Assignment 3: Principal Components

Mathematics for Big Data

Sergi Vilardell

The goal of these exercises is to build a function to compute principal components analysis, this will be done step by step. At the end, `PCA()` function in `FactoMineR` library is explored, and the resutls compared.

```
library(FactoMineR)
library(fBasics)
```

1 Preliminaries

1. Use the data `decathlon` in package `FactoMineR`:

```
data(decathlon, package="FactoMineR")
data<-decathlon
```

2. Transform the variables involving race times, so that: $t \to \max - t$:

```
time.cols<- data[c(1,5,6,10)]
max.cols<-apply(time.cols,2,max)
transf<- sweep(time.cols,2,max.cols,FUN="-")
transf<- -transf
data[c(1,5,6,10)]<-transf
```

3. Separate basic data from supplementary data: variables 11, 12 and 13 are considered supplementary, numerical or qualitative, and cases 39 to 41 are considered as supplementary too:

```
dat<-data[-(39:41),-(11:13)] # basic data subet
ind.supp<- data[39:41,-(11:13)] # supplementary individua
var.supp<- data[-(39:41),11:12] # supplementary numerical variables
qual.supp<- as.factor(data[-(39:41),13]) # supplementary categorical variable
```

We have our data set, now we can begin performing PCA.

2 Steps in the function

We want to create a new function `pc()` to get principal componenents and all the rellevant matrices involved in the interpretation.

2.1 Normalizing

4. Normalize (scale) the data, i.e., substract the mean and divide by the standard deviation by columns.

```
n<-nrow(dat)
p<-ncol(dat)

centr<-apply(dat,2,mean)    # global center: mean vector of the basic subset
s<-apply(dat,2,sd)          # global st.dev: st. dev of the basic subset
```

```
dat.centr<-sweep(dat,2,centr,FUN="-")
dat.std <-sweep(dat.centr,2,s,FUN="/")

# ind.supp (if not null) must be scaled with respect to global center and st.dev
if (!is.null(ind.supp)){
ind.supp.centr<-sweep(ind.supp,2,centr,FUN="-")
ind.supp.std <-sweep(ind.supp.centr,2,s,FUN="/")}

# var.supp subset (if not null) must be scaled with respect its own center
if (!is.null(var.supp)){
var.supp.centr<-sweep(var.supp,2,apply(var.supp,2,mean),FUN="-")
var.supp.std <-sweep(var.supp.centr,2,apply(var.supp,2,sd),FUN="/")}

# Warning in treating the quali.supp variable !!
# 1st: get the column means for the factor groups
if (!is.null(qual.supp)){
dat.qual.supp<-lapply(split(data.frame(dat),qual.supp),colMeans) # its a list

# 2nd: convert to a data frame
dat.qual.supp<-data.frame(dat.qual.supp)

# 3rd: transpose to trate groups as individual cases
dat.qual.supp<-t(dat.qual.supp)

# 4th: center and scale using descriptives of basic data
qual.supp.centr<-sweep(dat.qual.supp,2,centr,FUN="-")
qual.supp.std<-sweep(qual.supp.centr,2,s,FUN="/")}
```

5. Getting the data subsets to whom principal components apply (factor $1/\sqrt{n-1}$ applies to obtain the covariance).

```
corr<-TRUE # standardized data
if (corr==T){ X<-dat.std*(1/sqrt(n-1))
if (!is.null(ind.supp)){X.ind.supp<-ind.supp.std*(1/sqrt(n-1))}
if (!is.null(var.supp)){X.var.supp<-var.supp.std*(1/sqrt(n-1))}
if (!is.null(qual.supp)){X.qual.supp<-qual.supp.std*(1/sqrt(n-1))}
Sigma<-t(as.matrix(X))%*%as.matrix(X)
Sinv<-diag(rep(1,p))} # "Sinv"" is the identity if corr=TRUE

#corr<-FALSE
if (corr==F) {X<-dat.centr*(1/sqrt(n-1))
if (!is.null(ind.supp)){X.ind.supp<-ind.supp.centr*(1/sqrt(n-1))}
if (!is.null(var.supp)){X.var.supp<-var.supp.centr*(1/sqrt(n-1))}
if (!is.null(qual.supp)){X.qual.supp<-qual.supp.centr*(1/sqrt(n-1))}
Sigma<-t(as.matrix(X))%*%as.matrix(X)
Sinv<-diag((diag(Sigma))^(-(1/2)))}
```

6. Get U, V and D components in **SVD** of X, you can use the function `svd`.

```
sing <- svd(X)
U <- sing$u
V <- sing$v
D <- diag(sing$d)
Lambda <- D^(2)
```

