Milkweed genomics project: Main steps

- install axe v 0.3.3

Demultiplex

- Scripts here (step01_demultiplex)

Plates_1_2

```
module load gcc gsl

axe-demux -m0 -z 2 -c -2 -v -t plates1_2_demult_summary.txt -b PstI-188PlexKey.txt -f
↪  40559-1.2_R1.fastq.gz -r 40559-1.2_R2.fastq.gz -F
↪  demult_output_plates1_2/40559-1.2_R1 -R demult_output_plates1_2/40559-1.2_R2
```

Plate 3- run 1

```
module load gcc gsl

axe-demux -m0 -z 2 -c -2 -v -t plate3_run1_demult_summary_15hr.txt -b PstI-Plate1Key.txt
↪  -f 40559-3_R1.fastq.gz -r 40559-3_R2.fastq.gz -F
↪  demult_output_plate3_run1_15hr/40559-3_R1 -R
↪  demult_output_plate3_run1_15hr/40559-3_R2
```

Plate 3- run 2

```
module load gcc gsl

axe-demux -m0 -z 2 -c -2 -v -t plate3_run2_demult_summary_15hr.txt -b PstI-Plate1Key.txt
↪  -f 40559-3A_R1.fastq.gz -r 40559-3A_R2.fastq.gz -F
↪  demult_output_plate3_run2_15hr/40559-3A_R1 -R
↪  demult_output_plate3_run2_15hr/40559-3A_R2
```

Trim adaptors

- Install
  - Cutadapt v 4.1
  - TrimGalore-0.6.6
  - multiqc v 1.13.dev0
- Resources:
  - https://github.com/sbreitbart/GBS-PreProcess
  - Rob Elshire / gbs pipeline scripts · GitLab
- Scripts, output, and keyfiles here (step02_trimadapters)
- Made keyfiles for each plate
  - Structure:

```
sampleA_1.fq barcode1 barcode2
sampleB_1.fq barcode1 barcode2
(...)
```

- Downloaded batch_trim.pl script from https://github.com/sbreitbart/GBS-PreProcess, then put it and keyfile into same folder as samples
- The part that I edited in this perl script:

```
system("~/TrimGalore-0.6.6/trim_galore -q 0 --dont_gzip -e 0.1 --fastqc --paired -o
↪  trim_galore/ -a2 $x[2] -a $x[3] $x[0] $x[1]");

# -q refers to quality. 0 means no cutoff (NOTE: I did not notice that I hadn't changed
↪  the quality (Phred) score until later. So, this step was useful for trimming adapters
↪  but not for quality filtering. That comes later.)
# -e refers to Maximum error rate (default: 0.1)
```

- Ran scripts and generated MultiQC reports

MultiQC reports

Plates 1-2, run 1:

- Removed empty wells

```
# Run 1
mv *40559-1.2_R1_1A12*  *40559-1.2_R1_1E12*  *40559-1.2_R1_1G12* *40559-1.2_R1_1B12*
↪  *40559-1.2_R1_1F11*  *40559-1.2_R1_1G3* *40559-1.2_R1_1C12*  *40559-1.2_R1_1F12*
↪  *40559-1.2_R1_1H11* *40559-1.2_R1_1D12*  *40559-1.2_R1_1G11*  *40559-1.2_R1_1H12*
↪  blank_wells

# Run 2
mv *40559-1.2_R2_1A12*  *40559-1.2_R2_1E12*  *40559-1.2_R2_1G12* *40559-1.2_R2_1B12*
↪  *40559-1.2_R2_1F11*  *40559-1.2_R2_1G3* *40559-1.2_R2_1C12*  *40559-1.2_R2_1F12*
↪  *40559-1.2_R2_1H11* *40559-1.2_R2_1D12*  *40559-1.2_R2_1G11*  *40559-1.2_R2_1H12*
↪  blank_wells
```

Plates 1-2, run 2:

- Removed empty wells

```
# RUN 1
mv *40559-1.2_R1_2C10*  *40559-1.2_R1_2G11*  *40559-1.2_R1_2H11* *40559-1.2_R1_2A12*
↪  *40559-1.2_R1_2B12*  *40559-1.2_R1_2C12* *40559-1.2_R1_2D12*  *40559-1.2_R1_2E12*
↪  *40559-1.2_R1_2F12* *40559-1.2_R1_2G12*  *40559-1.2_R1_2H12*  blank_wells

# Run 2
mv *40559-1.2_R2_2C10*  *40559-1.2_R2_2G11*  *40559-1.2_R2_2H11* *40559-1.2_R2_2A12*
↪  *40559-1.2_R2_2B12*  *40559-1.2_R2_2C12* *40559-1.2_R2_2D12*  *40559-1.2_R2_2E12*
↪  *40559-1.2_R2_2F12* *40559-1.2_R2_2G12*  *40559-1.2_R2_2H12*  blank_wells
```

