Ex MDS

Vicent Caselles Ballester

2024-04-17

Exercici 1

```
dat <- matrix(c(0, 7, 5, 9, 5, 7, 9,
                7, 0, 4, 6, 4, 6, 7,
                5, 4, 0, 3, 4, 5, 6,
                9, 6, 3, 0, 3, 2, 2,
                5, 4, 4, 3, 0, 5, 4,
                7, 6, 5, 2, 5, 0, 4,
                9, 7, 6, 2, 4, 4, 0), ncol=7)
all(dat == t(dat)) # to make sure I haven't made mistakes
```

```
## [1] TRUE
```

a) Construir la matriz B $= -\frac{1}{2}HD^{(2)}H$, donde $D^{(2)}$ es la matriz de distancias al cuadrado y H es la matriz de centrado, y calcular sus valores propios. Observar si la matriz de distancias es euclídea.

From [1] :

> **Theorem**: Let D be a distance matrix and define K by (2). Then D is Euclidean if and only if K is positive semi-definite.

K is the Kernel matrix, refered to as B in Everitt, 2011. Therefore, let's use some of the functions I've created to check if a symmetric matrix is positive definite.

```
source('../../funcs/sym.R')
require(ade4); require(matrixcalc)
```

```
## Loading required package: ade4
```

```
## Loading required package: matrixcalc
```

```
positive_definite(dat)
```

```
## [1] FALSE
```

```
is.positive.semi.definite(dat)
```

```
## [1] FALSE
```

```
is.euclid(as.dist(dat))
```

```
## [1] FALSE
```

With this proof, I think we can be somewhat sure that the distances are not euclidean.

[1] https://www.math.uwaterloo.ca/ aghodsib/courses/f10stat946/notes/lec10-11.pdf

About the first part, we can calculate B using some functions I've written.

```
source('../../funcs/MDS.R')
D <- dat
B <- KernelMatrix(D**2)
B2 <- B_from_D(D)

max(abs(B-B2)) # We see that both functions using different approaches accomplish the same
```

```
## [1] 7.105427e-15
```

Eigenvalues of B:

```
evB <- eigen(B)$values
sum(evB < 0)
```

```
## [1] 2
```

We see that two eigenvalues of B are < 0, therefore B is not positive semi-definite, therefore D is not euclidean.

b) Obtener la representación con las dos primeras coordenadas principales e indicar el grado de bondad de esta representación. Se puede hacer a partir de la descomposición de la matriz B o con la función `cmdscale`.

I'm gonna choose the first option. We select fist two eigen{values, vectors} pairs of B.

```
evecs_2 <- eigen(B)$vectors[,1:2]
evalues_2 <- diag(sqrt(eigen(B)$values[1:2]))
new_X <- evecs_2%*%evalues_2

max(abs(new_X - cmdscale(dat, k=2)))
```

```
## [1] 5.107026e-15
```

Let's check how different the new euclidean distances are using the reduced data matrix (`new_X`).

```
max(abs(dat - as.matrix(dist(new_X))))
```

```
## [1] 2.934933
```

We see that there is a sufficiently enough large distance difference between the points in the original space and the ones in the new reduced space.

Using the criteria from Everitt 2011:

```
sum(eigen(B)$values[1:2])/sum(eigen(B)$values[eigen(B)$values>0])
```

```
## [1] 0.7881461
```

They also mention the following criteria in case B is not positive definite:

```
sum(abs(eigen(B)$values[1:2]))/sum(abs(eigen(B)$values))
```

```
## [1] 0.7062874
```

```
sum(eigen(B)$values[1:2] ** 2)/sum(eigen(B)$values ** 2)
```

```
## [1] 0.9135125
```

We see that the results differ a lot.

Exercici 2

Poner un ejemplo para comprobar que el escalado multidimensional clásico aplicado a las distancias euclídeas calculadas sobre una matriz de datos multivariantes X es equivalente a la solución que se obtiene por el análisis de componentes principales de la matriz de covarianzas de X.

```
data(crabs, package='MASS')
X <- data.matrix(crabs[, 4:8])
```

Theoretically, PCA analysis and MDS are equivalent as long as you select the k largest eigen{values, vectors} when constructing the new X.

```
require(matrixcalc)
D <- as.matrix(dist(X))
B <- B_from_D(D)
evalues <- eigen(B)$values
evectors <- eigen(B)$vectors
k <- min(matrix.rank(t(X)%*%X), sum(evalues>0))
```

```
k_evectors <- evectors[, 1:k, drop=FALSE]
k_evalues <- evalues[1:k]

new_X_MDS <- k_evectors%*%diag(sqrt(k_evalues))
max(abs(as.matrix(dist(new_X_MDS)) - D)) # cool
```

```
## [1] 1.070255e-13
```

```
new_X_PCA <- prcomp(X)$x
max(abs(new_X_PCA - new_X_MDS))
```

```
## [1] 3.02109e-13
```

Cool.

