Bioinformatics Pipeline for *Zostera marina* (Eelgrass)

About this resource:
This document provides a complete, step-by-step workflow for processing and analyzing whole-genome resequencing data from *Zostera marina* (eelgrass). It includes shell scripts and SLURM job submission commands for each major stage: adapter trimming, sequence concatenation, read mapping, variant calling, SNP filtering, and genotype imputation. The pipeline also covers downstream processing steps such as depth and missingness calculations, population- level VCF filtering, and genotype matrix generation for population genetic analyses. Designed for use on high-performance computing clusters, this pipeline supports reproducible genomic data analysis for ecological, evolutionary, and restoration genetics applications.

This resource is intended for bioinformaticians, population geneticists, and ecological genomics researchers working with *Zostera marina* or other marine species. It is specifically designed for users with intermediate to advanced experience in Linux-based high- performance computing environments, shell scripting, and genomics software such as GATK, vcftools, and samtools. This pipeline is suited for those conducting whole-genome resequencing projects who require a scalable, reproducible workflow for variant discovery, filtering, and population-level genotype analysis.

Citation: Kamel, S. & Scavo, K. (2025). *Bioinformatics Pipeline for Zostera marina (Eelgrass)*. NERRS Science Collaborative.

About the project:
This resource was developed through a 2022-2025 Collaborative Research project titled *Evaluating and Enhancing Eelgrass Resiliency and Restoration Potential in a Changing Climate*.

In the lower Chesapeake Bay, Virginia, warmer water temperatures in recent years have resulted in large scale diebacks of eelgrass meadows (Zostera marina). In contrast, many eelgrass populations in Back Sound, North Carolina appear to be more resilient to warming water temperatures. Understanding the drivers of these regional differences in eelgrass resilience could help more effectively restore eelgrass meadows in a changing climate.

With a network of the intended users from reserves, state agencies, and Chesapeake Bay nonprofits, this project compared resiliency traits of eelgrass populations in Virginia and North Carolina by conducting reciprocal restoration trials and genomic sequencing. The project results indicate the importance of seed sources in potential future eelgrass restoration, in addition to site selection.

This webpage provides more information about the project.

Bioinformatics Pipeline for *Zostera marina*

Trim adapters

****

```
# Define the input and output directories
input_dir="/scratch/lrs8888/NERRS"
output_dir="/scratch/lrs8888/workflow/NERRS_trimmed"
mkdir -p $output_dir

# Define the output file
trims_commands="trims_commands_NERRS.txt"
rm -f $trims_commands

# Iterate over every fastq file and create a bbduk command
for file in "$input_dir"/*.fastq.gz; do
    # If this is an R1 file, get the matching R2 name
    if [[ "${file}" == *_R1_001.fastq.gz ]]; then
       in1="${file}"
       in2="${file/_R1_/_R2_}"
       # If the second file exists, create the BBDuk command to
execute
        if [[ -f "${in2}" ]]; then
            base_in1=$(basename "${in1}")
            base_in2=$(basename "${in2}")
            out1="${output_dir}/${base_in1%.fastq.gz}.clean.fastq.gz"
            out2="${output_dir}/${base_in2%.fastq.gz}.clean.fastq.gz"

            echo "bbduk.sh in1=${in1} in2=${in2} out1=${out1}
out2=${out2} ref=adapters.fa ktrim=r mink=11 hdist=1 tpe tbo trimq=6
maq=10 maxns=1 minlen=50" >> $trims_commands
        else
            # Error, oops
            echo "Warning: Corresponding R2 file for ${in1} not
found."
        fi
    fi
done
```

```
Concatenate Sequences
*************************************************************

cat_temp=$(mktemp)

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a01_NERRS_trimmed"
output_dir="/scratch/lrs8888/workflow/a02_NERRS_concat"
mkdir -p $output_dir

commands_file="concat_commands.sh"

for file in "$input_dir"/*.clean.fastq.gz; do
    # Adjust the regex to match the sample ID correctly
    sample_id=$(echo "$file" | grep -oP '\d{5}Kam_[^_]+_[^_]+')

    # Extract R1 or R2
    r1_or_r2=$(echo "$file" | grep -oP '_R\d_' | sed 's/_//g')

    # Construct the output file name

    output_file="${output_dir}/${sample_id}_${r1_or_r2}.fastq.gz"

    # Write the cat command to the file cat3

    echo cat "${input_dir}/${sample_id}*${r1_or_r2}*.clean.fastq.gz >
$output_file" >> $cat_temp

done


# Deduplicate the cat commands in nerrs_cat_commands
echo "#! /bin/bash" > $commands_file
echo "#SARRAY --start" >> $commands_file
cat $cat_temp | sort | uniq >> $commands_file
```