Plate 3, Run 1:

- Removed empty wells

```
# Run 1
mv *40559-3_R1_H4*  *40559-3_R1_H12*   blank_wells
```

```
# Run 2 (just changed R1 to R2)
mv *40559-3_R2_H4*  *40559-3_R2_H12*   blank_wells
```

Plate 3, Run 2:

- Removed empty wells

```
# Run 1
mv *40559-3A_R1_H4*  *40559-3A_R1_H12*   blank_wells

# Run 2 (just changed R1 to R2)
mv *40559-3A_R2_H4*  *40559-3A_R2_H12*   blank_wells
```

Re-clean: trim to 110 bp and phred score of 20 with fastp

- Scripts and output log files here (step03_clean)
- Installed fastp v 0.23.1

Plates 1-2

```
module load fastp/0.23.1

while read ident; do
  fastp -i 40559-1.2_R1_${ident}_R1_val_1.fq -I 40559-1.2_R2_${ident}_R2_val_2.fq -q 20
  ↪  -A --length_limit 110 -o ../../step03_clean/plates1_2/1.2_R1_${ident}.fq -O
  ↪  ../../step03_clean/plates1_2/1.2_R2_${ident}.fq
done <unique_ident_fileName.txt
```

Plate 3, run 1

```
module load fastp/0.23.1

while read ident; do
  fastp -i 40559-3_R1_${ident}_R1_val_1.fq -I 40559-3_R2_${ident}_R2_val_2.fq -q 20 -A
  ↪  --length_limit 110 -o ../../step03_clean/plate3_r1/3_R1_${ident}.fq -O
  ↪  ../../step03_clean/plate3_r1/3_R2_${ident}.fq
done <unique_ident_fileName.txt
```

Plate 3, run 2

```
module load fastp/0.23.1

while read ident; do
  fastp -i 40559-3A_R1_${ident}_R1_val_1.fq -I 40559-3A_R2_${ident}_R2_val_2.fq -q 20 -A
  ↪  --length_limit 110 -o ../../step03_clean/plate3_r2/3A_R1_${ident}.fq -O
  ↪  ../../step03_clean/plate3_r2/3A_R2_${ident}.fq
done <unique_ident_fileName.txt
```

Index genome with BWA

- Install bwa v 0.7.17
- Upload genome
  - Saved it as "genome_RAW", then saved copy as "genome_Asyriaca"
- Index genome

```
module load bwa/0.7.17
bwa index ~/scratch/genome_Asyriaca/Asclepias_syriaca_v1.0_chromosomes.fa
```

Align sequences, convert to .bam, and QA/QC files

- Scripts here (step04_align)

Align sequences to reference genome

- Use BWA-mem to map sequences to ref genome

Plates 1-2

- Get filename text file

```
ls *.fq > whole_filenames.txt #put the names of all the files into a text file (do this
↪  while in the plates1_2 folder, for example)
awk -F_  '{print $3}' whole_filenames.txt | sort | uniq > unique_ident_fileName.txt #use
↪  awk to get the identifier out and store into a file (the identifier = the plate well.
↪  E.g. A11, C8)
```

- Script

```
module load bwa/0.7.17

while read ident; do # indent is the argument. It's each line in the .txt file
  bwa mem ~/scratch/genome_Asyriaca/Asclepias_syriaca_v1.0_chromosomes.fa
  ↪  1.2_R1_${ident}.fq 1.2_R2_${ident}.fq >
  ↪  ../../step04_align/plates1_2/1.2_${ident}.sam
done <unique_ident_fileName.txt # this is the input file
```

Plate 3, run 1

- Get filename text file

```
ls *.fq > whole_filenames.txt
awk -F_  '{print $3}' whole_filenames.txt | cut -d"." -f1 | sort | uniq >
↪  unique_ident_fileName.txt
```

- Script

```
module load bwa/0.7.17

while read ident; do
  bwa mem ~/scratch/genome_Asyriaca/Asclepias_syriaca_v1.0_chromosomes.fa
  ↪  3_R1_${ident}.fq 3_R2_${ident}.fq > ../../step04_align/plate3_r1/3_${ident}.sam
done <unique_ident_fileName.txt
```

Plate 3, run 2

- Get filename text file

```
ls *.fq > whole_filenames.txt
awk -F_  '{print $3}' whole_filenames.txt | cut -d"." -f1 | sort | uniq >
↪  unique_ident_fileName.txt
```

