Principal Components Analysis (PCA)

Gaston Sanchez

EDSS - SFSU

About

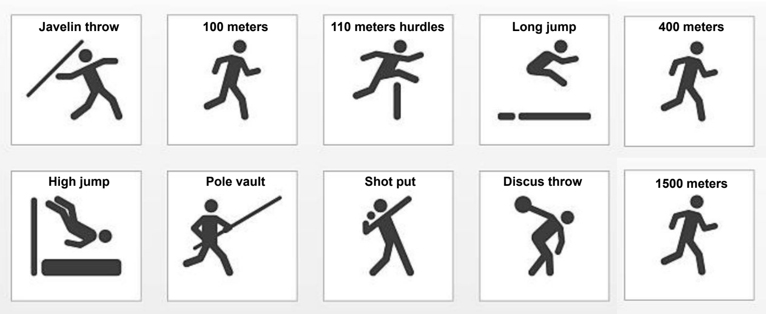
In this slides, we provide a high-level intuition of Principal Components Analysis (PCA).

The idea is to get a feeling of what the notion of low-dimensional representation of a data set means.

Packages

```r
# mandatory package
library(FactoMineR)

# recommended packages
library(dplyr)
library(reshape2)
library(ggplot2)
```

Decathlon Events



Dataset Decathlon

- `decathlon` data, from R package `"FactoMineR"`
- 41 athletes, 13 variables containing 10 events
  - `100m`
  - `Long.jump`
  - `Shot.put`
  - `High.jump`
  - `400m`
  - `110m.hurdle`
  - `Discus`
  - `Pole.vault`
  - `Javeline`
  - `1500m`
- involving 2 competitions (2004 Olympic Game or 2004 Decastar)

```r
data(decathlon)

# decathlon events (ignore 3 last columns)
dat <- decathlon[ ,1:10]
```

```
         100m  Long.jump  Shot.put  High.jump   400m
SEBRLE  11.04       7.58     14.83       2.07  49.81
CLAY    10.76       7.40     14.26       1.86  49.37
KARPOV  11.02       7.30     14.77       2.04  48.37
BERNARD 11.02       7.23     14.25       1.92  48.93
YURKOV  11.34       7.09     15.19       2.10  50.42

        110m.hurdle  Discus  Pole.vault  Javeline  1500m
SEBRLE        14.69   43.75        5.02     63.19  291.7
CLAY          14.05   50.72        4.92     60.15  301.5
KARPOV        14.09   48.95        4.92     50.31  300.2
BERNARD       14.99   40.87        5.32     62.77  280.1
YURKOV        15.31   46.26        4.72     63.44  276.4
```

Exploratory Data Analysis (EDA)

We can explore variables at different stages:

- Univariate: one variable at a time
- Bivariate: two variables simultaneously
- Multivariate: multiple variables

Multivariate EDA: Objects and Variables Perspectives

Data Perspectives

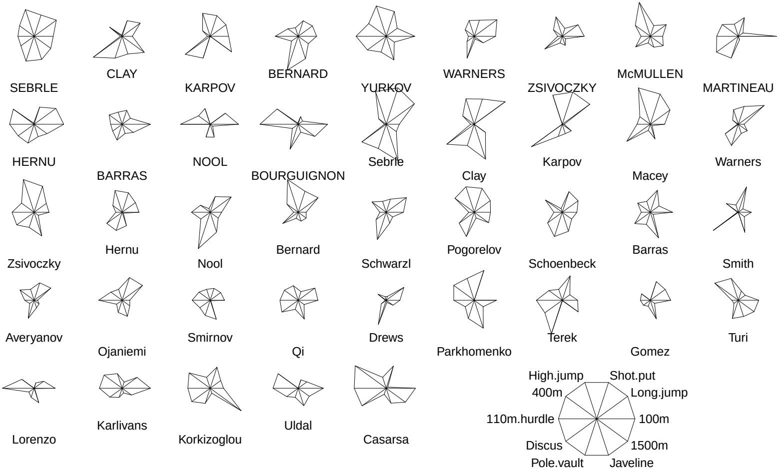
From a multivariate point of view, we are interested in analyzing a data set from both perspectives: **objects** and **variables**

