PROYECTO CLUSTERING

Federico Álvarez-Labrador

1. INTRODUCCIÓN Y DESGLOSE DE LIBRERÍAS

El dataset analizado en la presente práctica recoge los datos de 7 variables que corresponden a indicadores de criminalidad por 10.000 habitantes para cada uno de los 50 estados de Estados Unidos.

Los indicadores de criminalidad son los siguientes:

1. Asesinatos (variable MURDER)
2. Violaciones (RAPE)
3. Robos (ROBBERY)
4. Asaltos (ASSAULT)
5. Agresiones (BURGLARY)
6. Hurtos (LARCENY)
7. Robos de coches (AUTO_THEFT)

El objetivo de la presente práctica es realizar un análisis cluster agrupando estados que, respecto de dichas variables, tengan un comportamiento parecido.

Para la realización de esta práctica se han empleado diez paquetes de R que se enumeran a continuación:

- DPLYR - *https://cran.r-project.org/web/packages/dplyr/index.html*
- GGPLOT2 - *https://cran.r-project.org/web/packages/ggplot2/index.html*
- VIM - *https://cran.r-project.org/web/packages/VIM/index.html*
- GRIDEXTRA - *https://cran.r-project.org/web/packages/gridExtra/index.html*
- CORRPLOT - *https://cran.r-project.org/web/packages/corrplot/index.html*
- VEGAN - *https://cran.r-project.org/web/packages/vegan/index.html*
- SQLDF - *https://cran.r-project.org/web/packages/sqldf/index.html*
- FMSB - *https://cran.r-project.org/web/packages/fmsb/index.html*
- NBCLUST - *https://cran.r-project.org/web/packages/NbClust/index.html*
- FACTOEXTRA - *https://cran.r-project.org/web/packages/factoextra/index.html*

---

2. TRATAMIENTO DE VARIABLES

Empezamos cargando el dataset base (archivo “crime.csv”) incluido en la documentación correspondiente a esta práctica. Una vez cargado el dataset, lo asignamos a una variable creando un dataframe para utilizarlo según sea más conveniente para el análisis posterior:

```
crime_df <- read.csv("crime.csv", header=T, sep=" ", dec=".", stringsAsFactors=F)
```

Revisamos los datos cargados:

```
# Dimensiones del dataset
dim(crime_df)
```

```
## [1] 50  8
```

```
# Clase y muestra de cada variable
str(crime_df)
```

```
## 'data.frame':    50 obs. of  8 variables:
##  $ State     : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ Murder    : num  14.2 10.8 9.5 8.8 11.5 6.3 4.2 6 10.2 11.7 ...
##  $ Rape      : num  25.2 51.6 34.2 27.6 49.4 42 16.8 24.9 39.6 31.1 ...
##  $ Robbery   : num  96.8 96.8 138.2 83.2 287 ...
##  $ Assault   : num  278 284 312 203 358 ...
##  $ Burglary  : num  1136 1332 2346 973 2139 ...
##  $ Larceny   : num  1882 3370 4467 1862 3500 ...
##  $ Auto_Theft: num  281 753 440 183 664 ...
```

Revisados los datos y analizando el dataframe resultante parece que los mismos están organizados y se han cargado correctamente (cabecera de la database, separador de los datos, nombres y formatos de cada variable, decimales...).

Dado que las todas las variables son numéricas continuas salvo la primera que es categórica, vamos a utilizar dicha variable para nombrar las observaciones y dejar el resto de variables con los valores numéricos.

```
crime_df1 <- crime_df %>%
  select(-c("State"))

rownames(crime_df1) <- crime_df$State

# Clase y muestra de cada variable
str(crime_df1)
```

```
## 'data.frame':    50 obs. of  7 variables:
##  $ Murder    : num  14.2 10.8 9.5 8.8 11.5 6.3 4.2 6 10.2 11.7 ...
##  $ Rape      : num  25.2 51.6 34.2 27.6 49.4 42 16.8 24.9 39.6 31.1 ...
##  $ Robbery   : num  96.8 96.8 138.2 83.2 287 ...
##  $ Assault   : num  278 284 312 203 358 ...
##  $ Burglary  : num  1136 1332 2346 973 2139 ...
##  $ Larceny   : num  1882 3370 4467 1862 3500 ...
##  $ Auto_Theft: num  281 753 440 183 664 ...
```

```
# Nombres de las observaciones
rownames(crime_df1)[1:10]
```

```
##  [1] "Alabama"     "Alaska"      "Arizona"     "Arkansas"    "California"
##  [6] "Colorado"    "Connecticut" "Delaware"    "Florida"     "Georgia"
```

Una vez cargados y revisados los datos, el siguiente paso será analizar a fondo los mismos para detectar la presencia y hacer un tratamiento de los valores "missing" y "outliers".

2.1. ANÁLISIS Y TRATAMIENTO DE MISSING VALUES

El dataset contiene 50 observaciones (los 50 estados de Estados Unidos) y 8 variables (la variable "State" que identifica el estado y las otras 7 que corresponden a los indicadores de criminalidad).

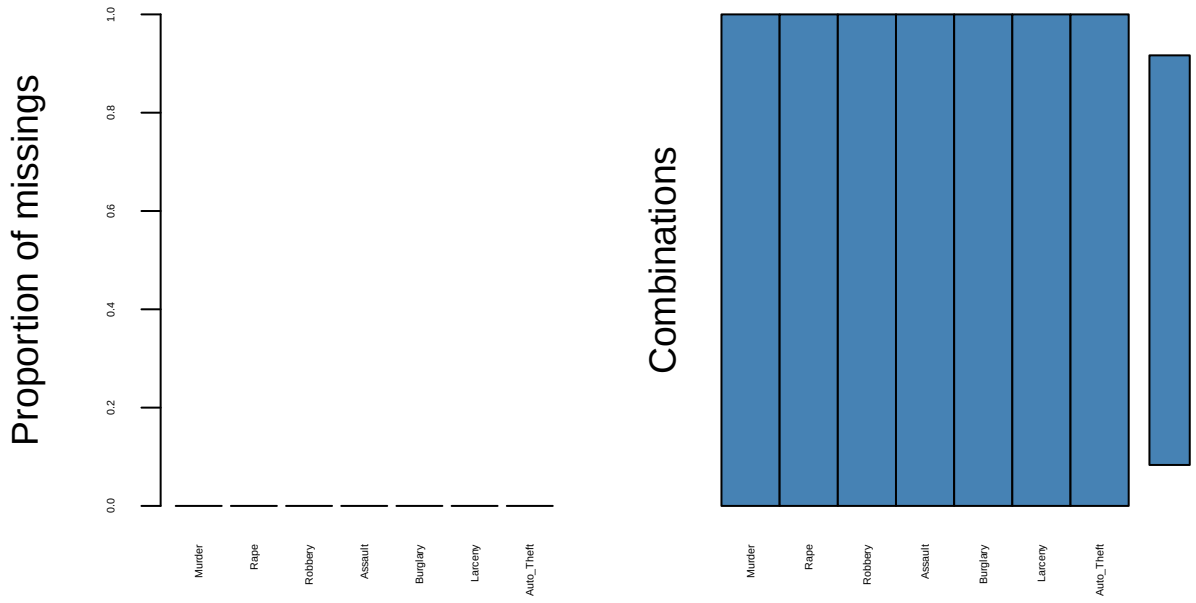
```
dim(crime_df1)
```

```
## [1] 50  7
```

Revisamos los "missing values" restantes. Analizamos el dataset para determinar su distribución (a qué variables afecta y cuánto) y tomar decisiones sobre qué hacer con ellos en cada caso.

No tenemos missing values en ninguna de las variables, como podemos observar también en el siguiente gráfico:

```
crime_nas2 <- crime_df1 %>%
  select(names(crime_df[crime_df!=0]))
aggr(crime_df1, prop=c(TRUE,TRUE), col=c("SteelBlue", "firebrick1", "LightGrey"), numbers=F,
     sortVar=TRUE, cex.axis=0.3)
```




```
##
##  Variables sorted by number of missings:
##    Variable Count
##      Murder     0
##        Rape     0
##     Robbery     0
##     Assault     0
##    Burglary     0
##     Larceny     0
##  Auto_Theft     0
```

En caso de tener missing values habría que analizar su tratamiento (si es más idóneo descartar las observaciones o imputar los valores), pero al no tener ninguno no es necesario.