- Script

```
module load bwa/0.7.17

while read ident; do
  bwa mem ~/scratch/genome_Asyriaca/Asclepias_syriaca_v1.0_chromosomes.fa
  ↪  3A_R1_${ident}.fq 3A_R2_${ident}.fq > ../../step04_align/plate3_r2/3A_${ident}.sam
done <unique_ident_fileName.txt
```

Convert to .bam

- Install samtools v 1.16.1
- Get filename text file

```
ls *.sam > whole_filenames.txt

awk -F_  '{print $2}' whole_filenames.txt | cut -d"." -f1 | sort | uniq >
↪  unique_ident_fileName.txt
```

Plates 1-2

```
module load samtools/1.16.1

while read ident; do
    samtools view -S -b 1.2_${ident}.sam >
    ↪  ../../step04_align/plates1_2/bamfiles/1.2_${ident}.bam
done <unique_ident_fileName.txt
```

Plate 3, run 1

```
module load samtools/1.16.1

while read ident; do
   samtools view -S -b 3_${ident}.sam >
   ↪  ../../step04_align/plate3_r1/bamfiles/3_${ident}.bam
done <unique_ident_fileName.txt
```

Plate 3, run 2

```
module load samtools/1.16.1

while read ident; do
  samtools view -S -b 3A_${ident}.sam >
  ↪  ../../step04_align/plate3_r2/bamfiles/3A_${ident}.bam
done <unique_ident_fileName.txt
```

QA/QC each bam file

- Use Bamtools stats
- Install bamtools v 2.5.1

Plates 1-2

```
module load bamtools/2.5.1

while read ident; do
  bamtools stats -in 1.2_${ident}.bam > 1.2_${ident}_stats.txt
done < ../unique_ident_fileName.txt
```

Plate 3, R1

```
module load bamtools/2.5.1

while read ident; do
  bamtools stats -in 3_${ident}.bam > 3_${ident}_stats.txt
done < ../unique_ident_fileName.txt
```

Plate 3, R2

```
module load bamtools/2.5.1

while read ident; do
  bamtools stats -in 3A_${ident}.bam > 3A_${ident}_stats.txt
done < ../unique_ident_fileName.txt
```

- Analyzed stats with R script: bamtools_stats_summary.R

Sort w/samtools

- Scripts here (step05_sort)
- Install samtools v 1.16.1

Plates 1-2

- Get names of files

```
ls 1.2_*.bam > whole_filenames_pl1_2.txt

awk -F_  '{print $2}' whole_filenames_pl1_2.txt | cut -d"." -f1 | sort | uniq >
↪  unique_ident_fileName_pl1_2.txt
```

- Sort

```
module load samtools/1.16.1

while read ident; do
  samtools sort 1.2_${ident}.bam -o 1.2_${ident}_sorted.bam
done < unique_ident_fileName_pl1_2.txt
```

Plate 3

- Get names of files

```
ls 3A_*.bam > whole_filenames_pl3.txt

awk -F_  '{print $2}' whole_filenames_pl3.txt | cut -d"." -f1 | sort | uniq >
↪  unique_ident_fileName_pl3.txt
```

- Sort

```
module load samtools/1.16.1

while read ident; do
  samtools sort 3A_${ident}.bam -o 3A_${ident}_sorted.bam
done < unique_ident_fileName_pl3.txt
```

Run the Stacks ref_map.pl script to create a filtered genome-wide SNP library

- Scripts and output here (step06_callSNPs) For the first run, I had included all 263 samples. This details the second run, which excludes 2 low-quality individuals (Plate 3, wells A1 and A10).

NOTE: I generated many vcf files that were frequently too large to push to github. That is why they aren't available on github.

- Create a population map text file EXCLUDING 2 low-quality individuals: Plate 3, A1 (MWI057) and A10 (MW018) ("pop_map2.txt")

```
module load stacks/2.62

ref_map.pl --samples sorted_bamfiles --popmap pop_map2.txt --out-path output_refmap_run2
```

- Evaluated new ref_map.pl output for different scenarios

Run Stacks populations to get f statistics

- Performed several trials, with different R and min-maf parameters.

