Appendix 1

John Wares & Paula Pappalardo

09 septiembre, 2015

R code to simulate populations and estimate genetic diversity parameters

We simulated populations evolving under the coalescent using Hudson's *ms* software (Hudson 2002). We called the *ms* program using the *gap* (Zhao 2015) package in R (R Core Team 2015). We used the package *PopGenome* to estimate haplotype diversity and check Tajima'ss.

We simulated three populations of 1000 individuals, with *theta* equal to two, ten and twenty. From each of the populations we sampled without replacement to simulate our "field samples" of known sampling size (2,4,8,16,32,64 and 128 individuals), we repeated the sampling 100 times to have replicates within each sampling size. For each sample we estimated the number of haplotypes and the number of segregating sites.

NOTE: the original R files are available upon request from the authors.

Simulating source populations

Below the code used to simulate populations and estimate summary statistics for each of the populations.

```
library(gap)
library(PopGenome)
library(knitr)

# -----gap----
# we call ms from R
# with -t we set the mutation rate theta
# we simulated 1 population of 1000 individuals

# Simulating our source populations
theta2NoGrowth <- system("ms 1000 1 -t 2", intern=TRUE)
theta10NoGrowth <- system("ms 1000 1 -t 10", intern=TRUE)
theta20NoGrowth <- system("ms 1000 1 -t 20", intern=TRUE)

# ONLY ONCE: write the outputs so we can read them using PopGenome

#write(theta2NoGrowth,"theta2NoGrowth.out")
#write(theta10NoGrowth,"theta10NoGrowth.out")
#write(theta20NoGrowth,"theta20NoGrowth.out")

# reading with PopGenome to calculate H and Tajima's D for each of the four populations
readMS("theta2NoGrowth.out")->popgen.t2ng
readMS("theta10NoGrowth.out")->popgen.t10ng
readMS("theta20NoGrowth.out")->popgen.t20ng

# run F_ST stats and check haplotype diversity
F_ST.stats(popgen.t2ng)->popgen.t2ng
F_ST.stats(popgen.t10ng)->popgen.t10ng
```

```
F_ST.stats(popgen.t20ng)->popgen.t20ng

# get haplotype diversity for each source population
unlist(popgen.t2ng@region.stats@haplotype.diversity)->hapDiv.t2ng
unlist(popgen.t10ng@region.stats@haplotype.diversity)->hapDiv.t10ng
unlist(popgen.t20ng@region.stats@haplotype.diversity)->hapDiv.t20ng

# make a vector with the haplotype diversities for each population
hapDiv<-c(hapDiv.t2ng,hapDiv.t10ng,hapDiv.t20ng)
hapDiv

# run neutrality.stats to check Tajima's D
neutrality.stats(popgen.t2ng)->popgen.t2ng
neutrality.stats(popgen.t10ng)->popgen.t10ng
neutrality.stats(popgen.t20ng)->popgen.t20ng

Taj.t2ng<-popgen.t10ng@Tajima.D
Taj.t10ng<-popgen.t10ng@Tajima.D
Taj.t20ng<-popgen.t20ng@Tajima.D

# make a vector with the Tajima's D value for each population
tajima<-c(Taj.t2ng,Taj.t10ng,Taj.t20ng)
tajima

# make a summary table
thetas<-c(2,10,,20)
pops<-c("Population 1", "Population 2", "Population 3")
newhap<-round(hapDiv,2)
newtaj<-round(tajima,2)
sumTable<-cbind(pops,thetas,newhap,newtaj)
colnames(sumTable)<-c("Population","Theta","Haplotype diversity","Tajima's D")
kable(sumTable)
```

Now that we have our 3 populations and we have H and Tajima's D for those, we need to take our "field samples".

Taking "field samples"

From the simulated populations we took "field samples" using the code below. Basically, we used the matrix of the variables sites for the 1000 individuals and took samples without replacement of 2,4,8,16,32,64 and 128 individuals. We repeated the sampling 100 times for each combination of sampling size and theta.

```
library(gap)
library(PopGenome)

# we read the files that were simulated only once
pop1 <- read.ms.output("theta2NoGrowth.out",is.file=T)
pop2 <- read.ms.output("theta10NoGrowth.out",is.file=T)
pop3 <- read.ms.output("theta20NoGrowth.out",is.file=T)

# --------sampling from each population---------
# vector with all the populations for loops
```

```
pop <- list(pop1,pop2,pop3)

# vector with the sizes we want to sample
popsizes <- c(2,4,8,16,32,64,128)

# we extract the gametes matrix for each population
# from there we sample and we put fill the list called "samples"
# the result for each population in saved in a list called "populations"

populations <- list() #empty list to put results
set.seed(123) #set seed for random sampling

# loop in each population to take the field samples, 100 replicates for each sampling size
for (i in 1:length(pop)){
  ourpop <- pop[[i]]  # pick a population
  namepop <- paste("popOrigen",i,sep="")
  mat <- t(ourpop$gametes[[1]])  # transpose matrix to have individuals as rows
  samples <- list()  # empty list to put the sampling of each sample size
    for (j in popsizes){
        name <- paste("sampleSize",j,sep="")
        replicates <- list()  # empty list to put each replicate (n=100)
        for (k in 1:100){
        namerep <- paste("replicate",k,sep="")
        rep <- mat[sample(nrow(mat),size=j,replace=FALSE),]  # take the sample
        rep -> replicates[[namerep]]
        }
        replicates -> samples[[name]]
      }
  samples->populations[[namepop]]
}
```

The object **populations** has 3 components (one for each population), each one include 7 lists (one for each sampling size), and within each of the sampling sizes we have a list with the 100 matrix sampled.