Genome Mapping
****

```
# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a02_NERRS_concat"
output_dir="/scratch/lrs8888/workflow/a03_NERRS_mapping"
mkdir -p $output_dir

commands_file="genome_mapping_commands.sh"

echo -e "#! /bin/bash
#SBATCH --job-name=genomemapping
#SBATCH --time=1-00:00:00
#SBATCH --mem=16G
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8

spack load bwa

#SARRAY --start" > $commands_file

for file_R1 in "$input_dir"/*_R1.fastq.gz; do

  #echo $file_R1
  file_R2="${file_R1/_R1/_R2}";

  base_name=$(basename "$file_R1" _R1.fastq.gz);

  echo bwa mem -t 8 -M zm_gen2 "$file_R1" "$file_R2" '>'
"${output_dir}/${base_name}.sam" >> $commands_file

done
```

```
Convert SAM files to BAM files
****************************************************************

# Define the input and output directories
inputdir="a03_NERRS_mapping"
outputdir="a04_NERRS_bamfiles"
mkdir -p $outputdir

command_file="sam_conversion_commands.sh"


cat <<EOL > $command_file
#!/bin/bash
#SBATCH --job-name=conversionsDec16
#SBATCH --mem=16G
#SBATCH --time=1-00:00:00
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1

spack load samtools@1.13

#SARRAY --start
EOL

for samfile in "$inputdir"/*.sam; do
    base=$(basename "${samfile}" .sam)

    echo "samtools view -Sb $samfile > ${outputdir}/$base.bam &&
samtools sort ${outputdir}/$base.bam -o
${outputdir}/${base}_sort.bam" >> $command_file
done
```

Call Haplotypes
****

```
#!/bin/bash

# Define the input directory
inputdir="/scratch/lrs8888/workflow/a04_NERRS_bamfiles"

commands_file="variants_commands.sh"

# Create or overwrite the output file. This creates the column header
row
echo "#! /bin/bash" > $commands_file
echo "#SBATCH --time=7-00:00:00" >>$commands_file
echo "#SBATCH --mem=32G" >>$commands_file
echo "#SBATCH --ntasks=1" >>$commands_file
echo "#SARRAY --start" >> $commands_file

# Loop through each .bam file in the input directory
for bamfile in "$inputdir"/*_sort.bam; do
        base=$(basename $bamfile)
        echo "./get_variants_script.sh $bamfile" >> $commands_file
done

----------------------
#! /bin/bash

# load modules
spack load gatk samtools@1.13

# Define the output directory
output_dir="a05_NERRS_variants"
mkdir -p $output_dir

input_file=$1

# define the output file names
basename=$(basename $input_file _sort.bam)
markedup="$output_dir/${basename}_markedup.bam"
metrics="$output_dir/${basename}_marked_dup_metrics.txt"
markedup_rg="$output_dir/${basename}_markedup_rg.bam"
vcf_output="$output_dir/${basename}.g.vcf"

# mark duplicates
gatk --java-options "-Xmx20G" MarkDuplicates -I $input_file -O
$markedup -M $metrics -MAX_FILE_HANDLES 1000
```

```
#add header
samtools addreplacerg -r 'ID:group1' -r 'LB:lib1' -r 'PL:illumina' -r
'PU:unit1' -r "SM:${basename}" -o $markedup_rg $markedup

# index
samtools index $markedup_rg

# call SNPs
gatk --java-options "-Xmx40G" HaplotypeCaller \
    -R Zmarina_668_v2.0.fa \
    -I $markedup_rg \
    -O $vcf_output \
    -ERC GVCF
```