2.2. ANÁLISIS Y TRATAMIENTO DE OUTLIERS

Una vez hemos organizado y preparado el dataset, pasamos a analizar los estadísticos básicos de cada variable para tratar de encontrar dispersiones altas o outliers entre las variables del mismo. Preparo el dataset y separo aquellas variables que no son relevantes (en este caso la primera variable "State" que es categórica).

```
str(crime_df1)
```

```
## 'data.frame':    50 obs. of  7 variables:
##  $ Murder    : num  14.2 10.8 9.5 8.8 11.5 6.3 4.2 6 10.2 11.7 ...
##  $ Rape      : num  25.2 51.6 34.2 27.6 49.4 42 16.8 24.9 39.6 31.1 ...
##  $ Robbery   : num  96.8 96.8 138.2 83.2 287 ...
##  $ Assault   : num  278 284 312 203 358 ...
##  $ Burglary  : num  1136 1332 2346 973 2139 ...
##  $ Larceny   : num  1882 3370 4467 1862 3500 ...
##  $ Auto_Theft: num  281 753 440 183 664 ...
```

Obtenemos los estadísticos básicos de todas las variables (nos fijaremos principalmente en: mínimo, máximo, media y mediana) y revisamos los posibles outliers o valores erróneos del dataset.

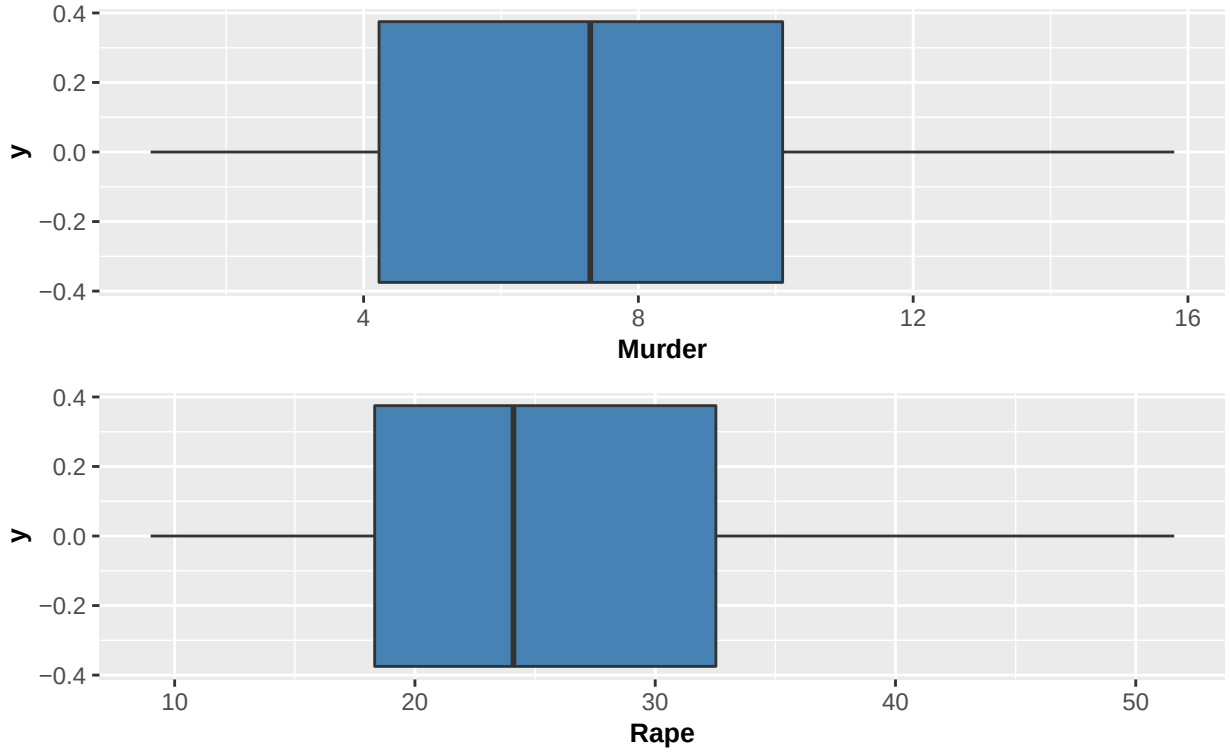
```
summary(crime_df1)
```

```
##      Murder           Rape          Robbery          Assault
##  Min.   : 0.900   Min.   : 9.00   Min.   : 13.30   Min.   : 43.8
##  1st Qu.: 4.225   1st Qu.:18.32   1st Qu.: 64.95   1st Qu.:148.8
##  Median : 7.300   Median :24.10   Median :106.05   Median :197.6
##  Mean   : 7.444   Mean   :25.73   Mean   :124.09   Mean   :211.3
##  3rd Qu.:10.100   3rd Qu.:32.52   3rd Qu.:155.85   3rd Qu.:282.6
##  Max.   :15.800   Max.   :51.60   Max.   :472.60   Max.   :485.3
##     Burglary        Larceny       Auto_Theft
##  Min.   : 446.1   Min.   :1240   Min.   : 144.4
##  1st Qu.:1000.1   1st Qu.:2249   1st Qu.: 245.8
##  Median :1265.0   Median :2617   Median : 333.9
##  Mean   :1291.9   Mean   :2671   Mean   : 377.5
##  3rd Qu.:1529.8   3rd Qu.:3008   3rd Qu.: 460.1
##  Max.   :2453.1   Max.   :4467   Max.   :1140.1
```

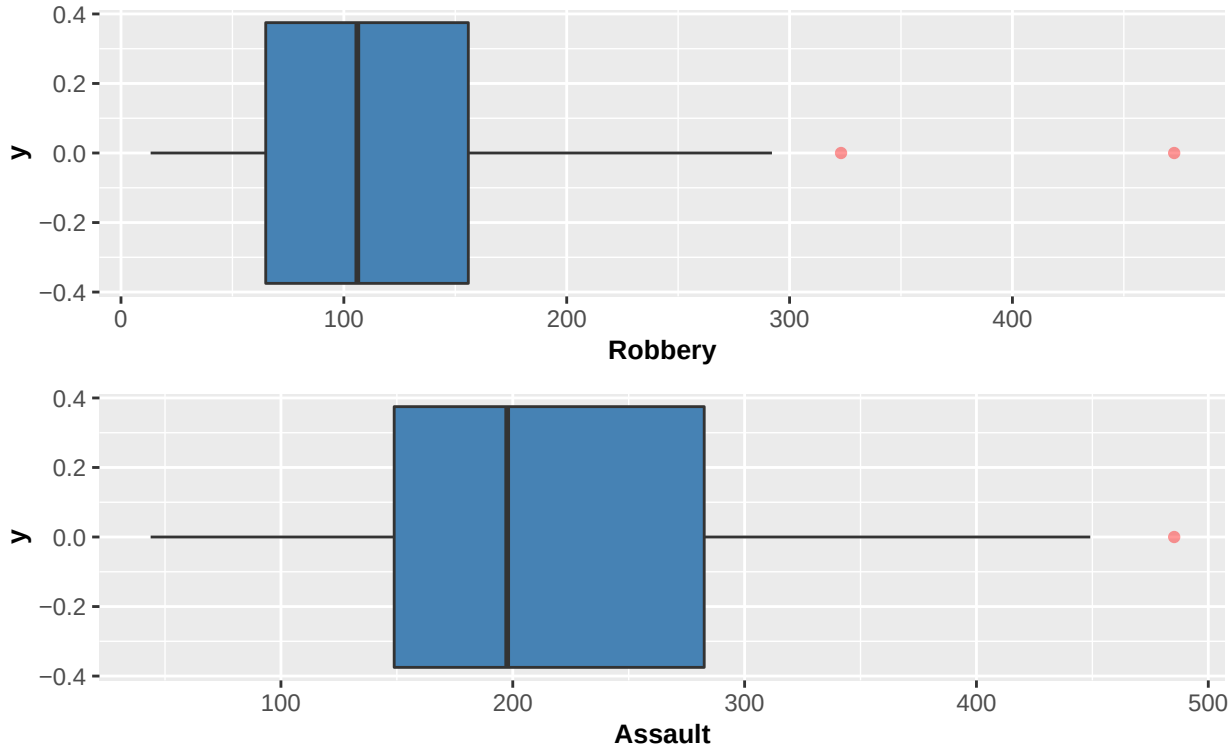
En el primer análisis numérico de los datos se observan algunos máximos altos dentro de las distribuciones de sus variables, pero no se observan variables con dispersiones de valores especialmente grandes o valores mínimos/máximos muy separados de las medias y medianas de dichas variables. Éstos valores pueden constituir los posibles outliers o valores erróneos que estamos buscando, para evitar fallos en el análisis posterior de los datos.

