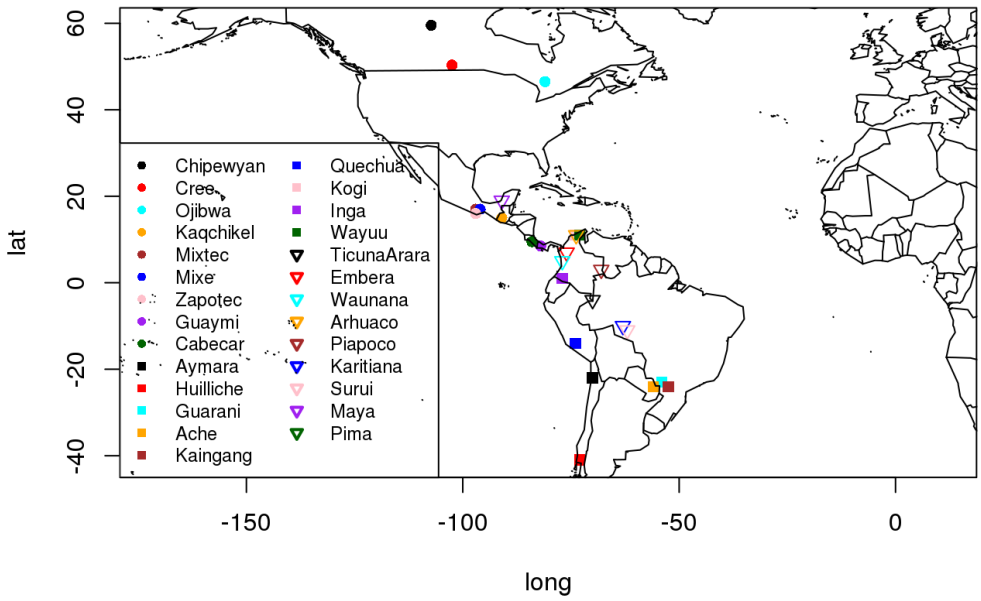
TP2 geneticData Mining - PCA-regression in genetics

Lamyaa BOUZBIBA Alexandre POUPEAU Eloise JULIEN

1 point

1. geneticData

```
NAm2 = read.table("NAm2.txt", header = TRUE)

# unique is used to get the name of all population / ethnics without duplicates
names=unique(NAm2$Pop)
npop=length(names)

# coordinates for each pop
coord=unique(NAm2[, c("Pop","long","lat")])
colPalette=rep(c("black","red","cyan","orange","brown","blue","pink","purple","darkgr
een"),3)

# type 16 = circle / type 15 = square / type 25 = triangle
pch=rep(c(16,15,25),each=9)

# the display works because for each population we have the latitude and the longitud
e
plot(coord[, c("long","lat")],pch=pch,col=colPalette,asp=1)

# asp allows to have the correct ratio between  axis  longitude and latitude
# Then the map is not deformed
legend("bottomleft",legend=names,col=colPalette,lty=-1,pch=pch,cex=.75,ncol=2,lwd=2)

library(maps);map("world",add=T)
```



The description of the script is in the commentaries above.

2 points

2. Regression

```
# turn the matrix into a geneticData frame
geneticData = NAm2[,-c(1:7)]

geneticDataFrame = data.frame(geneticData)

# creation of the linear model
reg = lm(formula = long ~ . , data = geneticDataFrame)

# uncomment that following part to see the summary of the regression
# summary(reg)
```

All values are NA (Not Available) because we have too much predictors (more than individuals). The X matrix is cannot be reversed so we have to use the PCA method to counter that problem.

a) 1 point
b) 1 point
c) 1 point
d) 1 point + 0.5 bonus

Very nice final investigation on the ideas of percentage of explained variance

3.PCA Principal Component Analysis

a)

PCA is a multivariate statistical technique, the main goal of this method is to reduce the number of predictors. In order to achieve that, we have to extract the important information from the geneticData set. These new variables are linear combinaisons of the original ones. The number of principal components is smaller than the number of original variables. The aim is to maximize the variation of the geneticData set.

