SNP data analysis in R

version 2017-01-05 (Filip Kolář)

1. What is a VCF file?

Our set of ~ 10,000 single nucleotide polymorphisms (SNPs) is stored in the compressed (gzipped) variant call format (VCF) file *diploid_arenosa_dp8.perc0.5.vcf.gz.* The dataset comprises 171 individuals from 64 populations of diploid *Arabidopsis arenosa* collected across entire geographical range of this cytotype (for further details see the attached article, Kolar et al. 2016, Mol Ecol). We collected two to four individuals per population what is too few for reliable estimation of intra-population variation. Thus, for the summary per-population statistics, we will work with populations lumped into five geographically and genetically homogeneous groups (Baltic (BAL), Dinaric (DIN), Pannonian (PAN), Western Carpathian (WCA), Southern Carpathian (SCA); see the Fig 1). The sample naming convention is: *GROUPNAME(3digits)_POPNAME(3digits)INDIVIDUALNAME(1digit) - e.g. DIN_AA106A*.

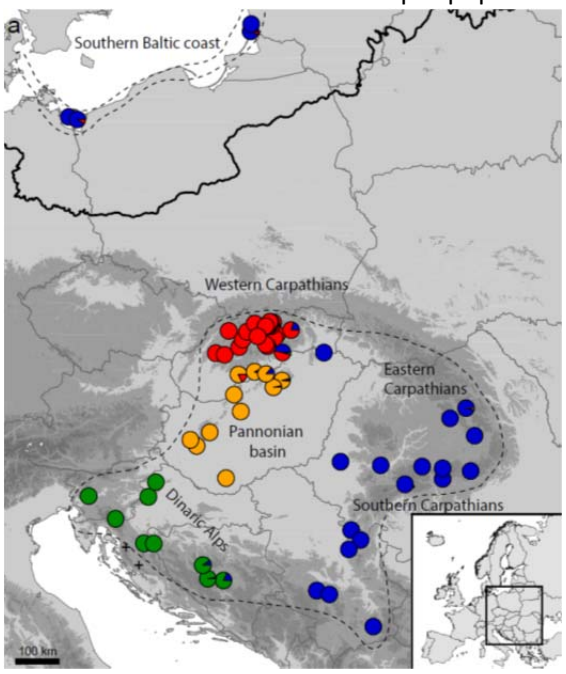

Fig. 1. The populations investigated, five geographical groups depicted by different colour (Baltic pops are geographically separate but most similar to the blue group).

VCF file is becoming a golden standard how to store SNP data, let's look at the structure of the file. The provided VCF was already filtered using some standard criteria in GATK (the variant calling software used); in addition, we retained only biallelic variable sites that were covered by at least 8 reads (8×) in at least 50 % of the individuals. **Do not open the huge gzipped VCF** *diploid_arenosa_dp8.perc0.5.vcf.gz* but look at the first nine SNPs that are saved in a separate file *mini_9SNP.vcf*. Open it e.g. in Notepad++ and inspect:

- **1a)** How many reads supported the first variable site (2 25723) over all samples (DP field)? How many alleles were called in this site over all samples (the AN filed) and how many of them were the alternative allele (AC)? Does the AN number correspond to the total number of alleles that should be available in the total dataset (171 diploid individuals)? Why yes/not?
- **1b)** What is the alternative allele in the second variable site (2 25749)? How many reads supported this allele in the first individual (WCA_AA007A) (Hint: this is the corresponding number in the "AD", = allele depth, field)?
- **1c)** What was genotype quality, GQ, of the heterozygote call (0/1) in the second variable site (2 25749) the first individual (WCA_AA007A)? What does this number mean (Hint look at GATK, the software used for variant calling in this VCF, manual pages here http://gatkforums.broadinstitute.org/gatk/discussion/1268/what-is-a-vcf-and-how-should-i-interpret-it, search for "Genotype quality"
- **1d)** Write a tiny fasta file by converting all the 9 variable sites of the VCF in the first two individuals (WCA_AA007A, WCA_AA007B) into DNA base positions (ATCG). Code the intra-

individual polymorphisms (heterozygotes) using standard IUPAC ambiguity codes (Hint you will need only this one: C/T=Y). NB: the slightly modified VCF without INFO field (*mini_9SNP_noinfofield.vcf*) is more easily readable for this purpose.