Para seguir analizando esos valores vamos a visualizar las distribuciones de valores de todas las variables de nuestro dataset, resaltando los posibles outliers.

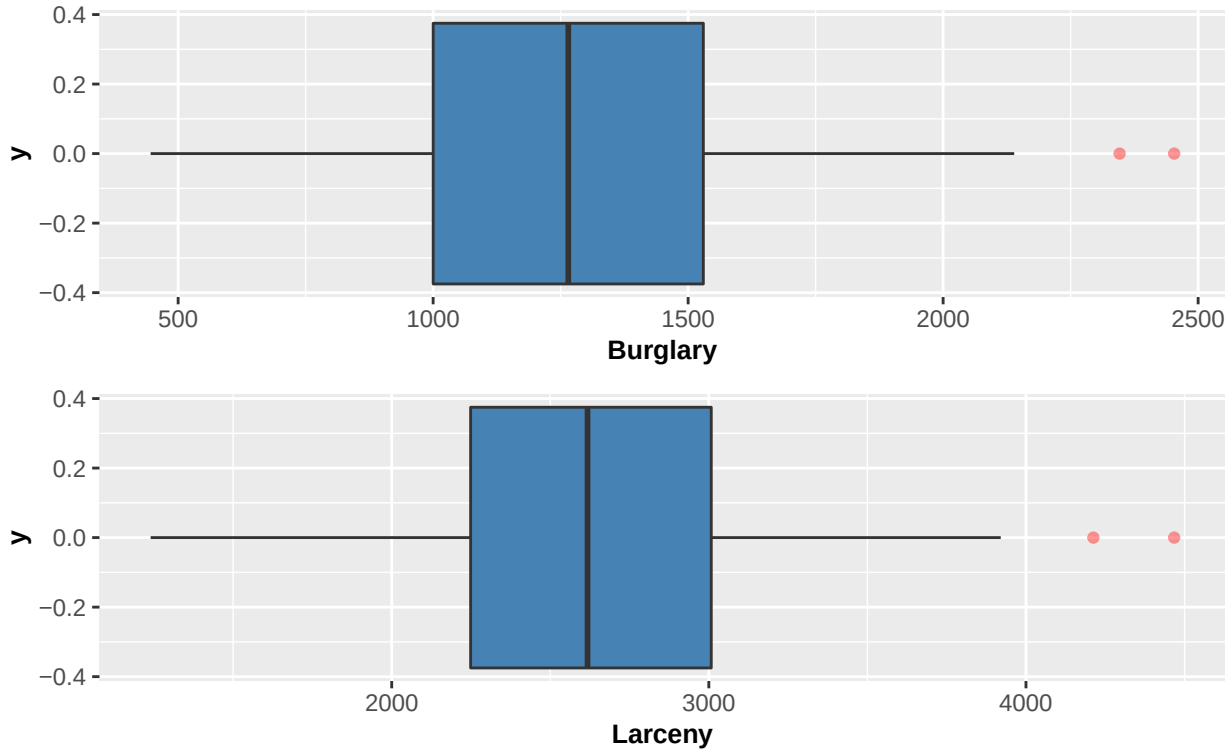
```
df_varsboxplots(crime_df1, c("Murder", "Rape"), nrows=2, ncols=1)
```



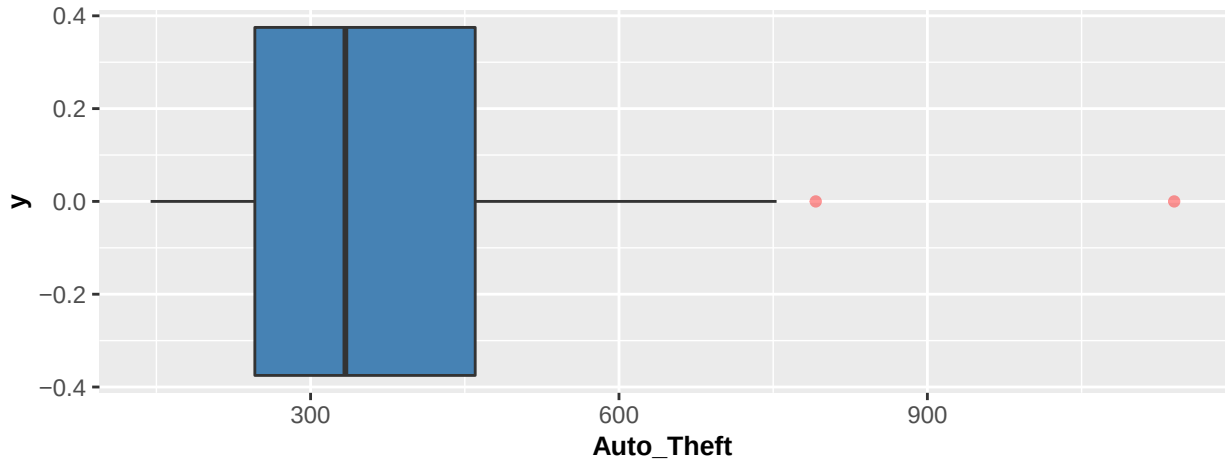

```
df_varsboxplots(crime_df1, c("Robbery", "Assault"), nrows=2, ncols=1)
```



```
df_varsboxplots(crime_df1, c("Burglary", "Larceny"), nrows=2, ncols=1)
```




```
df_varsboxplots(crime_df1, c("Auto_Theft"), nrows=2, ncols=1)
```



Como podemos observar en los gráficos, salvo en las 2 primeras variables (“Murder” y “Rape”), el resto sí que tienen valores que se alejan de la distribución de la mayoría de los valores de dicha variable.

La ausencia de outliers en las 2 primeras variables podría deberse a que los crímenes que representan estas variables son los de mayor gravedad (“Asesinato” y “Violación”) y no hay grandes diferencias entre los distintos estados (la distribución de estos crímenes es más homogénea en todo el país).

Analizamos los gráficos de las variables con presencia de outliers:

- Variable "Robbery" (relativa a robos con violencia)

```
crime_out1 <- crime_df1 %>%
  select(c("Robbery")) %>%
  arrange(rank=rank(Robbery))
head(crime_out1,5)
```

```
##             Robbery
## North Dakot    13.3
## South Dakot    17.9
## New Hampshi    23.2
## Vermont        30.8
## Maine          38.7
```

```
tail(crime_out1,5)
```

```
##            Robbery
## Michigan     261.9
## California   287.0
## Maryland     292.1
## Nevada       323.1
## New York     472.6
```

- Variable "Assault" (relativa a agresiones a personas)

```
crime_out2 <- crime_df1 %>%
  select(c("Assault")) %>%
  arrange(rank=rank(Assault))
head(crime_out2,5)
```

```
##             Assault
## North Dakot    43.8
## Wisconsin      63.7
## Hawaii         64.1
## New Hampshi    76.0
## Minnesota      85.8
```

```
tail(crime_out2,5)
```

```
##             Assault
## Nevada        355.0
## California    358.0
## Maryland      358.9
## Florida       449.1
## South Carol   485.3
```

- Variable “Burglary” (relativa a allanamientos de propiedades privadas con o sin hurto y/o violencia)

```
crime_out3 <- crime_df1 %>%
  select(c("Burglary")) %>%
  arrange(rank=rank(Burglary))
head(crime_out3,5)
```

```
##             Burglary
## North Dakot    446.1
## South Dakot    570.5
## West Virgin    597.4
## Nebraska       760.0
## Montana        804.9
```

```
tail(crime_out3,5)
```

```
##            Burglary
## Hawaii       1911.5
## Colorado     1935.2
## California   2139.4
## Arizona      2346.1
## Nevada       2453.1
```