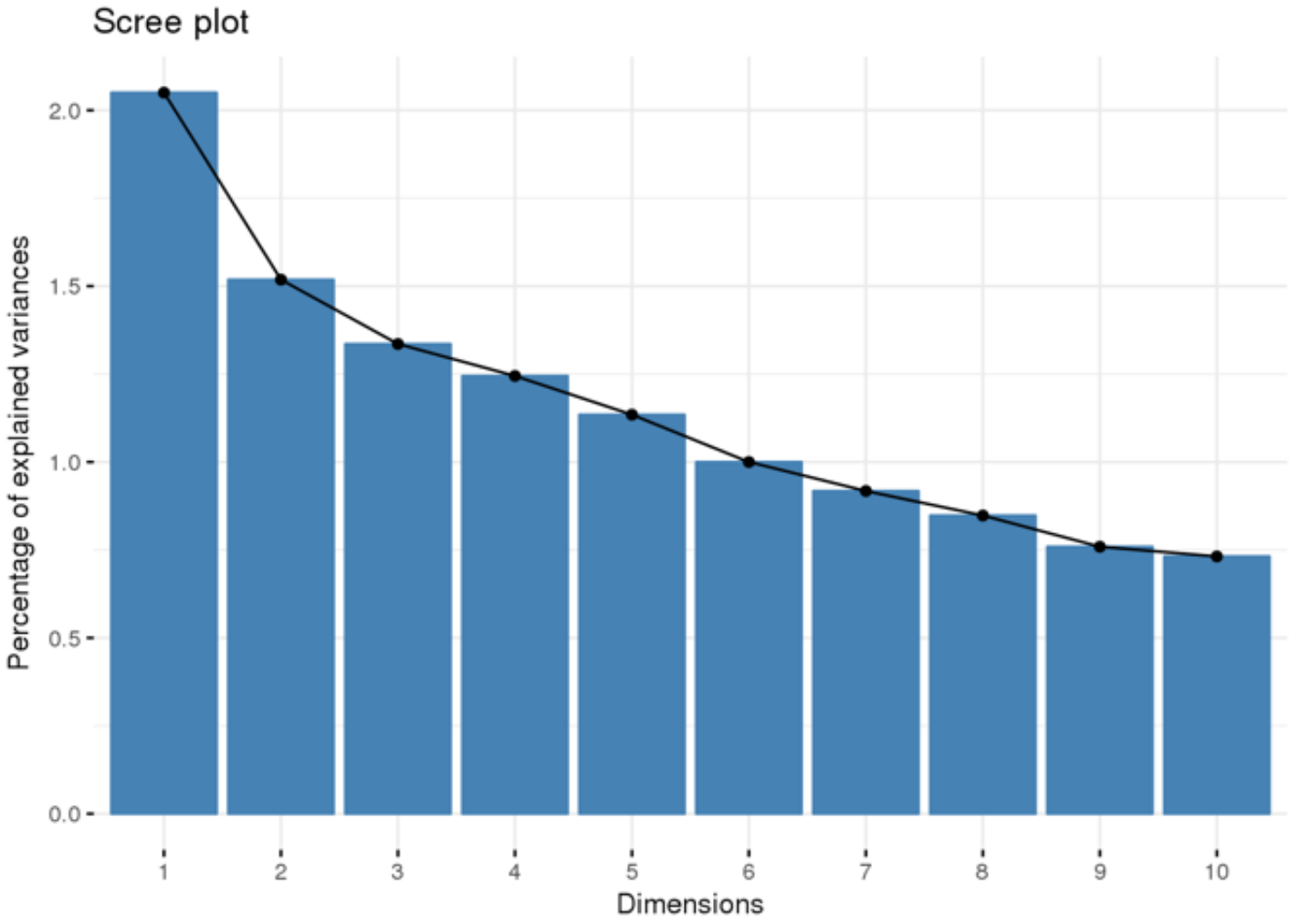
b)

```
# install.packages("factoextra")

library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFCZ
```

```
allGenes <- geneticDataFrame[, -1]

pcaNAm2 <- prcomp(allGenes, scale = FALSE)

fviz_eig(pcaNAm2)
```



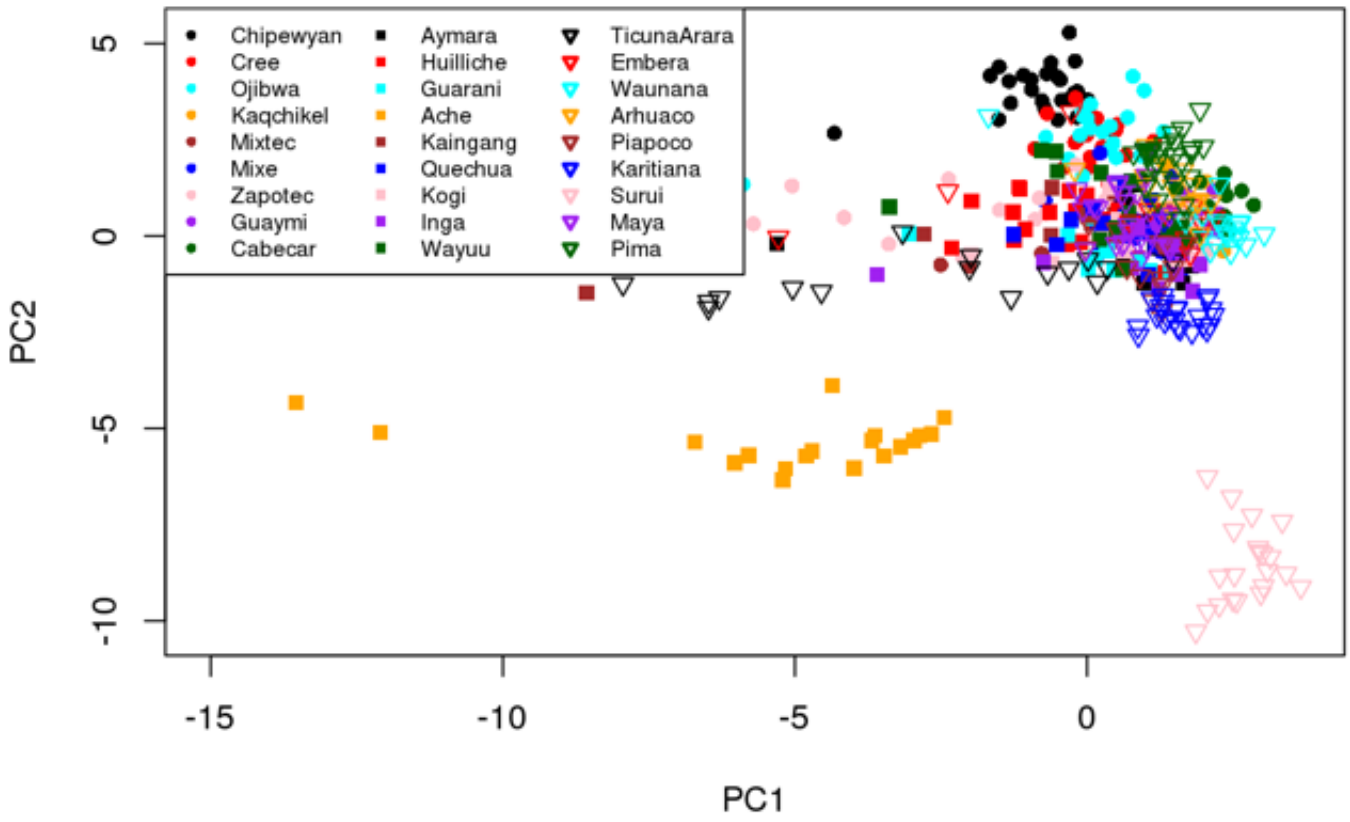

```
vectProportionOfVariance <- pcaNAm2$sdev^2/sum(pcaNAm2$sdev^2)

# summary(pcaNAm2.pca)
```

