Isaac Rodríguez Bribiesca 10/Feb/2020

1. Sean $C \in \mathbb{R}^d$, $F \in \mathbb{R}^d$ los vectores aleatorios que representan las temperaturas del cuerpo, en grados Celsius y grados Fahrenheit, respectivamente.

Se nos dice que ambos vectores están relacionados de la siguiente manera:

$$F = a + bC, \quad \text{con constantes } a \in \mathbb{R}^d, \ b \in \mathbb{R}$$

Sea $Cov(C)$ la matriz de covarianza de C. Entonces:

$$(Ea + bEC)(Ea^T + bEC^T)$$

$$Cov(F) = Cov(a+bC)$$

$$= E[(a+bC)(a+bC)^T] - E(a+bC)E(a+bC)^T$$

$$= E(aa^T + baC^T + bCa^T + b^2CC^T) - EaEa^T - bEaEC^T - bECEa^T - b^2ECEC^T$$

$$= E(aa^T) - E(aa^T) + baEC^T - baEC^T + ba^TEC - ba^TEC + b^2ECC^T - b^2ECEC^T$$

$$= b^2(ECC^T - ECEC^T) = b^2 Cov(C)$$

Por lo que las matrices de covarianza se relacionan por $Cov(F) = b^2 Cov(C)$

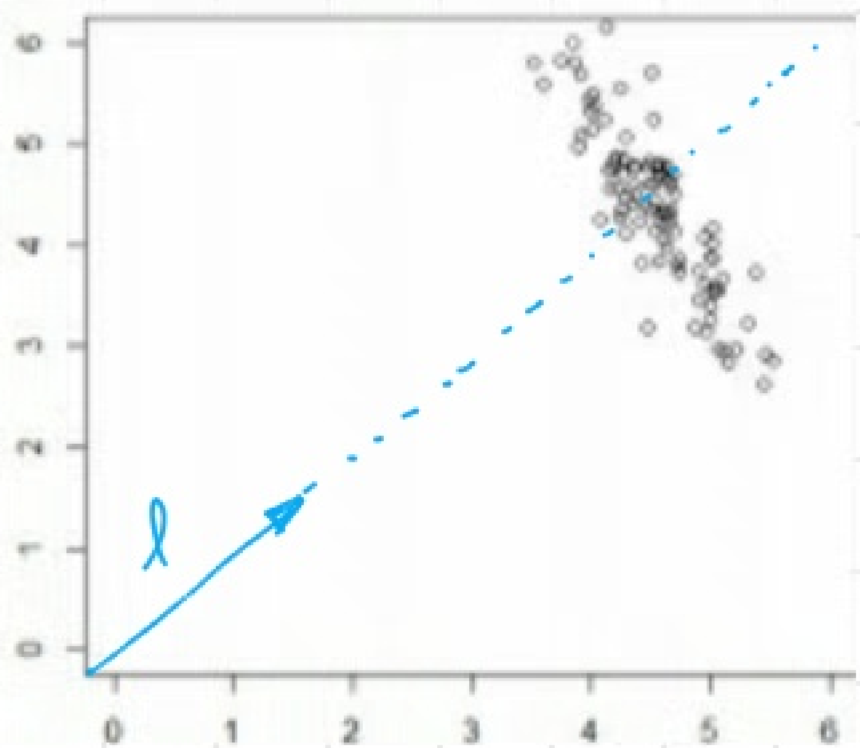
Proyectar en la dirección de máxima varianza implica encontra una dirección l tal que $var(l^tC)$ es máxima, con $var(l^tC) = l^t cov(C) l$.

Para el caso de F, se busca maximizar

$$var(l^tF) = l^t cov(F) l = b^2 l^t cov(C) l.$$

Por lo que al encontrar l para C, este mismo vector maximizará la varianza para F.

2.



La dirección del primer componente será aprox. en la dirección de la media de los datos.