- Variable “Larceny” (relativa relativa a robos sin violencia)

```
crime_out4 <- crime_df1 %>%
  select(c("Larceny")) %>%
  arrange(rank=rank(Larceny))
head(crime_out4,5)
```

```
##             Larceny
## Mississippi  1239.9
## West Virgin  1341.7
## Pennsylvani  1624.1
## Kentucky     1662.1
## South Dakot  1704.4
```

```
tail(crime_out4,5)
```

```
##          Larceny
## Florida   3840.5
## Colorado  3903.2
## Hawaii    3920.4
## Nevada    4212.6
## Arizona   4467.4
```

- Variable “Auto_Theft” (relativa a robos de vehículos)

```
crime_out5 <- crime_df1 %>%
  select(c("Auto_Theft")) %>%
  arrange(rank=rank(Auto_Theft))
head(crime_out5,5)
```

```
##             Auto_Theft
## Mississippi      144.4
## North Dakot      144.7
## South Dakot      147.5
## West Virgin      163.3
## Arkansas         183.4
```

```
tail(crime_out5,5)
```

```
##             Auto_Theft
## California       663.5
## New York         745.8
## Alaska           753.3
## Rhode Islan      791.4
## Massachuset     1140.1
```

Del análisis anterior podemos interpretar que los valores extremos que observamos en algunas de las variables no se pueden considerar “outliers” o valores erróneos.

Como se puede ver en los extractos ordenados de las distintas variables, los valores extremos en dichas variables se debe a las diferencias de ciertos estados.

2.3. RELACIONES ENTRE VARIABLES

Matrices de Covarianzas y de Correlación

- Matriz de covarianzas (S):

```
S <- cov(crime_df1)
paste("Det(S)=",det(S), " / ", "Sum(diag(S))=",sum(diag(S)), sep=" ")
```

```
## [1] "Det(S)= 4.16382109820627e+24  /  Sum(diag(S))= 769349.657559184"
```

- Matriz de correlaciones (R):

```
R <- cor(crime_df1)
paste("Det(R)=",det(R), " / ", "Sum(diag(R))=",sum(diag(R)), sep=" ")
```

```
## [1] "Det(R)= 0.00831900900246277  /  Sum(diag(R))= 7"
```

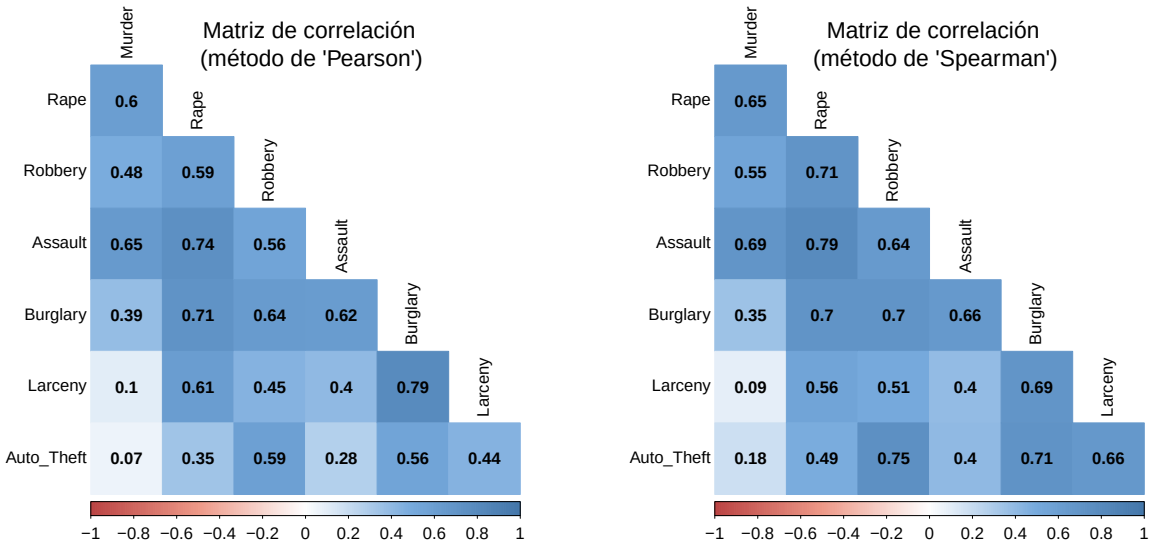
Las matrices anteriores nos sirven para hacernos una idea de lo homogéneos/heterogéneos que son los datos y de lo correlacionados (relaciones entre variables) o no que están los datos.

En este caso como las variables tienen unidades diferentes, se va a emplear la matriz de correlaciones para el análisis. El determinante de la matriz de correlación es próximo a cero por lo que podemos asumir que existen relaciones entre variables (éstas pueden ser lineales o no).

Preparamos una visualización de la matriz de correlación para analizar de forma más sencilla las variables relacionadas, cantidad de relaciones y fuerza de las mismas. Existen varios métodos para calcular las correlaciones entre variables, en este caso vamos a utilizar dos:

- Método de "Pearson"_ Analiza correlaciones de tipo lineal, cuando existe relación lineal entre variables.
- Método de "Spearman"_ Analiza correlaciones de tipo rango, cuando existe relación entre el crecimiento de una variable y a la vez el de otra, y viceversa (no la relación lineal solamente).

```
corr_01 <- cor(crime_df1, method = c("pearson"))
corr_02 <- cor(crime_df1, method = c("spearman"))

col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))

par(mfrow=c(1,2))
corrplot_01 <- corrplot(corr_01, method="color", type="lower",
                        tl.col="black",diag=FALSE, col=col(200),
                        addCoef.col = "black", tl.cex=2, number.cex=2, cl.cex=2)
mtext("\n\n Matriz de correlación \n (método de 'Pearson')", at=3.55, line=-5, cex=2.75)

corrplot_02 <- corrplot(corr_02, method="color", type="lower",
                        tl.col="black",diag=FALSE, col=col(200),
                        addCoef.col = "black", tl.cex=2, number.cex=2, cl.cex=2)
mtext("\n\n Matriz de correlación \n (método de 'Spearman')", at=3.55, line=-5, cex=2.75)
```



Este gráfico es más útil que los scatterplots separados. Nos fijamos en los valores azules/rojos oscuros, son aquellos con relaciones fuertes.

De todas maneras hay que tener en cuenta que se están analizando las posibles relaciones lineales, pudiendo haber variables en el gráfico con apenas relación lineal con otras variables, pero pudiendo tener una relación no lineal con éstas.

Al utilizar los métodos de "Pearson" y "Spearman" podemos comprobar las posibles relaciones lineales como no lineales entre variables. Tras revisar los gráficos de correlación observamos que existen relaciones considerables entre algunas variables.

En este caso como criterio se ha decidido reducir la correlación entre variables descartando aquellas más correlacionadas con el resto de variables, tratando de reducir al máximo la correlación en las variables restantes.

Después de revisar las correlaciones (tanto lineales como no lineales), las variables descartadas son:

- Variable "Rape"_ variable muy correlacionada con "Robbery", "Assault" y "Burglary"
- Variable "Burglary"_ variable muy correlacionada con "Rape", "Robbery", "Larcery" y "Auto_Theft".

```
crime_df2 <- crime_df1 %>%
  select(-c("Rape","Burglary"))
```

2.4. TRANSFORMACIÓN DE LOS DATOS

Reviso las variables restantes y analizo si es necesario realizar alguna transformación.

```
str(crime_df2)
```

```
## 'data.frame':    50 obs. of  5 variables:
##  $ Murder    : num  14.2 10.8 9.5 8.8 11.5 6.3 4.2 6 10.2 11.7 ...
##  $ Robbery   : num  96.8 96.8 138.2 83.2 287 ...
##  $ Assault   : num  278 284 312 203 358 ...
##  $ Larceny   : num  1882 3370 4467 1862 3500 ...
##  $ Auto_Theft: num  281 753 440 183 664 ...
```

```
summary(crime_df2)
```

```
##      Murder          Robbery          Assault         Larceny
##  Min.   : 0.900   Min.   : 13.30   Min.   : 43.8   Min.   :1240
##  1st Qu.: 4.225   1st Qu.: 64.95   1st Qu.:148.8   1st Qu.:2249
##  Median : 7.300   Median :106.05   Median :197.6   Median :2617
##  Mean   : 7.444   Mean   :124.09   Mean   :211.3   Mean   :2671
##  3rd Qu.:10.100   3rd Qu.:155.85   3rd Qu.:282.6   3rd Qu.:3008
##  Max.   :15.800   Max.   :472.60   Max.   :485.3   Max.   :4467
##    Auto_Theft
##  Min.   : 144.4
##  1st Qu.: 245.8
##  Median : 333.9
##  Mean   : 377.5
##  3rd Qu.: 460.1
##  Max.   :1140.1
```

En el presente caso, todas las variables a analizar son variables numéricas continuas. Las variables continuas pueden requerir en algunos casos, una transformación previa al proceso de clustering. La transformación más popular es la estandarización ya que con ello se evita la influencia de la unidad de medida.