7. Compute relevant matrices for the variables (column analysis).

```r
# by columns: contribution of the variables into the components
var.contr<- 100*(V^2)
colSums(var.contr)
```

```
##  [1] 100 100 100 100 100 100 100 100 100 100
```

```r
rownames(var.contr)<-colnames(dat)
colnames(var.contr)<-paste("PC",1:ncol(var.contr),sep="")
var.contr
```

```
##                   PC1          PC2       PC3        PC4        PC5
## 100m        17.3330712  2.327978543  3.657786  4.6373102  2.5356924
## Long.jump   14.6035731  5.779536395  4.659357  3.9880665 15.1137201
## Shot.put    13.8795326  2.110718700 17.772208  0.7399725  8.9088142
## High.jump    9.0986575  7.947955252  6.673909  7.5017364 37.8197139
## 400m        11.9861005  0.008841597 22.983984  1.0275684  8.7196020
## 110m.hurdle 15.3539305  1.630494985  4.694060  7.0498291  0.8119038
## Discus      14.5512385  0.697690566  7.452333 10.4141557 21.2259895
## Pole.vault   0.1212022 37.834947785  5.018283 12.7029744  2.1740865
## Javeline     2.6725347  8.056501774 12.305120 43.3314060  0.3200159
## 1500m        0.4001593 33.605334402 14.782960  8.6069807  2.3704617
##                   PC6        PC7         PC8         PC9        PC10
## 100m         9.7878495 13.3851967 30.30745229  0.69018396 15.33747911
## Long.jump    3.3626976 27.2061645  0.81134529 20.14745141  4.32808790
## Shot.put     3.8387178 11.8963502  3.25089122  8.41964555 29.18314919
## High.jump   14.8280692  4.1101241  1.98876490  6.54803169  3.48303840
## 400m        12.0506530  0.7632945  0.96701357 38.16487809  3.32806387
## 110m.hurdle 18.9882180  8.3381721 42.03210924  0.06961239  1.03166970
## Discus       0.1102243 19.2631398  0.07420697  1.16043349 25.05058851
## Pole.vault  16.5223122  6.0402679  9.83802134  0.04840533  9.69949916
## Javeline    13.6036131  5.1786736  4.86298766  9.64042837  0.02871931
## 1500m        6.9076453  3.8186165  5.86720753 15.11092972  8.52970485
```

```r
# correlations between variables and components
C <- Sinv%*%V%*%D
rownames(C)<-colnames(dat)
colnames(C)<-paste("PC",1:ncol(var.contr),sep="")

# coordinates of the initial variables
var.coord <- V%*%D
rownames(var.coord)<-colnames(dat)
colnames(var.coord)<-paste("PC",1:ncol(var.contr),sep="")

# squared cosines
var.cos2 <- C^2
rownames(var.cos2)<-colnames(dat)
colnames(var.cos2)<-paste("PC",1:ncol(var.contr),sep="")
```

8. Compute the relevant matrices for the individuals, as in 7.

```r
#Scores
Y <- as.matrix(X)%*%V*sqrt(n)
colnames(Y)<-paste("PC",1:ncol(var.contr),sep="")

#Contribution of the individuals into the components
```

```
sqrt.scores <- Y^2
sum <- colSums(sqrt.scores)
ind.contr <- sweep(sqrt.scores, 2, sum,FUN="/")
ind.contr <- 100*ind.contr
colSums(ind.contr)
```

```
##  PC1  PC2  PC3  PC4  PC5  PC6  PC7  PC8  PC9 PC10
##  100  100  100  100  100  100  100  100  100  100
```

```
#Individuals inertias
ind.iner <- rowSums(sqrt.scores)
ind.dist <- sqrt(ind.iner)

#Individuals squared cosines
ind.cos2 <- sweep(sqrt.scores, 2, ind.iner,FUN="/")
```

```
## Warning in sweep(sqrt.scores, 2, ind.iner, FUN = "/"): STATS is longer than
## the extent of 'dim(x)[MARGIN]'
```

9. Make a `list` with all the results above computed, using a reduced number of digits d, for instance $d = 3$. Add to the list result the coordinates for the supplementary objects:

```
# assume k=3
var.supp.coord <- as.matrix(t(X.var.supp))%*%U
qual.supp.coord <- as.matrix(X.qual.supp)%*%V*sqrt(n)
ind.supp.coord <- as.matrix(X.ind.supp)%*%V*sqrt(n)

result<-list(left.sing.vec =  round(U, digits = 3),
             right.sing.vec = round(V, digits = 3),
             diag.sing.vec = round(D, digits = 3),
             lambda. = round(Lambda, digits = 3),
             diag.sing = round(D, digits = 3),
             var.contr = round(var.contr, digits = 3),
             corr = round(C, digits = 3),
             var.coord = round(var.coord, digits = 3),
             var.cos2 = round(var.cos2, digits = 3),
             ind.coord = round(Y, digits = 3),
             ind.contr = round(ind.contr, digits = 3),
             ind.iner= round(ind.iner, digits = 3),
             ind.dist = round(ind.dist, digits = 3),
             ind.cos2 = round(ind.cos2, digits = 3),
             qual.supp.coord = round(qual.supp.coord, digits = 3),
             ind.supp.coord = round(ind.supp.coord, digits = 3))
```