- **1e)** What is expected heterozygosity, He, of the second variable site (2 25749) of the first population (WCA_AA007, has four individuals)? Hint: He = 2pq; during calculations disregard individuals with missing data (./.).

2. Basic inspection of a VCF in R

Now load the entire VCF into R and inspect it more seriously. We will sequentially create an R script file containing all the commands necessary for our analyses. If you save it during the process, you can rerun the entire set of analyses again whenever you want.

Open R Studio,

- *File -> New File -> R Script* - save this new file to your working directory (the same with input files)
- *Session -> Set Working Directory -> To source file location*
- Now sequentially copy-paste following commands to the new R file and run each batch. To run the command, highlight the lines you just inserted by mouse and press small icon with green arrow and „Run“ in the upper right corner.

First load the required libraries for all the downstream analyses – i.e. add the following text to your script and Run

```
library(vcfR)
library(adegenet)
library(adegraphics)
library(pegas)
library(StAMPP)
library(lattice)
library(gplots)
library(ape)
library(ggmap)
```

Now load the data and check the input. (NB: in this section we will use the vcfR package, for more info incl. tutorials see https://cran.r-project.org/web/packages/vcfR/vignettes/intro_to_vcfR.html).

```
vcf <- read.vcfR("diploid_arenosa_dp8.perc0.5.vcf.gz")   #read in all data
head(vcf)                #check the vcf object
vcf@fix[1:10,1:5]        #check
```

Then plot important statistics summed over entire VCF

```
chrom <- create.chromR(name='RAD_data', vcf=vcf)
plot(chrom) # plot the data
```

Then extract the allele depths per each sample (DP field of VCF) and plot distribution of allele depths of all sites per each sample. NB: You may inspect and visualize other fields of VCF, e.g. allele depth (AD) or genotype quality (GQ)

```
#quick check read depth distribution per individual
dp <- extract.gt(vcf, element='DP', as.numeric=TRUE)

pdf("DP_RAD_data.pdf", width = 10, height=3) # boxplot
par(mar=c(8,4,1,1))
```

```
boxplot(dp, las=3, col=c("#C0C0C0", "#808080"), ylab="Read Depth (DP)",
las=2, cex=0.4, cex.axis=0.5)
dev.off()

#zoom to smaller values
pdf("DP_RAD_data_zoom.pdf", width = 10, height=3) # boxplot
par(mar=c(8,4,1,1))
boxplot(dp, las=3, col=c("#C0C0C0", "#808080"), ylab="Read Depth (DP)",
las=2, cex=0.4, cex.axis=0.5, ylim=c(0,50))
abline(h=8, col="red")
dev.off()
```

- **2a)** Which two samples have the highest average depth (DP) of their variable sites?
- **2b)** In the intro we specified that after filtering this VCF, we retained only sites that were covered at least 8× in at least 50% of the individuals. Are there still sites with DP < 8 (i.e below the red line in the *DP_RAD_data_zoom.pdf* file)? Why?
- **2c)** Would you remove any sample from this VCF based on read depth filtering criteria? Which and why?

3. Basic manipulation of a SNP matrix (genlight)

Convert the data into adegenet-native format called **genlight**.

```
### convert to genlight
aa.genlight <- vcfR2genlight(vcf, n.cores=1)
locNames(aa.genlight) <- paste(vcf@fix[,1],vcf@fix[,2],sep="_")   # add
real SNP.names
pop(aa.genlight)<-substr(indNames(aa.genlight),1,3)               # add pop
names: here "population" (group) names are first 3 chars of ind name
```

Now, check the genlight using following commands.

