Statistics for the Sciences - Part 3

Instructor: Demetrio Labate

November 23, 2024

Covariance and correlation matrices

Covariance and Correlation

The **covariance** of two random variables X and Y is

$$\sigma_{XY} = E[(X - \mu_X)(Y - \mu_Y)]$$

and their **correlation coefficient** is

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

Covariance and Correlation - sample

Given a sample consisting of pairs $\{(x_i, y_i) : i = 1, \ldots, n\}$, the **Pearson's correlation coefficient** is

$$r_{xy} = \frac{s_{xy}}{s_x \, s_y}$$

where

$$\begin{aligned} s_{xy} &= \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ s_x^2 &= \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \\ s_y^2 &= \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 \end{aligned}$$

Covariance and Correlation

For random variables X and Y, ρ_{XY} measures **linear dependence** between X and Y.
ρ_{XY} is bouded:

$$-1 \leq \rho_{XY} \leq 1$$

- $\rho_{XY} = -1$ implies perfect negative linear relationship
- $\rho_{XY} = 1$ implies perfect positive linear relationship
- $\rho_{XY} = 0$ implies no linear relationship

NOTE: Correlation does not implies causation, not independence ($\rho_{XY} = 0$ does not imply that X and Y are independent).

Covariance and Correlation

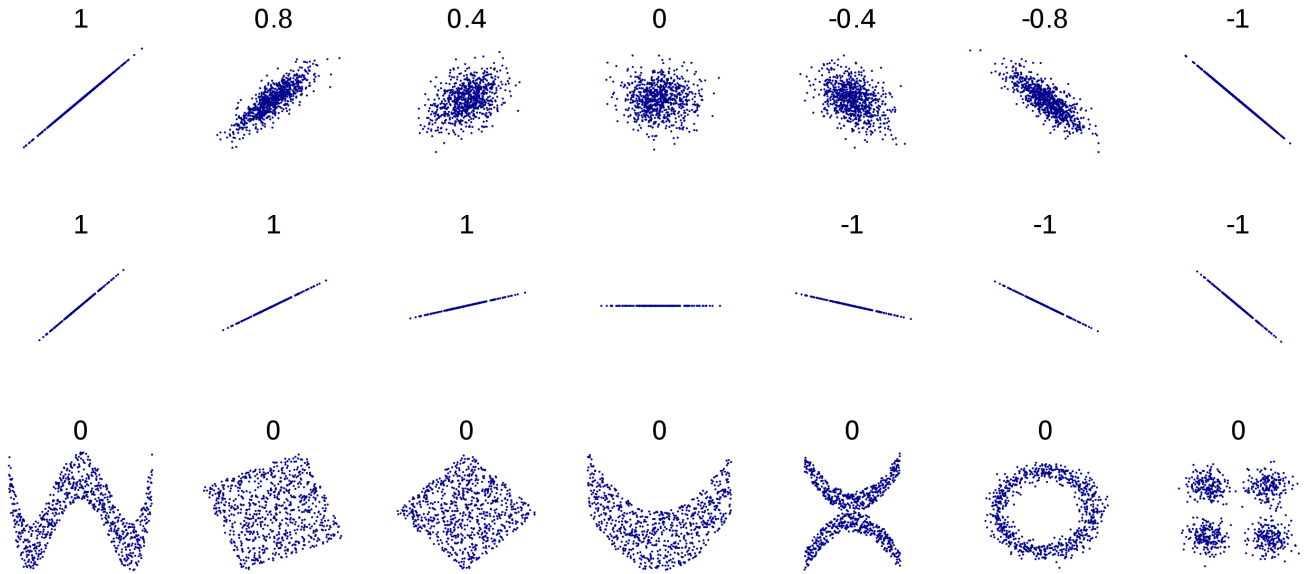
Property: ρ_{XY} is unaffected by linear transformations.

Suppose that $X' = aX + b$ and $Y' = cY + d$. Then

- $\rho_{X'Y'} = \rho_{XY}$ if $\text{sgn}(a) = \text{sgn}(c)$
- $\rho_{X'Y'} = -\rho_{XY}$ if $\text{sgn}(a) \neq \text{sgn}(c)$

This property can be proved directly using the definition of correlation.

Covariance and Correlation - sample



Several sets of points with the corresponding Pearson correlation coefficient r_{xy}.

- r_{xy} reflects noisiness and direction of linear relationship (top row)
- it does not measure the slope of the relationship (middle row)
- it does not capture many aspects of nonlinear relationships (bottom row)

Correlation - example

Example. The following survey reports several measurements collected by 5 instructors for students in their classes related to their nutrition education program. We want to explore the relationship between Sodium and Calories.

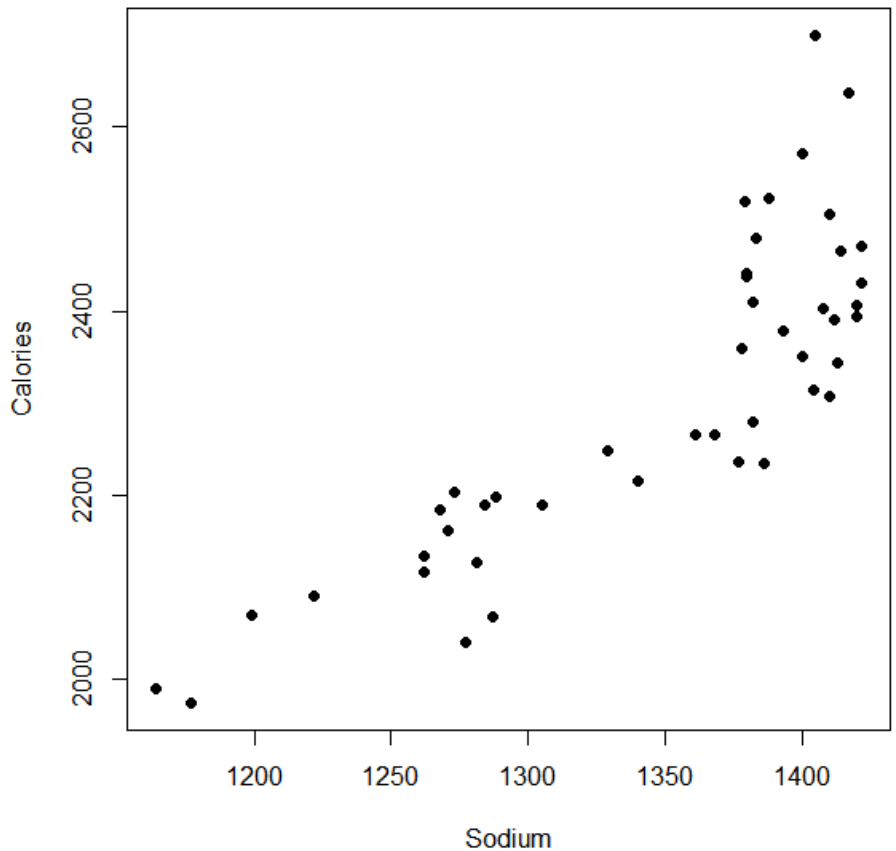
```
Data <- read.csv("C:/Users/student_survey.csv")
> head(Data)
```

	Instructor	*Grade*	*Weight*	*Calories*	*Sodium*	*Score*
1	*BrendonSmall*	6	43	2069	1287	77
2	*BrendonSmall*	6	41	1990	1164	76
3	*BrendonSmall*	6	40	1975	1177	76
4	*BrendonSmall*	6	44	2116	1262	84
5	*BrendonSmall*	6	45	2161	1271	86
6	*BrendonSmall*	6	44	2091	1222	87
...	...	...	...	...	...	...

Correlation - example

Here is the plot of the data showing the relationship between Calories (y) and Sodium (x).

```
> plot(Calories ~ Sodium,data=Data,pch=16,ylab =
"Calories", xlab = "Sodium")
```



Correlation - example

Compute correlation in R

Correlation coefficient can be computed using the functions `cor()` or `cor.test()`

- `cor()` computes the correlation coefficient
- `cor.test()` tests for association/correlation between paired samples. It returns both the correlation coefficient and the significance level (or p-value) of the correlation .

Commands formats are:

- `cor(x, y, method = c("pearson", "kendall", "spearman"))`
- `cor.test(x, y, method=c("pearson", "kendall", "spearman"))`

Correlation - example

The Pearson's correlation coefficient is a measure of linear association. The significance test works under the assumption that x and y are sampled from a bivariate normal distribution.

```
> cor(Data$Sodium,Data$Calories, method = "pearson")
[1] 0.8489548
```

The Kendall and Spearman correlation coefficients are **rank-based** measure of association. This may may be used if the data do not necessarily come from a bivariate normal distribution.

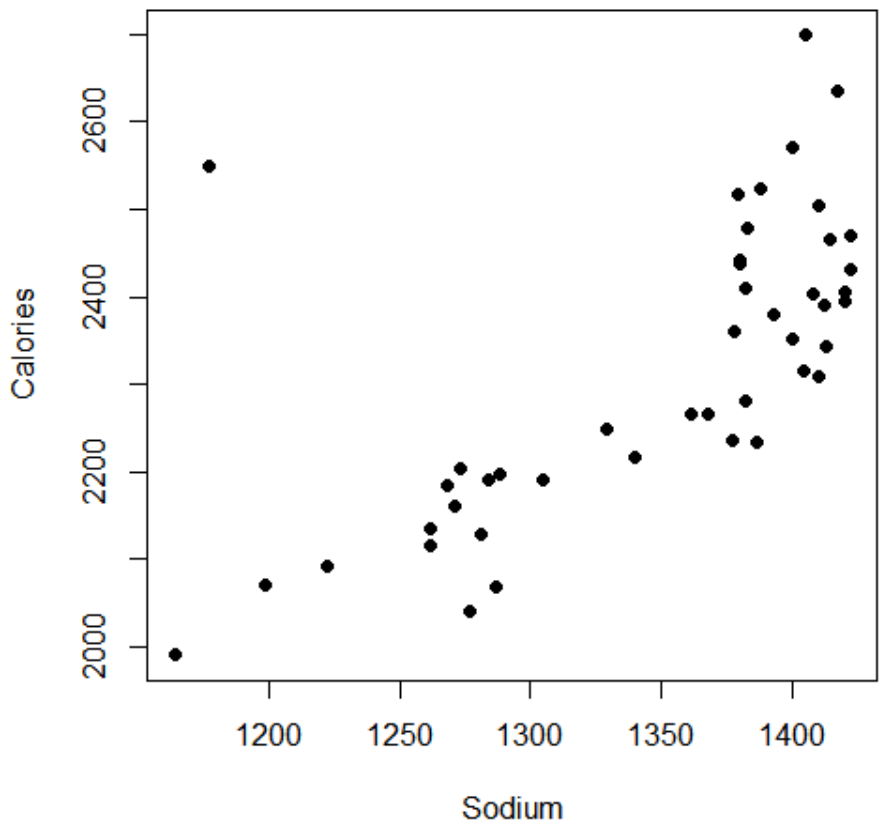
```
> cor(Data$Sodium,Data$Calories, method = "kendall")
[1] 0.6490902
```

```
> cor(Data$Sodium,Data$Calories, method = "spearman")
[1] 0.8201766
```

Correlation - example

Correlation is very sensitive to outliers.

Here we modify a single point in the dataset



Correlation - example

Here are the re-computed correlation coefficients; in parenthesis is the value we calculated above.

```
> cor(Data2$Sodium,Data$Calories, method = "pearson")
```

[1] 0.6866076 (0.8489548)

```
> cor(Data2$Sodium,Data$Calories, method = "kendall")
```

[1] 0.5699824 (0.6490902)

```
> cor(Data2$Sodium,Data$Calories, method =
"spearman")
```

[1] 0.7088825 (0.8201766)

Correlation - example

We want to test if there is a linear relationship between Calories (Y) and Sodium (X) at significance level $\alpha = 0.01$.

Hypothesis testing problem:

$$H_0 : \quad \rho_{X,Y} = 0$$
$$H_1 : \quad \rho_{X,Y} \neq 0$$

If (X, Y) follows a bivariate normal distribution with $\rho_{X,Y} = 0$ and if the samples $\{(x_i, y_i) : i = 1, \ldots n\}$ are i.i.d., then

$$T = \frac{r_{xy}\sqrt{n-2}}{\sqrt{1-r_{xy}^2}} \sim t_{n-2}$$

Hence, we can reject H_0 if $|T| > t_{\alpha/2;n-2}$ where, as usual, $t_{\alpha/2;n-2}$ is the critical value of the t distribution such that $P(T \geq t_{\alpha/2;n-2}) > \alpha/2$.

Correlation - example

Under the assumption above, we can apply the Pearson's correlation analysis

```
> cor.test(Data$Sodium,Data$Calories, method = "pearson")
```

Pearson's product-moment correlation

data: Data$Sodium and Data$Calories
t = 10.534, df = 43, p-value = 1.737e-13
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
0.7397691 0.9145785
sample estimates:
cor
0.8489548

Since p-value = 1.737e-13, we reject the null hypothesis at significance level α = 0.01 (or any other value above the p-value) and accept the alternative hypothesis that $\rho_{X,Y} \neq 0$.

Correlation - example

If the assumption that data come from a normal distribution cannot be satisfied, we can apply the **Spearman's rho statistic** that estimates a rank-based measure of association.

```
> cor.test(Data$Sodium,Data$Calories, method = "spearman")
```

Spearman's rank correlation rho

data: Data$Sodium and Data$Calories
S = 2729.7, p-value = 5.443e-12
alternative hypothesis: true rho is not equal to 0
sample estimates:
rho
0.8201766

Since p-value = 5.443e-12, we reject the null hypothesis at significance level α = 0.01 (or any other value above the p-value) and accept the alternative hypothesis that $\rho_{X,Y} \neq 0$.

Non-parametric correlation

Spearman's rho statistic measures a **rank-based measure of association.**

This test may be used if the data do not come from a bivariate normal distribution. it only requires that each variable at least be measured on the ordinal scale.

Spearman's correlation determines the strength and direction of the **monotonic relationship** between your two variables rather than the strength and direction of the **linear relationship** between your two variables, which is what Pearson's correlation determines.

Monotonicity is less restrictive than a linear relationship.

A monotonic relationship is a relationship that does one of the following: (1) as the value of one variable increases, so does the value of the other variable; or (2) as the value of one variable increases, the other variable value decreases.

Correlation test

If we want to test the more general hypothesis testing problem

$$\begin{aligned} H_0 : &\quad \rho_{X,Y} = \rho_0 \\ H_1 : &\quad \rho_{X,Y} \neq \rho_0, \end{aligned}$$

with $\rho_0 \neq 0$, the mathematical setting becomes more complicated since - still under the assumption that (X, Y) follows a bivariate normal distribution and that the samples $\{(x_i, y_i) : i = 1, \ldots n\}$ are i.i.d. - the formulation of the test statistic becomes more involved and can be solved using the Fisher's z-transformation.

Correlation test - example

Example: we test $H_0 : \rho_{X,Y} = 0.7$ vs $H_1 : \rho_{X,Y} \neq 0.7$, with

We define the `fisherz` function

```
> fisherz=function(r,n,rho0=0){
+ z=log((1+r)/(1-r))/2
+ z0=log((1+rho0)/(1-rho0))/2
+ zstar=(z-z0)*sqrt(n-3)
+ pval=2*(1-pnorm(abs(zstar)))
+ list(pval=pval)}
```

In our example, setting $n = 45$ and $\rho_0 = 0.7$, we compute

```
> fisherz(cor(Data$Sodium,Data$Calories),45,rho0=0.7)
```

$$\$\text{pval} = 0.01257023$$

Data matrix

In applications, it may be useful to compute correlations for several pairs of variables.

Suppose we have a $n \times p$ data matrix

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

- n items/subjects are listed as rows
- p variables are listed as columns

Example: mtcars (Motor Trend Car Road Tests)

The R data set `mtcars` was extracted from the 1974 Motor Trend magazine, and comprises fuel consumption and 10 aspects of car design and performance for 32 automobiles models from 1973-74.

Data frame consists of 32 observations on 11 variables.