2 Overall Goals

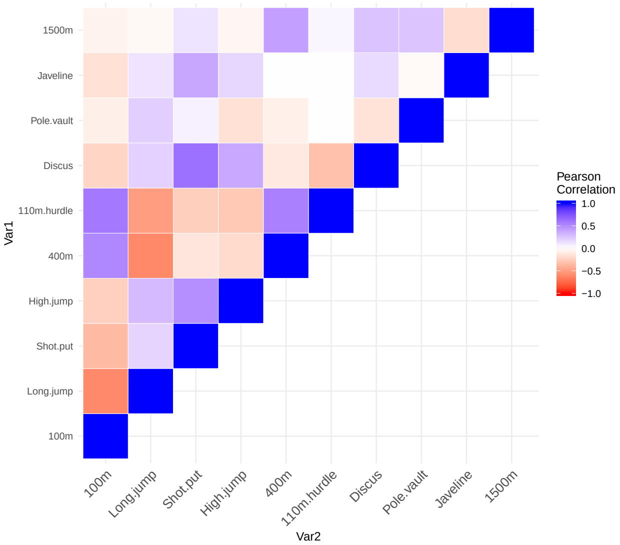
At its simplest we are interested in 2 fundamental purposes:

- Study **resemblance among individuals**: resemblance among athletes
- Study **relationship among variables**: relationship among events statistics

Stars or Glyphs plot



Correlation heatmap



100m
Long.jump
Shot.put
High.jump
400m
110m.hurdle
Discus
Pole.vault
Javeline
1500m

R Code for previous graphics

```r
# stars plot for looking at individuals
stars(dat, nrow = 5, key.loc = c(17,1.5))
```

```r
# correlation heatmap
cormat <- cor(dat)
cormat[upper.tri(cormat)] <- NA
cormat_melt <- melt(cormat, na.rm = TRUE)
ggplot(data = cormat_melt, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "red", high = "blue", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()
```

```r
# scatterplot matrix
pairs(dat, pch = 19, col = "#50505080")
```

EDA so far . . .

With the three previous graphics, we can get an idea of what's going on with certain (dis)similarities between the individuals, as well as certain relationships among the variables.

But none of these plots provide a larger "panoramic" view of the data.

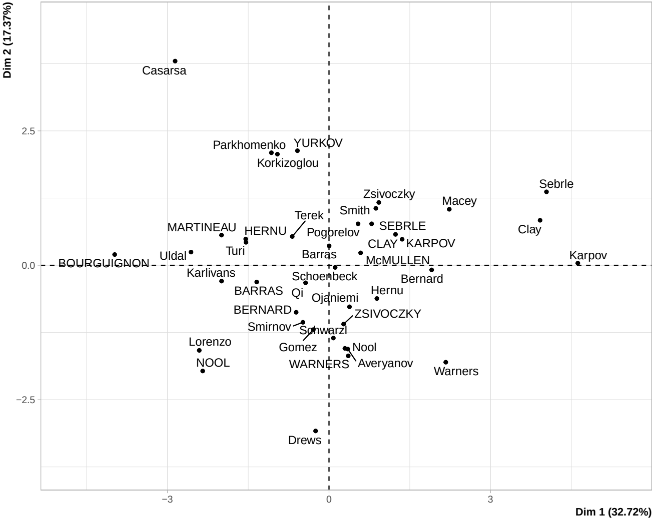
Also, keep in mind that a stars plot is good for small data sets, but it doesn't scale well with a large number of individuals.

Likewise, each of the scatterplots (and their associated correlations) gives an isolated 2-dimensional picture. Although together they seem to provide a rich view of the data, it is a highly compartmentalized view.

