Degraded DNA and population subsampling simulations

Anthony Schultz + Kasha Strickland

24/09/2020

Simulated degraded DNA

This document will describe the process, and provide code, to simulate degraded DNA typical of non-invasive genetic sampling approaches. This approach requires an existing genotype dataset for your study species/population, and some derived or estimated parameters of the reductions in loci callrates and SNP panel (number of loci) typically seen in data derived from the non-invasive sampling technique you select.

This code assumes that your data is in an acceptable dartR format (please see the package documentation for formatting details), and also that each individual with a genotype in your data also has a single location point of some kind (point of capture, centre of activity etc.). Finally, this script requires the observed (i.e. true) internal relatedness measures for each individual in your population. We recommend that you calculate these using your genotype data before proceeding with this script. Please see the formatting of the supplied example files (Location_Points.csv, Observed_IR.csv, Genotypes.csv) for guidance on formatting your own data.

The R script provided here has been tested in R Version 4.0.2. with up-to-date packages, although this script will not be updated for future versions of packages etc.

Thanks - AJS

Packages

Running certain packages required for this script requires the installation of **rtools**. If you are using R v4.0 or above, you will need to install rtools40, the instructions for this can be found at https://cran.r-project.org/bin/windows/Rtools/

You may also need to install **SNPrelate** manually in order for dartR to work. SNPrelate is not avaialable on CRAN, but can be downloaded from Bioconductor:

https://www.bioconductor.org/packages/release/bioc/html/SNPRelate.html

You will also need to source **GENHET**, an R Function created by Aurelie Coulon, which can be downloaded from: http://www.aureliecoulon.net/research/ac-computer-programs.html

Packages to install/functions to source:

```
library(raster)

library(sf)

library(spatialEco)

library(spatstat)
```

```
library(sp)

library(maptools)

library(SOAR)

library(rgeos)

library(SDraw)

library(BalancedSampling)

library(poppr)

library(dartR)

library(dplyr)

library(StAMPP)

library(hierfstat)

library(gtools)

library(Matrix)

library(data.table)

library(PopGenReport)

library(truncnorm)

source("GENHETv3.1.R")
```

Load and format required files

Load and check the structure of required files (observed internal relatedness, spatial locations points). Convert spatial location points into a Spatial Points object, and set appropriate coordinate reference system. Here we convert to EPSG32756 (UTM 56S), as koala tracking points are in Australia.

```
observedIR<-read.csv("observedIR.csv")
str(observedIR)
locations<-read.csv("Tracking_data.csv")
str(locations)

spat.pts <- SpatialPoints(locations[,2:3])
projection(spat.pts)<- crs("+init=epsg:32756")

spat.pts.df<-SpatialPointsDataFrame(spat.pts,locations[c("Name")])
k2ks<-as.data.frame(spat.pts.df)
```

Assign animal ID and convert to Spatial Points Data Frame

```
spat.pts.df<-SpatialPointsDataFrame(spat.pts,locations[c("Name")])
k2ks<-as.data.frame(spat.pts.df)
```

Read in the genotype file. This must be in a dartR compatible format (either 1 row or 2 row format), and also requires a metadata file. For the metadata file, "id" and "pop" are fixed columns, the rest are user defined. See the package documentation for more details.

```
gl1 <- gl.read.dart(filename = "genotypes_file.csv",covfilename = "idmeta.csv")
```

Remove individuals from genotype file that we do not have spatial data for (i.e. don't occur in locations)

```
gl1 <- gl1[gl1$ind.names%in%locations$Name]
ids<-indNames(gl1)
```

Read in genotypes file as a Data Frame so that we can degrade the data to match that of non-invasively derived genotypes. Extract column names from dartR format, and exclude individuals for which we don't have spatial data, as above. Check structure to ensure that this has worked.

```
double_raw<-read.csv( "genotypes_file.csv" , colClasses = "character")
colnames(double_raw)<-double_raw[5,]
db<-double_raw[colnames(double_raw)%in%locations$Name]
db<-cbind(double_raw[,1:17],db)
str(db)
```

Create a new copy of db - we will need the top rows from db later in order to reform a dartR compatible file post-degradation

```
db2<-db
```

