

SLLD - Module 1

PCA

S.Tonini, F.Chiaromonte

Sant'Anna School of Advanced Study - Pisa

8/2/2024

Libraries

```
library(mvtnorm)    # for the toy simulated example
library(factoextra) # contains also decathlon data
library(corrplot)   # correlation plots
```

Data

We use both simulated data from a multivariate normal distribution and real data

```
Sigma.x1=diag(4)
mu=c(1,1,1,1)
set.seed(1)
x1=rmvnorm(50, mu, Sigma.x1)

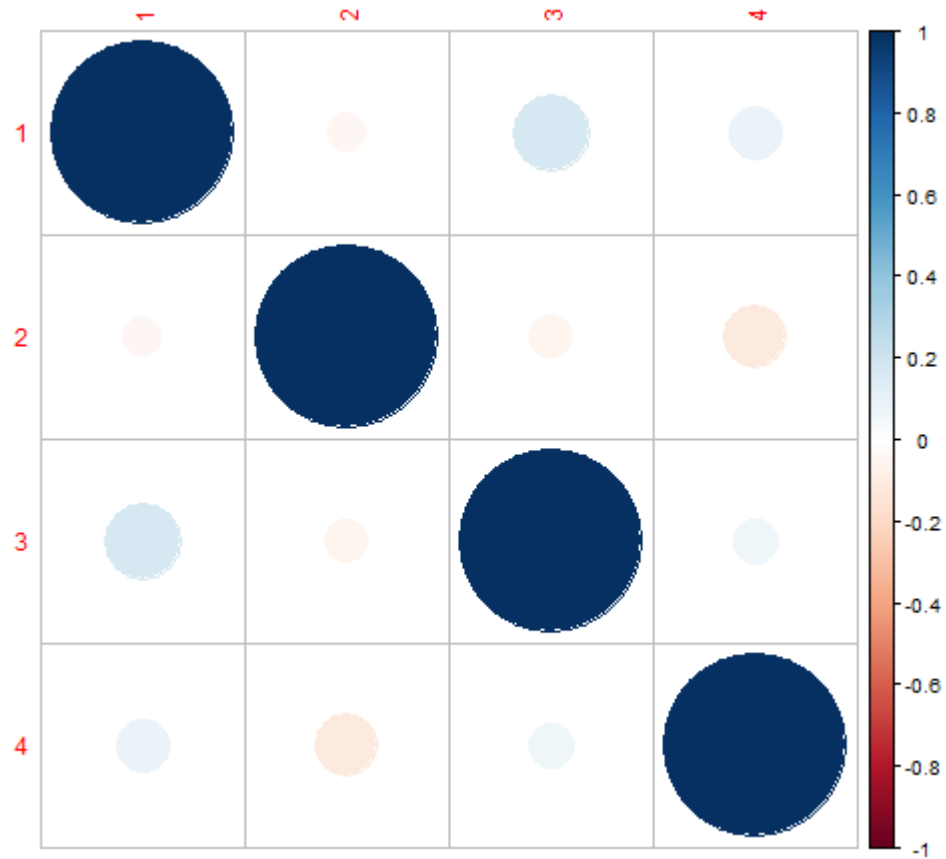
Sigma.x2=diag(4)
for(i in 1:nrow(Sigma.x2)) {for(j in 1:ncol(Sigma.x2)) {
  Sigma.x2[[i,j]] = 0.9^(abs(i-j)) }}
mu1=c(1,1,1,1)
set.seed(1)
x2=rmvnorm(50, mu1, Sigma.x2)

data("USArrests")
x3=USArrests
dim(x3)
```

