Prova d'avaluació continuada 1

Vicent Caselles Ballester

2024-05-07

Contents

Exercici 1

Carrego les dades i utilitzo `str` per a fer un resum del número d'observacions, número i tipus de variables.

```
library(boot)
data(urine)
str(urine)
```

```
## 'data.frame':    79 obs. of  7 variables:
##  $ r      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gravity: num  1.02 1.02 1.01 1.01 1 ...
##  $ ph     : num  4.91 5.74 7.2 5.51 6.52 5.27 5.62 5.67 5.41 6.13 ...
##  $ osmo   : num  725 577 321 408 187 ...
##  $ cond   : num  NA 20 14.9 12.6 7.5 25.3 17.4 35.9 21.9 25.7 ...
```

```
##  $ urea   : num  443 296 101 224 91 252 195 550 170 382 ...
##  $ calc   : num  2.45 4.49 2.36 2.15 1.16 3.34 1.4 8.48 1.16 2.21 ...
```

Com podem veure, i tal i com es diu a l'enunciat de l'exercici, hi ha 79 observacions.

a)

Per a veure quantes observacions amb `NA`s hi ha al *dataframe* (*df*) i a quines variables corresponen, podem utilitzar la funció `is.na.dataframe`. Aquesta torna un *df* booleà amb les mateixes dimensions que el *df* original. Utilitzant `sum` podem trobar quants `TRUE` hi han en aquest nou *df* – que es correspondràn al número de `NA`s.

```
sum(is.na.data.frame(urine))
```

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

Veiem que hi han dues instàncies de `NA` al *df* `urine`. Això també ho podriem haver fet amb la funció `summary`[1].

```
summary(urine)
```

```
##        r             gravity            ph             osmo
##  Min.   :0.0000   Min.   :1.005   Min.   :4.760   Min.   : 187.0
##  1st Qu.:0.0000   1st Qu.:1.012   1st Qu.:5.530   1st Qu.: 411.5
##  Median :0.0000   Median :1.018   Median :5.940   Median : 612.5
##  Mean   :0.4304   Mean   :1.018   Mean   :6.028   Mean   : 615.0
##  3rd Qu.:1.0000   3rd Qu.:1.024   3rd Qu.:6.385   3rd Qu.: 797.5
##  Max.   :1.0000   Max.   :1.040   Max.   :7.940   Max.   :1236.0
##                                                   NA's   :1
##       cond            urea            calc
##  Min.   : 5.10   Min.   : 10.0   Min.   : 0.170
##  1st Qu.:14.38   1st Qu.:160.0   1st Qu.: 1.460
##  Median :21.40   Median :260.0   Median : 3.160
##  Mean   :20.90   Mean   :266.4   Mean   : 4.139
##  3rd Qu.:26.77   3rd Qu.:372.0   3rd Qu.: 5.930
##  Max.   :38.00   Max.   :620.0   Max.   :14.340
##  NA's   :1
```

Per veure a quines observacions (files) i variables (columnes) corresponen aquests `NA`, utilitzem `which` (dient-li que retorni els índex de l'*array* o *df* original).

```
which(is.na.data.frame(urine), arr.ind = T)
```

```
##    row col
## 55  55   4
## 1    1   5
```

Veiem que aquests corresponen a la variable `osmo` observació 55 i a la variable `cond` observació número 1. Veiem que això es correspon als resultats obtinguts amb `summary` (en quant a les variables a les quals corresponen).

Ja que es menciona el tutorial del *footnote* anterior a l'enunciat, vaig a utilitzar-lo per a entendre els *missing data* de `urine`. Anem a mirar quin percentatge de dades *missing* tenim al nostre *dataset* (copiat descaradament del tutorial anterior).

```
pMiss <- function(x){sum(is.na(x))/length(x)*100}
apply(urine,2,pMiss) # cap variable supera el threshold de 5% mencionat a l'article
```

```
##        r  gravity       ph     osmo     cond     urea     calc
```

[1] https://www.r-bloggers.com/2015/10/imputing-missing-data-with-r-mice-package/

```
## 0.000000 0.000000 0.000000 1.265823 1.265823 0.000000 0.000000
```

```
sort(apply(urine,1,pMiss), T)[1:4] # en canvi, si que hi ha observacions que ho superen
```

```
##        1       55        2        3
## 14.28571 14.28571  0.00000  0.00000
```

Veiem que correspon a les observacions que hem dit anteriorment. Per a dur a terme la imputació de les dades fem servir la funció `mice` del paquet homònim tal i com es menciona a l'article abans referenciat.

Em costa interpretar si, quan es diu a l'enunciat

> Realizaremos m = 50 imputaciones y calcularemos la mediana del conjunto de valores.

vol dir que hem d'establir el paràmetre `m=50` o `maxit=50`. Entenc que es refereix a `m`, així que això faré.

```
require(mice)
imputed_urine <- mice(urine, m = 50, seed = 123, method='pmm', print=F)
```

```
## Warning: Number of logged events: 500
```

Accedim a les dades imputades per a les dues variables així:

```
imputed_osmo <- imputed_urine$imp$osmo
imputed_cond <- imputed_urine$imp$cond
```

Si recordem, allà on hi havia valors `NA` eren els següents indexos:

```
which(is.na.data.frame(urine), arr.ind = T)
```

```
##    row col
## 55  55   4
## 1    1   5
```

La columna 4 correspon a la variable:

```
colnames(urine)[4]
```

```
## [1] "osmo"
```

Per tant, anem a substituir aquests elements de la matriu de dades amb les medianes tal i com es demana.

```
urine[55, 4] <- median(as.matrix(imputed_osmo))
urine[1, 5] <- median(as.matrix(imputed_cond))
```

Comprovem que ho haguem fet bé.

```
sum(is.na.data.frame(urine))
```

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

Perfecte.

b)

Per a fer el que es demana, i sense utilitzar cap *for loop*, ho podem fer amb màscares booleanes. Primer de tot, calculo el vector de mitjanes per les dades corresponents a aquells casos on no hi ha presència de cristalls d'oxalat de calci (`urine$r == 0`).

```
colMeans(urine[urine$r==0, ])
```

```
##          r    gravity         ph       osmo       cond       urea       calc
##   0.000000   1.015489   6.098667 565.288889  20.486667 237.111111   2.624889
```

Podem observar que, la mitjana de la variable `r` és, efectivament, 0 (això és indicador de que la màscara booleana ha funcionat). Fem el mateix però per a aquells casos on si hi ha cristalls.

```
colMeans(urine[urine$r==1,])
```

```
##          r    gravity         ph       osmo       cond       urea       calc
##   1.000000   1.021588   5.935588 684.794118  21.355882 305.176471   6.142941
```

Veiem que les diferències més grans són a la variable `osmo` (osmolaritat – concentració de molècules/soluts? de la urina), a la variable `urea` (concentració d'urea a l'urina), i la variable `calc` (concentració de calci); totes aquestes variables tenen mitjanes majors al grup 1, fets que considero més que lògics.

En quant a les matrius de covariança, ho faig a continuació. Substrec la informació respecte a la variable `r` ja que és realment un factor o variable categòrica.

```
round(cov(urine[urine$r==0, ])[-1, -1], 6)
```

```
##           gravity         ph         osmo        cond         urea       calc
## gravity  0.000038  -0.001099     1.372106    0.040754     0.659763   0.004676
## ph      -0.001099   0.492857   -43.236879   -0.661905   -33.211439  -0.180200
## osmo     1.372106 -43.236879 54369.982828 1790.101667 24393.421717 201.101510
## cond     0.040754  -0.661905  1790.101667   76.751182   555.803788   7.380385
## urea     0.659763 -33.211439 24393.421717  555.803788 15062.782828 102.333535
## calc     0.004676  -0.180200   201.101510    7.380385   102.333535   3.470739
```

```
round(cov(urine[urine$r==1,])[-1,-1], 6)
```

```
##           gravity         ph         osmo        cond         urea       calc
## gravity  0.000052  -0.001091     1.279973    0.018821     0.726166   0.010497
## ph      -0.001091   0.567262   -26.348815   -0.487291   -11.123440  -0.089914
## osmo     1.279973 -26.348815 52098.653298 1184.902763 28115.643494 442.655472
## cond     0.018821  -0.487291  1184.902763   45.077692   464.177718   9.649528
## urea     0.726166 -11.123440 28115.643494  464.177718 17917.483066 231.071283
## calc     0.010497  -0.089914   442.655472    9.649528   231.071283  13.229270
```

A primera vista, no sabria dir que un grup presenti més variabilitat que l'altre. Vaig a intentar esbrinar més calculant la variança total i la variança generalitzada. Primer de tot calculem la variança total ($tr(S)$).

```
sum(diag(cov(urine[urine$r==0, ])[-1, -1]))
```

```
## [1] 69513.48
```

```
# comprovació
stopifnot(abs(sum(diag(cov(urine[urine$r==0, ])[-1, -1])) -
            sum(eigen(cov(urine[urine$r==0, ])[-1, -1])$values)) < 1e-10)
```

Per el grup amb cristalls d'oxalat de calci.

```
sum(diag(cov(urine[urine$r==1,])[-1,-1]))
```

```
## [1] 70075.01
```

```
# comprovació
stopifnot(abs(sum(diag(cov(urine[urine$r==1, ])[-1, -1])) -
            sum(eigen(cov(urine[urine$r==1, ])[-1, -1])$values)) < 1e-10)
```

Ara anem a calcular la variança generalitzada ($|\ S\ |$).

```
det(cov(urine[urine$r==0,])[-1,-1])
```

```
## [1] 748.4151
```

```r
way2 <- prod(eigen(cov(urine[urine$r==0,])[-1,-1])$values)
stopifnot(abs(way2-det(cov(urine[urine$r==0,])[-1,-1])) < 1e-6)
```

Per al grup amb cristalls:

```r
det(cov(urine[urine$r==1,])[-1,-1])
```

```
## [1] 51350.41
```

```r
way2 <- prod(eigen(cov(urine[urine$r==1,])[-1,-1])$values)
stopifnot(abs(way2-det(cov(urine[urine$r==1,])[-1,-1])) < 1e-6)
```

Com veiem, la variabilitat generalitzada del grup amb cristalls és bastant més alta que la del grup que no en presenta Aquests detalls no són tan clars amb una inspecció *a ull nu* de la matriu de variàncies-covariàncies, ni tampoc són tan evidents amb el càlcul de la variació total com amb el càlcul del determinant de la matriu de covariàncies. Això, encara que no tinc clar els detalls, entenc que és degut a que les covariàncies entre les variables no estan siguent comptades per al càlcul de la variació total (ja que és la traça de la matriu de variances-covariàncies $\boldsymbol{S}$). Al grup amb presència de cristalls, segurament hi ha variables que augmenten o disminueixen en conjunció de forma molt més pronunciada, fet que és observable amb el còmput de la variança generalitzada i no amb el de la variança total.

c)

Aquest tema em sembla força interessant, ja que és un algoritme d'optimització – minimització de la distància entre tots els punts i el punt a $\mathbb{R}^n$ representat per aquesta mediana. Buscant una mica per internet, he trobat el següent paquet escrit en C^{++}, anomenat `Gmedian`[2], que utilitza *Stochastic Gradient Descent* (*SGD*) per a solucionar-ho. Tot i això, m'havia descarregat el paquet i entès com funcionava però m'he adonat que utilitza la norma 1 – L_1 norm – en comptes de la L_2, que és la que volem. He intentat adaptar el codi de C^{++} a `R` i canviant el tipus de *norma* però no convergeix de la mateixa manera que un altre paquet que he trobat, anomenat `bigutilsr`, que utilitza un altre tipus de lògica per a arribar al resultat òptim; així doncs, utilitzo aquest últim paquet [3].

```r
# install.packages('bigutilsr')
require(bigutilsr)
X <- as.matrix(urine[, -1])
geometric_median(X)
```

```
##    gravity         ph       osmo       cond       urea       calc
##   1.018245   6.062341 628.549815  21.836844 266.949964   4.533956
```

Aquesta funció té un paràmetre que permet *splittejar* la matriu de dades d'acord a un vector, el qual marco per a que sigui la variable `r`.

```r
geometric_median(urine, by_grp = urine$r)
```

```
##   r  gravity       ph     osmo     cond     urea     calc
## 0 0 1.015329 6.114315 560.0419 20.86302 228.2203 2.454072
## 1 1 1.022498 6.185119 723.2479 23.48124 318.6518 7.491731
```

Podem observar que, a diferència del que passava si comparavem els vectors de mitjanes, amb els vectors de medianes *totes* les variables mostren una mediana major en el cas de les observacions amb presència de cristalls d'oxalat de calci (amb les mitjanes, la variable `ph` tenia una mitjana menor en aquest grup – fet que no passa amb les medianes). Anem a explorar-ho més comparant la distància euclídea entre els vectors de mitjanes per grup i els vectors de medianes geomètriques per grup.