[,1]	mpg	Miles/gallon
[,2]	cyl	Number of cylinders
[,3]	disp	Displacement (cu.in.)
[,4]	hp	Gross horsepower
[,5]	drat	Rear axle ratio
[,6]	wt	Weight (lbs/1000)
[,7]	qsec	1/4 mile time
[,8]	vs	Engine (0 = V-shaped, 1 = straight)
[,9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears
[,11]	carb	Number of carburetors

Data matrix example - mtcars

We load the dataset. As remarked above, it consists of 32 observations of 11 variables.

```
> data(mtcars)
> dim(mtcars)
[1] 32 11
> head(mtcars)
```

	mpg	*cyl*	*disp*	*hp*	*drat*	*wt*	*qsec*	*vs*	*am*	*gear*	*carb*
MazdaRX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
MazdaRX4Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet4Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
HornetSportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Data matrix example - mtcars

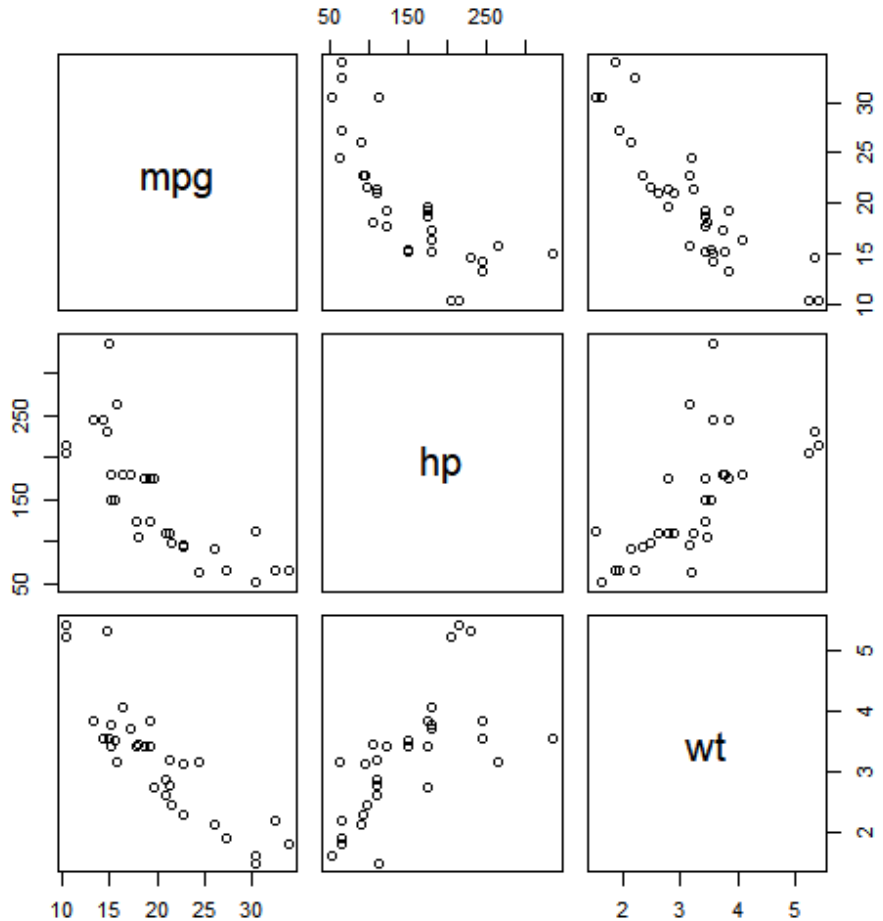
One can explore some statistical properties of the dataset using `summary(mtcars)`.

```
> summary(mtcars)
      mpg             cyl             disp             hp             drat      
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0   Min.   :2.760  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5   1st Qu.:3.080  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0   Median :3.695  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7   Mean   :3.597  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0   3rd Qu.:3.920  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0   Max.   :4.930  
       wt             qsec             vs              am              gear      
 Min.   :1.513   Min.   :14.50   Min.   :0.0000   Min.   :0.0000   Min.   :3.000  
 1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:3.000  
 Median :3.325   Median :17.71   Median :0.0000   Median :0.0000   Median :4.000  
 Mean   :3.217   Mean   :17.85   Mean   :0.4375   Mean   :0.4062   Mean   :3.688  
 3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:4.000  
 Max.   :5.424   Max.   :22.90   Max.   :1.0000   Max.   :1.0000   Max.   :5.000  
      carb      
 Min.   :1.000  
 1st Qu.:2.000  
 Median :2.000  
 Mean   :2.812  
 3rd Qu.:4.000  
 Max.   :8.000  
```

Data matrix example - mtcars

You can also visually explore the dataset by to visualizing the relationship for several pairs of variable

```
> pairs(mtcars[, c("mpg", "hp", "wt")])
```



Data matrix - example

We will use this example to illustrate how to work with a data matrix in R.

Computation of **column means** (first 3 columns)

```
> X <- as.matrix(mtcars)
> colMeans(X)[1:3]
```

mpg	*cyl*	*disp*
20.09062	6.18750	230.72188

This is an alternate way to carry out the same computation

```
> apply(X,2,mean)[1:3]
```

mpg	*cyl*	*disp*
20.09062	6.18750	230.72188

Data matrix - example

One can similarly compute other statistical functions.

Computation of **column median** (first 3 columns)

```
> apply(X,2,median)[1:3]
```

mpg	*cyl*	*disp*
19.2	6.0	196.3

Computation of **column range** (first 3 columns)

```
> apply(X,2,range)[,1:3]
```

	mpg	*cyl*	*disp*
[1,]	10.4	4	71.1
[2,]	33.9	8	472.0

Correlation matrix

Let X be a $n \times p$ data matrix, containing p vectors of length n ordered by columns.

The **correlation matrix** of X is the $p \times p$ symmetric matrix

$$R = \begin{pmatrix} 1 & r_{12} & \dots & r_{1p} \\ r_{21} & 1 & \dots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \dots & 1 \end{pmatrix}$$

where the entries are the pairwise correlations $r_{ij} = r_{ji} = \frac{s_{ij}}{s_i s_j}$ for $1 \leq i, j \leq p$ and $r_{ii} = 1$.

The matrix R contains all the pair-wise correlation coefficients between any column vectors in the matrix X.

Correlation matrix

We want to compute the correlation matrix of the `mtcars` matrix

We first remove the categorical variables `vs` and `am`.

```
library(tidyverse)
X <- mtcars %>% select(-vs, -am)
```

The new data matrix X has dimension 32×9

```
> dim(X)
[1] 32 9
```

```
> head(X)
```

	mpg	*cyl*	*disp*	*hp*	*drat*	*wt*	*qsec*	*gear*	*carb*
MazdaRX4	21.0	6	160	110	3.90	2.620	16.46	4	4
MazdaRX4Wag	21.0	6	160	110	3.90	2.875	17.02	4	4
Datsun710	22.8	4	108	93	3.85	2.320	18.61	4	1
Hornet4Drive	21.4	6	258	110	3.08	3.215	19.44	3	1
HornetSportabout	18.7	8	360	175	3.15	3.440	17.02	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	3	1

Correlation matrix

We compute the correlation matrix of the modified `mtcars` matrix. For easier reading, correlation coefficients are rounded to 2 decimal digits.

```
> round(cor(X), digits = 2)
```

	mpg	*cyl*	*disp*	*hp*	*drat*	*wt*	*qsec*	*gear*	*carb*
mpg	1.00	−0.85	−0.85	−0.78	0.68	−0.87	0.42	0.48	−0.55
cyl	−0.85	1.00	0.90	0.83	−0.70	0.78	−0.59	−0.49	0.53
disp	−0.85	0.90	1.00	0.79	−0.71	0.89	−0.43	−0.56	0.39
hp	−0.78	0.83	0.79	1.00	−0.45	0.66	−0.71	−0.13	0.75
drat	0.68	−0.70	−0.71	−0.45	1.00	−0.71	0.09	0.70	−0.09
wt	−0.87	0.78	0.89	0.66	−0.71	1.00	−0.17	−0.58	0.43
qsec	0.42	−0.59	−0.43	−0.71	0.09	−0.17	1.00	−0.21	−0.66
gear	0.48	−0.49	−0.56	−0.13	0.70	−0.58	−0.21	1.00	0.27
carb	−0.55	0.53	0.39	0.75	−0.09	0.43	−0.66	0.27	1.00

The matrix is symmetric and has diagonal 1, as expected.
This correlation matrix gives an overview of the correlations for all combinations of two variables in X.

Note: we can compute in a very similar way the Spearman correlation matrix. The modified command is as follows:

```
> round(cor(X,method="spearman"), digits = 2 )
```

Correlation matrix

Interpretation of correlation matrix

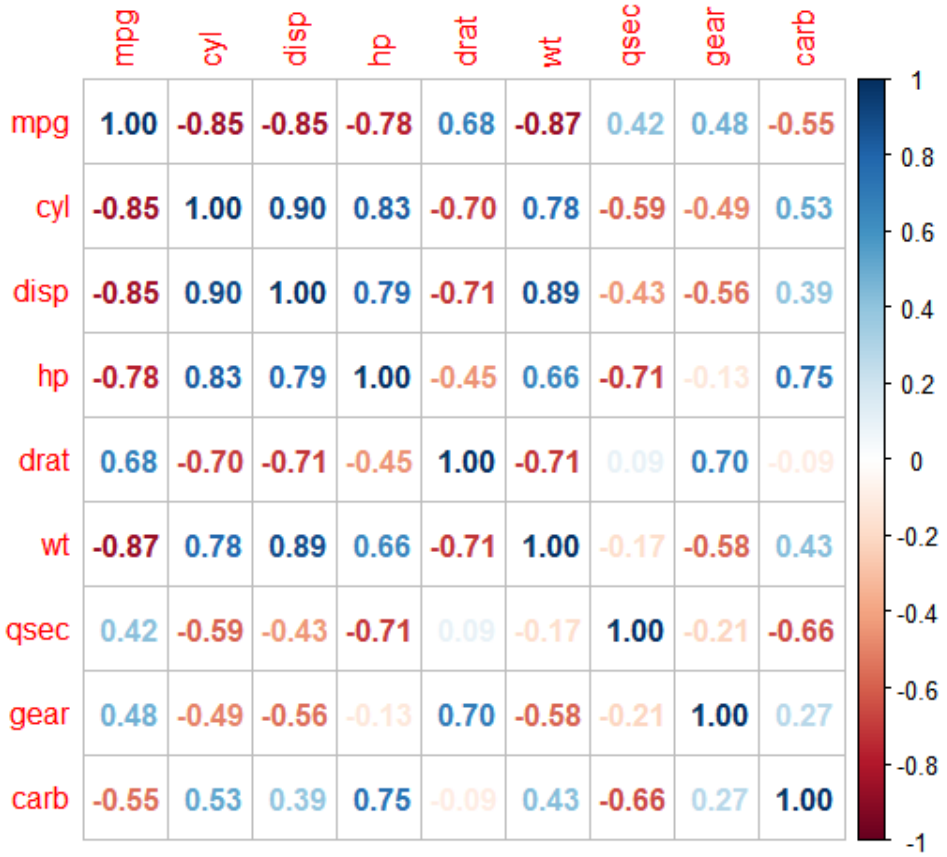
	mpg	*cyl*	*disp*	*hp*	*drat*	*wt*	*qsec*	*gear*	*carb*
mpg	1.00	−0.85	−0.85	−0.78	0.68	−0.87	0.42	0.48	−0.55
cyl	−0.85	1.00	0.90	0.83	−0.70	0.78	−0.59	−0.49	0.53
disp	−0.85	0.90	1.00	0.79	−0.71	0.89	−0.43	−0.56	0.39
hp	−0.78	0.83	0.79	1.00	−0.45	0.66	−0.71	−0.13	0.75
drat	0.68	−0.70	−0.71	−0.45	1.00	−0.71	0.09	0.70	−0.09
wt	−0.87	0.78	0.89	0.66	−0.71	1.00	−0.17	−0.58	0.43
qsec	0.42	−0.59	−0.43	−0.71	0.09	−0.17	1.00	−0.21	−0.66
gear	0.48	−0.49	−0.56	−0.13	0.70	−0.58	−0.21	1.00	0.27
carb	−0.55	0.53	0.39	0.75	−0.09	0.43	−0.66	0.27	1.00

The correlation between horsepower (hp) and miles per gallon (mpg) found above is -0.78, meaning that the 2 variables vary in opposite direction. This makes sense, cars with more horsepower tend to consume more fuel (and thus have a lower millage par gallon). On the contrary, from the correlation matrix we see that the correlation between miles per gallon (mpg) and the time to drive 1/4 of a mile (qsec) is 0.42, meaning that fast cars (low qsec) tend to have a worse millage per gallon (low mpg). This again make sense as fast cars tend to consume more fuel.

Correlation matrix

The R package `corrplot` provides a visual exploratory tool for the correlation matrix.

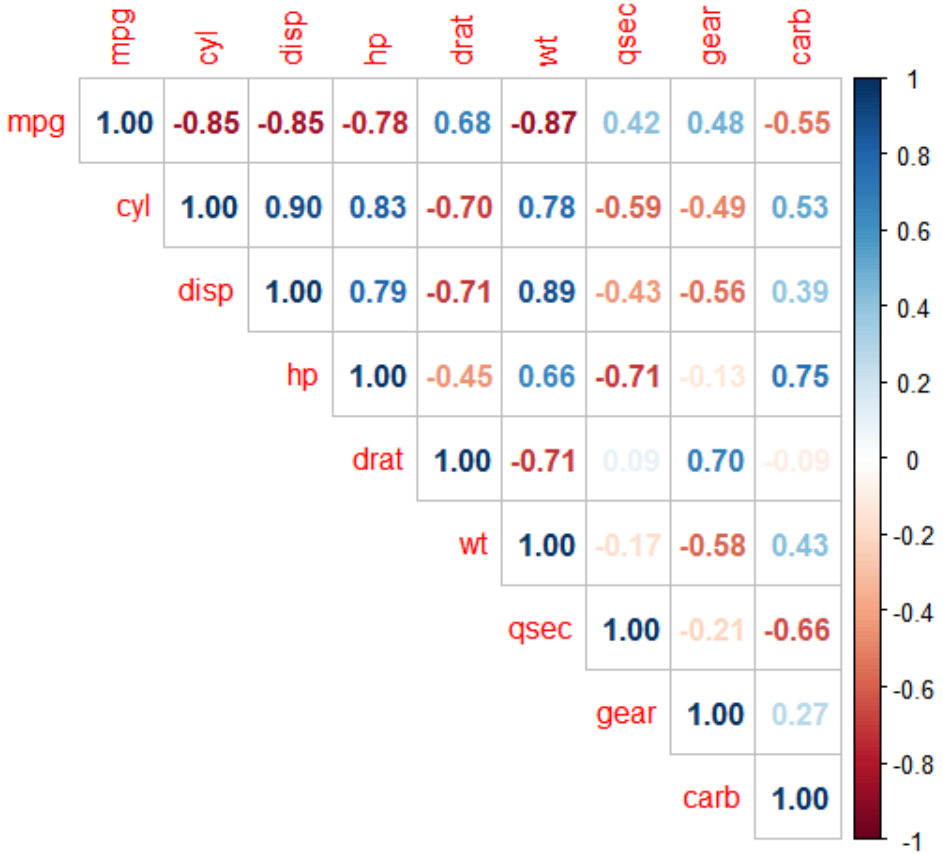
```
> library(corrplot)
> corrplot(cor(X),method = "number")
```



Correlation matrix

Since the matrix is symmetric, we only need to visualize the upper or lower half.

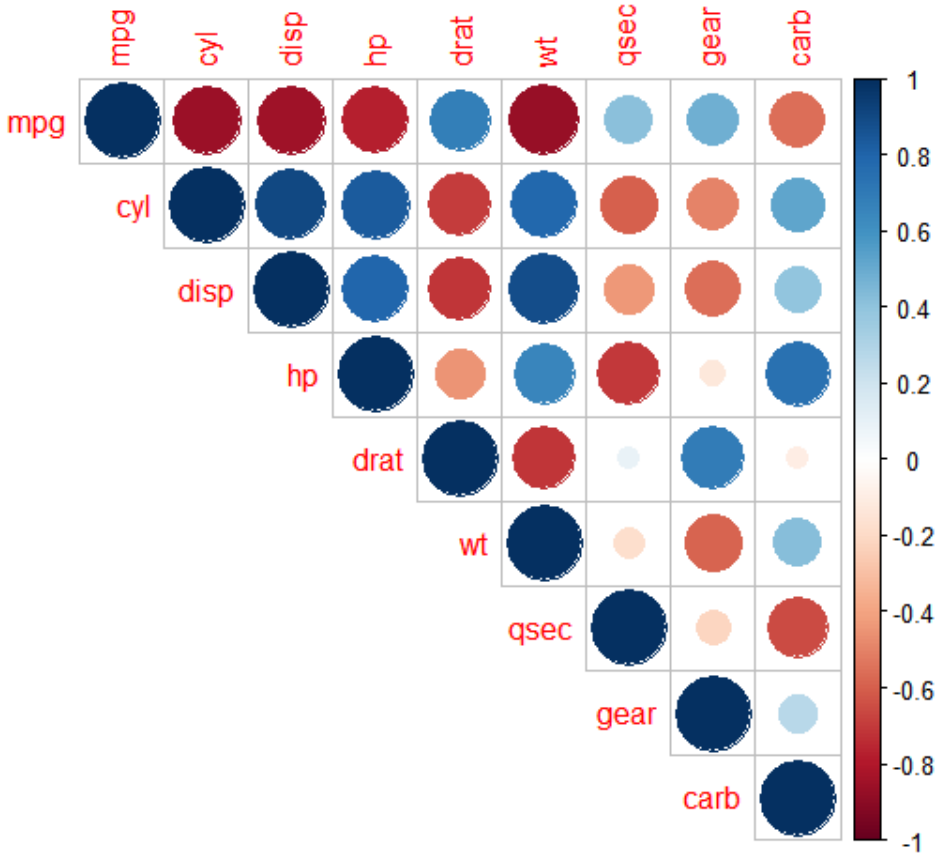
```
> corrplot(cor(X),method = "number",type = "upper")
```



Correlation matrix

Rather than displaying numerical values, one can use colors.

```
> corrplot(cor(X), method = "circle",type = "upper")
```



Correlation matrix

One important application of the correlation matrix is to investigate if there are patterns among specific groups of variables.

