\_R1:\_R2:" --rpl "{sample} s:.+\/(.+?)\_R1.+.gz:\1:" "ALIGN\_DIR=$RUN\_DIR/aligned\_reads/bwamem\_B{2}\_O{3}; mkdir -p \$ALIGN\_DIR && bwa-mem2 mem -B {2} -O {3} -R \"@RG\tID:{1 sample}\tSM:{1 sample}\tLB:{1 sample}\tPU:H3HGFDSX2\tPL:ILLUMINA\tCN:AGRF\" -t \$[SLURM\_CPUS\_PER\_TASK - 2] $GENOME.fa {1} {1 file2} | bamsormadup tmpfile=\${TMPDIR:-/scratch/project/adna/tmp/}/\${SLURM\_JOB\_NAME} inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] > \$ALIGN\_DIR/{1 sample}.dedup.rg.csorted.bam; unset DISPLAY; qualimap bamqc -bam \$ALIGN\_DIR/{1 sample}.dedup.rg.csorted.bam --java-mem-size=32G -c -gff $GENOME.bed -outdir \$ALIGN\_DIR/{1 sample}\_bamqc" ::: $(find $RUN\_DIR/trimmed\_reads -name "AR\*\_R1.trimmed.fastq.gz") ::: 4 5 6 7 8 ::: 6 7 8 > $RUN\_DIR/$JOBNAME.cmds  
# submit it as a Slurm job array  
sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm  
  
JOBNAME="bwa-mem-align-Murdoch"  
# Create the bwa-mem2 commands to align the reads to the genome.  
parallel -k --dry-run --rpl "{file2} s:\_R1:\_R2:" --rpl "{sample} s:.+\/(.+?)\_R1.+.gz:\1:" "ALIGN\_DIR=$RUN\_DIR/aligned\_reads/bwamem\_B{2}\_O{3}; mkdir -p \$ALIGN\_DIR && bwa-mem2 mem -B {2} -O {3} -R \"@RG\tID:{1 sample}\tSM:{1 sample}\tLB:{1 sample}\tPU:E250038400\tPL:MGI-DNBSEQ-T7\tCN:Murdoch\" -t \$[SLURM\_CPUS\_PER\_TASK - 2] $GENOME.fa {1} {1 file2} | bamsormadup tmpfile=\${TMPDIR:-/scratch/project/adna/tmp/}/\${SLURM\_JOB\_NAME} inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] > \$ALIGN\_DIR/{1 sample}.dedup.rg.csorted.bam; unset DISPLAY; qualimap bamqc -bam \$ALIGN\_DIR/{1 sample}.dedup.rg.csorted.bam --java-mem-size=32G -c -gff $GENOME.bed -outdir \$ALIGN\_DIR/{1 sample}\_bamqc" ::: $(find $RUN\_DIR/trimmed\_reads -name "Ar\*\_R1.trimmed.fastq.gz") ::: 4 5 6 7 8 ::: 6 7 8 > $RUN\_DIR/$JOBNAME.cmds  
# submit it as a Slurm job array  
sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm  
  
NCORES=12  
MEM=64  
WALLTIME="5:00:00"  
JOBNAME="bowtie2-align-Murdoch"  
# Create the bowtie2 commands commands to repair all read pairs.  
parallel -k --dry-run --rpl "{file2} s:\_R1:\_R2:" --rpl "{sample} s:.+/(.+?)\_R1.+:\1:" "ALIGN\_DIR=$RUN\_DIR/aligned\_reads/bt2\_{2}{3}; mkdir -p \$ALIGN\_DIR && bowtie2 --{2}{3} --rg-id {1 sample} --rg 'SM:{1 sample}' --rg 'LB:{1 sample}' --rg 'PU:E250038400' --rg 'PL:MGI-DNBSEQ-T7' --rg 'CN:Murdoch' -p \$[SLURM\_CPUS\_PER\_TASK - 2] -x $GENOME -1 {1} -2 {1 file2} | bamsormadup tmpfile=\${TMPDIR:-/scratch/project/adna/tmp/}/\${SLURM\_JOB\_NAME} inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] > \$ALIGN\_DIR/{1 sample}.dedup.rg.csorted.bam; unset DISPLAY; qualimap bamqc -bam \$ALIGN\_DIR/{1 sample}.dedup.rg.csorted.bam --java-mem-size=32G -c -gff $GENOME.bed -outdir \$ALIGN\_DIR/{1 sample}\_bamqc" ::: $(find $RUN\_DIR/trimmed\_reads -name "Ar\*\_R1.trimmed.fastq.gz") ::: "fast" "sensitive" "very-sensitive" ::: "-local" "" > $RUN\_DIR/$JOBNAME.cmds  
  