It is better to use scale = FALSE by convention because we get a better representation of the percentage of explained values. As we work on genes and every component are composed of 1 and 0, vectors are already normed.

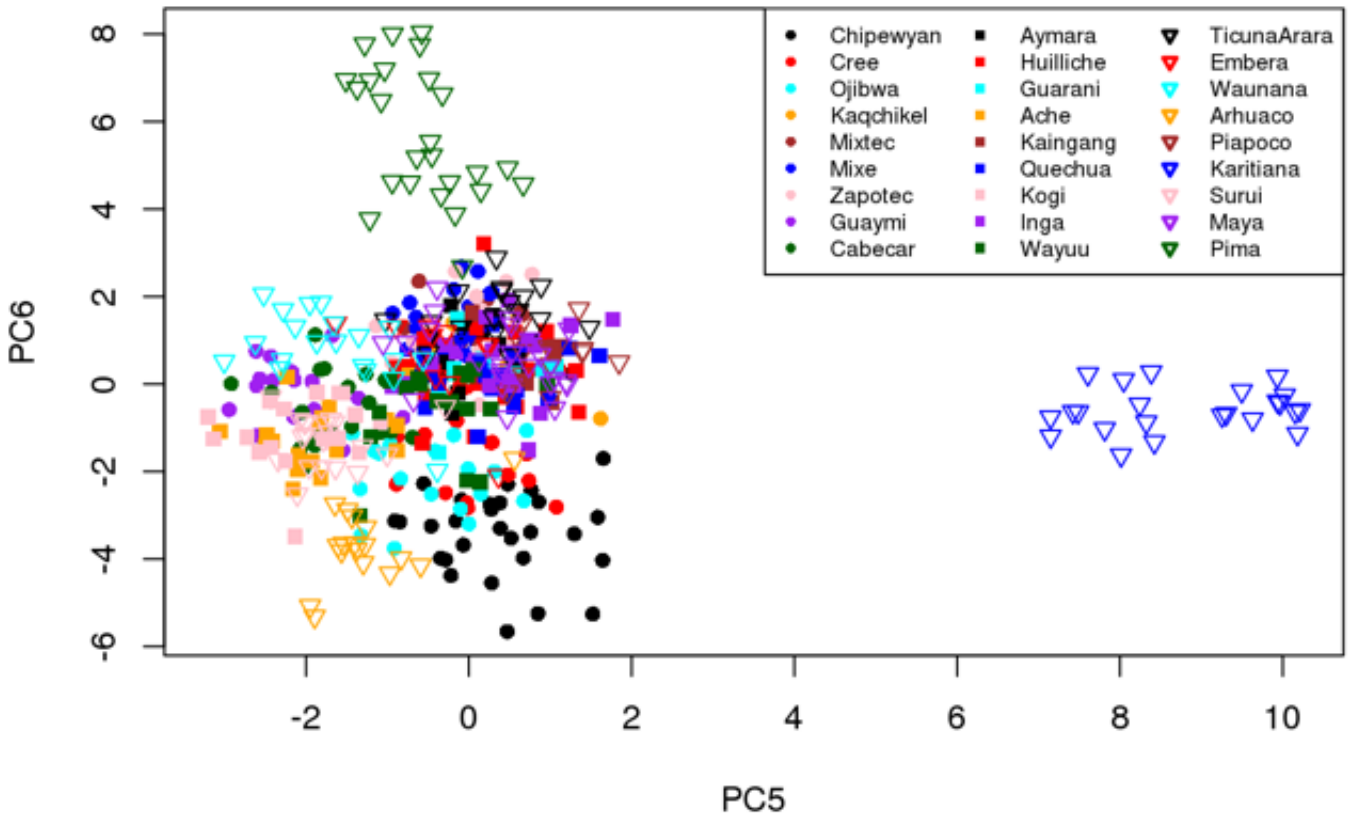
We notice that we get only 494 principal components instead of approximately 5000 which correspond to the number of genes. This can be explained because the PCA make linear combinaisons to turn 5000 components to 494.

c)

```
caxes=c(1,2)
plot(pcaNAm2$x[,caxes],col="white")

for (i in 1:npop) {
  # print(names[i])
  lines(pcaNAm2$x[which(NAm2[,3]==names[i]),caxes],
  type="p",col=colPalette[i],pch=pch[i])
}
legend("topleft",legend=names,col=colPalette,lty=-
1,pch=pch,cex=.65,ncol=3,lwd=2)
```



```
caxes=c(5,6)
plot(pcaNAm2$x[,caxes],col="white")

for (i in 1:npop) {
  # print(names[i])
  lines(pcaNAm2$x[which(NAm2[,3]==names[i]),caxes],
  type="p",col=colPalette[i],pch=pch[i])
}
legend("topright",legend=names,col=colPalette,lty=-
1,pch=pch,cex=.65,ncol=3,lwd=2)
```



The Ache population is well identified by both components. The Surui can also be identified by both components but more especially by the PC1.