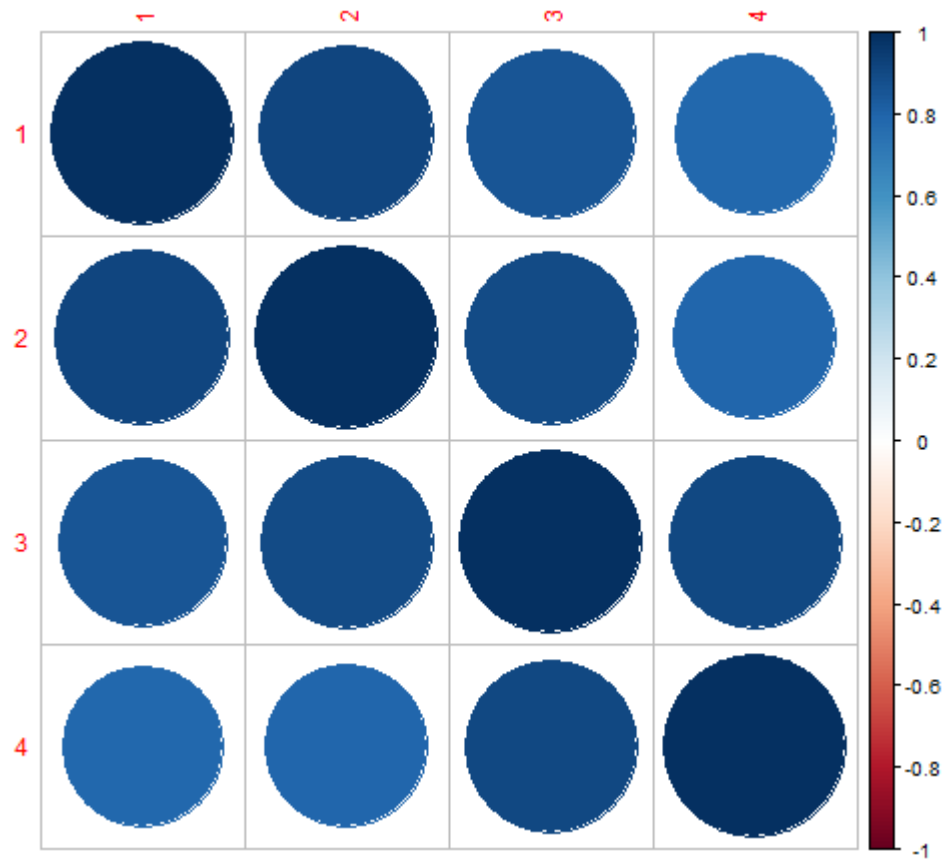
```
## [1] 50 4
```

We can use the function **corrplot** to display the correlation matrices

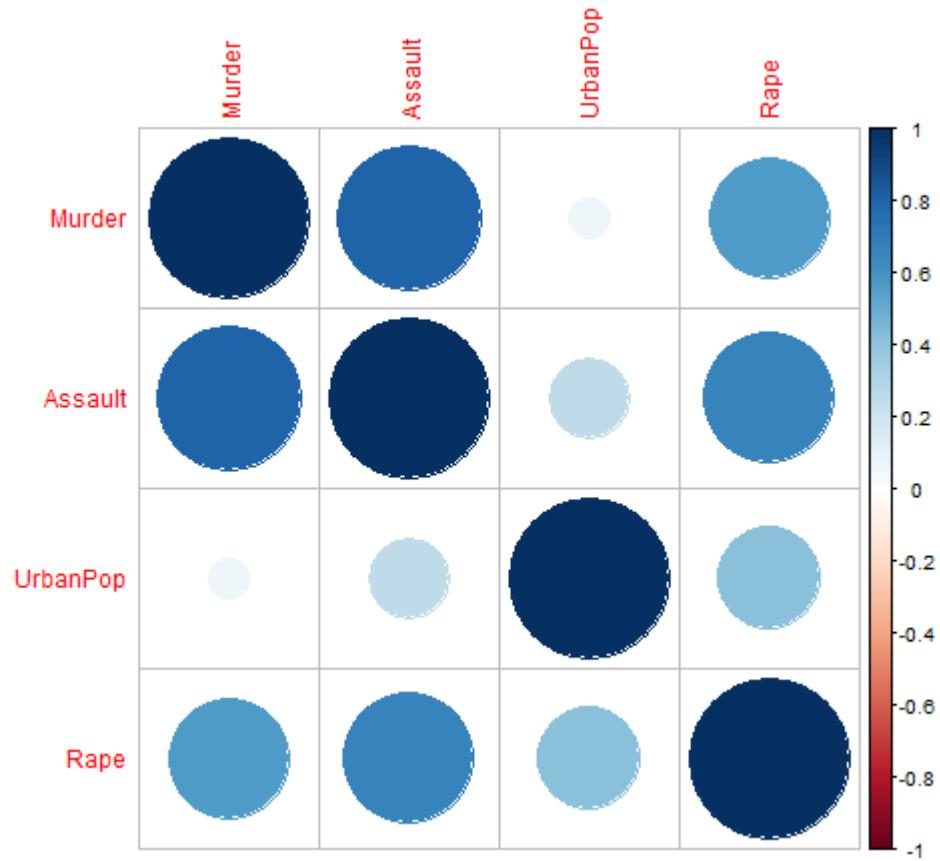
```
corrplot(cor(x1))
```



```
corrplot(cor(x2))
```



```
corrplot(cor(x3))
```



To perform principal components analysis we use the **prcomp()** function. It requires:

- **data:** a data frame
- **scale:** a logical value (TRUE/FALSE)

```
PCA.1 <- prcomp(x1, scale = TRUE)
PCA.2 <- prcomp(x2, scale = TRUE)
PCA.3 <- prcomp(x3, scale = TRUE)
str(PCA.1) # a diagnostic function
```

```
## List of 5
## $ sdev      : num [1:4] 1.131 1.009 0.94 0.904
## $ rotation: num [1:4, 1:4] 0.559 -0.408 0.545 0.474 0.426 ...
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : NULL
## .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
## $ center   : num [1:4] 0.959 1.038 1.184 0.96
## $ scale    : num [1:4] 0.811 0.988 1.006 0.908
## $ x        : num [1:50, 1:4] -0.163 1.179 1.51 1.036 0.318 ...
## ..- attr(*, "dimnames")=List of 2
## .. ..$ : NULL
## .. ..$ : chr [1:4] "PC1" "PC2" "PC3" "PC4"
## - attr(*, "class")= chr "prcomp"
```

The result is a list containing 5 elements: **sdev**: the standard deviations of the principal components (square root of the eigenvalues) **rotation**: the matrix of variable loadings (eigenvectors) **center**: the centering used if *scale=TRUE* **scale**: the scaling used if *scale=TRUE* **x**: the scores, i.e. the rotated data

Selecting the number of components

The selection of a "suitable" number of components is non-trivial since in unsupervised learning tasks no ground-truth/labels/response variable is available (which may be used to assess and validate our results otherwise, e.g. through cross-validation -- more on this later).