Trial name	R	r	min-maf
01	0.00760456273764	0.2	0
02	0.01	0.2	0
03	0.25	0.2	0
04	0.5	0.2	0
05	0.75	0.2	0
06	1	0.2	0
13	0.0038, AKA 1/261	0.2	0.05
07	0.0076, AKA 2/261	0.2	0.05
08	0.01	0.2	0.05
09	0.25	0.2	0.05
10	0.5	0.2	0.05
11	0.75	0.2	0.05
12	1	0.2	0.05

Example script for Trial 1

```
module load stacks/2.62

populations -P output_refmap_run2 -O output_filtering_fstats_trial01 --popmap
↪  pop_map2.txt -t 8 --hwe --fstats --plink --structure --vcf --verbose -R
↪  0.00760456273764 -r 0.2 --min-maf 0
```

Evaluate f stats output for different scenarios

- Chose -R = 1 and -min_maf = 0.05 for gene flow analyses
- Chose -R = 0.5 and -min_maf 0.0076 for demographic/genetic diversity analyses

Find out how many clones/relatives are in my dataset

- Output here (step07_plink)
- Using the mmaf = 0.05 and R = 1 plink files
  - PLINK documentation
- First, get the input files:

```
plink --file populations.plink --maf 0.05 --geno 0.5 --make-bed --allow-extra-chr --out
↪  mmaf0.05_R1

# --file: the prefix of the .map and .ped files (i.e., populations.plink.map)
# --maf: filters out all variants with minor allele frequency below the provided
↪  threshold (default 0.01). This removes monomorphic alleles too (where all SNPS are
↪  the same among individuals --> no information about differences, just wastes computer
↪  power)
# --geno: filters out all variants with missing call rates exceeding the provided value
↪  (default 0.1)
# --make_bed: creates a new PLINK 1 binary fileset, after applying sample/variant filters
↪  and other operations below. (see example on website)
# --allow-extra-chr: force my chromosome names to be accepted (they dont' start
↪  w/numbers). this package was made for human genomic data
# --out: output prefix
```

- Create eigenvector and eigenvalue files (.eigenval & .eigenvec files)

```
plink --bfile mmaf0.05_R1 --allow-extra-chr --pca --out mmaf0.05_R1

# --bfile: binary file prefix
# --pca: create pca files
# --out: prefix of output files
```

- Script for finding relatives:

```
plink --file populations.plink --genome --min 0.25 --allow-extra-chr --out
↪  mmaf0.0078_R0.5_testingIBD

# min 0.25 = keep the individuals that are at least 25% related by descent (Identity by
↪  Descent = IBD)
```

- 5 individual pairs with IBD > 25%. Removed 1 individual from each pair (see below).

Run the Stacks ref_map.pl script to create a new filtered genome-wide SNP library

Third run- excluding 2 low-quality individuals PLUS (*new*) 5 IBD individuals

- Script and some output here (step08_no-IBD/refmap_no-IBD)
- Create a NEW population map text file EXCLUDING 2 low-quality individuals and 5 IBD individuals ("pop_map3.txt")
- *Chose these 5 individuals because they were in the first of two columns of individuals- simple and somewhat random.*
  - 1.2_1A6_sorted
  - 1.2_1E9_sorted
  - 1.2_1F6_sorted
  - 1.2_2D3_sorted
  - 3A_E7_sorted
- New directory: step08_no-IBD

```
module load stacks/2.62

mkdir output_refmap_no-IBD

ref_map.pl --samples sorted_bamfiles_no-IBD --popmap pop_map3.txt --out-path
↪  output_refmap_no-IBD
```

Run Stacks populations

- Scripts and some output here (step08_no-IBD/populations_no-IBD)
- Changed R from 1 to 0.75 because the former left too few SNPs.

Run populations script to get global Fst:

- See main directory

```
module load stacks/2.62

mkdir global_fst

populations -P output_refmap_no-IBD -O global_fst --fstats --vcf -t 8 -R 0.75 -r 0.2
↪  --min-maf 0.05 --write-single-snp
```

Run populations again with -vcf-all flag

- See main directory
- Named new .sh file populations_no-IBD_vcf-all.sh
- ended up splitting this into two .sh files to make it faster

Run populations script to get global Fst:

```
module load stacks/2.62

mkdir global_fst_vcf-all

populations -P output_refmap_no-IBD -O global_fst_vcf-all --vcf-all -t 8 -R 0.75 -r 0.2
↪  --min-maf 0.05 --write-single-snp --plink
```

Run populations script to get global pi:

```
module load stacks/2.62

mkdir global_pi_vcf-all

populations -P output_refmap_no-IBD -O global_pi_vcf-all --vcf-all -t 8 -R 0.5 -r 0.2
↪  --min-maf 0.007662 --write-single-snp --plink
```

