**AGRO-932 HW1**

**Santos Barrera Lemus**

**02-20-2020**

**1.** Establish a version-controlled directory system to host the project.

[**https://github.com/Sybalemus1/Agro932hw.git**](https://github.com/Sybalemus1/Agro932hw.git)

**2.** Describe your simulation strategy and the hypothesis to test (positive, negative, or neutral selection).

**Simulation strategy**

I used the wgsim program to generate the NGS simulated data of two populations with 10 individuals each one (I ran the program twice to generate the 20 individual). Then I download the reference genome (Zea\_mays) using the wget program. The genome used was small with about 500k, 20x coverage, 100bp pair ends reads, only SNP no Indel, and a low mutation rate.

**Hypothesis**

To determinate how different the both simulated populations are, is necessary to determinate the Fst values. Two scenarios 1). If the Fst value is = 0 it indicated that there is not population differentiation which indicates no evolution forces involved. 2) If the Fst value is = 1 it indicated completely differentiation (completely homozygous). Under this concept we can assume that if on average the Fst value is low there is a negative selection. (me falta mas en esta parte)

**3.** Use ANGSD to calculate thetas for each population and Fst between the two populations.

**Distribution of site frequency spectrum of alleles (SFS) and pairwise theta distribution**



Scatter plot of the Fst values:

A screenshot of a cell phone

Description automatically generated

Fst values are low (between 0.02 and 0.006) indicating that there is no population differentiation

**4.** Breakdown the theta ratios and Fst values into different genomic features (i.e., genic and intergenic regions) according to the general feature format (GFF) file for the reference genome of your choice.



FstRplot

FstRplot FstRplot

hw1

Santos Barrera

2/26/2020

# NGS data simulation using wgsim

#### Type in the following command:

wgsim lambda.fa -e 0 -d 500 -N 5000 -1 100 -2 100 -r 0.01 -R 0 -X 0 -S 1234567 -h l1.read1.fq l1.read2.fq

# Download Reference from EnsemblPlants

Maize [reference genome](https://plants.ensembl.org/Zea_mays/Info/Index)

#### Change to largedata\lab4 folder:

cd largedata  
mkdir Thetas\_Fst  
cd Thetas\_Fst

#### Then use wget to download the reference genome:

wget ftp://ftp.ensemblgenomes.org/pub/plants/release-46/fasta/zea\_mays/dna/Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa.gz  
  
### then unzip it  
gunzip Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa.gz

# NGS data simulation using wgsim

#### Type in the following command:

wgsim Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa \  
-e 0 -d 500 -N 5000 -1 100 -2 100 -r 0.01 \  
-R 0 -X 0 -S 1234567 l1.read1.fq l1.read2.fq

* Reference (about 500k)
  + Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa
* 20x coverage
  + N 5000
* PE 100bp
  + -1 100 -2 100
* Only SNP no Indel
  + -R 0 -X 0
* Mutation rate is low
  + -r 0.01

# NGS data simulation using wgsim

## simulate 10 individals population 1

for i in {1..10}; do wgsim Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa -e 0 -d 500 -N 50000 -1 100 -2 100 -r 0.1 -R 0 -X 0 p1l$i.read1.fq l$i.read2.fq; done  
mv l1.read1.fq p1l1.read1.fq #(Change name pop 1. all 10 idividuals)

## simulate 10 individals population 2

–

for i in {1..10}; do wgsim Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa -e 0 -d 500 -N 50000 -1 100 -2 100 -r 0.1 -R 0 -X 0 l$i.read1.fq l$i.read2.fq; done  
mv l1.read1.fq p2l1.read1.fq #(Change name pop 2. all 10 idividuals)

|  |
| --- |
| #### check how many reads |
| bash wc -l p1l1.read1.fq wc -l p2l1.read1.fq # suppose to be 200,000 lines = 50,000 reads |

# A procedure to calculate and values

### 1. Align the NGS reads to the reference genome

* [bwa](https://github.com/lh3/bwa)
* [samtools](https://github.com/samtools/samtools)

### 2. Calculate SFS

* [ANGSD](http://www.popgen.dk/angsd/index.php/ANGSD)