The Karitiana population is well identified by the PC5 and the Pima population is well identified by the PC6.

d)

```
inertia = vectProportionOfVariance[1] + vectProportionOfVariance[2]

sprintf("The inertia is equal to %f.", inertia)
```

```
## [1] "The inertia is equal to 0.035684."
```

According to what we have seen until now, we have two ideas.

The first idea is that we have seen that we can distinguish two population using two principal components. Therefore, as there are 27 different populations, we could deduce that we would need 14 pairs of principal components.

However, we can ask ourselves if we can really each time distinguish two different populations using two principal components. It might not be the case because we have just tested it with two pairs of principal components. Thus, we could think about an alternative way, which is to add the variance of the first principal components until the sum is equal to a certain level. This level could be 25% or 50%. This last one means using a lot of principal components.

```
numberOfPCs <- function(vectorProportionOfVariance, percentage) {
  tmp = 0.0
  i = 1
  while (tmp < percentage) {
    tmp = tmp + vectProportionOfVariance[i]
    i = i + 1
  }
  return(i)
}

nbrPC25 <- numberOfPCs(vectProportionOfVariance, 0.25)
nbrPC50 <- numberOfPCs(vectProportionOfVariance, 0.5)
nbrPC75 <- numberOfPCs(vectProportionOfVariance, 0.75)
nbrPC99 <- numberOfPCs(vectProportionOfVariance, 0.99)

sprintf("Number of principal components needed in order to cover at least 25 percent
of variance : %i", nbrPC25)
```

```
## [1] "Number of principal components needed in order to cover at least 25 percent o
f variance : 40"
```

```
sprintf("Number of principal components needed in order to cover at least 50 percent
of variance : %i", nbrPC50)
```

```
## [1] "Number of principal components needed in order to cover at least 50 percent o
f variance : 120"
```

```
sprintf("Number of principal components needed in order to cover at least 75 percent
of variance : %i", nbrPC75)
```

```
## [1] "Number of principal components needed in order to cover at least 75 percent o
f variance : 240"
```

```
sprintf("Number of principal components needed in order to cover at least 99 percent
of variance : %i", nbrPC99)
```

```
## [1] "Number of principal components needed in order to cover at least 99 percent o
f variance : 473"
```

With the test we have established just before we can see that less than $1/10$ of the data contains 25 of the variance : So it seems like a good idea to use only 50 PC to get most of the information within least data. So we might use that number of PC in order to represent genetic markers. The other numbers give a idea of the number needed if we want to cover more information.