# submit it as a Slurm job array  
sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm  
  
  
JOBNAME="bowtie2-align-AGRF"  
# Create the bowtie2 commands commands to repair all read pairs.  
parallel -k --dry-run --rpl "{file2} s:\_R1:\_R2:" --rpl "{sample} s:.+/(.+?)\_R1.+:\1:" "ALIGN\_DIR=$RUN\_DIR/aligned\_reads/bt2\_{2}{3}; mkdir -p \$ALIGN\_DIR && bowtie2 --{2}{3} --rg-id {1 sample} --rg 'SM:{1 sample}' --rg 'LB:{1 sample}' --rg 'PU:H3HGFDSX2' --rg 'PL:ILLUMINA' --rg 'CN:AGRF' -p \$[SLURM\_CPUS\_PER\_TASK - 2] -x $GENOME -1 {1} -2 {1 file2} | bamsormadup tmpfile=\${TMPDIR:-/scratch/project/adna/tmp/}/\${SLURM\_JOB\_NAME} inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] > \$ALIGN\_DIR/{1 sample}.dedup.rg.csorted.bam; unset DISPLAY; qualimap bamqc -bam \$ALIGN\_DIR/{1 sample}.dedup.rg.csorted.bam --java-mem-size=32G -c -gff $GENOME.bed -outdir \$ALIGN\_DIR/{1 sample}\_bamqc" ::: $(find $RUN\_DIR/trimmed\_reads -name "AR\*\_R1.trimmed.fastq.gz") ::: "fast" "sensitive" "very-sensitive" ::: "-local" "" > $RUN\_DIR/$JOBNAME.cmds  
  
# submit it as a Slurm job array  
sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm  
  
# Multi-bamqc: create/copy a file describing the sample names (sample\_info.txt) to the parent folder and use it in qualimap   
  
find `pwd` -name "\*\_bamqc" | gawk '{sample\_name=gensub(/.+\/(.+)\_bamqc/, "\\1", $1); printf "%s\t%s\n", sample\_name, $1}' > multibamqc.samples  
   
  
QUALIMAP\_MULTI=$( echo "cd $RUN\_DIR ; source ~/.bashrc; conda activate $CONDA\_NAME; unset DISPLAY ; find `pwd` -name \"\*\_bamqc\" | gawk '{sample\_name=gensub(/.+\/(.+)\_bamqc/, \"\\\1\", \$1); printf \"%s\t%s\n\", sample\_name, \$1}' > multibamqc.samples ; qualimap multi-bamqc --java-mem-size=16G -d multibamqc.samples -outformat PDF:HTML -outdir ${BATCH}\_bamqc " | qsub -V -l select=1:ncpus=${NCORES}:mem=32GB,walltime=5:00:00 -N ${QUALIMAP\_JOB:0:11} | egrep -o "^[0-9]+")

#### Compare variant calling tools

Next, we call variants with different variant callers (GATK, FreeBayes, Clair3) and a complete Nextflow pipeline implementing those (Sarek).

### Run Variant Calling with Selected Pipeline (Murdoch 2024)

The selected pipeline (using bwa-mem2 for alignment and freebayes in diploid mode for variant calling (see sections @ref(bwamem-align) and @ref(freebayes)) was then used to call SNPs from the entire population.

#### Process reads and align to the genome

See detailed methods in Section @ref(bwamem-align).

# setup workspace  
CONDA\_NAME="genomics"   
mamba install -n $CONDA\_NAME mosdepth # megadepth   
REF\_DIR="/scratch/project/adna/A\_rabiei/A\_rabiei\_TECAN\_2022/ref\_genome"  
GENOME="$REF\_DIR/ArME14\_v2\_CCDM"  
WORK\_DIR="/scratch/project/adna/A\_rabiei/Murdoch\_WGRS"  
FQ\_DIR="$WORK\_DIR/fungal344"  
# \_H3HGFDSX2\_CCTCCGGTTG-TGGTGTTATG\_L002\_R2.fastq.gz  
  
# DATE=$(date +%d\_%m\_%Y)  
RUN\_DIR="$WORK\_DIR/Batch2024\_FB\_SNP\_calling"  
BAM\_DIR="$RUN\_DIR/aligned\_reads"  
mkdir -p $BAM\_DIR $FQ\_DIR/combined\_libs/trimmed\_reads/QC && cd $RUN\_DIR  
  
NCORES=8  
MEM=32  
WALLTIME="3:00:00"  
JOBNAME="MU-batch24-combine-reads"  
  
# merge files from duplicated libraries  
ls -1 $FQ\_DIR/\*.R1.fq.gz | egrep "E[0-9]+" | sed -r 's:.+\/(.+?).E[0-9]+.R([12]).fq.gz:\1:' | uniq | parallel -a - --dry-run "cat $FQ\_DIR/{1}.\*.R{2}.fq.gz > $FQ\_DIR/combined\_libs/{1}.R{2}.fq.gz" ::: 1 2 > $JOBNAME.cmds  
  
# submit the job array to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/parallel\_jobs\_run.slurm | cut -f 4 -d " ")  
  
