practical 4: Population structure

Jerome Goudet and Bruce Weir

2021-07-16

For this practical, you will need to install a new package `JGTeach" from github:

```
library(gaston)
```

```
library(devtools)
install_github("jgx65/JGTeach")
```

```
library(hierfstat)
library("JGTeach")
```

Drift

1. We will start by doing *in silico* Buri's drosophila cages experiment. We will generate using the `JGTeach::drift` function the allelic frequency trajectories for a number of replicates, corresponding to the different cages:

```
dum<-drift(nind=100,p0=0.5,nrep=1000,ngen=100,PlotIt = TRUE)
```

- Describe the figure produced
- set the number of individuals to 10; to 1000. How does population size affects the drifting process?
- Set the number of individuals to 100 and the number of generation to 1000. Take time slices (generations 1, 2, 3, 5, 10, 50, 100,1000) to look at the distribution of the replicates allele frequencies over time.

```
x<-drift(nind=100,ngen=1000)
gens<-c(1,2,3,5,10,20,50,100,1000)

par(mfrow=c(3,3))
for (i in gens)
hist(x[i,],breaks=0:50/50,main=paste("generation:", i),xlab="Freq",ylab="")
par(mfrow=c(1,1))
```

- Describe the resulting figure and compare it to Buri's results

2 [optional]. We will use the `drift` function to explore some properties of drift, namely the probability and time to fixation of an allele.

Theory tells us that the probability of fixation of an allele is its frequency, and that time to fixation is

$$\frac{4N(1-p)\ln(1-p)}{p}$$

which, for a frequency of 0.5, corresponds to $2.7N$ generations

- [optional] simulate $10,000$ replicates of a population of 50 diploid individuals starting with a frequency $p0 = 0.5$. The expected probability of fixation is 0.5, and it should take on average 135 generations to achieve fixation.

```
x1<-drift(nind=50,ngen=1000,nrep=10000)
par(mfrow=c(3,3))
for (i in gens)
  hist(x1[i,],breaks=seq(0,1,0.01),main=paste("generation:", i),xlab="Freq",ylab="")
par(mfrow=c(1,1))
```

- [optional] extract the time to fixation of the different replicates, from which you will extract both the probability and mean time to fixation.

```
 # distribution of fixation times
dist.tfix<-apply(x1,2,function(y) which(y>=1.0)[1])
# fixation probability
(pfix<-sum(!is.na(dist.tfix))/10000)
# mean time to fixation
(tfix<-mean(dist.tfix,na.rm=T))
```

- [optional] Make a histogram of the time to fixation and describe it

```
hist(dist.tfix, main="Distribution of time to fixation, p0=0.50, N=50",xlab="Generations",ylab=""
)
abline(v=tfix, col="red",lwd=2)
```

Individual populations Betas

3. Simulate using `hierfstat::sim.genot.t` 3 populations evolving independently one from another, in order to check the approximation $\beta_i \approx \frac{t}{2N_i}$. The 3 populations have diploid size $N_1 = 100$, $N_2 = 1000$ and $N_3 = 10000$ respectively. after 50 generations, we expect $\beta_{1,2,3}$ to be approximately 0.25, 0.025, 0.0025

```
# 50 loci with 20 alleles each,
dat<-sim.genot.t(size=100,nbal=20,N=c(100,1000,10000),
                 nbloc=50,mig=0.0,t=50)

# the betas estimates population specific Fst from a
# fstat format data frame assuming random mating
betas(dat,nboot=100)$ci
```

- Check whether the approximation holds after 200 generations

```
# 50 loci with 20 alleles each,
dat<-sim.genot.t(size=100,nbal=20,N=c(100,1000,10000),
                 nbloc=50,mig=0.0,t=200)

# the betas estimates population specific Fst from a
# fsat format data frame assuming random mating
betas(dat,nboot=100)$ci
#expectation
200/c(200,2000,20000)
```