### 3. Calculate the thetas and Fst for each site

* ANGSD

# Submit a Slurm job (I couldn’t do it)

* We wrap our jobs in little batch scripts, which is nice because these also help make steps reproducible.
* To keep your directory organized, I usually keep a scripts directory (i.e., slurm-script/ ) and log dir (i.e., slurm-log ) for Slurm’s logs.
  + Tip: use these logs, as these are very helpful in debugging. I separate them from my project because they fill up directories rather quickly.
* D sets your project directory.
* o sets where standard output (of your batch script) goes.
* e sets where standard error (of your batch script) goes.
* J sets the job name.
* t sets the time limit for the job, 24:00:00 indicates 24 hours.
* --mail: will email you if the job is “END” or “FAIL”

–

vi my\_theta.sh  
  
 #!/bin/bash -l  
 #SBATCH -D /home/agro932/sybarreral/Agro932hw  
 #SBATCH -o /home/agro932/sybarreral/Agro932hw/slurm-log/stdout-%j.txt  
 #SBATCH -e /home/agro932/sybarreral/Agro932hw/slurm-log/stderr-%j.txt  
 #SBATCH -J theta  
 #SBATCH -t 1:00:00  
 #SBATCH --mail-user=sybarreral@huskers.unl.edu  
 #SBATCH --mail-type=END #email if ends  
 #SBATCH --mail-type=FAIL #email if fails  
 set -e  
 set -u  
  
# insert your script here  
module load bwa samtools  
# alignment pop.1  
cd largedata/Thetas\_Fst/bam\_files  
for i in {1..10}; do bwa mem Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa p1l$i.read1.fq p1l$i.read2.fq | samtools view -bSh - > p1l$i.bam; done  
# alignment pop.2  
for i in {1..10}; do bwa mem Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa p2l$i.read1.fq p2l$i.read2.fq | samtools view -bSh - > p2l$i.bam; done  
# sort  
for i in \*.bam; do samtools sort $i -o sorted\_$i; done  
# index them  
for i in sorted\*.bam; do samtools index $i; done

# A procedure to calculate values

### 2. Calculate SFS using ANGSD

#### Install ANGSD first

cd ~/bin/ # if you don't have one, do `mkdir bin`  
git clone https://github.com/samtools/htslib.git  
git clone https://github.com/ANGSD/angsd.git   
cd htslib; make;   
cd ../angsd;  
make HTSSRC=../htslib

#### run angsd

#write the bam files to a txt file  
mkdir bam\_files  
mv sorted\*.bam bam\_files  
cd bam\_files/  
ls sorted\_p1\*.bam > bam.txt  
#see file 1  
cat bam\_p1.txt   
ls sorted\_p2\*.bam > bam.txt  
#see file 2  
cat bam\_p2.txt

# A procedure to calculate values

#calculate Sfs   
../../../../angsd/angsd -bam bam.txt -doSaf 1 -anc ../Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa -GL 1 -out out  
# use realSFS to calculate sfs  
../../../../angsd/misc/realSFS out.saf.idx > out.sfs

#### Copy the result to cache/ folder

## cp sfs to the cache/ folder  
cp out.sfs ../../../cache/

### 3. Calculate the thetas for each site

The output from the above command are two files out.thetas.gz and out.thetas.idx. A formal description of these files can be found in the doc/formats.pdf in the angsd package. It is possible to extract the logscale persite thetas using the ./thetaStat print program.

../../../../angsd/angsd -bam bam.txt -out out -doThetas 1 -doSaf 1 -pest out.sfs -anc ../Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa -GL 1  
../../../../angsd/misc/thetaStat print out.thetas.idx > theta.txt  
## cp theta to the cache/ folder  
cp theta.txt ../../../cache/

# A procedure to calculate

#### Create two list bam files

cp bam\_p1.txt pop1.txt  
cp bam\_p2.txt pop2.txt

#### Two population Fst

# first calculate per pop saf for each populatoin  
../../../../angsd/angsd -b pop1.txt -anc ../Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa -out pop1 -dosaf 1 -gl 1  
../../../../angsd/angsd -b pop2.txt -anc ../Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa -out pop2 -dosaf 1 -gl 1  
# calculate the 2dsfs prior  
../../../../angsd/misc/realSFS pop1.saf.idx pop2.saf.idx > pop1.pop2.ml  
# prepare the fst for easy window analysis etc  
../../../../angsd/misc/realSFS fst index pop1.saf.idx pop2.saf.idx -sfs pop1.pop2.ml -fstout out  
  