[2] https://github.com/cran/Gmedian/blob/master/src/Gmedian.cpp

[3] Tot i així, podeu veure el codi on faig proves amb aquest tema a https://github.com/vcasellesb/analisis-multi/blob/main/funcs/Gmedian.R

```r
meanvec0 <- colMeans(urine[urine$r == 0, -1])
meanvec1 <- colMeans(urine[urine$r==1, -1])
medianvec0 <- geometric_median(urine[urine$r==0, -1])
medianvec1 <- geometric_median(urine[urine$r==1, -1])
```

Un cop tenim construïts els vectors que volem, procedem a fer els càlculs.

```r
eucldistmean <- sqrt(sum((meanvec0 - meanvec1)**2)) # distància euclídea entre
# els vectors amb les mitjanes
eucldistmedian <- sqrt(sum((medianvec0 - medianvec1)**2))
```

La distància entre els vectors mitjana és `137.58`, mentre que en el cas dels vectors de les medianes geomètriques és de `186.67`.

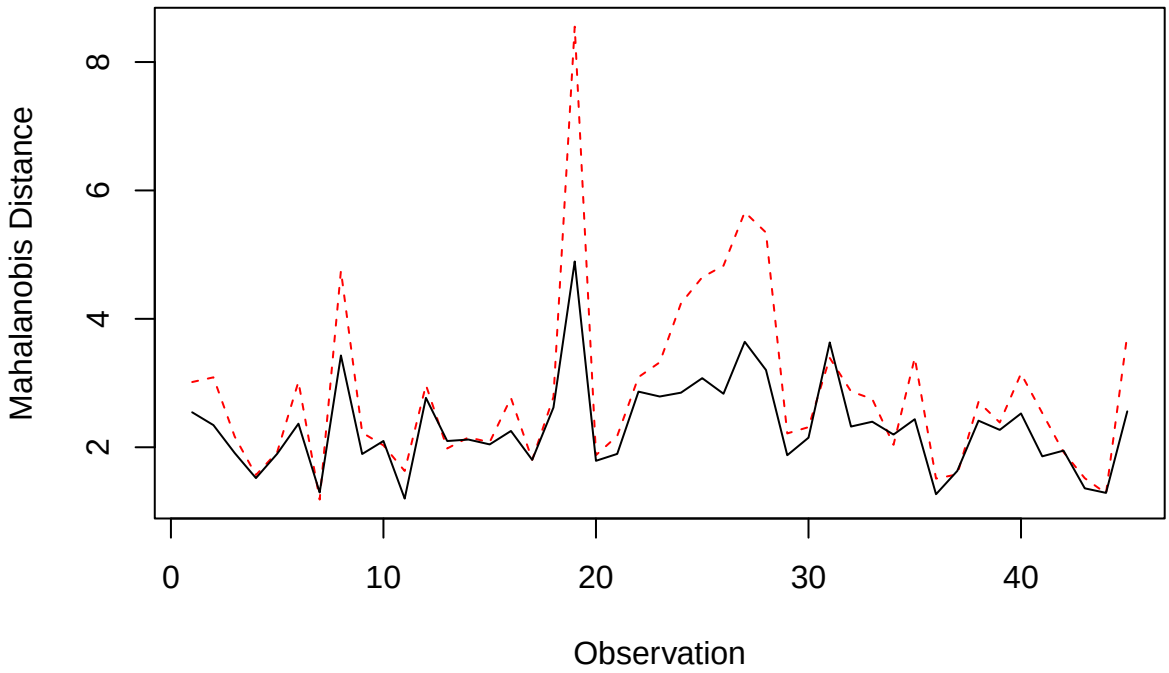
d)

Per a estudiar la presència de dades atípiques en el nostre conjunt de dades, em basaré amb l'apartat d) de l'exercici 15 dels exercicis de R1 (Estadística Descriptiva). Primer de tot ho durem a terme per a les dades corresponents a les observacions sense cristalls d'oxalat de calci.

```r
X_NC <- as.matrix(urine[urine$r == 0, -1]) # X NO CALCI
# Calculem les distàncies de mahalanobis a les mitjanes i matrius de covariança
# "no robustes"
d2m_NC <- mahalanobis(X_NC, colMeans(X_NC), cov(X_NC))

# A continuació fem el mateix pero amb les mitjanes i cov-var matrix robustes
set.seed(123)
rob_cov_NC <- MASS::cov.rob(X_NC, method='mcd')
d2m_rob_NC <- mahalanobis(X_NC, rob_cov_NC$center, rob_cov_NC$cov)

# Grafiquem el resultat
plot(sqrt(d2m_rob_NC), col="red", typ="l", ylab="Mahalanobis Distance",
     xlab="Observation",lty=2)
lines(sqrt(d2m_NC), typ="l")
```



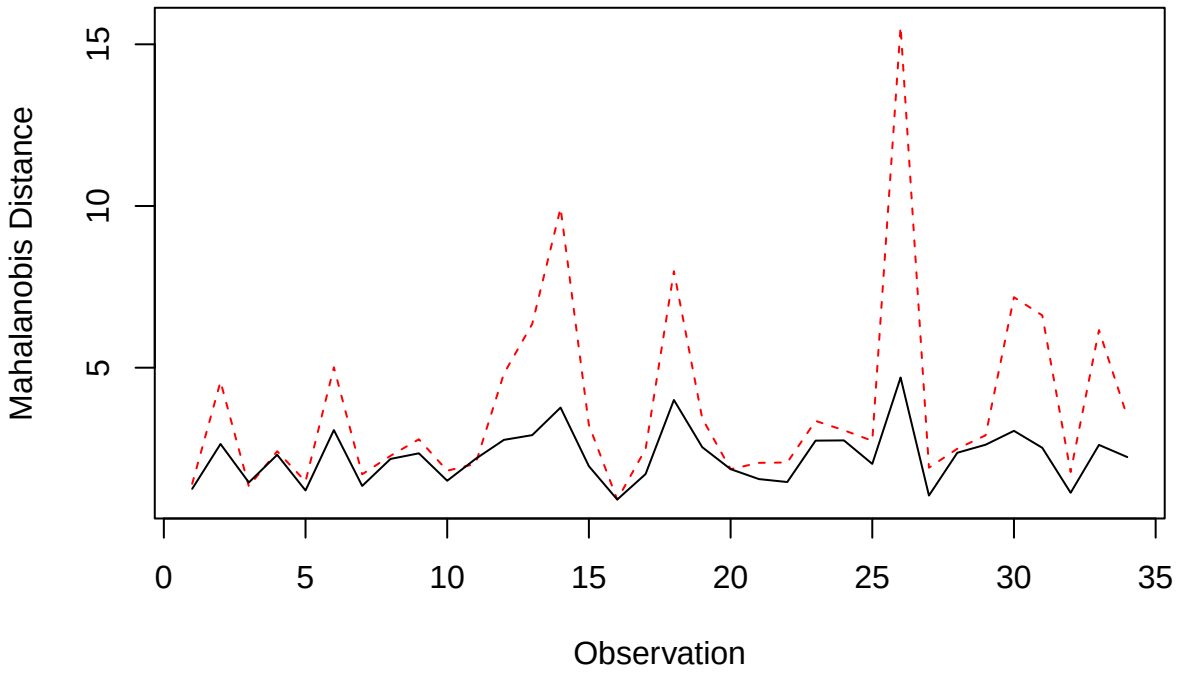
Com podem observar, les distàncies de mahalanobis són majors en el cas d'utilitzar les estimacions *robustes* de la matriu de covariances i el vector $\boldsymbol{\mu}$.

Anem a fer el mateix per el cas dels pacients amb cristalls d'oxalat de calci.

```
X_C <- as.matrix(urine[urine$r == 1, -1]) # X CALCI
# Calculem les distàncies de mahalanobis a les mitjanes i matrius de covariança
# "no robustes"
d2m_C <- mahalanobis(X_C, colMeans(X_C), cov(X_C))

# A continuació fem el mateix pero amb les mitjanes i cov-var matrix robustes
set.seed(123)
rob_cov_C <- MASS::cov.rob(X_C, method='mcd')
d2m_rob_C <- mahalanobis(X_C, rob_cov_C$center, rob_cov_C$cov)

# Grafiquem el resultat
plot(sqrt(d2m_rob_C), col="red", typ="l", ylab="Mahalanobis Distance",
     xlab="Observation",lty=2)
lines(sqrt(d2m_C), typ="l")
```

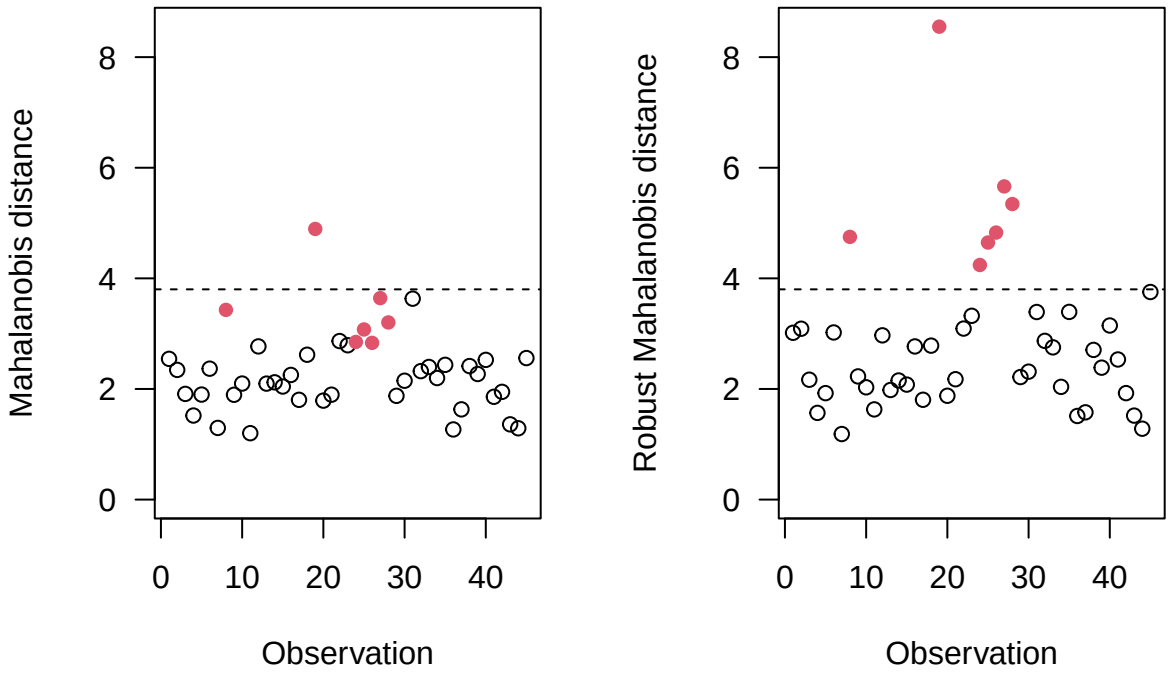


Veiem que, un altre cop, la línea vermella, corresponent a les distàncies calculades amb el vector de mitjanes i matriu de covariances robustes són generalment majors que les distàncies calculades de forma *no robusta*.

A continuació, faig un gràfic on mostro, *side by side*, les observacions que serien considerats *outliers* o atípics tenint en compte les distàncies calculades utilitzant el vector $\boldsymbol{\mu}$ i la matriu de covariances Σ de forma *no robusta* i *robusta*, respectivament. En vermell, podem observar aquelles observacions que, tenint en compte l'arrel quadrat del quantil 0.975 de la distribució $\tilde{\chi}^2$, serien considerades com a *outliers* tenint en compte les estimacions robustes mencionades anteriorment.