Mapping Rates
****

```
#! /bin/bash
input_file=$1

# Load modules
spack load samtools@1.13

# Extract the sample name (optional, if needed for naming)
sample=$(basename "$input_file" _sort.bam)

# Calculate stats
avg_read_depth_per_position=$(samtools depth -a "$input_file" | awk
'{c++;s+=$3}END{print s/c}' &)
percent_positions_covered_by_reads=$(samtools depth -a "$input_file"
| awk '{c++; if($3>0) total+=1}END{print (total/c)*100}' &)
percent_mapped_reads=$(samtools flagstat "$input_file" | awk -F
"[(|%]" 'NR == 7 {print $2}')
wait

# Write this row to the output file
echo -e
"${sample}\t${avg_read_depth_per_position}\t${percent_positions_cover
ed_by_reads}\t${percent_mapped_reads}"

#!/bin/bash

# Define the input and output directories
inputdir="/scratch/lrs8888/workflow/a04_NERRS_bamfiles"
outputfile="mapping_stats_NERRS.txt"
tmp_scratch_dir="/scratch/kamels_stats"
commands_file="mapping_stats_commands.sh"

mkdir -p $tmp_scratch_dir

# Create or overwrite the output file. This creates the column header
row
echo -e
"Sample\tAvgReadDepth\tPercentPositionsCovered\tPercentMappedReads" >
$outputfile

echo "#! /bin/bash" > $commands_file
echo "#SBATCH --ntasks=3" >>$commands_file
echo "#SBATCH --time=7-00:00:00" >>$commands_file
echo "#SBATCH --mem=16G" >>$commands_file
echo "#SARRAY --start" >> $commands_file
```

```
# Loop through each .bam file in the input directory
for bamfile in "$inputdir"/*_sort.bam; do
        base=$(basename $bamfile)
        echo "./get_mapping_stats_single_file.sh $bamfile >
$tmp_scratch_dir/$base.stats.part" >> $commands_file
done
```

Import into Genomics DB
****

```
#!/bin/bash
#SBATCH --job-name=import
#SBATCH --mem=128G
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00
#SBATCH --array=1-6

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a05_NERRS_variants"
output_dir="a06_NERRS_chr_gvcf"
mkdir -p $output_dir

# Load modules
spack load gatk

# Chromosomes to process
chromosomes=("Chr01" "Chr02" "Chr03" "Chr04" "Chr05" "Chr06")

# Get the chromosome for this array task
chr=${chromosomes[$SLURM_ARRAY_TASK_ID-1]}

# Disable file locking for TileDB
export TILEDB_DISABLE_FILE_LOCKING=1

# Function to run GenomicsDBImport for a given chromosome
run_genomicsdbimport() {
  local chr=$1
  local mem=$2
  gatk --java-options "-Xmx${mem}G" GenomicsDBImport \
    --genomicsdb-workspace-path ${output_dir}/genomicsDB_${chr} \
    --batch-size 50 \
    --intervals ${chr} \
    --sample-name-map NERRS_SampleList.txt \
    --reader-threads 5
}
```

Joint Genotyping
****

```
#!/bin/bash
#SBATCH --job-name=jointgenotyping
#SBATCH --mem=64G
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00
#SBATCH --array=1-6

# Load modules
spack load gatk

#### Joint genotyping on multiple samples stored in a GenomicsDB
workspace ####

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a06_NERRS_chr_gvcf"
output_dir="a07_NERRS_chr_raw_vcf"
mkdir -p $output_dir

# Disable file locking for TileDB
export TILEDB_DISABLE_FILE_LOCKING=1

# Reference genome
reference_genome="Zmarina_668_v2.0.fa"

# Chromosomes to process
chromosomes=("Chr01" "Chr02" "Chr03" "Chr04" "Chr05" "Chr06")

# Get the chromosome for this array task
chr=${chromosomes[$SLURM_ARRAY_TASK_ID-1]}

# Loop through each chromosome and run GenotypeGVCFs
gatk --java-options "-Xmx96G" GenotypeGVCFs \
  -R $reference_genome \
  -V gendb://$input_dir/genomicsDB_$chr \
  --intervals $chr \
  -O ${output_dir}/Zostera_${chr}_raw.vcf
```