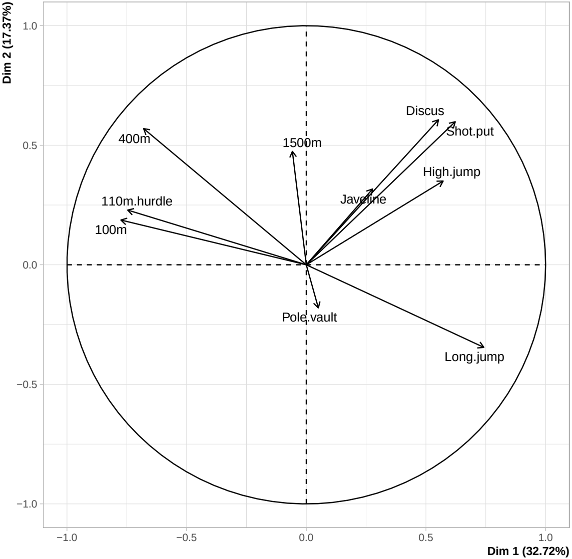
What if we could get a better low-dimensional summary of the data?

- e.g. a more informative scatterplot
- e.g. a comprehensive view of relationships among variables

What if we could get a more informative scatterplot?



Or a “radar” view of the variables?



About PCA

Principal Components Analysis (PCA) is a multivariate method that allows us **to study and explore** a set of quantitative variables measured on some objects.

Core Idea

With PCA we seek to **reduce the dimensionality** (condense information in variables) of a data set while retaining as much as possible of the variation present in the data

PCA: Overall Goals

- Summarize a data set with the help of a small number of synthetic variables (i.e. the Principal Components).
- Visualize the position (resemblance) of individuals.
- Visualize how variables are correlated.
- Interpret the synthetic variables.

Common PCA applications

- Dimension Reduction
- Visualization
- Feature Extraction
- Data Compression
- Smoothing of Data
- Detection of Outliers
- Preliminary process for further analyses

Geometric Mindset

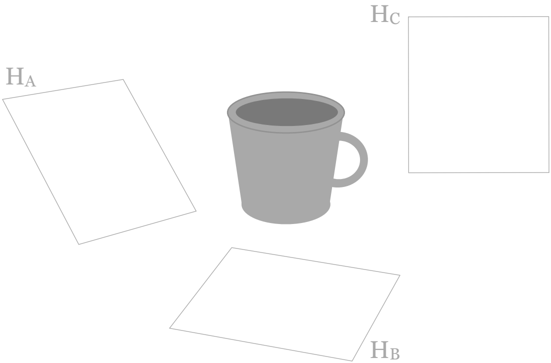
One way to present PCA is based on a data visualization approach.

To help you understand the main idea of PCA from a geometric standpoint, I'd like to begin showing you my **mug-data** example.

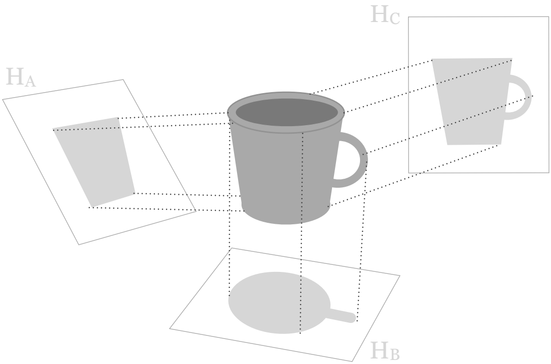
Imagine a data set in a "high-dimensional space"

Figure 1: Cloud of points in the form of a mug

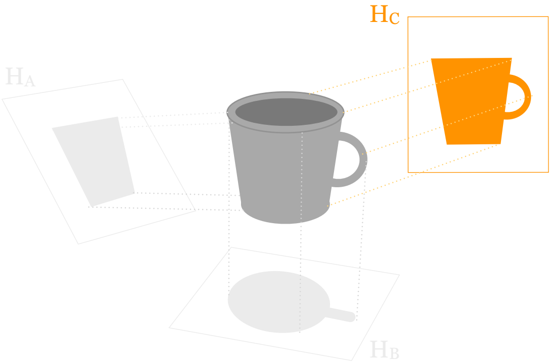
We are looking for Candidate Subspaces



with the best low-dimensional representation



Best low-dimensional projection



Geometric Idea

Looking at the cloud of points

Under a purely geometric approach, PCA aims to represent the cloud of points into a space with reduced dimensionality (usually 2-dimensions) in an "optimal" way.
By "optimal" we mean obtaining a low-dimensional representation of the data as less distorted as possible from the its original configuration.