- [optional] Imagine a way to do this simulation with `ms`.

```
# assumes mu=1e-8, r=1e-8, N0=1e+5, nbp=1e+5,
# simulate 10 replicates, 100 diploid individuals from each pop
#
# pop 4 will be the ancestral population with N4=N0=1e+5
# Only pops 1-3 are sampled, and there is no migration:
# -I 4 200 200 200 0 0
# pop1 is a 1000th of anc pop, pop2 a 100th and pop 3 a 10th:
# -n 1 0.001 -n 2 0.01 -n 3 0.1
# split occured (backward) 50 generation ago
# in unit of 4* Anc pop size ->50/4/100000=0.000125:
# -ej 0.000125 1 4 -ej 0.000125 2 4 -ej 0.000125 3 4
```

```
ms 600 10 -t 400 -r 400 100000 -I 4 200 200 200 0 0 -n 1 0.001 -n 2 0.01 -n 3 0.1
-ej 0.000125 1 4 -ej 0.000125 2 4 -ej 0.000125 3 4 > 3popsdrift.txt
```

```
bed<-ms2bed("3popsdrift.txt")
# expectation is (50/2/c(100,1000,10000)),
# but works well for t<0.2N only
fst.dosage(bed,pop=rep(1:3,each=100))
50/c(200,2000,20000)
```

2 populations system

Here, we are reproducing figs 1. and 3 of Weir and Goudet (2017) (https://www.genetics.org/content/206/4/2085).

The two populations system presented during the lecture and given in eq 5 of the paper has been implemented in `JGTeach::thet.bet.2pops`.

4 [optional]. Have a look at the help page, and run the examples (fig 1 of the paper)

```
# reproduces fig 1 in Weir and Goudet (2017)
#top row
first<-thet.bet.2pops(mu=0.0,m2=0.0,m1=0.0,n1=10000,n2=100,ngen=10000)
#middle row
second<-thet.bet.2pops(mu=0.001,m2=0.0,m1=0.0,n1=10000,n2=100,ngen=10000)
#bottom row
third<-thet.bet.2pops(mu=0.001,m2=0.0,m1=0.01,n1=10000,n2=100,ngen=10000)
```

5 [optional]. Next you will show that the estimates matches the expected values. For this, we will make use of the hierfstat function `sim.genot.metapop.t` to generate genetic data according to this model

```
#library(hierfstat)
gens<-1:10*100
ngen<-gens[length(gens)]
#generate data
mig.mat2<-matrix(c(.99,0.01,0,1),nrow=2,byrow=TRUE)
mig.mat2
mutrate<-10^{-3} #10^{-6} initial value

# As sim.genot.metapop.t only outputs genotypes from the
# last generation, necessary to run the function
# with several end time points. This is the essence of
# using lapply to a here

#  x<-lapply(gens,function(y){
#  sim.genot.metapop.t(t=y,nbal=20,nbpop=2,N=c(10000,100),
#                      mig=mig.mat2,nbloc=1000,mut=mutrate)})

# estimate betas for data sets
# beta.x<-lapply(x,betas,nboot=1000)
```

This may take some time, so a data file fig3.RData (https://www2.unil.ch/popgen/teaching/SISG17/fig3.RData) containing the results is available from the website, you should download it.

```
load("fig3.RData")
betas.ci<-lapply(beta.x,function(x) x$ci)
#expected values

etb1<-thet.bet.2pops(mu=mutrate,n1=10000,n2=100,m1=0.01,m2=0,ngen=ngen,plotit=FALSE)

gens<-1:10*100
ngen<-gens[length(gens)]

#create the plot
plot(1:ngen,etb1$Be[,1],type="l",ylim=range(etb1$Be,betas.ci),col="red",lwd=2,
     xlab="Generations",ylab=expression(beta))
lines(1:ngen,etb1$Be[,2],col="blue",lwd=2)
lines(1:ngen,etb1$Be[,3],lwd=2)
abline(h=0)
segments(gens,unlist(lapply(betas.ci,function(x) x[1,2])),gens,
         unlist(lapply(betas.ci,function(x) x[2,2])),lwd=2,col="blue")
segments(gens,unlist(lapply(betas.ci,function(x) x[1,1])),gens,
         unlist(lapply(betas.ci,function(x) x[2,1])),lwd=2,col="red")
points(gens,unlist(lapply(beta.x,function(x) x$betaiovl[1])),
       col="red",cex=1.5,pch=16)
points(gens,unlist(lapply(beta.x,function(x) x$betaiovl[2])),
       col="blue",cex=1.5,pch=16)
#beta_w added too
points(gens,unlist(lapply(beta.x,function(x) mean(x$betaiovl))),
       col="black",cex=1.5,pch=16)
title("2 populations with migration model. \n        N1=10'000; N2=100; m1=0.01; m2=0.0")
```

Population specific Fsts from the 1000 genomes

We will now explore the first 20 megabases of chromosome 22 from samples of the 1000 genome project and will find that negative population specific F_{ST}^{i} are not uncommon.