However, some "rules of thumb" have been developed. For instance, one may consider the **percentage of variance explained (PVE)**, or the **cumulative PVE**. Here we use the **get_eig** function of the **factoextra** package.

```
get_eig(PCA.1)
```

##	eigenvalue	variance.percent	cumulative.variance.percent
## Dim.1	1.2800721	32.00180	32.00180
## Dim.2	1.0185759	25.46440	57.46620
## Dim.3	0.8835112	22.08778	79.55398
## Dim.4	0.8178408	20.44602	100.00000

```
get_eig(PCA.2)
```

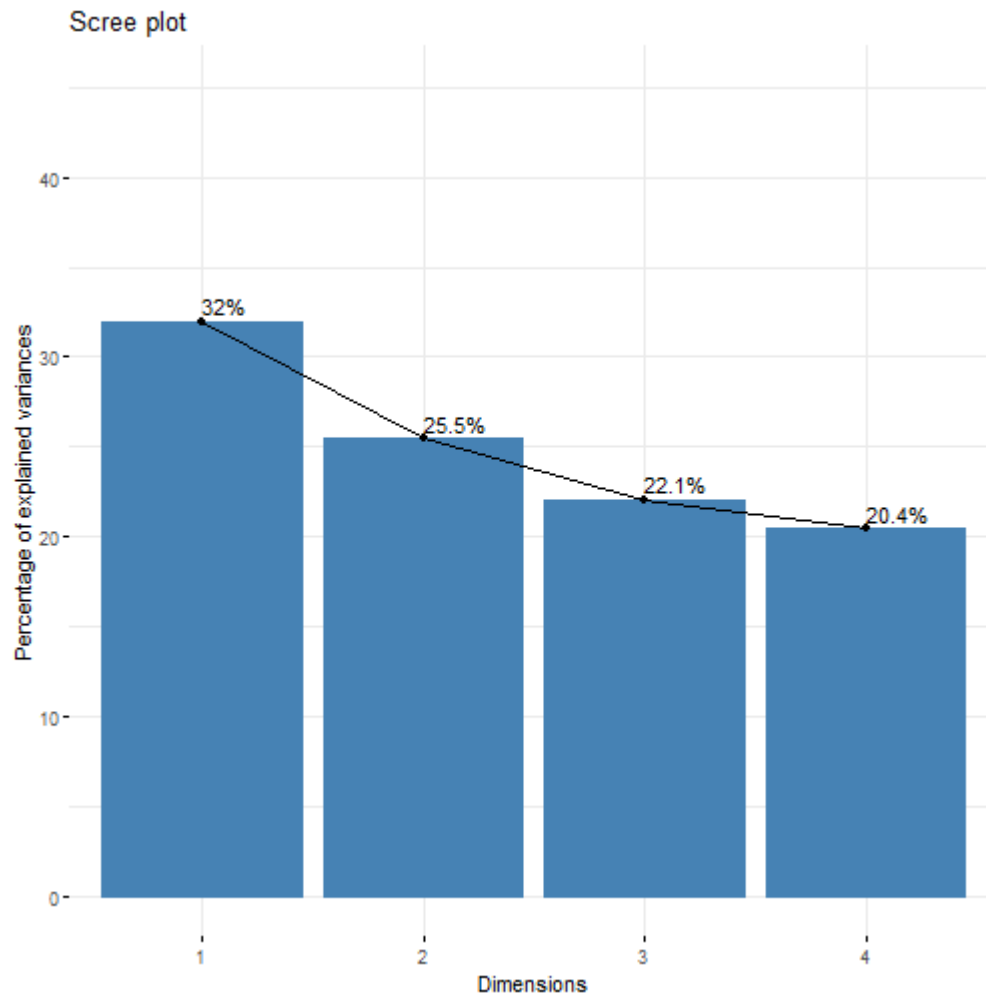
##	eigenvalue	variance.percent	cumulative.variance.percent
## Dim.1	3.57835025	89.458756	89.45876
## Dim.2	0.26573517	6.643379	96.10214
## Dim.3	0.09266963	2.316741	98.41888
## Dim.4	0.06324494	1.581124	100.00000

```
get_eig(PCA.3)
```