What PCA is doing?

A PC is obtained by combining the input X-variables in a way that we maximize the “information” captured by the PC

$$PC_k = v_{1,k}X_1 + v_{2,k}X_2 + \cdots + v_{p,k}X_p$$

such that $\max\{Var(PC_k)\}$

- Think of a PC as a **weighted sum** of the input X-variables.
- Each PC captures a **unique amount of information** or variation about X-variables
- PC_1 captures the largest amount of variation
- PC_2 captures the second largets amount of variation
- and so on

PCA in Practice

Considerations

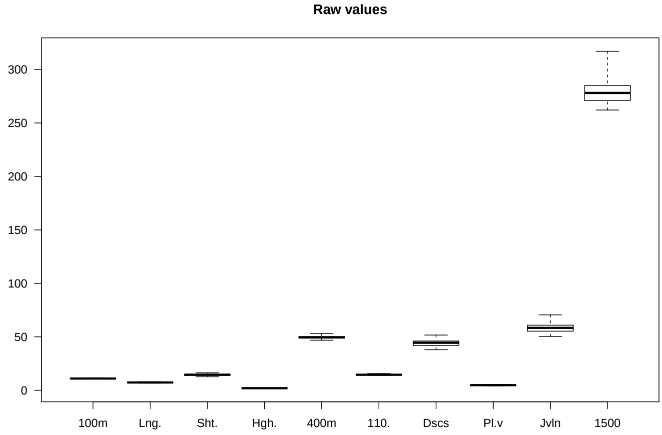
- PCA applies to a data table of quantitative (real-valued) variables
- Decide if variables need to be normalized to a comparable scale
- I will show you how to carry out PCA with the function `PCA()` from the package `"FactoMineR"`

To standardize or not?

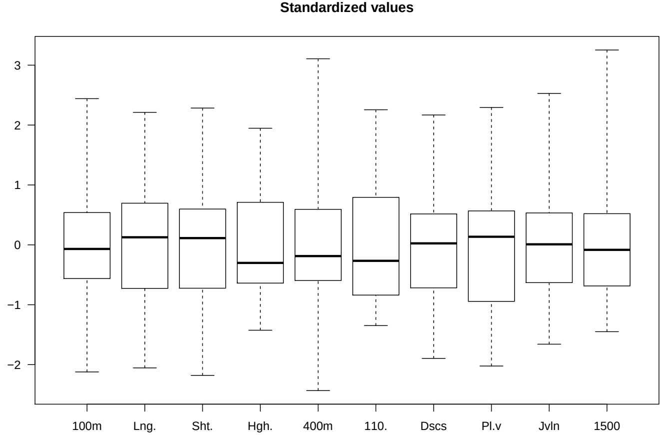
- A key issue has to do with the scale of the variables.
- If variables have different units of measurement, then we should standardize them to avoid variables with larger scales dominate the analysis.
- If variables have the same units:
  - you could leave them unstandardized
  - or you could standardize them (strongly suggested)

Regardless of the scaling decision, we operate on "mean-centered data", that is, variables that have zero mean.

If you use the raw scales, the variable `1500m` will dominate the analysis due to its larger scale.



By normalizing the variables (dividing by their standar deviations), they all play the same role, and have comparable scales.