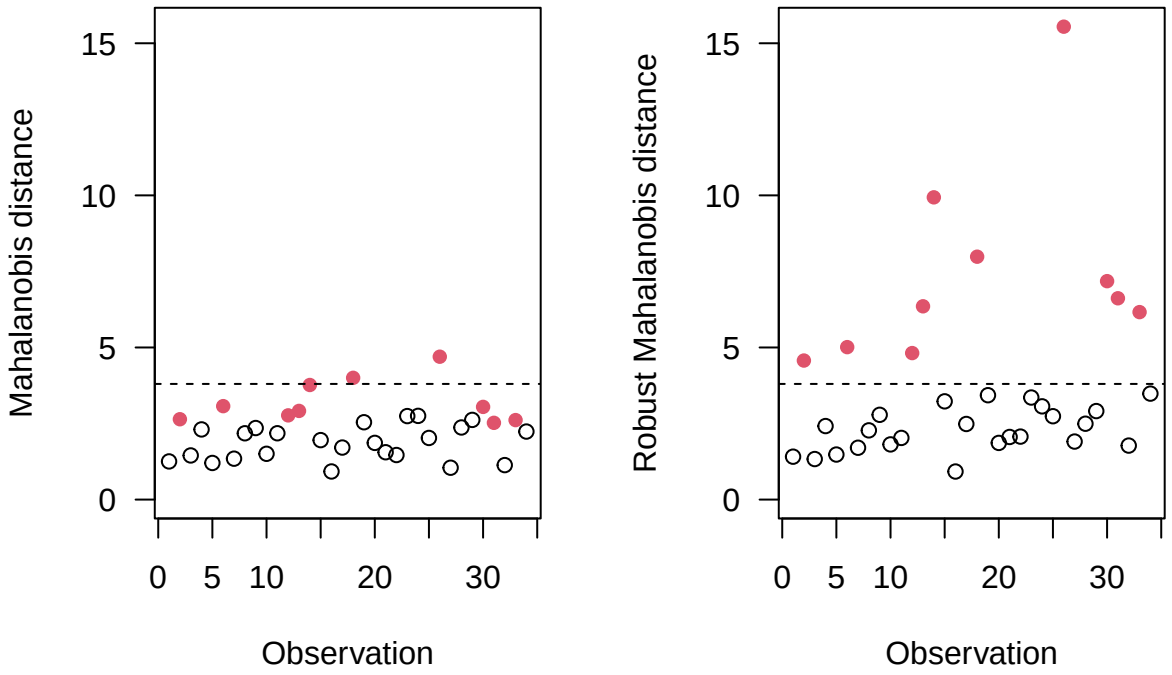
Primer de tot es mostra el gràfic per a pacients sense cristalls d'oxalat de calci.

```
maha1_NC <- sqrt(d2m_NC)
maha2_NC <- sqrt(d2m_rob_NC)
max_maha_NC <- max(sqrt(d2m_NC), sqrt(d2m_rob_NC))
thr <- sqrt(qchisq(0.975, df=6))
outliers <- sqrt(d2m_rob_NC) > thr
par(mfrow = c(1, 2), las = 1)
plot(sqrt(d2m_NC), xlab = "Observation" ,ylab = "Mahalanobis distance",
     ylim = c(0, max_maha_NC), col = outliers + 1, pch = 15 * outliers + 1)
abline(h = thr,lty=2)
plot(sqrt(d2m_rob_NC), xlab = "Observation", ylab = "Robust Mahalanobis distance",
     ylim = c(0, max_maha_NC), col = outliers + 1, pch = 15 * outliers + 1)
abline(h = thr,lty=2)
```



Així doncs, veiem que hi ha unes ≈ 7 observacions atípiques. A continuació grafico els resultats per al grup que presenta cristalls d'oxalat de calci.

```
maha1_C <- sqrt(d2m_C)
maha2_C <- sqrt(d2m_rob_C)
max_maha_C <- max(sqrt(d2m_C), sqrt(d2m_rob_C))
thr <- sqrt(qchisq(0.975, df=6))
outliers <- sqrt(d2m_rob_C) > thr
par(mfrow = c(1, 2), las = 1)
plot(sqrt(d2m_C), xlab = "Observation" ,ylab = "Mahalanobis distance",
     ylim = c(0, max_maha_C), col = outliers + 1, pch = 15 * outliers + 1)
abline(h = thr,lty=2)
plot(sqrt(d2m_rob_C), xlab = "Observation", ylab = "Robust Mahalanobis distance",
     ylim = c(0, max_maha_C), col = outliers + 1, pch = 15 * outliers + 1)
abline(h = thr,lty=2)
```



Observem que, encara que el número d'observacions atípiques en el cas de l'estimació *no robusta* és el doble, mentre que les observacions atípiques en el cas de les estimacions robustes és de 10 (3 més que en el cas dels pacients sense cristalls de calci). També, observant les escales d'aquests últims gràfics (casos robustos), podem determinar que els *outliers* corresponents al grup amb presència de cristalls d'oxalat de calci mostren unes distàncies de mahalanobis majors que no pas les del grup sense cristalls (el màxim en el segon cas – `NC` – no arriba a ≈ 10 mentre que en el primer cas – `C` – n'hi ha dues que superen 10).

e)

Escric a continuació una funció que duu a terme el que es demana.

```r
pooledS <- function(X, fac)
{
  n <- nrow(X)
  groups <- unique(fac)
  k <- length(groups)
  Sp <- 0

  i <- which(colSums(sweep(X, 1, fac, "==")) == nrow(X)) # THIS IS SO DUMB

  for (g in groups){
    Xi <- X[fac == g, , drop=FALSE]
    Si <- cov(Xi[, -i])
    ni <- nrow(Xi)
    Sp <- Sp + (ni - 1) * Si
  }
  Sp <- Sp * 1 / (n - k)
}
```

Per a comprovar si ho he fet bé, vaig a utilitzar dues funcions que he trobat que pretenen calcular aquest estadístic. La primera candidata indica que ho he fet malament.

```r
# install.packages('vcvComp')
library(vcvComp)
```

```r
cov.W(X=as.matrix(urine), urine$r)[-1, -1]
```

```
##               [,1]          [,2]         [,3]          [,4]          [,5]
## [1,]  4.474498e-05  -0.001094959     1.326039    0.02978754     0.6929642
## [2,] -1.094959e-03   0.530059519   -34.792847   -0.57459799   -22.1674398
## [3,]  1.326039e+00 -34.792846702 53234.318063 1487.50221480 26254.5326055
## [4,]  2.978754e-02  -0.574597995  1487.502215   60.91443672   509.9907531
## [5,]  6.929642e-01 -22.167439840 26254.532605  509.99075312 16490.1329471
## [6,]  7.586864e-03  -0.135057028   321.878491    8.51495624   166.7024094
##               [,6]
## [1,]   0.007586864
## [2,]  -0.135057028
## [3,] 321.878491236
## [4,]   8.514956239
## [5,] 166.702409388
## [6,]   8.350004534
```

Però, si utilitzo aquest altre paquet (`Morpho`), si que obtinc els mateixos resultats.

```r
# install.packages('Morpho')
library(Morpho)

covW(as.matrix(urine), urine$r)[-1, -1]
```

```
##               gravity            ph        osmo          cond          urea
## gravity  0.0000437335  -0.001095486     1.33262    0.03135423     0.6882211
## ph      -0.0010954861   0.524744912   -35.99914   -0.58707036   -23.7451541
## osmo     1.3326202869 -35.999137000 53396.55589 1530.73070792 25988.6596214
## cond     0.0313542297  -0.587070359  1530.73071   63.17682888   516.5354724
## urea     0.6882211188 -23.745154062 25988.65962  516.53547237 16286.2257873
## calc     0.0071710860  -0.141506045   304.62464    8.35287461   157.5068560
##                 calc
## gravity   0.007171086
## ph       -0.141506045
## osmo    304.624636788
## cond      8.352874612
## urea    157.506855955
## calc      7.652966628
```

```r
all(covW(as.matrix(urine), urine$r)[-1,-1] == pooledS(urine, urine$r))
```

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

Així doncs, considero que ho he fet bé (m'he mirat una mica el codi de cada una de les funcions, i em fio més de la segona funció – i.e. $\sim$ entenc el que fa i coincideix amb els meus resultats). A continuació calculo la *lambda de Wilks*

```r
n <- nrow(urine)
k <- length(unique(urine$r))
W <- (n-k) * pooledS(urine, urine$r)
T_ <- (n-1) * cov(urine[, -1])
lambda_wilks <- det(W) / det(T_)
lambda_wilks
```

```
## [1] 0.5839534
```

Veiem que tenim una $\Lambda = 0.5839534$.

Buscant per internet he trobat que aquest valor s'utilitza per a determinar si hi ha diferències entre les *group means* en un conjunt de dades multivariat. Amb el següent paquet puc reproduïr la seva computació.

```
# install.packages(rrcov)
require(rrcov)
Wilks.test(r ~ ., urine)
```

```
##
##  One-way MANOVA (Bartlett Chi2)
##
## data:  x
## Wilks' Lambda = 0.58395, Chi2-Value = 39.807, DF = 6.000, p-value =
## 4.97e-07
## sample estimates:
##    gravity       ph     osmo     cond     urea     calc
## 0 1.015489 6.098667 565.2889 20.48667 237.1111 2.624889
## 1 1.021588 5.935588 684.7941 21.35588 305.1765 6.142941
```

Veiem que el valor de la *Wilks' lambda* és el mateix. Sembla ser que es pot rebutjar la hipòtesi nul · la de que no hi han diferències entre grups.

f)

Per a dur a terme aquest apartat, m'he debatut força sobre quina seria la millor aproximació. La meva intuïció em diu que, sense dubte, hauria d'escalar les variables originals abans d'aplicar l'algoritme d'anàlisi de components principals, però sento que em falta coneixements per estar-ne 100% segur. Així doncs, vaig a provar de realitzar l'anàlisi de les dues maneres [4].

Primer, anem a fer l'anàlisi a partir dels *eigenvectors* de la matriu de covariànces de $\boldsymbol{X}$, $\boldsymbol{S}$. Per a això, utilitzaré la matriu idempotent $\boldsymbol{H}$.

```
# Construcció de H
X <- as.matrix(urine[, -1])
n <- nrow(X)
I = diag(n)
J = matrix(rep(1, n*n), ncol=n)
H <- I - 1/n * J
# Extracció de S, matriu de cov-var, a partir de H i X
S = t(X)%*%H%*%X/(n-1)

stopifnot(max(abs(S - cov(X))) <= 1e-10) # I'm insecure
```

Un cop tenim la matriu $\boldsymbol{S}$, podem extreure fàcilment els components principals, que són aquells vectors $\boldsymbol{y_1}, \boldsymbol{y_2}, ...\boldsymbol{y_m}$, construits mitjançant la combinació lineal de les columnes de la matriu de dades $\boldsymbol{X}$, que maximitzen la seva variança $var(\boldsymbol{y_i}) = var(\boldsymbol{X a_i^T}) = \boldsymbol{a_i^T S a_i}$.

Es pot demostrar fàcilment que aquests vectors $\boldsymbol{a_i}$ que maximitzen la variança de les components principals són els *eigenvectors* de $\boldsymbol{S}$. Ja que es podria augmentar la variança de y_i simplement augmentant la norma de a_i, es posa com a *constraint* per aquest procés de maximització que els vectors a_i siguin unitaris (la seva longitud o norma $L_2 = 1$) (també s'aplica el *constraint* de que els vectors $\boldsymbol{a_i}$ estiguin *uncorrelated*, i.e. $\boldsymbol{a_i^t a_j} = 0 \forall i \neq j$)

Així doncs, els eigenvectors i eigenvalues es poden aconseguir així:

```
evectors <- eigen(S)$vectors
evalues <- eigen(S)$values
round(evalues, 6)
```

[4] La lògica i procediment d'aquest apartat estan fortament influenciats per l'apartat 3.4 del llibre d'Everitt, 2011.

```
## [1] 70646.203516  2899.456659     8.105626     5.239339     0.482842
## [6]     0.000006
```

Veiem que tenim 6 eigenvalues que podriem considerar majors que zero. Això sembla indicar que la matriu de dades $\boldsymbol{X}$ és de rank 6. Comprovem-ho.

```
require(matrixcalc)
matrixcalc::matrix.rank(t(X)%*%X) # l'àlgebra lineal mola
```

```
## [1] 6
```

Així doncs, ja que $\boldsymbol{S}$ és una matriu simètrica i *positive-definite*, els seus *eigenvectors* seran 6 i conformaran una matriu ortogonal. Anem a comprovar-ho:

```
m <- ncol(evectors)
for (i in 1:(m-1)){
  vi <- evectors[, i]
  for (j in (i+1):m){
    vj <- evectors[, j]
    stopifnot(abs(t(vi) %*% vj) < 1e-15)
  }
}
```

Ara si, vaig a calcular els components principals a partir dels *eigenvectors* de $\boldsymbol{S}$. Veiem que, gràcies a les propietats d'àlgebra lineal, hi ha diferents maneres de calcular la variança de les components principals.

```
Y <- X %*% evectors
vars <- evalues

for (i in 1:ncol(Y)){
  varv1 <- var(Y[, i])

  a_i <- evectors[, i]
  varv2 <- as.numeric(t(a_i) %*% S %*% a_i)

  varv3 <- vars[i]

  stopifnot(all(round(varv1, 9) == round(varv2, 9),
                round(varv2, 9) == round(varv3, 9)))
}
```

Tot i això, com a producte de les diferències d'escala entre les variables de $\boldsymbol{X}$, segurament hi ha variables que estan dominant completament els components principals (ja que presenten una variança astronòmicament major a les altres). Això ho podem comprovar inspeccionant els eigenvalues de la descomposició espectral de $\boldsymbol{S}$.

```
round(evalues / sum(evalues) * 100, 2)
```

```
## [1] 96.04  3.94  0.01  0.01  0.00  0.00
```

Veiem que, efectivament, la primera component principal correspon a un 96% de la variança de les dades originals. Si inspeccionem els eigenvectors de $\boldsymbol{S}$, podrem esbrinar quines són les variables originals que estan tenint més pes en la construcció de la primera component principal.

```
round(evectors[, 1], 3)
```

```
## [1]  0.000 -0.001  0.888  0.023  0.460  0.006
```

Veiem que, principalment, estan actuant la tercera i la cinquena variables. Aquestes, com podem veure, corresponen a aquelles que presenten una major variança.

```r
rbind(round(evectors[, 1], 3), round(diag(S), 3))
```

```
##      gravity     ph      osmo   cond     urea   calc
## [1,]       0 -0.001     0.888  0.023     0.46  0.006
## [2,]       0  0.525 56258.024 62.554 17227.76 10.628
```

Si inspeccionem amb més profunditat la matriu d'*eigenvectors*, podem observar que els dos primers són bàsicament les variables 3 i 5 (amb els valors dels seus components canviats), els dos següents *eigenvectors* mostren el mateix patró però amb les variables 4 i 6 (que són les dues següents variables originals amb una variança més alta), i finalment els dos *eigenvectors* finals corresponen bàsicament a multiplicar per 1 les altres dues variables que manquen i per ≈ 0 les demés.

```r
round(evectors, 3)
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]  [,6]
## [1,]  0.000  0.000  0.000  0.001  0.000 1.000
## [2,] -0.001 -0.002  0.000 -0.019 -1.000 0.000
## [3,]  0.888 -0.457 -0.019  0.056 -0.001 0.000
## [4,]  0.023 -0.084  0.400 -0.912  0.017 0.001
## [5,]  0.460  0.886  0.030 -0.057 -0.001 0.000
## [6,]  0.006  0.002 -0.916 -0.402  0.008 0.000
```

Això sembla complir el que es menciona a la pàgina 68 del llibre d'Everitt 2011 (apartat 3.4):

> (...) the principal components from the covariance matrix simply reflect the order of the sizes of the variances of the observed variables.