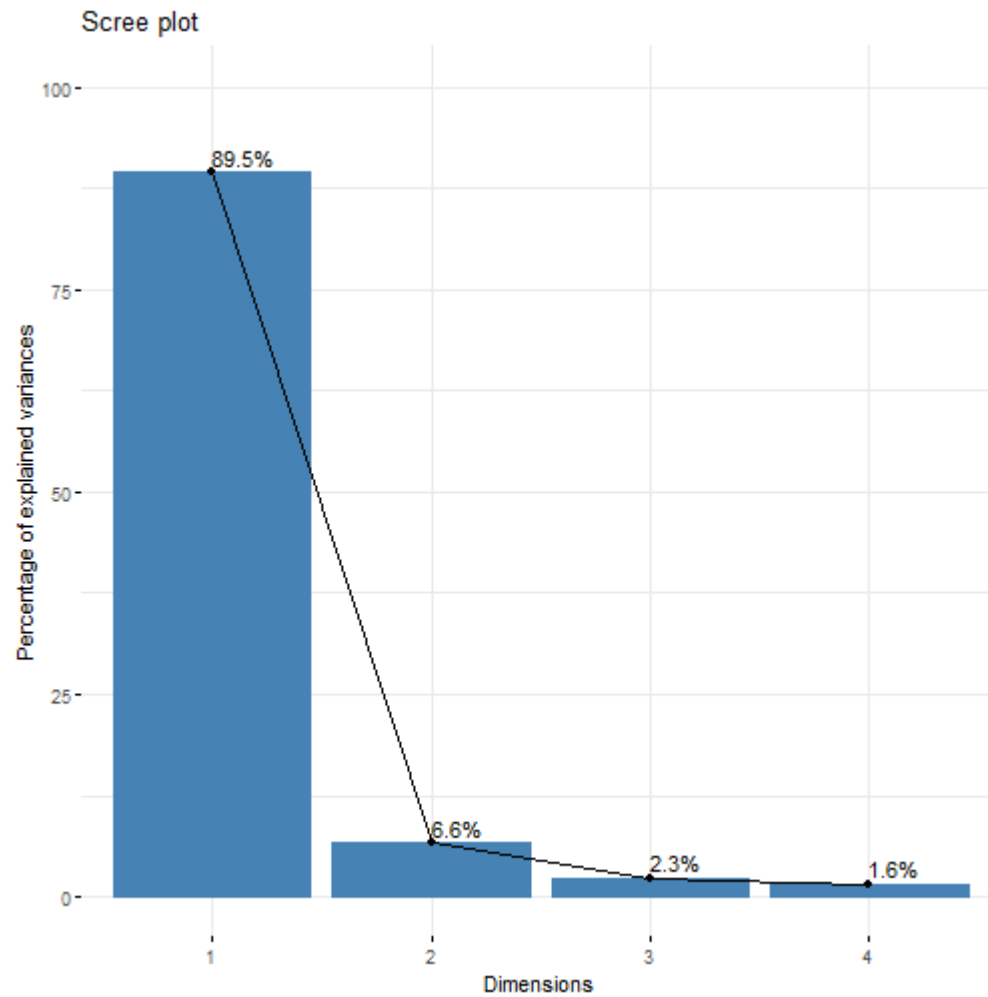
##	eigenvalue	variance.percent	cumulative.variance.percent
## Dim.1	2.4802416	62.006039	62.00604
## Dim.2	0.9897652	24.744129	86.75017
## Dim.3	0.3565632	8.914080	95.66425
## Dim.4	0.1734301	4.335752	100.00000

Based on this information we can create a **scree plot**, and try to find an **elbow** (i.e. an inflection point) therein.

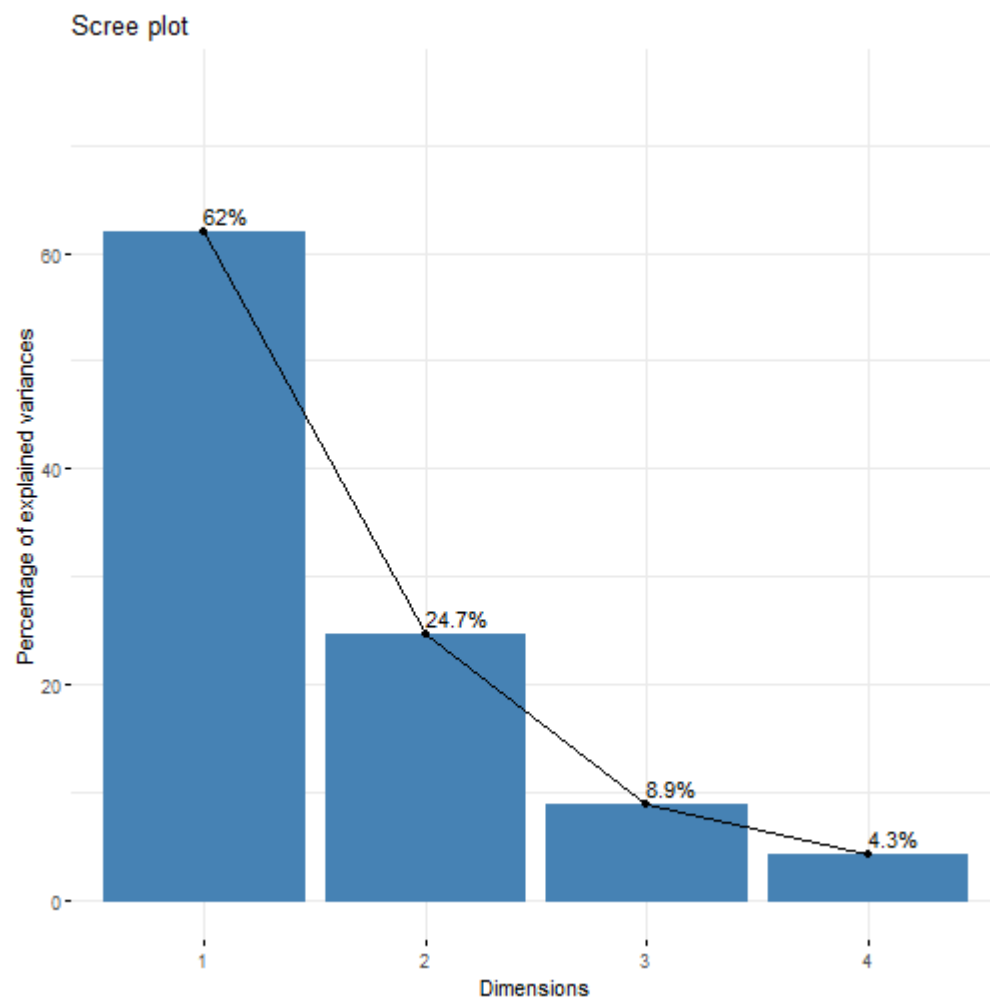
```
fviz_eig(PCA.1, addlabels = TRUE, ylim = c(0, 45))
```



```
fviz_eig(PCA.2, addlabels = TRUE, ylim = c(0, 100))
```