Esto se puede interpretar bajo el objetivo de minimizar $E\|X - \hat{X}\|$, donde X es un punto en el espacio original y $\hat{X}$ su reconstrucción a partir del componente principal l ($\hat{x} = \langle l, x \rangle l$). Cualquier otra dirección distinta de l incrementará la distancia entre X y $\hat{X}$.

3.

$$Y = \langle l, X \rangle = \sum_{j=1}^{d} l_j X_j$$

$$\mathrm{Cov}(Y, X_i) = \mathrm{Cov}\left(\sum_{j=1}^{d} l_j X_j, X_i\right)$$

$$= \sum_{j=1}^{d} \mathrm{Cov}(l_j X_j, X_i)$$

$$= \sum_{j=1}^{d} l_j \mathrm{Cov}(X_j, X_i)$$

Devido a que $\mathrm{Cov}(X, Y) = \mathrm{Cov}(Y, X)$,

$$\sum_{j=1}^{d} l_j \mathrm{Cov}(X_j, X_i) = \sum_{j=1}^{d} l_j \mathrm{Cov}(X_i, X_j)$$

$\mathrm{Cov}(X_i, X_j)$ representa la entrada en el renglón i y columna j de la matriz $\mathrm{Cov}(X)$.

Por lo que $\sum_{j=1}^{d} l_j \mathrm{Cov}(X_i, X_j)$ representa el elemento i-ésimo del vector $\mathrm{Cov}(X)l$ el cual es igual a λl ya que l es vector propio de $\mathrm{Cov}(X)$.

Por lo que $\sum_{j=1}^{d} l_j \mathrm{Cov}(X_i, X_j)$ es la entrada i-ésima del vector λl, es decir, λl_i. Y así, $\mathrm{Cov}(Y, X_i) = \lambda l_i$.

$\square$

Tarea 1

Se incluyen bibliotecas y se leen los datos

```
library(tidyverse)
```

```
## -- Attaching packages -----------------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ---------------------------------------------------------------------------------- ti
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggfortify)

setwd("~/Downloads/")
do <- read.table("hepatlon (1)")
d <- read.table("hepatlon (1)")

# Se filtra columna 'score'

d <- d[, -8]
```

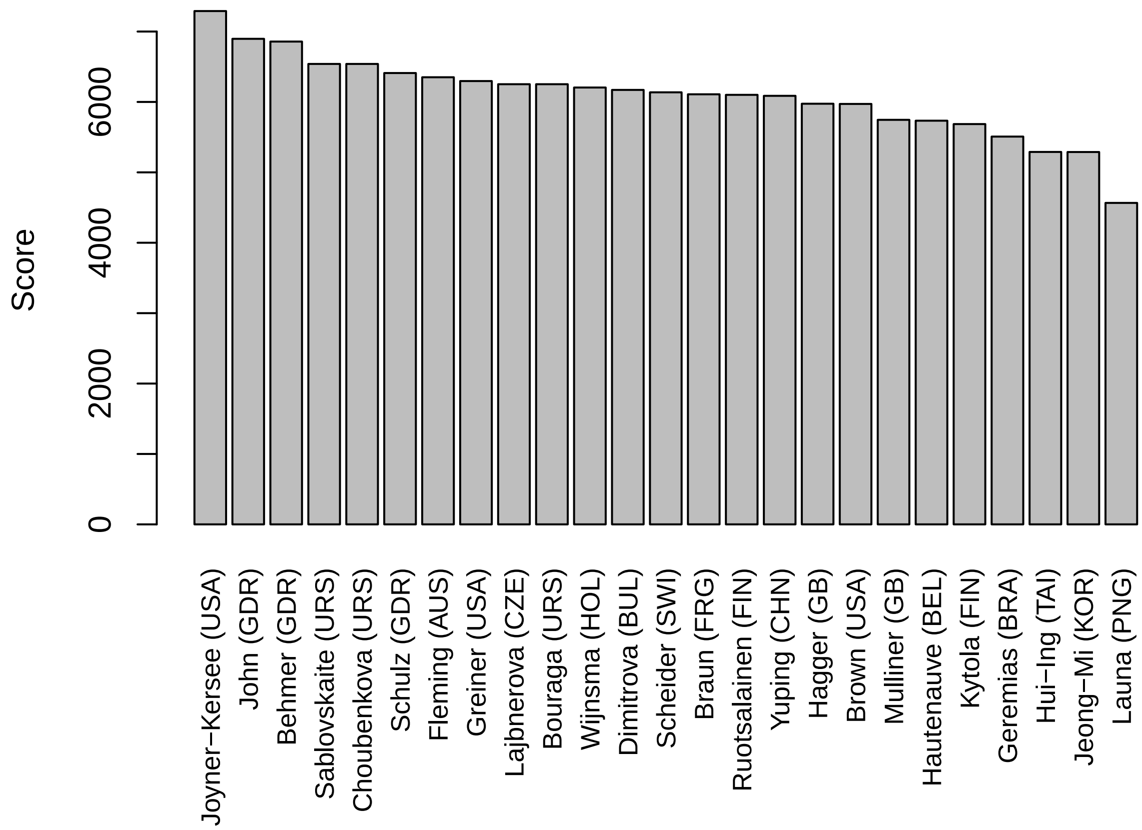
Se calcula la varianza por cada atributo. De donde se observa que hay atributos con mucha variabilidad como en run800m y javelin, y al mismo tiempo, atributos como highjump o longjump con variabilidad muy pequeña. Por lo que será conveniente escalarlos para que tengan varianza 1.

```
sapply(d, var)
```

```
##    hurdles   highjump       shot    run200m   longjump    javelin    run800m
##  0.5426500  0.0060750  2.2257190  0.9400410  0.2248773 12.5716773 68.7421417
```

