PCA BATEADORES

Antonio Hernández

2022-05-25

Para este analisis de componentes principales es necesario instalar la paqueteria datos que trae por defecto base de datos precargadas.

```
install.packages("datos")
```

```
## Installing package into '/cloud/lib/x86_64-pc-linux-gnu-library/4.2'
## (as 'lib' is unspecified)
```

Ya teniendo descargada la pqueteria será solo cuestion de mandarla a llamr.

```
##Llamar a la libreria
library(datos)
```

La base de datos a elegir es bateadores y esta la guardaremos en un odjeto llamado x

```
x<-data.frame(bateadores)
```

Para la depuracion de la base de datos ocuparemos la libreria *dplyr*, y seleccionaremos de la columna 6 a la comuna 12, recortando las filas a 300 para una mejor manipulación.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

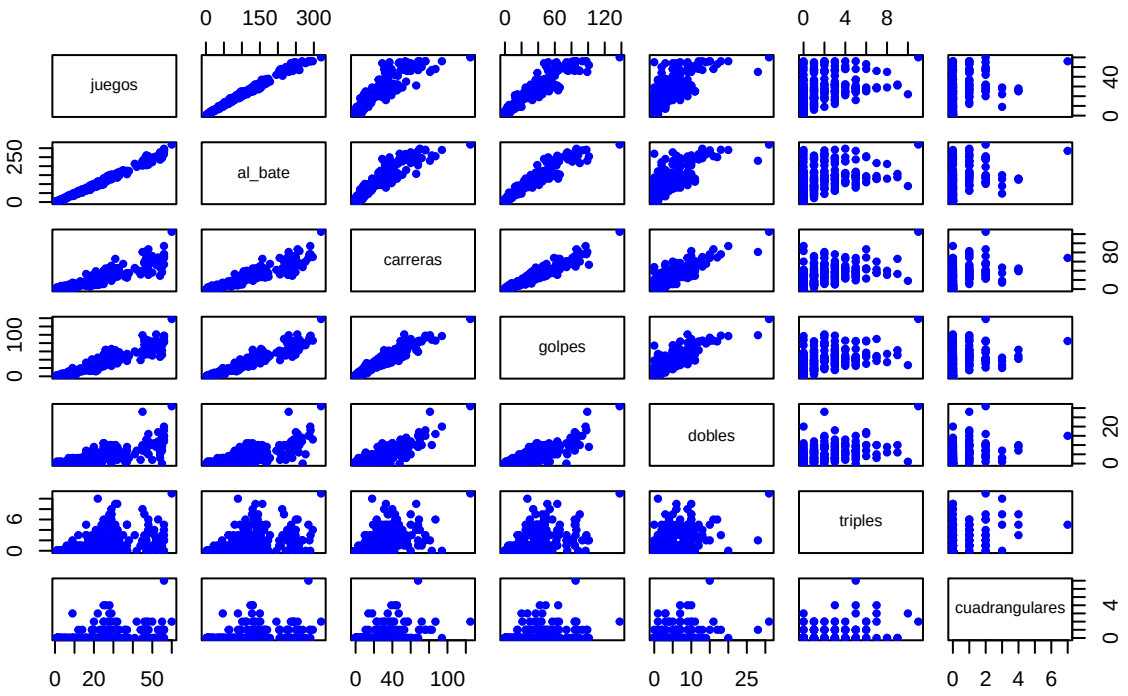
```
x<-x[1:300,6:12]
```

Definiremos como n y p el número de filas y columnas.

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

Se genera un scatterplot de las variables originales.

```
pairs(x,col="blue", pch=20,
      main="Variables originales")
```



Se obtienen los componentes principales con base en la matriz de covarianza muestral.

```
mu<-colMeans(x)
s<-cov(x)
```

Obtención de los componentes principales con base a la matriz de covarianza. muestral

```
es<-eigen(s)
es
```

```
## eigen() decomposition
## $values
## [1] 7257.781725   86.719935   18.425852    4.320619    3.039264    1.896881
## [7]    0.505043
##
## $vectors
##               [,1]        [,2]         [,3]          [,4]          [,5]
## [1,] -0.182171562  0.16452868  0.021349903  0.1450689820 -0.4554342130
## [2,] -0.910402018  0.33570774 -0.077265228 -0.0003754638  0.1006374790
## [3,] -0.231181369 -0.70031454 -0.651574632 -0.1339743544 -0.0218155680
## [4,] -0.287470814 -0.57375471  0.754292129 -0.1281965394 -0.0002413109
## [5,] -0.041567428 -0.18890525  0.006932106  0.9629530302  0.1826146844
## [6,] -0.012554147 -0.06824178 -0.002107990  0.1273491825 -0.8596331129
## [7,] -0.003454904 -0.01592395 -0.003956120  0.0327874443 -0.0982781081
##              [,6]         [,7]
## [1,]  0.84213928  0.040300422
## [2,] -0.20561596 -0.009622859
## [3,]  0.11462989 -0.000400816
## [4,]  0.05261846  0.002516171
## [5,] -0.03837975 -0.020957964
## [6,] -0.46946290 -0.139995318
## [7,] -0.10366474  0.989059583
```