The function `pc()`

11. Write the complete code for the function.

```
pc<-function(dat,ind.supp=NULL,var.supp=NULL,qual.supp=NULL,corr=TRUE,k=3) #default
{
data(decathlon, package="FactoMineR")
data<-decathlon

time.cols<- data[c(1,5,6,10)]
```

```
max.cols<-apply(time.cols,2,max)
transf<- sweep(time.cols,2,max.cols,FUN="-")
transf<- -transf
data[c(1,5,6,10)]<-transf

n<-nrow(dat)
p<-ncol(dat)

centr<-apply(dat,2,mean)   # global center: mean vector of the basic subset
s<-apply(dat,2,sd)         # global st.dev: st. dev of the basic subset

dat.centr<-sweep(dat,2,centr,FUN="-")
dat.std <-sweep(dat.centr,2,s,FUN="/")

# ind.supp (if not null) must be scaled with respect to global center and st.dev
if (!is.null(ind.supp)){
ind.supp.centr<-sweep(ind.supp,2,centr,FUN="-")
ind.supp.std <-sweep(ind.supp.centr,2,s,FUN="/")}

# var.supp subset (if not null) must be scaled with respect its own center
if (!is.null(var.supp)){
var.supp.centr<-sweep(var.supp,2,apply(var.supp,2,mean),FUN="-")
var.supp.std <-sweep(var.supp.centr,2,apply(var.supp,2,sd),FUN="/")}

# Warning in treating the quali.supp variable !!
# 1st: get the column means for the factor groups
if (!is.null(qual.supp)){
dat.qual.supp<-lapply(split(data.frame(dat),qual.supp),colMeans) # its a list

# 2nd: convert to a data frame
dat.qual.supp<-data.frame(dat.qual.supp)

# 3rd: transpose to trate groups as individual cases
dat.qual.supp<-t(dat.qual.supp)

# 4th: center and scale using descriptives of basic data
qual.supp.centr<-sweep(dat.qual.supp,2,centr,FUN="-")
qual.supp.std<-sweep(qual.supp.centr,2,s,FUN="/")}

#corr<-TRUE
if (corr==T){ X<-dat.std*(1/sqrt(n-1))
if (!is.null(ind.supp)){X.ind.supp<-ind.supp.std*(1/sqrt(n-1))}
if (!is.null(var.supp)){X.var.supp<-var.supp.std*(1/sqrt(n-1))}
if (!is.null(qual.supp)){X.qual.supp<-qual.supp.std*(1/sqrt(n-1))}
Sigma<-t(as.matrix(X))%*%as.matrix(X)
Sinv<-diag(rep(1,p))} # "Sinv"" is the identity if corr=TRUE

#corr<-FALSE
if (corr==F) {X<-dat.centr*(1/sqrt(n-1))
if (!is.null(ind.supp)){X.ind.supp<-ind.supp.centr*(1/sqrt(n-1))}
if (!is.null(var.supp)){X.var.supp<-var.supp.centr*(1/sqrt(n-1))}
if (!is.null(qual.supp)){X.qual.supp<-qual.supp.centr*(1/sqrt(n-1))}
Sigma<-t(as.matrix(X))%*%as.matrix(X)
```

```r
Sinv<-diag((diag(Sigma))^(-(1/2)))}

#SVD
sing <- svd(X)
U <- sing$u
V <- sing$v
D <- diag(sing$d)
Lambda <- D^(2)

# by columns: contribution of the variables into the components
var.contr<- 100*(V^2)
colSums(var.contr)
rownames(var.contr)<-colnames(dat)
colnames(var.contr)<-paste("PC",1:ncol(var.contr),sep="")


# correlations between variables and components
C <- Sinv%*%V%*%D
rownames(C)<-colnames(dat)
colnames(C)<-paste("PC",1:ncol(var.contr),sep="")

# coordinates of the initial variables
var.coord <- V%*%D
rownames(var.coord)<-colnames(dat)
colnames(var.coord)<-paste("PC",1:ncol(var.contr),sep="")

# squared cosines
var.cos2 <- C^2
rownames(var.cos2)<-colnames(dat)
colnames(var.cos2)<-paste("PC",1:ncol(var.contr),sep="")

var.supp.coord <- as.matrix(t(X.var.supp))%*%U
qual.supp.coord <- as.matrix(X.qual.supp)%*%V*sqrt(n)
ind.supp.coord <- as.matrix(X.ind.supp)%*%V*sqrt(n)

result<-list(left.sing.vec =  round(U, digits = k),
             right.sing.vec = round(V, digits = k),
             diag.sing.vec = round(D, digits = k),
             lambda.sing.vec = round(Lambda, digits = k),
             var.contr = round(var.contr, digits = k),
             corr = round(C, digits = k),
             var.coord = round(var.coord, digits = k),
             var.cos2 = round(var.cos2, digits = k),
             ind.coord  = round(Y, digits = k),
             ind.contr = round(ind.contr, digits = k),
             ind.iner = round(ind.iner, digits = k),
             ind.dist = round(ind.dist, digits = k),
             ind.cos2 = round(ind.cos2, digits = k),
             qual.supp.coord = round(qual.supp.coord, digits = k),
             ind.supp.coord = round(ind.supp.coord, digits = k),
             var.supp.coord = round(var.supp.coord, digits = k))
}
```