# get the global estimate  
../../../../angsd/misc/realSFS fst stats out.fst.idx  
# below is not tested that much, but seems to work  
../../../../angsd/misc/realSFS fst stats2 out.fst.idx -win 500 -step 100 > fst\_win.txt

#### Copy the result to cache/ folder

## cp sfs to the cache/ folder  
cp fst\_win.txt ../../../cache/

# Visualize the results

In local computer, using R:

#### Barplot for SFS

#Barplot for SFS  
sfs <- scan('cache/out.sfs')  
sfs <- sfs[-c(1,length(sfs))]  
sfs <- sfs/sum(sfs)  
  
barplot(sfs,names=1:length(sfs), main='SFS')  
barplot(sfs, col="#cdc0b0", xlab="No. of segregating sites",   
 ylab="Minor allele frequency",   
 names=1:length(sfs))

#### Histgram distribution of the theta values

library("data.table")  
#Histgram distribution of the theta values  
#theta <- read.table("cache/theta.txt", header=TRUE)  
theta <- fread("cache/theta.txt", data.table =FALSE)  
hist(theta$Pairwise)   
hist(theta$Pairwise, col="green", xlab="Theta pairwise values (log10)")

#### Scatter plot of the Fst values

#Scatter plot of the Fst values  
fst <- read.table("cache/fst\_win.txt", skip=1, header=FALSE)  
names(fst)[c(3,5)] <- c("midp", "fst")  
plot(fst$midp, fst$fst, xlab="Physical position", ylab="Fst", col="#5f9ea0", pch=16)

# General feature format (GFF) from EnsemblPlants

Maize [reference genome](https://plants.ensembl.org/Zea_mays/Info/Index)

change to largedata\lab4 folder:

wget ftp://ftp.ensemblgenomes.org/pub/plants/release-46/fasta/zea\_mays/dna/Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa.gz  
  
### then unzip it  
gunzip Zea\_mays.B73\_RefGen\_v4.dna.chromosome.Mt.fa.gz

Similarly, we will download and unzip the [Mt GFF3](ftp://ftp.ensemblgenomes.org/pub/plants/release-46/gff3/zea_mays/Zea_mays.B73_RefGen_v4.46.chromosome.Mt.gff3.gz) file

–

#### Use R to process the GFF3 file

# install.package("data.table")  
library("data.table")  
## simply read in wouldn't work  
gff <- fread("largedata/Zea\_mays.B73\_RefGen\_v4.46.chromosome.Mt.gff3", skip="#", header=FALSE, data.table=FALSE)  
## grep -v means select lines that not matching any of the specified patterns  
gff <- fread(cmd='grep -v "#" largedata/Zea\_mays.B73\_RefGen\_v4.46.chromosome.Mt.gff3', header=FALSE, data.table=FALSE)

# General feature format (GFF) version 3

V1 V2 V3 V4 V5 V6 V7 V8  
1 Mt Gramene chromosome 1 569630 . . .  
2 Mt ensembl gene 6391 6738 . + .  
3 Mt NCBI mRNA 6391 6738 . + .  
4 Mt NCBI exon 6391 6738 . + .  
5 Mt NCBI CDS 6391 6738 . + 0  
6 Mt ensembl gene 6951 8285 . + .  
 V9  
1 ID=chromosome:Mt;Alias=AY506529.1,NC\_007982.1;Is\_circular=true  
2 ID=gene:ZeamMp002;biotype=protein\_coding;description=orf115-a1;  
3 ID=transcript:ZeamMp002;Parent=gene:ZeamMp002;  
4 Parent=transcript:ZeamMp002;Name=ZeamMp002.exon1;constitutive=1;ensembl\_end\_phase=0;  
5 ID=CDS:ZeamMp002;Parent=transcript:ZeamMp002;  
6 ID=gene:ZeamMp003;biotype=protein\_coding;description=orf444

# Work with GFF

names(gff) <- c("seq", "source", "feature", "start", "end", "score", "strand", "phase", "att")  
table(gff$feature)