Las variables de los datos, pese a representar sucesos diferentes utilizan unidades similares, midiendo el número de dichos sucesos por cada 10.000 habitantes por lo que podemos considerar dichas unidades como equivalentes. Aún teniendo en cuenta este hecho, las variables que tienen un mayor rango de variación tienden a tener más importancia a la hora de conformar los clusters por ello se van estandarizar todas las variables.

```
crime_df3 <- as.data.frame(scale(crime_df2, center = FALSE,
                                 scale = apply(crime_df2, 2, sd, na.rm = TRUE)))
str(crime_df3)
```

```
## 'data.frame':    50 obs. of  5 variables:
##  $ Murder    : num  3.67 2.79 2.46 2.28 2.97 ...
##  $ Robbery   : num  1.096 1.096 1.564 0.942 3.248 ...
##  $ Assault   : num  2.78 2.83 3.12 2.03 3.57 ...
##  $ Larceny   : num  2.59 4.64 6.15 2.57 4.82 ...
##  $ Auto_Theft: num  1.451 3.895 2.273 0.948 3.431 ...
```

2.5. DATOS PREPARADOS

Teniendo en cuenta todo el análisis anterior, definimos los datos finales a utilizar en el clustering.

```
str(crime_df3)
```

```
## 'data.frame':    50 obs. of  5 variables:
##  $ Murder    : num  3.67 2.79 2.46 2.28 2.97 ...
##  $ Robbery   : num  1.096 1.096 1.564 0.942 3.248 ...
##  $ Assault   : num  2.78 2.83 3.12 2.03 3.57 ...
##  $ Larceny   : num  2.59 4.64 6.15 2.57 4.82 ...
##  $ Auto_Theft: num  1.451 3.895 2.273 0.948 3.431 ...
```

```
summary(crime_df3)
```

```
##      Murder          Robbery          Assault          Larceny
##  Min.   :0.2328   Min.   :0.1505   Min.   :0.4369   Min.   :1.708
##  1st Qu.:1.0926   1st Qu.:0.7352   1st Qu.:1.4847   1st Qu.:3.098
##  Median :1.8879   Median :1.2004   Median :1.9710   Median :3.606
##  Mean   :1.9251   Mean   :1.4046   Mean   :2.1077   Mean   :3.680
##  3rd Qu.:2.6120   3rd Qu.:1.7640   3rd Qu.:2.8186   3rd Qu.:4.143
##  Max.   :4.0861   Max.   :5.3493   Max.   :4.8408   Max.   :6.154
##    Auto_Theft
##  Min.   :0.7467
##  1st Qu.:1.2708
##  Median :1.7263
##  Mean   :1.9521
##  3rd Qu.:2.3792
##  Max.   :5.8952
```

3. ANÁLISIS CLUSTER

3.1. SELECCIÓN DE LA MÉTRICA

Para realizar el clustering de los datos es preciso determinar cómo se medirá la distancia entre los elementos. En este caso vamos a utilizar la distancia "Euclídea". $d(u, v) = \sqrt{(u1 - u2)^2 + (v1 - v2)^2}$

```
matrizDistancias <- vegdist(crime_df3, method = "euclidean")
```

3.2. SELECCIÓN DEL ALGORITMO

Dado el tamaño pequeño de los datos, se podría emplear directamente un método jerárquico que, aunque más costoso computacionalmente (se suele aplicar a una muestra de los datos), permite obtener clasificaciones más cercanas a la óptima.

De todas maneras, con fines prácticos, para el presente análisis es preciso aclarar que se va a seguir el método "bietápico". En este método el preocedimiento a seguir es el siguiente:

- **i.** Aplicar un algoritmo "jerárquico" (obteniendo el número "K" de clusters y los centroides de partida)
- **ii.** Aplicar un algoritmo de "optimización" (utilizando los resultados anteriores)

Para el método de "optimización" necesitamos definir el número de clusters en el que vamos a dividir los elementos y los centroides de dichos clusters (ya que son métodos muy sensibles a los centroides de partida). Para ello utilizaremos los datos obtenidos de la aplicación del método "jerárquico".

Vamos a realizar una comprobación adicional calculando la distribución porcentual de elementos entre los clusters obtenidos del método "jerárquico" y comparándola con la resultante al aplicar el método de "optimización" (si el procedimiento se ha realizado correctamente, deberían ser muy similares).

3.3. MÉTODO JERÁRQUICO

Los métodos jerárquicos suelen tener un coste computacional alto por lo que es recomendable no aplicarlos a bases de datos muy grandes (por necesidad de capacidad y tiempo de procesamiento). Dado que los datos con los que estamos trabajando tienen tan sólo 50 observaciones, no se considera necesario extraer una muestra para aplicar el método jerárquico.

Método jerárquico (distancias entre grupos mediante el método "WARD")

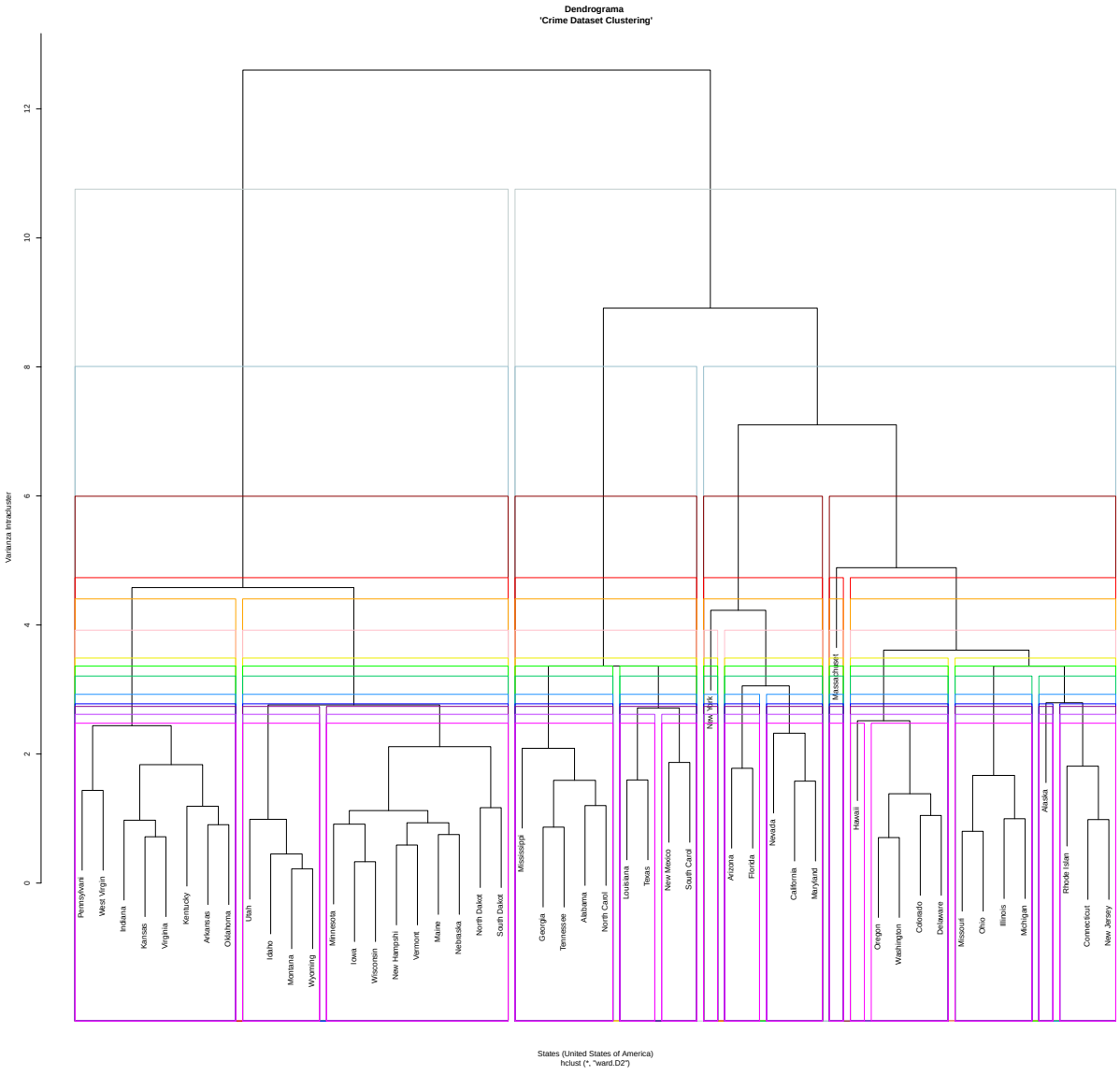
El método aglomerativo a utilizar va a ser uno de tipo disgregativo o de división. Éstos métodos parten del conjunto formado por todos los elementos y, en cada iteración, realizan una separación dando lugar a clusters más pequeños).