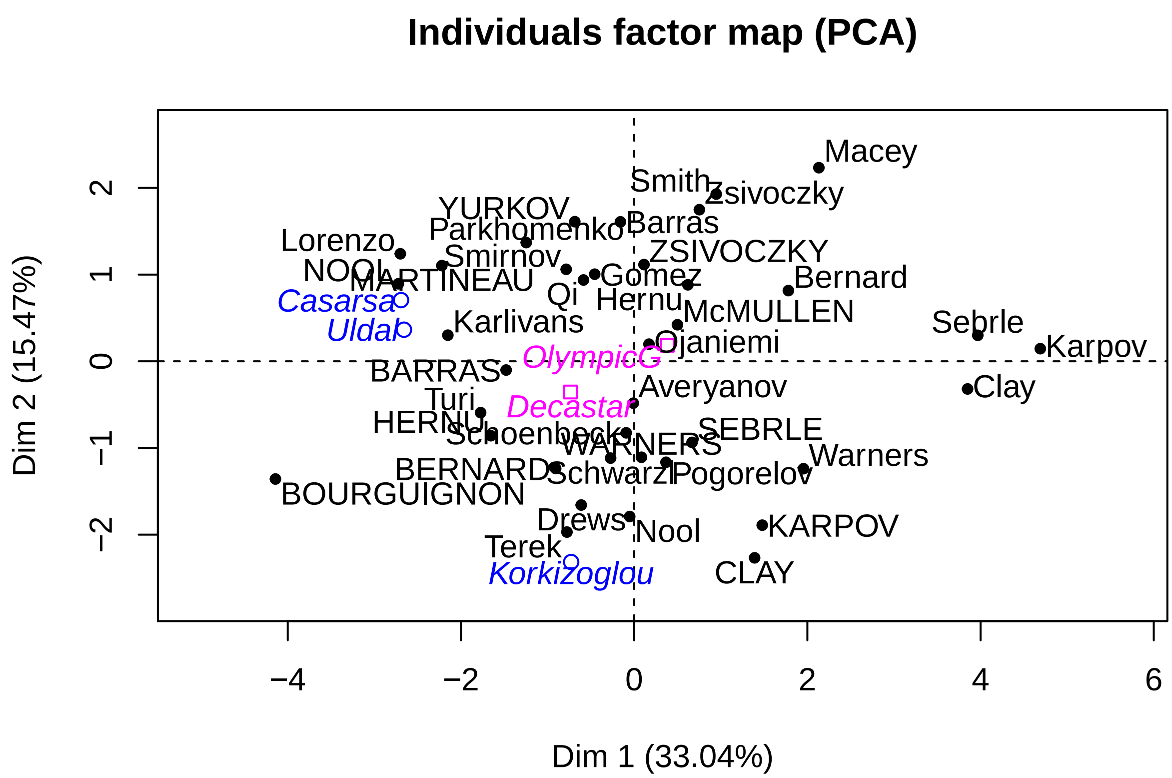
12. Apply function `pc()` in the following two cases and explore within objects `out` and `out2`.