Run populations script to get pi for urban vs. rural pops:

```
module load stacks/2.62

mkdir UR_dist_pi_vcf-all

populations -P output_refmap_no-IBD -O UR_dist_pi_vcf-all --popmap pop_map3_UR_dist.txt
↪  --vcf-all -t 8 -R 0.5 -r 0.2 --min-maf 0.007662 --write-single-snp --plink
```

Run populations script to get Fst for urban vs. rural pops:

```
module load stacks/2.62

mkdir UR_dist_fst_vcf-all

populations -P output_refmap_no-IBD -O UR_dist_fst_vcf-all --popmap pop_map3_UR_dist.txt
↪  --vcf-all -t 8 -R 0.75 -r 0.2 --min-maf 0.05 --write-single-snp --plink
```

NOTE: I created a new step10 folder and redid analyses with min_maf = 2/251 because I incorrectly subtracted all 10 related individuals from the pool instead of just 5. That data is in the step10_mmaf0.007968 directory, but wasn't used. Just explaining it here for clarity.

Redo populations again for mmaf=2/256 (the correct total after removing 1 related individual from 5 pairs)

- 2/256 = 0.0078125 —> 0.007813
- Scripts and output here (step11_mmaf0.007813)

```
module load stacks/2.62

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O
↪  ~/scratch/step11_mmaf0.007813/output_populations_no-IBD-vcf-all/mmaf0.007813_R0.5/
↪  --popmap ~/step08_no-IBD/pop_map3.txt -t 8 --hwe --fstats --plink --structure
↪  --vcf-all --genepop --verbose -R 0.5 -r 0.2 --min-maf 0.0078125 --write-single-snp
```

Run populations script to get global pi:

```
module load stacks/2.62

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O
↪  ~/scratch/step11_mmaf0.007813/global_pi_vcf-all --vcf-all -t 8 -R 0.5 -r 0.2
↪  --min-maf 0.0078125 --write-single-snp --plink
```

Run populations script to get pi for urban vs. rural pops (based on distance to the city center):

```
module load stacks/2.62

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O UR_dist_pi_vcf-all --popmap
↪  ~/step08_no-IBD/pop_map3_UR_dist.txt --vcf-all -t 8 -R 0.5 -r 0.2 --min-maf 0.0078125
↪  --write-single-snp --plink
```

Run populations script to get pi for urban vs. rural pops (based on urbanization score):

```
module load stacks/2.62

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O UR_usc_pi_vcf-all --popmap
↪  ~/step08_no-IBD/pop_map3_UR_usc.txt --vcf-all -t 8 -R 0.5 -r 0.2 --min-maf 0.0078125
↪  --write-single-snp --plink
```

Run populations script to get global FIS:

```
module load stacks/2.62

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O
↪  ~/scratch/global_urban_rural_fis/global_fis --vcf-all -t 8 -R 0.75 -r 0.2 --min-maf
↪  0.05 --write-single-snp --plink
```

Run populations script to get FIS for urban vs. rural pops (based on distance to the city center):

```
module load stacks/2.62

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O global_urban_rural_fis/UR_fis_dist
↪  --popmap ~/step08_no-IBD/pop_map3_UR_dist.txt --vcf-all -t 8 -R 0.75 -r 0.2 --min-maf
↪  0.05 --write-single-snp --plink
```

Run populations script to get FIS for urban vs. rural pops (based on urbanization score):

```
module load stacks/2.62

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O global_urban_rural_fis/UR_fis_usc
↪  --popmap ~/step08_no-IBD/pop_map3_UR_usc.txt --vcf-all -t 8 -R 0.75 -r 0.2 --min-maf
↪  0.05 --write-single-snp --plink
```

Analyze gene flow

Create PCA to find number of clusters

- run script that filters out SNPs with missing call rates >50% and creates .bed file

```
plink --file populations.plink --maf 0.05 --geno 0.5 --make-bed --allow-extra-chr --out
↪  mmaf0.05_R0.75
```

- Create eigenvector and eigenvalue files (.eigenval & .eigenvec files)

```
plink --bfile mmaf0.05_R0.75 --allow-extra-chr --pca --out mmaf0.05_R0.75
```

- Analyze with R

conStruct

- See scripts and output here (conStruct)
- Also see R script for more context and process for running program

Fst

- These scripts involve running Stacks populations scripts with various popmaps to obtain vcfs used for calculating fst for specific subgroups of individuals.