NB: the original matrix of biallelic SNPs is stored in a way of one number per individual per site that reflects the number of alternative alleles in that site in that individuals (i.e. 0, 1, or 2 in a diploid individual).

```
# check the genlight
aa.genlight                         # check the basic info on the genlight
object
indNames(aa.genlight)               # check individual names
as.matrix(aa.genlight)[1:16,1:10]  # see tiny bit of the data
pop(aa.genlight)                    # population assignment

# look at the total data matrix (0,1,2; white = missing data)
glPlot (aa.genlight)  # takes some time

# N missing SNPs per sample
x <- summary(t(as.matrix(aa.genlight)))
write.table(x[7,], file = "missing.persample.txt", sep = "\t")  # NAs, if
present, are in seventh row of summary
```

- **3a)** How many SNPs are in the total dataset?
- **3b)** Look at the small piece of the data matrix using command `as.matrix(aa.genlight)[1:16,1:10]`. Compare the 0,1,2 coding of the first

individuals with the *mini_9SNP.vcf* file you inspected in the first section. Into which number of the genlight matrix is translated the 1/1 genotype of the original VCF?

- **3c)** Look at the figure produced by glPlot. Why are the missing data depicted by white colour and not by the blue colour (i.e. zeros, 0, in the genlight matrix)
- **3d)** How much missing data is there? How do the missing data correspond with average read depth per sample? (Hint: visually compare the *DP_RAD_data_zoom.pdf* file with numbers in the *missing.persample.txt* output file)

4. Draw allele frequency spectra (AFS)

Now let's look at some population-level summary statistics. We will plot the unfolded **allele frequency spectrum** (AFS, SFS), i.e. distribution of counts of alternative (non-reference) alleles across all sites in the population. This is essential population summary characteristics frequently used e.g. in the coalescent simulations; for more info on AFS see e.g. Wikipedia. NB: Missing data are problem in AFS construction; in our toy example we simply remove all SNPs with missing data, in reality there are more clever ways of maximizing the information which is used for AFS calculation.

Fist show AFS for the entire dataset.

```
###plot total AFS of the dataset
mySum <- glSum(aa.genlight, alleleAsUnit = TRUE)
barplot(table(mySum), col="blue", space=0, xlab="Allele counts",
main="Distribution of ALT allele counts in total dataset")
```

Second, visualize AFS only for one selected population (BAL)

```
###plot AFS per one pop
aa.genlight.sep <- seppop(aa.genlight, drop=TRUE)                    #
separate genlights per population
aa.genlight.sep$BAL

# after seppop you must remove the nonvariant positions within the
population
n.alleles.BAL <-colSums(as.matrix(aa.genlight.sep$BAL))      # how many
alternative alleles are in each locus?
summary(as.factor(n.alleles.BAL))                            # how many
particular categories of alternative allele counts are in my pop?
aa.genlight.BAL <- new("genlight", (as.matrix(aa.genlight.sep$BAL))
                       [,(colSums(as.matrix(aa.genlight.sep$BAL)) > 0) &
                         (colSums(is.na(as.matrix(aa.genlight.sep$BAL))) ==
0)]) # remove the reference-only positions AND remove columns with NA
aa.genlight.BAL
summary(colSums(as.matrix(aa.genlight.BAL)))  #  check if there are no
zeros
# plot unfolded AFS - for one pop.
mySum <- glSum(aa.genlight.BAL, alleleAsUnit = TRUE)
barplot(table(mySum), col="blue", space=0, xlab="Allele counts",
main="Distribution of ALT allele counts in BEL")    # plot the original
counts of each category
```

Now, plot AFS for all populations in a batch and save this into a pdf file (i.e., using `lapply` function which goes over all elements of the list of genlights)

```
#### plot AFS for all pops in a batch
aa.genlight.sep <- seppop(aa.genlight, drop=TRUE)  # separate genlight per
population
# remove the nonvariant positions AND columns with NA within that pop.
aa.genlight.sep.2 <- lapply (aa.genlight.sep, function (pop)
{new("genlight", (as.matrix(pop))[,(colSums(as.matrix(pop)) > 0)
                 & (colSums(is.na(as.matrix(pop))) == 0)])})
##add pop identity to list elements
listnames<-names(aa.genlight.sep.2)
for (i in seq(listnames)) {pop(aa.genlight.sep.2[[i]])<-
substr(indNames(aa.genlight.sep.2[[i]]),1,3)}

# loop over each population in a list of populations and draw AFS into one
fig
pdf("AFS_all_barplot.pdf", width=5, height=5)
par(mfrow=c(2,3),mar=c(2,2,2,0))
mySum <- lapply (aa.genlight.sep.2, function (pop) {
  barplot(table(glSum(pop, alleleAsUnit=T)), col="blue", space=0,
          xlab="Allele counts",
main=paste(levels(pop(pop)),sum(table(glSum(pop, alleleAsUnit=T))),"SNPs",
sep=" "))
})
dev.off()
par(mfrow=c(1,1))
```