Matriz de auto-valores.

```
eigen.val<-es$values
```

Matriz de auto-vectores.

```
eigen.vec<-es$vectors
```

Proporción de variabilidad para cada vector.

```
pro.var<-eigen.val/sum(eigen.val)
```

Proporción de variabilidad acumulada.

```
pro.var.acum<-cumsum(eigen.val)/sum(eigen.val)
pro.var.acum
```

```
## [1] 0.9844144 0.9961767 0.9986760 0.9992620 0.9996742 0.9999315 1.0000000
```

Media de los auto-valores.

```
mean(eigen.val)
```

```
## [1] 1053.241
```

##La obrtención de los coeficientes de las nuevas variables.

Centrar los datos con respecto a la media.

```
ones<-matrix(rep(1,n),nrow=n, ncol=1)
```

Construcción de la matriz centrada.

```
X.cen<-as.matrix(x)-ones%*%mu
```

Construcción de la matriz diagonal de las varianzas.

```
Dx<-diag(diag(s))
Dx
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]      [,7]
## [1,] 245.2835    0.000   0.0000   0.0000  0.00000 0.000000 0.0000000
## [2,]   0.0000 6025.475   0.0000   0.0000  0.00000 0.000000 0.0000000
## [3,]   0.0000    0.000 438.3485   0.0000  0.00000 0.000000 0.0000000
## [4,]   0.0000    0.000   0.0000 638.8867  0.00000 0.000000 0.0000000
## [5,]   0.0000    0.000   0.0000   0.0000 19.74666 0.000000 0.0000000
## [6,]   0.0000    0.000   0.0000   0.0000  0.00000 4.291761 0.0000000
## [7,]   0.0000    0.000   0.0000   0.0000  0.00000 0.000000 0.6573467
```

Construcción de la matriz centrada multiplicada por (Dx)^1/2. Donde Y son los datos normalizados.

```
Y<-X.cen%*%solve(Dx)^(1/2)
```

Construcción de los coeficientes o scores eigen.vec matriz de autovectores.

```
scores<-Y%*%eigen.vec
```

Nombramos las columnas PC1...PC8.

```
colnames(scores)<-c("PC1","PC2","PC3","PC4","PC5",
                    "PC6", "PC7")
```