De la siguiente gráfica de barras, podemos ver que la competidora Joyner-Kersee tiene el mayor puntaje.

```
par(mar=c(8,5,1,1))
barplot(height = do$score, names.arg = row.names(do), horiz = FALSE, las = 3, cex.names = 0.8, ylab = "S
```



Se aplica PCA sobre la matriz de datos (centrados y escalados). Donde se observa que la varianza explicada sólamente con los 2 primeros componentes principales, es del 80.78%. Es decir, el 80.78% de la varianza en los datos se concentra en 2 de las columnas de los datos.

```
prin_comp <- prcomp(d, scale = TRUE, center = TRUE)
summary(prin_comp)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6    PC7
## Standard deviation     2.1119 1.0928 0.72181 0.67614 0.49524 0.27010 0.2214
## Proportion of Variance 0.6372 0.1706 0.07443 0.06531 0.03504 0.01042 0.0070
## Cumulative Proportion  0.6372 0.8078 0.88223 0.94754 0.98258 0.99300 1.0000
```

A continuación se muestra la matriz de rotación o de loadings. Si tomamos el primer componente principal (PC1), y agrupamos las entradas de acuerdo al signo, vemos que se están juntando atributos que miden el desempeño en unidades de distancia, con signo negativo, y atributos que miden desempeño en unidades de tiempo, con signo positivo. Además, dentro de las competencias que miden distancias, la que más importancia tiene es la prueba de longjump o salto de longitud. Mientras que dentro de las pruebas que miden tiempo, la que más importancia tiene es la prueba de hurdles o carrera con obstáculos.

```
prin_comp
```

```
## Standard deviations (1, .., p=7):
## [1] 2.1119364 1.0928497 0.7218131 0.6761411 0.4952441 0.2701029 0.2213617
##
## Rotation (n x k) = (7 x 7):
##                PC1         PC2         PC3         PC4         PC5         PC6
## hurdles   0.4528710 -0.15792058 -0.04514996  0.02653873 -0.09494792 -0.78334101
## highjump -0.3771992  0.24807386  0.36777902 -0.67999172 -0.01879888 -0.09939981
## shot     -0.3630725 -0.28940743 -0.67618919 -0.12431725 -0.51165201  0.05085983
```

```
## run200m   0.4078950  0.26038545  0.08359211 -0.36106580 -0.64983404  0.02495639
## longjump -0.4562318  0.05587394 -0.13931653 -0.11129249  0.18429810 -0.59020972
## javelin  -0.0754090 -0.84169212  0.47156016 -0.12079924 -0.13510669  0.02724076
## run800m   0.3749594 -0.22448984 -0.39585671 -0.60341130  0.50432116  0.15555520
##                  PC7
## hurdles  -0.38024707
## highjump -0.43393114
## shot     -0.21762491
## run200m   0.45338483
## longjump  0.61206388
## javelin   0.17294667
## run800m   0.09830963
```