Calculating haplotypes

Now we are going to calculate the number of haplotypes and number of segregating sites for each of the replicates in each of the samples in each of the 3 populations. The loops may take a while depending the computer.

```
# the haplotype number can be extracted by doing unique() of the gametes matrix
hapResults <- list()
for (i in 1:length(pop)){
  popnow <- populations[[i]]
  namepop <- paste("popOrigen",i,sep="")
  samples <- list()
  for (j in 1:length(popsizes)){
    namesample <- paste("sampleSize",j,sep="")
    samplenow <- popnow[[j]]
    for (k in 1:100){
      sapply(samplenow,function(x) nrow(unique(x))) -> result
      unlist(result) -> replicates
```

```
      }
    replicates -> samples[[namesample]]
    }
  samples -> hapResults[[namepop]]
}
# now hapResults is a list of vectors
# with the haplotypes number for each population, in each sample size
```

Since this process is really time consuming, we want to save our results in a dataframe.

```
# we want to loop in the vector "hapresults"
# and for each population extract the number of haplotypese

library(reshape)

# create dataframe to hold the data
zerodata <- data.frame(Pop=NA,Theta=NA,variable=NA,value=NA)
thetas <- c("two","ten","twenty")

# loop through populations and rbind data
for (i in 1:length(pop)){
  hapResults[[i]] -> pophaps
  haps <- data.frame(do.call(rbind, pophaps))
  as.data.frame(t(haps)) -> datahaps
  datahaps$Theta <- thetas[i]
  datahaps$Pop <- i
  newdata <- melt(datahaps,id=c("Pop","Theta"))
  rbind(zerodata,newdata) -> zerodata
}

# organize and save data
zerodata[complete.cases(zerodata$Pop),]->numHaps
as.factor(numHaps$variable)->numHaps$variable
names(numHaps)<-c("Pop","Theta","samplingSize","n.haplotypes")
levels(numHaps$samplingSize)<-c("2","4","8","16","32","64","128")

# run only once
#write.csv(numHaps,"numberHaplotypes.csv")
```

The final product is the .csv file with the dataframe "numberHaplotypes" that for each row has information on theta, sampling size and number of haplotypes.

Calculating segregating sites

We are going to do a similar procedure to estimate the number of segregating sites. We first created a function to estimate the number of segregating sites from the gametes matrix provided by the MS simulation. And then we applied the custome function (called "estimate.segSites") in a loop over our samples.

```
# define and load the function
estimate.segSites <- function (myMatrix){
  # this function takes a matrix of gametes from a MS simulation
  # and returns a numeric value, the number of segregating sites
```

```
  answers <- rep(NA,(nrow(myMatrix)-1))
  sites <- rep(NA,ncol(myMatrix))
    for (j in 1:ncol(myMatrix)){
      for (i in 1:(nrow(myMatrix)-1)){
        (myMatrix[i+1,j] == myMatrix[1,j]) -> answers[i]
      }
      if (all(answers) == TRUE) {0->sites[j]}
      else {1 -> sites[j]}
      }
    sum(sites,na.rm=T) -> segSites
    return(segSites)
  }

# use the function in a loop to get segreating sites
segResults <- list()
for(i in 1:length(pop)){
  populations[[i]] -> popnow
  namepop <- paste("popOrigen",i,sep="")
  samples <- list()
  for (j in 1:length(popsizes)){
    namesample <- paste("sampleSize",j,sep="")
    popnow[[j]] -> samplenow
    sapply(samplenow,function(x) estimate.segSites(x)) -> result
    result -> samples[[namesample]]
    }
  samples -> segResults[[namepop]]
}
# now segResults is a list of 3 lists (one for each population)
# within each there is a list of 7 vectors, one for each sample size
# the vector length is 100, with each component of the vector
# holding the number of segregating sites for that replicate.
```

To save our results in a dataframe:

```
# we want to loop in "segResults" and for each population extract
# the number of segregating sites and put all in the same dataframe
library(reshape)

# create dataframe to hold the data
zerodata <- data.frame(Pop=NA,Theta=NA,variable=NA,value=NA)
thetas <- c("two","ten","twenty")

# loop through populations and rbind data
for (i in 1:length(pop)){
  segResults[[i]] -> popsegs
  segs <- data.frame(do.call(rbind, popsegs))
  as.data.frame(t(segs)) -> dataseg
  dataseg$Theta <- thetas[i]
  dataseg$Pop <- i
  newdata <- melt(dataseg,id=c("Pop","Theta"))
  rbind(zerodata,newdata) -> zerodata
}
```

```r
# organize and save data
zerodata[complete.cases(zerodata$Pop),] -> numSegSites
as.factor(numSegSites$variable) -> numSegSites$variable
names(numSegSites) <- c("Pop","Theta","samplingSize","n.seg.sites")
levels(numSegSites$samplingSize) <- c("2","4","8","16","32","64","128")

# run only once
#write.csv(numSegSites,"numberSegSites.csv")
```

The final product is the .csv file with the dataframe "numSegSites" that for each row has information on theta, sampling size and number of segregating sites.

References

Hudson, R. R. 2002. "Generating Samples Under a Wright-Fisher Neutral Model of Genetic Variation." Journal Article. *Bioinformatics* 18 (2): 337–8. http://www.ncbi.nlm.nih.gov/pubmed/11847089.

R Core Team. 2015. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org/.

Zhao, J. H. 2015. *Gap: Genetic Analysis Package*. http://cran.r-project.org/package=gap.