Compute fst for urban sites ONLY, urbanization based on distance to the city center

- Run population script again, with popmap having ONLY urban sites and mmaf = 0.05, R = 0.75
  - made new pop map with only urban pops: pop_map3_U_dist.txt (see R script)

```
module load stacks/2.62

mkdir U_dist_fst_vcf-all

populations -P output_refmap_no-IBD -O U_dist_fst_vcf-all --popmap pop_map3_U_dist.txt
↪  --vcf-all -t 8 -R 0.75 -r 0.2 --min-maf 0.05 --write-single-snp --plink
```

Compute fst for urban vs. rural sites, urbanization based on urbanization score

```
module load stacks/2.62

mkdir ~/scratch/step09_vcftools/fst_mmaf0.05_R0.75_no-IBD_vcf-all/UR_usc_fst_vcf-all

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O UR_usc_fst_vcf-all --popmap
↪  ~/step08_no-IBD/pop_map3_UR_usc.txt --vcf-all -t 8 -R 0.75 -r 0.2 --min-maf 0.05
↪  --write-single-snp --plink
```

Run populations script to get Fst for urban vs. rural pops (urbanization based on urbanization score):

Compute fst for urban sites ONLY, urbanization based on urbanization score

```
module load stacks/2.62

mkdir U_usc_fst_vcf-all

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O U_usc_fst_vcf-all --popmap
↪  ~/step08_no-IBD/pop_map3_U_usc.txt --vcf-all -t 8 -R 0.75 -r 0.2 --min-maf 0.05
↪  --write-single-snp --plink
```

Compute fst for rural sites ONLY, urbanization based on urbanization score

```
module load stacks/2.62

mkdir R_usc_fst_vcf-all

populations -P ~/step08_no-IBD/output_refmap_no-IBD -O R_usc_fst_vcf-all --popmap
↪  ~/step08_no-IBD/pop_map3_R_usc.txt --vcf-all -t 8 -R 0.75 -r 0.2 --min-maf 0.05
↪  --write-single-snp --plink
```

- then analyzed with Python and R.

MEMgene

Run script to filter vcf for urban pops (based on distance to the city center):

#1: with all sites, including monomorphic sites

- Load vcftools v 0.1.16

```
vcftools  --vcf ~/step08_no-IBD/global_fst_vcf-all/populations.all.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/urban_pops_dist_mmaf0.05 --keep
↪  ~/step08_no-IBD/pop_map3_U_dist.txt

# Keeping urban (based on distance) individuals in the pop_map3_U_dist.txt popmap
```

#2: without monomorphic sites

```
# here, I use populations.snps.vcf instead of populations.all.vcf, as I did above

vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/urban_pops_dist_mmaf0.05_variant_sites --keep
↪  ~/step08_no-IBD/pop_map3_U_dist.txt
```

- analyze this and next datasets in R with the #2 option (without monomorphic sites) because the monomorphic sites would've been tossed anyway

1 individual per population

- created new txt in filtered_vcf folder by copy/pasting clean_data/pop_map3_1indiv_per_pop.txt

Run script to filter vcf for just 1 individual per pop:

```
# forgot the "pop" in the output file name

vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/1indiv_per_mmaf0.05_variant_sites --keep
↪  ~/scratch/filtered_vcfs/pop_map3_1indiv_per_pop.txt
```

1 individual per population for just RURAL (based on distance to the city center)

created new vcf with just 1 individual/population

- created new txt in filtered_vcf folder by copy/pasting clean_data/pop_map3_1indiv_per_pop_RURAL.txt (same name on the cluster)

Run script to filter vcf for just 1 individual per pop, RURAL:

```
# woops, forgot the "pop" in the output file name

vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/1indiv_per_mmaf0.05_variant_sites_RURAL --keep
↪  ~/scratch/filtered_vcfs/pop_map3_1indiv_per_pop_RURAL.txt
```

create vcf w/1 individual per population for just URBAN sites

create new vcf with just 1 individual/population

- created new txt in filtered_vcf folder by copy/pasting clean_data/pop_map3_1indiv_per_pop_URBAN.txt (same name on the cluster)

Run script to filter vcf for just 1 individual per pop, URBAN:

```
# forgot the "pop" in the output file name

vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/1indiv_per_mmaf0.05_variant_sites_URBAN --keep
↪  ~/scratch/filtered_vcfs/pop_map3_1indiv_per_pop_URBAN.txt
```

Mantel correlogram

- Create new filtered vcfs

Rural, based on distance to the city center

Run script to filter vcf for rural pops (based on distance to the city center):

```
vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/rural_pops_dist_mmaf0.05_variant_sites --keep
↪  ~/step08_no-IBD/pop_map3_R_dist.txt
```

Rural, based on urbanization score

Run script to filter vcf for rural pops (based on urbanization score):

```
vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/rural_pops_usc_mmaf0.05_variant_sites --keep
↪  ~/step08_no-IBD/pop_map3_R_usc.txt
```