  
# copy across control isolates reads from WGS 2020 batch  
ISOLATES="Ar0020|Ar0023|Ar0212|AR0210|AR0022|AR0128" # AR0242|AR0052|  
find /scratch/project/adna/A\_rabiei/AGRF\_CAGRF20114478 -name "\*.fastq.gz" | egrep -i $ISOLATES | parallel --rpl "{target} s:.+\/(.+?)\_H3HGFDSX2\_[CATG-]+\_L002\_R([12]).fastq.gz:\1.R\2.fq.gz:" ln -s {} $FQ\_DIR/combined\_libs/{target}  
  
FQ\_DIR="$FQ\_DIR/combined\_libs"  
  
JOBNAME="MU-batch24-fastp-bwa"  
NCORES=12  
MEM=96  
WALLTIME="3:00:00"  
RGPM="DNBseq\_T7"  
RGPL="MGI"  
RGPU="E250076899" # also E250076916  
RGCN="MU\_SABC"  
  
find $FQ\_DIR/\*.R1.fq.gz | parallel -k --dry-run --rpl "{file2} s:.R1:.R2:; uq()" --rpl "{sample} s:.+\/(.+?).R1.fq.gz:\1:" "fastp -i {} -I {file2} --detect\_adapter\_for\_pe -c -l 30 -p -w \$SLURM\_CPUS\_PER\_TASK -z 7 -o $FQ\_DIR/trimmed\_reads/{sample}\_R1.trimmed.fastq.gz -O $FQ\_DIR/trimmed\_reads/{sample}\_R2.trimmed.fastq.gz -j $FQ\_DIR/trimmed\_reads/QC/{sample}.fastp.json -h $FQ\_DIR/trimmed\_reads/QC/{sample}.fastp.html && bwa-mem2 mem -R \"@RG\tID:{sample}\tSM:{sample}\tLB:{sample}\tPU:$RGPU\tPL:$RGPL\tPM:$RGPM\tCN:$RGCN\" -t \$[SLURM\_CPUS\_PER\_TASK - 2] $GENOME.fa $FQ\_DIR/trimmed\_reads/{sample}\_R1.trimmed.fastq.gz $FQ\_DIR/trimmed\_reads/{sample}\_R2.trimmed.fastq.gz | bamsormadup tmpfile=\$TMPDIR/bamsormadup\_\$(hostname)\_\$SLURM\_ARRAY\_JOB\_ID inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] indexfilename=$BAM\_DIR/{sample}.dedup.rg.csorted.bam.bai > $BAM\_DIR/{sample}.dedup.rg.csorted.bam ; unset DISPLAY ; qualimap bamqc -bam $BAM\_DIR/{sample}.dedup.rg.csorted.bam --java-mem-size=32G -c -gff $GENOME.bed -outdir $BAM\_DIR/{sample}\_bamqc; mosdepth -t \$SLURM\_CPUS\_PER\_TASK -x -n $BAM\_DIR/{sample}\_bamqc/{sample} $BAM\_DIR/{sample}.dedup.rg.csorted.bam" > $RUN\_DIR/$JOBNAME.cmds  
  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")  
  
# extract coverage information  
find aligned\_reads -name "genome\_results.txt" | xargs grep "mean coverageData" | sed 's/X$//' | gawk '{sum+=$NF}END{print "# Mean Coverage =", sum/NR}' > MU-batch24-coverage-summary.txt  
find aligned\_reads -name "genome\_results.txt" | xargs grep "mean coverageData" >> MU-batch24-coverage-summary.txt

#### Calling variants using Freebayes

See detailed methods in Section @ref(freebayes).

WORK\_DIR="/scratch/project/adna/A\_rabiei/Murdoch\_WGRS"  
RUN\_DIR="$WORK\_DIR/Batch2024\_FB\_SNP\_calling"  
FQ\_DIR="$WORK\_DIR/fungal344"  
REF\_DIR="/scratch/project/adna/A\_rabiei/A\_rabiei\_TECAN\_2022/ref\_genome"  
GENOME="$REF\_DIR/ArME14\_v2\_CCDM"  
  
CONDA\_NAME="genomics"  
  
BAM\_DIR="$RUN\_DIR/aligned\_reads"  
  
# bring in BAM files (if not prepared in the previous step)  
# select isolates  
# ISOLATES="Ar0020|Ar0023|Ar0212|AR0210|AR0022|AR0128" # AR0242|AR0052|  
# find /scratch/project/adna/A\_rabiei/AGRF\_gatk\_13\_03\_2025/aligned\_reads -maxdepth 1 -name "\*.rg.csorted.bam\*" | egrep -i $ISOLATES | parallel ln -s {} $BAM\_DIR/  
# ln -s /scratch/project/adna/A\_rabiei/Murdoch\_WGRS/GATK\_Murdoch\_WGRS\_04\_04\_2025/aligned\_reads/\*.rg.csorted.bam\* $BAM\_DIR/  
  