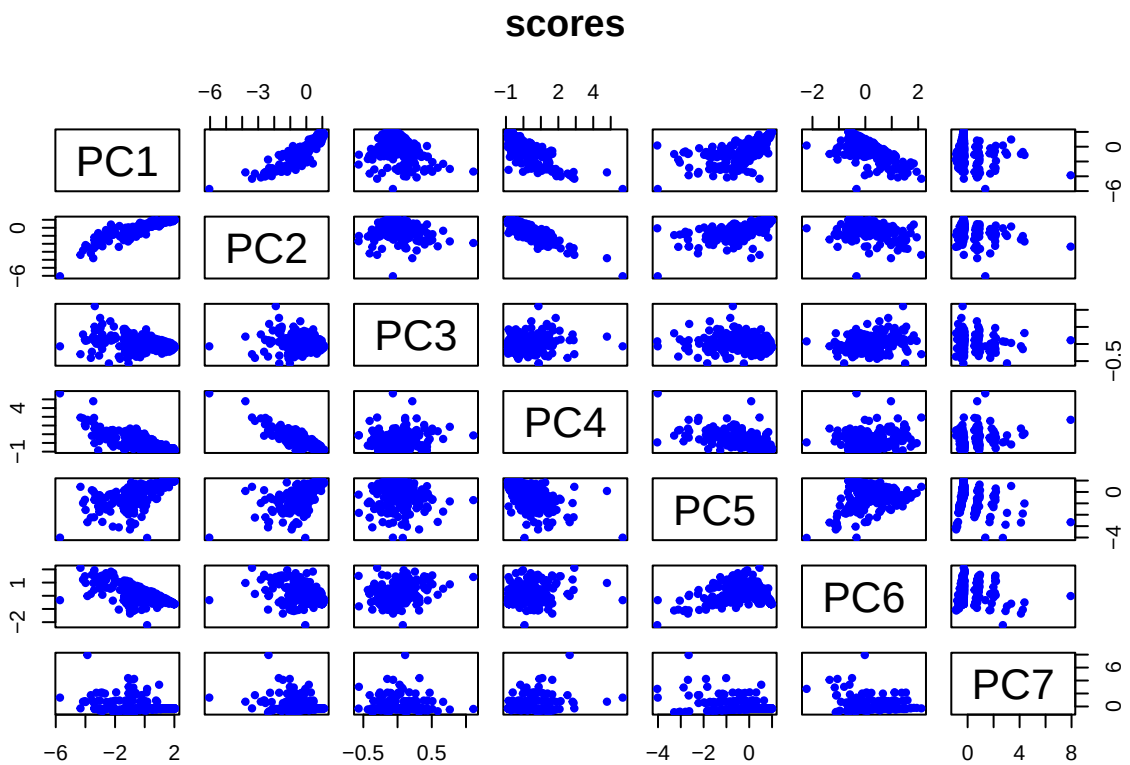
La visualizamos los scores.

```
scores[1:10,1:7]
```

```
##            PC1        PC2         PC3        PC4        PC5        PC6
## 1    1.9908961  0.9960832 -0.08901533 -0.8312306  0.9591387 -0.6232981
## 2   -0.3766240 -0.2436783 -0.13910023  0.3364541  0.6240267  0.5442853
## 3   -0.7264832 -0.3138654  0.14021072  0.2195980 -1.6225899 -0.4010426
## 4   -0.7482667 -0.6384954  0.26348730  1.3774661 -0.3206673 -0.1173662
## 5   -0.5336019 -0.6718505  0.10367122  1.5752557 -0.4117790 -0.1749365
## 6    1.0859197  0.6375606 -0.06853748 -0.3480674  0.3553598 -0.3227256
## 7    1.9795229  0.9733838 -0.05917335 -0.8363024  0.9591291 -0.6212163
## 8   -1.7458537 -2.3863342 -0.36791558  1.4242997 -3.1078630 -1.0488160
## 9    1.9473985  0.9017489 -0.08972982 -0.6260065  1.0004786 -0.6270270
## 10   0.5152655  0.6814149 -0.16198755 -0.3284094  0.2246636 -0.0720430
##             PC7
## 1   -0.3105721
## 2   -0.2886340
## 3   -0.6083118
## 4    2.0016730
## 5   -0.5144770
## 6   -0.3639317
## 7   -0.3104725
## 8   -0.9026865
## 9   -0.3153319
## 10  -0.3529567
```

Elaboración del grafico de los scores.

```
pairs(scores, main="scores", col="blue", pch=20)
```



##Sinteti-
zado PCA Aplicar el calculo de la varianza a las columnas # 1=filas, 2=columnas

```
apply(x, 2, var)
```

```
##         juegos        al_bate       carreras         golpes         dobles
##    245.2835117   6025.4748718    438.3484504    638.8867224     19.7466555
##        triples cuadrangulares
##      4.2917614      0.6573467
```