Select Variants
****

```
#!/bin/bash
#SBATCH --mem=32G
#SBATCH --job-name=selectSNPs
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00
#SBATCH --array=1-6

# Load modules
spack load gatk

#### Filtering for SNPS only####

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a07_NERRS_chr_raw_vcf"
output_dir="a08_NERRS_selected_variants"
mkdir -p $output_dir

# Reference genome
reference_genome="Zmarina_668_v2.0.fa"

# Chromosomes to process
chromosomes=("Chr01" "Chr02" "Chr03" "Chr04" "Chr05" "Chr06")

# Get the chromosome for this array task
chr=${chromosomes[$SLURM_ARRAY_TASK_ID-1]}

# Select only SNPs from VCF files
  gatk --java-options "-Xmx96G" SelectVariants \
    -R $reference_genome \
    -V $input_dir/Zostera_${chr}_raw.vcf \
    --select-type-to-include SNP \
    -O ${output_dir}/Zostera_${chr}_snp.vcf
```

Mark Variants
****

```
#!/bin/bash
#SBATCH --mem=128G
#SBATCH --job-name=markvariants
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00
#SBATCH --array=1-6

# Load modules
spack load gatk

#### Marking SNPS #######

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a08_NERRS_selected_variants"
output_dir="a09_NERRS_marked_variants"
mkdir -p $output_dir

# Define the reference genome
reference_genome="Zmarina_668_v2.0.fa"

# Chromosomes to process
chromosomes=("Chr01" "Chr02" "Chr03" "Chr04" "Chr05" "Chr06")

# Get the chromosome for this array task
chr=${chromosomes[$SLURM_ARRAY_TASK_ID-1]}

# Next, mark variants by specific criteria
#parameters
#"MQ < 40.0"  = marks variants with a Mapping Quality (MQ) less than
40 (low MQ values indicate poor alignment quality, suggesting that
the variant call may be unreliable)
#"FS > 60.0"  = marks variants with a Fisher Strand (FS) value
greater than 60 (high FS values indicate strand bias, where the
variant is observed more frequently on one DNA strand than the other,
which can b>
#"QD < 7.0"  = marks variants with a Quality by Depth (QD) value less
than 7 (low QD values suggest that the variant quality score is low
relative to the depth of coverage, indicating potential false
positive>
#"DP > 12500.0" = marks variants with a Depth of Coverage (DP)
greater than 12,500 (high DP values may indicate duplicated regions
or sequencing artifacts)
```

```
#Schielbelhut et al. used "QD < 10.0"
# can decide what depth to filter at based on sequencing coverage:
#sequencing depth (use whatever file you want)

# Mark SNPs from VCF files
  gatk --java-options "-Xmx96G" VariantFiltration \
        -R $reference_genome \
        -V $input_dir/Zostera_${chr}_snp.vcf\
        -O  ${output_dir}/Zostera_${chr}_mk_snp.vcf \
        --filter-expression "MQ < 40.0" \
        --filter-name "MQ_filter" \
        --filter-expression "FS > 60.0" \
        --filter-name "FS_filter" \
        --filter-expression "QD < 2.0" \
        --filter-name "QD_filter" \
        --filter-expression "DP > 12500.0" \
        --filter-name "DP_large"
```

Filter Marked Variants
****

```
#!/bin/bash
#SBATCH --mem=32G
#SBATCH --job-name=filtermarked
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00
#SBATCH --array=1-6

# Load modules
spack load gatk

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a09_NERRS_marked_variants"
output_dir="a10_NERRS_filtered_marked_variants"
mkdir -p $output_dir

# Disable file locking for TileDB
export TILEDB_DISABLE_FILE_LOCKING=1

# Reference genome
reference_genome="Zmarina_668_v2.0.fa"

# Chromosomes to process
chromosomes=("Chr01" "Chr02" "Chr03" "Chr04" "Chr05" "Chr06")

# Get the chromosome for this array task
chr=${chromosomes[$SLURM_ARRAY_TASK_ID-1]}

# Run GATK SelectVariants to exclude filtered variants
    gatk --java-options "-Xmx96G" SelectVariants \
        -R $reference_genome \
        -V $input_dir/Zostera_${chr}_mk_snp.vcf \
        -O ${output_dir}/Zostera_${chr}_fmk_snp.vcf \
        --exclude-filtered true
```