# Distributed freebayes (each node runs freebayes-parallel on one contig)  
# start\_interactive\_job  
conda activate $CONDA\_NAME  
# fix library dependencies  
find $CONDA\_PREFIX -name "libtabixpp.so\*" | parallel ln -s {} {.}.0  
# ln -s $CONDA\_PREFIX/lib/libtabixpp.so.1 $CONDA\_PREFIX/lib/libtabixpp.so.0  
# split each contig/chromosome to smaller 1e6 bits  
# prepare BAM files  
JOBNAME="prep\_bams"  
NCORES=2  
MEM=16  
WALLTIME="1:00:00"  
# submit it as a Slurm job  
echo "~/bin/split\_ref\_by\_bai\_datasize.py -s 1e6 -r $GENOME.fa.fai $(ls -1S $BAM\_DIR/\*.dedup.rg.csorted.bam | tail -n1) > $RUN\_DIR/ArME14\_target\_1e6\_regions\_chr.tsv" > $JOBNAME.cmds  
# submit the job   
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
  
JOBNAME="Batch2024\_WGRS\_FB\_diploid"  
NCORES=10  
MEM=64  
WALLTIME="10:00:00"  
# RUN\_DIR=$WORK\_DIR/SNP\_calling\_24\_01\_2025  
PLOIDY=2  
MIN\_DP=7  
MAX\_DP=100000  
# create a folder  
mkdir -p $RUN\_DIR/FB\_array\_output  
# prepare commands  
BAM\_FILES=$( find $BAM\_DIR -maxdepth 1 -name "\*.rg.csorted.bam" | xargs )  
cut -f1 $GENOME.fa.fai | parallel --dry-run "freebayes-parallel <(grep '{}' $RUN\_DIR/ArME14\_target\_1e6\_regions\_chr.tsv | gawk '{printf \"%s:%s-%s\n\", \$1, \$2, \$3}') \$SLURM\_CPUS\_PER\_TASK -f $GENOME.fa --genotype-qualities -g $MAX\_DP -C $MIN\_DP -p $PLOIDY $BAM\_FILES > $RUN\_DIR/FB\_array\_output/{}.combined.vcf" > $RUN\_DIR/$JOBNAME.cmds  
  
# exit interactive job  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")  
  
FAILED\_TASKS=$(sacct -n -X -j $ARRAY\_ID -o state%20,jobid%20 | grep -v COMPLETED | gawk '{print $2}' | cut -d"\_" -f2 | paste -sd,)  
  
  
# merge VCFs  
JOBNAME="freebayes-merge"  
echo "cat $RUN\_DIR/FB\_array\_output/ArME14\_ctg\_\*.combined.vcf | vcffirstheader | vcfstreamsort -w 1000 | vcfuniq | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf.gz; bcftools index A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf.gz" > $JOBNAME.cmds  
# submit job to cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")

#### Create and filter final variant file

See detailed methods in Section @ref(vcf-filter).

CONDA\_NAME="genomics"  
# Recode genotypes as missing if below a certain threshold, such as genotyping quality or depth (GQ:DP)   
# filter only polymorphic bi-allelic SNPs, using QUAL>20, 7<DP<100000  
  
# filter Freebayes variants with SnpSift and vcftools (wipe any heterozygote genotype with DP<7 with bcftools)  
QUAL=30 # 30  
MQ=30  
MAX\_DP=100000  
MIN\_DP=10  
IND\_DP=7  
JOBNAME="Batch2024-wgrs-fb-filter"  
  
echo "bcftools filter -S . -e \"GT=='het' | FMT/DP<$MIN\_DP\" $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf.gz -O v | SnpSift filter \"( QUAL>=$QUAL ) & ( countRef()>=1 & countVariant()>=1 )\" | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz; bcftools index $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz   
vcftools --gzvcf $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz --recode --recode-INFO-all --minQ $QUAL --remove-indels -c | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.snps.Q$QUAL.noHet.poly.recode.vcf.gz ; bcftools index $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.snps.Q$QUAL.noHet.poly.recode.vcf.gz  
vcftools --gzvcf $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz --recode --recode-INFO-all --minQ $QUAL --keep-only-indels -c | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.indels.Q$QUAL.noHet.poly.recode.vcf.gz; bcftools index $RUN\_DIR/A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.indels.Q$QUAL.noHet.poly.recode.vcf.gz "> $RUN\_DIR/$JOBNAME.cmds  
  
NCORES=6  
MEM=32  
WALLTIME="1:00:00"  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
  
#bcftools filter -S . -e "GT=='het' | FMT/DP<$MIN\_DP" A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf.gz -O v | SnpSift filter "( QUAL>=$QUAL ) & ( DP<=$MAX\_DP ) & ( DP>=$MIN\_DP ) & ( countRef()>=1 & countVariant()>=1 )" | vcftools --vcf - --recode --recode-INFO-all --minQ $QUAL --max-missing 0.75 --remove-indels -c | bgzip -o A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.GT75.noRep.noHet.poly.recode.vcf.gz && tabix A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.GT75.noRep.noHet.poly.recode.vcf.gz  
# generate stats   
JOBNAME="bcftools\_stats"  
WALLTIME=2:00:00  
MEM=32  
NCORES=8  
find . -name "\*.vcf.gz" | parallel --dry-run "bcftools stats -s - {} > {.}.bcfstats.txt" > $JOBNAME.cmds  
# send to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/parallel\_jobs\_run.slurm | cut -f 4 -d " ")

An in-house R script (estimate\_error\_rates\_vcf\_files.R) was used to estimate the error rates based on the presence of duplicated samples.