6. Load chromosome 22 fragment VCF file from the 1000 genome into R (see practical 1) and the samples description

```
ch22<-read.VCF("chr22_Mb0_20.recode.vcf.gz")
samp.desc.url<-"https://www2.unil.ch/popgen/teaching/SISG18/integrated_call_samples_v3.20130502.ALL.panel"
samp.desc<-read.table(samp.desc.url,header=TRUE,stringsAsFactors = TRUE)
```

- Describe object `samp.desc` : Print a table of the number of samples per super_pop and pop

```
with(samp.desc,table(pop,super_pop))
```

- Verify that samples are the same and in the same order in `ch22` and `samp.desc` .

```
all.equal(ch22@ped$id,as.character(samp.desc$sample),check.attributes = FALSE)

# if not (in different order and or some missing) :
# use match(ch22@ped$id,samp.desc$sample)
```

- Estimate population and continent F_{ST}s from this dataset (it takes some time). You will have to download the Allele sharing (matching) file (https://www2.unil.ch/popgen/teaching/SISGData/matching.ch22.RDS) first

```
# From dosage data:
# fst.ch22.pop<-fst.dosage(ch22,pop=samp.desc$pop)
# fst.ch22.cont<-fst.dosage(ch22,pop=samp.desc$super_pop)

# More efficiently from matching data
# Matching.ch22<-matching(ch22)
# takes some time, so saved in matching.ch22.RDS available from # website

Matching.ch22<-readRDS("matching.ch22.RDS")
fs.ch22.pop<-fs.dosage(Matching.ch22,pop=samp.desc$pop,matching=TRUE)
fs.ch22.cont<-fs.dosage(Matching.ch22,pop=samp.desc$super_pop,matching=TRUE)
fs.ch22.pop$Fs[2,]
fs.ch22.cont$Fs[2,]
```

- There is a `plot` function associated to `fs.dosage` . Produce the corresponding plot after having ordeered the population by continent and discuss the results in the light of what you know about human demographic history: the top left panel show individual inbreeding coefficients per population, relative to the mean kinship in their population. The right column panels show F_{ST}^{XY} (or β^{XY}) on top and population specific F_{ST} at the bottom. Last, the bottom left panel shows pairwise F_{ST}. Do you see a similarity between F_{ST}^{XY} and the heatmap of K_{AS} kinship?

```
#extract popnames in the correct order
pnames<-unlist(lapply(strsplit(with(samp.desc,
      levels(factor(paste(super_pop,pop,sep=":")))),":"),function(x) x[[2]]))
#forces factor to take  this order
pop<-with(samp.desc,factor(pop,levels=pnames))
fs.ch22.pop<-fs.dosage(Matching.ch22,pop=pop,matching=TRUE)
coul<-rep(c("orange","gold","red","green","blue","purple"),c(2,5,4,5,5,5))
plot(fs.ch22.pop,las=2,cex.axis=0.5,col=coul)
```

PCA

7. In this part, we will learn how to conduct Principal Component Analyses (PCA) on a genomic data set, [optionally] look at the different flavors of PCA, and get a feel for how to interpret the results

- Start by simulating the genotypes of 250 individuals, 50 from each of 5 populations, at 1000 bi-allelic loci, assuming an island model at equilibrium between migration, mutation and drift. Assume each island is made of 1000 individuals, with $m = 0.003$ between them. For this, use the `sim.genot` function of the `hierfstat` package.

```
np<-5
dat5<-sim.genot(nbal=2,nbloc=1000,nbpop=np,mig=0.003)
```

