Compte Rendu TD ACP

Samy Mekkaoui

26/02/2021

Introduction

Dans ce TP, nous allons nous intéresser à une méthode de codage de l'ACP que nous allons tester sur une base de données de 27 individus qui ont effectué des épreuves sportives. Le dataset que nous allons utiliser s'appelera "decathlon2" et on essayera de plot un graphique qui permettra de comprendre les différentes interactions entre les individus par rapport aux épreuves auxquels ils ont participé

Réponses aux Questions

Question n°4

```
d=10
n=2000
X=matrix(rnorm(d*n),nrow=d,ncol=n)

t0=proc.time()[3]
eigen(X%*%t(X))$values
```

```
##  [1] 2208.222 2150.224 2099.200 2072.880 1983.864 1952.912 1892.549
1852.639
##  [9] 1849.421 1799.979
```

```
proc.time()[3]-t0# On calcule le temps pour calculer les valeurs propres de
la matrice X%*%t(X)
```

```
## elapsed
##       0
```

```
t1=proc.time()[3]
ev=eigen(t(X)%*%X)$values
proc.time()[3]-t1#On calcule le temps pour calculer les valeurs propres de la
matrice t(X)X%*%X
```

```
## elapsed
##   10.41
```

```
#On voit que le temps de calcul des valeurs propres de X%*%t(X) est plus
rapide
```

Question n°5

On peut utiliser ce package car on sait que l'ACP permet de réduire la dimension et que les valeurs propres de module maximal sont celles qui vont permettre de mieux représenter la base de données initiale (On doit chercher l'inertie maximale pour ne pas que le dataset de base soit totalement différent)