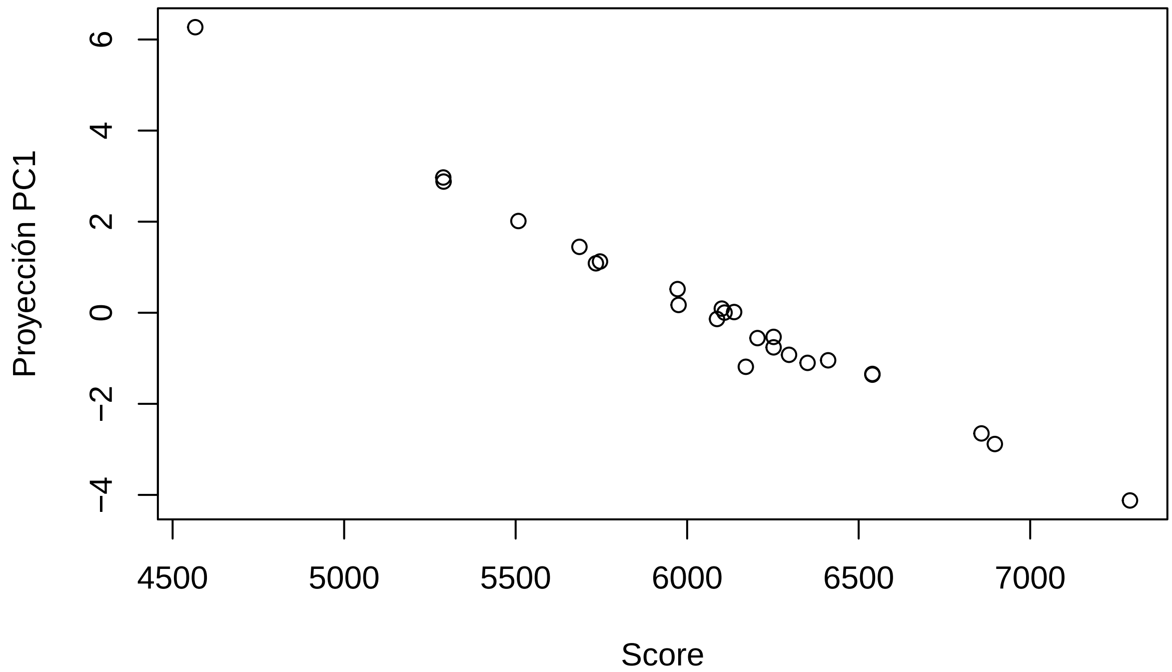
Mediante la función predict() podemos proyectar los datos en los componentes principales. Consideremos la proyección de cada uno de los datos sobre el primer componente principal (PC1).

```
predict(prin_comp)[,1]
```

```
## Joyner-Kersee (USA)          John (GDR)        Behmer (GDR)  Sablovskaite (URS)
##        -4.121447626        -2.882185935        -2.649633766        -1.343351210
##   Choubenkova (URS)        Schulz (GDR)       Fleming (AUS)       Greiner (USA)
##        -1.359025696        -1.043847471        -1.100385639        -0.923173639
##    Lajbnerova (CZE)       Bouraga (URS)       Wijnsma (HOL)     Dimitrova (BUL)
##        -0.530250689        -0.759819024        -0.556268302        -1.186453832
##      Scheider (SWI)         Braun (FRG)  Ruotsalainen (FIN)        Yuping (CHN)
##         0.015461226         0.003774223         0.090747709        -0.137225440
##         Hagger (GB)         Brown (USA)        Mulliner (GB)    Hautenauve (BEL)
##         0.171128651         0.519252646         1.125481833         1.085697646
##        Kytola (FIN)      Geremias (BRA)       Hui-Ing (TAI)       Jeong-Mi (KOR)
##         1.447055499         2.014029620         2.880298635         2.970118607
##         Launa (PNG)
##         6.270021972
```