The package `corrplot` supports automatic variable reordering.

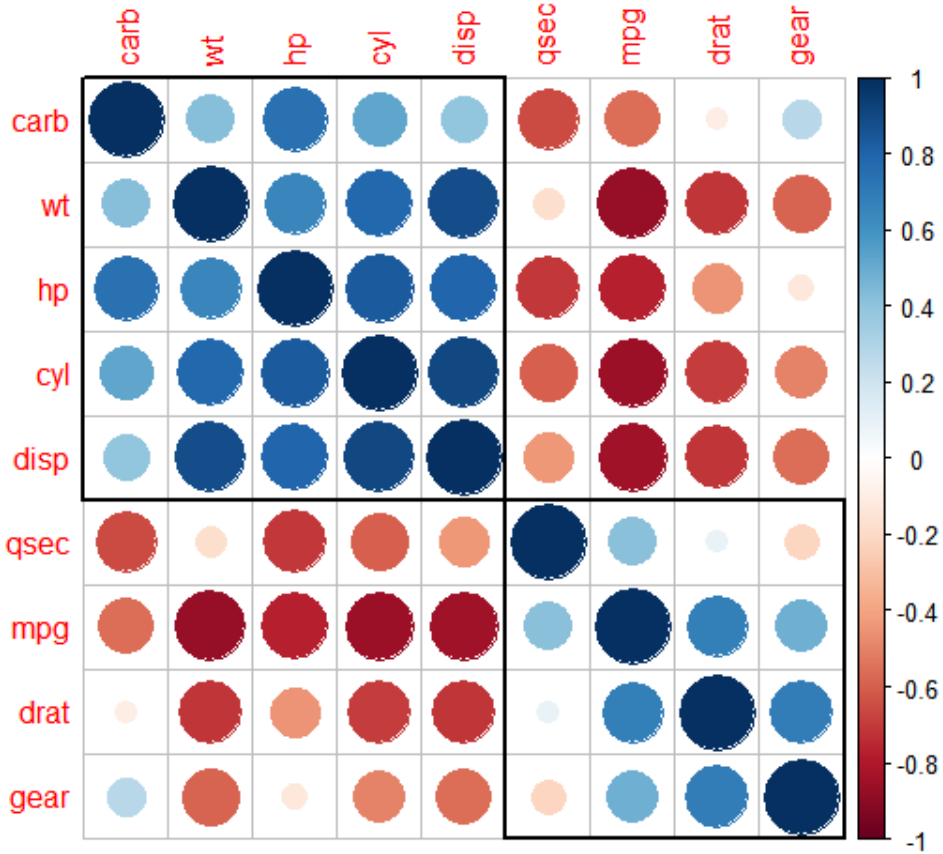
Four order algorithms are available: 'AOE', 'FPC', 'hclust', 'alphabet'.

- 'AOE' is for the angular order of the eigenvectors. It is calculated from the order of the angles corresponding to the largest two eigenvalues of the correlation matrix.
- 'FPC' for the first principal component order.
- 'hclust' for hierarchical clustering order, and 'hclust.method' for the agglomeration method to be used. 'hclust.method' should be one of 'ward', 'ward.D', 'ward.D2', 'single', 'complete', 'average', 'mcquitty', 'median' or 'centroid'.
- 'alphabet' for alphabetical order.

Correlation matrix

The option `'hclust'` draws rectangles around the plot of correlation matrix based on the results of hierarchical clustering.

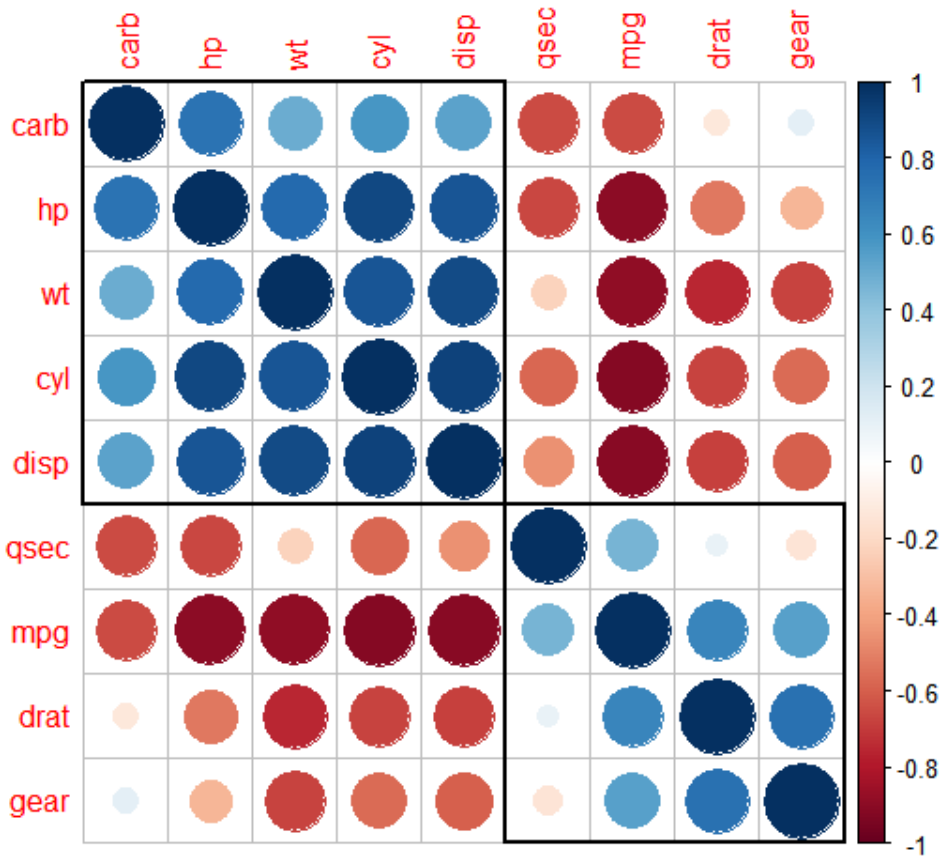
```
> corrplot(cor(X), order = 'hclust', addrect = 2)
```



Correlation matrix

Here we compute again the correlation matrix with the variables reordered based on the results of hierarchical clustering but we use the Spearman correlation.

```
> corrplot(cor(X,method="spearman"), order = 'hclust',
addrect = 2)
```



Correlation matrix

Similar to the `cor()` command used to compute correlation for several pairs of variables in a matrix, the `rcorr()` function from the `Hmisc` package is useful to analyze the correlation matrix.

```
rcorr(X, type=c("pearson","spearman"))
```

It returns

1. the correlation matrix of X
2. the number of observation
3. the p-values for all pairwise correlations

Correlation matrix

```
library(Hmisc)
X <- as.matrix(X) res <- rcorr(X)
```

```
> str(res)
List of 3
$ r: num [1:11, 1:11] 1 -0.852 -0.848 -0.776 0.681 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
.. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
$ n: int [1:11, 1:11] 32 32 32 32 32 32 32 32 32 32 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
.. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
$ P: num [1:11, 1:11] NA 6.11e-10 9.38e-10 1.79e-07 1.78e-05 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
.. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
- attr(*, "class")= chr "rcorr"
```

Correlation matrix

We are displaying only the matrix of p values, with 3 decimal digits

```
> round(res$P, 3)
```

	mpg	*cyl*	*disp*	*hp*	*drat*	*wt*	*qsec*	*gear*	*carb*
mpg	*NA*	0.000	0.000	0.000	0.000	0.000	0.017	0.005	0.001
cyl	0.000	*NA*	0.000	0.000	0.000	0.000	0.000	0.004	0.002
disp	0.000	0.000	*NA*	0.000	0.000	0.000	0.013	0.001	0.025
hp	0.000	0.000	0.000	*NA*	0.010	0.000	0.000	0.493	0.000
drat	0.000	0.000	0.000	0.010	*NA*	0.000	0.620	0.000	0.621
wt	0.000	0.000	0.000	0.000	0.000	*NA*	0.339	0.000	0.015
qsec	0.017	0.000	0.013	0.000	0.620	0.339	*NA*	0.243	0.000
gear	0.005	0.004	0.001	0.493	0.000	0.000	0.243	*NA*	0.129
carb	0.001	0.002	0.025	0.000	0.621	0.015	0.000	0.129	*NA*

Principal Components Analysis

Principal Components Analysis

Principal Components Analysis (PCA) is a statistical method designed to reduce data dimensionality.

PCA extracts the most significant information from a set of multiple variables and represents this information as a set of new variables, called **principal components**, obtained as linear combinations of the original ones.

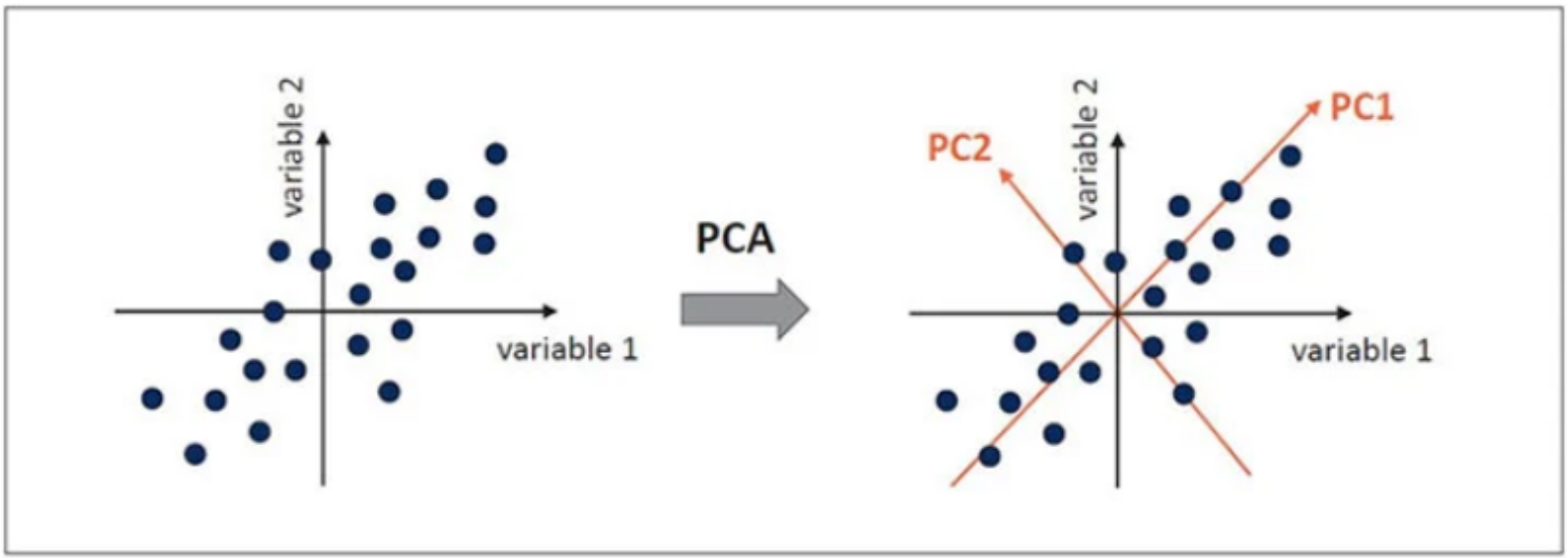
PCA allows one to identify a few principal components that can be visualized graphically with minimal loss of information. It is especially useful when dealing with three or higher dimensional data.

The dominant principal components are the most important for explaining covariation in the data.

Principal Components Analysis

The key idea is to apply a change of coordinates where the new coordinate axes are aligned along the directions of largest variation.

In the 2-dimensional examples below, the PC1 axis is the first principal direction along which the samples show the largest variation. The PC2 axis is the second most important direction, and it is orthogonal to the PC1 axis.



Principal Components Analysis

PCA differs from clustering which is also used for data interpretation.

- Clustering assumes that each data point is a member of one, and only one, cluster. Clusters are mutually exclusive.
- PCA assumes that each data point is a linear combination of multiple basic "ingredients" which are not mutually exclusive.

Principal Components Analysis

I will use dataset `decathlon2` from the `factoextra` package to illustrate the ue of PCA.

```
> install.packages("factoextra")
> library("factoextra")
> data(decathlon2)
> head(decathlon2)
```

	X100m	L.jump	Shot.put	H.jump	X400m	X110m.hurdle	Discus	Pole.vault	Javeline	X1500m	Rank	Points	Competition
SEBRLE	11.04	7.58	14.83	2.07	49.81	14.69	43.75	5.02	63.19	291.7	1	8217	Decastar
CLAY	10.76	7.40	14.26	1.86	49.37	14.05	50.72	4.92	60.15	301.5	2	8122	Decastar
BERNARD	11.02	7.23	14.25	1.92	48.93	14.99	40.87	5.32	62.77	280.1	4	8067	Decastar
YURKOV	11.34	7.09	15.19	2.10	50.42	15.31	46.26	4.72	63.44	276.4	5	8036	Decastar
ZSIVOCZKY	11.13	7.30	13.48	2.01	48.62	14.17	45.67	4.42	55.37	268.0	7	8004	Decastar
McMULLEN	10.83	7.31	13.76	2.13	49.91	14.38	44.41	4.42	56.37	285.1	8	7995	Decastar

Principal Components Analysis

The dataset `decathlon2` describes athletes' performance during two sporting events (Desctar and OlympicG).

It contains 27 athletes described by 13 variables (sport disciplines).

For further analysis, I will subset active individuals (rows 1:23) and active variables (columns 1:10) from the `decathlon2` dataset, therefore I will create new dataset decathlon2b to conduct the principal component analysis.

```
decathlon2b <- decathlon2[1:23, 1:10]
```

The `decathlon2b` dataset consists of 23 observations and 10 variables.

```
> dim(decathlon2b)
[1] 23 10
```

Principal Components Analysis

The summary statistics below shows the distribution of observations.

```
> summary(decathlon2b)
```

X100m	Long.jump	Shot.put	High.jump	X400m
Min. :10.44	Min. :6.800	Min. :12.68	Min. :1.860	Min. :46.81
1st Qu.:10.84	1st Qu.:7.165	1st Qu.:14.17	1st Qu.:1.940	1st Qu.:48.95
Median :10.97	Median :7.310	Median :14.65	Median :2.010	Median :49.40
Mean :11.00	Mean :7.350	Mean :14.62	Mean :2.007	Mean :49.43
3rd Qu.:11.23	3rd Qu.:7.525	3rd Qu.:15.14	3rd Qu.:2.095	3rd Qu.:50.02
Max. :11.64	Max. :7.960	Max. :16.36	Max. :2.150	Max. :51.16
X110m.hurdle	Discus	Pole.vault	Javeline	X1500m
Min. :13.97	Min. :37.92	Min. :4.400	Min. :52.33	Min. :262.1
1st Qu.:14.17	1st Qu.:43.74	1st Qu.:4.610	1st Qu.:55.40	1st Qu.:268.8
Median :14.37	Median :44.75	Median :4.820	Median :57.44	Median :278.1
Mean :14.53	Mean :45.16	Mean :4.797	Mean :59.11	Mean :277.9
3rd Qu.:14.94	3rd Qu.:46.93	3rd Qu.:5.000	3rd Qu.:62.98	3rd Qu.:283.6
Max. :15.67	Max. :51.65	Max. :5.320	Max. :70.52	Max. :301.5

Principal Components Analysis

The R command to perform a principal components analysis on a data matrix x is the following

```
prcomp(x, retx = TRUE, center = TRUE, scale = FALSE,
tol = NULL, ...)
```

- retx: A logical value indicating whether the rotated variables should be returned. The default is TRUE.
- center: A logical value indicating whether the variables should be shifted to be 0 centered. Alternately, a vector of length equal the number of columns of x can be supplied. The value is passed to scale. The default is TRUE.
- scale: A logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place. The default is FALSE. In general, scaling is advisable.
- tol: a value indicating the magnitude below which components should be omitted. With the default NULL setting, no components are omitted.