En los métodos jerárquicos se debe definir también la distancia entre grupos, para el presente análisis se va a utilizar a su vez el Método "WARD". Éste método busca agrupar los elementos buscando el mínimo incremento de "varianza intracluster" (desventajas: sensible a "outliers" y poco eficiente computacionalmente; ventaja: capaz de acercarse más a la clasificación óptima).

```
clusterJer <- hclust(matrizDistancias, method="ward.D2",)
clusterJer_labs <- rownames(crime_df3)
```

Para visualizar mejor los clusters generados en cada iteración y su influencia en la "varianza intracluster", representamos el dendrograma asociado al método aplicado:

```
plot(x=clusterJer, labels=clusterJer_labs, main="Dendrograma \n 'Crime Dataset Clustering'",
     xlab="States (United States of America)" , ylab= "Varianza Intracluster")
rect.hclust(clusterJer, k=2, border="azure3")
rect.hclust(clusterJer, k=3, border="lightblue3")
rect.hclust(clusterJer, k=4, border="darkred")
rect.hclust(clusterJer, k=5, border="red")
rect.hclust(clusterJer, k=6, border="orange")
rect.hclust(clusterJer, k=7, border="pink")
rect.hclust(clusterJer, k=8, border="yellow2")
rect.hclust(clusterJer, k=9, border="green")
rect.hclust(clusterJer, k=10, border="springgreen3")
rect.hclust(clusterJer, k=11, border="dodgerblue")
rect.hclust(clusterJer, k=12, border="blue")
rect.hclust(clusterJer, k=13, border="darkmagenta")
rect.hclust(clusterJer, k=14, border="darkorchid1")
rect.hclust(clusterJer, k=15, border="magenta1")
```

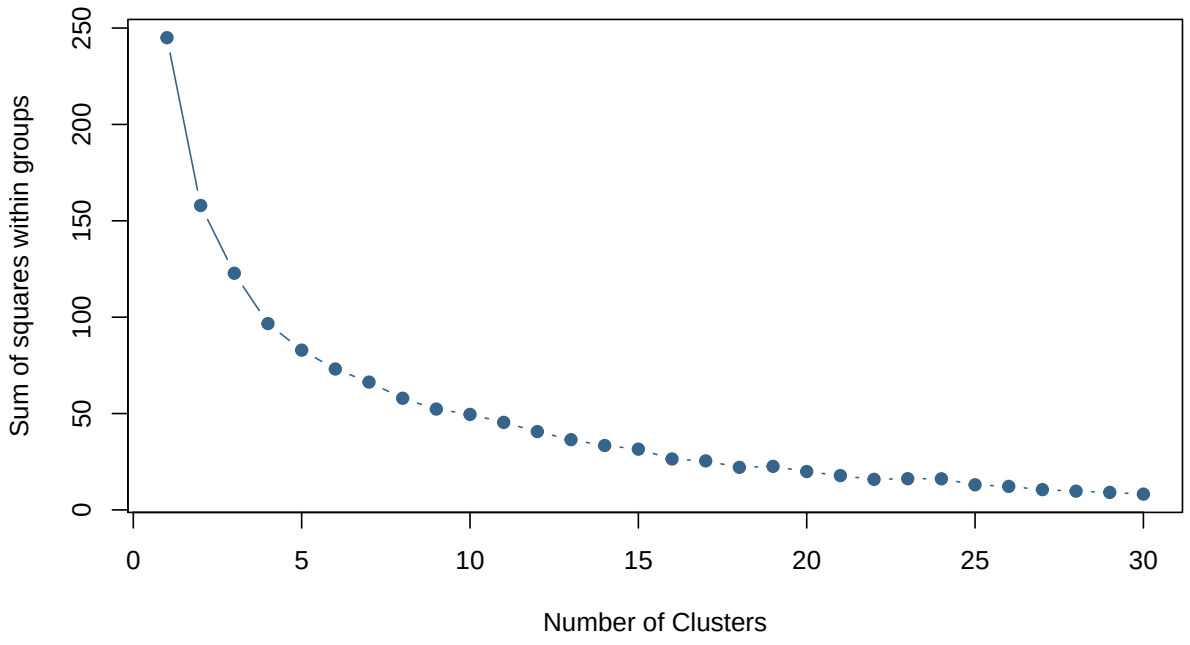


Después de revisar el dendrograma obtenido y de acuerdo a los valores de varianza intracluster, se puede observar que la disminución de la misma se reduce en gran medida a partir de K=4 (4 clusters). Además de este dendrograma, vamos a aplicar otros dos métodos alternativos para definir el número más óptimo de clusters para la segregación del dataset.

Método de Elbow (comprobación adicional)

No forma parte del método bietápico pero también podemos emplear el método de Elbow para definir el número de clusters, que evalúa la varianza intracluster en función del número de clusters.

```
# Calcular SSW (Sum of Squares Wihin) para distintos n?mero de grupos
set.seed(12345)
n <- dim(crime_df3)[1] # Numero de registros
p <- dim(crime_df3)[2] # Numero de variables

SSW <- (n - 1) * sum(apply(crime_df3,2,var))

# Se aplica el metodo k-means con 5 inicializaciones distintas de centroides
# para que no sea tan sensible a los centroides de partida
for (i in 2:30) SSW[i] <-
  sum(kmeans(crime_df3,centers=i,nstart=3,iter.max=20)$withinss)

# Metodo de Elbow
plot(1:30, SSW, type="b", xlab="Number of Clusters",
     ylab="Sum of squares within groups",pch=19, col="steelblue4")
```



Para determinar el número óptimo de clusters se debe buscar el "codo" en el gráfico, que da a entender que la consideración de un cluster adicional no mejora mucho la segmentación (en ocasiones se trata de un criterio subjetivo). En este caso, el "codo" podría considerarse en torno a 6 clusters pero no queda muy claro.