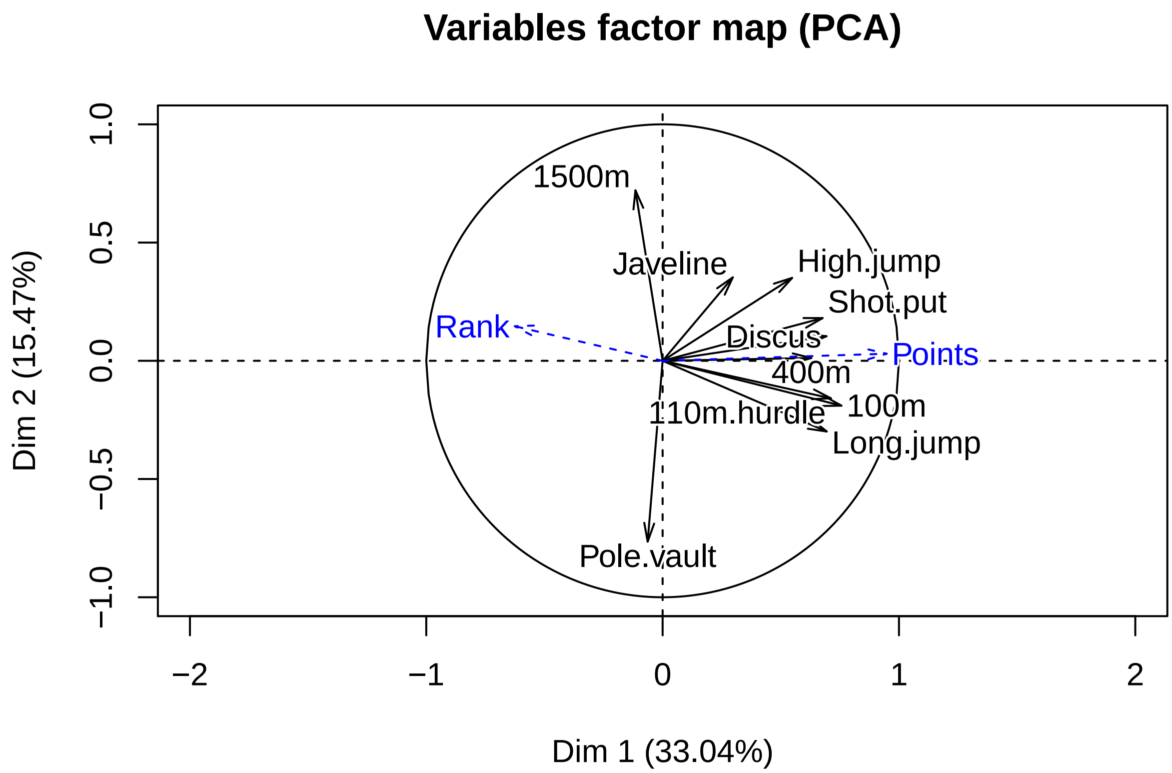
```
out<-pc(dat,ind.supp=NULL,var.supp=NULL,qual.supp=NULL,corr=TRUE)
out2<-pc(dat, ind.supp=ind.supp, var.supp=var.supp, qual.supp=qual.supp, corr=TRUE,k=8)
```

4 PCA() function in package FactoMineR

13. Compare the result in out2 with the ones provided by the function PCA().

```
res<-PCA(data, ncp = 10, ind.sup = 39:41, quanti.sup = c(11: 12), quali.sup = 13,  graph=T)
```





We can compare some matrices of `pca()` and `PCA()` to see if they match. Some eigenvectors are multiplied by a factor (-1). In order to compare them properly we take the absolute value of the matrices:

```
#SVD
all.equal(abs(out2$right.sing.vec),abs(res$svd$V))
```

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

```
#Var. Coord. (Change the name of the columns, if not it gives an error...)
colnames(out2$var.coord)<-paste("Dim.",1:ncol(var.contr),sep="")
all.equal(abs(out2$var.coord),abs(res$var$coord))
```

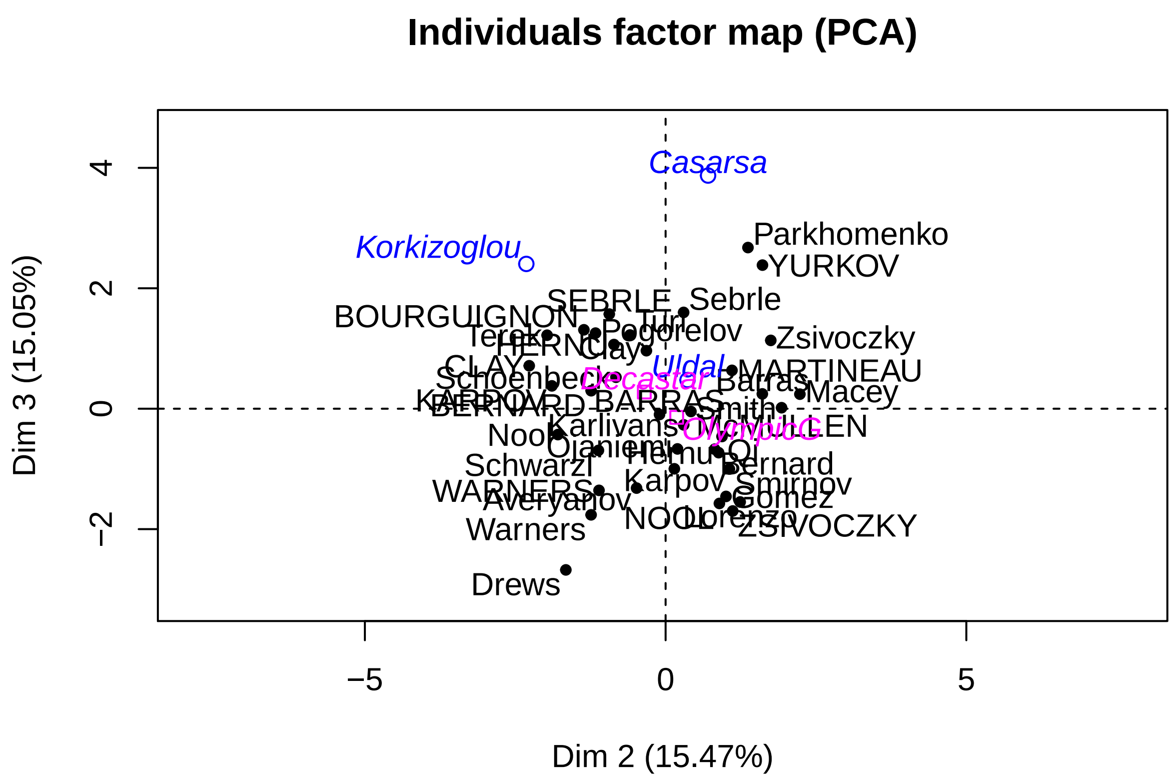
```
## [1] TRUE
```

```
#Ind. Coord.
colnames(out2$ind.coord)<-paste("Dim.",1:ncol(var.contr),sep="")
all.equal(abs(out2$ind.coord),abs(res$ind$coord))
```