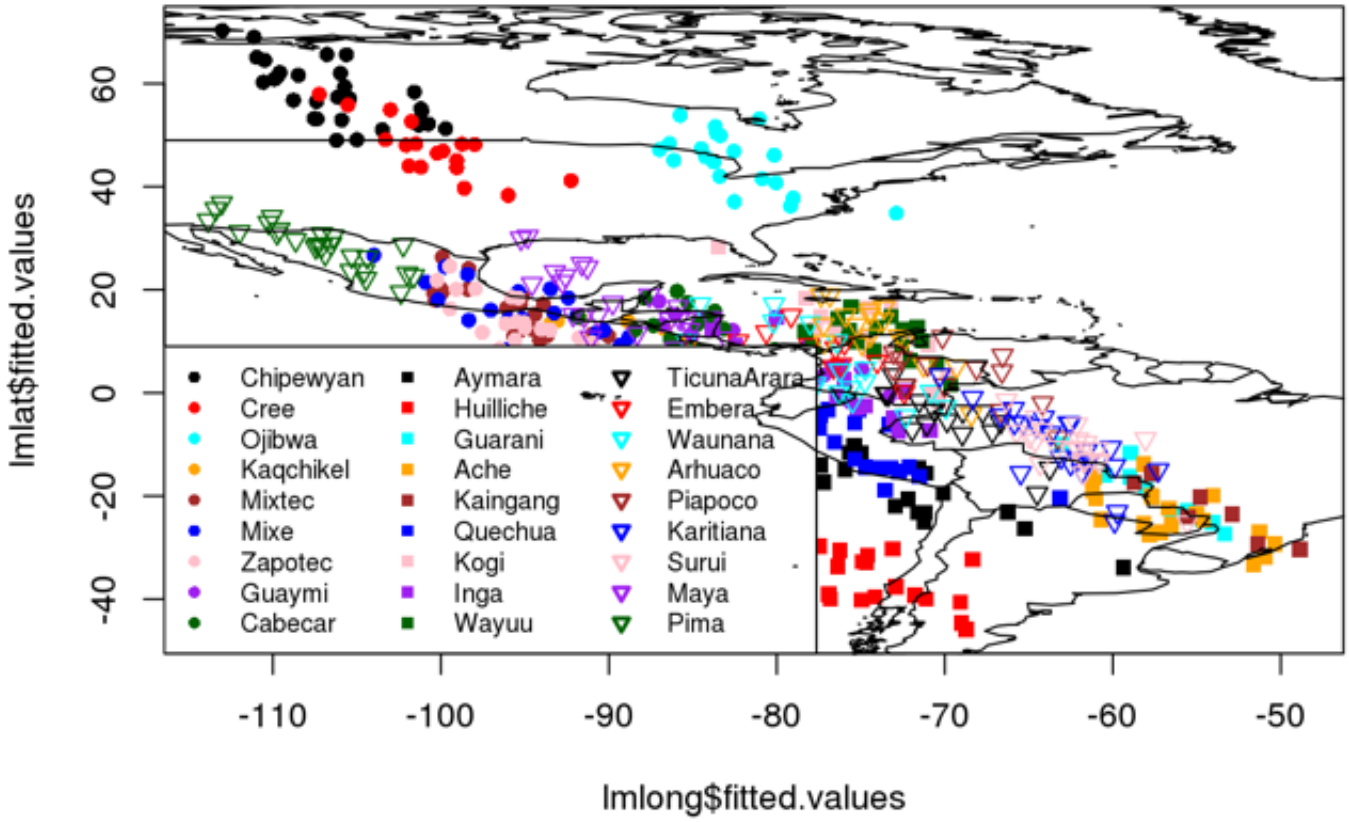
4. PCR Principal Components Regression

a)

a) 1.5 out of 3.0 points
b) 1 point

It was not really this the idea with the question of whether the estimation was too optimistic or not ! In fact, the problem is that we use the same dataset to estimate the model and also evaluate the error

```
# pcaNAm2$x are scores : size 494 * 494 matrix. We take all the ligns but only 250 co
lumns
first250PCs <- pcaNAm2$x[, c(1:250)]

# size 494 * 251 => first column become the longitude
long <- c(NAm2$long, first250PCs)
longMatrix <- matrix(data = long, nrow = 494, ncol = 251)
dataFrameLong <- data.frame(longMatrix)
#dataFrameLong
lmlong = lm(formula = X1 ~ . , data = dataFrameLong)

# size 494 * 251 => first column become the latitude
lat <- c(NAm2$lat, first250PCs)
latMatrix <- matrix(data = lat, nrow = 494, ncol = 251)
dataFrameLat <- data.frame(latMatrix)
lmlat = lm(formula = X1 ~ . , data = dataFrameLat)

plot(lmlong$fitted.values,lmlat$fitted.values,col="white")

for (i in 1:npop) {
  #print(names[i])
  lines(lmlong$fitted.values[which(NAm2[,3]==names[i])],
        lmlat$fitted.values[which(NAm2[,3]==names[i])],
        type="p",col=colPalette[i],pch=pch[i])
}

legend("bottomleft",legend=names,col=colPalette,lty=-1,pch=pch,cex=.75,ncol=3,lwd=2)
library(maps);map("world",add=T)
```



If we compare it to the map from the question 1, we can clearly say that this map represent very optimistically places where populations live. It looks quite messy if we just take a look around Panama. However, if we examine the estimated places of the Chipewyan, Cree, Ojibwa, Pima and Huiliche populations, we can very distinctly see that the estimation is pretty accurate.

b)

```
# import the library needed to use rdist.earth
library("fields")
```

```
## Loading required package: spam
```

```
## Loading required package: dotCall64
```

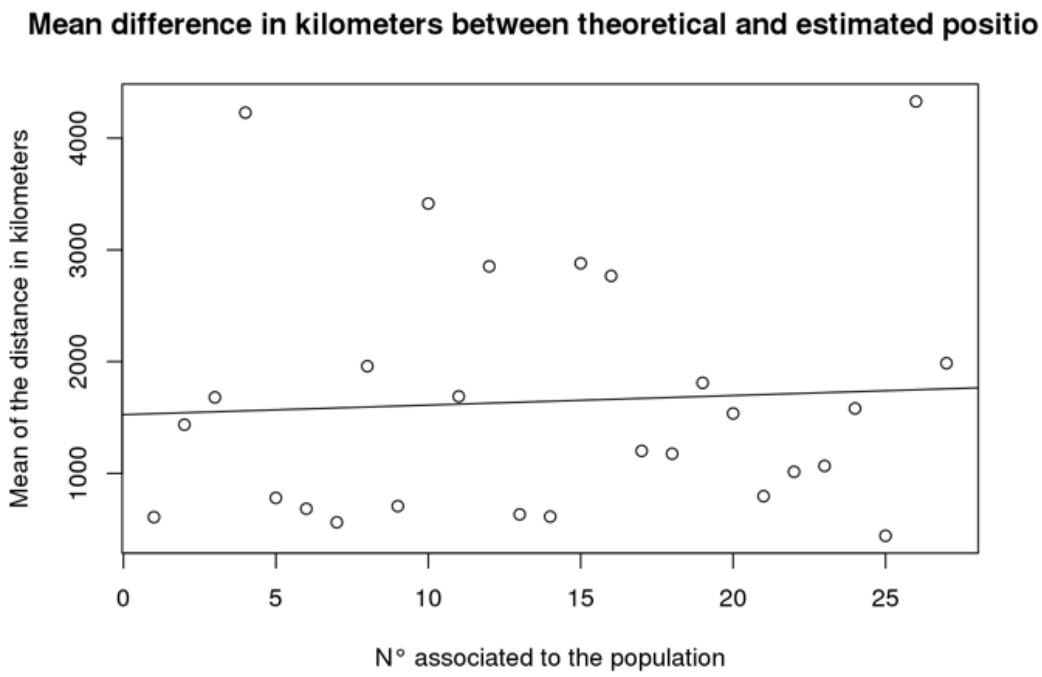
```
## Loading required package: grid
```

```
## Spam version 2.1-1 (2017-07-02) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.
```

```
##
## Attaching package: 'spam'
```

```
## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve
```

```
# select the theory latitude and longitude of populations
theoryPosition <- matrix(data = c(NAm2$long, NAm2$lat), nrow = 494, ncol = 2)
#theoryPosition

# vector used to stock the mean of distance for each population
stock <- c()

tmpIndex <- 0
for (i in 1:npop) {
  # vector long estimated for population n??i
  tmpLong <- lmlong$fitted.values[which(NAm2[,3]==names[i])]

  # vector lat estimated for population n??i
  tmpLat <- lmlat$fitted.values[which(NAm2[,3]==names[i])]

  # matrix of the estimated positions
  estimatedPosition <- matrix(data = c(tmpLong, tmpLat), ncol = 2)
  #print(estimatedPosition)

  len <- length(estimatedPosition)/2

  # by making unique we assume that two estimated coordinates will never be at the sa
me exact position

  diff <- rdist.earth(x1 = theoryPosition[tmpIndex:(tmpIndex+len), ], x2 = estimatedP
osition, miles = F)[1, ]

  # stock the mean of distance from the theoretical position in kilometers
  stock[i] <- mean(diff)

  tmpIndex <- tmpIndex + len
}

plot(c(1:npop), stock, main = "Mean difference in kilometers between theoretical and
estimated positions",
     xlab = "N° associated to the population",
     ylab = "Mean of the distance in kilometers")
linearTest <- lm(stock ~ c(1:npop))
abline(linearTest)
```