- **4a)** How many SNPs were used for construction of AFS in the BAL population?
- **4b)** Compare AFS of all five populations (groups) in the fig. *AFS_all_barplot.pdf*. Which population has most distinct AFS? Which allele count categories in this population are underrepresented relatively to the other populations? What biological process might have been responsible for this?
- **4c)** In all the AFS plots you may observe the rightmost column, i.e. number of fixed alternative alleles, is quite high. These sites are in fact non-variant sites across our entire dataset. Why did such sites remain in our VCF, as our dataset includes only single nucleotide polymorphisms? Try to remove such sites with fixed alternative alleles from the BAL population and plot the AFS again. Hint: calculate maximum number of alleles per pop: `nchr <- (nrow(as.matrix(aa.genlight.sep$BAL))*2)` and then add this criterion to the selection of SNPs when constructing the corresponding genlight (aa.genlight.BAL): `& (colSums(as.matrix(aa.genlight.sep$BAL)) != nchr`

5. Principal component analysis (PCA) and subsetting of the data

Calculate and visualize PCA for the total dataset. NB: if you have problems with parallelization, add and Run into R the function `glPcaFast`, that is available in the appendix at the very end of this document.

```
pca.1 <- glPca(aa.genlight, nf=300, n.cores=1)     # retain first 300 axes
(for later use in find.clusters); slow function
#pca.1 <- glPcaFast(aa.genlight, nf=300)

# proportion of explained variance by first three axes
pca.1$eig[1]/sum(pca.1$eig) # proportion of variation explained by 1st axis
pca.1$eig[2]/sum(pca.1$eig) # proportion of variation explained by 2nd axis
```

```
pca.1$eig[3]/sum(pca.1$eig) # proportion of variation explained by 3rd axis

# save fig
pdf ("PCA_all_SNPs_ax12.pdf", width=14, height=7)
col <- funky(5)
g1 <- s.class(pca.1$scores, pop(aa.genlight),  xax=1, yax=2,
col=transp(col,.6),
             ellipseSize=0, starSize=0, ppoints.cex=4, paxes.draw=T,
pgrid.draw =F, plot = FALSE)
g2 <- s.label (pca.1$scores, xax=1, yax=2, ppoints.col = "red", plabels =
list(box = list(draw = FALSE),

optim = TRUE), paxes.draw=T, pgrid.draw =F, plabels.cex=1, plot = FALSE)
ADEgS(c(g1, g2), layout = c(1, 2))
dev.off()
```

Subset the genlight to select only Carpathian and Baltic populations and calculate PCA. Compare the figure with Fig 2 in the article

```
include.list <- grep("(WCA|SCA|BAL)", indNames(aa.genlight), value = T)
# get list of samples matching the desired groupnames
aa.genlight.BALCARP <- new("genlight",
(as.matrix(aa.genlight)[include.list, ]))  # create new genlight using this
selection
aa.genlight.BALCARP <- new("genlight", (as.matrix(aa.genlight.BALCARP))
[,(colSums(is.na (as.matrix(aa.genlight.BALCARP))) < 60)]) # retain only
positions with no-missing data in > 50% individuals
pop(aa.genlight.BALCARP)<-substr(indNames(aa.genlight.BALCARP),1,3) # add
pop names: here pop names are first 3 chars of ind name
aa.genlight.BALCARP

#pca.2 <- glPca(aa.genlight.BALCARP, nf=300, n.cores=1)     # retain first
300 axes (for later use in find.clusters); slow function
pca.2 <- glPcaFast(aa.genlight.BALCARP, nf=300)             # for running
this, firstly tun the modified function at the end of this doc

# save fig
pdf ("PCA_BALCARP_SNPs_ax12.pdf", width=14, height=7)
col <- funky(3)
g1 <- s.class(pca.2$scores, pop(aa.genlight.BALCARP),  xax=1, yax=2,
col=transp(col,.6),
             ellipseSize=0, starSize=0, ppoints.cex=4, paxes.draw=T,
pgrid.draw =F, plot = FALSE)
g2 <- s.label (pca.2$scores, xax=1, yax=2, ppoints.col = "red", plabels =
list(box = list(draw = FALSE),

optim = TRUE), paxes.draw=T, pgrid.draw =F, plabels.cex=1, plot = FALSE)
ADEgS(c(g1, g2), layout = c(1, 2))
dev.off()
```