Select SNP panel

Create SNP panel (number of loci) for all simulations. This will be based on your own estimates / data. Here we used 1300 loci, based on Schultz et al (2018) Fresh is Best: Accurate SNP genotyping from koala scats.

```
AlleleNumber<-strsplit(db2$AlleleID,"\\:")
AlleleNumber<-unlist(lapply(AlleleNumber, "[[",1))
all.dups<-AlleleNumber[6:length(AlleleNumber)]
al.index<-rep(seq(1:(length(all.dups)/2)),each=2)
al.index<-c(rep("*",5),al.index)
db2$alleleindex<-al.index
allele.ss<-sample(unique(al.index[6:length(AlleleNumber)]),1300,replace=FALSE)
db2a<-subset(db2,db2$alleleindex%in%allele.ss)
db2a<-rbind(db2[1:5,],db2a)
db3<-db2a
```

Degrade genotype data

Set call-rate parameters for DNA degradation, as expected due to non-invasive sampling. Here we use parameters from experimental ageing of koala scats in Schultz et al (2018) Fresh is Best: Accurate SNP genotyping from koala scats.

```
minCR=0.43
maxCR=1
meanCR=0.62
sdevcR=0.13
```

Begin simulations

Define vectors for simulation outputs

```
He=vector()
Fis=vector()
I=vector()
I_se=vector()
IRdfs=vector()
sp_pval=list()
sp_rvals=list()
sp_CIs=list()
sp_TF=list()
```

Define population subsampling parameters. Here we sample between 40 and 420 koalas, in increments of 20, with 100 repeat simulations for each sample size.

```
ss<-rep(seq(40,420,by=20),each=100)
reps<-rep(seq(1,100),length.out=length(ss))
nameslist<-ss # this makes a list of the sequence numbers to apply as row
              #names to the final dataset
```