Additional Filtering
****

```
#!/bin/bash
#SBATCH --mem=128G
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00
#SBATCH --array=1-6

# Load modules
spack load vcftools

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a10_NERRS_filtered_marked_varian
ts"
output_dir="a11_NERRS_filtered_vcftools"
mkdir -p $output_dir

# Chromosomes to process
chromosomes=("Chr01" "Chr02" "Chr03" "Chr04" "Chr05" "Chr06")

# Get the chromosome for this array task
chr=${chromosomes[$SLURM_ARRAY_TASK_ID-1]}

# Apply additional filtering criteria with vcftools
    vcftools --vcf $input_dir/Zostera_${chr}_fmk_snp.vcf \
        --minGQ 18 \
        --minDP 6 \
        --recode \
        --recode-INFO-all \
        --out ${output_dir}/Zostera186_${chr}_fmkvt_snp.vcf
```

Merge VCF files
****

```
#!/bin/bash
#SBATCH --mem=128G
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00

# Load modules
spack load vcftools

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a11_NERRS_filtered_vcftools"
output_dir="a12_NERRS_merged_vcf"
mkdir -p $output_dir

# Merge VCF files using vcftools
vcf-concat ${input_dir}/Zostera186_Chr01_fmkvt_snp.vcf.recode.vcf
${input_dir}/Zostera186_Chr02_fmkvt_snp.vcf.recode.vcf
${input_dir}/Zostera186_Chr03_fmkvt_snp.vcf.recode.vcf
${input_dir}/Zostera186_Chr04_fmkvt_snp.vcf.recode.vcf
${input_dir}/Zostera186_Chr05_fmkvt_snp.vcf.recode.vcf
${input_dir}/Zostera186_Chr06_fmkvt_snp.vcf.recode.vcf >
${output_dir}/Zostera_vcftoolsmerged_186.vcf
```

VCF Sequencing Quality
****

```
#!/bin/bash
#SBATCH --mem=64G
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00

# Load modules
spack load vcftools

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a12_NERRS_merged_vcf"
output_dir="a13_NERRS_vcf_quality"
mkdir -p $output_dir

# Get statistics on sequencing depth
vcftools --vcf $input_dir/Zostera_vcftoolsmerged_186.vcf --depth --
out ${output_dir}/depth_stats_Zostera_vcftoolsmerged_186

# Get statistics on missing data
vcftools --vcf $input_dir/Zostera_vcftoolsmerged_186.vcf --missing-
indv --out ${output_dir}/missing_data_Zostera_vcftoolsmerged_186
```

Finalize VCF
****

```
#!/bin/bash
#SBATCH --mem=16G
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00

# Load modules
spack load vcftools

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a12_NERRS_merged_vcf"
output_dir="a14_NERRS_final_vcf"
mkdir -p $output_dir
chown :hpc_kamels_lab $output_dir
chmod g+w $output_dir

vcftools --vcf ${input_dir}/Zostera_vcftoolsmerged_186.vcf \
        --remove NERRS_remove_low_quality_vcf.txt \
        --recode \
        --recode-INFO-all \
        --out ${output_dir}/Zostera_vcftoolsfinal_186
```