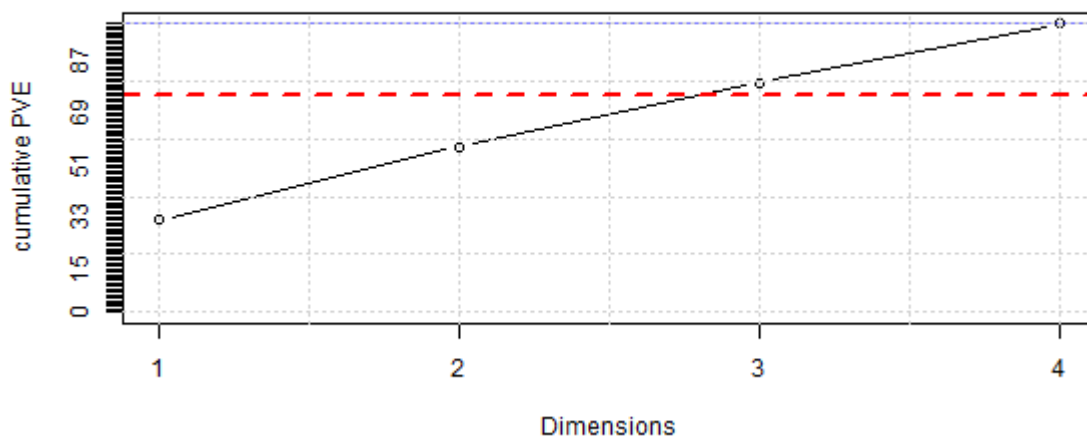


```
fviz_eig(PCA.3, addlabels = TRUE, ylim = c(0, 75))
```



Sometimes, it is not easy to identify an elbow, or this might be associated to a particularly low cumulative PVE. Thus, it is common to fix an **acceptance threshold** a priori (e.g. 75%), and then look at the **cumulative PVE**.