Urban, based on urbanization score

Run script to filter vcf for urban pops (based on urbanization score):

```
vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/urban_pops_usc_mmaf0.05_variant_sites --keep
↪  ~/step08_no-IBD/pop_map3_U_usc.txt
```

1 individual per population, based on urbanization score (RURAL)

create new vcf with just 1 individual/population

- create new txt with R script create_popmaps.Rmd

Run script to filter vcf for just 1 individual per pop, RURAL:

```
vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/rural/1indiv_per_pop_mmaf0.05_variant_sites_RURAL_usc --keep
↪  ~/scratch/filtered_vcfs/rural/pop_map3_1indiv_per_pop_RURAL_usc.txt
```

1 individual per population, based on urbanization score (URBAN)

create new vcf with just 1 individual/population

- create new txt with R script create_popmaps.Rmd

Run script to filter vcf for just 1 individual per pop, URBAN:

```
vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/filtered_vcfs/urban/1indiv_per_pop_mmaf0.05_variant_sites_URBAN_usc --keep
↪  ~/scratch/filtered_vcfs/urban/pop_map3_1indiv_per_pop_URBAN_usc.txt
```

hierfstat: creating new vcf without populations with 1 individual

All individuals

- created new pop map with only pops with \>1 individual in R

  (pop_map3_MORE_THAN_1indiv_per_pop.txt)

Run script to filter vcf by removing pops with 1 individual

```
vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/hierfstat_vcf/pops_MORE_THAN_1indiv_per_pop_mmaf0.05_variant_sites --keep
↪  ~/scratch/hierfstat_vcf/pop_map3_MORE_THAN_1indiv_per_pop.txt
```

Urban individuals (based on distance to city center)

- created new pop map with only pops with \>1 individual in R

  (pop_map3_MORE_THAN_1indiv_per_pop_URBAN.txt)

Run script to filter vcf by removing pops with 1 individual from URBAN pops

```
vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/hierfstat_vcf/pops_MORE_THAN_1indiv_per_pop_URBAN_mmaf0.05_variant_sites
↪  --keep ~/scratch/hierfstat_vcf/pop_map3_MORE_THAN_1indiv_per_pop_URBAN.txt
```

Rural individuals (based on distance to city center)

- created new pop map with only pops with >1 individual in R

  (pop_map3_MORE_THAN_1indiv_per_pop_RURAL.txt)

Run script to filter vcf by removing pops with 1 individual from RURAL pops

```
vcftools  --vcf ~/step08_no-IBD/global_fst/populations.snps.vcf  --recode  --out
↪  ~/scratch/hierfstat_vcf/pops_MORE_THAN_1indiv_per_pop_RURAL_mmaf0.05_variant_sites
↪  --keep ~/scratch/hierfstat_vcf/pop_map3_MORE_THAN_1indiv_per_pop_RURAL.txt
```

Genetic diversity and demographic analyses

Tajima's D

- with mmaf = 2/256 dataset (CORRECT- 0.007813)
- Scripts and output here (step09_vcftools/TajD_mmaf0.007813...)

For ALL SITES (1 population)

- Window = 1,000,000 bp

Ran script:

```
vcftools --vcf ~/scratch/step11_mmaf0.007813/global_pi_vcf-all/populations.all.vcf --out
↪  global_TajD_window_1mil --TajimaD 1000000

# added the --TajimaD argument of 1,000,000 (window length to calculate Tajima's D over)
```

- Window = 100,000 bp

Ran script:

```
vcftools --vcf ~/scratch/step11_mmaf0.007813/global_pi_vcf-all/populations.all.vcf --out
↪  global_TajD_window_100k --TajimaD 100000

# added the --TajimaD argument of 100,000
```

For JUST URBAN sites

Urbanization based on distance to the city center Run script to filter vcf for urban pops (based on distance to the city center):

```
mkdir ~/scratch/temp_step10/Urban_dist_pi_vcf-all

vcftools  --vcf ~/scratch/step11_mmaf0.007813/UR_dist_pi_vcf-all/populations.all.vcf
↪  --recode  --out ~/scratch/temp_step10/Urban_dist_pi_vcf-all/urban_populations --keep
↪  ~/step08_no-IBD/pop_map3_U_dist.txt

# Keeping urban (based on distance) individuals in the pop_map3_U_dist.txt popmap
```

- Window = 1,000,000 bp

Ran script:

```
vcftools --vcf ~/scratch/temp_step10/Urban_dist_pi_vcf-all/urban_populations.recode.vcf
↪  --out Urban_dist_TajD_window_1mil --TajimaD 1000000
```