Thanks to the previous graph, we can say that the estimated positions are on average approximately 1100 kilometers away from the theoretical positions. It might seem a lot at first sight because it sounds huge if we just have a human sensible approach. Nonetheless, it is not really a lot in reality if we size to the scale of the earth. Indeed, we could just think about the fact that the earth's circonference is equal to 40075 kilometers. If we draw a circle around the theoretical position for each population, on average, the estimation of the position will be contained inside the circle.

a) 1 point
b) 2 points
c) 1.5 point
d) 1 point

We would have liked to see the comparison of the training error and testing error as well

5) PCR and Cross-Validation

a)

Cross-Validation is a technique used in model selection to better estimate the test error of a predictive model. The principle of K-fold cross validation method is to divide randomly the data into K subsets where K-1 subsets are used as the training sets and the remaining subset is used as a validation subset to compute a prediction error. We repeat this procedure K times by considering each subset as the validation subset and the other subsets as the training sets. The procedure is finished when each subset has been used once as the validation subset. The K results from the folds can then be averaged to produce a single estimation.

The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

```
labels <- rep(1:10, each=10)
set <- sample(labels, 494, replace = TRUE)
```

b)

```
# we define naxes to 4 as a variable so that it will be easier to make the final algo
rithm
naxes <- 4
predictedCoord <- matrix(data = NA, nrow = 494, ncol = 2)

# Regression for the longitude
#pcalong <- data.frame(cbind(long=NAm2[,c("long")], pcaNAm2$x[,1:naxes], set))
pcalong = data.frame(cbind(long=NAm2[,c("long")],pcaNAm2$x))
subsetpcalong <- subset(pcalong, set != 1)
reglong <- lm(subsetpcalong[, 1] ~ . , data = subsetpcalong[, 2:naxes])
#reglong <- lm(formula = subsetpcalong[,1] ~ . , data = subset(pcalong[, 1:(naxes+1)]
, set != 1))

# Regression for the latitude
pcalat = data.frame(cbind(lat=NAm2[,c("lat")],pcaNAm2$x))
subsetpcalat <- subset(pcalat, set != 1)
reglat <- lm(subsetpcalat[, 1] ~ . , data = subsetpcalat[, 2:naxes])
#reglat <- lm(formula = lat ~ . , data = subset(pcalat[, 1:(naxes+1)], set != 1))


# Prediction
# i = 1
predict_long <- predict(reglong, newdata=data.frame(subset(pcalong, set == 1)))
predict_lat <- predict(reglat, newdata=data.frame(subset(pcalat, set == 1)))

idx = 1
for (indice in 1:494) {

  if (set[indice] == 1) {

    predictedCoord[indice, 1] = predict_long[idx]
    predictedCoord[indice, 2] = predict_lat[idx]
    idx = idx + 1
  }

}

# predictedCoord
```