Exercici 3

Too long to paste here

First let's define the data.

```
X <- matrix(c(0.21, 0.06, 0.06, 0.67,
              0.25, 0.04, 0.14, 0.57,
              0.22, 0.06, 0.08, 0.64,
              0.19, 0.04, 0.02, 0.75,
              0.18, 0, 0.15, 0.67,
              0.23, 0, 0.28, 0.49,
              0.3, 0, 0.06, 0.64,
              0.1, 0.06, 0.13, 0.71,
              0.27, 0.04, 0.06, 0.63,
              0.21, 0.05, 0.2, 0.54), ncol = 4, byrow=T)
```

a) Obtener las distancias de Bhattacharyya según la fórmula 1.

Function definition:

```
bhattacharyya <- function(X){
  # X: data matrix
  n <- nrow(X)
  D <- matrix(0, nrow=n, ncol=n)
```

```
  for (i in 1:(n-1)){
    for (j in (i+1):n){
      pi <- X[i,]
      pj <- X[j,]
      D[i,j] <- acos(sum(sqrt(pi*pj)))
      D[j,i] <- D[i,j]
    }
  }
  D
}
```

Calculation on our data:

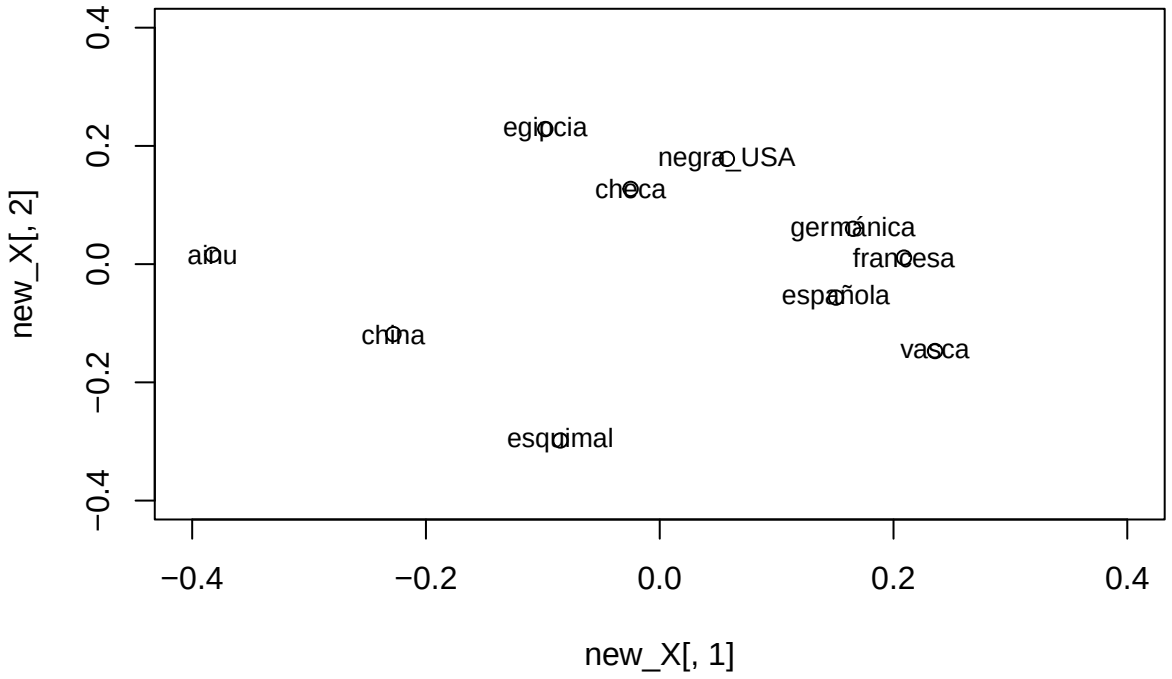
```
D_bhat <- bhattacharyya(X)
```

Same result as the one on the solution, albeit different methodology.