PCA with "FactoMineR"

```r
# PCA() from FactoMineR
pca <- PCA(dat)
```

- the main input for `PCA()` is a data table (e.g. `matrix` or `data.frame`)
- by default, `PCA()` standardizes all variables (zero mean, unit variance)

Output of PCA()

```
names(pca)
```

```
## [1] "eig"  "var"  "ind"  "svd"  "call"
```

`PCA()` produces an object of class `"PCA"` which is a list that contains:

- `eig`: table of *eigenvalues* containing the variances of the PCs
- `var`: list of outputs for the variables
- `ind`: list of outputs for the individuals (e.g. PCs)
- `svd`: results from *singular value decomposition*

PCA Essetial Results

The core results of a PCA consists of:

- Principal Components (PCs) or Scores: new coordinates for the individuals; these are available in `pca$ind$coord`
- Variance of PCs (how much variation each PC captures); these are available in `pca$eig`
- Loadings: how much each variable *weighs* on the formation of the PCs; these are available in `pca$svd$V`

How many PCs to retain

How many PCs to retain?

There are various ways to determine the number of PCs to be retained. The most common ones are:

- Screeplot (see if there's an "elbow")
- Predetermined amount of variation
- Kaiser's rule

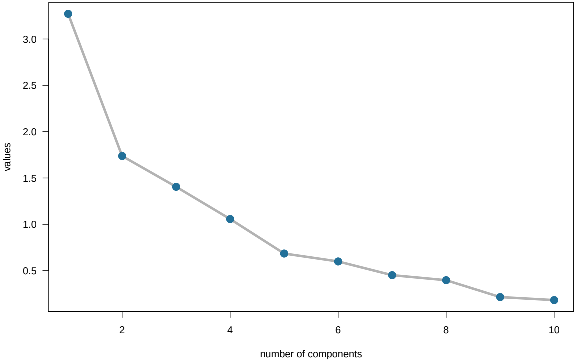
Table of Eigenvalues

```
# eigenvalues
pca$eig
```

```
        eigenvalue percentage of variance cumulative percentage of variance
comp 1   3.2719055              32.719055                          32.71906
comp 2   1.7371310              17.371310                          50.09037
comp 3   1.4049167              14.049167                          64.13953
comp 4   1.0568504              10.568504                          74.70804
comp 5   0.6847735               6.847735                          81.55577
comp 6   0.5992687               5.992687                          87.54846
comp 7   0.4512353               4.512353                          92.06081
comp 8   0.3968766               3.968766                          96.02958
comp 9   0.2148149               2.148149                          98.17773
comp 10  0.1822275               1.822275                         100.00000
```

Screeplot: look for an "elbow"

```
eigs <- pca$eig
plot(1:nrow(eigs), eigs[,1], las = 1, type = "n",
     ylab = "values", xlab = "number of components")
lines(1:nrow(eigs), eigs[,1], lwd = 4, col = "gray70")
points(1:nrow(eigs), eigs[,1], pch = 20, cex = 2.5, col = "#227099")
```



Predetermined amount of variation

One option to decide how many PCs to retain, consists of predefining a specified portion of variation: e.g. 70%

```
# 70% or more
print(round(eigs[eigs[ ,3] <= 80, ], 4))
```

```
       eigenvalue percentage of variance cumulative percentage of variance
comp 1     3.2719                32.7191                            32.7191
comp 2     1.7371                17.3713                            50.0904
comp 3     1.4049                14.0492                            64.1395
comp 4     1.0569                10.5685                            74.7080
```

Kaiser's Rule

Another criterion to decide how many PCs to keep, is the so-called Kaiser's rule, which consists of retaining those PCs with eigenvalues $\lambda_k > 1$

```
# Kaiser criterion
eigs[eigs[ ,1] > 1, ]
```

```
       eigenvalue percentage of variance cumulative percentage of variance
comp 1   3.271906               32.71906                          32.71906
comp 2   1.737131               17.37131                          50.09037
comp 3   1.404917               14.04917                          64.13953
comp 4   1.056850               10.56850                          74.70804
```