Now begin the simulation loop. This code below has been annotated in-line to prevent breaking up the code the entire loop.

```
for(i in 1:length(ss)){

##subset population using location points, based on sample size in sequence (ss)
  #, and retain spatial spread across entire sample site

  #specify the parameters that do this
  koalas.wrs <- pp.subsample(spat.pts.df, n=ss[[i]], window='extent',
                             sigma = 'Scott',gradient=2)

  #set coordinate reference system for these subsampled points
  projection(koalas.wrs) <- crs("+init=epsg:32756")

##check spatial subsampling by plotting (might need pop-out graph
  #window [e.g. windows()] for larger datasets)

  plot(koalas.wrs)

  k2k<-as.data.frame(koalas.wrs)
  ids.df<-subset(k2ks,rownames(k2ks)%in%rownames(k2k))
  db4<-db3[colnames(db3)%in%ids.df$Name]
  db4<-cbind(db3[,1:17],db4)
```

```
##degrade DNA randomly to match selected call-rate parameters

  #make a vector of rows which index each unique SNP - e.g. in example data,
  #SNPs are in double row format (2 per locus) so need to loop through each pair

  col=seq(6,nrow(db4),2)
  CR_2a=rtruncnorm(n=length(unique(all.dups)),a=minCR,b=maxCR,mean=meanCR,sd=sdevcR)
  cr_seq=trunc((1-CR_2a)*nrow(ids.df))#set the range of observed snpCR (two per SNP locus)
  for(j in 1:length(col)){
    k=col[[j]]+1
    rep<-sample(seq(18,ncol(db4)-1), cr_seq[[j]],replace = FALSE)
    db4[col[[j]]:k,rep]<- "-"
  }
  db.fin<-db4[,1:ncol(db4)-1] # <- this is the final degraded dataset to analyse

  # write it out and then back in as a genlight object (correct format for dartR package)
  write.table(db.fin,"simgendat1.csv",row.names=FALSE,col.names=FALSE,sep=",")

  #read back in using same metadata file
  gl2a<-gl.read.dart(filename = "simgendat1.csv",
                     covfilename = "idmeta.csv")


  #windows()  #<- this may be required to visualise resulting graphs


  #Apply appropriate filtering parameters: here we use those described in the manuscript
  gl1F <- gl2a  %>%
     # Filter out monomorphic loci
    gl.filter.monomorphs(v = 0) %>%
     # Filter CallRate by Individuals
    gl.filter.callrate(method = "ind", threshold = 0.025) %>%
     # Filter CallRate by loci
    gl.filter.callrate(method = "loc", threshold = 0.70) %>%
    # Filter on reproducibility, threshold 50% reproducible
    gl.filter.RepAvg(t = 0.95) %>%
    gl.filter.maf(threshold=0.01)%>%
     # Remove all but one locus where there is more than one locus per sequence tag.
    gl.filter.secondaries()

##create a gi object for population genetics analyses

  gi1<-gl2gi(gl1F)
  #expected heterozygosity (He)
  He[[i]]=Hs(genind2genpop(gi1))
  #observed heterozygosity (Ho)
  H1=summary(gi1)
  #str(H1)    #<- check structure of H1 if necessary
  Ho=mean(H1$Hobs)
  #Inbreeding coefficient (Fis)
  Fis[[i]]=(He[[i]]-Ho)/He[[i]]
  #Shannon's Index (I)
  div<-as.data.frame(locus_table(gi1,index='shannon'))
```

```
  I[[i]]=mean(div$H)
  I_se[[i]]<-sd(div$H)/sqrt(length(div$H))

  #Internal relatedness (IR)

  #calculate internal relatedness
  HZ<-gl2demerelate(gl1F)
  HZ<-HZ[,-c(2)]
  LociNames <- paste("SNP", seq(1:((length(HZ)-1)/2)), sep = "")
  ghet <- GENHET(HZ, estimfreq = "T", locname = LociNames)
  HZdf <- as.data.frame(ghet)
  HZdf$IR <- as.numeric(as.character(HZdf$IR))
  IR_s<-HZdf[,c("sampleid","IR")]

##merge and calculate correlation with observed IR - remember that because IR
  #is an individual measure, we assess how strong the correlation is between the
  #new (degraded DNA) IR and the observed IR

  IR_sm<-merge(IR_s,observedIR,by="sampleid")
  colnames(IR_sm)<-c("id","randomIR","observedIR")
  IRdfs[[i]]<-cor(IR_sm$randomIR,IR_sm$observedIR)

##spatial autocorrelation calculations

  #this require a geographic distance between all pairs in the pop, and a genetic
  #distance between all pairs in the population

  #calculate geographic proximity matrix [euclidean distance]
  euc<-spDists(koalas.wrs)
  colnames(euc)<-rownames(euc)<-ids.df$Name
  head(euc)
  #calculate a genetic distance matrix [nei distance]
  gen<-stamppNeisD(gl1F,FALSE)
  colnames(gen)<-rownames(gen)
  euc<-euc[rownames(euc)%in%colnames(gen),colnames(euc)%in%colnames(gen)]
  #reorder genetic distance matrix to match geographic distance matrix
  gen2<-gen[colnames(euc),colnames(euc)]


##run observed spatial autocorrelation under given conditions (population
  #sample size, degradation etc)

  #bins = number of even distance classes- you may need to play with
  #these to get the distance classes you are after


  spaut<-spautocor(gen2,euc, bins = 58)  # 250 m distance classes
  sp_rvals[[i]]<-spaut[,3]##extract r value at each distance bin


#bootstrap matrix to get expected r under no structure (resample
  #matrix with replacement)
```

```
    spautRAND<-list()
  nr<-dim(gen2)[1]
  for(j in 1:999){
    gen3<-gen2[sample.int(nr,replace=TRUE),]
    euc2<-euc[sample.int(nr,replace=TRUE),]
    diag(gen3)<-0
    diag(euc2)<-0
    spautRAND[[j]]<-(spautocor(gen3,euc2,bins=58))[,3]
  }

  ##bind observed and expected
  sp<-as.data.frame(cbind(t(do.call("rbind",spautRAND)),spaut[,3]))
  sp[is.na(sp)] <- 0 ##tidy up
  #extract confidence intervals of expected r values
  sp_CIs[[i]]<-t(apply(sp, 1,function(x) quantile(x[1:999],probs=c(0.95,0.05))))
  #and calculate whether observed is different from observed
  sp_TF[[i]]<-between(spaut[,3],sp_CIs[[i]][,2],sp_CIs[[i]][,1])


##end loop
  cat(i)

}
```

Tidy up and compile results from loop

```
names(He)<-nameslist
names(Fis)<-nameslist
names(I)<-nameslist
names(I_se)<-nameslist
names(IRdfs)<-nameslist
names(sp_TF)<-nameslist

dg_results<-data.frame(He=He,Fis=Fis,Shannons=I,Shan_SE=I_se,
                       IR=IRdfs,spCorr250=as.data.frame(t(bind_cols(sp_TF)))[,1],
                       spCorr500=as.data.frame(t(bind_cols(sp_TF)))[,2],samplesize=names(He))
```

You should now have your analysis outputs ready for evaluation!

Good luck!