#1. Check MD5 sums

```
BASE_DIR=”dir_with_MD5_data”

cat $BASE_DIR/MD5* > MD5_all.txt
cd $BASE_DIR
bash
touch MD5_generated.txt
for file in `find . -name "*.fq.gz" -type f`;
do
  echo $file;
  md5sum $file >>MD5_generated.txt
done

sort MD5_all.txt | uniq > MD5_all_sort_uniq.txt
sort MD5_generated.txt | uniq | sed 's/.\///g' > MD5_generated_sort_uniq.txt
diff MD5_all_sort_uniq.txt MD5_generated_sort_uniq.txt
```

#2a. Check quality and filter out low-quality sequences and Illumina adapters (fastp)

```
BASE_DIR=”dir_with_raw_data”

cd $BASE_DIR
for s in AW17663_MKDN250001895-1A_22LMNGLT4_L1
AW17663_MKDN250001895-1A_22LMNGLT4_L2;
do
  echo ${s};
  singularity exec app/fastp_0.24.1.sif fastp --in1 AW_WGS_data/${s}_1.fq.gz --out1
fastp/${s}_1_fastp.fq.gz --in2 AW_WGS_data/${s}_2.fq.gz --out2
fastp/${s}_2_fastp.fq.gz --failed_out fastp/${s}_failed_fastp.fq.gz
--detect_adapter_for_pe --trim_front1 10 --trim_tail1 0 --trim_front2 10 --trim_tail2 0
--cut_front --cut_right --cut_window_size 4 --cut_mean_quality 20 --average_qual 30
--trim_poly_g --html fastp/${s}_report.html;
done
```

#2b. Remove low-quality files based on fastp reports

The following files were not processed further due to low data quality, based on fastp reports after filtering:

```
AW201821M_MKDN250001953-1A_22LK5CLT4_L5_1_fastp.fq.gz
AW201821M_MKDN250001953-1A_22LK5CLT4_L5_2_fastp.fq.gz
AW56724_MKDN250001940-1A_22YH7WLT4_L3_1_fastp.fq.gz
AW56724_MKDN250001940-1A_22YH7WLT4_L3_2_fastp.fq.gz
```

#3. Reference genome masking with Earl Grey

```
# Allow a week for the Earl Grey run!

# === Define paths ===
GENOME="path_to_reference_genome"
SPECIES="acrola"
OUTDIR="path_to_output_dir"

# === Run Earl Grey ===
earlGrey \
  -g "$GENOME" \
  -s "$SPECIES" \
  -o "$OUTDIR"

# Hard-mask the reference genome

singularity shell --bind $(pwd):/path_to_container_dir/ bedtools_2.31.1.sif

#remove the ‘factory’ soft-masking (lowercase letters in the genome):
awk '/^>/ {print; next} {print toupper($0)}' /mnt/GCA_965287075.1_bAcrPal2.2.fna >
/mnt/GCA_965287075.1_bAcrPal2.2.upper.fna

#mask:
bedtools maskfasta -fi /mnt/GCA_965287075.1_bAcrPal2.2.upper.fna -bed
/mnt/AW_masked_genome_earlgrey/acrola_summaryFiles/acrola.filteredRepeats.bed
-fo /mnt/genome.masked.fa
```

#4a. Mapping to masked reference genome - BWA-MEM - separately for each lane

```
BASE_DIR=”dir_with_reference_genome”

cd $BASE_DIR
for sample in AW17663_MKDN250001895-1A_22LMNGLT4_L1
AW17663_MKDN250001895-1A_22LMNGLT4_L2;
```

```
do
  echo ${sample};
  singularity exec app/bwa_0.7.19.sif bwa mem -t 20 ref/genome.masked.fa
fastp/${sample}_1_fastp.fq.gz fastp/${sample}_2_fastp.fq.gz > bwa/${sample}.sam;
  singularity exec app/samtools_1.21.sif samtools view -b -S bwa/${sample}.sam >
bwa/${sample}.bam;
  singularity exec app/samtools_1.21.sif samtools sort bwa/${sample}.bam -o
bwa/${sample}_sorted.bam;
  singularity exec app/samtools_1.21.sif samtools index bwa/${sample}_sorted.bam;
done
```