Studying the Individuals

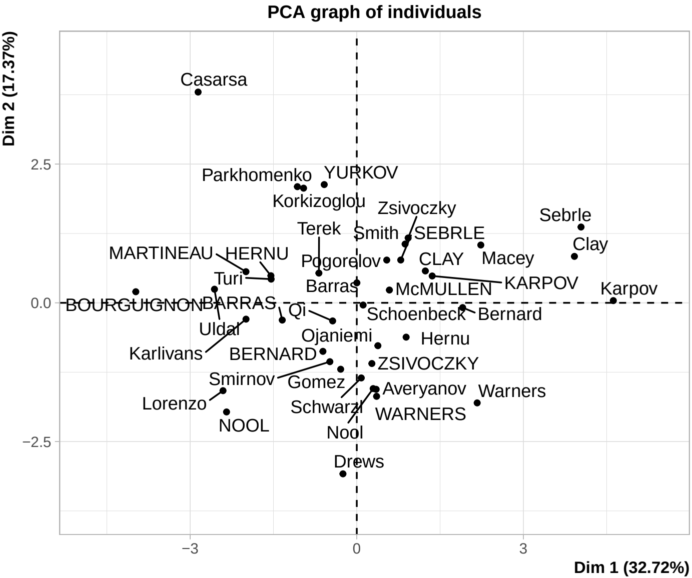
Studying the Individuals

When studying the individuals, we typically look at scatterplots of PCs

```r
# scatterplot PC1 -vs- PC2
plot(pca, choix = 'ind', axes = c(1, 2))
```

Optionally, we could also look at:

- Quality of representation: `pca$ind$cos2`
- Individual Contributions to PCs: `pca$ind$contrib`


PCA graph of individuals
Dim 2 (17.37%)
Dim 1 (32.72%)
2.5
0.0
−2.5
−3
0
3
Casarsa
Parkhomenko
YURKOV
Korkizoglou
Zsivoczky
Sebrle
Terek
Smith
SEBRLE
Clay
MARTINEAU
HERNU
Pogorelov
CLAY
Macey
Turi
Barras
KARPOV
McMULLEN
Karpov
BOURGUIGNON
BARRAS
Qi
Schoenbeck
Bernard
Uldal
Ojaniemi
Hernu
Karlivans
BERNARD
ZSIVOCZKY
Smirnov
Gomez
Averyanov
Warners
Lorenzo
Schwarzl
WARNERS
NOOL
Nool
Drews

Quality of Representation

First 5 rows of $cos^2(i, PC_k)$ for $k = 1, 2, 3, 4$

```r
# quality of positioning
print(head(pca$ind$cos2, n = 5), digits = 3)
```

```
         Dim.1  Dim.2 Dim.3  Dim.4  Dim.5   Dim.6    Dim.7    Dim.8   D
SEBRLE  0.1117 0.1061 0.122 0.2459 0.0891 0.18929 0.054207 0.033823 0.0
CLAY    0.1240 0.0268 0.373 0.0102 0.3170 0.03872 0.040753 0.029605 0.0
KARPOV  0.1599 0.0203 0.332 0.2988 0.0548 0.04655 0.003127 0.005431 0.0
BERNARD 0.0487 0.1002 0.104 0.6461 0.0171 0.00995 0.000322 0.000596 0.0
YURKOV  0.0377 0.4986 0.165 0.0838 0.1719 0.00120 0.036166 0.000983 0.0
         Dim.10
SEBRLE  0.04469
CLAY    0.00576
KARPOV  0.02374
BERNARD 0.00465
YURKOV  0.00035
```

Quality of Representation

Adding the squared cosines over all principal axes for a given individual, we get:

$$\sum_{PC_k} cos^2(i, PC_k) = 1$$

This sum provides, in percentages, the “quality” of the representation of an individual on the subspace defined by the principal axes.

```
# sum of squred-cosines for 1st athlete
sum(pca$ind$cos2[1, ])
```

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

Quality of Representation

The squared cosine is used to evaluate the quality of the representation.

On a given PC, some distances between individuals will be well represented, while other distances will be highly distorted.

You can add the squared cosines of an individual over different axes, resulting in a quality measure of how well that individual is represented in that subspace.