b) Representar estas poblaciones con las dos primeras coordenadas principales. ¿Se observa algún tipo de agrupación?

```
B <- B_from_D(sqrt(D_bhat))
evalues <- eigen(B)$values[1:2]
evectors <- eigen(B)$vectors[, 1:2]

new_X <- evectors %*% diag(sqrt(evalues))
plot(new_X[, 1], new_X[, 2], ylim=c(-0.4, 0.4), xlim = c(-0.4, 0.4))
labels = c("francesa","checa","germánica","vasca","china",
           "ainu","esquimal","negra_USA","española","egipcia")
text(new_X[, 1], new_X[, 2], labels=labels,cex=0.8)
```



c) ¿Es ésta una distancia euclídea? ¿Cuál es la dimensión de la representación euclídea? Determinar el porcentaje de variabilidad explicado por las dos primeras coordenadas principales.

Following the same logic as before:

```
require(ade4)
is.positive.semi.definite(B)
```

```
## [1] TRUE
```

```
positive_definite(B)
```

```
## [1] TRUE
```

```
is.euclid(as.dist(sqrt(D_bhat)))
```

```
## [1] TRUE
```

```
sum(eigen(B)$values<0)
```

```
## [1] 0
```

Therefore, we can almost 100% certainly conclude that B is positive semi-definite. The dimension of the original space of the data is 4.

Second part of the question, same solution as before:

```
sum(eigen(B)$values[1:2])/sum(eigen(B)$values) * 100
```

```
## [1] 56.58885
```

Exercici 4

```
X <- matrix(c(1,1,0,0,1,1,
              1,1,1,0,0,1,
              1,0,0,1,0,1,
              0,0,0,0,1,0),
            ncol=6, nrow=4, byrow=T)

individuals <- c("A", "B", "C", "D")
```

a) Calcular los coeficientes de Sokal y Michener para cada par de individuos y obtener la matriz de distancias asociada.

```
sokal <- function(X){
  n <- nrow(X)
  SOKAL = matrix(0, ncol=n, nrow=n)
  for (i in 1:(n-1)){
    for (j in (i+1):n){
      ind1 <- X[i,]
      ind2 <- X[j,]
      summy <- ind1 + ind2
      a <- sum(summy == 2)
      d <- sum(summy == 0)
      p <- length(ind1)
      SOKAL[i, j] <- SOKAL[j,i] <- (a+d) / p
    }
  }
  SOKAL
}
```

```
S <- sokal(X)
```

Checking everything is ok.

```
SM.simil <- function(x,y){
 if (length(x) != length(y)){
 stop("los vectores han de tener la misma longitud")}
  else {
 a <- sum(x==1 & y==1)
 d <- sum(x==0 & y==0)
 (a+d)/length(x) }
}

bin.simil <- function(X,simil){
 n <- nrow(X)
 S <- matrix(0,nrow=n,ncol=n)
 for(i in 1:(n-1)){
   for(j in (i+1):n) {
    S[i,j]<-S[j,i] <- simil(X[i,], X[j,])
   }
 }
 S
}

all.equal(bin.simil(X, SM.simil), S)
```

```
## [1] TRUE
```

```
D_from_S <- function(S){
  # S is a matrix of similitudes, i.e. SOKAL or JACCARD indices between pairs
  # of observations
  n <- nrow(S) # assumes S is square
  2 - 2*S
}

D <- D_from_S(S)
as.dist(D) # Same as teacher
```

```
##           1         2         3
## 2 0.6666667
## 3 1.0000000 1.0000000
## 4 1.0000000 1.6666667 1.3333333
```

b) Idem with Jaccard

```
jaccard <- function(X){
  n <- nrow(X)
  JACKIE <- matrix(0, ncol=n, nrow=n)
  for (i in 1:(n-1)){
    for (j in (i+1):n){
      ind1 <- X[i,]
      ind2 <- X[j,]
      summy <- ind1 + ind2
      a <- sum(summy == 2)
      d <- sum(summy == 0)
```

```
      p <- length(ind1)
      JACKIE[i, j] <- JACKIE[j,i] <- a / (p - d)
    }
  }
  JACKIE
}

S <- jaccard(X)
D <- D_from_S(S)
as.dist(D) # same as teacher
```

```
##     1   2   3
## 2 0.8
## 3 1.2 1.2
## 4 1.5 2.0 2.0
```