Així doncs, podem determinar que ÉS NECESSARI escalar les variables prèviament a aplicar PCA. Així doncs, vaig a dur a terme això a continuació. Tenim la matriu de covariances guardada a la variable `S`. Podem obtenir la matriu de correlacions, que anomeno $\boldsymbol{R}$, així:

```r
sds <- apply(X, 2, sd)
D <- diag(1/sds)
R <- D%*% S %*% D

max(abs(R - cor(X))) # ben fet
```

```
## [1] 2.229328e-13
```

Ara si, anem a obtenir la descomposició espectral de $\boldsymbol{R}$ que ens permetrà calcular les components principals de les dades originals escalades, desfent aquestes variacions entre les escales de les diferents variables originals.

```r
evalues <- eigen(R)$values
evectors <- eigen(R)$vectors

evalues
```

```
## [1] 3.685878595 0.949747371 0.693044923 0.487306294 0.176746989 0.007275829
```

Veiem que, un altre cop, tenim 6 *eigenvalues* majors a 0, indicant que $\boldsymbol{R}$ és de rang 6 (fet que ja podíem assumir ja que $\boldsymbol{S}$ ja ho era). Ara si, vaig a construir les components principals. Com ara demostraré, és el mateix construir les components principals a partir de la matriu de covariances calculada amb $\boldsymbol{X}$ escalada ($\boldsymbol{X}$ amb la mitjana substreta i dividida per la desviació estàndard), que amb la matriu de correlacions de $\boldsymbol{X}$.

```r
X_scaled = scale(X, T, T)

max(abs(cov(X_scaled) - cor(X)))
```

```
## [1] 5.551115e-16
```

Així doncs, utilitzant els *eigenvectors* de $\boldsymbol{R}$, podem dur a terme l'anàlisi PCA generant els components principals multiplicant la matriu de $\boldsymbol{X}$ escalada per aquests.

```
Y <- X_scaled%*%evectors
# veiem que es compleix el mateix
vars <- evalues
for (i in 1:ncol(Y)){
  varv1 <- var(Y[, i])

  a_i <- evectors[, i]
  varv2 <- as.numeric(t(a_i) %*% R %*% a_i)

  varv3 <- vars[i]

  stopifnot(all(round(varv1, 9) == round(varv2, 9),
                round(varv2, 9) == round(varv3, 9)))
}

# També veiem que obtenim els mateixos resultats utilitzant la funció
# "ground truth" per a realitzar PCA de R
stopifnot(max(abs(Y) - abs(prcomp(X,scale=T, center=T)$x)) < 1e-10)
```

Si observem els *eigenvalues* de $\boldsymbol{R}$, podem observar els valors de les variàncies de les components principals.

```
evalues
```

```
## [1] 3.685878595 0.949747371 0.693044923 0.487306294 0.176746989 0.007275829
```

```
round(evalues / sum(evalues) * 100, 2)
```

```
## [1] 61.43 15.83 11.55  8.12  2.95  0.12
```

Veiem que la primera component principal conté un 61.43% de la variança de les variables originals. Si seleccionessim les tres primeres components principals, aquestes contendrien un `88.81` de la variança de les variables originals.

Responent a la pregunta de l'enunciat, referent al número de components principals necessaris per a obtenir una bona representació de les variables originals, avaluaré els diferents mètodes que es mencionen.

1. Criteri de *Kaiser*

Aquest criteri diu que agafem aquelles y_j primeres components principals que compleixen que els seus valors propis (m'he estat referint a ells com a *eigenvalues* fins ara) són superiors a 1. Anem a mirar quins són aquests.

```
which(evalues > 1)
```

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

Només el primer *eigenvalue* compleix el criteri de *Kaiser*.

2. *Cum sum* (suma acumulativa) de la variança total superior a $\approx$ 70/80%.

```
cumsum(round(evalues / sum(evalues) * 100, 2))
```

```
## [1]  61.43  77.26  88.81  96.93  99.88 100.00
```

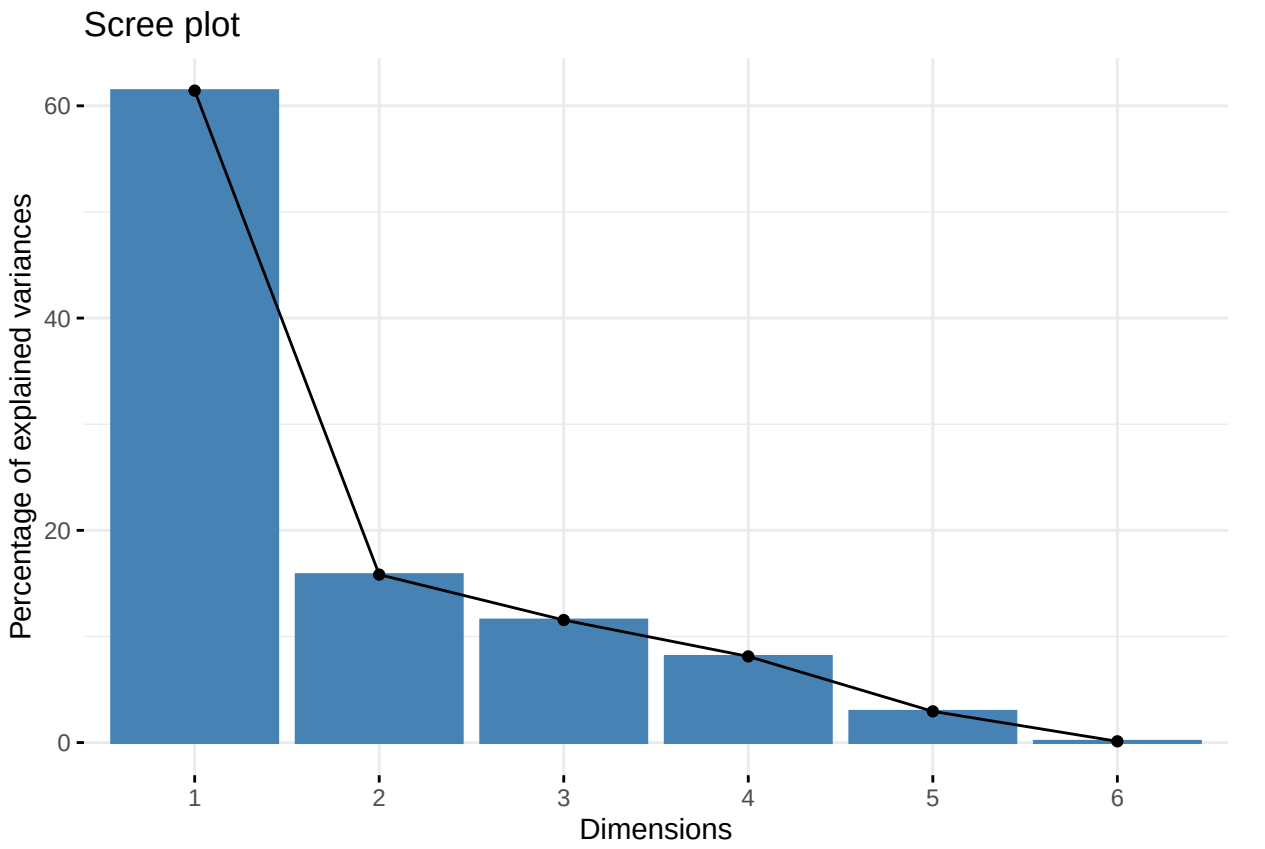
Veiem que, tal i com he mencionat abans, la primera component principal *explica* un 61.43% de la variança de les dades, mentre que amb les dues primeres ja superaríem el llindar de 70% mencionat al criteri. Tot i així, només seria fins a incloure la tercera, que superaríem el llindar major de 80%. Així doncs, ens quedaríem

amb dues o tres, depenent de quin subcriteri dins d'aquest criteri seguissim. Com que 77.26 arrodoneix a 80, jo optaria per agafar-ne dues.

3. *Scree plot*

Per a dur a terme aquest criteri, volia utilitzar la llibreria `factorextra` tal i com es recomana a les solucions dels exercicis de la setmana 2 de l'assignatura, però no està disponible per a la meva versió de `R`. Tot i això, he aconseguit fer-ho instal · lant directament des de `Github`.

```
# if(!require(devtools)) install.packages("devtools")
# devtools::install_github("kassambara/factoextra")
library(factoextra)
fviz_eig(prcomp(X, scale=T, center=T))
```



Veiem així doncs, que el *colze* del *Scree plot* es situa a les dues components principals.

4. Criteri *interpretabilitat*

Entenc que, per a aplicar aquest criteri, he d'inspeccionar els *eigenvectors* que construeixen les components principals. Per a facilitar l'interpretabilitat, afegeixo informació a la variable que els conté.

```r
rownames(evectors) <- colnames(X)
colnames(evectors) <- paste('PC', 1:6, sep='')

round(evectors, 4)
```

```
##             PC1     PC2     PC3     PC4     PC5     PC6
## gravity  0.4734 -0.0030  0.0319 -0.3668  0.7800  0.1788
## ph      -0.1680  0.9551  0.0529 -0.2380 -0.0089  0.0017
## osmo     0.5092  0.0917 -0.1996 -0.0371 -0.1298 -0.8211
## cond     0.3934  0.2494 -0.5205  0.6104 -0.0150  0.3732
## urea     0.4682 -0.0508  0.0441 -0.4987 -0.6108  0.3931
## calc     0.3382  0.1208  0.8267  0.4314 -0.0377  0.0087
```

Veiem que `PC1` conté un *weighted average* de les 6 components principals, com sol ocórrer quan es duu a terme aquest anàlisi amb les variables escalades. Cal tenir en compte que la única variable que té un escalar amb signe negatiu multiplicant-la és `ph`. Això té sentit ja que, totes les altres variables sembla que el seu augment està correlacionat amb una major presència de cristalls d'oxalat de calci, menys el pH, que segons he trobat a la lliteratura, la formació de cristalls d'oxalat de calci sol ocórrer més (sol potenciar-se) a pHs més àcids (més baixos) (Carvalho 2018; Werner et al. 2021; Berg and Tiselius 1986). Així que, fins on entenc sobre interpretació de components principals, sembla que té sentit.

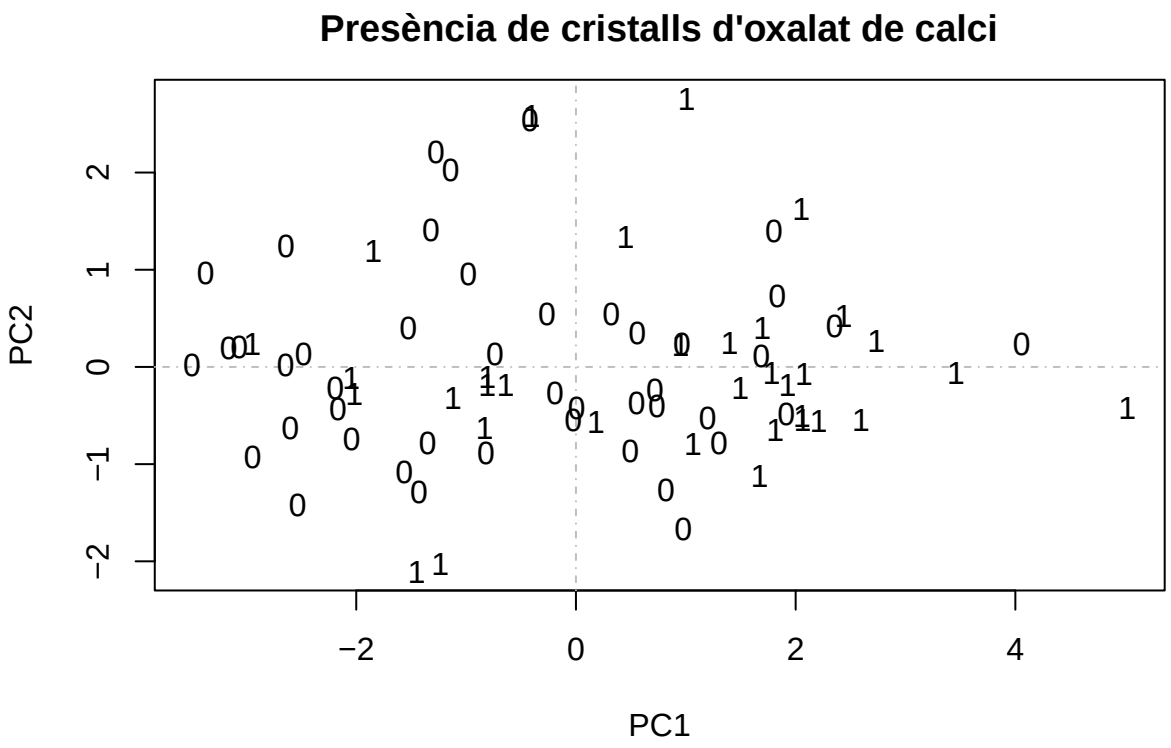
La segona component principal, en canvi, sembla que mostra el patró contrari. La variable amb més pes (un escalar major multiplicant-la) és `ph`, amb signe positiu, mentre que les altres tenen un pes força menor. Amb aquest anàlisi, jo m'aventuraria a dir que amb aquestes dues components principals hauriem d'obtenir una separació força clara d'aquelles observacions que presenten cristalls i les que no en presenten.