- **5a)** Which group is the most divergent from the rest?
- **5b)** Does the position of the BAL population in the second, Carpathian+Baltic-only PCA, support its origin through admixture of the two Carpathian groups (SCA, WCA)?
- **5c)** In the section 3, we used the `substr` command for creating the population names using the "group" part of the individual name (i.e. first 3 digits). Modify this command to create population assignment using real population names (i.e. the AA007 etc. codes) and rerun the

PCA to see how are the populations organized. (Hint: look what `substr` does by typing `?substr`)

6. K-means clustering and discriminant analysis of principal components (DAPC)

K-means clustering is a fast method how to assign individuals into groups. Unlike STRUCTURE it does not make assumptions on population genetic parameters such as Hardy Weinberg equilibrium. We will perform K-means clustering using the already calculated glPCA object to save some time. Following lines are just a set of commands how to calculate K-means clustering and DAPC, however, on the way you will have to do several important decisions. If you would like to seriously use this method, you should first read the following tutorial adegenet.r-forge.r-project.org/files/tutorial-dapc.pdf.

Type the following command (we will do clustering for K from 1 to 20 clusters, using many, 1,000,000, random starts) and then answer two interactive questions:

```
grp <- find.clusters(aa.genlight, max.n.clust=20, glPca = pca.1, perc.pca =
100, n.iter=1e6, n.start=1000)
```

- At the first question ("Choose the number PCs to retain (>=1):") use all PC axes for the grouping, i.e., just press Enter and wait ca 30 seconds
- At the second question ("Choose the number of clusters (>=2:") select the optimal number of groups (K). This is the trickiest task in any clustering method. Adegenet offers Bayesian information criterion (BIC, simultaneously plotted in the Plots panel) to aid our decision. Optimally you should select the lowest BIC value before the values starts to rise again. Type 4 in our case.

You may save this grouping and open it later, e.g. in excel...

```
write.table(grp$grp, file="grouping_Kmeans_all.txt", sep="\t", quote=F,
col.names=F)
```

Let's plot this grouping onto a map using ggmap. Make sure that you have the file with coordinates of the populations (*pop_coords.txt*) in the working directory.

```
# ugly quick hack to prepare the grouping of populations, not individuals
x <- data.frame(keyName=names(grp$grp), value=grp$grp, row.names=NULL)  #
convert vector into data frame
x$pop <- as.factor(substr(x$keyName,5,9))   # get population identity
y<-x[order(x$pop),]                # order it
grp.pop<-y[!duplicated(y$pop),]    # remove duplicates (one line per pop)
coords <- read.csv ("pop_coords.txt", sep ="\t")  # import tab-sep file

# plot it!
map <- get_map(location =  c(lon = 14, lat = 50), zoom = 5)
mapPoints <- ggmap(map) + geom_point(aes(x = coords$lon, y = coords$lat,
colour = factor(grp.pop$value)),
                           data = grp.pop) + scale_colour_manual(values =
c("red","blue","black","green"))
mapPoints
```

Based on this this grouping, we may calculate discriminant analysis of principal components (DAPC).

```
dapc1<-dapc(aa.genlight, grp$grp, glPca = pca.1)
```

- At the first question, select a reasonable N of PCs without sacrificing too much information, e.g. 50
- At the second question, select N of discriminant functions. As we selected K=4 we have only three discriminant axes, let's select all three.