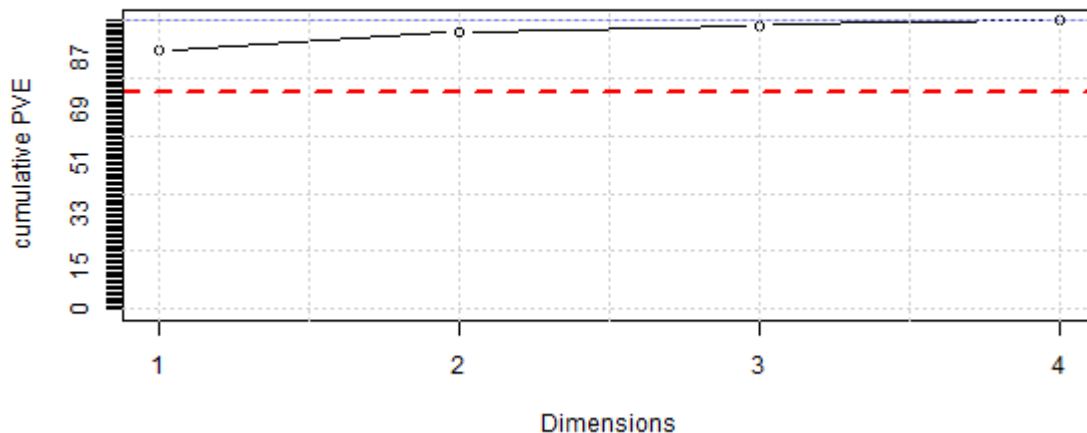
```
plot(get_eig(PCA.1)$cumulative.variance.percent,  
     type='b', axes=F, xlab='Dimensions', ylab='cumulative PVE',  
     ylim=c(0,100))  
abline(h=100, col=alpha('blue',0.5))  
abline(h=75, lty=2, col='red', lwd=2) # thresholding  
box()  
axis(2, at=0:100, labels=0:100)  
axis(1, at=1:ncol(decathlon2), labels=1:ncol(decathlon2))  
grid()
```



```

plot(get_eig(PCA.2)$cumulative.variance.percent,
     type='b', axes=F, xlab='Dimensions', ylab='cumulative PVE',
     ylim=c(0,100))
abline(h=100, col=alpha('blue',0.5))
abline(h=75, lty=2, col='red', lwd=2) # thresholding
box()
axis(2, at=0:100, labels=0:100)
axis(1, at=1:ncol(decathlon2), labels=1:ncol(decathlon2))
grid()

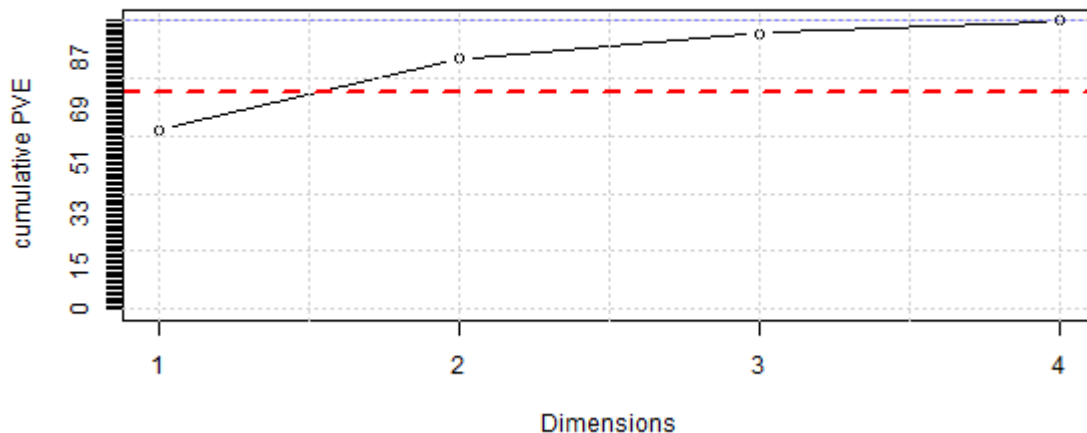
```



```

plot(get_eig(PCA.3)$cumulative.variance.percent,
     type='b', axes=F, xlab='Dimensions', ylab='cumulative PVE',
     ylim=c(0,100))
abline(h=100, col=alpha('blue',0.5))
abline(h=75, lty=2, col='red', lwd=2) # thresholding
box()
axis(2, at=0:100, labels=0:100)
axis(1, at=1:ncol(decathlon2), labels=1:ncol(decathlon2))
grid()

```



Loadings interpretation

Let's focus on the loadings, i.e., the eigenvectors representing the directions of the PCs.

We can plot the first two PCs (PC1 and PC2) in the **graph of variables** (also called correlation circle). To this end, we use the function **fviz_pca_var**

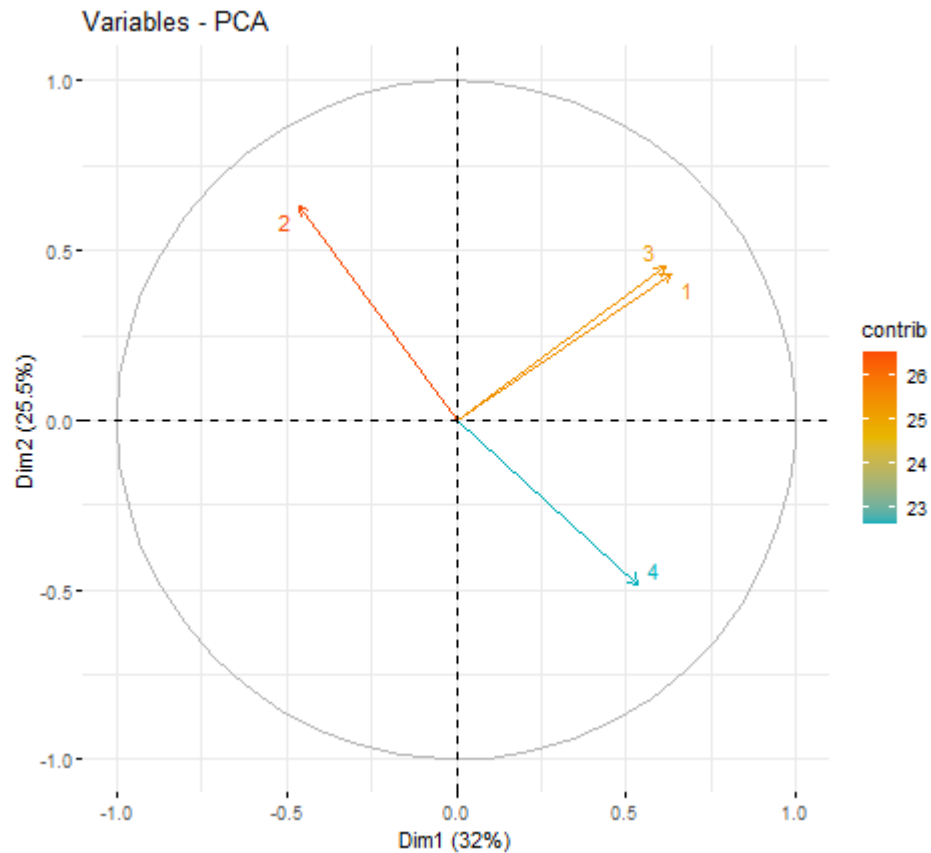
Here the importance of the original features is represented by the **color code**:

- red: high
- blue: medium
- white: low

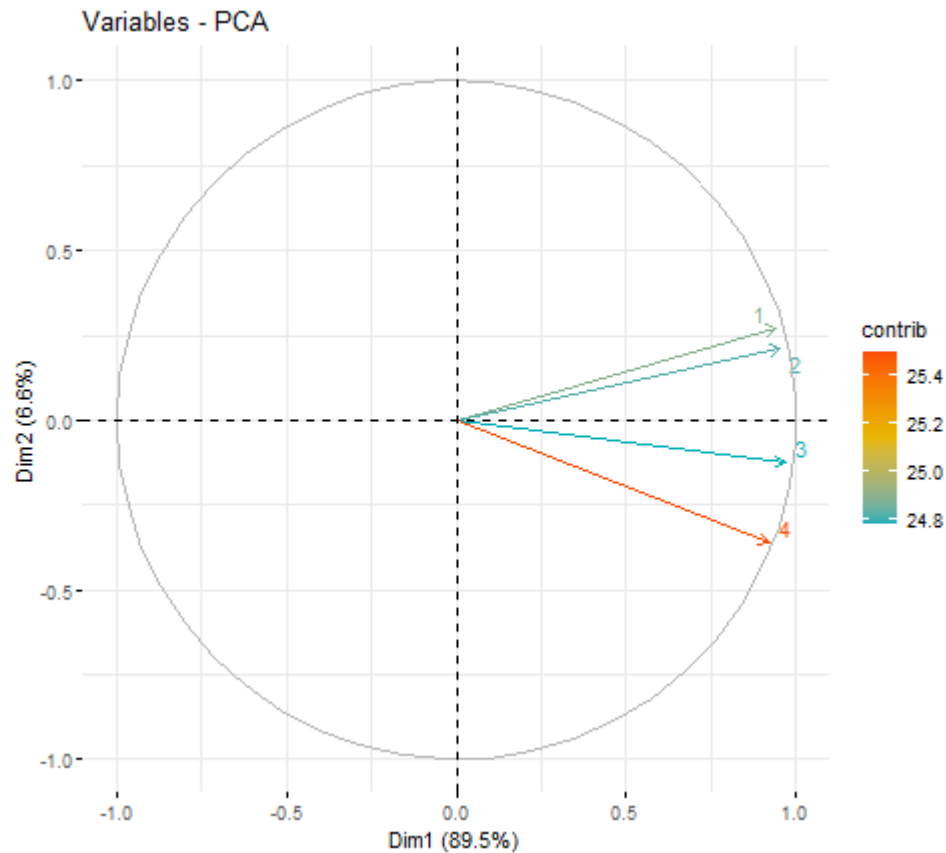
and by the length of the vector (its **closeness** to the circle).

In these plots, positively correlated variables have same direction (they are close to each other, i.e. they have a small angle in between). Negatively correlated variables have opposite directions. Uncorrelated variables are orthogonal.

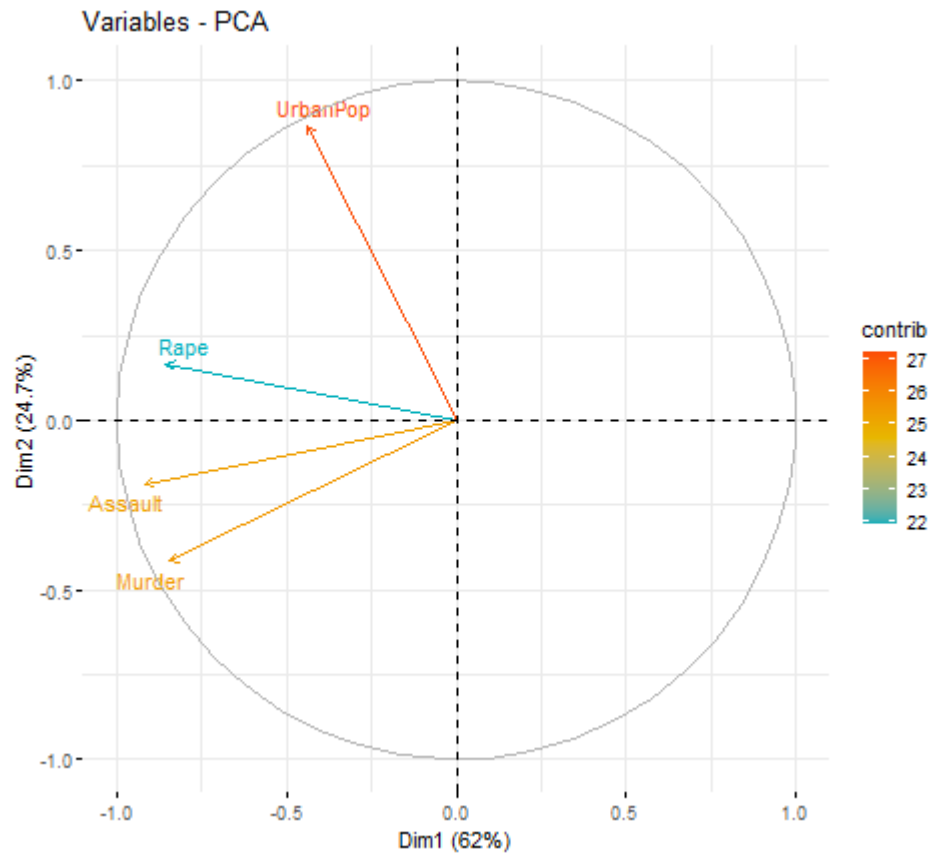
```
fviz_pca_var(PCA.1,  
  col.var = "contrib", # Color by contributions to the PC  
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
  repel = TRUE )      # Avoid text overlapping
```




```
fviz_pca_var(PCA.2,  
  col.var = "contrib", # Color by contributions to the PC  
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
  repel = TRUE )      # Avoid text overlapping
```