RClone
****

```
#!/bin/bash
#SBATCH --mem=128G
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00

# Load modules
spack load vcftools

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a14_NERRS_final_vcf"
output_dir="a15_NERRS_Rclone"
mkdir -p $output_dir

# Define the input VCF file
input_vcf="${input_dir}/Zostera_186_ALL.vcf"

# Define an array of population identifiers
populations=("NC1" "NC2" "NC3" "NC4" "NC5" "NC6" "NC7" "NC8" "NC9"
"NC10" "VA1" "VA2" "VA3" "VA4" "VA5" "VA6" "VA7" "VA8" "VA9" "VA10"
"NC1S" "NC4S" "NC5S" "NC8S" "NC9S" "NC10S" "VA1S" "VA2S" "VA3S"
"VA4S" "VA5S" "VA6S" "VA7S" "VA8S" "VA9S" "VA10S" "NCB" "NCT" "VAA"
"VAG" "VAP")

# Extract sample IDs from the VCF file
sample_ids=$(grep "^#CHROM" $input_vcf | cut -f10-)

# Convert sample IDs to an array
IFS=$'\t' read -r -a sample_ids_array <<< "$sample_ids"

# Generate keep files for each population
for pop in "${populations[@]}"; do
    keep_file="${output_dir}/keep_${pop}.txt"
    # Initialize the keep file
    > "$keep_file"
    # Extract sample IDs for the population and save to keep file
    for sample_id in "${sample_ids_array[@]}"; do
        if [[ $sample_id == ${pop}-* ]]; then
            echo "$sample_id" >> "$keep_file"
        fi
    done
done
```

```
# Loop through each population to remove variants with missing data
for pop in "${populations[@]}"; do
    keep_file="${output_dir}/keep_${pop}.txt"
    echo "Processing population: $pop"
    echo "Keep file: $keep_file"
    cat "$keep_file"
    if [ -s "$keep_file" ]; then
        # Extract individuals from the population of interest
        echo "Running vcftools for population: $pop"
        vcftools --vcf "$input_vcf" --keep "$keep_file" --recode --
recode-INFO-all --out
"${output_dir}/Zostera_vcftoolsfinal_renamed_${pop}"

        # Check if the intermediate VCF file was created
        if [ -f
"${output_dir}/Zostera_vcftoolsfinal_renamed_${pop}.recode.vcf" ];
then
            echo "Intermediate VCF file created for population: $pop"
            # Remove variants with any missing data
            vcftools --vcf
"${output_dir}/Zostera_vcftoolsfinal_renamed_${pop}.recode.vcf" --
max-missing 1.00 --recode --recode-INFO-all --out
"${output_dir}/Zostera_vcftoolsfinal_renamed_${pop}_filtered"
        else
            echo "Error: Intermediate VCF file not found for
population: $pop"
        fi
    else
        echo "Warning: File $keep_file is empty or not found."
    fi
done
```

Remove samples (i.e. clones)
****

```
#!/bin/bash
#SBATCH --job-name=remove_samples
#SBATCH --time=1-00:00:00  # 1 day
#SBATCH --mem=32G  # 32 GB of memory

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a14_NERRS_final_vcf"
output_dir="/scratch/lrs8888/workflow/a14_NERRS_final_vcf"

# Load module
spack load vcftools

# Process a VCF file with several individuals to remove
#vcftools --vcf ${input_dir}/Zostera_final_186_renamed.vcf --remove
NERRS_remove_clones.txt --recode --out
${output_dir}/Zostera_final_186_renamed_noclones
```

Filter Missing
****

```
#!/bin/bash
#SBATCH --job-name=filter_missing
#SBATCH --time=1-00:00:00  # 1 day
#SBATCH --mem=32G  # 32 GB of memory

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a14_NERRS_final_vcf"
output_dir="/scratch/lrs8888/workflow/a14_NEERS_final_vcf"

# Load module
spack load vcftools

# Process the first VCF file
vcftools  --vcf ${input_dir}/Zm_May_186_renamed_noclones.vcf --maf
0.01 --min-alleles 2 --max-alleles 2 --max-missing 0.50 --recode --
out ${output_dir}/Zm_May_186_renamed_noclones_filt

# Process the second VCF file
vcftools  --vcf ${input_dir}/Zm_Sept_186_renamed_noclones.vcf --maf
0.01 --min-alleles 2 --max-alleles 2 --max-missing 0.50 --recode --
out ${output_dir}/Zm_Sept_186_renamed_noclones_filt
```