- Window = 100,000 bp

Ran script:

```
vcftools --vcf ~/scratch/temp_step10/Urban_dist_pi_vcf-all/urban_populations.recode.vcf
↪  --out Urban_dist_TajD_window_100k --TajimaD 100000
```

Urbanization based on urbanization score Run script to filter vcf for urban pops (based on urbanization score):

```
mkdir ~/scratch/temp_step10/Urban_usc_pi_vcf-all

vcftools  --vcf ~/scratch/step11_mmaf0.007813/UR_usc_pi_vcf-all/populations.all.vcf
↪  --recode  --out ~/scratch/temp_step10/Urban_usc_pi_vcf-all/urban_populations --keep
↪  ~/step08_no-IBD/pop_map3_U_usc.txt
```

- Window = 1,000,000 bp

Ran script:

```
vcftools --vcf ~/scratch/temp_step10/Urban_usc_pi_vcf-all/urban_populations.recode.vcf
↪  --out Urban_usc_TajD_window_1mil --TajimaD 1000000
```

- Window = 100,000 bp

Ran script:

```
vcftools --vcf ~/scratch/temp_step10/Urban_usc_pi_vcf-all/urban_populations.recode.vcf
↪  --out Urban_usc_TajD_window_100k --TajimaD 100000
```

For JUST RURAL sites

Urbanization based on distance to the city center Run script to filter vcf for rural pops (based on distance to the city center):

```
mkdir ~/scratch/temp_step10/Rural_dist_pi_vcf-all

vcftools  --vcf ~/scratch/step11_mmaf0.007813/UR_dist_pi_vcf-all/populations.all.vcf
↪  --recode  --out ~/scratch/temp_step10/Rural_dist_pi_vcf-all/rural_populations --keep
↪  ~/step08_no-IBD/pop_map3_R_dist.txt
```

- Window = 1,000,000 bp

Ran script:

```
vcftools --vcf ~/scratch/temp_step10/Rural_dist_pi_vcf-all/rural_populations.recode.vcf
↪  --out Rural_dist_TajD_window_1mil --TajimaD 1000000
```

- Window = 100,000 bp

Ran script:

```
vcftools --vcf ~/scratch/temp_step10/Rural_dist_pi_vcf-all/rural_populations.recode.vcf
↪  --out Rural_dist_TajD_window_100k --TajimaD 100000
```

Urbanization based on urbanization score Run script to filter vcf for rural pops (based on urbanization score):

```
mkdir ~/scratch/temp_step10/Rural_usc_pi_vcf-all

vcftools  --vcf ~/scratch/step11_mmaf0.007813/UR_usc_pi_vcf-all/populations.all.vcf
↪  --recode  --out ~/scratch/temp_step10/Rural_usc_pi_vcf-all/rural_populations --keep
↪  ~/step08_no-IBD/pop_map3_R_usc.txt
```

- Window = 1,000,000 bp

Ran script:

```
vcftools --vcf ~/scratch/temp_step10/Rural_usc_pi_vcf-all/rural_populations.recode.vcf
↪  --out Rural_usc_TajD_window_1mil --TajimaD 1000000
```

- Window = 100,000 bp

Ran script:

```
vcftools --vcf ~/scratch/temp_step10/Rural_usc_pi_vcf-all/rural_populations.recode.vcf
↪  --out Rural_usc_TajD_window_100k --TajimaD 100000
```

Stairway Plot

- Scripts and output here (stairway_plot)
- downloaded files from https://github.com/xiaoming-liu/stairway-plot-v2/blob/master/stairway_plot_v2.1.1.zip
- created sfs in R for all populations, just urban, just rural, etc. ("sfs_for_stairwayplot.R")
- make java script executable:

```
chmod -R 777 ./
```

- Chose four mutation rates:
  1. 1 x 10^-8
  2. 1.2 x 10^-8
  3. 1.8 x 10^-8
  4. 2.6 x 10^-8
- Chose two generation times:
  - 1 year
  - 2 years
- Ran demographic modelling for
  - entire population
  - urban based on distance to the city center
  - urban based on urbanization score
  - rural based on distance to the city center
  - rural based on urbanization score
  - Only some mutation rates & generation timeshave all fivemodels, but all mutation rates have data for entire population.
- Example workflow (see program instructions for full details):
  1. Make blueprint file (e.g., "full_mmaf0.007813.blueprint")
  2. run this code in the terminal : `java -cp stairway_plot_es Stairbuilder full_mmaf0.007813.blueprint`
  3. Make a .sh script
  4. Run it: `bash full_mmaf0.007813.blueprint.sh`