- Convert the data into dosage format using the `biall2dos` function

```
dos<-biall2dos(dat5[,-1])
```

- Transform the dosage matrix into matrix X, as done in Patterson et al. (2006). You may find the `scale` function useful for this.

```
ps<-colMeans(dos)/2
sdp<-(ps*(1-ps))^(0.5)
#first, filter out the fixed loci
nfixed<-which(ps>0 & ps <1)
dosnf<-dos[,nfixed]

X<-scale(dosnf)

# or
# X<-sweep(dosnf,2,ps[nfixed],FUN="-")
# X<-sweep(X,2,sdp[nfixed],FUN="/")
```

- Calculate XX' using function `tcrossprod`

```
XXT<-tcrossprod(X)
```

- Obtain the eigen value decomposition of XX' using the `eigen` function

```
eigx<-eigen(XXT)
```

- Obtain the individual's coordinates $UV^{1/2}$

```
ind.coord<-sweep(eigx$vectors,2,eigx$values^0.5,FUN="*")
```

- Compare the results you obtain to what is obtained using the `prcomp` function on matrix X

```
prx<-prcomp(X)
#check equality of the absolute values of the  PCs

all.equal(abs(matrix(prx$x)),abs(matrix(ind.coord)))
```

- In a four panel graphic windows, produce the following 4 plots:
  - A scree plot of the first 20 eigen values, expressed as proportion of the sum of all eigenvalues
  - A plot of PC2 against PC1, using a different color for each population
  - A plot of PC4 against PC3
  - A plot of PC6 against PC5

```
par(mfrow=c(2,2))
plot((eigx$values/sum(eigx$values))[1:20],type="h",
     main="screeplot",xlab="Eigen values",ylab="")
plot(prx$x[,1:2],col=rep(1:np,each=50),pch=16)
plot(prx$x[,3:4],col=rep(1:np,each=50),pch=16)
plot(prx$x[,5:6],col=rep(1:np,each=50),pch=16)

par(mfrow=c(1,1))
```

- Describe what you see: How many eigen values stand out? are samples grouped according to their population of origin? If so along which axes? How many PCs are necessary to describe the structure of this dataset?
- Produce a parallel coordinate plot, where the x-axis correspond to the first 5 PCA axes, and the y axis shows the coordinates of each individuals along the 5 axes, drawn as a line with color corresponding to the population in which the individual has been sampled

```
colpca<-rep(1:np,each=50)
ns<-5
plot(1:ns,prx$x[1,1:ns],col=colpca[1],type="l",ylim=range(prx$x[,1:ns]),
     xlab="Axis",ylab="coord.",main="par. coord. plot")
for (i in 2:nrow(prx$x)) lines(1:ns,prx$x[i,1:ns],col=colpca[i])
```

8 [optional]. Redo this analysis on the matrix of dosages centered but not scale to $sd = 1$. Conclusions?

```
X<-scale(dosnf,scale=FALSE)
XXT<-tcrossprod(X)
eigx<-eigen(XXT)
ind.coord<-sweep(eigx$vectors,2,eigx$values^0.5,FUN="*")

prx<-prcomp(X)

all.equal(abs(matrix(prx$x)),abs(matrix(ind.coord)))


par(mfrow=c(2,2))
plot((eigx$values/sum(eigx$values))[1:20],type="h",
     main="screeplot",xlab="Eigen values",ylab="")
plot(prx$x[,1:2],col=rep(1:np,each=50),pch=16)
plot(prx$x[,3:4],col=rep(1:np,each=50),pch=16)
plot(prx$x[,5:6],col=rep(1:np,each=50),pch=16)

par(mfrow=c(1,1))
```

9 [optional]. Instead of using XX' for eigenvalue decomposition, use the kinship matrix K_{AS} obtained from the allele sharing matrix, you might want to use the `beta.dosage` function for this. Conclusions?

```
Kas<-beta.dosage(dos)
eigKas<-eigen(Kas)

ind.coord<-sweep(eigKas$vectors,2,eigKas$values^0.5,FUN="*")

par(mfrow=c(2,2))
plot((eigKas$values/sum(eigKas$values))[1:20],type="h",
     main="screeplot",xlab="Eigen values",ylab="")
plot(ind.coord[,1:2],col=rep(1:np,each=50),pch=16)
plot(ind.coord[,3:4],col=rep(1:np,each=50),pch=16)
plot(ind.coord[,5:6],col=rep(1:np,each=50),pch=16)

par(mfrow=c(1,1))
```