Impute VCF
****

```
#!/bin/bash
#SBATCH --mem=128G
#SBATCH --job-name=beagle
#SBATCH --ntasks=1
#SBATCH -t 7-00:00:00

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a14_NERRS_final_vcf"
output_dir="a16_NERRS_imputed"
mkdir -p $output_dir

# Load modules
spack load vcftools
spack load htslib@1.19.1

# First compress the VCF file
bgzip -c ${input_dir}/Zm_final_186_renamed_noclones_filt.vcf >
${output_dir}/Zm_final_186_renamed_noclones_filt.vcf.gz

# Then index the compressed file
tabix -p vcf ${output_dir}/Zm_final_186_renamed_noclones_filt.vcf.gz

# Run the imputation as a job through beagle
java -Xmx16g -jar beagle.17Dec24.224.jar
gt=${output_dir}/Zm_final_186_renamed_noclones_filt.vcf.gz
out=${output_dir}/Zm_final_186_renamed_noclones_filt_imputed
```

Map Sample File Names
****

```
#!/bin/bash

# Directory containing GVCF files
input_dir="/scratch/lrs8888/workflow/a05_NERRS_variants"

# Output sample name map file
sample_map="NERRS_SampleList.txt"

# Create or clear the sample name map file
> $sample_map

# Loop through each GVCF file in the directory
for gvcf in $input_dir/*.g.vcf; do
  # Check if the file exists
  if [ -e "$gvcf" ]; then
    # Extract the sample name from the file name (assuming the sample
name is part of the file name)
    sample_name=$(basename "$gvcf" .g.vcf)
    # Add the sample name and file path to the sample name map file
    echo -e "$sample_name\t$gvcf" >> $sample_map
    echo "Added $sample_name to $sample_map"
  else
    echo "No files found matching pattern $gvcf_dir/*.g.vcf"
  fi
done
```

Calculate Read Depth
****

```
#!/bin/bash
#SBATCH --job-name=avgdepth
#SBATCH --output=depth.txt
#SBATCH --time=07:00:00
#SBATCH --mem=16G

# Load module
spack load vcftools

# Create or clear the output file
output_file="average_depth_fmk_snp_less_vcf.txt"
> "$output_file"

# Loop through all .g.vcf files in the current directory
for file in *.vcf; do
    # Extract the base name of the file (without the .g.vcf
extension)
    base_name=$(basename "$file" .vcf)

    # Run vcftools --depth and extract the required information
    vcftools --vcf "$file" --depth --out temp4_depth
    awk 'NR>1 {print $1, $2, $3}' temp4_depth.idepth >>
"$output_file"

done

# Clean up temporary files
rm temp4_depth.*
```

Calculate Missingness
****

```
#!/bin/bash
#SBATCH --job-name=missingness
#SBATCH --output=missingness.txt
#SBATCH --time=07:00:00
#SBATCH --mem=16G

# Load module
spack load vcftools

# Create or clear the output file
output_file="missingness_fmk_snp_less_vcf.txt"
> "$output_file"

# Loop through all .g.vcf files in the current directory
for file in *.vcf; do
    # Extract the base name of the file (without the .g.vcf
extension)
    base_name=$(basename "$file" .vcf)

    # Run vcftools --missing and extract the required information
    vcftools --vcf "$file" --missing-indv --out temp4_missing
    awk 'NR>1 {print $1, $2, $5}' temp4_missing.imiss >>
"$output_file"

done

# Clean up temporary files
rm temp4_missing.*
```

For Downstream Pop Gen Analyses
****

```
#!/bin/bash
#SBATCH --job-name=genomat
#SBATCH --ntasks=1
#SBATCH --time=11:00:00
#SBATCH --mem=128G

# Define the input and output directories
input_dir="/scratch/lrs8888/workflow/a16_NERRS_imputed"
output_dir="/scratch/lrs8888/workflow/a17_NERRS_pop_gen"
mkdir -p $output_dir

# Load modules
spack load vcftools

##Write file with pruned positions to use in pop structure analyses
vcftools --vcf
${input_dir}/Zm_May_186_renamed_noclones_filt_imputed.vcf --012 --out
${output_dir}/Zm_May_186_renamed_noclones_filt_imputed_genomat

###Read in genotype data - need to upload the "ld.pruned_XXX.012" to
the HPC made with vcftools using thinned positions from PCA script

vcftools --vcf
${input_dir}/Zm_May_186_renamed_noclones_filt_imputed.vcf --positions
${output_dir}/ld.pruned_Zm_May_186_renamed_noclones_filt_imputed_geno
mat.012.pos --012 --out
```