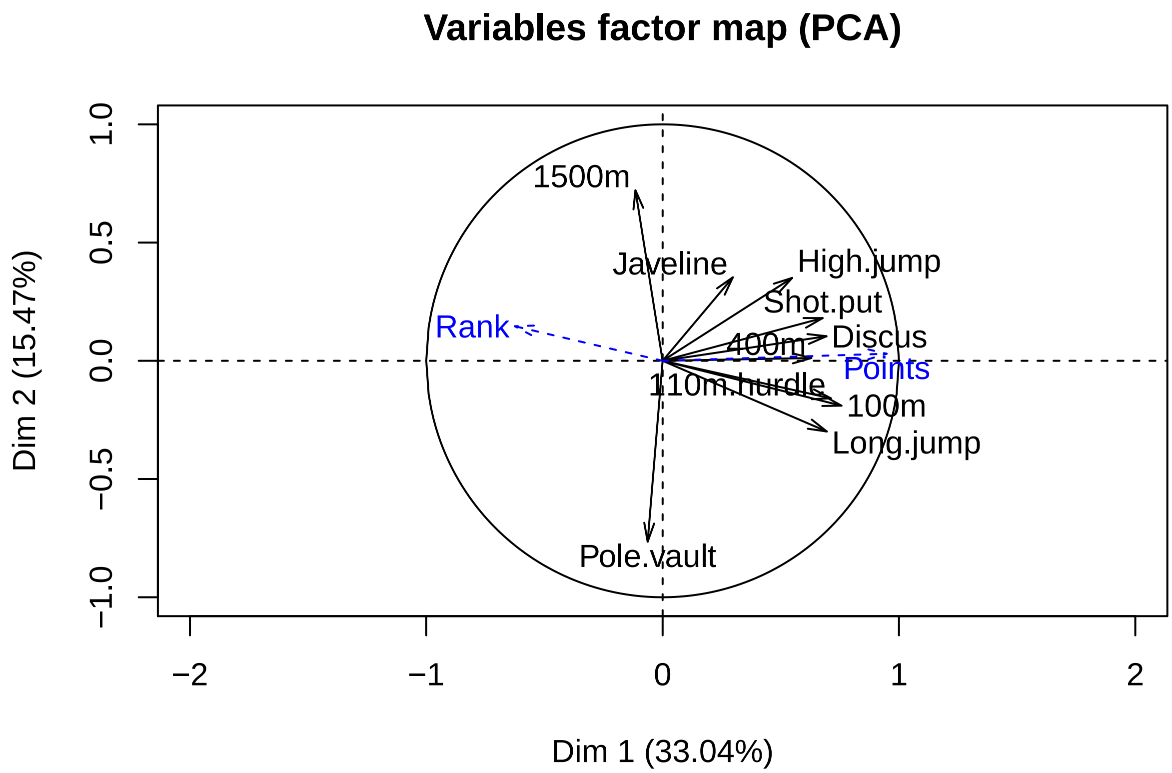
```
## [1] TRUE
```

14. Function `PCA()` has the default option graph=TRUE that displays 2 graphs. In particular, explore the arguments options: `axes`, `choix`, `ellipse`, `xlim`, `ylim`, `habillage`, `invisible`, `lim.cos2.var`, `select`, etc. Use these options to explore other components 3 and 4, to represent the athletes in a diferent color depending on `Decastar` or `OlymplicG`, etc.