#### MutilQC

See detailed methods in Section @ref(multiqc).

WORK\_DIR="/scratch/project/adna/A\_rabiei/Murdoch\_WGRS"  
RUN\_DIR="$WORK\_DIR/Batch2024\_FB\_SNP\_calling"  
cd $RUN\_DIR  
NCORES=8  
MEM=32  
WALLTIME="10:00:00"  
JOBNAME="Batch2024-wgrs-multiqc"  
# link fastp reports  
ln -s $FQ\_DIR/trimmed\_reads/QC $RUN\_DIR/  
# multiqc report  
MULTIQC\_JOB=QC\_$(date +%d\_%m\_%Y)  
# Prepare MultiQC command  
echo "multiqc --interactive -z --force -i $MULTIQC\_JOB -o $MULTIQC\_JOB ." > $JOBNAME.cmds  
# submit the job   
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
# Done!  
mkdir -p slurm\_logs  
mv \*.log slurm\_logs/  
# Copy files to SharePoint  
rclone copy -P --exclude "\*\*/\*.html" $RUN\_DIR GRDC\_rabiei:General/Projects/Hayley\_PhD/Genetics/Murdoch\_WGRS/Batch2024\_FB\_SNP\_calling  
# Copy html files to SharePoint  
rclone copy -P --ignore-checksum --ignore-size --include "\*\*/\*.html" $RUN\_DIR GRDC\_rabiei:General/Projects/Hayley\_PhD/Genetics/Murdoch\_WGRS/Batch2024\_FB\_SNP\_calling

### Run Variant Calling with Selected Pipeline (AGRF 2021)

The selected pipeline (using bwa-mem2 for alignment and freebayes in diploid mode for variant calling (see sections @ref(bwamem-align) and @ref(freebayes)) was then used to call SNPs from the entire population.

#### Process reads and align to the genome

See detailed methods in Section @ref(bwamem-align).

# setup workspace  
CONDA\_NAME="genomics"   
mamba install -n $CONDA\_NAME mosdepth # megadepth   
WORK\_DIR="/scratch/project/adna/A\_rabiei"  
REF\_DIR="/scratch/project/adna/A\_rabiei/ref\_genome"  
rclone copy -P GRDC\_rabiei:General/GRI2007-001RTX-data/A\_rabiei\_TECAN\_2022/ref\_genome $REF\_DIR  
GENOME="$REF\_DIR/ArME14\_v2\_CCDM"  
  
FQ\_DIR="/scratch/project/adna/A\_rabiei/AGRF\_snippy\_05\_03\_2025"  
rclone copy -P --include "\*.trimmed.fastq.gz" GRDC\_rabiei:General/GRI2007-001RTX-data/A\_rabiei\_WGS/Variant\_Calling/AGRF\_snippy\_05\_03\_2025 $FQ\_DIR  
# \_H3HGFDSX2\_CCTCCGGTTG-TGGTGTTATG\_L002\_R2.fastq.gz  
  
# DATE=$(date +%d\_%m\_%Y)  
RUN\_DIR="$WORK\_DIR/AGRF2021\_FB\_SNP\_calling"  
BAM\_DIR="$RUN\_DIR/aligned\_reads"  
mkdir -p $BAM\_DIR $FQ\_DIR/trimmed\_reads/QC && cd $RUN\_DIR  
  
JOBNAME="AGRF2021-bwa-mem2"  
NCORES=12  
MEM=96  
WALLTIME="3:00:00"  
RGPM="NovaSeq"  
RGPL="ILLUMINA"  
RGPU="H3HGFDSX2"  
RGCN="AGRF"  
  
  
find $FQ\_DIR -name "\*\_R1.trimmed.fastq.gz" | parallel -k --dry-run --rpl "{file2} s:\_R1:\_R2:" --rpl "{sample} s:.+\/(.+?)\_R1.+:\1:" "bwa-mem2 mem -R \"@RG\tID:{sample}\tSM:{sample}\tLB:{sample}\tPU:$RGPU\tPL:$RGPL\tPM:$RGPM\tCN:$RGCN\" -t \$[SLURM\_CPUS\_PER\_TASK - 2] $GENOME.fa {} {file2} | bamsormadup tmpfile=\$TMPDIR/bamsormadup\_\$(hostname)\_\$SLURM\_ARRAY\_JOB\_ID inputformat=sam threads=\$[SLURM\_CPUS\_PER\_TASK - 2] indexfilename=$BAM\_DIR/{sample}.dedup.rg.csorted.bam.bai > $BAM\_DIR/{sample}.dedup.rg.csorted.bam ; unset DISPLAY ; qualimap bamqc -bam $BAM\_DIR/{sample}.dedup.rg.csorted.bam --java-mem-size=32G -c -gff $GENOME.bed -outdir $BAM\_DIR/{sample}\_bamqc; mosdepth -t \$SLURM\_CPUS\_PER\_TASK -x -n $BAM\_DIR/{sample}\_bamqc/{sample} $BAM\_DIR/{sample}.dedup.rg.csorted.bam" > $RUN\_DIR/$JOBNAME.cmds  
  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")  
  