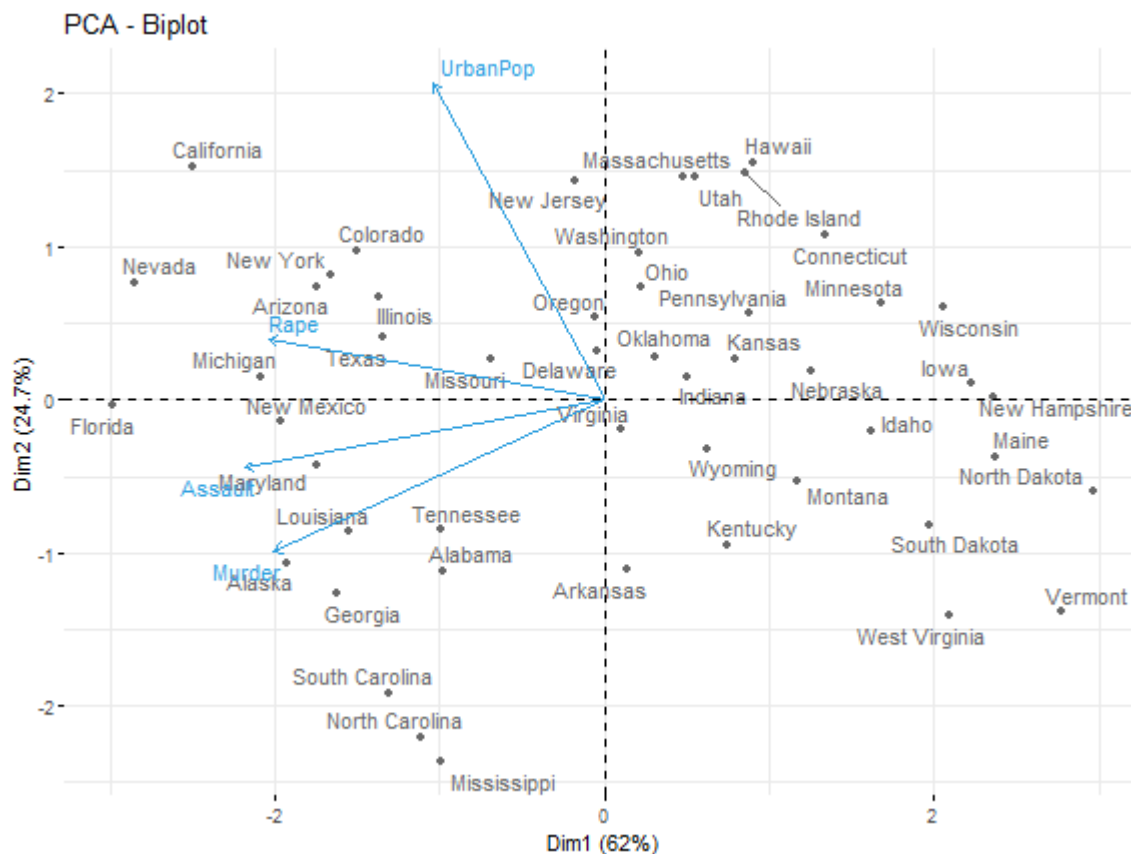


```
fviz_pca_var(PCA.3,  
  col.var = "contrib", # Color by contributions to the PC  
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),  
  repel = TRUE )      # Avoid text overlapping
```



If we want to show also the individuals we can run the **biplot of individuals and variables**.

```
fviz_pca_biplot(PCA.3, repel = TRUE,  
                 col.var = "#2E9FDF", # Variables color  
                 col.ind = "#696969"  # Individuals color  
)
```



Now it's your turn!!!