Si visualizamos una comparación de los datos proyectados en la dirección de PC1 y el score dado en el dataset original para cada competidor, vemos que están altamente correlacionados. Donde la competidora Joyner-Kersee, quien tiene el mayor puntaje, resulta con el valor más pequeño en la proyección sobre el PC1. Por lo que el valor de PC1 de cada dato podría servir como un buen predictor del puntaje o desempeño en el heptatlón de cada competidora.

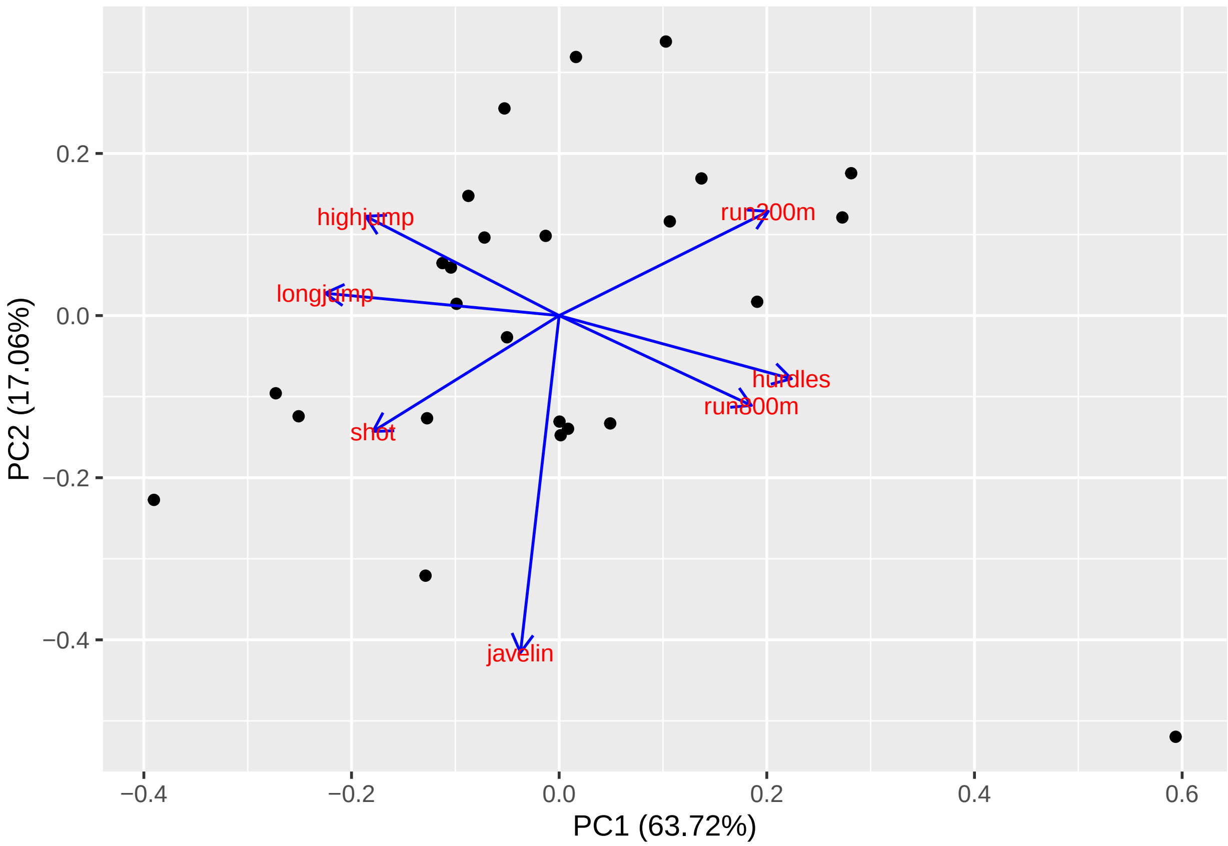
```
plot(x = do$score, y = predict(prin_comp)[,1], xlab = "Score", ylab = "Proyección PC1")
```