#4b. Add read groups and samples to resultant bam files by picard AddOrReplaceReadGroups

```
singularity exec app/picard_3.4.0.sif java -Xmx10g -jar /usr/picard/picard.jar
AddOrReplaceReadGroups
I=bwa/AW17663_MKDN250001895-1A_22LMNGLT4_L1_sorted.bam
O=bwa/AW17663_MKDN250001895-1A_22LMNGLT4_L1_sorted_rg.bam RGLB=lib
RGPL=ILLUMINA RGID=L1 RGSM=AW17663
RGPU=MKDN250001895-1A_22LMNGLT4_L1
singularity exec app/picard_3.4.0.sif java -Xmx10g -jar /usr/picard/picard.jar
AddOrReplaceReadGroups
I=bwa/AW17663_MKDN250001895-1A_22LMNGLT4_L2_sorted.bam
O=bwa/AW17663_MKDN250001895-1A_22LMNGLT4_L2_sorted_rg.bam RGLB=lib
RGPL=ILLUMINA RGID=L2 RGSM=AW17663
RGPU=MKDN250001895-1A_22LMNGLT4_L2
```

#4c. Merge bam files within sample

```
BASE_DIR=”path_to_base_dir”

cd $BASE_DIR
for sample in AW17663;
do
  echo ${sample};
  singularity exec app/samtools_1.21.sif samtools merge bwa_merge/${sample}.bam
bwa/${sample}*_sorted_rg.bam;
  singularity exec app/samtools_1.21.sif samtools index bwa_merge/${sample}.bam;
  singularity exec app/samtools_1.21.sif samtools sort bwa_merge/${sample}.bam -o
bwa_merge/${sample}_sorted.bam;
```

```
  singularity exec app/samtools_1.21.sif samtools index
bwa_merge/${sample}_sorted.bam;
done;
```

#5. Qualimap

```
BASE_DIR=”path_to_base_dir”
cd  $BASE_DIR
for sample in AW17663;
do
  echo ${sample};
  singularity exec app/qualimap_2.3.sif qualimap bamqc -bam
bwa_merge/${sample}_sorted.bam -nt 10 --java-mem-size=20G -outdir
qualimap/${sample} -outfile ${sample}.pdf -outformat PDF;
done
```

#6. Remove PCR duplicates - Picard

```
BASE_DIR=”path_to_base_dir”
cd  $BASE_DIR
for sample in AW17663
do
  echo ${sample};
  singularity exec app/picard_3.4.0.sif java -Xmx10g -jar /usr/picard/picard.jar
MarkDuplicates VALIDATION_STRINGENCY=LENIENT
INPUT=bwa_merge/${sample}_sorted.bam
OUTPUT=picard/${sample}_sorted_picard.bam
METRICS_FILE=picard/${sample}_picard_markduplicates_metrics.txt
REMOVE_DUPLICATES=true ASSUME_SORTED=true CREATE_INDEX=true;
done
```

#7. Qualimap one more time - after picard

```
BASE_DIR=”path_to_base_dir”
cd  $BASE_DIR
for sample in AW17663;
do
  echo ${sample};
  singularity exec app/qualimap_2.3.sif qualimap bamqc -bam
picard/${sample}_sorted_picard.bam -nt 10 --java-mem-size=20G -outdir
qualimap_after_picard/${sample} -outfile ${sample}.pdf -outformat PDF;
done
```

#8a. Variant calling - GATK4 HaplotypeCaller for each sample

```
BASE_DIR=”path_to_base_dir”
cd  $BASE_DIR/ref

singularity exec ../app/samtools_1.21.sif samtools faidx genome.masked.fa
singularity exec ../app/gatk_4.6.2.0.sif gatk CreateSequenceDictionary -R
genome.masked.fa

cd  $BASE_DIR/picard
for sample in AW17663
do
  echo ${sample};
  singularity exec app/gatk_4.6.2.0.sif gatk HaplotypeCaller -R ref/genome.masked.fa -I
picard/${sample}_sorted_picard.bam -O gatk/${sample}_haplotypecaller.vcf.gz -ERC
GVCF
done;
```