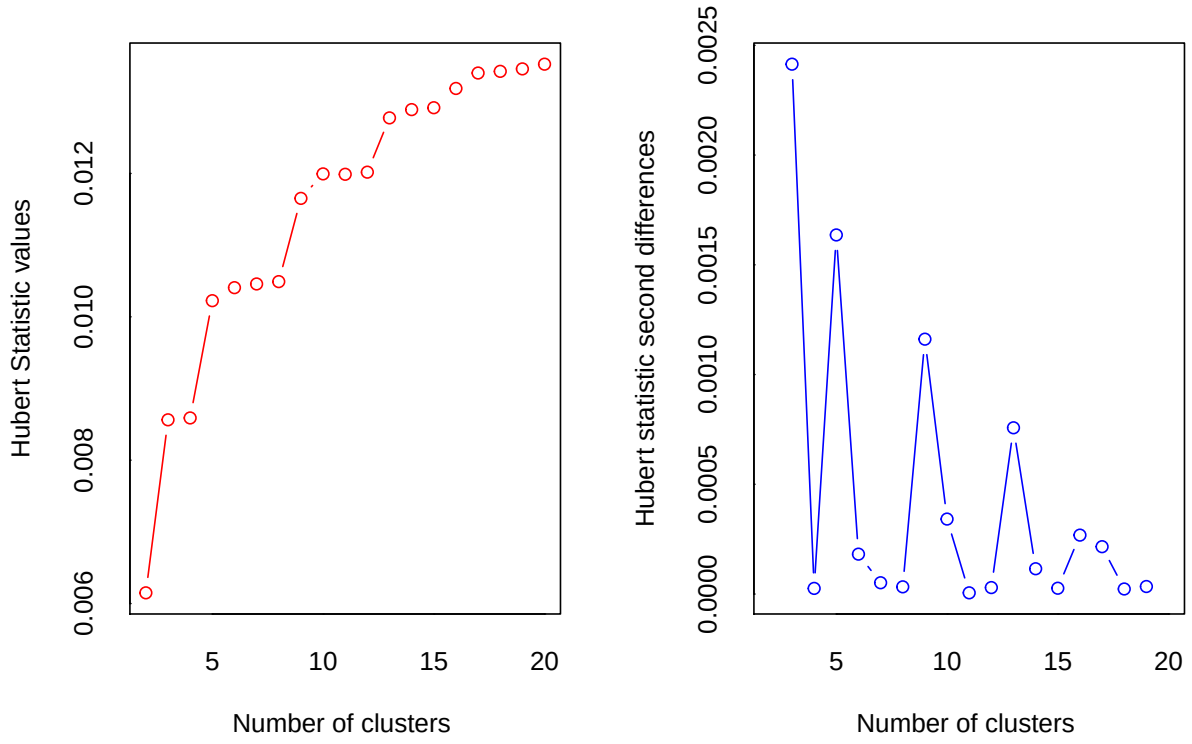
Función "NbClust" (comprobación adicional)

La segunda alternativa que vamos a emplear es aplicar la función "NbClust". La función "NbClust" toma los datos y tras definir una serie de parámetros, como la distancia entre elementos a utilizar y un rango de valores para los clusters buscados, aplica 30 índices para determinar los nº de clusters más óptimos. Como resultado, proporciona para los nº de cluster en el rango definido en la función el número de índices que lo consideran como el nº óptimo de clusters.

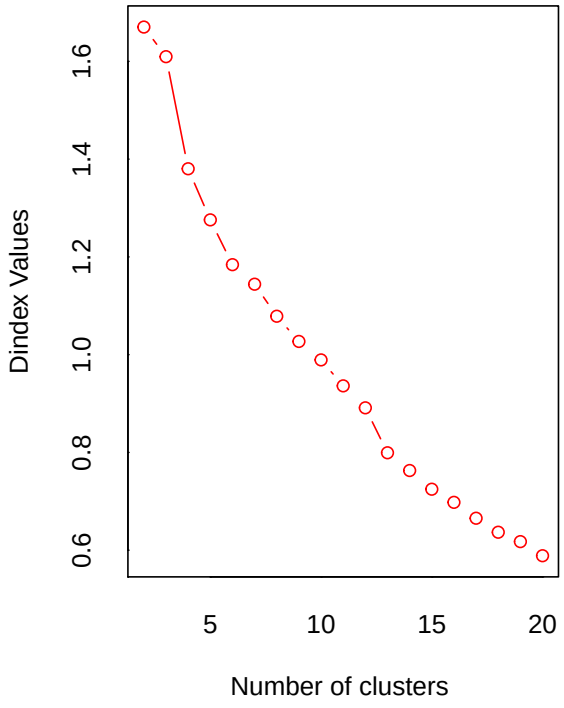
Esta función no debe considerarse como un resultado absoluto o definitivo, si no como una ayuda para visualizar los posibles nº óptimos de clusters que se deberían considerar. Tampoco se deben olvidar nunca los posibles criterios solicitados por un cliente o las condiciones que se consideren oportunas en el análisis y que pueden llegar a condicionar dicho análisis (también determinar el orden del nº de clusters).

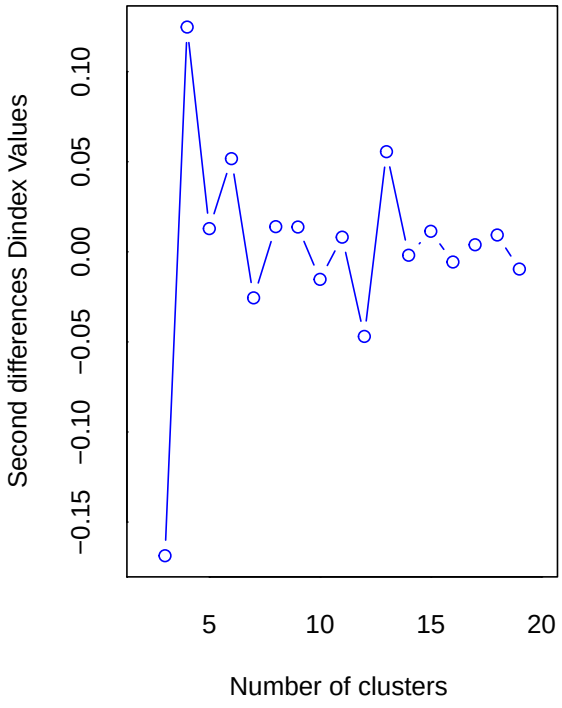
Por tanto, aplicamos la función "NbClust" con un primer rango de nº de clusters amplio (n=[2;20]) para determinar cuáles pueden ser los números óptimos de los mismos.

```
nbclust_01 <- NbClust(crime_df3, distance = "euclidean", min.nc = 2,
                      max.nc = 20, method = "complete", index ="all")
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

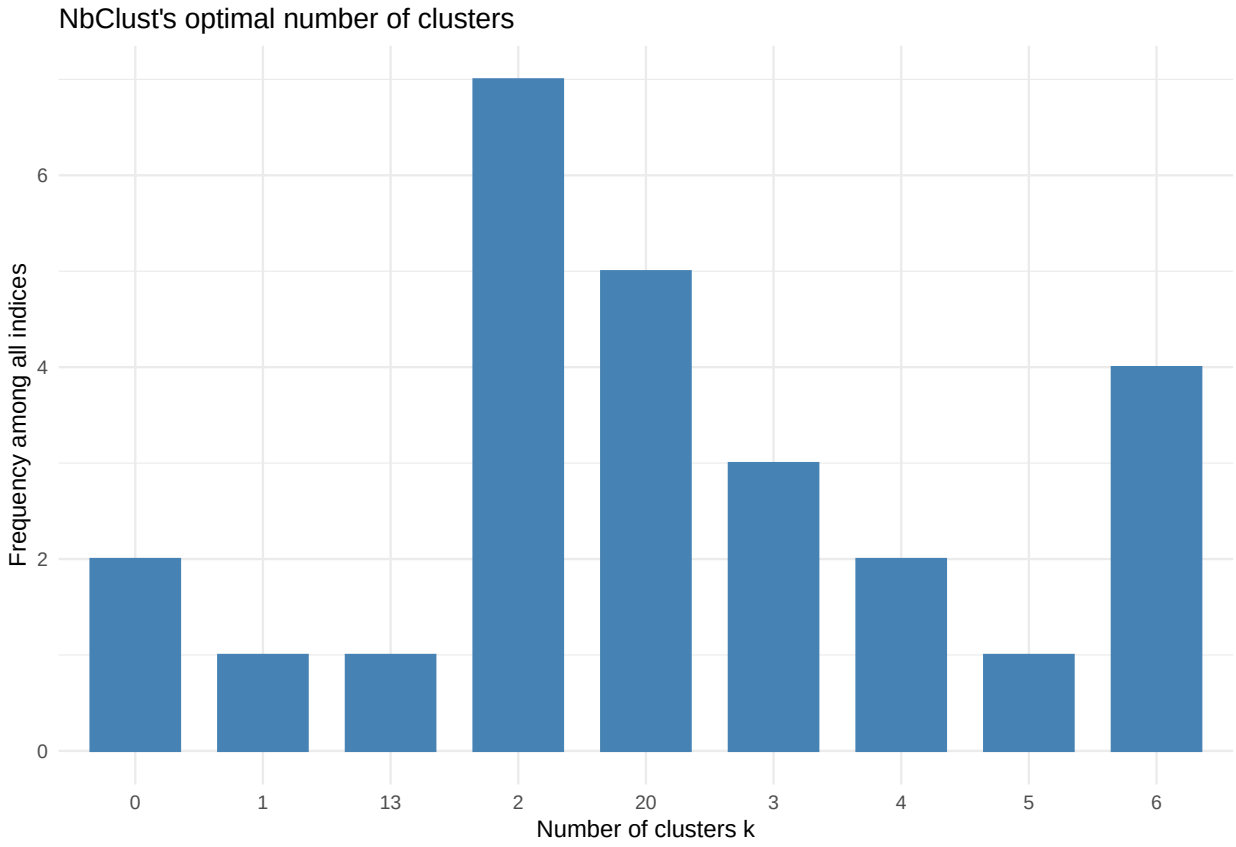