A continuación se grafican un biplot, la cual es una superposición de 2 gráficas: una gráfica de loadings, donde se puede visualizar la influencia de cada atributo sobre los primeros 2 componentes principales, y cada uno de los datos del dataframe proyectados en el espacio generado por los componentes principales 1 y 2.

De la gráfica de loadings, también se puede ver que las pruebas donde se mide el desempeño en términos de distancia influye significativamente en el PC1 en la dirección negativa. Mientra que las pruebas de carrera influyen en el PC1 en la dirección contraria, formando así, dos grupos de pruebas que se pueden ponderar.

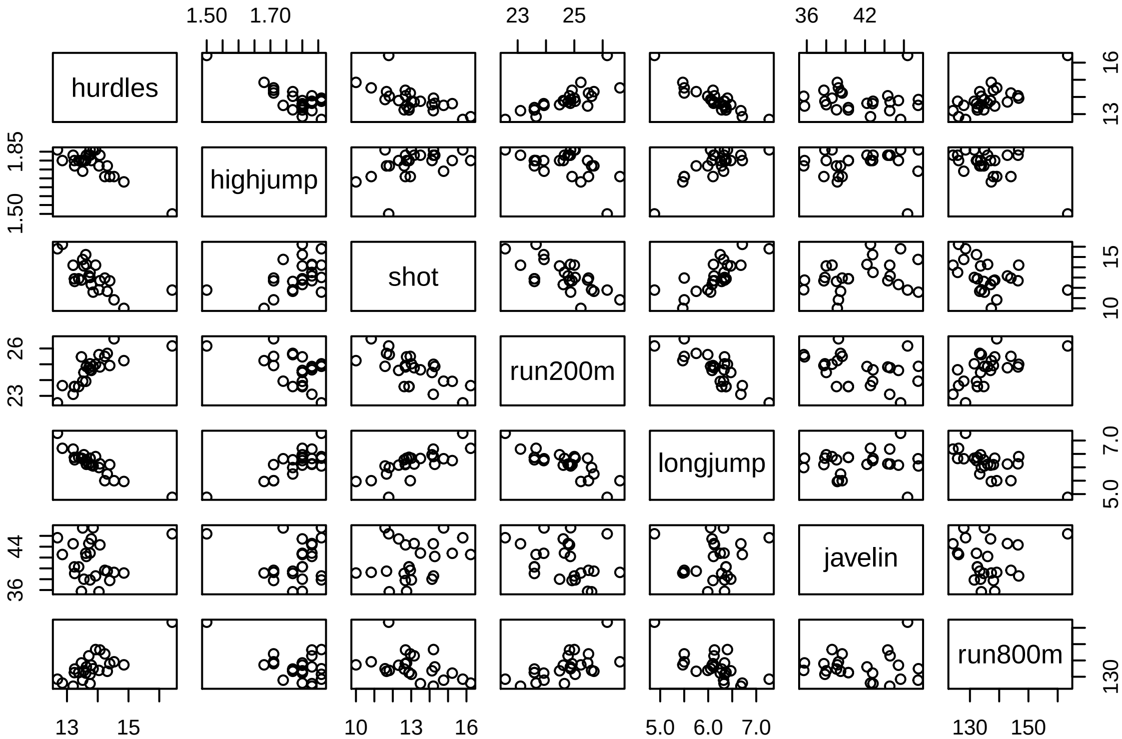
```
autoplot(prin_comp, data = d, loadings = TRUE, loadings.colour = 'blue', loadings.label = TRUE, loadings
```



La forma de ponderar las pruebas de acuerdo al componente principal 1 tiene sentido ya que para las pruebas highjump, longjump y shot mientras más grande sea este valor mejor será el score, mientras que para las pruebas de tiempo, entre menor sea su valor, mejor será el desempeño en dicha prueba. Esta relación inversa es capturada por el signo de las entradas del PC1, haciendo que la competidora con el mejor desempeño, tenga la proyección sobre PC1 más negativa.