En definitiva, la meva elecció en quant al nombre de components principals que representen les dades, la meva aportació seria que la resposta resta entre 2 i 3. Si haguéssim de separar les dues classes del factor `r`, segurament triaria 3, ja que segurament donaria millors resultats a la hora de generar models predictius (i l'àmbit mèdic és força important prendre decisions ben informades). Si l'objectiu és obtenir un nombre de variables amb els que dur a terme una recerca exploratòria de les dades bàsica, llavors amb 2 potser seria suficient.

g)

Primer de tot, vaig a graficar els punts d'acord a les dues primeres components principals. Com podeu veure, sembla que hi ha un relatiu clustering de les observacions amb cristalls (`r`= 1) a la zona dreta del gràfic. Sincerament, em costa saber quina seria la línea que separaria millor les dues classes. He intentat utilitzar la funció `e1071::svm` per a trobar-la però no m'ha donat resultats gaire satisfactoris. El que és clar és que, el grup amb cristalls d'oxalat de calci té valors de 5/6 variables més alts, i per tant valors de la `PC1` majors.

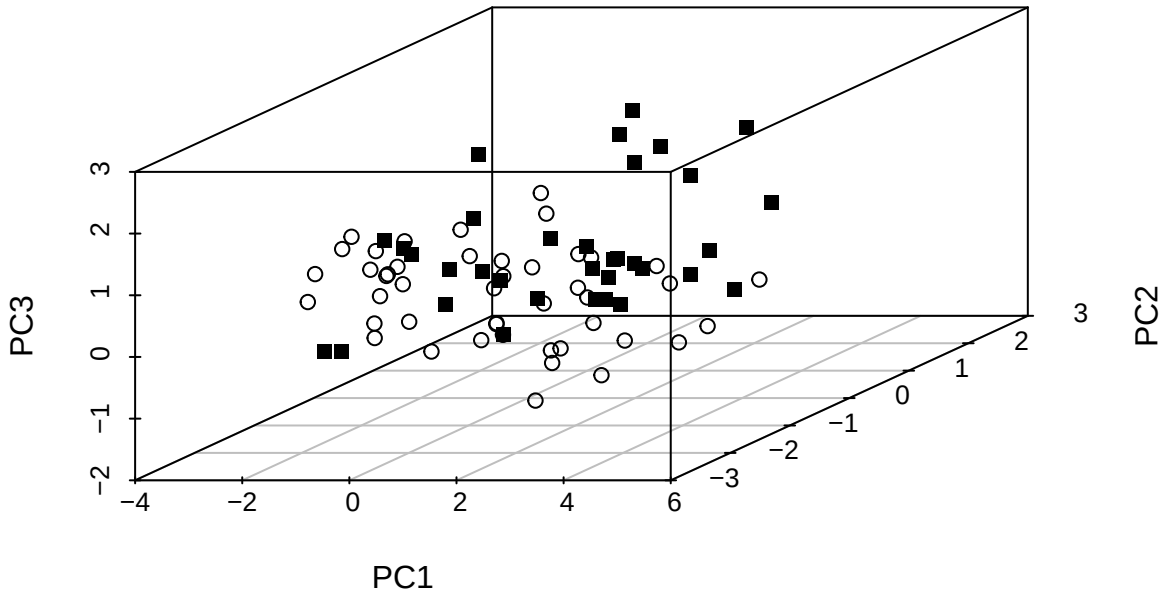
```r
plot(Y[, 1:2], xlab="PC1", ylab="PC2", pch="")
abline(v=0,
       h=0,
       lty=4,col="gray")
text(Y[, 1],Y[, 2],labels=urine$r,cex=1)
title(main="Presència de cristalls d'oxalat de calci",line=1)
```



Ara vaig a fer el mateix però amb les tres primeres components principals.

```
# install.packages('scatterplot3d')
library(scatterplot3d)

scatterplot3d(Y[,1], Y[,2], Y[,3],
              xlab="PC1", ylab="PC2", zlab="PC3",
             pch = ifelse(urine$r == 0, 1, 15))
```



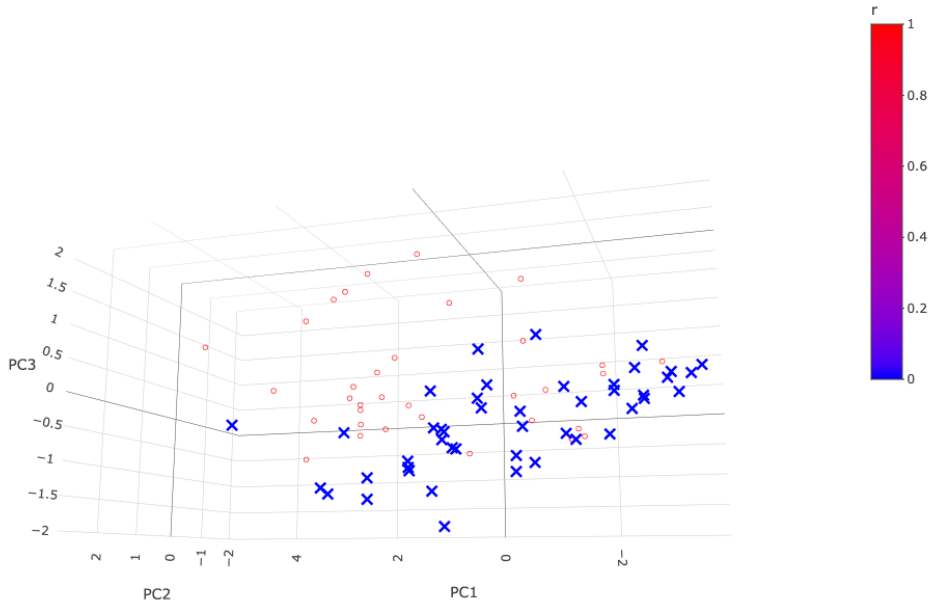
Aquest gràfic és difícil d'interpretar, doncs a continuació genero un gràfic tridimensional rotable amb `plotly`. Degut a que, en format `pdf`, no es poden *incrustar* gràfics rotables, el que faig és rotar jo el gràfic fins a què es pugui dur a terme una visualització satisfactòria, genero una captura en format `.png`, i l'adjunto més

endavant. A continuació deixo el codi utilitzat per a generar el gràfic del que parlo.

```
# codi utilitzat per a generar el gràfic, copiat dels apunts (exercicis) de
# l'assignatura
library(plotly)
fig <- plot_ly(urine, x = ~Y[,1], y = ~Y[,2], z = ~Y[,3],
               color = ~r, colors = c("blue","red"),
               symbol = ~r, symbols = c('x','o')) %>%
  add_markers(marker = list(size = 3)) %>%
  layout(scene = list(xaxis = list(title = "PC1"),
                      yaxis = list(title = "PC2"),
                      zaxis = list(title = "PC3")))
```

I el resultat.

```
library(png)
img <- readPNG('pca3.png')
grid::grid.raster(img)
```



Sembla ser que, en conclusió, si volguéssim separar les dues classes (presència o no de cristalls d'oxalat de calci), tres components principals serien força útils, més que (òbviament) que dues. Podem intuir amb el gràfic "rotable" que la presència d'aquests cristalls sembla estar associat amb valors de la primera component més alts, valors menors de la segona component, i valors majors de la tercera. Entre quina component és més útil, si la segona o la tercera, no n'estic segur. Anem a repassar com es construeixen aquestes tres PCs.

```
evectors[, 1:3]
```

```
##                PC1          PC2         PC3
## gravity  0.4733778 -0.002961087  0.03189092
```

```
## ph      -0.1680088  0.955112480  0.05293882
## osmo     0.5092403  0.091662957 -0.19963241
## cond     0.3934346  0.249359232 -0.52050808
## urea     0.4681913 -0.050764924  0.04414000
## calc     0.3381819  0.120798099  0.82671056
```

Sembla ser que la tercera PC (les altres dues primeres ja han sigut comentades prèviament) és, principalment, la variable `calc` (concentració de calci) menys la `cond`uctivitat i la `osmo`laritat. Possiblement, la formació de cristalls d'oxalat de calci disminueixi la quantitat de ions carregats positivament (ja que el calci no es troba en estat lliure).

El que em costa més d'interpretar, és el coeficient negatiu de la variable `osmo`, ja que, amb els meus – encara que limitats – coneixements de fisiologia humana i una breu recerca bibliogràfica he trobat que, com un deuria esperar [5], la osmolaritat està positivament correlacionada amb la presència de cristalls d'oxalat de calci (Kavouras et al. 2021).

Exercici 2

Carrego les dades.

```
seabirds <- read.csv('seabirds.csv', row.names = 1)
head(seabirds)
```

```
##                            CH    PLI     CI    NS    CL     CT   SI    SPI
## Northern.fulmar             0 124000      0     0     0      0  224    700
## Glaucous-winged.gull      150    400    150   274    50    300    0      0
## Black-legged.kittiwake  40000  58000  60000  7550 25000  26500 5000  31000
## Red-legged.kittiwake        0      0      0     0     0      0    0   2200
## Thick-billed.murre     280000 172000 320000   400 30000 233578    0 110000
## Common.murre                0      0      0 41800 70000 155719 5320  39000
##                           SGI
## Northern.fulmar         70000
## Glaucous-winged.gull        0
## Black-legged.kittiwake  72000
## Red-legged.kittiwake   220000
## Thick-billed.murre    1500000
## Common.murre           190000
```

```
dim(seabirds)
```

```
## [1] 23  9
```

Com podem veure, el conjunt de dades `seabirds` té, codificat a les files i columnes respectivament, les espècies i poblacions a les quals fan referència les entrades de la taula. Les entrades de la taula corresponen al número d'ocells de cada espècie i població.

a)

Calcular les freqüències relatives és relativament fàcil. Cal obtenir la suma total de la taula, i dividir cada entrada d'aquesta per la suma total.

```
X <- as.matrix(seabirds)
freq.rel <- X / sum(X)

round(freq.rel, 4)
```

[5] D'acord a la documentació del *dataset* `urine`: *Osmolarity is proportional to the concentration of molecules in solution.* Això sembla reforçar el fet de que major presència de cristalls, major osmolaritat.

```
##                              CH    PLI     CI     NS     CL     CT     SI
## Northern.fulmar          0.0000 0.0260 0.0000 0.0000 0.0000 0.0000 0.0000
## Glaucous-winged.gull     0.0000 0.0001 0.0000 0.0001 0.0000 0.0001 0.0000
## Black-legged.kittiwake   0.0084 0.0122 0.0126 0.0016 0.0052 0.0056 0.0010
## Red-legged.kittiwake     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Thick-billed.murre       0.0588 0.0361 0.0671 0.0001 0.0063 0.0490 0.0000
## Common.murre             0.0000 0.0000 0.0000 0.0088 0.0147 0.0327 0.0011
## Black.guillemot          0.0000 0.0017 0.0000 0.0000 0.0000 0.0000 0.0000
## Pigeon.guillemot         0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Horned.puffin            0.0000 0.0000 0.0000 0.0007 0.0003 0.0003 0.0000
## Tufted.puffin            0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Atlantic.puffin          0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0034
## Pelagic.cormorant        0.0000 0.0000 0.0000 0.0001 0.0000 0.0000 0.0000
## Red-faced.cormorant      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Shag                     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Parakeet.auklet          0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Crested.auklet           0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Least.auklet             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Razorbill                0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0009
## Manx.shearwater          0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0546
## Storm.petrel             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0027
## Herring.gull             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0016
## Great.black-backed.gull  0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001
## Lesser.black.backed.gull 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0042
##                             SPI    SGI
## Northern.fulmar          0.0001 0.0147
## Glaucous-winged.gull     0.0000 0.0000
## Black-legged.kittiwake   0.0065 0.0151
## Red-legged.kittiwake     0.0005 0.0462
## Thick-billed.murre       0.0231 0.3147
## Common.murre             0.0082 0.0399
## Black.guillemot          0.0000 0.0000
## Pigeon.guillemot         0.0000 0.0000
## Horned.puffin            0.0009 0.0059
## Tufted.puffin            0.0002 0.0013
## Atlantic.puffin          0.0000 0.0000
## Pelagic.cormorant        0.0000 0.0000
## Red-faced.cormorant      0.0005 0.0010
## Shag                     0.0000 0.0000
## Parakeet.auklet          0.0071 0.0315
## Crested.auklet           0.0013 0.0059
## Least.auklet             0.0048 0.0525
## Razorbill                0.0000 0.0000
## Manx.shearwater          0.0000 0.0000
## Storm.petrel             0.0000 0.0000
## Herring.gull             0.0000 0.0000
## Great.black-backed.gull  0.0000 0.0000
## Lesser.black.backed.gull 0.0000 0.0000
```

Per a calcular les freqüències marginals, utilitzo funcions que he creat jo i que estan disponibles al meu repositori de `Github` de l'assignatura [6]. Aquestes van ser creades en els meus esforços per entendre els materials de l'assignatura, i vull utilitzar-les.

