Data Set USArrests

Podemos ver los rangos entre las variables con `summary`:

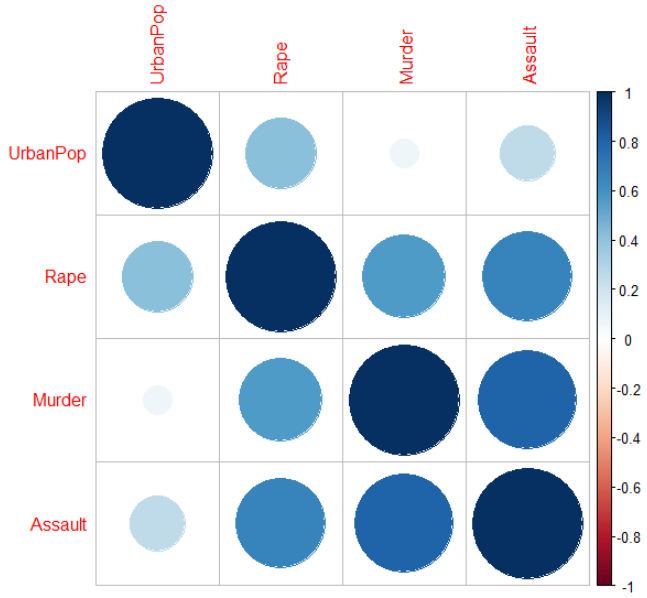
```
> summary(USArrests)
     Murder          Assault         UrbanPop          Rape
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
```

Mostramos las columnas y valores que tenemos, para tener idea de como son los datos con los que estamos trabajando.

```
> glimpse(USArrests)
Rows: 50
Columns: 4
$ Murder   <dbl> 13.2, 10.0, 8.1, 8.8, 9.0, 7.9, 3.3, 5.9, 15.4, 17.4, 5.3, 2.6, 10.4, 7.2,
$ Assault  <int> 236, 263, 294, 190, 276, 204, 110, 238, 335, 211, 46, 120, 249, 113, 56, 11
$ UrbanPop <int> 58, 48, 80, 50, 91, 78, 77, 72, 80, 60, 83, 54, 83, 65, 57, 66, 52, 66, 51,
$ Rape     <dbl> 21.2, 44.5, 31.0, 19.5, 40.6, 38.7, 11.1, 15.8, 31.9, 25.8, 20.2, 14.2, 24.
```

Analisamos las correlaciones entre las variables para identificar las principales.

```
> corrplot(cor(USArrests), order = "hclust")
```



Analizando el resultado la variable siempre estará colacionada consigo misma, pero hablando de algo menos obvio podemos ver que que el asalto y el asesinato están muy relacionadas al igual que el asesinato y la violación, por otro lado el UrbanPop esta muy poco relacionado con el asesinato. Todas las variables son importantes o principales y representativas.

Al hacer `glimpse(USArrests)` tiene 50 observaciones cada variable y no hay valores nulos.

kmeans(df, 4, iter.max=1000,nstart=25)

Al utilizar este comando tendríamos un nuevo clustering obtenido con K-medias, utilizando el mismo de datos `df`, antes definido, con 4 centroides, un maximo de 1000 iteraciones para ajustarlos y con 25 particiones aleatorias iniciales.

Al aumentar o disminuir el número de centroides en el método anterior cambia el número de clusters, además de como se distribuyen las ciudades entre ellos pues se hace un nuevo calculo e distancias.

print(km.res)

Este comando nos da el resultado de haber utilizado kmeans, mostrándonos las medias, el vector de clustering, es decir, en que cluster se encuentra cada dato, la suma de cuadrados por grupo y los componentes.