# extract coverage information  
find aligned\_reads -name "genome\_results.txt" | xargs grep "mean coverageData" | sed 's/X$//' | gawk '{sum+=$NF}END{print "# Mean Coverage =", sum/NR}' > AGRF2021-coverage-summary.txt  
find aligned\_reads -name "genome\_results.txt" | xargs grep "mean coverageData" >> AGRF2021-coverage-summary.txt

#### Calling variants using Freebayes

See detailed methods in Section @ref(freebayes).

WORK\_DIR="/scratch/project/adna/A\_rabiei"  
REF\_DIR="/scratch/project/adna/A\_rabiei/ref\_genome"  
RUN\_DIR="$WORK\_DIR/AGRF2021\_FB\_SNP\_calling"  
FQ\_DIR="/scratch/project/adna/A\_rabiei/AGRF\_snippy\_05\_03\_2025"  
  
GENOME="$REF\_DIR/ArME14\_v2\_CCDM"  
  
CONDA\_NAME="genomics"  
  
BAM\_DIR="$RUN\_DIR/aligned\_reads"  
  
# bring in BAM files (if not prepared in the previous step)  
# select isolates  
# ISOLATES="Ar0020|Ar0023|Ar0212|AR0210|AR0022|AR0128" # AR0242|AR0052|  
# find /scratch/project/adna/A\_rabiei/AGRF\_gatk\_13\_03\_2025/aligned\_reads -maxdepth 1 -name "\*.rg.csorted.bam\*" | egrep -i $ISOLATES | parallel ln -s {} $BAM\_DIR/  
# ln -s /scratch/project/adna/A\_rabiei/Murdoch\_WGRS/GATK\_Murdoch\_WGRS\_04\_04\_2025/aligned\_reads/\*.rg.csorted.bam\* $BAM\_DIR/  
  
# Distributed freebayes (each node runs freebayes-parallel on one contig)  
# start\_interactive\_job  
conda activate $CONDA\_NAME  
# fix library dependencies  
find $CONDA\_PREFIX -name "libtabixpp.so\*" | parallel ln -s {} {.}.0  
# ln -s $CONDA\_PREFIX/lib/libtabixpp.so.1 $CONDA\_PREFIX/lib/libtabixpp.so.0  
# split each contig/chromosome to smaller 1e6 bits  
# prepare BAM files  
JOBNAME="prep\_ref"  
NCORES=2  
MEM=16  
WALLTIME="1:00:00"  
# submit it as a Slurm job  
echo "~/bin/split\_ref\_by\_bai\_datasize.py -s 1e6 -r $GENOME.fa.fai $(ls -1S $BAM\_DIR/\*.dedup.rg.csorted.bam | tail -n1) > $RUN\_DIR/ArME14\_target\_1e6\_regions\_chr.tsv" > $JOBNAME.cmds  
# submit the job   
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
  
JOBNAME="AGRF2021\_WGRS\_FB\_diploid"  
NCORES=10  
MEM=64  
WALLTIME="10:00:00"  
# RUN\_DIR=$WORK\_DIR/SNP\_calling\_24\_01\_2025  
PLOIDY=2  
MIN\_DP=7  
MAX\_DP=100000  
# create a folder  
mkdir -p $RUN\_DIR/FB\_array\_output  
# prepare commands  
BAM\_FILES=$( find $BAM\_DIR -maxdepth 1 -name "\*.rg.csorted.bam" | xargs )  
cut -f1 $GENOME.fa.fai | parallel --dry-run "freebayes-parallel <(grep '{}' $RUN\_DIR/ArME14\_target\_1e6\_regions\_chr.tsv | gawk '{printf \"%s:%s-%s\n\", \$1, \$2, \$3}') \$SLURM\_CPUS\_PER\_TASK -f $GENOME.fa --genotype-qualities -g $MAX\_DP -C $MIN\_DP -p $PLOIDY $BAM\_FILES > $RUN\_DIR/FB\_array\_output/{}.combined.vcf" > $RUN\_DIR/$JOBNAME.cmds  
  
# exit interactive job  
# submit to the cluster  
ARRAY\_ID=$(sbatch -a 1-$(cat $RUN\_DIR/$JOBNAME.cmds | wc -l) --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/array.slurm | cut -f 4 -d " ")  
  
FAILED\_TASKS=$(sacct -n -X -j $ARRAY\_ID -o state%20,jobid%20 | grep -v COMPLETED | gawk '{print $2}' | cut -d"\_" -f2 | paste -sd,)  
  
  
# merge VCFs  
JOBNAME="freebayes-merge"  
echo "cat $RUN\_DIR/FB\_array\_output/ArME14\_ctg\_\*.combined.vcf | vcffirstheader | vcfstreamsort -w 1000 | vcfuniq | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.vcf.gz; bcftools index A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.vcf.gz" > $JOBNAME.cmds  
# submit job to cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")

#### Create and filter final variant file

See detailed methods in Section @ref(vcf-filter).