Study of cloud of Variables

Studying the Variables

When studying the variables, we typically pay attention to:

- Scatterplots of loadings (or some loading-based results)
- Quality of representation of variables
- Variables Contributions to PCs

Loadings

```r
# first 4 vectors of loadings (associated to first 4 PCs)
print(pca$svd$V[, 1:4], digits = 3)
```

```
         [,1]   [,2]    [,3]    [,4]
 [1,] -0.4283  0.142 -0.1556 -0.0368
 [2,]  0.4102 -0.262  0.1537  0.0990
 [3,]  0.3441  0.454 -0.0197  0.1854
 [4,]  0.3162  0.266 -0.2189 -0.1319
 [5,] -0.3757  0.432  0.1109  0.0285
 [6,] -0.4126  0.174 -0.0782  0.2829
 [7,]  0.3054  0.460  0.0362 -0.2526
 [8,]  0.0278 -0.137  0.5836  0.5365
 [9,]  0.1532  0.241 -0.3287  0.6929
[10,] -0.0321  0.360  0.6599 -0.1567
```

The entries of a loadings-vector are the coefficients that produce a PC as a weighted sum; for example PC_1 is given by:

$$
\begin{aligned}
PC_1 = {} & (-0.4283)\texttt{100m} + (0.4102)\texttt{Long.jump} \\
& + (0.3441)\texttt{Shot.jump} + (0.3162)\texttt{High.jump} \\
& + (-0.3757)\texttt{400m} + (-0.4126)\texttt{110m.hurdle} \\
& + (0.3054)\texttt{Discus} + (0.0278)\texttt{Pole.vault} \\
& + (0.1532)\texttt{Javeline} + (-0.0321)\texttt{1500m}
\end{aligned}
$$

Interpreting PCs

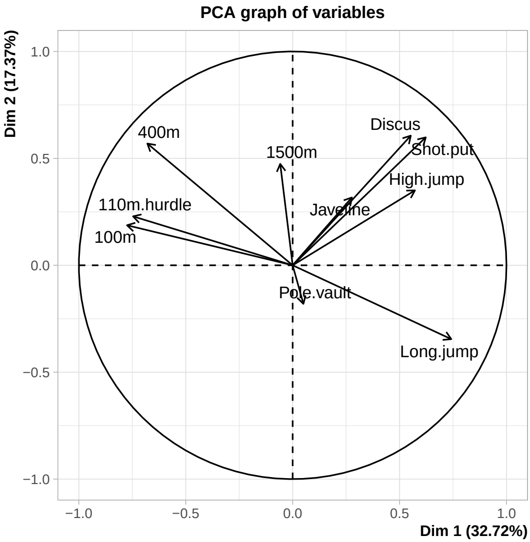
- Because PCs are obtained by combinaning the input variables, we typically try to give PCs a meaningful interpretation
- This interpretation can be very useful, but not always possible
- You can look at the magnitude of the loadings
- You can also look at the correlations between variables and PCs

```r
# correlations between X-variables and PCs
print(pca$var$coord[ ,1:4], digits = 4)
```

```
               Dim.1   Dim.2    Dim.3    Dim.4
100m        -0.77472  0.1871 -0.18441 -0.03782
Long.jump    0.74190 -0.3454  0.18221  0.10179
Shot.put     0.62250  0.5983 -0.02338  0.19059
High.jump    0.57195  0.3503 -0.25951 -0.13559
400m        -0.67961  0.5694  0.13147  0.02930
110m.hurdle -0.74625  0.2288 -0.09264  0.29083
Discus       0.55247  0.6063  0.04295 -0.25967
Pole.vault   0.05034 -0.1804  0.69176  0.55153
Javeline     0.27711  0.3170 -0.38966  0.71228
1500m       -0.05808  0.4742  0.78214 -0.16109
```

A more informative interpretation can be obtained by calculating the correlations between the Variables and PCs, and use them to plot a **Circle of Correlations**

```r
# circle of correlations
plot(pca, choix = "var", axes = c(1, 2))
```