#8b. Variant calling - GATK4 CombineGVCFs

```
BASE_DIR=”path_to_base_dir”
cd  $BASE_DIR
singularity exec app/gatk_4.6.2.0.sif gatk CombineGVCFs -R ref/genome.masked.fa \
--variant gatk/AW17663_haplotypecaller.vcf.gz \
# ... list the remaining samples ...
--variant gatk/AW56724_haplotypecaller.vcf.gz \
-O gatk/combined.vcf.gz
```

#8c. Variant calling - GATK4 GenotypeGVCFs

```
BASE_DIR=”path_to_base_dir”
cd  $BASE_DIR
singularity exec app/gatk_4.6.2.0.sif gatk --java-options "-Xmx40g" GenotypeGVCFs -R
ref/genome.masked.fa -V gatk/combined.vcf.gz -O gatk/combined_genotyped.vcf.gz
```

#9a. Variant filtering - mark variants to be filtered out with GATK

```
# === Define paths ===
BASE_DIR="base_directory"
SIF="path_to_gatk_container"

# === Run filtering with gatk ===
singularity exec \
--bind ${BASE_DIR}:/mnt \
${SIF} \
gatk VariantFiltration \
-R /mnt/ref/genome.masked.fa \
-V /mnt/gatk_additional_sequences/conbined_genotyped.vcf.gz \
-O
/mnt/filtering_additional_sequences/QD2_FS60_SOR3_MQ40_MQRankSum_ReadPos
RankSum.vcf.gz \
                --filter-expression "QD < 2.0" --filter-name "QD2" \
                --filter-expression "FS > 60.0" --filter-name "FS60" \
                --filter-expression "SOR > 3.0" --filter-name "SOR3" \
                --filter-expression "MQ < 40.0" --filter-name "MQ40"
```

#9b. Variant filtering - remove the marked variants with bcftools

```
# === Define paths ===
BASE_DIR="base_directory"
SIF="path_to_bcftools_container"

# === Run filtering with bcftools ===
singularity exec \
       --bind ${BASE_DIR}:/mnt \
       ${SIF} \
         bcftools view \
         -f PASS \
         -Oz -o
/mnt/filtering_additional_sequences/filtered_QD2_FS60_SOR3_MQ40.vcf.gz \
         /mnt/filtering_additional_sequences/QD2_FS60_SOR3_MQ40.vcf.gz

# === Index the filtered VCF ===
singularity exec \
   --bind ${BASE_DIR}:/mnt \
   ${SIF} \
   bcftools index
/mnt/filtering_additional_sequences/filtered_QD2_FS60_SOR3_MQ40.vcf.gz
```

#9.c. Variant filtering continued: remove SNPs located near indels, keep only biallelic SNPs, remove monomorphic SNPs, remove SNPs with depth <3 and genotype quality <20; remove SNPs with mean depth <10 and above 2x mean depth

```
# === Define paths ===
BASE_DIR="base_directory”
SIF="path_to_container_with_bcftools"
SIF_VCFTOOLS="path_to_container_with_vcftools"

# === Run filtering with bcftools ===
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} bash -c "bcftools view -v snps
/mnt/filtered_QD2_FS60_SOR3_MQ40.vcf.gz -Oz -o
/mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels.vcf.gz && bcftools view -m2 -M2 -v
snps /mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels.vcf.gz -Oz -o
/mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs.vcf.gz && bcftools view
-c1 /mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs.vcf.gz -Oz -o
/mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs.vcf.g
z && bcftools +setGT
/mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs.vcf.g
z -- -t q -n . -i 'FMT/DP<3 || FMT/GQ<20' | \
bcftools view -Oz -o
/mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs_DP
3_GQ20.vcf.gz"

# === Calculate mean site depth with VCFtools ===
singularity exec --bind ${BASE_DIR}:/mnt ${SIF_VCFTOOLS} \
      vcftools --gzvcf
/mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs_DP
3_GQ20.vcf.gz --site-mean-depth --out
/mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs_DP
3_GQ20

# === Filter by mean site depth ===
singularity exec --bind ${BASE_DIR}:/mnt ${SIF_VCFTOOLS} \
         vcftools --gzvcf
/mnt/filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs_DP
```