You may plot the results either as a scatter plot or using the barplot showing the proportions of membership to each of the K-means groups, somewhat similarly to Structure.

```
## plot results
scatter(dapc1)           # scatterplot
col <- funky(5)
compoplot(dapc1, cex.names = 0.4, legend=F, col=col)


# barplot
pdf("DAPC_all.pdf",width=20,height=5)
compoplot(dapc1, cex.names = 0.4, legend=F, col=col)
dev.off()
```

7. Calculation and visualization of Nei's distances

First calculate the matrices of Nei's (1972) distances among individuals and populations as well as pairwise Fst among populations (and save them in phylip format)

```
### Calculate Nei's distances between individuals/pops
aa.D.ind <- stamppNeisD(aa.genlight, pop = FALSE)  # Nei's 1972 distance
between indivs
stamppPhylip(aa.D.ind, file="aa.indiv_Neis_distance.phy.dst") # export
matrix - for SplitsTree
aa.D.pop <- stamppNeisD(aa.genlight, pop = TRUE)   # Nei's 1972 distance
between pops
stamppPhylip(aa.D.pop, file="aa.pops_Neis_distance.phy.dst") # export
matrix - for SplitsTree


### Calculate pairwise Fst among populations
aa.genlight@ploidy <- as.integer(ploidy(aa.genlight))
aa.fst<-stamppFst(aa.genlight, nboots = 1, percent =95, nclusters=4)
#modify the matrix for opening in SplitsTree
aa.fst.sym <- aa.fst
aa.fst.sym[upper.tri(aa.fst.sym)] <- t(aa.fst.sym)[upper.tri(aa.fst.sym)]
# add upper triangle
aa.fst.sym[is.na(aa.fst.sym)] <- 0
#replace NAs with zero
stamppPhylip(aa.fst.sym, file="ALL_aa.pops_pairwise_Fst.phy.dst")   #
export matrix - for SplitsTree
```

Once the first matrix is calculated and saved, take a small detour, open the **dst* files in SplitsTree.

- **7a)** How does the interpopulation Neis's and pairwise Fst distances differ? What can you tell about position of the WCA/SCA/BAL groups?

Another nice visualization of the distances is a heatmap. Plot it, save and look into the *Neis_dist_heatmap.pdf*.

- **7b)** What would you say about the homogeneity of the SCA group and about the position of the BAL group?

```
### heatmap of the indivs distance matrix
colnames(aa.D.ind) <- rownames(aa.D.ind)
pdf(file="Neis_dist_heatmap.pdf", width=10, height=10)
heatmap.2(aa.D.ind, trace="none", cexRow=0.4, cexCol=0.4)
dev.off()
```

Optionally, you may also visualize the distances as Neighbor joining tree or heatmap. The saved *NJ.Neis.dist.tree.tre* file could be then opened and edited, e.g. in Figtree

```
# plot and save NJ tree
plot(nj(aa.D.ind))
write.tree(nj(aa.D.ind),file="NJ.Neis.dist.tree.tre")
```

8. Calculation of AMOVA and isolation-by-distance based on Nei's distances

Now let's work with the real 64 populations, coded by the AAXXX codes. The five geographical groups will serve as grouping variables in hierarchical AMOVA. First define a new genlight with different population definition plus define the grouping variable:

```
#### DEFINE the 64 original populations using the AAXXX codes
aa.genlight2 <- aa.genlight
pop(aa.genlight2)<-substr(indNames(aa.genlight2),5,9)  # define populations
as the AAXXX codes
```

Then calculate interpopulation distances using this new population definition, and modify these distance matrices into a "dist" object, used in the analyses below

```
aa.D.pop2 <- stamppNeisD(aa.genlight2, pop = TRUE)     # Nei's 1972
distance between pops
stamppPhylip(aa.D.pop2, file="aa.pops2_Neis_distance.phy.dst") # export
matrix - for SplitsTree

# create the dist objects used in analyses below
colnames(aa.D.ind) <- rownames(aa.D.ind)
aa.D.ind.dist<-as.dist(aa.D.ind, diag=T)
attr(aa.D.ind.dist, "Labels")<-rownames(aa.D.ind)          # name the rows
of a matrix

colnames(aa.D.pop2) <- rownames(aa.D.pop2)
aa.D.pop.dist<-as.dist(aa.D.pop2, diag=T)
attr(aa.D.pop.dist, "Labels")<-rownames(aa.D.pop2)          # name the rows
of a matrix
```