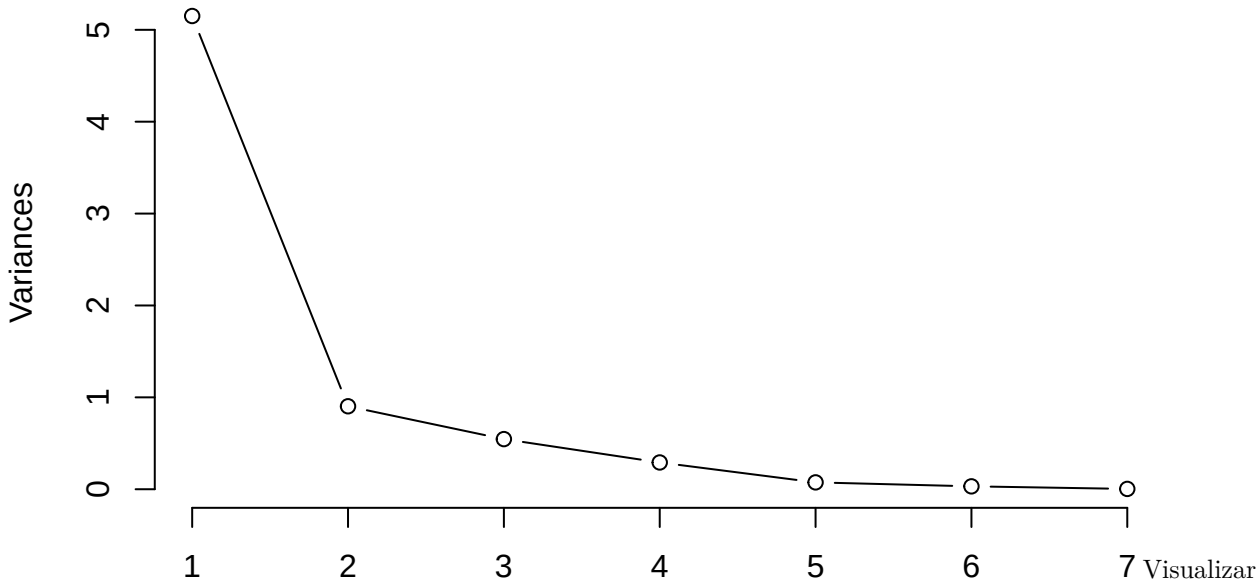
Centrar por la media y escalada por la desviacion standar (dividir entre sd).

```
acp<-prcomp(x, center=TRUE, scale=TRUE)
acp
```

```
## Standard deviations (1, .., p=7):
## [1] 2.26938475 0.95017347 0.73869027 0.53989725 0.27346086 0.17747643 0.06026748
##
## Rotation (n x k) = (7 x 7):
##                       PC1         PC2         PC3         PC4         PC5
## juegos          0.4146375  0.22045186 -0.12667847  0.39592548  0.42681087
## al_bate         0.4211933  0.21347764 -0.13209092  0.31911765  0.21069390
## carreras        0.4265479  0.10954975 -0.01824044 -0.10434728 -0.74495874
## golpes          0.4297059  0.14734399 -0.05087653  0.02898195 -0.26729087
## dobles          0.3899897  0.04407697  0.03457415 -0.83548958  0.37895284
## triples         0.2914381 -0.45093978  0.82344922  0.17310996  0.05753562
## cuadrangulares 0.2169181 -0.81659501 -0.53321613  0.04085020  0.01081801
##                         PC6          PC7
## juegos          -0.196985593 -0.621048712
## al_bate         -0.127729328  0.772678904
## carreras        -0.479664408 -0.098960690
## golpes           0.843705006 -0.083256417
## dobles          -0.054109031  0.013927538
## triples         -0.005935334  0.018327088
## cuadrangulares   0.001672873 -0.003331711
```

Generar del gr?fico screeplot

```
plot(acp, type="l")
```



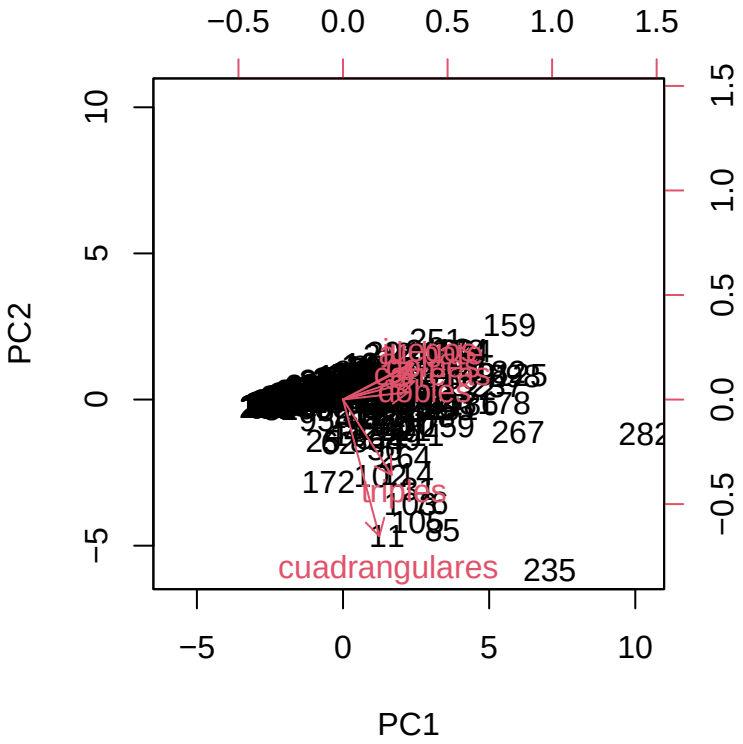
Visualizar el resumen

```
summary(acp)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5    PC6     PC7
## Standard deviation     2.2694 0.9502 0.73869 0.53990 0.27346 0.1775 0.06027
## Proportion of Variance 0.7357 0.1290 0.07795 0.04164 0.01068 0.0045 0.00052
## Cumulative Proportion  0.7357 0.8647 0.94266 0.98430 0.99498 0.9995 1.00000
```

Construcción del Biplot.

```
biplot(acp, scale=0)
```


−0.5
0.0
0.5
1.0
1.5
10
5
0
−5
PC2
1.5
1.0
0.5
0.0
−0.5
159
267
282
cuadrangulares
235
−5
0
5
10
PC1