Principal Components Analysis

```
> res.pca <- prcomp(decathlon2b, scale = TRUE)
> print(res.pca)
Standard deviations (1, .., p=10):

 [1] 2.0308159 1.3559244 1.1131668 0.9052294 0.8375875 0.6502944 0.5500742
 [8] 0.5238988 0.3939758 0.3492435

Rotation (n x k) = (10 x 10):

                     PC1         PC2         PC3         PC4         PC5
X100m        -0.418859080  0.13230683 -0.27089959  0.03708806 -0.2321476
Long.jump     0.391064807 -0.20713320  0.17117519 -0.12746997  0.2783669
Shot.put      0.361388111 -0.06298590 -0.46497777  0.14191803 -0.2970589
High.jump     0.300413236  0.34309742 -0.29652805  0.15968342  0.4807859
X400m        -0.345478567 -0.21400770 -0.25470839  0.47592968  0.1240569
X110m.hurdle -0.376265119  0.01824645 -0.40325254 -0.01866477  0.2676975
Discus        0.365965721 -0.03662510 -0.15857927  0.43636361 -0.4873988
Pole.vault   -0.106985591 -0.59549862 -0.08449563 -0.37447391 -0.2646712
Javeline      0.210864329 -0.28475723 -0.54270782 -0.36646463  0.2361698
X1500m        0.002106782 -0.57855748  0.19715884  0.49491281  0.3142987
                     PC6         PC7         PC8         PC9         PC10
X100m         0.054398099 -0.16604375 -0.19988005 -0.76924639  0.12718339
Long.jump    -0.051865558 -0.28056361 -0.75850657 -0.13094589  0.08509665
Shot.put     -0.368739186 -0.01797323  0.04649571  0.12129309  0.62263702
High.jump    -0.437716883  0.05118848  0.16111045 -0.28463225 -0.38244596
X400m        -0.075796432  0.52012255 -0.44579641  0.20854176 -0.09784197
X110m.hurdle  0.004048005 -0.67276768 -0.01592804  0.41058421 -0.04475363
Discus        0.305315353 -0.25946615 -0.07550934  0.03391600 -0.49418361
Pole.vault   -0.503563524 -0.01889413  0.06282691 -0.06540692 -0.39288155
Javeline      0.556821016  0.24281145  0.10086127 -0.10268134 -0.01103627
X1500m        0.064663250 -0.20245828  0.37119711 -0.25950868  0.17991689
```

Principal Components Analysis

- The `prcomp()` function was set to center data around 0 by shifting the variables (center = TRUE) and to rescale the variance to 1 (scale = FALSE); data standarization is needed since variables were measured in different scales.
- The `prcomp()` function computed the 10 principal components which also correspond to the number of variables (10 sport disciplines) in the data.
  Recall that data matrix contains 23 observations (athletes) of 10 variables (sport disciplines).
- Each principal component (PC) explains a percentage of the total variance in the data set.

Principal Components Analysis

Each PC explains a percentage of the total variance in the data set.

```
> summary(res.pca)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5
Standard deviation     2.0308 1.3559 1.1132 0.90523 0.83759
Proportion of Variance 0.4124 0.1839 0.1239 0.08194 0.07016
Cumulative Proportion  0.4124 0.5963 0.7202 0.80213 0.87229
                           PC6     PC7     PC8     PC9   PC10
Standard deviation     0.65029 0.55007 0.52390 0.39398 0.3492
Proportion of Variance 0.04229 0.03026 0.02745 0.01552 0.0122
Cumulative Proportion  0.91458 0.94483 0.97228 0.98780 1.0000
```

- PC1 explains 41.24% of total variance, PC2 explains 18.39% of total variance and so on.
- The Cumulative Proportion section shows that the first 3 PCs explains about 72% of the total variance and the first 4 PCs explains about 80% of the total variance.

Principal Components Analysis

The amount of variation held by each principal component is associated with the **eigenvalues** of the PCA.

The eigenvalues are extracted by `get_eigenvalue()` function.

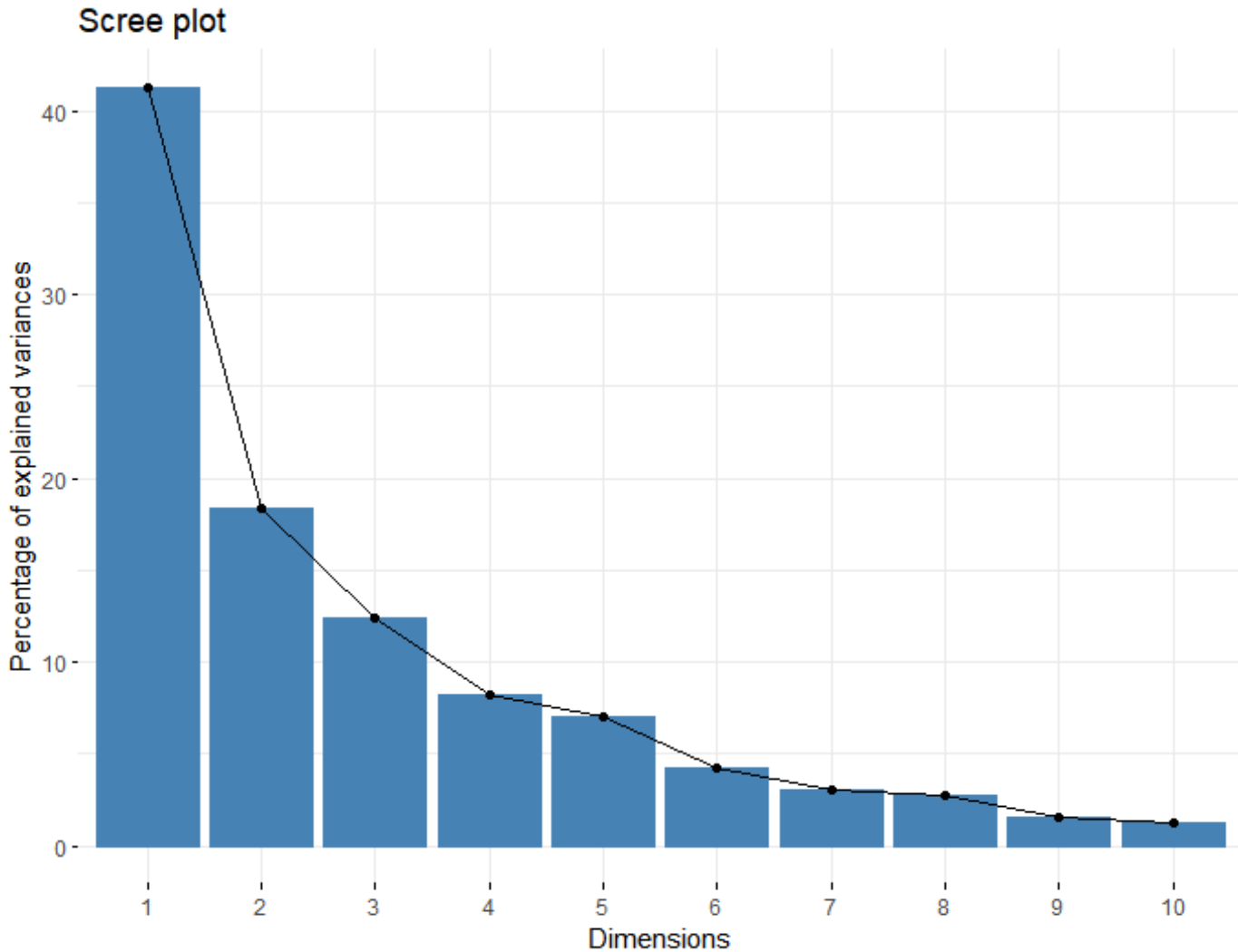
```
> eig.val<-get_eigenvalue(res.pca)
> eig.val
```

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	4.1242133	41.242133	41.24213
Dim.2	1.8385309	18.385309	59.62744
Dim.3	1.2391403	12.391403	72.01885
Dim.4	0.8194402	8.194402	80.21325
Dim.5	0.7015528	7.015528	87.22878
Dim.6	0.4228828	4.228828	91.45760
Dim.7	0.3025817	3.025817	94.48342
Dim.8	0.2744700	2.744700	97.22812
Dim.9	0.1552169	1.552169	98.78029
Dim.10	0.1219710	1.219710	100.00000

Principal Components Analysis

The importance of PCs decreases rapidly with the dimension.

```
> fviz_eig(res.pca, col.var="blue")
```



Principal Components Analysis

The **Scree plot** displays the variance by each PC.

It always displays a downward curve. It typically starts high, then falls rather quickly and finally flattens out. This is because the first component usually explains much of the variability, the next few components explain a moderate amount, and the latter components only explain a small fraction of the overall variability.

The scree plot is useful to decide the number of PCs that are sufficient to provide a satisfactory approximation to explain the data. scree plot criterion looks for the **"elbow"** in the curve and selects all components just before the line flattens out.

Note: it is called a 'scree' plot (in the PCA literature) because it often looks like a 'scree' slope, where rocks have fallen down and accumulated on the side of a mountain.

Principal Components Analysis

PCA results can be assessed with regard to **variables** (sport disciplines) and **observations/individuals** (athletes).

For that purpose, the function `get_pca_var()` provides a list of matrices containing all the results for the **variables**: coordinates, correlation between variables and axes, squared cosine, and contributions.

Similarly, the function `get_pca_ind()` provides a list of matrices containing all the results for the **observations/individuals**: coordinates, squared cosine, and contributions.

Principal Components Analysis

```
> var <- get_pca_var(res.pca)
> var
Principal Component Analysis Results for variables
===================================
  Name       Description
1 "$coord"   "Coordinates for the variables"
2 "$cor"     "Correlations between variables and dimensions"
3 "$cos2"    "Cos2 for the variables"
4 "$contrib" "contributions of the variables"

> ind <- get_pca_ind(res.pca)
> ind
Principal Component Analysis Results for individuals
===================================
  Name       Description
1 "$coord"   "Coordinates for the individuals"
2 "$cos2"    "Cos2 for the individuals"
3 "$contrib" "contributions of the individuals"
```

Principal Components Analysis

We start by assessing PCA results with regard to variables.

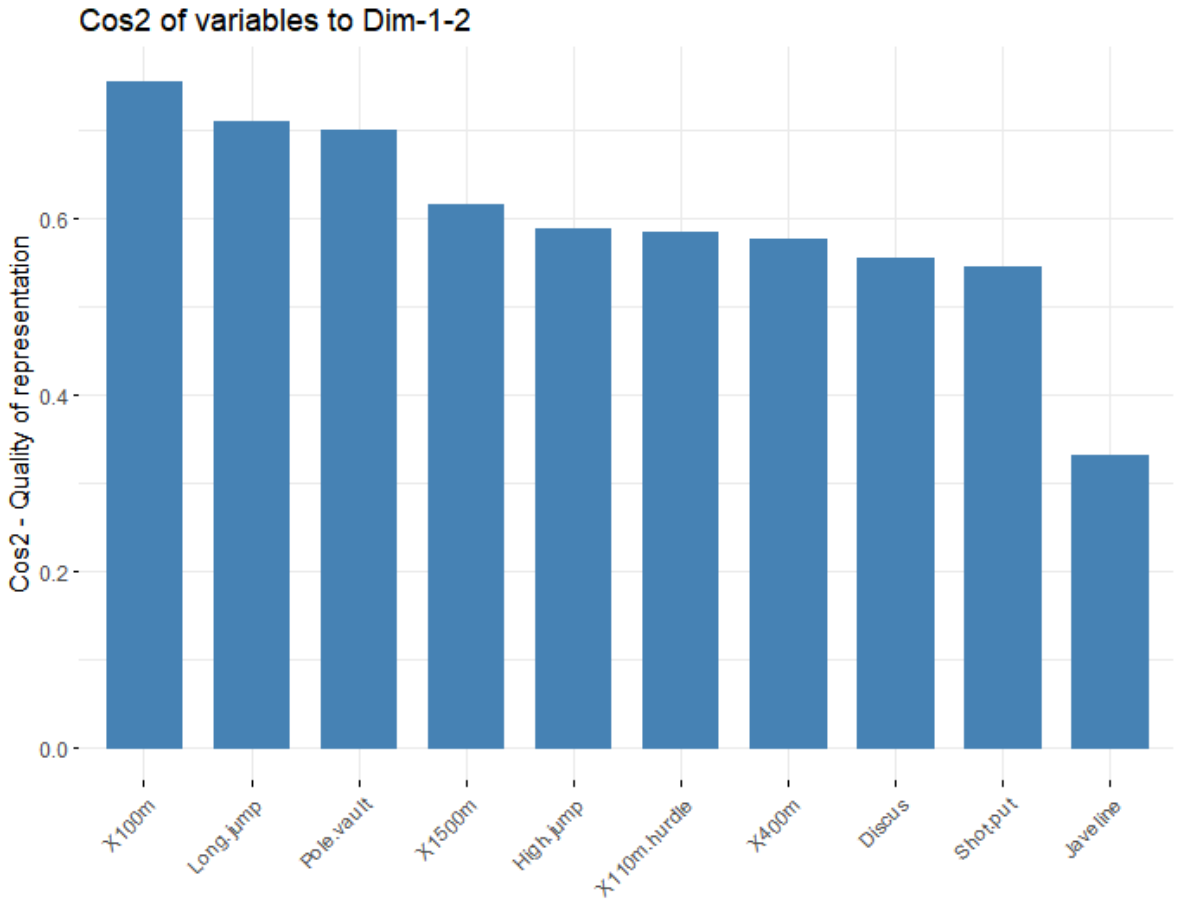
In the list above:

- `Cos2` is called **square cosine** and shows the importance of a principal component for a given observation.
  - A low value means that the variable is not well represented by that component
  - A high value, on the other hand, means a good representation of the variable on that component.
- `Contrib` indicates the **contribution** of the variables.

Principal Components Analysis

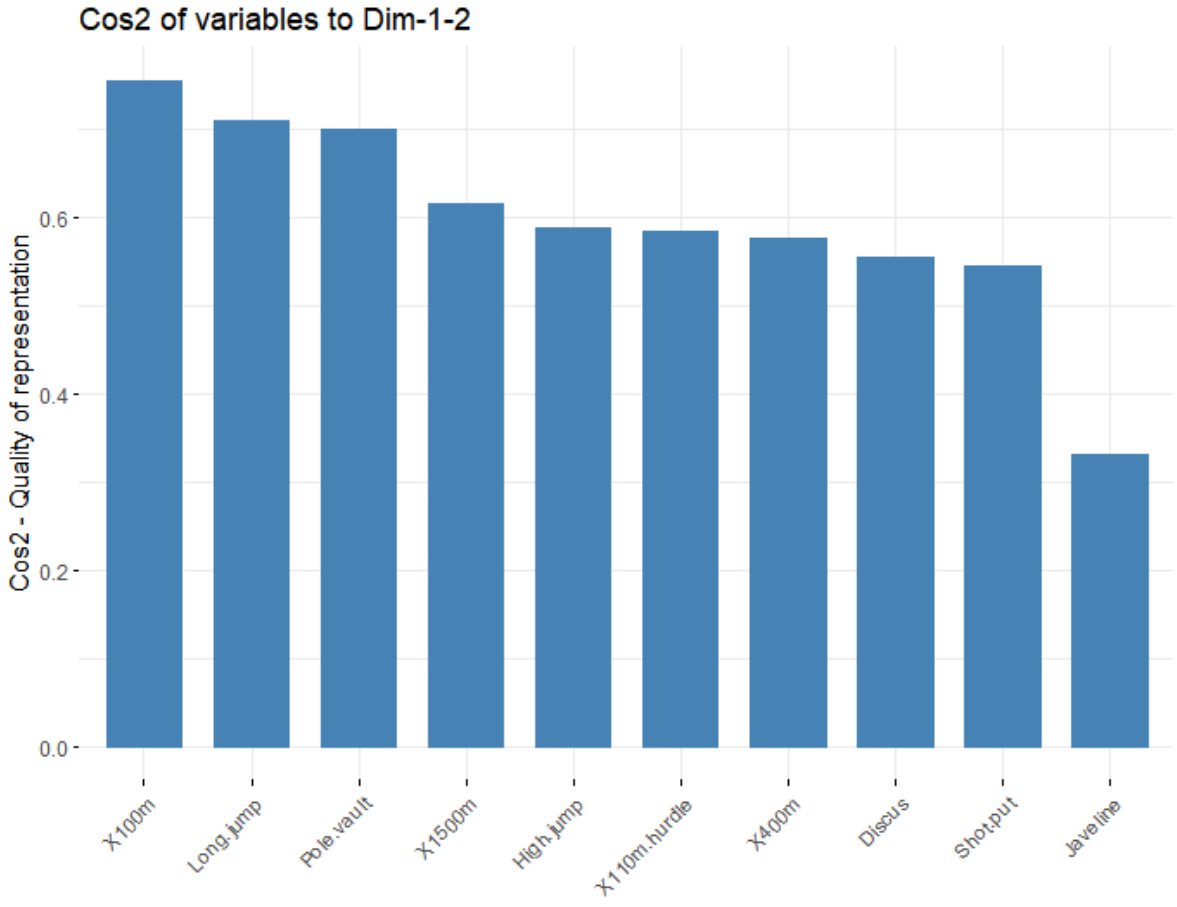
The code below computes the square cosine value for each variable with respect to the first two principal components PC1, PC2.

```
> fviz_cos2(res.pca, choice = "var", axes = 1:2)
```