Exercici 5

```
require(cluster)
```

```
## Loading required package: cluster
```

```
data(flower )
flower
```

```
##    V1 V2 V3 V4 V5 V6  V7 V8
## 1   0  1  1  4  3 15  25 15
## 2   1  0  0  2  1  3 150 50
## 3   0  1  0  3  3  1 150 50
## 4   0  0  1  4  2 16 125 50
## 5   0  1  0  5  2  2  20 15
## 6   0  1  0  4  3 12  50 40
## 7   0  0  0  4  3 13  40 20
## 8   0  0  1  2  2  7 100 15
## 9   1  1  0  3  1  4  25 15
## 10  1  1  0  5  2 14 100 60
## 11  1  1  1  5  3  8  45 10
## 12  1  1  1  1  2  9  90 25
## 13  1  1  0  1  2  6  20 10
## 14  1  1  1  4  2 11  80 30
## 15  1  0  0  3  2 10  40 20
## 16  1  0  0  4  2 18 200 60
## 17  1  0  0  2  2 17 150 60
## 18  0  0  1  2  1  5  25 10
```

```
daisy(flower, metric='gower')
```

```
## Dissimilarities :
##           1         2         3         4         5         6         7
## 2  0.8875408
## 3  0.5272467 0.5147059
## 4  0.3517974 0.5504493 0.5651552
## 5  0.4115605 0.6226307 0.3726307 0.6383578
## 6  0.2269199 0.6606209 0.3003268 0.4189951 0.3443627
## 7  0.2876225 0.5999183 0.4896242 0.3435866 0.4197712 0.1892974
```

```
## 8  0.4234069 0.4641340 0.6038399 0.2960376 0.4673203 0.5714869 0.4107843
## 9  0.5808824 0.4316585 0.4463644 0.8076797 0.3306781 0.5136846 0.5890931
## 10 0.6094363 0.4531046 0.4678105 0.5570670 0.3812908 0.4119281 0.5865196
## 11 0.3278595 0.7096814 0.5993873 0.6518791 0.3864788 0.4828840 0.5652369
## 12 0.4267565 0.5857843 0.6004902 0.5132761 0.5000817 0.5248366 0.6391340
## 13 0.5196487 0.5248366 0.5395425 0.7464461 0.2919118 0.4524510 0.5278595
## 14 0.2926062 0.5949346 0.6096405 0.3680147 0.5203431 0.3656863 0.5049837
## 15 0.6221814 0.3903595 0.5300654 0.5531454 0.4602124 0.5091503 0.3345588
## 16 0.6935866 0.3575163 0.6222222 0.3417892 0.7301471 0.5107843 0.4353758
## 17 0.7765114 0.1904412 0.5801471 0.4247141 0.6880719 0.5937092 0.5183007
## 18 0.4610294 0.4515114 0.7162173 0.4378268 0.4755310 0.6438317 0.4692402
##           8         9        10        11        12        13        14
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9  0.6366422
## 10 0.6639706 0.4256127
## 11 0.4955474 0.4308007 0.3948121
## 12 0.4216503 0.4194036 0.3812092 0.2636029
## 13 0.5754085 0.2181781 0.3643791 0.3445670 0.2331699
## 14 0.4558007 0.4396650 0.3609477 0.2838644 0.1591503 0.3784314
## 15 0.4512255 0.2545343 0.4210784 0.4806781 0.4295752 0.3183007 0.4351307
## 16 0.6378268 0.6494690 0.3488562 0.7436683 0.6050654 0.5882353 0.4598039
## 17 0.4707516 0.6073938 0.3067810 0.7015931 0.5629902 0.5461601 0.5427288
## 18 0.1417892 0.5198529 0.8057598 0.5359477 0.5495507 0.5733252 0.5698121
##          15        16        17
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16 0.3949346
## 17 0.3528595 0.1670752
## 18 0.5096814 0.7796160 0.6125408
##
## Metric :  mixed ;  Types = N, N, N, N, O, O, I, I
## Number of objects : 18
```

```r
# TODO
gower_flower <- function(x, y){
  nums_x <- x[7:8]; nums_y <- y[7:8]
```

```
  bin_as_x <- x[4]; bin_as_y <- y[4]
}
```