### Get genes and upstream and downstream 5kb regions

g <- subset(gff, feature %in% "gene")  
g$geneid <- gsub(".\*gene:|;biotype.\*", "", g$att)  
  
### + strand  
gp <- subset(g, strand %in% "+")   
# nrow(gp) 75  
  
### get the 5k upstream of the + strand gene model  
gp\_up <- gp  
gp\_up$end <- gp\_up$start - 1  
gp\_up$start <- gp\_up$end - 5000   
  
### get the 5k downstream of the + strand gene model  
gp\_down <- gp  
gp\_down$start <- gp\_down$end + 1  
gp\_down$end <- gp\_down$start + 5000

### Get genes and upstream and downstream 5kb regions

### - strand  
gm <- subset(g, strand %in% "-")   
dim(gm) # 82  
  
fwrite(g, "cache/mt\_gene.txt", sep="\t", row.names = FALSE, quote=FALSE)

## Intepret the theta results

library("data.table")  
library("GenomicRanges")  
library("plyr")  
  
  
theta <- fread("cache/theta.txt", data.table=FALSE)  
names(theta)[1] <- "seq"  
  
up5k <- read.table("cache/mt\_gene\_up5k.txt", header=TRUE)  
  
### define the subject file for theta values  
grc <- with(theta, GRanges(seqnames=seq, IRanges(start=Pos, end=Pos)))  
  
### define the query file for genomic feature  
grf <- with(up5k, GRanges(seqnames=seq, IRanges(start=start, end=end), geneid=geneid))  
   
### find overlaps between the two  
tb <- findOverlaps(query=grf, subject=grc)  
tb <- as.matrix(tb)  
   
out1 <- as.data.frame(grf[tb[,1]])  
out2 <- as.data.frame(grc[tb[,2]])  
### for each genomic feature, find the sites with non-missing data  
out <- cbind(out1, out2[, "start"])   
names(out)[ncol(out)] <- "pos"

## Intepret the theta results

#define unique identifier and merge with the thetas  
out$uid <- paste(out$seqnames, out$pos, sep="\_")  
theta$uid <- paste(theta$seq, theta$Pos, sep="\_")  
  
df <- merge(out, theta[, c(-1, -2)], by="uid")  
# for each upstream 5k region, how many theta values  
  
mx <- ddply(df, .(geneid), summarise,  
 Pairwise = mean(Pairwise, na.rm=TRUE),  
 thetaH = mean(thetaH, na.rm=TRUE),  
 nsites = length(uid))

## Intepret the theta results

Copy and paste everything above, and pack it into an R function:

get\_mean\_theta <- function(gf\_file="cache/mt\_gene\_up5k.txt"){  
 # gf\_file: gene feature file [chr, ="cache/mt\_gene\_up5k.txt"]  
   
 theta <- fread("cache/theta.txt", data.table=FALSE)  
 names(theta)[1] <- "seq"  
  
 up5k <- read.table(gf\_file, header=TRUE)  
  
 ### define the subject file for theta values  
 grc <- with(theta, GRanges(seqnames=seq, IRanges(start=Pos, end=Pos)))  
  
 ### define the query file for genomic feature  
 grf <- with(up5k, GRanges(seqnames=seq, IRanges(start=start, end=end), geneid=geneid))  
   
 ### find overlaps between the two  
 tb <- findOverlaps(query=grf, subject=grc)  
 tb <- as.matrix(tb)  
   
 out1 <- as.data.frame(grf[tb[,1]])  
 out2 <- as.data.frame(grc[tb[,2]])  
 ### for each genomic feature, find the sites with non-missing data  
 out <- cbind(out1, out2[, "start"])   
 names(out)[ncol(out)] <- "pos"  
  
 #define unique identifier and merge with the thetas  
 out$uid <- paste(out$seqnames, out$pos, sep="\_")  
 theta$uid <- paste(theta$seq, theta$Pos, sep="\_")  
  
 df <- merge(out, theta[, c(-1, -2)], by="uid")  
 # for each upstream 5k region, how many theta values  
  
 mx <- ddply(df, .(geneid), summarise,  
 Pairwise = mean(Pairwise, na.rm=TRUE),  
 thetaH = mean(thetaH, na.rm=TRUE),  
 nsites = length(uid))  
 return(mx)  
}