3_GQ20.vcf.gz --out
filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs_DP3_GQ
20_meanDPfilter --min-meanDP 10 --max-meanDP 38.2282 --recode

#9.d. Variant filtering continued - MAF filter, SNP and individual missingness filter, LD pruning, removing related individuals, removing variants that mapped to unplaced scaffolds and sex chromosomes

# === Define paths ===
BASE_DIR="directory_with_vcf_file"
SIF="path_to_container_with_PLINK"
INPUT_VCF="input_vcf_filename"
OUT_BASE="/mnt/basename"

# === Run filtering with plink ===
# Convert VCF to PLINK format
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --vcf ${INPUT_VCF}
--make-bed --out ${OUT_BASE} --allow-extra-chr

# Filter by MAF ≥ 0.05
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile ${OUT_BASE} --maf
0.05 --make-bed --out ${OUT_BASE}_maf05 --allow-extra-chr

# Filter by SNP missingness ≤ 0.1
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile ${OUT_BASE}_maf05
--geno 0.1 --make-bed --out ${OUT_BASE}_maf05_SNPmiss0.1 --allow-extra-chr

# Filter by individual missingness ≤ 0.1
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile
${OUT_BASE}_maf05_SNPmiss0.1 --mind 0.1 --make-bed --out
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1 --allow-extra-chr

# Fix SNP IDs, which are missing (otherwise LD pruning fails)
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1 --allow-extra-chr --set-missing-var-ids
@_# --make-bed --out ${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids

# LD pruning (50kb windows, step 10 SNPs, r^2 threshold 0.2) - not done for
GONE/CurrentNe

```
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids --indep-pairwise 50 10
0.2 --out ${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids --allow-extra-chr

# Extract pruned SNPs to create final dataset
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids --extract
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids.prune.in \
--make-bed --out
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids_LDpruned
--allow-extra-chr

# Generate a pairwise relatedness file and remove highly related individuals
(pi-hat>0.125)
singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids_LDpruned --genome
--out relatedness --allow-extra-chr

singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids_LDpruned --rel-cutoff
0.125 --allow-extra-chr \
--make-bed --out
${OUT_BASE}_maf05_SNPmiss0.1_indMiss0.1_fixedSNPids_LDpruned_unrelated

#Extract unique chromosome names and then select those that are sex chromosomes
and unplaced scaffolds
cut -f1
filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs_DP3_GQ
20_meanDPfilter_SNPmiss0.1_indMiss0.1_fixedSNPids.bim | sort | uniq >
chr_names.txt

#Extract snps from unwanted chromosomes
awk '($1 == "CBDIYS010000007.1" || \
    $1 == "CBDIYS010000009.1" || \
    $1 == "CBDIYS010000014.1" || \
    $1 == "CBDIYS010000015.1" || \
    $1 == "CBDIYS010000028.1" || \
    $1 == "CBDIYS010000033.1" || \
    $1 == "CBDIYS010000045.1" || \
    $1 == "CBDIYS010000049.1" || \
	$1 == "CBDIYS010000056.1" || \
	$1 == "CBDIYS010000057.1" || \
	$1 == "CBDIYS010000060.1" || \
	$1 == "OZ261405.1" || \
	$1 == "OZ261420.1") {print $2}' \
filtered_QD2_FS60_SOR3_MQ40_noIndels_biallelicSNPs_polymorphicSNPs_DP3_GQ
20_meanDPfilter_SNPmiss0.1_indMiss0.1_fixedSNPids.bim > snps_to_exclude.txt

# Exclude these snps
	singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink --bfile
${OUT_BASE}_SNPmiss0.1_indMiss0.1_fixedSNPids \
	--exclude /mnt/snps_to_exclude.txt \
	--allow-extra-chr \
	--make-bed --out
${OUT_BASE}_SNPmiss0.1_indMiss0.1_fixedSNPids_noSexChrnoScaff
```