Principal Components Analysis

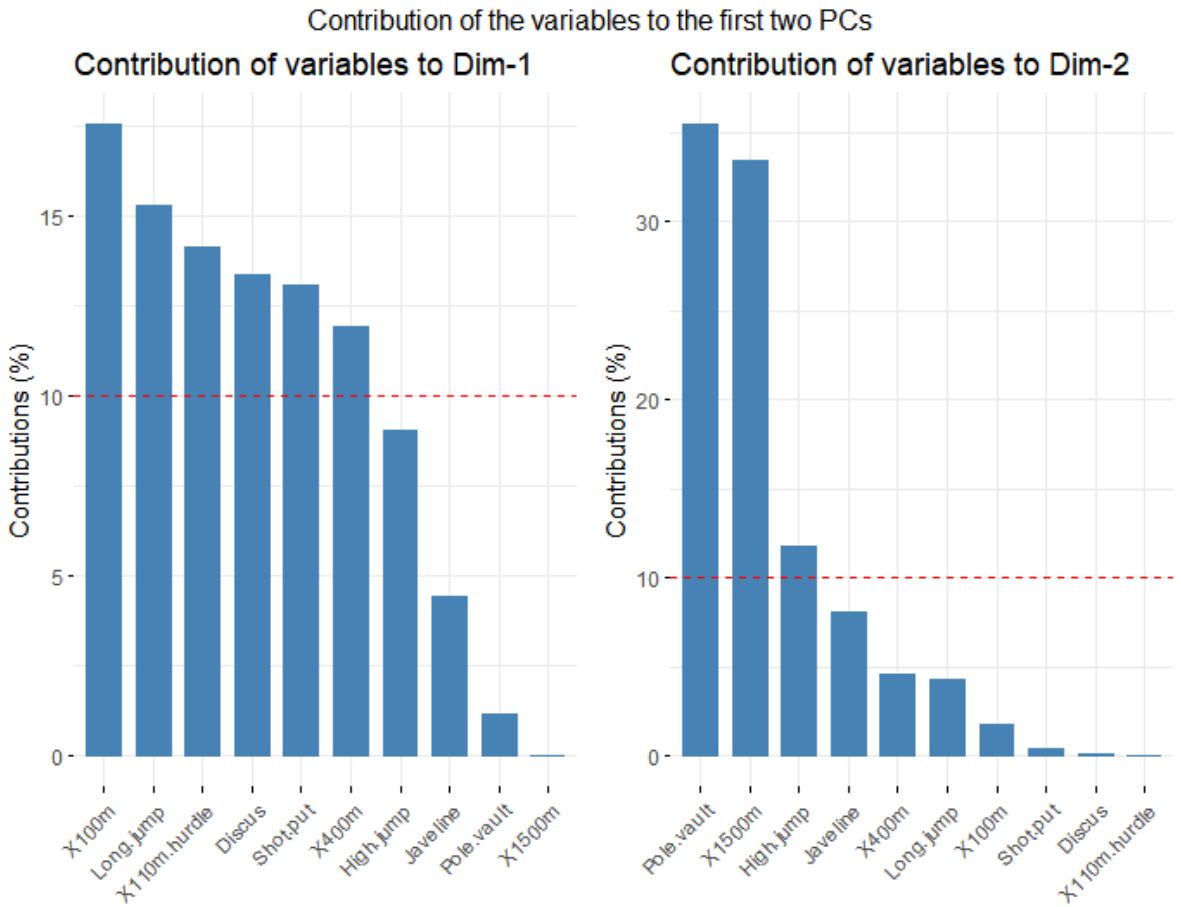
From the plot, X100m, Long jump and Pole vault are the top three variables with the highest cos2, hence they are well represented in PC1 and PC2.



Principal Components Analysis

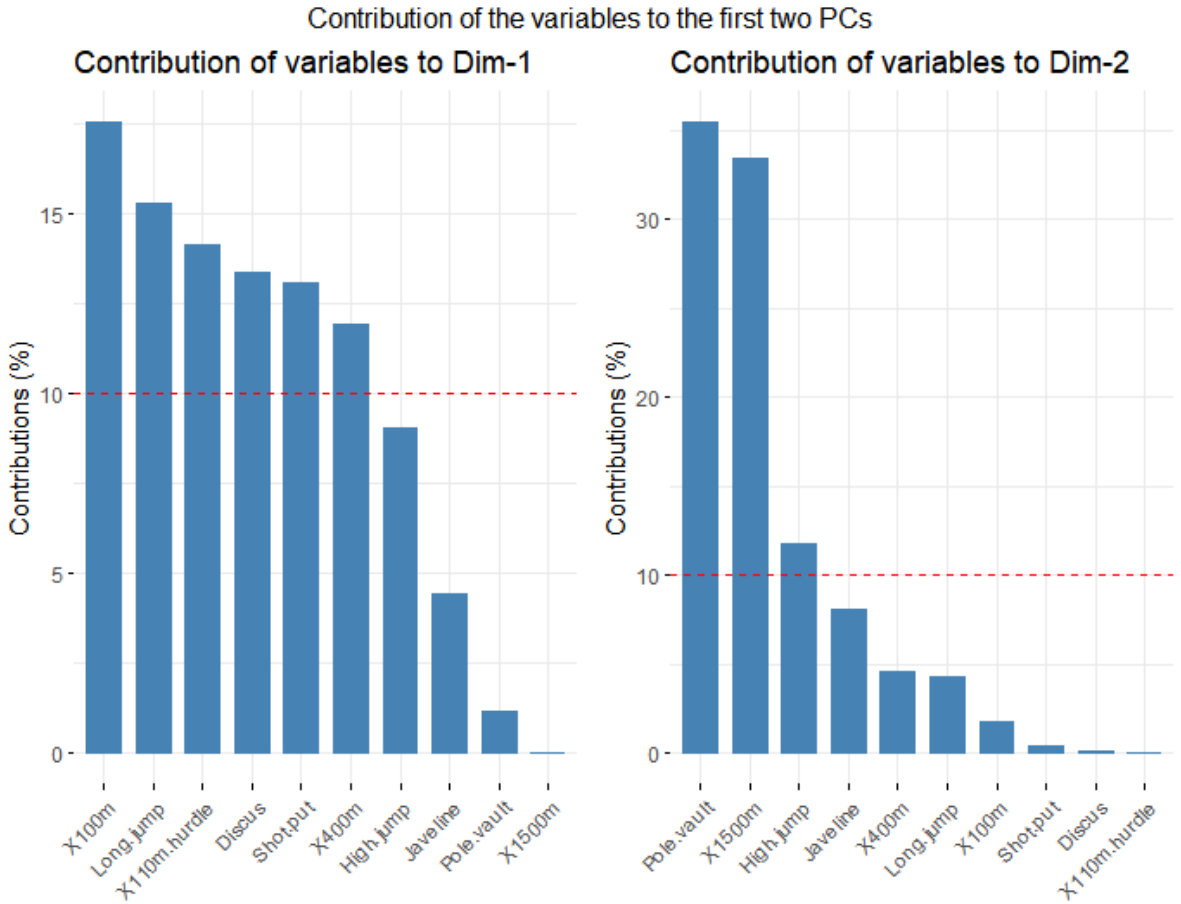
The code below shows variable contributions to PC1 and PC2.

```
> a1<-fviz_contrib(res.pca, choice = "var", axes = 1)

> a2<-fviz_contrib(res.pca, choice = "var", axes = 2)

> library("gridExtra")

> grid.arrange(a1,a2,ncol=2, top='Contribution of the variables to the first two PCs')
```



Principal Components Analysis

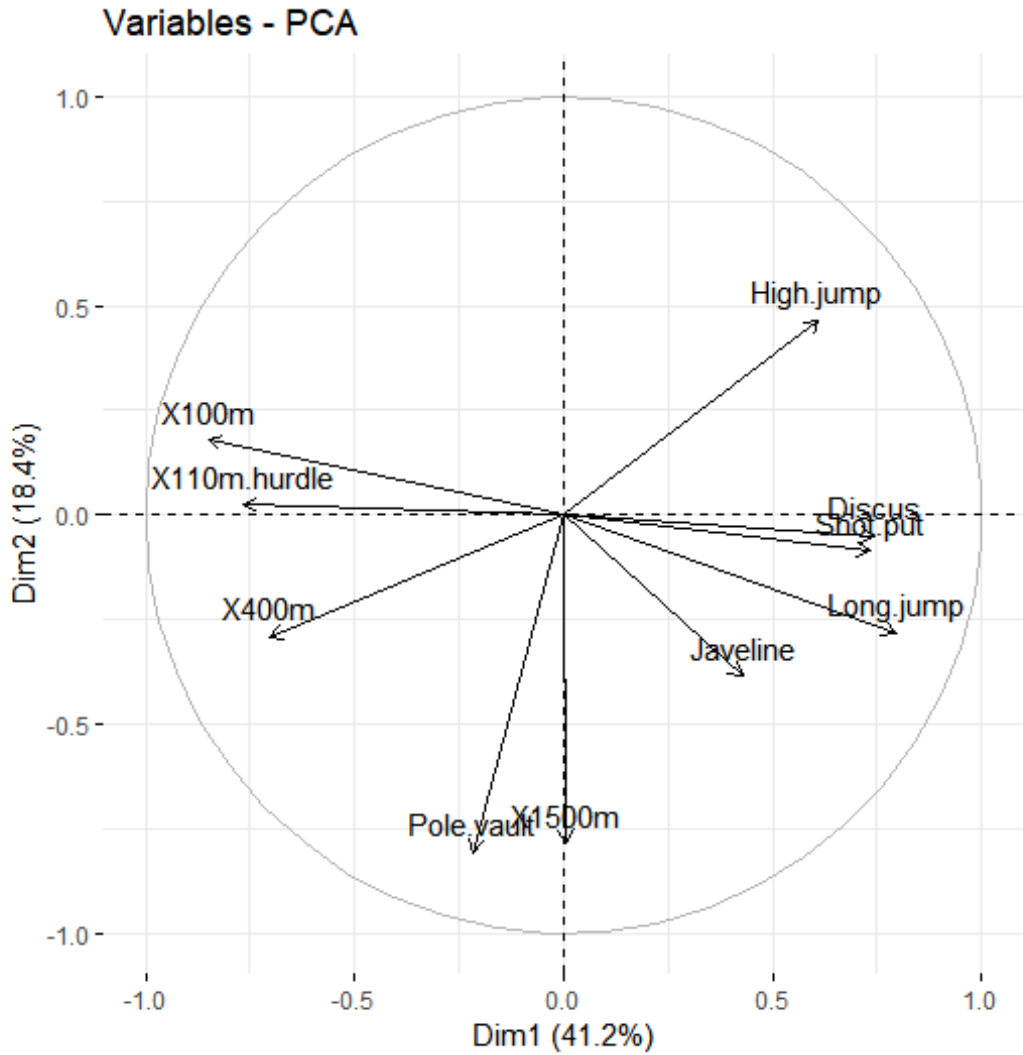
The red dashed line on the graph above indicates the expected average contribution. A variable with a contribution exceeding this benchmark is considered as important in contributing to the PC. It can be seen that the variables X100m, Long.jump and Pole.vault contribute the most to both dimensions.



Principal Components Analysis

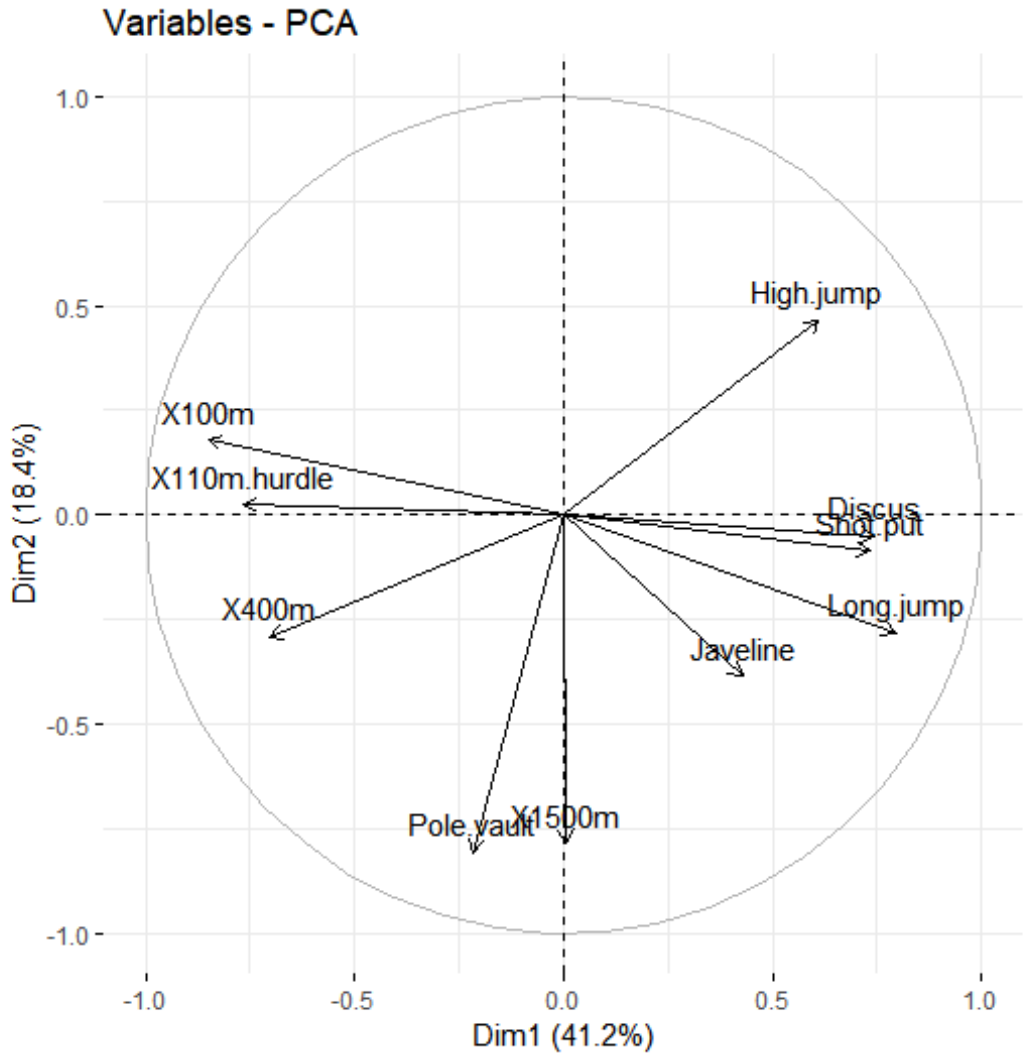
The **biplot** visualizes similarities/dissimilarities between variables.

```
> fviz_pca_var(res.pca, col.var = "black")
```



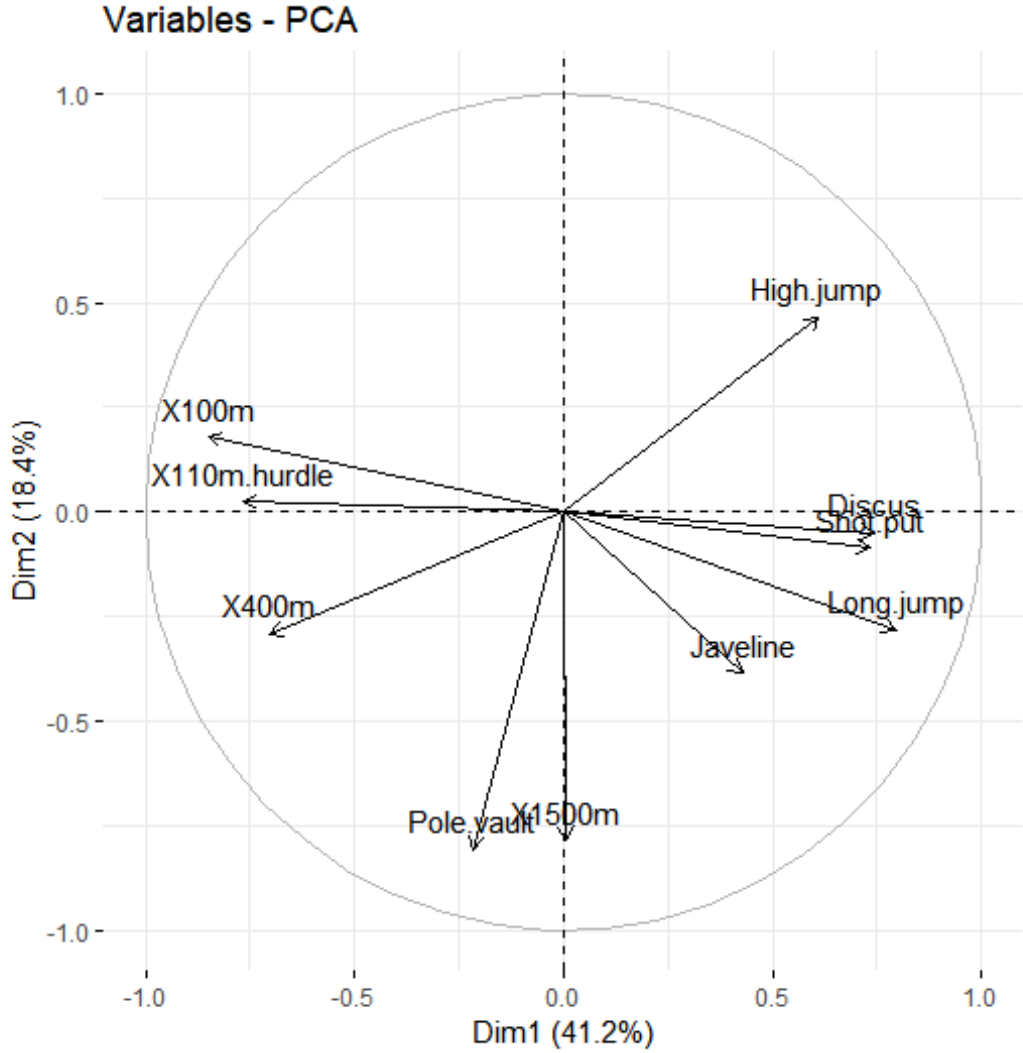
Principal Components Analysis

- Variables that are grouped together are positively correlated to each other and variables that are negatively correlated are displayed to the opposite sides of the biplot's origin.