CONDA\_NAME="genomics"  
# Recode genotypes as missing if below a certain threshold, such as genotyping quality or depth (GQ:DP)   
# filter only polymorphic bi-allelic SNPs, using QUAL>20, 7<DP<100000  
  
# filter Freebayes variants with SnpSift and vcftools (wipe any heterozygote genotype with DP<7 with bcftools)  
QUAL=30 # 30  
MQ=30  
MAX\_DP=100000  
MIN\_DP=10  
IND\_DP=7  
JOBNAME="AGRF2021-wgs-fb-filter"  
  
echo "bcftools filter -S . -e \"GT=='het' | FMT/DP<$MIN\_DP\" $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.vcf.gz -O v | SnpSift filter \"( QUAL>=$QUAL ) & ( countRef()>=1 & countVariant()>=1 )\" | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz; bcftools index $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz   
vcftools --gzvcf $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz --recode --recode-INFO-all --minQ $QUAL --remove-indels -c | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.snps.Q$QUAL.noHet.poly.recode.vcf.gz ; bcftools index $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.snps.Q$QUAL.noHet.poly.recode.vcf.gz  
vcftools --gzvcf $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.noHet.poly.recode.vcf.gz --recode --recode-INFO-all --minQ $QUAL --keep-only-indels -c | bgzip -@ \$SLURM\_CPUS\_PER\_TASK -c > $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.indels.Q$QUAL.noHet.poly.recode.vcf.gz; bcftools index $RUN\_DIR/A\_rabiei\_AGRF\_WGS2021\_ArME14\_v2.bwa2.fb.diploid.indels.Q$QUAL.noHet.poly.recode.vcf.gz "> $RUN\_DIR/$JOBNAME.cmds  
  
NCORES=6  
MEM=32  
WALLTIME="1:00:00"  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
  
#bcftools filter -S . -e "GT=='het' | FMT/DP<$MIN\_DP" A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.vcf.gz -O v | SnpSift filter "( QUAL>=$QUAL ) & ( DP<=$MAX\_DP ) & ( DP>=$MIN\_DP ) & ( countRef()>=1 & countVariant()>=1 )" | vcftools --vcf - --recode --recode-INFO-all --minQ $QUAL --max-missing 0.75 --remove-indels -c | bgzip -o A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.GT75.noRep.noHet.poly.recode.vcf.gz && tabix A\_rabiei\_2024\_Murdoch\_WGRS\_ArME14\_v2.bwa2.fb.diploid.Q$QUAL.GT75.noRep.noHet.poly.recode.vcf.gz  
# generate stats   
JOBNAME="bcftools\_stats"  
WALLTIME=2:00:00  
MEM=32  
NCORES=8  
find . -name "\*.vcf.gz" | parallel --dry-run "bcftools stats -s - {} > {.}.bcfstats.txt" > $JOBNAME.cmds  
# send to the cluster  
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/parallel\_jobs\_run.slurm | cut -f 4 -d " ")

An in-house R script (estimate\_error\_rates\_vcf\_files.R) was used to estimate the error rates based on the presence of duplicated samples.

#### MutilQC

See detailed methods in Section @ref(multiqc).

WORK\_DIR="/scratch/project/adna/A\_rabiei"  
REF\_DIR="/scratch/project/adna/A\_rabiei/ref\_genome"  
RUN\_DIR="$WORK\_DIR/AGRF2021\_FB\_SNP\_calling"  
cd $RUN\_DIR  
NCORES=8  
MEM=32  
WALLTIME="10:00:00"  
JOBNAME="AGRF2021-wgrs-multiqc"  
# link fastp reports  
mkdir -p $RUN\_DIR/QC  
find $FQ\_DIR -name "\*.json" | xargs ln -s {} $RUN\_DIR/QC/  
# ln -s $FQ\_DIR/trimmed\_reads/QC $RUN\_DIR/  
# multiqc report  
MULTIQC\_JOB=$JOBNAME\_$(date +%d\_%m\_%Y)  
# Prepare MultiQC command  
echo "multiqc --interactive -z --force -i $MULTIQC\_JOB -o $MULTIQC\_JOB ." > $JOBNAME.cmds  
# submit the job   
JOB\_ID=$(sbatch --job-name=$JOBNAME --cpus-per-task=$NCORES --mem=${MEM}G --time=$WALLTIME --export=ALL,CMDS\_FILE=$RUN\_DIR/$JOBNAME.cmds,CONDA\_NAME=$CONDA\_NAME ~/bin/serial\_jobs\_run.slurm | cut -f 4 -d " ")  
# Done!  
mkdir -p logs  
mv \*.log logs/  
# Copy files to SharePoint  
# Copy html files to SharePoint  
rclone copy -P --ignore-checksum --ignore-size --include "\*\*/\*.html" $RUN\_DIR GRDC\_rabiei:General/GRI2007-001RTX-data/A\_rabiei\_WGS/Variant\_Calling/AGRF2021\_FB\_SNP\_calling  
# copy rest of files  
rclone copy -P --exclude "\*\*/\*.html" $RUN\_DIR GRDC\_rabiei:General/GRI2007-001RTX-data/A\_rabiei\_WGS/Variant\_Calling/AGRF2021\_FB\_SNP\_calling