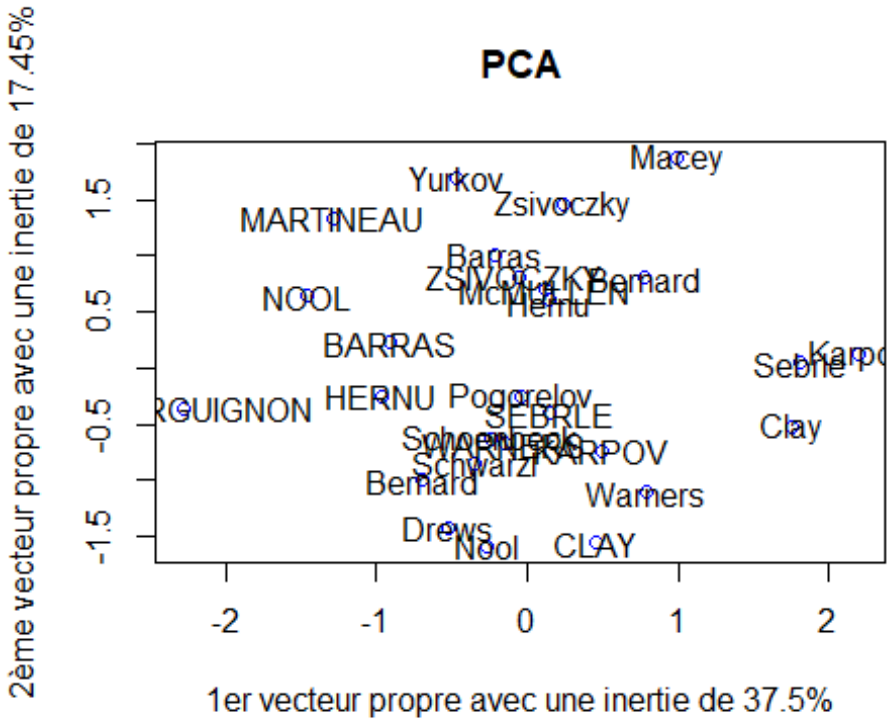
Question sur le codage de l'ACP

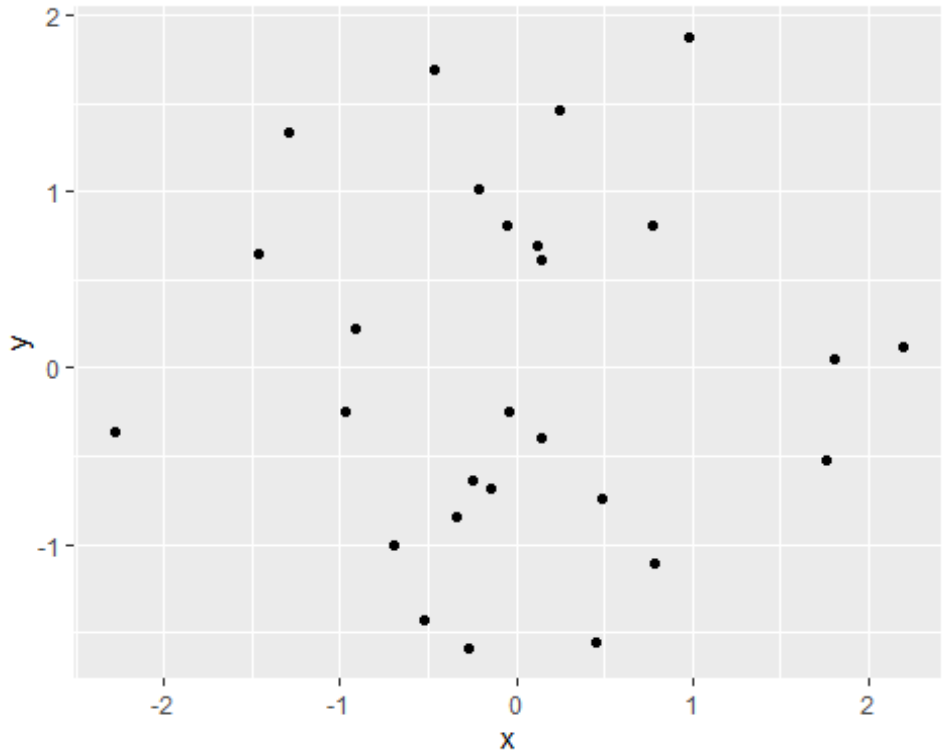
Partie Code

```
library(dplyr)
library(ggplot2)
library("FactoMineR")
library("factoextra")

decathlon2.active <- t(decathlon2[,1:10])
View(decathlon2.active)

a1=mean(decathlon2.active[1,])
a2=mean(decathlon2.active[2,])
a3=mean(decathlon2.active[3,])
a4=mean(decathlon2.active[4,])
a5=mean(decathlon2.active[5,])
a6=mean(decathlon2.active[6,])
a7=mean(decathlon2.active[7,])
a8=mean(decathlon2.active[8,])
a9=mean(decathlon2.active[9,])
a10=mean(decathlon2.active[10,])

a<-c(a1,a2,a3,a4,a5,a6,a7,a8,a9,a10)

m=matrix(1:1,nrow=10,ncol=27,byrow=TRUE)

b1=a1*m[1,]
b2=a2*m[2,]
b3=a3*m[3,]
b4=a4*m[4,]
b5=a5*m[5,]
b6=a6*m[6,]
b7=a7*m[7,]
b8=a8*m[8,]
b9=a9*m[9,]
b10=a10*m[10,]

l=c(b1,b2,b3,b4,b5,b6,b7,b8,b9,b10)

c=matrix(l,nrow=10,ncol=27,byrow=TRUE)
```

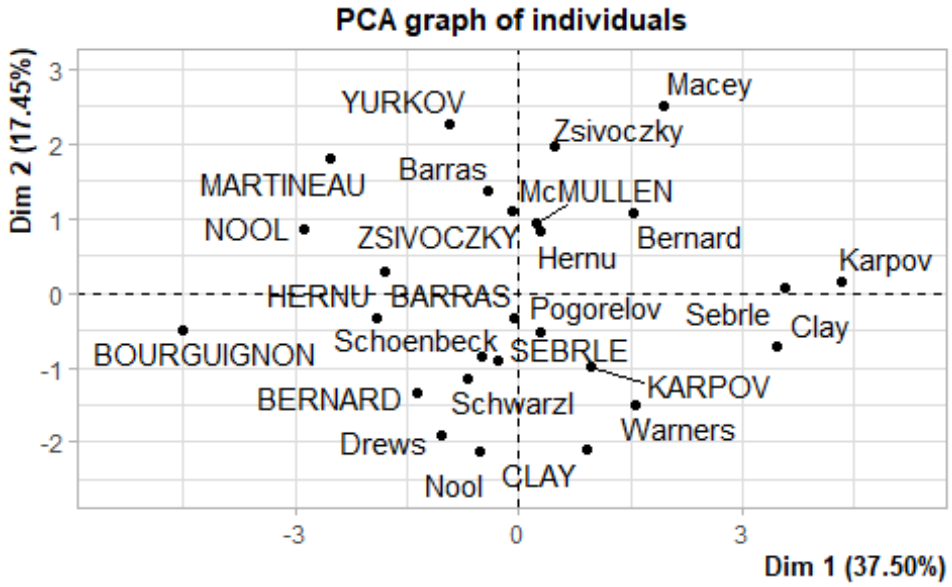
```r
De<-decathlon2.active-c


d1=sd(decathlon2.active[1,])
d2=sd(decathlon2.active[2,])
d3=sd(decathlon2.active[3,])
d4=sd(decathlon2.active[4,])
d5=sd(decathlon2.active[5,])
d6=sd(decathlon2.active[6,])
d7=sd(decathlon2.active[7,])
d8=sd(decathlon2.active[8,])
d9=sd(decathlon2.active[9,])
d10=sd(decathlon2.active[10,])

De[1,]<-De[1,]/d1
De[2,]<-De[2,]/d2
De[3,]<-De[3,]/d3
De[4,]<-De[4,]/d4
De[5,]<-De[5,]/d5
De[6,]<-De[6,]/d6
De[7,]<-De[7,]/d7
De[8,]<-De[8,]/d8
De[9,]<-De[9,]/d9
De[10,]<-De[10,]/d10

HatSigma<- (De%*%t(De))/26
View(HatSigma)

print(sum(diag(HatSigma))) # Variance totale du nuage de points

## [1] 10

a<-eigen(HatSigma)$vectors[,1]/(sqrt(eigen(HatSigma)$values[1]))

b<-eigen(HatSigma)$vectors[,2]/(sqrt(eigen(HatSigma)$values[2]))
P<-matrix(c(a,b),nrow=10,ncol=2,byrow=FALSE)

Projection2prem=t(De)%*%P
Projection2prem=-Projection2prem
```