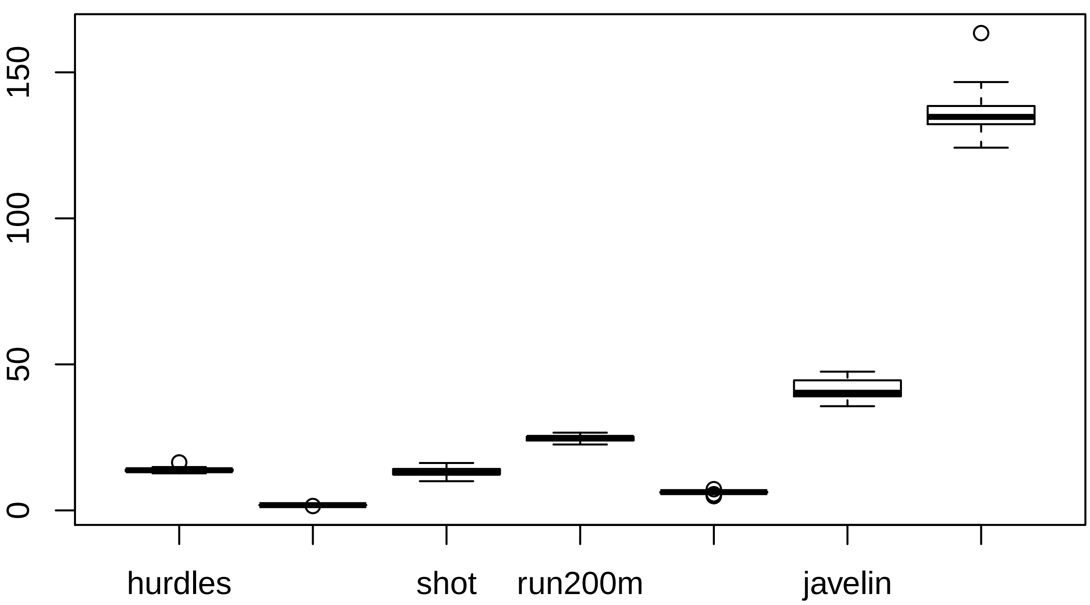
También cabe notar que la prueba javelin o lanzamiento de javalina aunque se inclina del lado de las pruebas de distancia, parece casi no estar correlacionada a ninguna prueba. Esto se puede visualizar con un scatterplot.

```
plot(d)
```



Finalmente, se puede notar que hay un valor atípico, cuyo valor en PC1 es cercano a 0.6. Dicho dato corresponde a la competidora Launa. Visualizando un boxplot por prueba, vemos que hay algunos valores atípicos, lo cual se nota más para la prueba de carrera de 800 metros (run800m)

```
boxplot(d)
```



Obteniendo los puntajes máximos y mínimos para cada prueba y comparándolo con la competidora Launa, vemos que para las pruebas hurdles y run800m tiene los tiempos más altos. Mientras que en las pruebas de distancia highjump y longjump tiene los puntajes más bajos. Esto nos ayuda a corroborar que este dato es atípico y tendría que hacerse otro análisis para ver que tanto puede afectar dicho dato para aplicar PCA.

```r
sapply(d, max)
```

```
##  hurdles highjump     shot  run200m longjump  javelin  run800m
##    16.42     1.86    16.23    26.61     7.27    47.50   163.43
```

```r
sapply(d, min)
```

```
##  hurdles highjump     shot  run200m longjump  javelin  run800m
##    12.69     1.50    10.00    22.56     4.88    35.68   124.20
```

```r
d[25,]
```

```
##             hurdles highjump  shot run200m longjump javelin run800m
## Launa (PNG)   16.42      1.5 11.78   26.16     4.88   46.38  163.43
```