## General information

This document was last updated on 21 August 2025 using R Markdown (built with R version 4.4.3 (2025-02-28 ucrt)). The source code for this website can be found on <https://github.com/IdoBar/HPC_SNP_calling_documentation>.

Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. It is especially powerful at authoring documents and reports which include code and can execute code and use the results in the output. For more details on using R Markdown see <http://rmarkdown.rstudio.com>, [R Markdown: The Definitive Guide](https://bookdown.org/yihui/rmarkdown/) and [Rmarkdown cheatsheet](https://rstudio.github.io/cheatsheets/html/rmarkdown.html).

## Bibliography

Cleary JG, Braithwaite R, Gaastra K, et al. (2015) [Comparing variant call files for performance benchmarking of next-generation sequencing variant calling pipelines](https://doi.org/10.1101/023754).

Danecek P, Auton A, Abecasis G, et al. (2011) The variant call format and VCFtools. Bioinformatics 27:2156–2158. doi: [10.1093/bioinformatics/btr330](https://doi.org/10.1093/bioinformatics/btr330)

Danecek P, Bonfield JK, Liddle J, et al. (2021) Twelve years of SAMtools and BCFtools. GigaScience 10:giab008. doi: [10.1093/gigascience/giab008](https://doi.org/10.1093/gigascience/giab008)

Ewels P, Magnusson M, Lundin S, Käller M (2016) MultiQC: summarize analysis results for multiple tools and samples in a single report. Bioinformatics 32:3047–3048. doi: [10.1093/bioinformatics/btw354](https://doi.org/10.1093/bioinformatics/btw354)

Garrison E, Marth G (2012) [Haplotype-based variant detection from short-read sequencing](https://doi.org/10.48550/ARXIV.1207.3907).

Heldenbrand JR, Baheti S, Bockol MA, et al. (2019) Recommendations for performance optimizations when using GATK3.8 and GATK4. BMC Bioinf 20:557. doi: [10.1186/s12859-019-3169-7](https://doi.org/10.1186/s12859-019-3169-7)

Li H (2011) A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. Bioinformatics 27:2987–2993. doi: [10.1093/bioinformatics/btr509](https://doi.org/10.1093/bioinformatics/btr509)

Li X, Muñoz JF, Gade L, et al. (2023) Comparing genomic variant identification protocols for *Candida auris*. Microb Genomics 9:979. doi: [10.1099/mgen.0.000979](https://doi.org/10.1099/mgen.0.000979)

Okonechnikov K, Conesa A, García-Alcalde F (2016) Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data. Bioinformatics 32:292–294. doi: [10.1093/bioinformatics/btv566](https://doi.org/10.1093/bioinformatics/btv566)

Pedersen BS, Quinlan AR (2018) Mosdepth: quick coverage calculation for genomes and exomes. Bioinformatics 34:867–868. doi: [10.1093/bioinformatics/btx699](https://doi.org/10.1093/bioinformatics/btx699)

Ruden DM, Cingolani P, Patel VM, et al. (2012) Using *drosophila melanogaster* as a model for genotoxic chemical mutational studies with a new program, SnpSift. Toxicogenomics 3:35. doi: [10.3389/fgene.2012.00035](https://doi.org/10.3389/fgene.2012.00035)

Tischler G, Leonard S (2014) biobambam: tools for read pair collation based algorithms on BAM files. Source Code Biol Med 9:13. doi: [10.1186/1751-0473-9-13](https://doi.org/10.1186/1751-0473-9-13)

Van der Auwera GA, Carneiro MO, Hartl C, et al. (2002) [From FastQ data to high-confidence variant calls: the genome analysis toolkit best practices pipeline](http://onlinelibrary.wiley.com/doi/10.1002/0471250953.bi1110s43/abstract). Current Protocols in Bioinformatics

Vasimuddin Md, Misra S, Li H, Aluru S (2019) [Efficient architecture-aware acceleration of BWA-MEM for multicore systems](https://doi.org/10.1109/IPDPS.2019.00041). 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp 314–324

Yue J, VanBuren R, Liu J, et al. (2022) [SunUp and Sunset genomes revealed impact of particle bombardment mediated transformation and domestication history in papaya](https://www.nature.com/articles/s41588-022-01068-1). Nature Genetics 54:715–724.