Principal Components Analysis

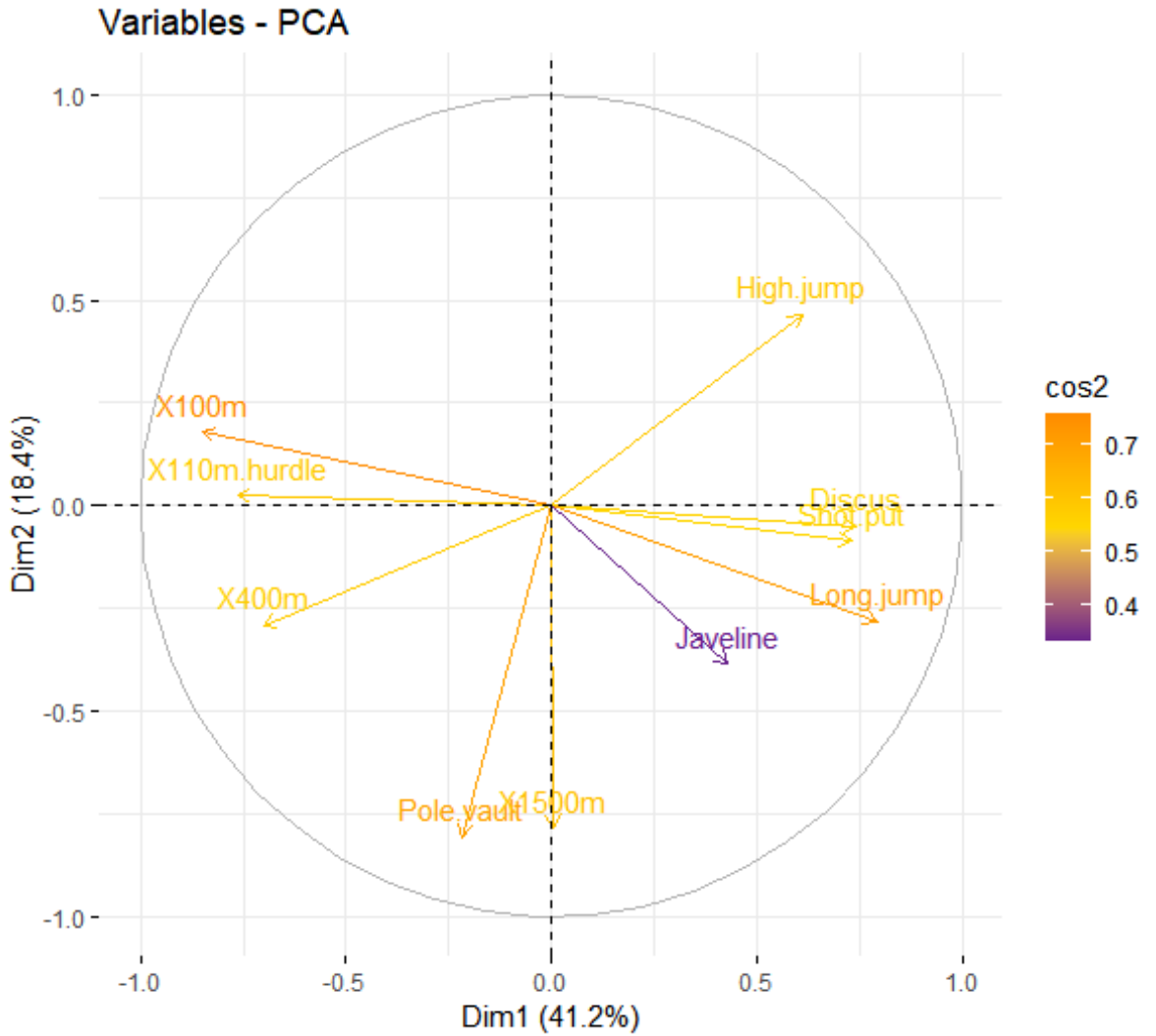
- The higher the distance between the variable and the origin, the better represented that variable is in the principal components PC1 and PC2.



Principal Components Analysis

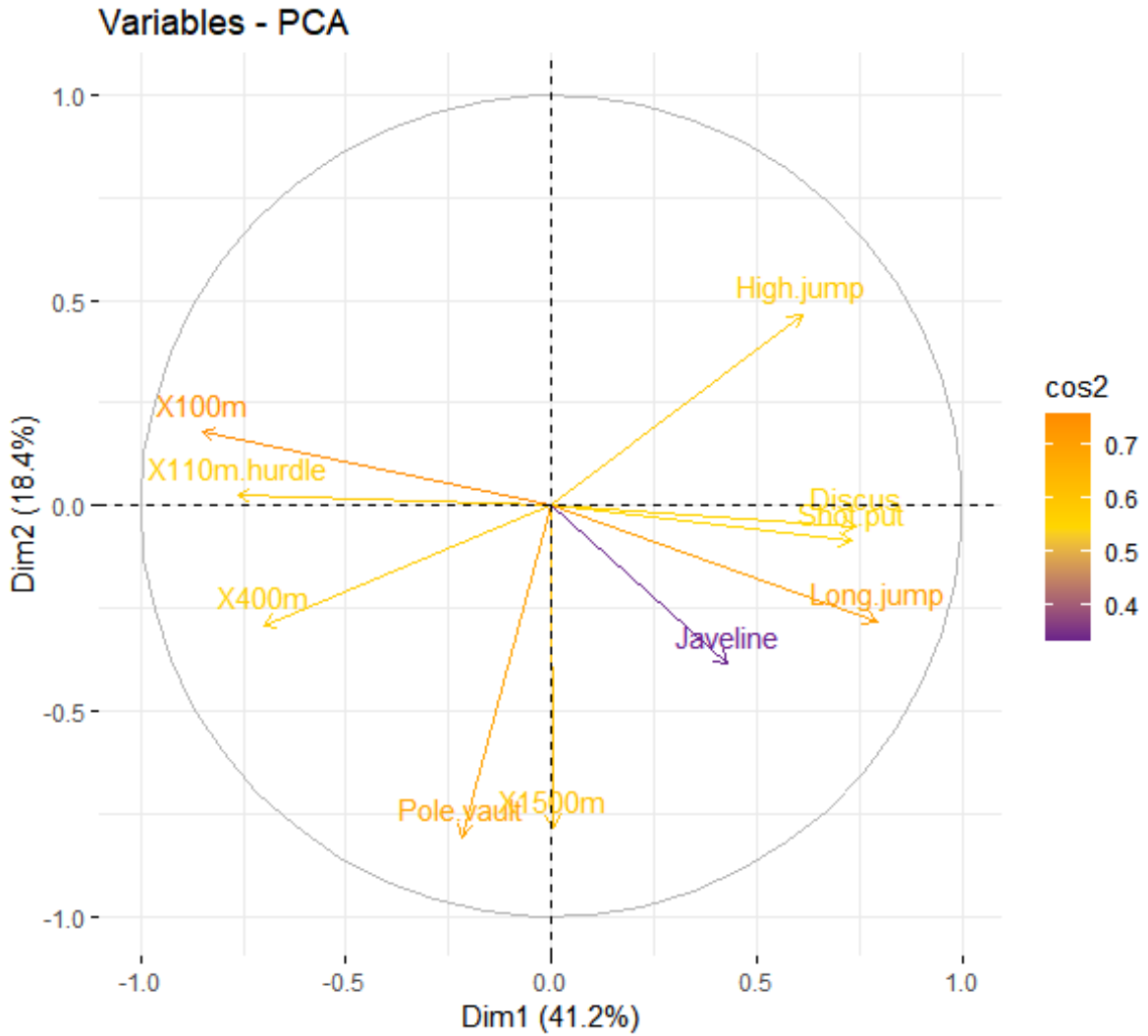
The quality of representation of variables can be drawn on the plot.

```
> fviz_pca_var(res.pca, col.var = "cos2", gradient.cols =
c("darkorchid4", "gold", "darkorange"), )
```



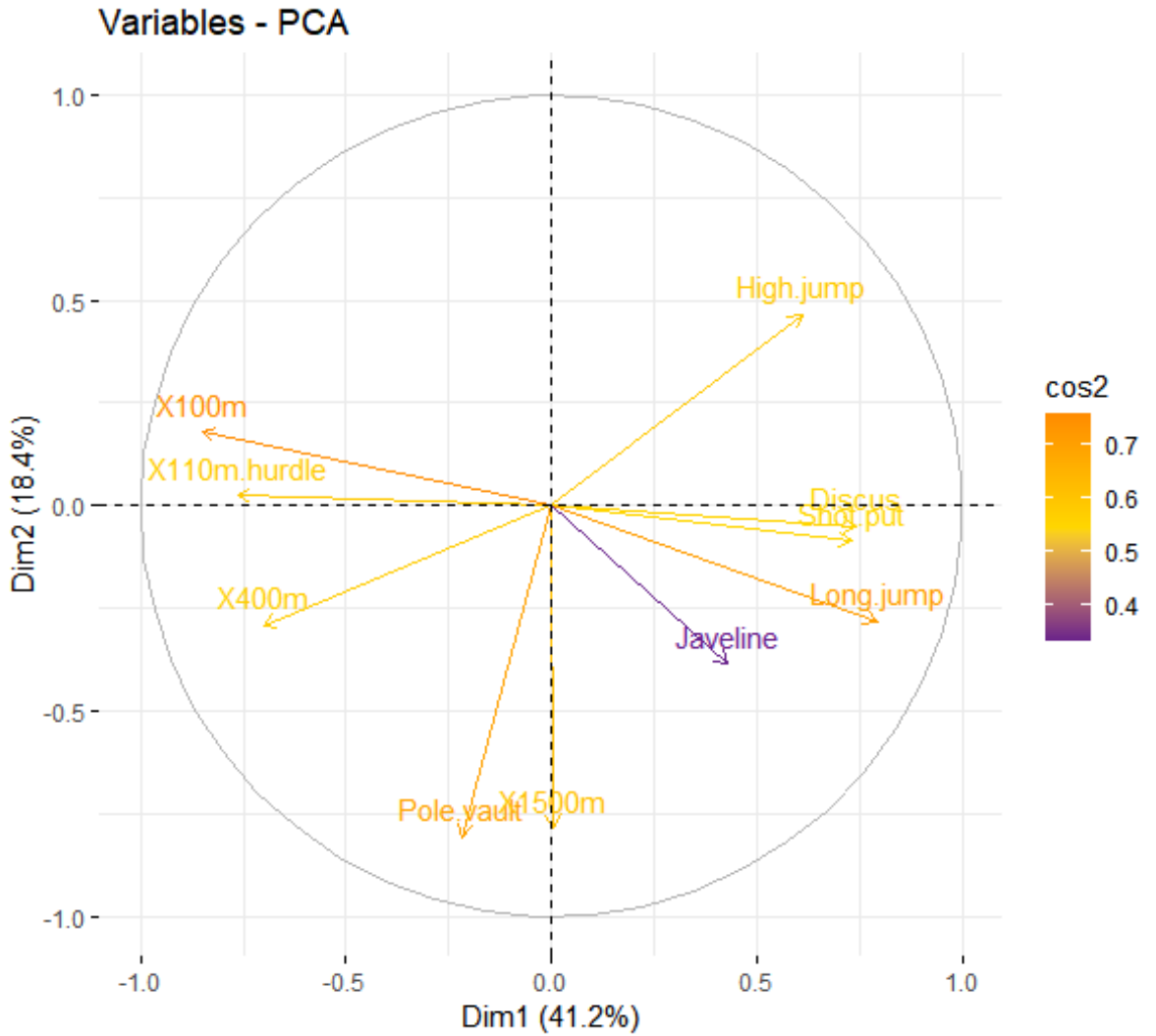
Principal Components Analysis

In the plot, cos2 values differ by gradient colors: variables with low cos2 values are colored "darkorchid4", medium cos2 values - "gold", high co2 values - "darkorange".



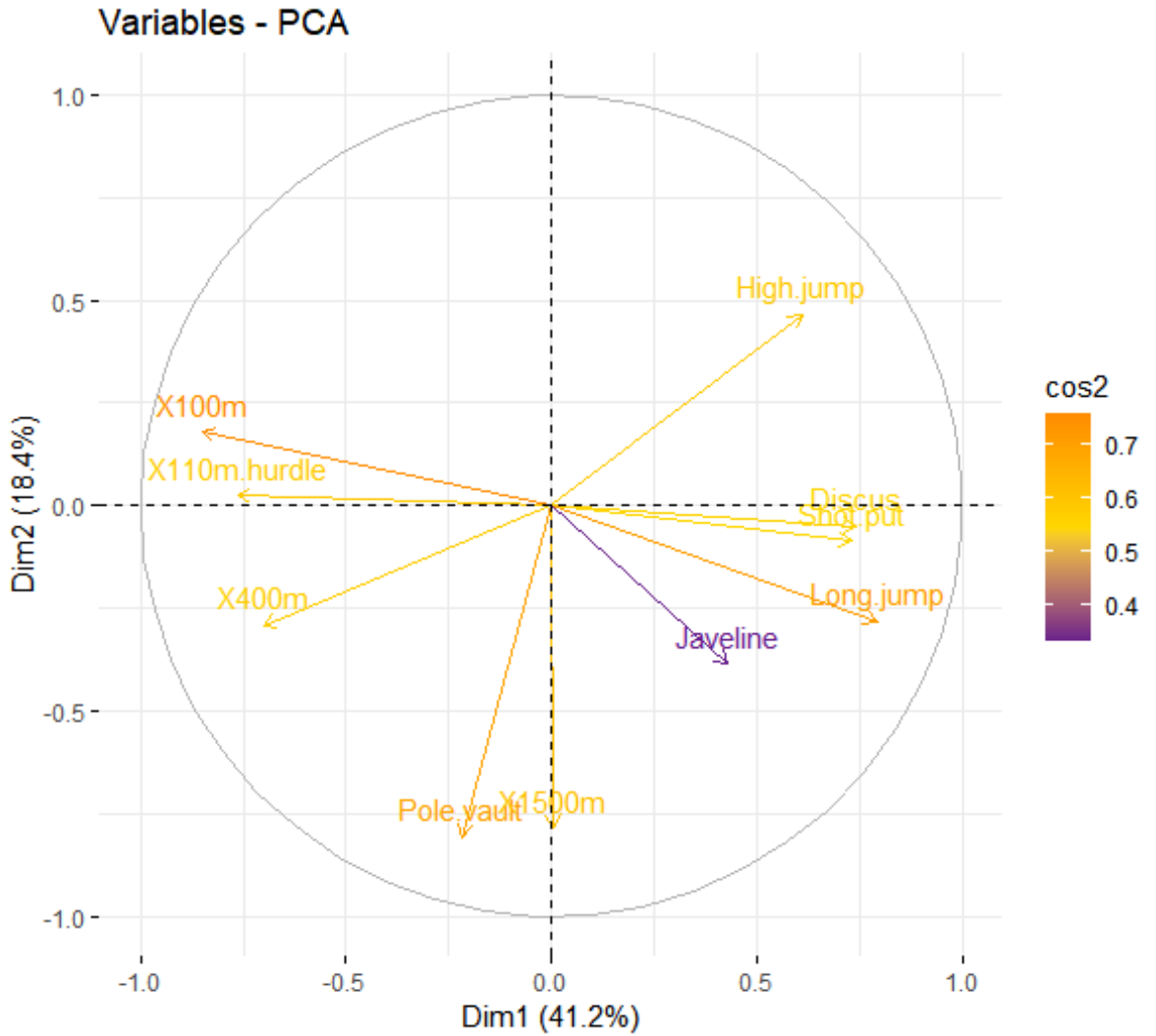
Principal Components Analysis

X100m, Long.jump and Pole.vault have high cos2 implying a good representation on the principal component. Variables are positioned close to the circumference of the correlation circle.