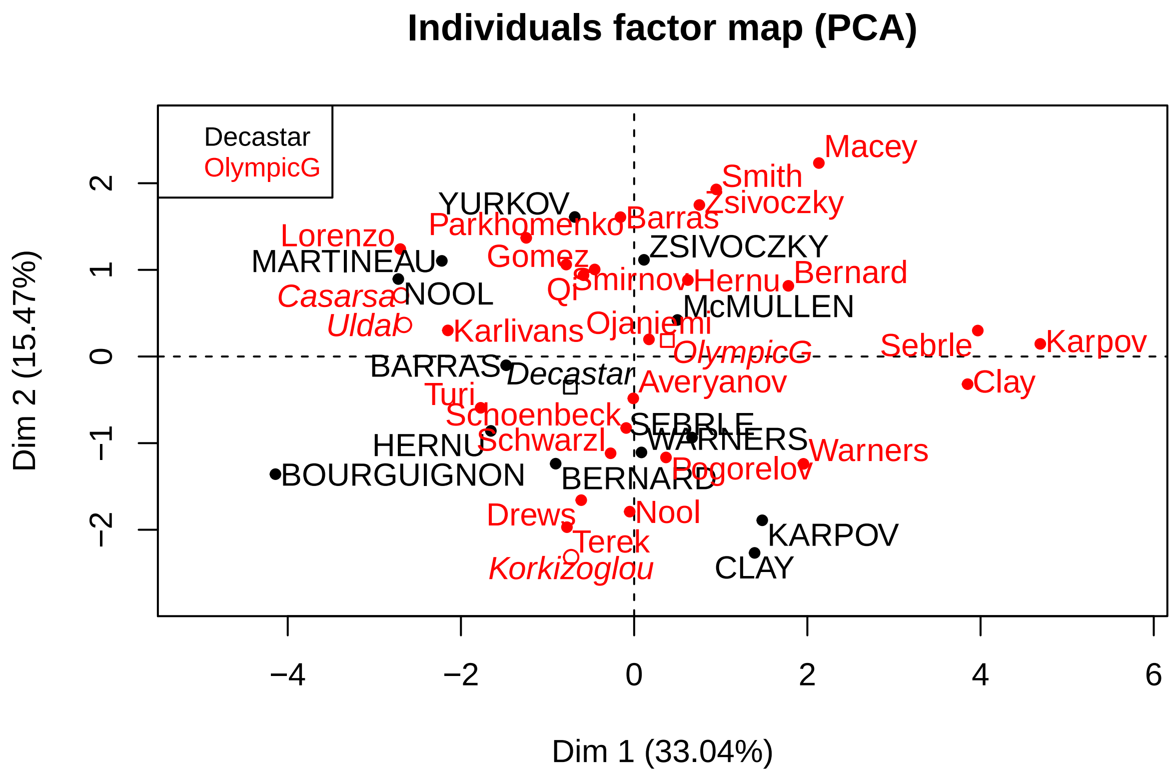
```
#Axes selects the components to plot
plot.PCA(res,axes = c(2,3))
```



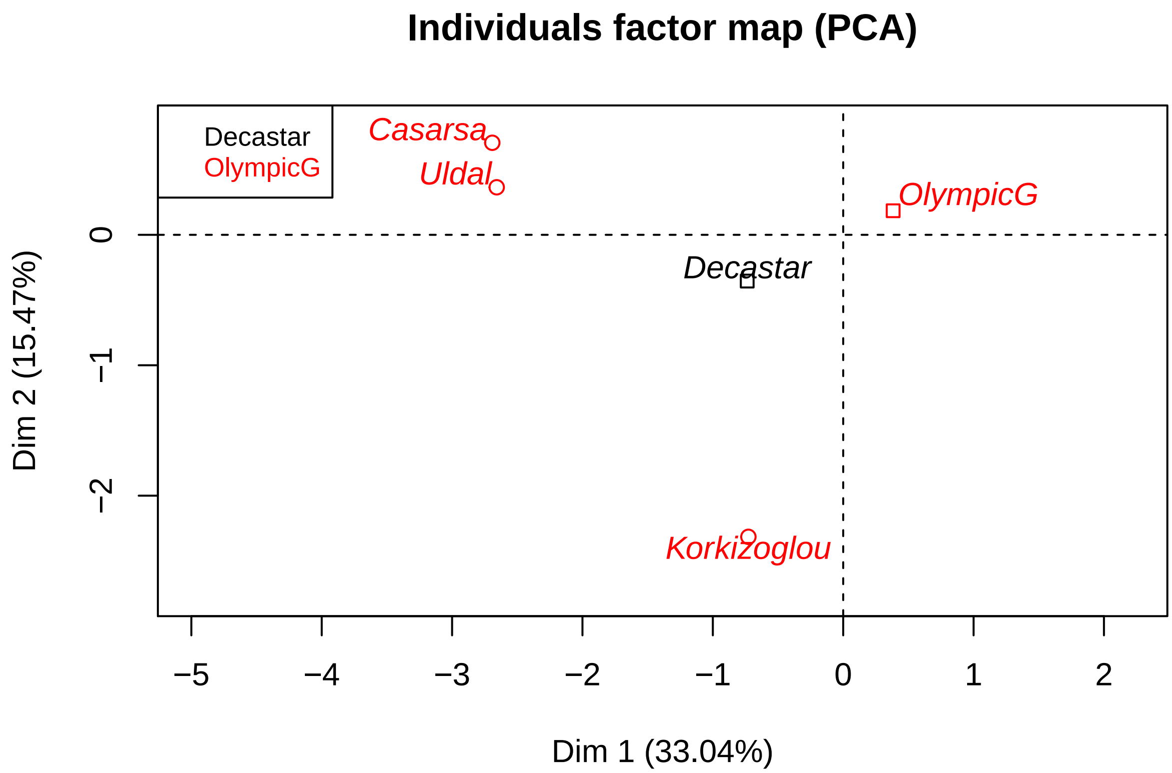

```r
#plot the variables
plot.PCA(res, choix = "var")
```