10a. Population analyses - general diversity parameters

```
# === Define paths ===
BASE_DIR="path_to_base_directory"
SIF="path_to_plink_container"
SIF_VCFTOOLS="path_to_VCFtools_container"
INPUT="basename_of_input_file"

# Calculate observed & expected heterozygosity per population
# if working on renamed chromosomes, use --chr-set 50
cd ${BASE_DIR}
for pop in B18 B97 P99; do
	singularity exec \
	--bind ${BASE_DIR}:/mnt \
	${SIF} /opt/plink \
	--bfile /mnt/filtering_additional_sequences/${INPUT}_${pop} \
	--het \
	--out
/mnt/population_analysis/0_Diversity_stats/additional_sequences/${INPUT}_${pop} \
	--allow-extra-chr
done

# Calculate per-SNP pairwise Fst between B18&B97; and between B97&P99
```

```
cd ${BASE_DIR}
for pair in B97_P99 B97_B18; do
      singularity exec \
      --bind ${BASE_DIR}:/mnt \
      ${SIF} /opt/plink \
      --bfile /mnt/filtering_additional_sequences/${INPUT} \
        --fst \
        --within /mnt/population_analysis/0_Diversity_stats/populations_${pair}.txt \
      --out
/mnt/population_analysis/0_Diversity_stats/additional_sequences/${INPUT}_${pair} \
      --allow-extra-chr
done

# Calculate inbreeding coefficients per sample
cd ${BASE_DIR}
        singularity exec \
      --bind ${BASE_DIR}:/mnt \
      ${SIF} /opt/plink \
      --bfile /mnt/filtering_additional_sequences/${INPUT} \
      --ibc \
      --out /mnt/population_analysis/0_Diversity_stats/additional_sequences/${INPUT} \
      --allow-extra-chr

# === Convert the PLINK format file to .vcf.gz ===
for pop in B18 B97 P99; do
        singularity exec \
        --bind ${BASE_DIR}:/mnt \
        ${SIF_PLINK} \
        /opt/plink \
        --bfile /mnt/filtering_additional_sequences/${INPUT}_${pop} \
        --recode vcf bgz \
        --out /mnt/filtering_additional_sequences/${INPUT}_${pop} \
        --allow-extra-chr
done

# === Calculate pi in overlapping windows ===
for pop in B18 B97 P99; do
        singularity exec \
        --bind ${BASE_DIR}:/mnt \
      ${SIF_VCFTOOLS} \
        vcftools \
        --gzvcf /mnt/filtering_additional_sequences/${INPUT}_${pop}.vcf.gz \
        --window-pi 50000 --window-pi-step 25000 \
        --out
/mnt/population_analysis/0_Diversity_stats/additional_sequences/${INPUT}_${pop}
done

# === Calculate per site pi ===
for pop in B18 B97 P99; do
      singularity exec \
      --bind ${BASE_DIR}:/mnt \
      ${SIF_VCFTOOLS} \
      vcftools \
      --gzvcf /mnt/filtering_additional_sequences/${INPUT}_${pop}.vcf.gz \
      --site-pi \
      --out
/mnt/population_analysis/0_Diversity_stats/additional_sequences/${INPUT}_${pop}
done
```

#10b. Population analyses - runs of homozygosity

```
# === Define paths ===
BASE_DIR="path_to_base_directory"
SIF="path_to_plink_container"
INPUT="basename_of_input_file"

# Calculate ROHs - use homozyg-kb 1000 and 250 for ROHs >1Mb and >250 kb,
respectively; use an input file that has not been filtered for MAF

singularity exec --bind ${BASE_DIR}:/mnt ${SIF} /opt/plink \
        --bfile /mnt/filtering_additional_sequences/${INPUT} \
        --allow-extra-chr \
        --homozyg \
        --homozyg-snp 50 \
        --homozyg-kb 1000 \
        --homozyg-density 50 \
      --homozyg-gap 200 \
      --homozyg-window-snp 50 \
      --homozyg-window-het 1 \
      --homozyg-window-missing 5 \
        --out
/mnt/population_analysis/2_ROHs/ROH_1Mb_gap200_additional_sequences_noMafFilt
er_unrelated
```