Principal Components Analysis

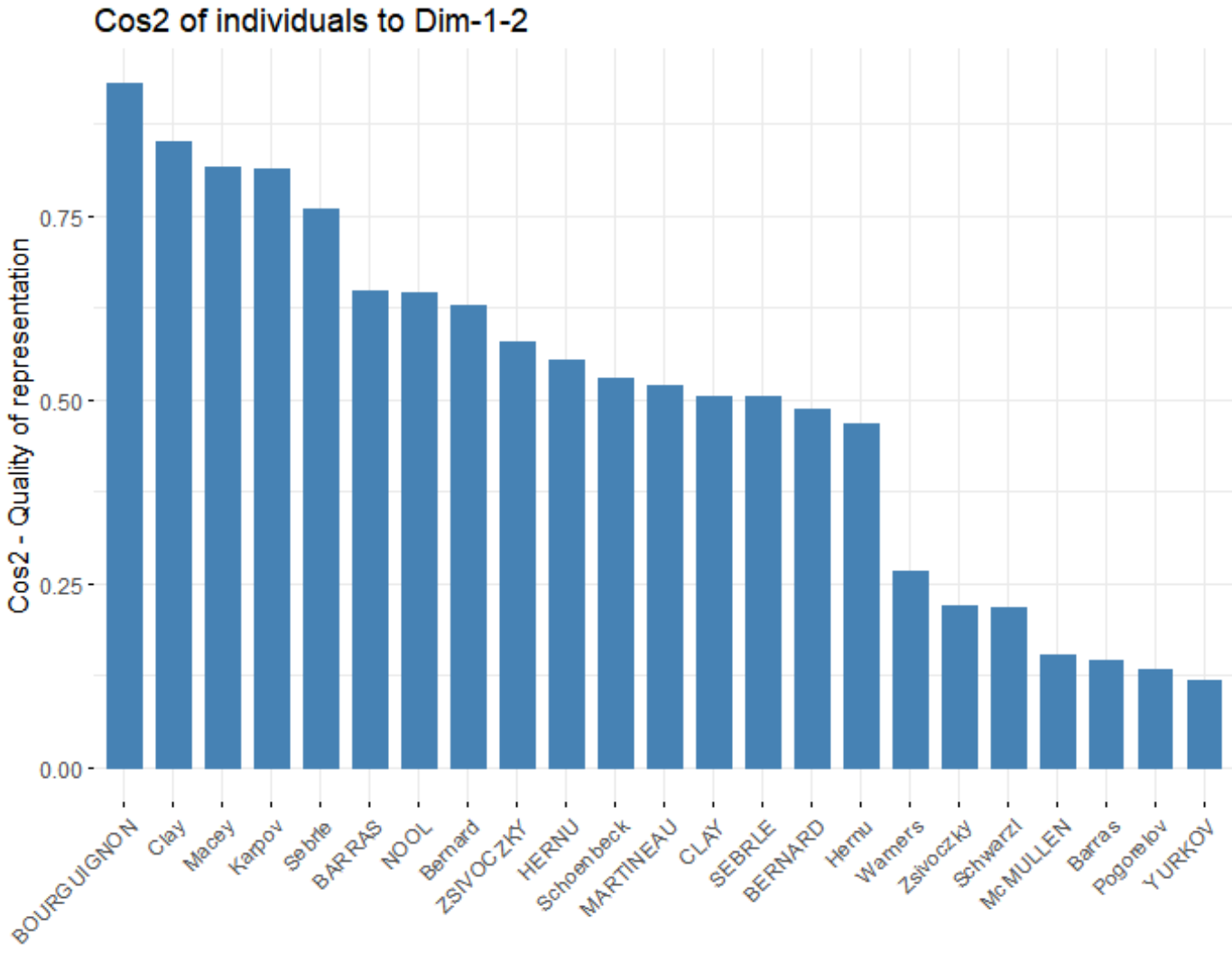
Javeline has the lowest cos2 indicating that the variable is not well represented by the PCs. The variable is close to the center of the circle, so it is less important for the first components.



Principal Components Analysis

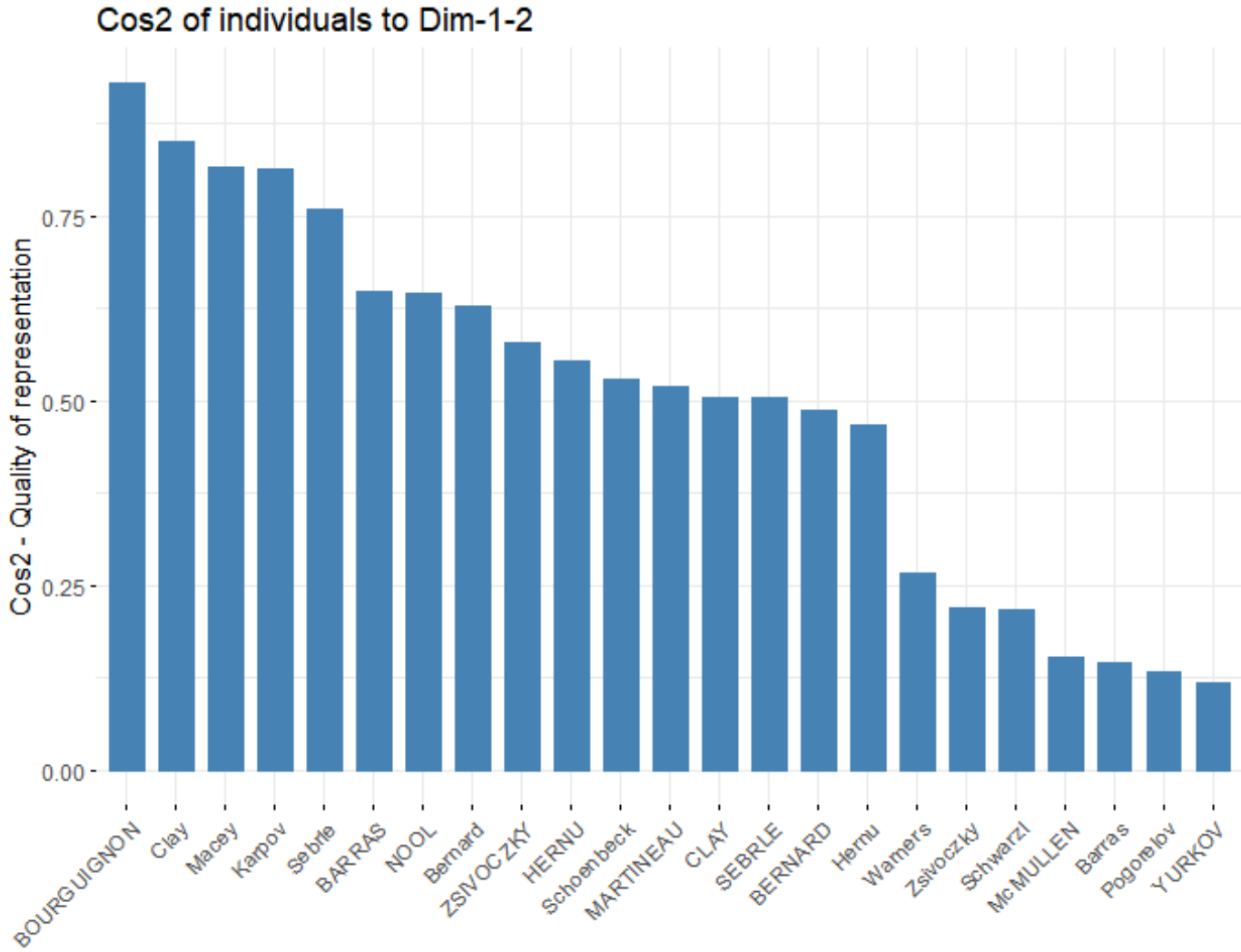
We now similarly assess PCA results with regard to individuals using cos2.

```
> fviz_cos2(res.pca, choice = "ind", axes = 1:2)
```



Principal Components Analysis

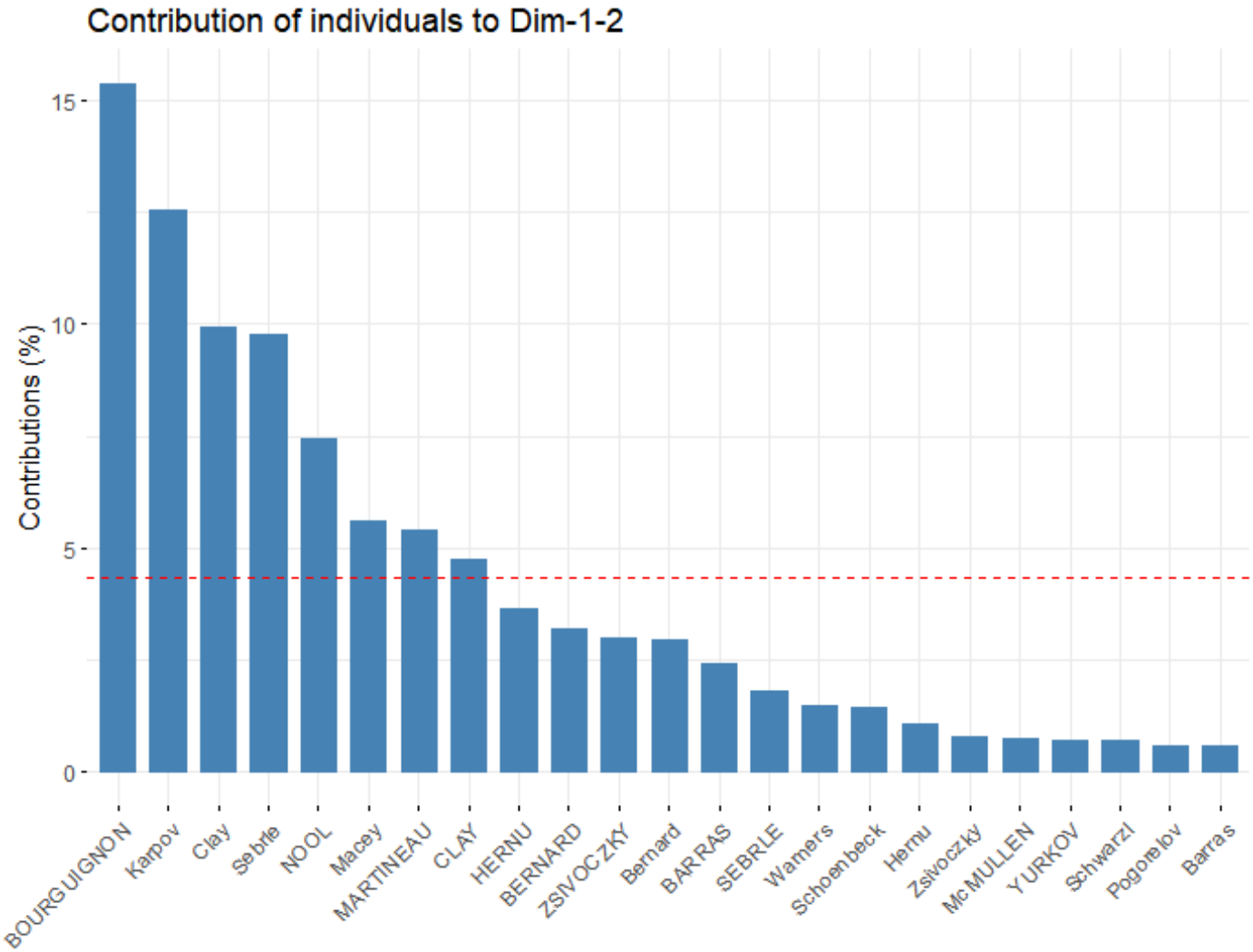
The plot shows the athletes on the left are well represented in PC1, PC2.



Principal Components Analysis

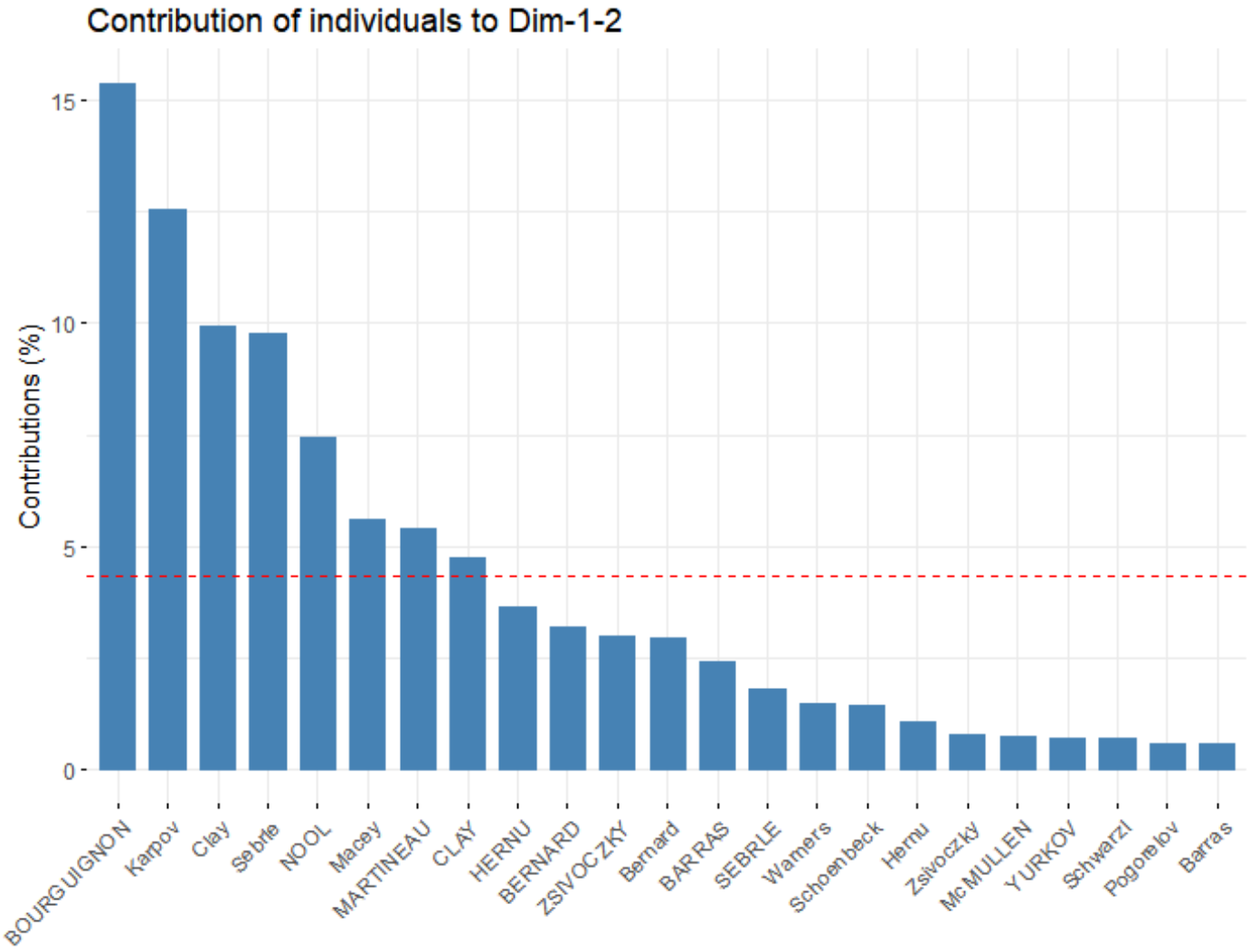
We next assess PCA results for individuals using contrib.

```
> fviz_contrib(res.pca, choice = "ind", axes = 1:2)
```



Principal Components Analysis

The plot shows the athletes contributing the most to PC1, PC2.



Principal Components Analysis

Here is **another example** of PCA.

The `tidyverse` package contains several useful functions for visualizing and manipulating data

We consider the `USArrests` dataset built into R, which contains the number of arrests per 100,000 residents in each U.S. state in 1973 for `Murder, Assault, and Rape`.

It also includes the percentage of the population in each state living in urban areas, `UrbanPop`.