Now calculate analysis of molecular variance (AMOVA) using the Nei's inter-individual distances with AAXXX populations and the five major geographical groups as grouping factors

```
### AMOVA
pops <- as.factor(pop(aa.genlight2))                        # define
populations
groups <- as.factor(substr(indNames(aa.genlight2),1,3))     # define groups
```

```
# one-level AMOVA
(res <- pegas::amova(aa.D.ind.dist ~ pops))           # one-level AMOVA,
default nperm=1000

# hierarchical AMOVA
(res <- pegas::amova(aa.D.ind.dist ~ groups/pops))    # hierarchical AMOVA
```

- **8a)** What proportion of variation was explained by the real populations (Hint: divide the pops SSD / Total SSD)? Why is this number so high? What proportion of variation was explained by the five geographic groups?

Finally, calculate whether isolation-by distance relationship holds for our populations, i.e. whether there is significant correlation among matrices of genetic (Nei's or Fst) and geographical distances among the real (AAXXX) populations. Make sure that you have the file with coordinates of the populations (*pop_coords.txt*) in the working directory (the order should be exactly the same as that of individuals in the genlight object).

First, load in the coordinates and check the input on a map

```
### Isolation by distance
coords <- read.csv ("pop_coords.txt", sep ="\t")     # tab-separated file
for all pops
xy.coords.only<- subset(coords, select=c("lat","lon"))
Dgeo <- dist(xy.coords.only)

#optionally, check plotting the points on a map
library(ggmap)
map <- get_map(location =  c(lon = 14, lat = 50), zoom = 5)
mapPoints <- ggmap(map) + geom_point(data = coords, aes(x = lon, y = lat,
colour="blue")) + geom_text(data = coords, aes(x = lon, y = lat,label = pop,
colour = "red"), size = 4, vjust = 0, hjust = -0.5)
mapPoints
```

Then, calculate the Mantel test

```
#test IBD
IBD <- mantel.randtest(Dgeo,aa.D.pop.dist)
IBD
plot(Dgeo,aa.D.pop.dist, pch=20,cex=.5)
abline(lm(aa.D.pop.dist~Dgeo))
```

Finally, plot the isolation-by distance relationship in a nicer plot. The density of points is plotted in a kernel-smoothed colour scale.

```
#plot and check for denser areas in the plot indicating sub-groups
library(MASS)
dens <- kde2d(Dgeo,aa.D.pop.dist, n=300, lims=c(-1, 16, 0, 0.08))
myPal <- colorRampPalette(c("white","blue","gold", "orange", "red"))
plot(Dgeo, aa.D.pop.dist, pch=20,cex=.5)
image(dens, col=transp(myPal(300),.7), add=TRUE)
abline(lm(aa.D.pop.dist~Dgeo))
title("Correlation of Genetic and Geographical distances")
```

- **8b)** Is there significant isolation-by distance? What is the correlation coefficient of geographical and genetic distances (see the “Observation:” line)?