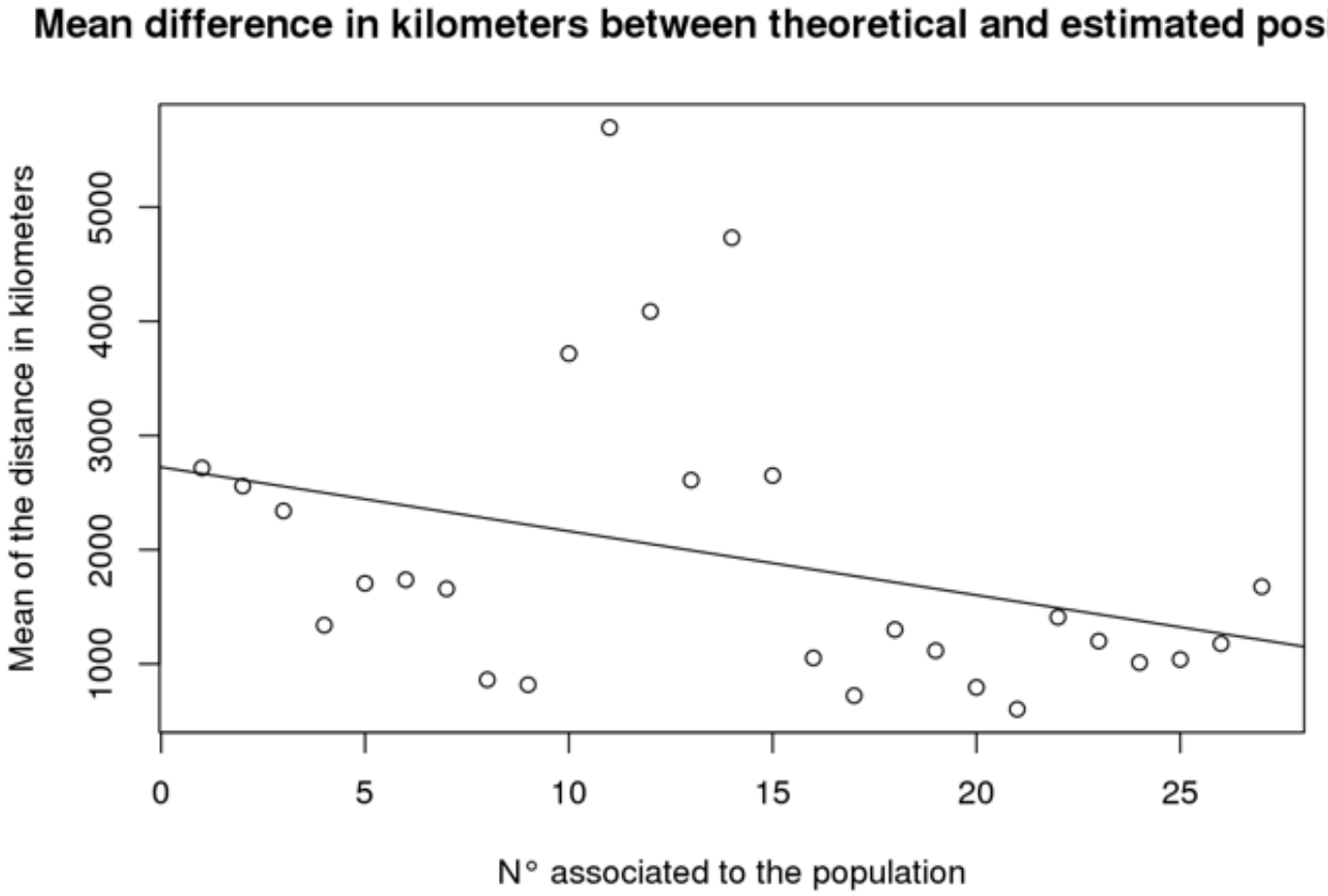
We predict only the coordinates where the lines are equals to 1 in the set.

```
for(i in 2:10){

    #reglong <- lm(formula = long ~ . , data = subset(pcalong[, 1:(naxes+1)], set != i))
    #reglat <- lm(formula = lat ~ . , data = subset(pcalat[, 1:(naxes+1)], set != i))

    subsetpcalong <- subset(pcalong, set != i)
    reglong <- lm(subsetpcalong[, 1] ~ . , data = subsetpcalong[, 2:naxes])

    subsetpcalat <- subset(pcalat, set != i)
    reglat <- lm(subsetpcalat[, 1] ~ . , data = subsetpcalat[, 2:naxes])

    predict_long <- predict(reglong, newdata=data.frame(subset(pcalong, set == i)))
    predict_lat <- predict(reglat, newdata=data.frame(subset(pcalat, set == i)))

    idx = 1
    for (j in 1:494) {

      if (set[j] == i){

        predictedCoord[j, 1] <- predict_long[idx]
        predictedCoord[j, 2] <- predict_lat[idx]
        idx = idx + 1

      }

    }

}

# just to check if at that step we've got something that makes sense
#data.frame(predictedCoord)
#data.frame(theoryPosition)
```

We can clearly see that, even when we change naxes, we have the predictedCoord that makes sense compared to the theoryPosition.

```
dist_earth <- rdist.earth.vec(x1 = predictedCoord, x2 = theoryPosition, miles = F)

vect <- dist_earth
#mean(dist_earth)
# mean(vect[1:30]) # moyenne ecart pour la premiere population
```

```
# we get the index where population change. The length of index_vect is npop = 27
index_vect <- c(1:npop)

indice_main_vect <- 1
index = 1
tmp = NAm2$Pop[1]
for (popName in NAm2$Pop) {

  string = popName

  if (string != tmp) {
    index_vect[indice_main_vect] <- index
    indice_main_vect <- indice_main_vect + 1
  }

  tmp = string
  index = index + 1
}

# Now we calculate the average difference infor each population

index_vect[npop] <- 494
main_vect <- c(1:npop)
indice1 <- 1
for (indice in 1:npop) {

  indice2 <- index_vect[indice]
  main_vect[indice] <- mean(vect[indice1:indice2])
  indice1 <- indice2

}

stock <- main_vect
#stock

plot(c(1:npop), stock, main = "Mean difference in kilometers between theoretical and
estimated positions",
     xlab = "N° associated to the population",
     ylab = "Mean of the distance in kilometers")
linearTest <- lm(stock ~ c(1:npop))
abline(linearTest)
```