Squared Correlations

- The correlation between a component and a variable estimates the information they share.
- Note that the sum of the squared coefficients of correlation between a variable and all the components is equal to 1.
- As a consequence, the squared correlations are easier to interpret than the loadings.
- This is because the squared correlations give the proportion of the variance of the variables explained by the components.

```r
# squared correlations
print(pca$var$cos2[ ,1:5], digits = 4)
```

```
              Dim.1   Dim.2     Dim.3     Dim.4    Dim.5
100m       0.600191 0.03502 0.0340060 0.0014302 0.091323
Long.jump  0.550415 0.11932 0.0332009 0.0103603 0.001345
Shot.put   0.387509 0.35797 0.0005466 0.0363252 0.012355
High.jump  0.327121 0.12271 0.0673464 0.0183858 0.308513
400m       0.461870 0.32426 0.0172843 0.0008586 0.007690
110m.hurdle 0.556882 0.05235 0.0085817 0.0845827 0.027001
Discus     0.305219 0.36762 0.0018449 0.0674293 0.010989
Pole.vault 0.002534 0.03253 0.4785273 0.3041897 0.108873
Javeline   0.076790 0.10048 0.1518313 0.5073389 0.093104
1500m      0.003373 0.22489 0.6117474 0.0259497 0.023581
```

Clustering

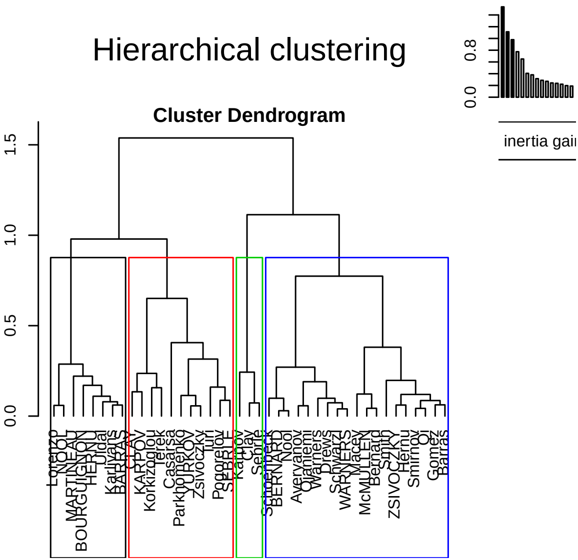
Often, it is interesting to use the output of a PCA, and take a further step by performing clustering analysis.

The most common type of clustering is hierarchical clustering. This can be easily performed with the `HCPC()` function from `"FactoMineR"`

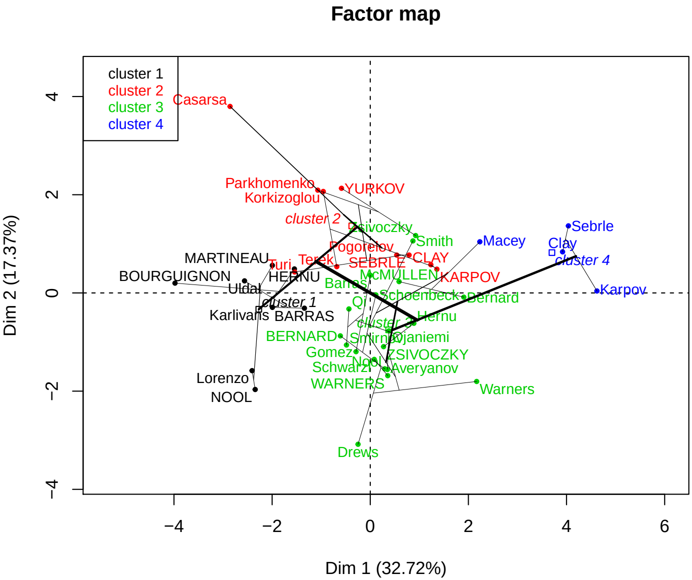
```
# looking for 4 clusters
clustering <- HCPC(pca, nb.clust = 4, graph = FALSE)

names(clustering)
```

```
## [1] "data.clust" "desc.var"   "desc.axes"  "call"       "desc.ind"
```

```r
# Dendrogram (Hierarchical Clustering)
plot(clustering, choice = "tree")
```



```r
# PCA plot with clusters
plot(clustering, choice = "map")
```



```r
# PCA 3D-plot with clusters
plot(clustering, choice = "3D.map")
```