## Plot the results

Run the customized R function

### apply the function  
up5k <- get\_mean\_theta(gf\_file="cache/mt\_gene\_up5k.txt")  
down5k <- get\_mean\_theta(gf\_file="cache/mt\_gene\_down5k.txt")  
gene <- get\_mean\_theta(gf\_file="cache/mt\_gene.txt")

And then plot the results:

library("ggplot2")  
  
up5k$feature <- "up 5k"  
down5k$feature <- "down 5k"  
gene$feature <- "genic"  
res <- rbind(up5k, down5k)  
res\_nongenic <- res  
res\_nongenic$feature <- "intergenic"  
res\_t <- rbind(res, gene, res\_nongenic)  
ggplot(res\_t, aes(x=feature, y=Pairwise, fill=feature)) +   
 geom\_violin(trim=FALSE)+  
 labs(title="Theta value", x="", y = "Log10 (theta)")+  
 geom\_boxplot(width=0.1, fill="white")+  
 scale\_fill\_brewer(palette="Blues") +   
 theme\_classic()

## Intepret the Fst results

get\_mean\_Fst <- function(gf\_file="cache/mt\_gene\_up5k.txt"){  
 # gf\_file: gene feature file [chr, ="cache/mt\_gene\_up5k.txt"]  
   
 fst <- fread("cache/fst\_win.txt", data.table=FALSE)  
 names(fst)[2] <- "seq"  
 up5k <- read.table(gf\_file, header=TRUE)  
 ### define the subject file for Fst values  
 grc\_f <- with(fst, GRanges(seqnames=seq, IRanges(start=chr, end=chr)))  
 ### define the query file for genomic feature  
 grf <- with(up5k, GRanges(seqnames=seq, IRanges(start=start, end=end), geneid=geneid))  
   
 ### find overlaps between the two  
 tb\_f <- findOverlaps(query=grf, subject=grc\_f)  
 tb\_f <- as.matrix(tb\_f)  
   
 out1\_f <- as.data.frame(grf[tb\_f[,1]])  
 out2\_f <- as.data.frame(grc[tb\_f[,2]])  
 ### for each genomic feature, find the sites with non-missing data  
 out\_f <- cbind(out1\_f, out2\_f[, "start"])   
 names(out\_f)[ncol(out\_f)] <- "pos"  
 #define unique identifier and merge with the Fst  
 out\_f$uid <- paste(out\_f$seqnames, out\_f$pos, sep="\_")  
 fst$uid <- paste(fst$seq, fst$chr, sep="\_")  
 df\_f <- merge(out\_f, fst[, c(-1, -2, -3, -4)], by="uid")  
 # for each upstream 5k region, how many Fst values  
 names(df\_f)[ncol(df\_f)] <- "fst"  
   
 mx\_f <- ddply(df\_f, .(geneid), summarise,  
 Fst = mean(fst, na.rm=TRUE),  
 nsites = length(uid))  
 return(mx\_f)  
}

## Plot the results

Run the customized R function

### apply the function  
up5k\_f <- get\_mean\_Fst(gf\_file="cache/mt\_gene\_up5k.txt")  
down5k\_f <- get\_mean\_Fst(gf\_file="cache/mt\_gene\_down5k.txt")  
gene\_f <- get\_mean\_Fst(gf\_file="cache/mt\_gene.txt")

And then plot the results:

library("ggplot2")  
up5k\_f$feature <- "up 5k"  
down5k\_f$feature <- "down 5k"  
gene\_f$feature <- "genic"  
res\_f <- rbind(up5k\_f, down5k\_f)  
res\_nongenic\_f <- res\_f  
res\_nongenic\_f$feature <- "intergenic"  
res\_t\_f <- rbind(res\_f, gene\_f, res\_nongenic\_f)  
ggplot(res\_t\_f, aes(x=feature, y=Fst, fill=feature)) +   
 geom\_violin(trim=FALSE)+  
 labs(title="Fst distribution", x="", y = "Fst")+  
 geom\_boxplot(width=0.1, fill="white")+  
 scale\_fill\_brewer(palette="greens") +   
 theme\_classic()