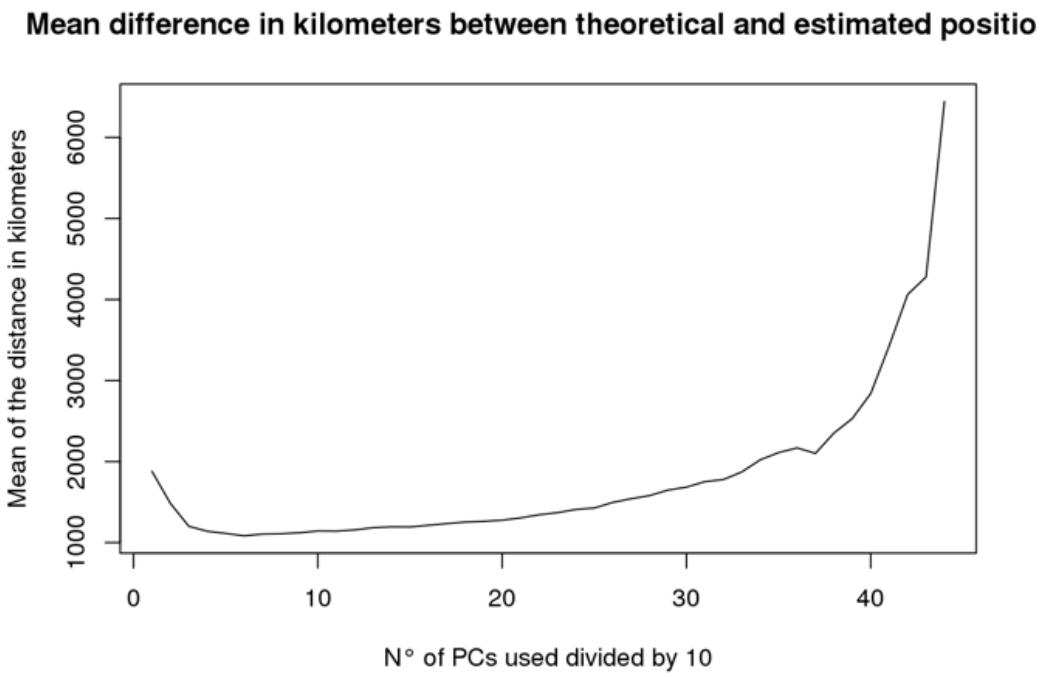


Here we can see that by using the 4 first principal components, we have a 2000 kilometers difference. We expect that as we increase the number of principal components used, the model is better and predict more precisely the theoretical coordinates.

c)

```
mean_final <- c(1:44)

index <- 1
for (naxes in seq(4,440, by = 10)){

      predictedCoord <- matrix(data = NA, nrow = 494, ncol = 2)

      for(i in 1:10){

      #reglong <- lm(formula = long ~ . , data = subset(pcalong[, 1:(naxes+1)], set !
= i))
      #reglat <- lm(formula = lat ~ . , data = subset(pcalat[, 1:(naxes+1)], set != i
))
```

```
      subsetpcalong <- subset(pcalong, set != i)
      reglong <- lm(subsetpcalong[, 1] ~ . , data = subsetpcalong[, 2:naxes])

      subsetpcalat <- subset(pcalat, set != i)
      reglat <- lm(subsetpcalat[, 1] ~ . , data = subsetpcalat[, 2:naxes])

      predict_long <- predict(reglong, newdata=data.frame(subset(pcalong, set == i)))
      predict_lat <- predict(reglat, newdata=data.frame(subset(pcalat, set == i)))

      idx = 1
      for (j in 1:494) {

        if (set[j] == i){

          predictedCoord[j, 1] <- predict_long[idx]
          predictedCoord[j, 2] <- predict_lat[idx]
          idx = idx + 1

        }

      }

  }

  dist_earth <- rdist.earth.vec(x1 = theoryPosition, x2 = predictedCoord, miles = F)
  mean_final[index] <- mean(dist_earth)
  index <- index + 1

}


plot(c(1:(index-1)), mean_final, main = "Mean difference in kilometers between theore
tical and estimated positions",xlab = "N° of PCs used divided by 10", ylab = "Mean of
the distance in kilometers", type = "l")
```



As we can see, the final graph shows that as we increase the number of principal components used for the regression at the beginning, the difference of distance decreases. This is quiet expected because we have more information to describe the model. However, when the number of principal components used is too large, it increases the difference : this is caused by overfitting. There is the limit where the model explains the errors that we do not want the model to explain. If we compare to the result obtained in the answer 4.b, we can see in abscissa 25 that we have a difference of distance equal to 1300. This result is really close to what we have found before 1100 so that makes sense.

According to this graph, we should use the first 80 principal components to get the best prediction because this is approximately the minimum of the curve. This is the model we would keep.

2 points

6) Conclusion

The idea of principal components regression (PCR) is to calculate the principal components and then use some of these components as predictors in a linear regression model fitted using the typical least squares procedure.

By using PCR we can easily perform dimensionality reduction on a high dimensional dataset and then fit a linear regression model to a smaller set of variables, while at the same time keep most of the variability of the original predictors.

Since the use of only some of the principal components reduces the number of variables in the model, this can help in reducing the model complexity, which is always good. In case we need many principal components to explain most of the variability in our data, say roughly as many principal components as the number of variables in our dataset, then PCR might not perform that well in that scenario. PCR tends to perform well when the first principal components are sufficient to explain most of the variation in the predictors.

Thus, PCR is a good approach with this study because previously we have seen that by using the first 80 principal components, we had the best way to predict the geographical origin of an individual. Then, the number of principal components is not high compared to the number of variables in our dataset.

Moreover, we have seen through this work that the more we increase the number of principal components, the better is the model and predict more precisely the theoretical coordinates. However there is a limit because of overfitting value, so now we know that we have to pay attention to that.

In this study, all the variables are genes, maybe with a study where variables do not represent the same thing, the PCR could be not enough well. For example, if there is multicollinearity between variables, that could be a problem.