[6] https://github.com/vcasellesb/analisis-multi.git

```r
source('../funcs/CA.R')
freq_marg_col <- column_profiles(X, average=T)[, (ncol(X)+1)]

freq_marg_row <- rowprofile(X, average=T)[(nrow(X)+1), ]
```

Un cop he generat la fila i la columna *marginals*, les afegeixo iterativament a continuació.

```r
freq.marg <- freq.rel
freq.marg <- rbind(freq.marg, freq_marg_row)
freq.marg <- cbind(freq.marg, c(freq_marg_col, sum(freq_marg_col)))
colnames(freq.marg) <- c(colnames(X), 'Sum')
rownames(freq.marg) <- c(rownames(X), 'Sum')

stopifnot(max(abs(freq.marg - addmargins(freq.rel))) < 1e-15) # comprovació
```

El resultat és el següent.

```r
round(freq.marg, 4)
```

```
##                             CH    PLI     CI     NS     CL     CT     SI
## Northern.fulmar          0.0000 0.0260 0.0000 0.0000 0.0000 0.0000 0.0000
## Glaucous-winged.gull     0.0000 0.0001 0.0000 0.0001 0.0000 0.0001 0.0000
## Black-legged.kittiwake   0.0084 0.0122 0.0126 0.0016 0.0052 0.0056 0.0010
## Red-legged.kittiwake     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Thick-billed.murre       0.0588 0.0361 0.0671 0.0001 0.0063 0.0490 0.0000
## Common.murre             0.0000 0.0000 0.0000 0.0088 0.0147 0.0327 0.0011
## Black.guillemot          0.0000 0.0017 0.0000 0.0000 0.0000 0.0000 0.0000
## Pigeon.guillemot         0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Horned.puffin            0.0000 0.0000 0.0000 0.0007 0.0003 0.0003 0.0000
## Tufted.puffin            0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Atlantic.puffin          0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0034
## Pelagic.cormorant        0.0000 0.0000 0.0000 0.0001 0.0000 0.0000 0.0000
## Red-faced.cormorant      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Shag                     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Parakeet.auklet          0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Crested.auklet           0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Least.auklet             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Razorbill                0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0009
## Manx.shearwater          0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0546
## Storm.petrel             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0027
## Herring.gull             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0016
## Great.black-backed.gull  0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001
## Lesser.black.backed.gull 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0042
## Sum                      0.0672 0.0760 0.0798 0.0113 0.0266 0.0876 0.0696
##                             SPI    SGI    Sum
## Northern.fulmar          0.0001 0.0147 0.0409
## Glaucous-winged.gull     0.0000 0.0000 0.0003
## Black-legged.kittiwake   0.0065 0.0151 0.0682
## Red-legged.kittiwake     0.0005 0.0462 0.0466
## Thick-billed.murre       0.0231 0.3147 0.5552
## Common.murre             0.0082 0.0399 0.1053
## Black.guillemot          0.0000 0.0000 0.0018
## Pigeon.guillemot         0.0000 0.0000 0.0000
## Horned.puffin            0.0009 0.0059 0.0081
## Tufted.puffin            0.0002 0.0013 0.0015
```

```
## Atlantic.puffin          0.0000 0.0000 0.0034
## Pelagic.cormorant        0.0000 0.0000 0.0001
## Red-faced.cormorant      0.0005 0.0010 0.0016
## Shag                     0.0000 0.0000 0.0000
## Parakeet.auklet          0.0071 0.0315 0.0386
## Crested.auklet           0.0013 0.0059 0.0071
## Least.auklet             0.0048 0.0525 0.0573
## Razorbill                0.0000 0.0000 0.0009
## Manx.shearwater          0.0000 0.0000 0.0546
## Storm.petrel             0.0000 0.0000 0.0027
## Herring.gull             0.0000 0.0000 0.0016
## Great.black-backed.gull  0.0000 0.0000 0.0001
## Lesser.black.backed.gull 0.0000 0.0000 0.0042
## Sum                      0.0533 0.5285 1.0000
```

Per a generar la matriu de la taula 12.6 del llibre de Krebs, degut a que a primera vista sembla que les columnes sumen 1 (i per tant es tracten de *column profiles*), utilitzaré la meva funció homònima.

```
profiles <- column_profiles(X)
round(profiles, 4)
```

```
##                             CH    PLI     CI     NS     CL     CT     SI
## Northern.fulmar          0.0000 0.3422 0.0000 0.0000 0.0000 0.0000 0.0007
## Glaucous-winged.gull     0.0005 0.0011 0.0004 0.0051 0.0004 0.0007 0.0000
## Black-legged.kittiwake   0.1249 0.1600 0.1577 0.1402 0.1972 0.0634 0.0151
## Red-legged.kittiwake     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Thick-billed.murre       0.8740 0.4746 0.8413 0.0074 0.2367 0.5593 0.0000
## Common.murre             0.0000 0.0000 0.0000 0.7765 0.5522 0.3728 0.0160
## Black.guillemot          0.0006 0.0221 0.0005 0.0000 0.0013 0.0000 0.0000
## Pigeon.guillemot         0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Horned.puffin            0.0000 0.0000 0.0000 0.0592 0.0114 0.0036 0.0000
## Tufted.puffin            0.0000 0.0000 0.0000 0.0008 0.0002 0.0000 0.0000
## Atlantic.puffin          0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0482
## Pelagic.cormorant        0.0000 0.0000 0.0000 0.0096 0.0006 0.0001 0.0001
## Red-faced.cormorant      0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Shag                     0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0001
## Parakeet.auklet          0.0000 0.0000 0.0000 0.0012 0.0000 0.0000 0.0000
## Crested.auklet           0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Least.auklet             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
## Razorbill                0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0130
## Manx.shearwater          0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.7838
## Storm.petrel             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0389
## Herring.gull             0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0229
## Great.black-backed.gull  0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0009
## Lesser.black.backed.gull 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0603
##                            SPI    SGI
## Northern.fulmar          0.0028 0.0278
## Glaucous-winged.gull     0.0000 0.0000
## Black-legged.kittiwake   0.1221 0.0286
## Red-legged.kittiwake     0.0087 0.0873
## Thick-billed.murre       0.4334 0.5955
## Common.murre             0.1537 0.0754
## Black.guillemot          0.0000 0.0000
## Pigeon.guillemot         0.0000 0.0000
## Horned.puffin            0.0173 0.0111
```

```
## Tufted.puffin             0.0039 0.0024
## Atlantic.puffin           0.0000 0.0000
## Pelagic.cormorant         0.0000 0.0000
## Red-faced.cormorant       0.0099 0.0020
## Shag                      0.0000 0.0000
## Parakeet.auklet           0.1340 0.0595
## Crested.auklet            0.0236 0.0111
## Least.auklet              0.0906 0.0992
## Razorbill                 0.0000 0.0000
## Manx.shearwater           0.0000 0.0000
## Storm.petrel              0.0000 0.0000
## Herring.gull              0.0000 0.0000
## Great.black-backed.gull   0.0000 0.0000
## Lesser.black.backed.gull 0.0000 0.0000
```

He estat observant aquesta taula i la 12.6 *side-by-side*, i em sembla que són idèntiques.

b)

Per a calcular les distàncies Xi-quadrat, també faig servir una funció que he creat (comprovant que no m'hagi equivocat amb la funció que s'utilitza als exercicis de l'assignatura).

```
d.chisq.col <- d_chisq(X, col=T)
```

Comprovo que estigui bé amb la funció que proporciona Everitt 2011.

```
D_true <- D <- function(x) {
  a <- t(t(x) / colSums(x))
  ret <- sqrt(colSums((a[,rep(1:ncol(x), ncol(x))] -
                         a[, rep(1:ncol(x), rep(ncol(x), ncol(x)))])^2 *
                        sum(x) / rowSums(x)))
  matrix(ret, ncol = ncol(x))
}

stopifnot(max(abs(d.chisq.col - D_true(X))) < 1e-10)
```

Si que està bé.

Un altre cop, el càlcul de la inercia total el duc a terme amb codi escrit per mi mateix durant l'estudi de l'assignatura.

```
total_inertia <- inertia(X)
total_inertia
```

```
## [1] 1.565692
```

Veiem que la inercia total és de `1.566`. Aquest resultat l'he verificat amb el procediment que mostra el professor a les solucions dels exercicis del tema d'anàlisi de correspondencia. Donat que per a verificar aquestes computacions es requereix forces línies de codi i explicació d'aquestes, he decidit deixar aquesta comprovació a l'apèndix (m'estic allargant força en aquesta PAC i considero que tant autor com lector estaran cansats de mi). En resum, si no he fet algun error estúpid, hauria de ser correcte.

c)

Per a dur a terme el anàlisi *MDS*, utilitzo codi que he generat durant l'estudi d'aquest apartat de l'assignatura. Bàsicament, és una funció que utilitza la teoria que es troba a diferents fonts, com el llibre d'Everitt, 2011, per a primer calcular el que s'anomena com a matriu $\boldsymbol{B}$ (també anomenada com a *double centering matrix*, *kernel matrix*) a partir de la matriu de distàncies $\boldsymbol{D}$.

```
source('../funcs/MDS.R')
B <- B_from_D(d.chisq.col)
```

Un cop tinc la matriu $\boldsymbol{B}$, puc aplicar *MDS* extraient els seus *eigenvalues* i *eigenvectors*.

```
evalues <- eigen(B)$values
evectors <- eigen(B)$vectors

evalues
```

```
## [1]  1.363746e+01  7.459209e+00  2.567215e+00  8.796498e-01  3.385369e-01
## [6]  2.806026e-01  7.038339e-02  8.644579e-05 -2.694645e-15
```

Veiem que tenim 8 *eigenvalues* aparentment majors que 0. Així doncs, podem presuposar que el rank de $\boldsymbol{B}$ és de 8. És fàcil de comprovar.

```
require(matrixcalc)
matrix.rank(B)
```

```
## [1] 8
```

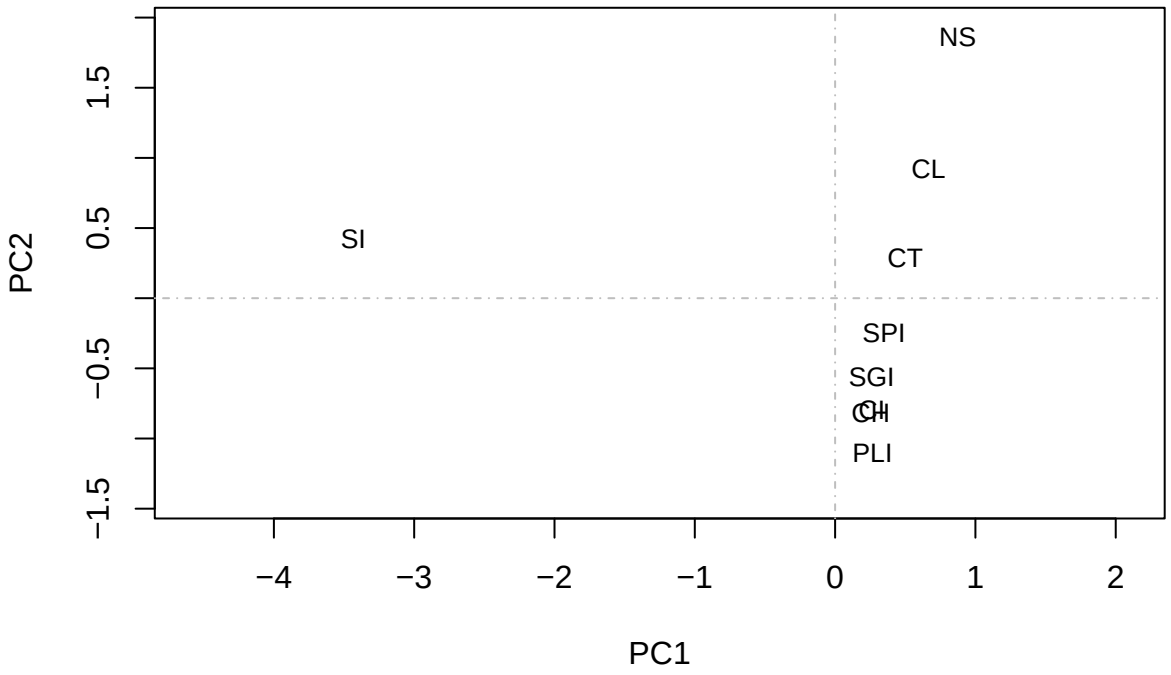
Veiem que és així. Els punts de $\boldsymbol{X}$ que han generat aquesta matriu de distàncies $\boldsymbol{D}$ els podem estimar utilitzant *Multidimensional Scaling* a partir dels *eigenvectors* anteriorment extrets. Utilitzo una dimensió de dos per a les variables de $\boldsymbol{X}$ estimades tal i com fa `cmdscale` per *default*.

```
evecs2 <- evectors[, 1:2]
evalues2 <- diag(sqrt(evalues[1:2]))
X_pred <- evecs2 %*% evalues2

stopifnot(max(abs(X_pred - cmdscale(d.chisq.col))) < 1e-10)
```