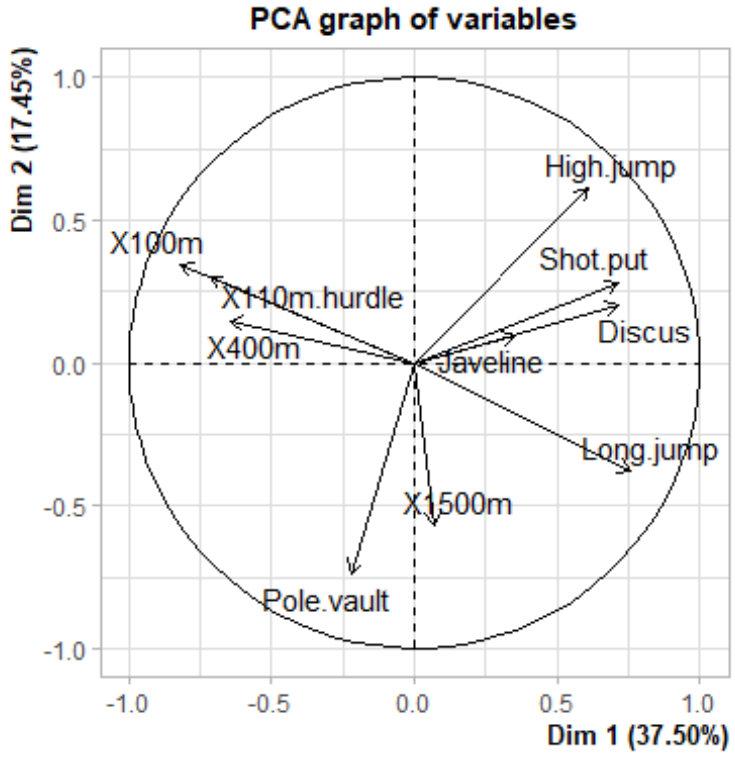
Partie sur le Plot des Graphiques






```
## Warning: ggrepel: 1 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```






```
## **Results for the Principal Component Analysis (PCA)**
## The analysis was performed on 27 individuals, described by 10 variables
## *The results are available in the following objects:
##
```

```
##    name               description
## 1  "$eig"             "eigenvalues"
## 2  "$var"             "results for the variables"
## 3  "$var$coord"       "coord. for the variables"
## 4  "$var$cor"         "correlations variables - dimensions"
## 5  "$var$cos2"        "cos2 for the variables"
## 6  "$var$contrib"     "contributions of the variables"
## 7  "$ind"             "results for the individuals"
## 8  "$ind$coord"       "coord. for the individuals"
## 9  "$ind$cos2"        "cos2 for the individuals"
## 10 "$ind$contrib"     "contributions of the individuals"
## 11 "$call"            "summary statistics"
## 12 "$call$centre"     "mean of the variables"
## 13 "$call$ecart.type" "standard error of the variables"
## 14 "$call$row.w"      "weights for the individuals"
## 15 "$call$col.w"      "weights for the variables"
```

Analyse des Plots

On peut remarquer que le plot ressemble bien à celui que l'on aurait obtenu en utilisant directement la fonction ACP déjà implémentée dans R donc nos résultats semblent cohérents