APPENDIX

Modified function for faster PCA calculations

```
glPcaFast <- function(x,
                      center=TRUE,
                      scale=FALSE,
                      nf=NULL,
                      loadings=TRUE,
                      alleleAsUnit=FALSE,
                      returnDotProd=FALSE){

  if(!inherits(x, "genlight")) stop("x is not a genlight object")
  # keep the original mean / var code, as it's used further down
  # and has some NA checks..
  if(center) {
    vecMeans <- glMean(x, alleleAsUnit=alleleAsUnit)
    if(any(is.na(vecMeans))) stop("NAs detected in the vector of means")
  }
  if(scale){
    vecVar <- glVar(x, alleleAsUnit=alleleAsUnit)
    if(any(is.na(vecVar))) stop("NAs detected in the vector of variances")
  }
  # convert to full data, try to keep the NA handling as similar
  # to the original as possible
  # - dividing by ploidy keeps the NAs
  mx <- t(sapply(x$gen, as.integer)) / ploidy(x)
  # handle NAs
  NAidx <- which(is.na(mx), arr.ind = T)
  if (center) {
    mx[NAidx] <- vecMeans[NAidx[,2]]
  } else {
    mx[NAidx] <- 0
  }
  # center and scale
  mx <- scale(mx,
              center = if (center) vecMeans else F,
              scale = if (scale) vecVar else F)
  # all dot products at once using underlying BLAS
  # to support thousands of samples, this could be
  # replaced by 'Truncated SVD', but it would require more changes
  # in the code around
  allProd <- tcrossprod(mx) / nInd(x) # assume uniform weights
  ## PERFORM THE ANALYSIS ##
  ## eigenanalysis
  eigRes <- eigen(allProd, symmetric=TRUE, only.values=FALSE)
  rank <- sum(eigRes$values > 1e-12)
  eigRes$values <- eigRes$values[1:rank]
  eigRes$vectors <- eigRes$vectors[, 1:rank, drop=FALSE]
  ## scan nb of axes retained
  if(is.null(nf)){
    barplot(eigRes$values, main="Eigenvalues", col=heat.colors(rank))
    cat("Select the number of axes: ")
    nf <- as.integer(readLines(n = 1))
  }
  ## rescale PCs
```

```
  res <- list()
  res$eig <- eigRes$values
  nf <- min(nf, sum(res$eig>1e-10))
  ##res$matprod <- allProd # for debugging
  ## use: li = XQU = V\Lambda^(1/2)
  eigRes$vectors <- eigRes$vectors * sqrt(nInd(x)) # D-normalize vectors
  res$scores <- sweep(eigRes$vectors[, 1:nf, drop=FALSE],2,
sqrt(eigRes$values[1:nf]), FUN="*")
  ## GET LOADINGS ##
  ## need to decompose X^TDV into a sum of n matrices of dim p*r
  ## but only two such matrices are represented at a time
  if(loadings){
    if(scale) {
      vecSd <- sqrt(vecVar)
    }
    res$loadings <- matrix(0, nrow=nLoc(x), ncol=nf) # create empty matrix
    ## use: c1 = X^TDV
    ## and X^TV = A_1 + ... + A_n
    ## with A_k = X_[k-]^T v[k-]
    myPloidy <- ploidy(x)
    for(k in 1:nInd(x)){
      temp <- as.integer(x@gen[[k]]) / myPloidy[k]
      if(center) {
        temp[is.na(temp)] <- vecMeans[is.na(temp)]
        temp <- temp - vecMeans
      } else {
        temp[is.na(temp)] <- 0
      }
      if(scale){
        temp <- temp/vecSd
      }
      res$loadings <- res$loadings + matrix(temp) %*% eigRes$vectors[k,
1:nf, drop=FALSE]
    }
    res$loadings <- res$loadings / nInd(x) # don't forget the /n of X_tDV
    res$loadings <- sweep(res$loadings, 2, sqrt(eigRes$values[1:nf]),
FUN="/")
  }
  ## FORMAT OUTPUT ##
  colnames(res$scores) <- paste("PC", 1:nf, sep="")
  if(!is.null(indNames(x))){
    rownames(res$scores) <- indNames(x)
  } else {
    rownames(res$scores) <- 1:nInd(x)
  }
  if(!is.null(res$loadings)){
    colnames(res$loadings) <- paste("Axis", 1:nf, sep="")
    if(!is.null(locNames(x)) & !is.null(alleles(x))){
      rownames(res$loadings) <- paste(locNames(x),alleles(x), sep=".")
    } else {
      rownames(res$loadings) <- 1:nLoc(x)
    }
  }
  if(returnDotProd){
    res$dotProd <- allProd
    rownames(res$dotProd) <- colnames(res$dotProd) <- indNames(x)
  }
  res$call <- match.call()
  class(res) <- "glPca"
  return(res)
}
```