Per a representar els *column profiles* de la taula, utilitzo codi tobat a les solucions dels exercicis de *Correspondance Analysis* al campus de l'assignatura.

```
require(MASS)
eqscplot(X_pred,ty="n",xlab="PC1",ylab="PC2", xlim= c(-4.5, 2), ylim=c(-1.5, 2))
abline(v=0,h=0, col="gray",lty=4)
text(X_pred[, 1],X_pred[,2],labels=colnames(X),cex=0.8)
```



Com podem veure, amb aquestes dues primeres components de l'escalat multidimensional, aconseguim separar força bé totes les columnes.

d)

Calculo la matriu $\boldsymbol{Z}$ a partir de la qual calcular la seva descomposició en valors singulars tal i com es fa a l'exercici 5 del tema de *Correspondence Analysis*.

```
Df <- diag(freq_marg_col)
Dc <- diag(freq_marg_row)
Dfmh <- diag(1/sqrt(freq_marg_col))
Dcmh <- diag(1/sqrt(freq_marg_row))
Z <- Dfmh %*% (freq.rel - freq_marg_col %o% freq_marg_row) %*% Dcmh

# com a curiositat, fent proves he trobat que la matriu a partir de la qual
# calculo les inercies és igual a Z ^ 2
stopifnot(max(abs(Z**2 - inertia(X, F))) < 1e-15)
```

Un cop tenim la matriu $\boldsymbol{Z}$, podem obtenir la seva descomposició amb la comanda `svd`.

```
Z.svd <- svd(Z)
```

A partir de la descomposició en valors singulars de $\boldsymbol{Z}$ podem calcular les inèrcies principals i la inèrcia total. Això també ho podem fer amb la meva funció `inertia`.

```
c.sc <- Dcmh %*% Z.svd$v
c.pc <- c.sc %*% diag(Z.svd$d)
f.sc <- diag(1/sqrt(freq_marg_col)) %*% Z.svd$u
f.pc <- f.sc %*% diag(Z.svd$d)

P_inertias <-  Z.svd$d^2

# Inèrcies principals en %
round(P_inertias / sum(P_inertias) * 100, 2)
```

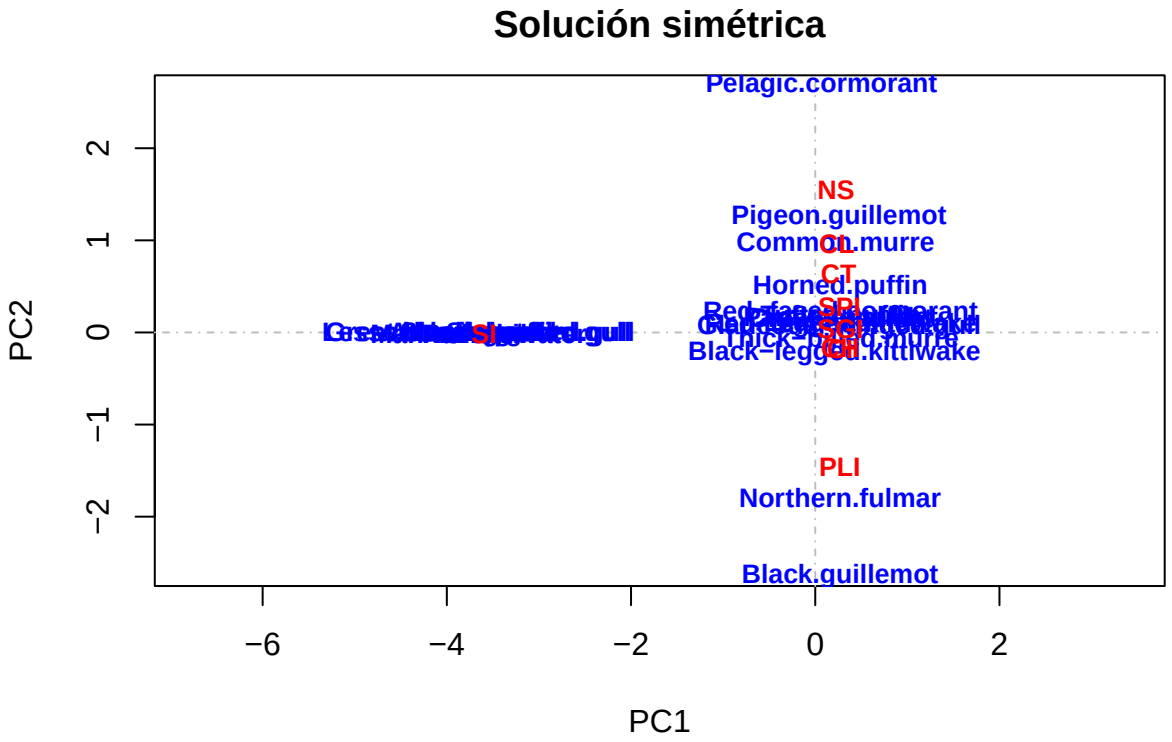
```
## [1] 61.71 16.39 13.15  6.30  1.67  0.53  0.24  0.00  0.00
```

```
inertia(X)
```

```
## [1] 1.565692
```

Al *chunk* de codi anterior podeu veure les inèrcies principals en percentatge. La inèrcia total és de `1.57`. A continuació grafico les distàncies entre les files i les columnes. He intentat canviar la disposició dels noms de les files, però no he sabut fer-ho. Així doncs, degut al solapament entre *labels*, és difícil discernir les espècies que caracteritzen a la colònia `SI`.

```
eqscplot(f.pc[,1:2],type="n",xlab="PC1",ylab="PC2")
abline(v=0,h=0, col="gray",lty=4)
text(f.pc[,1],f.pc[,2],labels=rownames(X),cex=0.8,font=2,col="blue")
text(c.pc[,1],c.pc[,2],labels=colnames(X),cex=0.8,font=2,col= "red")
title(main="Solución simétrica",line=1)
```



Tot i això, penso que no hauria de ser difícil de trobar utilitzant lògica. Veiem que les *labels* blaves de `SI` es troben allà on la `PC1` és menor a 2 i la `PC2` es troba a prop de 0.

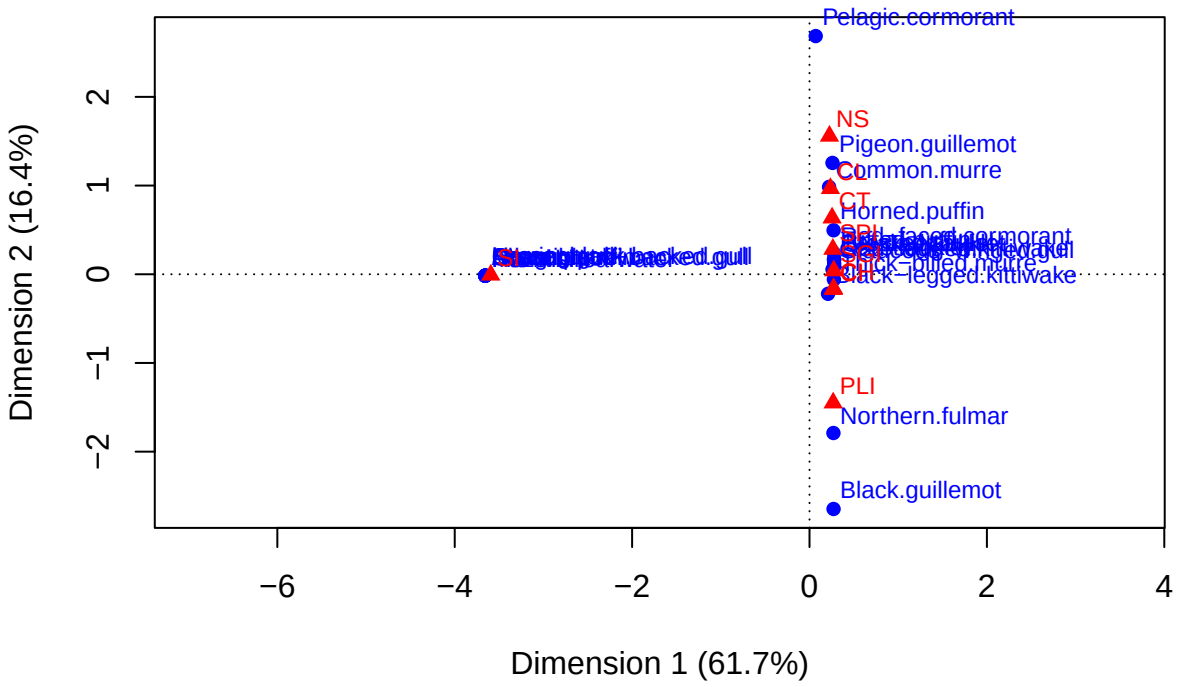
```
mask1 <- f.pc[,1] < -2
mask2 <- f.pc[,2] > -0.1 & f.pc[,2] < 0.1

row.names(X)[mask1&mask2]
```

```
## [1] "Atlantic.puffin"          "Shag"
## [3] "Razorbill"                "Manx.shearwater"
## [5] "Storm.petrel"             "Herring.gull"
## [7] "Great.black-backed.gull"  "Lesser.black.backed.gull"
```

Jo diria que aquests són els noms de les espècies que caracteritzen la colònia denominada com a `SI`. Utilitzant el paquet `ca` veiem que s'aconsegueix pràcticament el mateix gràfic.

```
library(ca)
plot(ca(freq.rel))
```



e)

Creo la funció que permet calcular la distància de Canberra tal i com es demana al principi a continuació.

```
cranberra <- function(mat){
  n <- ncol(mat)
  res <- matrix(0, n, n)
  for (i in 1:(n-1)){
    p <- mat[,i]
    for (j in (i+1):n){
      q <- mat[,j]
      num <- abs(p - q)
      denom <- abs(p) + abs(q)
      resy <- num/denom
      res[i, j] <- res[j, i] <- sum(resy, na.rm = T)
    }
  }
  if (!is.null(colnames(mat))) dimnames(res) <- list(colnames(mat), colnames(mat))
  res
}
```

La provem.

```
profiles <- column_profiles(X)
cranberra(profiles)
```

```
##             CH       PLI         CI        NS        CL        CT       SI
## CH   0.0000000  2.768916  0.3066923  7.872672  5.248740  5.726277 14.78458
## PLI  2.7689165  0.000000  2.7128716  8.678678  6.797518  6.724760 14.82392
## CI   0.3066923  2.712872  0.0000000  7.897404  5.108834  5.883053 14.82557
```

```
## NS   7.8726718  8.678678  7.8974045  0.000000  6.345527  8.318004 16.74805
## CL   5.2487396  6.797518  5.1088341  6.345527  0.000000  5.609152 16.56029
## CT   5.7262773  6.724760  5.8830525  8.318004  5.609152  0.000000 15.66589
## SI  14.7845812 14.823919 14.8255729 16.748052 16.560290 15.665893  0.00000
## SPI 11.3480265 11.163693 11.4472135 10.909901 11.221283 12.517229 19.19796
## SGI 11.8170207 11.659654 11.8643193 11.619124 11.827739 12.586763 18.91136
##          SPI      SGI
## CH  11.34803 11.81702
## PLI 11.16369 11.65965
## CI  11.44721 11.86432
## NS  10.90990 11.61912
## CL  11.22128 11.82774
## CT  12.51723 12.58676
## SI  19.19796 18.91136
## SPI  0.00000  4.67864
## SGI  4.67864  0.00000
```

Veiem que, efectivament, els resultats no són els mateixos que els de la taula 12.7.