Principal Components Analysis

```
> install.packages("tidyverse")
> library(tidyverse)
> #load data
> data("USArrests")
> head(USArrests)
```

	Murder	*Assault*	*UrbanPop*	*Rape*
Alabama	13.2	236	58	21.2
Alaska	10.0	263	48	44.5
Arizona	8.1	294	80	31.0
Arkansas	8.8	190	50	19.5
California	9.0	276	91	40.6
Colorado	7.9	204	78	38.7

Principal Components Analysis

```
> res.pca <- prcomp(USArrests, scale = TRUE)
> print(res.pca)
```

```
Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

- Table shows that the first principal component (PC1) has high values for Murder, Assault, and Rape which indicates that this principal component describes the most variation in these variables.
- It also shows that the second principal component (PC2) has a high value for UrbanPop, which indicates that this principle component places most of its emphasis on urban population.

Principal Components Analysis

About 87% of information is accounted by PC1 and PC2

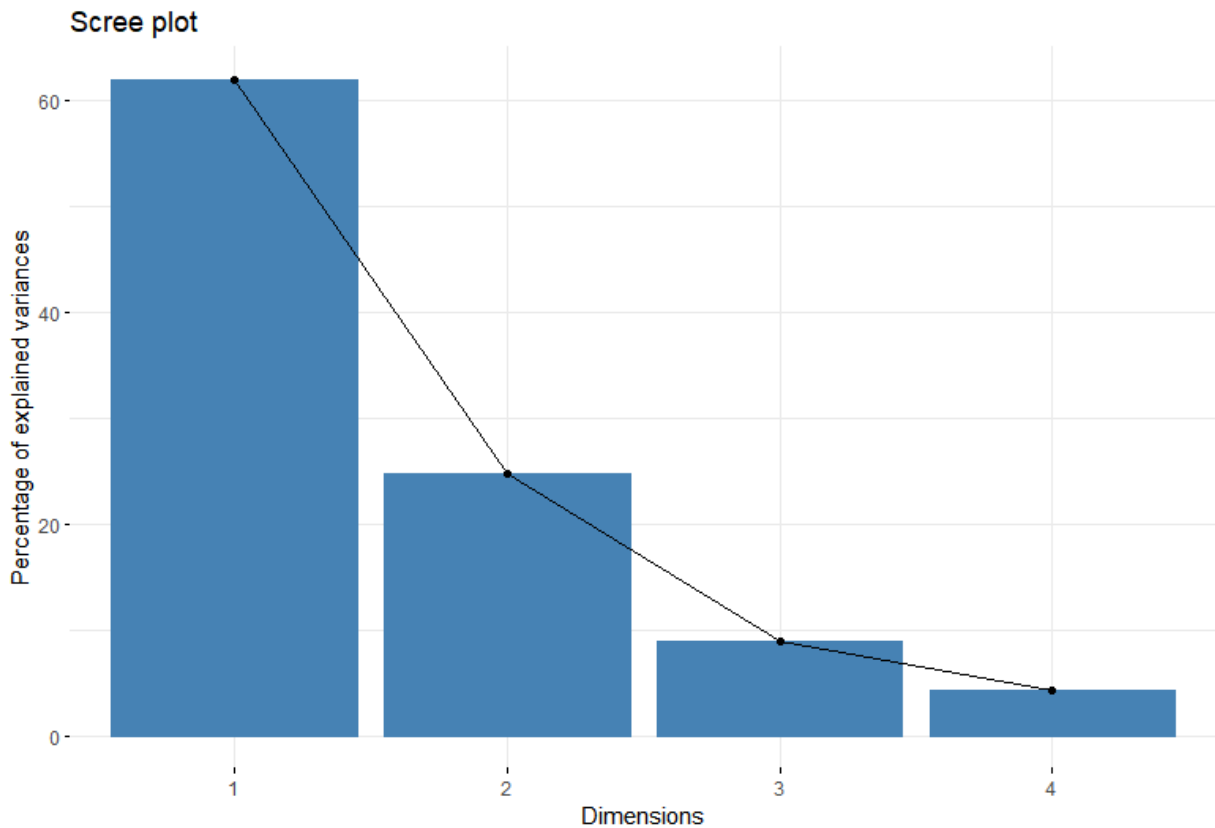
```
> summary(res.pca)
```

```
Importance of components:
                          PC1     PC2     PC3     PC4
Standard deviation     1.5749 0.9949 0.59713 0.41645
Proportion of Variance 0.6201 0.2474 0.08914 0.04336
Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

Principal Components Analysis

Here is the scree plot

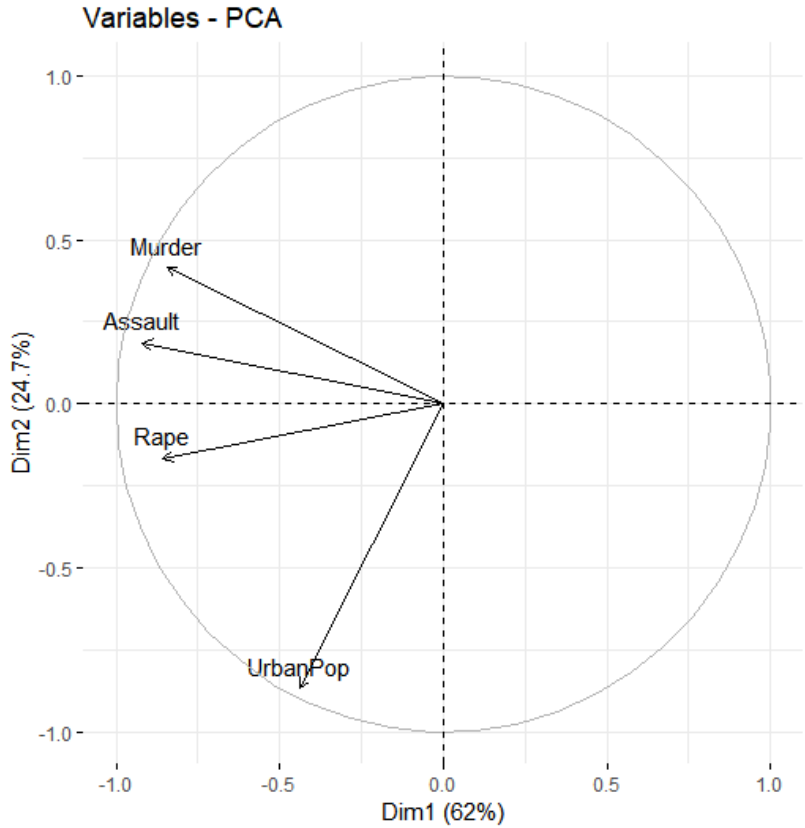
```
> library("factoextra")
> eig.val<-get_eigenvalue(res.pca)
> fviz_eig(res.pca, col.var="blue")
```



Principal Components Analysis

The biplot visualizes similarities/dissimilarities between variables.

```
> fviz_pca_var(res.pca, col.var = "black")
```

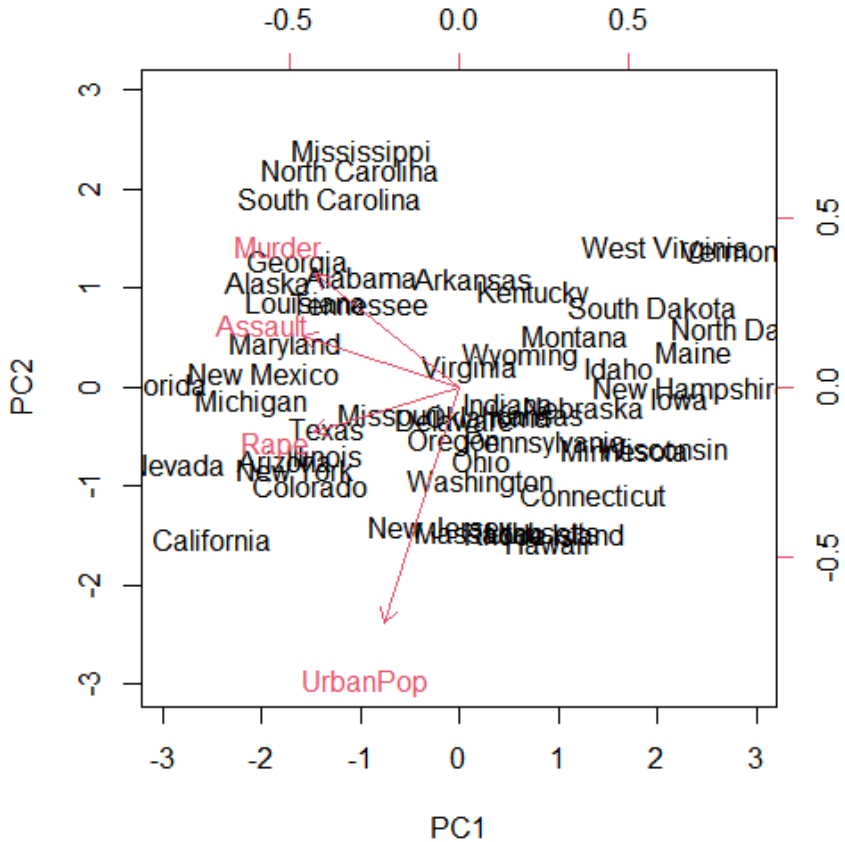


Murder, Assault, and Rape point in similar directions.

Principal Components Analysis

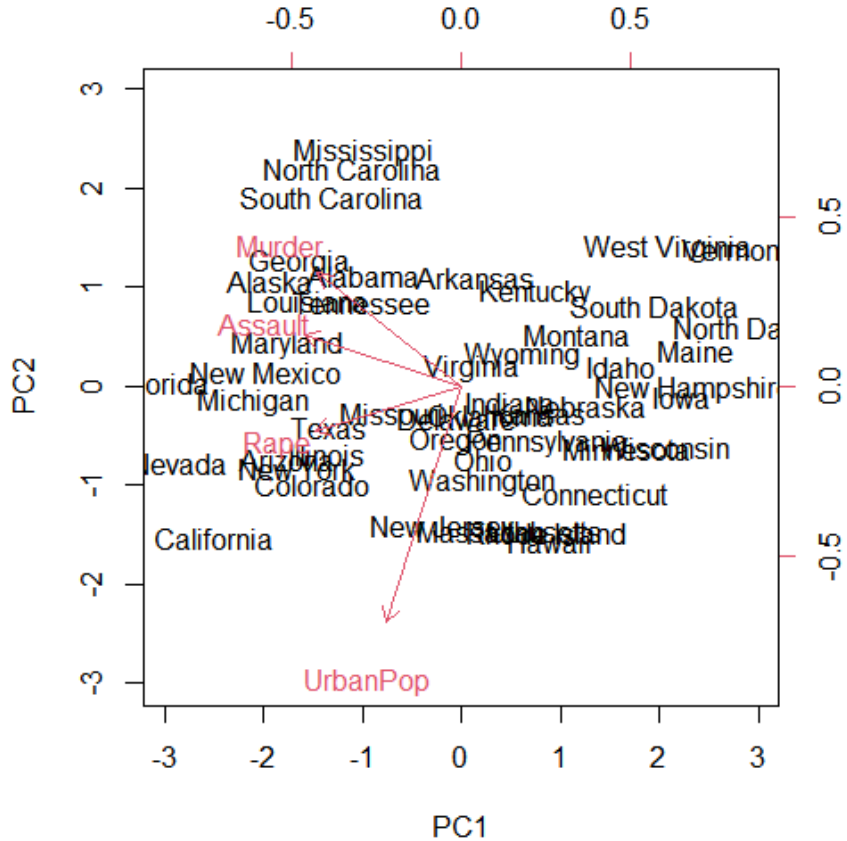
Here is another version of the biplot showing both the variables and the subjects (= states).

```
> biplot(res.pca, scale = 0)
```



Principal Components Analysis

- States close to each other on the plot have similar data patterns in regards to the variables in the original dataset.
- Certain states are more highly associated with certain crimes than others. For example, Georgia is the state closest to the variable Murder in the plot.



Principal Components Analysis

The following example uses data from the paper

"Single-Cell RNA-Seq Reveals Lineage and X Chromosome Dynamics in Human Preimplantation Embryos"

In the preimplantation embryos cells transform from being one type to become three types at day 5.

The experiment examines around 100 specific gene for each cell type

There are some genes are associated with a specific type of cell and these genes could act as marker to identify these cell types.

PCA is used to aggregate cells with similar genes profiles

https://bioinfo4all.wordpress.com/2021/01/31/tutorial-6-how-to-do-principal-component-analysis-pca-in-r/

Linear Discrimination Analysis

Linear Discrimination Analysis (LDA)

Linear Discrimination Analysis (LDA), which is also called **Canonical Variates Analysis (CVA)** is a technique to reduce data dimensionality in the spirit of PCA

Similar to PCS, LDA is also used for analyzing group structure in multivariate data.

The analogues of the Principal Components, here are the Canonical Variate axes.

These axes are directions in multivariate space that maximally separate (discriminate) the pre-defined groups of interest specified in the data.

Unlike PCA, **canonical variate axes are not, in general, orthogonal** in the space of the original variables.

https://bio723-class.github.io/Bio723-book/canonical-variates-analysis.html

Resampling Methods

Resampling Methods

Suppose we have a data set and believe it shows a non-random pattern, so that we want to infer the mean or the variance and to establish appropriate confidence intervals.

Recall: **Central Limit Theorem (CLT)**

In a nutshell, the CLT states that:

For a population with mean μ and standard deviation σ, we can estimate the population mean using **random sampling**.

If the sample size $N > 30$, the sampling distribution of the mean will be approximately normal with mean μ.
The standard error of the sample mean is $\sigma/\sqrt{N}$

Confidence Intervals: This method allows us to determine the level of confidence that the true population mean lies within an interval, using either the z-statistic or t-statistic.

Resampling Methods

Suppose that you only have a sample with no idea about the underlying population.

How can we understand the structure of the data?

Resampling methods explore the data by sampling from the sample itself (not from the population) and can be used to create empirical sampling distributions to test statistical hypotheses, estimate standard errors and construct confidence intervals.

Key idea: *resamples from the sample itself, treating it as the estimated population.*

These methods are nonparametric, meaning they don't make distribution assumptions like normality.

Popular resampling methods include:
randomization, jackknife, and bootstrap.

Randomization Tests

Randomization tests, introduced by R. A. Fisher (1935), use resampling techniques to construct a sampling distribution that can be used to make inferences about the population. What makes a randomization distribution different from other resampling methods is that it is constructed under the assumption that the null hypothesis is true.

Example. Suppose the height and weight are measured on a sample of males and females, and we wish to test whether a height-weight score S, say

$$S = \text{Height}/\text{Weight}^{1/3}$$

is different in males vs. females.

Further suppose that from a plot of S, we clearly see that the data are highly non-normal and also find that no data transformations appear to cleanly normalized the data.
Thus, t or z tests are inappropriate to assess significance.

Randomization Tests

We can apply a randomization test by computing S for each individual and then shuffling the S values random over gender. We use a randomization test to compare the observed difference in the means to the distribution of differences that we would expect to observe if the labels male and female were randomly applied to samples of equal size from the data at hand.

Suppose that there are N_m males and N_f females.
Drawing (without replacement) $N_m S$ values from the original sample and assigning them as males, and the remainder as females, generates a randomized sample.

Randomization Tests

Suppose that, in the original sample, the mean value of S in males was 10.2 units larger than the mean value for females.
In a resampling distribution with 2000 values, 12 show male minus female differences of 10.2 or greater and 17 show male minus females differences of -10.2 or less.

Thus under the one-sided tests that males are larger than females, randomization estimates the probability under the null hypothesis as $12/2000 = 0.006$. Under a two-sided test, the probability is $0.006 + 17/2000 = 0.0145$.

A critical question in this method is how many randomized samples one should generate. The general consensus is around 1000 samples for tests at the 5% level and 5000 for tests at the 1% level

Bootstrap

Bootstrap approaches (originally introduced by Efron in 1979) attempt to estimate the sampling distribution of a population by generating new samples by drawing (with replacement) from the original data.

Bootstrap Process in a Nutshell

1. Draw random samples from the sample itself (estimated population).
2. Resampling is allowed, so an observation may appear multiple times in the same sample.
3. Repeat the process thousands of times to build an actual sampling distribution.
4. Calculate the mean and standard errors of the estimator.
5. Determine the confidence interval for the population mean directly from the sampling distribution.

Jackknife

The **jackknife** method is another resampling technique that was popular in the past when computers were less advanced.

Unlike bootstrap, which repeatedly draws samples of the same size with replacement, jackknife samples are selected by taking the original observed data sample and leaving out one observation at a time without replacement.

For an estimated population of size M, jackknife usually requires M repetitions, whereas bootstrap can have many more resamples as repetition is allowed.

Like bootstrap, the statistic for each resample is calculated, and the data forms the sampling distribution, which determines the confidence intervals for the population statistic.

Resampling Methods

Jackknife vs. Bootstrap

- Jackknife is less computationally intensive, making it popular in the past when computers were less advanced.
- Bootstrap allows for more resamples as repetition is allowed, resulting in a more robust sampling distribution.
- Jackknife produces the same results for every run, as the resampling process is exhaustive.
- Bootstrap usually gives different results, as resamples are randomly drawn.

Randomized test vs. Bootstrap

- Like bootstrapping procedures, randomization procedures use resampling techniques to construct a sampling distribution that can be used to make inferences about the population. What makes a randomization distribution different is that it is constructed given that the null hypothesis is true.

Resampling Methods using R

https://bio723-class.github.io/Bio723-book/randomization-jackknife-and-bootstrap.html

https://bookdown.org/jgscott/DSGI/the-bootstrap.html