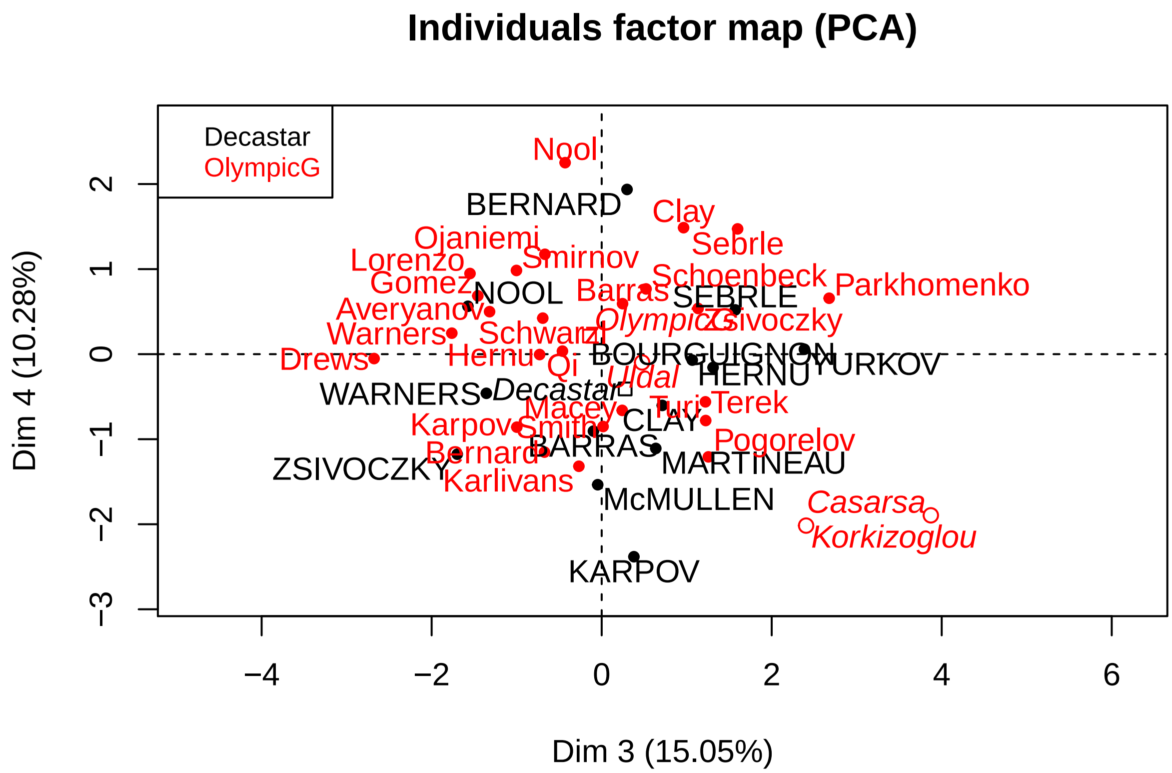

```r
#plot ind. by group
plot.PCA(res, choix="ind", habillage=13)
```




```r
#Make ind invisible
plot.PCA(res, choix="ind", habillage=13, invisible = "ind")
```




```r
#Comp 3 and 4
plot.PCA(res,axes = c(3,4), habillage = 13)
```



Individuals factor map (PCA)
Decastar
OlympicG
Dim 4 (10.28%)
Dim 3 (15.05%)
Nool
BERNARD
Clay
Sebrle
Ojaniemi
Lorenzo
Smirnov
Gomez
NOOL
Schoenbeck
Parkhomenko
Barras
SEBRLE
Averyanov
OlympicG
Zsivoczky
Warners
Schwarzl
Drews
Hernu
BOURGUIGNON
Qi
YURKOV
Uldal
HERNU
WARNERS
Decastar
Terek
Macey
Turi
Karpov
Smith
CLAY
Bernard
BARRAS
Pogorelov
ZSIVOCZKY
MARTINEAU
Karlivans
McMULLEN
Casarsa
Korkizoglou
KARPOV