Modifico el codi i creo la nova funció d'acord amb el que es demana.

```
cranberra <- function(mat){
  n <- ncol(mat)
  res <- matrix(0, n, n)
  for (i in 1:(n-1)){
    p <- mat[,i]
    for (j in (i+1):n){
      q <- mat[,j]
      num <- abs(p - q)
      denom <- abs(p) + abs(q)
      resy <- num/denom
      res[i, j] <- res[j, i] <- sum(resy, na.rm = T)
    }
  }
  if (!is.null(colnames(mat))) dimnames(res) <- list(colnames(mat), colnames(mat))
  res / nrow(mat)
}
```

Veiem que podem reproduir amb força fidelitat la taula 12.17.

```
as.dist(round(1 - cranberra(profiles), 2))
```

```
##       CH  PLI   CI   NS   CL   CT   SI  SPI
## PLI 0.88
## CI  0.99 0.88
## NS  0.66 0.62 0.66
## CL  0.77 0.70 0.78 0.72
## CT  0.75 0.71 0.74 0.64 0.76
## SI  0.36 0.36 0.36 0.27 0.28 0.32
## SPI 0.51 0.51 0.50 0.53 0.51 0.46 0.17
## SGI 0.49 0.49 0.48 0.49 0.49 0.45 0.18 0.80
```

Finalment, la última implementació que es demana.

```
cranberra <- function(mat){
  n <- ncol(mat)
  k <- nrow(mat)
```

```
  res <- matrix(0, n, n)
  for (i in 1:(n-1)){
    p <- mat[,i]
    for (j in (i+1):n){
      q <- mat[,j]
      num <- abs(p - q)
      denom <- abs(p) + abs(q)
      nzeros <- sum(denom==0)
      resy <- num/denom
      res[i, j] <- res[j, i] <- sum(resy, na.rm = T) * (k)/(k - nzeros)
    }
  }
  if (!is.null(colnames(mat))) dimnames(res) <- list(colnames(mat), colnames(mat))
  res
}
```

```
canberra_R_vicent <- cranberra(profiles)
max(abs(as.matrix(dist(t(profiles), method='canberra')) - canberra_R_vicent))
```

```
## [1] 3.552714e-15
```

Veiem que la major diferència entre la meva implementació i la de `R` és de l'ordre de 10^{-15}.

f)

De [7]:

> 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.

$\boldsymbol{K}$ és la matriu *kernel matrix*, anomenada com a $\boldsymbol{B}$ a Everitt 2011. Calculo primer de tot $\boldsymbol{B}$.

```
canberra <- as.matrix(dist(t(profiles), method='canberra'))
B <- B_from_D(canberra)
```

Un cop fet això, puc utilitzar un altre cop funcions fetes per mi per a comprovar si les distàncies són euclidianes.

```
source('../funcs/sym.R')
require(ade4)
positive_definite(B)
```

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

```
is.positive.definite(B) | is.positive.semi.definite(B)
```

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

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

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

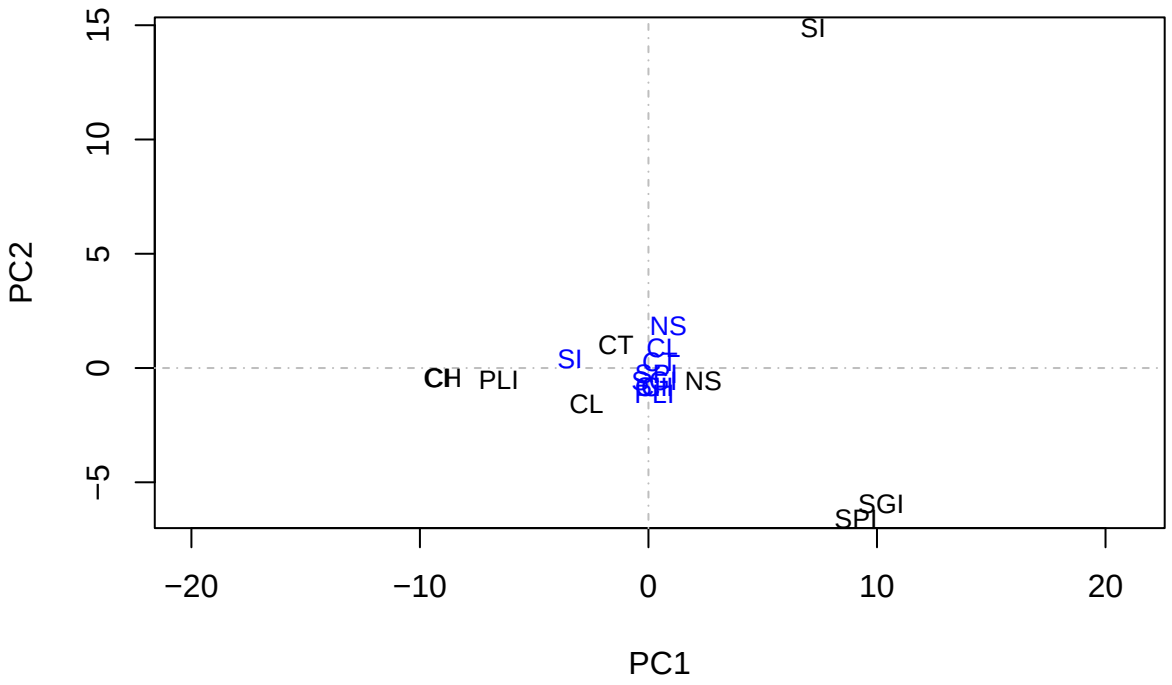
Veiem que $\boldsymbol{B}$ és positiva semi-definida, però no positiva definida. Així doncs, amb bastanta seguretat podem afirmar que aquesta matriu de distàncies és euclidiana. Vaig a obtenir una representació dels punts que han originat aquesta matriu de distàncies extraient els *eigen{values, vectors}* de $\boldsymbol{B}$.

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

```r
evecs <- eigen(B)$vectors[, 1:2]
evalues <- diag(sqrt(eigen(B)$values[1:2]))
X_pred_cran <- evecs %*% evalues
```

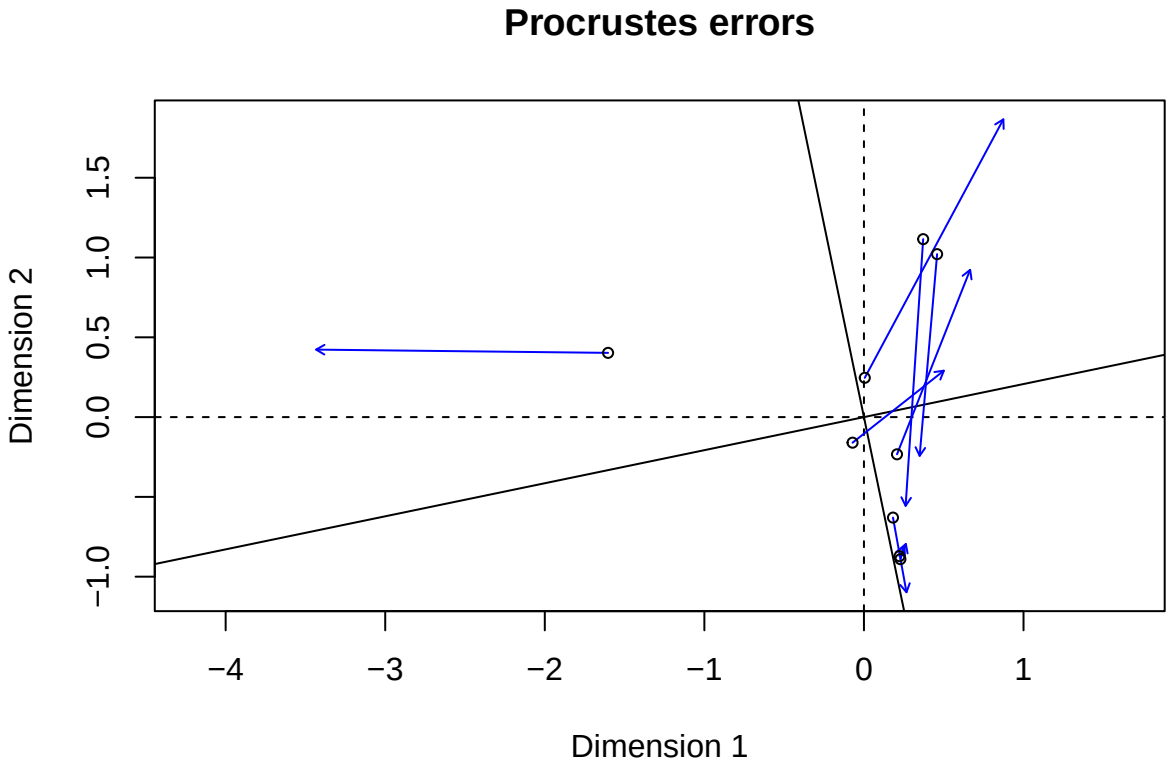
Un cop tenim aquests punts a l'espai bidimensional generat pels dos primers *eigenvectors* de $\boldsymbol{B}$, podem graficar-los amb les *labels* de les columnes del *dataset* `seabirds`. Primer de tot grafico els punts corresponents a les components principals obtingudes realitzant *MDS* amb la distància Xi-quadrat en blau i les components principals obtingudes en el cas d'utilitzar la distància de canberra en negre (amb `eqscplot`).

```r
eqscplot(X_pred_cran,ty="n",xlab="PC1",ylab="PC2")
abline(v=0,h=0, col="gray",lty=4)
text(X_pred_cran[, 1],X_pred_cran[,2],labels=colnames(X),cex=0.8)
text(X_pred[,1], X_pred[,2], labels=colnames(X), col="blue", cex=0.8)
```



Ara ho duc a terme amb la funció `procrustes` del paquet `vegan`, tal i com es demana.

```r
# install.packages('vegan')
require(vegan)
proc <- procrustes(X_pred, X_pred_cran)
plot(proc)
```



Podem observar que hi ha forces diferències entre l'escalat multidimensional utilitzant distàncies de *Canberra* i distàncies *Xi*-quadrat. Tot i això, utilitzant la funció `procrustes` sembla que les distàncies no són tan grans. Seguim tenint una columna (`SI`) força separada de les altres.

Apèndix

Comprovació dels meus resultats.

Exercici 2b)

```
tabla.N <- as.table(X)
n <- sum(tabla.N)
tabla.F <- tabla.N / n
stopifnot(max(abs(tabla.F - freq.rel)) < 1e-16)

margin.f <- apply(tabla.F,1,sum)
margin.c <- apply(tabla.F,2,sum)

tabla.Pc <- tabla.F %*% diag(1/margin.c)
stopifnot(max(abs(tabla.Pc - profiles)) < 1e-10)

colnames(tabla.Pc) <- colnames(tabla.N)
nc <- ncol(tabla.N)
D2c.chisq <- matrix(0,nc,nc)
for(i in 1:(nc-1))
  for(j in i:nc)
    D2c.chisq[i,j] <-
  t(tabla.Pc[,i]-tabla.Pc[,j]) %*% diag(1/margin.f) %*% (tabla.Pc[,i]-tabla.Pc[,j])
D2c.chisq <- D2c.chisq + t(D2c.chisq);
```

```r
rownames(D2c.chisq ) <- colnames(D2c.chisq) <- colnames(tabla.N);

stopifnot(max(abs(sqrt(D2c.chisq) - d.chisq.col)) < 1e-12)

mds.c <- cmdscale(sqrt(D2c.chisq),eig=TRUE)

stopifnot(max(abs(mds.c$points - X_pred)) < 1e-14)

Dc <- diag(margin.c)
Df <- diag(margin.f)
Dfmh <- diag(1/sqrt(margin.f))
Dcmh <- diag(1/sqrt(margin.c))
Z <- Dfmh %*% (tabla.F - margin.f %o% margin.c) %*% Dcmh
Z.svd <- svd(Z)
inercias <- Z.svd$d^2
stopifnot(abs(sum(inercias) - inertia(X)) < 1e-15)
```

Referències

Berg, Clas, and Hans-Göring Tiselius. 1986. "The Effect of pH on the Risk of Calcium Oxalate Crystallization in Urine." *European Urology* 12 (1): 59–61. https://doi.org/10.1159/000472578.

Carvalho, Mauricio. 2018. "Urinary pH in Calcium Oxalate Stone Formers: Does It Matter?" *Brazilian Journal of Nephrology* 40 (1): 6–7. https://doi.org/10.1590/1678-4685-jbn-2018-00010002.

Kavouras, Stavros A., Hyun-Gyu Suh, Marion Vallet, Michel Daudon, Andy Mauromoustakos, Mariacristina Vecchio, and Ivan Tack. 2021. "Urine Osmolality Predicts Calcium-Oxalate Crystallization Risk in Patients with Recurrent Urolithiasis." *Urolithiasis* 49 (5): 399–405. https://doi.org/10.1007/s00240-020-01242-2.

Werner, Helen, Shalmali Bapat, Michael Schobesberger, Doris Segets, and Sebastian P. Schwaminger. 2021. "Calcium Oxalate Crystallization: Influence of pH, Energy Input, and Supersaturation Ratio on the Synthesis of Artificial Kidney Stones." *ACS Omega* 6 (40): 26566–74. https://doi.org/10.1021/acsomega.1c03938.