#10c. Population analyses - effective population size (Ne) - GONE, CurrentNe

#Ne short-term trajectory calculated with GONE2 - example for one spatio-temporal population

```
# === Define paths ===
BASE_DIR="path_to_base_dir"

# === Run GONE ===
# I tried to limit the number of SNPs (option -s) as the programme doesn't accept more
than 2000000, but this option does not work! I thinned the SNPs with PLINK instead to
1900000 SNPs
# I used an input file without the MAF filter

$BASE_DIR/GONE2/GONE2/gone2 \
-r 1 \
-o
$BASE_DIR/2_Biebrza18/GONE2_additional_sequences/filtered_B18_LargeChroms_t
hinned_SINGLE_POPULATION \
$BASE_DIR/2_Biebrza18/GONE2_additional_sequences/filtered_B18_LargeChroms_t
hinned.ped
```

#Ne short-term trajectory calculated with CurrentNe2 - example for one spatio-temporal population

```
# === Define paths ===
BASE_DIR="path_to_base_dir"
INPUT="$BASE_DIR/input_basename.ped"

# === Run CurrentNe ===
# metapopulation (e.g. for two populations)
$BASE_DIR/5_CurrentNe/currentNe/currentNe2/currentne2 \
-r 1 \
-x \
-o
$BASE_DIR/5_CurrentNe/currentNe_B18_METAPOPULATION_additional_sequences \
$INPUT

# single population
$BASE_DIR/5_CurrentNe/currentNe/currentNe2/currentne2 \
-r 1 \
-o
$BASE_DIR/5_CurrentNe/currentNe_B18_SINGLE_POPULATION_additional_sequenc
es \
$INPUT
```

#10d. Population analyses - genetic structure (PCA & ADMIXTURE)

```
# === Define paths ===
BASE_DIR="path_to_base_dir"
INPUT="bim_file_basename"
SIF=”path_to_plink_container”
SIF_ADMIXTURE="path_to_admixture_container"


# ==== Run PCA with PLINK ====
singularity exec --bind $BASE_DIR:/mnt $SIF /opt/plink --bfile $INPUT --allow-extra-chr
--pca 10 --out /mnt/population_analysis/PCA/$INPUT

# === Run ADMIXTURE with several K values and cross-validation with 10 folds ===
cd ${BASE_DIR}/population_analysis/4_ADMIXTURE
for K in 1 2 3 4 5 6; do
        singularity exec --bind ${BASE_DIR}:/mnt \
            ${SIF} \
            admixture /mnt/population_analysis/4_ADMIXTURE/${INPUT}.bed ${K} \
            -j8 -cv=10
done
```