10 [optional]. Simulate a new genotype data set, this time with only three populations. Rerun step 2-9 above. Conclusions?

```
np<-3
dat3<-sim.genot(nbal=2,nbloc=1000,nbpop=np,mig=0.003)
dos<-biall2dos(dat3[,-1])
ps<-colMeans(dos)/2
sdp<-(ps*(1-ps))^(0.5)
#first, filter out the fixed loci
nfixed<-which(ps>0 & ps <1)
dosnf<-dos[,nfixed]

X<-scale(dosnf)
prx<-prcomp(X)
par(mfrow=c(2,2))
plot((prx$sdev^2/sum(prx$sdev^2))[1:20],type="h",
     main="screeplot",xlab="Eigen values",ylab="")
plot(prx$x[,1:2],col=rep(1:np,each=50),pch=16)
plot(prx$x[,3:4],col=rep(1:np,each=50),pch=16)
plot(prx$x[,5:6],col=rep(1:np,each=50),pch=16)

par(mfrow=c(1,1))
```

11 [optional]. You might want to experiment with higher migration rate in the simulated data set, and observe the effect it has. Conclusions

```
np<-5
#migration 3 times larger

dat5<-sim.genot(nbal=2,nbloc=1000,nbpop=np,mig=0.01)
dos<-biall2dos(dat5[,-1])
ps<-colMeans(dos)/2
sdp<-(ps*(1-ps))^(0.5)
#first, filter out the fixed loci
nfixed<-which(ps>0 & ps <1)
dosnf<-dos[,nfixed]

X<-scale(dosnf)
prx<-prcomp(X)
par(mfrow=c(2,2))
plot((prx$sdev^2/sum(prx$sdev^2))[1:20],type="h",
     main="screeplot",xlab="Eigen values",ylab="")
plot(prx$x[,1:2],col=rep(1:np,each=50),pch=16)
plot(prx$x[,3:4],col=rep(1:np,each=50),pch=16)
plot(prx$x[,5:6],col=rep(1:np,each=50),pch=16)

par(mfrow=c(1,1))

colpca<-rep(1:np,each=50)
ns<-5
plot(1:ns,prx$x[1,1:ns],col=colpca[1],type="l",
     ylim=range(prx$x[,1:ns]),xlab="Axis",
     ylab="coord.",main="par. coord. plot")
for (i in 2:nrow(prx$x)) lines(1:ns,prx$x[i,1:ns],col=colpca[i])
```

12 [optional]. Rather than an island model of population structure, you might want to simulate a stepping stone in one dimension, using the `sim.genot.metapop.t` function. Assume 8 demes, connected by migration with nearest neighbors only, and with 5% migration between neighbors. Redo the PCA on the data set. Interpret the figure.

```
np<-8
nl<-1000
t<-1000
M<-matrix(0,ncol=np,nrow=np)
diag(M[-1,-np])<-0.05
diag(M[-np,-1])<-0.05
diag(M)<-1-rowSums(M,na.rm=TRUE)

ss1d<-sim.genot.metapop.t(nbal=2,nbpop=np,nbloc=nl,mig=M,t=t)
dos<-biall2dos(ss1d[,-1])

ps<-colMeans(dos)/2
sdp<-(ps*(1-ps))^(0.5)
#first, filter out the fixed loci
nfixed<-which(ps>0 & ps <1)
dosnf<-dos[,nfixed]

X<-scale(dosnf)
prx<-prcomp(X)
par(mfrow=c(2,2))
plot((prx$sdev^2/sum(prx$sdev^2))[1:20],type="h",
     main="screeplot",xlab="Eigen values",ylab="")
plot(prx$x[,1:2],col=rep(1:np,each=50),pch=16)
plot(prx$x[,3:4],col=rep(1:np,each=50),pch=16)
plot(prx$x[,5:6],col=rep(1:np,each=50),pch=16)

par(mfrow=c(1,1))
```