#10e. Population analyses - adaptive variation (Fst outlier analysis with pcadapt and outFLANK in R)

```
#### Wth pcadapt ###

library(pcadapt)
library(qvalue)
```

```
# Load the .bed file with two populations: Biebrza 97 and W Pomerania 99, the
corresponding bim and fam files need to be in the same dir
x <- read.pcadapt(input = “filename.bed", type = "bed")

# run PCA and selection scan
pcadapt_res <- pcadapt(input = x)

#get p-values and adjust them
pvals <- pcadapt_res$pvalues
qvals <- qvalue(pvals)$qvalues

#extract outliers
outliers <- which(qvals < 0.05)  # FDR threshold of 5%
length(outliers)


#see which SNP IDs are outliers
snp_ids <- read.table("filename.bim")[, 2]  # column 2 = SNP names
outlier_snps <- snp_ids[outliers]
head(outlier_snps)

#draw a Manhattan plot with the outlier SPNs and with chromosome grouping
bim <- fread("filename.bim", header = FALSE)
colnames(bim) <- c("CHR", "SNP", "CM", "BP", "A1", "A2")

# create a dataframe with the .bim file columns and outlier SNPs, pvals, qvals and
-log10P
manhattan_df <- bim %>% mutate(P = pvals,
        Q = qvals,
        LOG10P = -log10(P),
        OUTLIER = Q < 0.05)  # Mark significant SNPs

# order by chromosome and base pair
manhattan_df <- manhattan_df %>%
  arrange(CHR, BP)

# create a cumulative base pair position
manhattan_df <- manhattan_df %>%
  group_by(CHR) %>%
```

```r
  mutate(BP_cum = BP + min(BP) - first(BP)) %>%
  ungroup()
head(manhattan_df)
tail(manhattan_df)

# create cumulative index for x-axis
chr_lengths <- manhattan_df %>% group_by(CHR) %>%
  summarize(chr_len = max(BP_cum)) %>%
  mutate(tot = cumsum(chr_len) - chr_len)

manhattan_df <- manhattan_df %>%
  left_join(chr_lengths, by = "CHR") %>%
  mutate(BP_plot = BP_cum + tot)

manhattan_df <- manhattan_df %>%
  filter(!is.na(P), P > 0) %>%   # remove NA and 0 p-values
  mutate(LOG10P = -log10(P),
         OUTLIER = Q < 0.05)

# add chromosome center for x-axis labels
axis_df <- manhattan_df %>%
  group_by(CHR) %>%
  summarize(center = (min(BP_plot) + max(BP_plot)) / 2)

# plot the Manhattan plot with ggplot2
ggplot(manhattan_df, aes(x = BP_plot, y = LOG10P)) +
  geom_point(aes(color = as.factor(CHR)), alpha = 0.6, size = 0.8) +
  scale_color_manual(values = rep(c("steelblue", "gray40"), 22)) +
  scale_x_continuous(label = axis_df$CHR, breaks = axis_df$center) +
  labs(x = "Chromosome", y = expression(-log[10](p))) +
  geom_hline(yintercept = -log10(0.05), color = "red", linetype = "dashed") +
  geom_point(data = filter(manhattan_df, OUTLIER),
             aes(x = BP_plot, y = LOG10P), color = "red", size = 1.2) +
  theme_bw() +
  theme(legend.position = "none",
        panel.border = element_blank(),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        axis.text = element_text(size=20),
        axis.title = element_text(size=24))
```

```
ggsave("filename.png", width = 35, height = 10, dpi = 300)


#### With outFLANK ###
# outFLANK installation
library(devtools)
BiocManager::install("qvalue")
library(qvalue)
install_github("whitlock/OutFLANK")
library(OutFLANK)

# Load other libraries
library(ggplot2)
library(dplyr)
library(vcfR) #reading VCF files
library(RColorBrewer) #custom colours
library(dartR) #to use outFLANK with a genind/genlight object

# run outFLANK analysis
#1. Read .vcf file
# use a .vcf wth changed SNP names, because e.g. "OZ261402.1_146976" is not
parsed correctly by outFLANK
vcf <- read.vcfR(file = "file.vcf")

#2.Convert to a genlight object, add population info and subset
gen <- vcfR2genlight(vcf)

popIDs <- c(rep("b97",22),rep("b18",17),rep("p99",16))
pop(gen)<-popIDs

gen1<-gen[pop(gen)!="b18"] #only p99 and b97

#3. Run outFLANK

outFst<-gl.outflank(
  gen1,
  plot = TRUE,
  LeftTrimFraction = 0.05,
  RightTrimFraction = 0.05,
  Hmin = 0.1,
  qthreshold = 0.05)

# View summary
resWGS<-outFst$outflank$results
min(resWGS$qvalues, na.rm=T)
min(resWGS$pvalues, na.rm=T)

# Extract outlier loci
outliers <- resWGS[resWGS$OutlierFlag == TRUE, 1]
outliers<-outliers[is.na(outliers)==FALSE]
outliers
```