```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 7 proposed 2 as the best number of clusters
## * 3 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 4 proposed 6 as the best number of clusters
## * 1 proposed 13 as the best number of clusters
## * 5 proposed 20 as the best number of clusters
##
##                   ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
factoextra::fviz_nbclust(nbclust_01) + theme_minimal() +
  ggtitle("NbClust's optimal number of clusters")
```

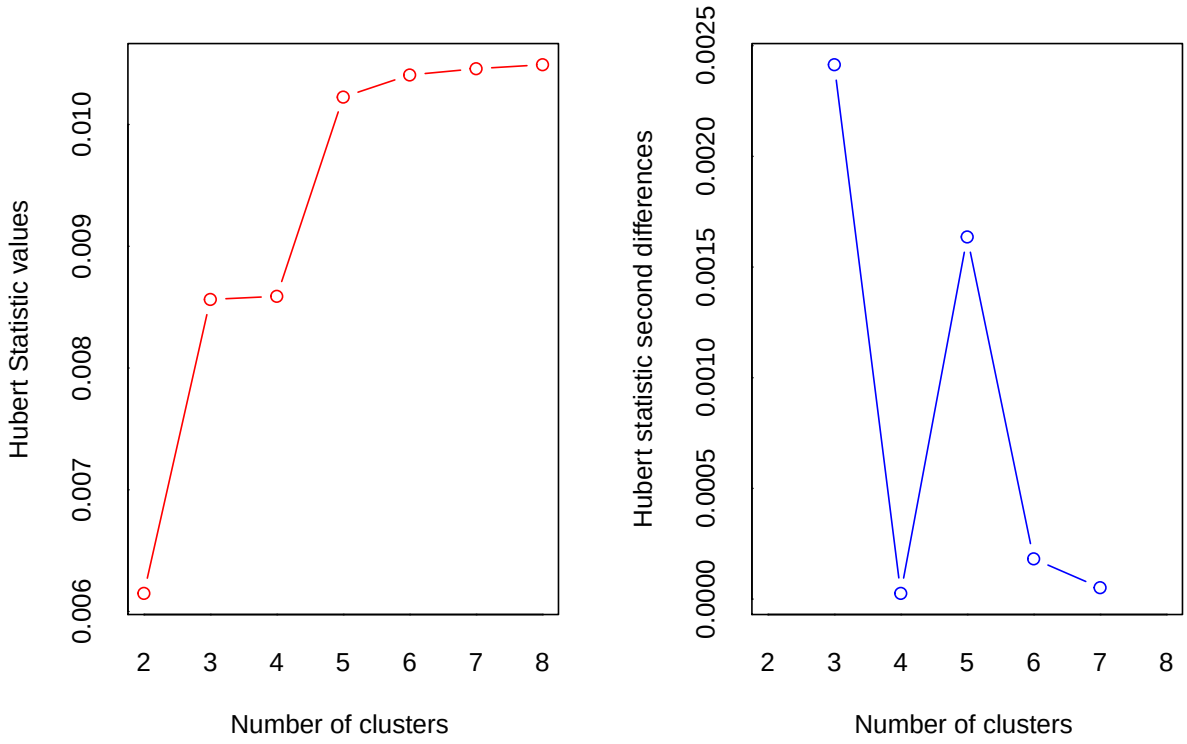
```
## Among all indices:
## ==================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 7 proposed  2 as the best number of clusters
## * 3 proposed  3 as the best number of clusters
## * 2 proposed  4 as the best number of clusters
## * 1 proposed  5 as the best number of clusters
## * 4 proposed  6 as the best number of clusters
## * 1 proposed  13 as the best number of clusters
## * 5 proposed  20 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```



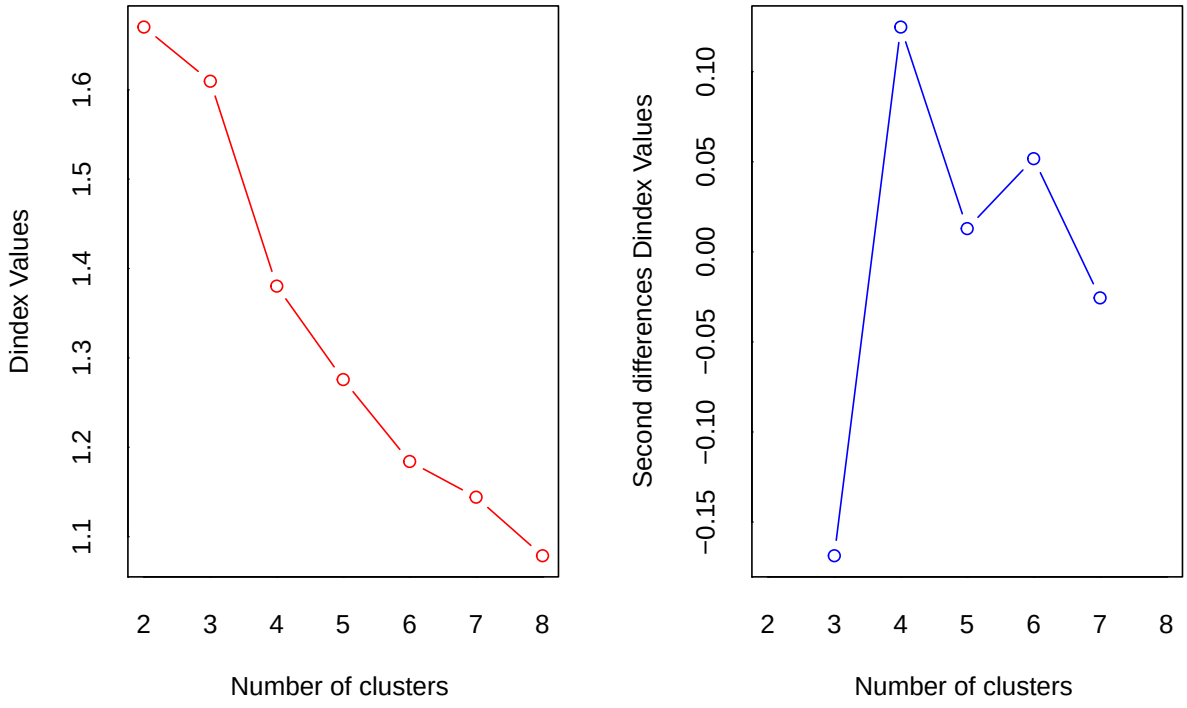
Los nº de clusters que obtienen los mayores números de indicadores son: 2, 6 y 20. Teniendo en cuenta el orden de magnitud del dataset (número de observaciones) se considera que no tendría mucho sentido hacer una división del mismo en 20 clusters, muchos de los cuales tendrían 1 ó 2 elementos tan sólo.

En base a los resultados anteriores y a la consideración tomada, se vuelve a aplicar la función "NbClust" pero sobre un rango más acotado del nº de clusters a obtener (n=[2;8]) y se analizan los resultados.

```
nbclust_02 <- NbClust(crime_df3, distance = "euclidean", min.nc = 2,
                      max.nc = 8, method = "complete", index ="all")
```



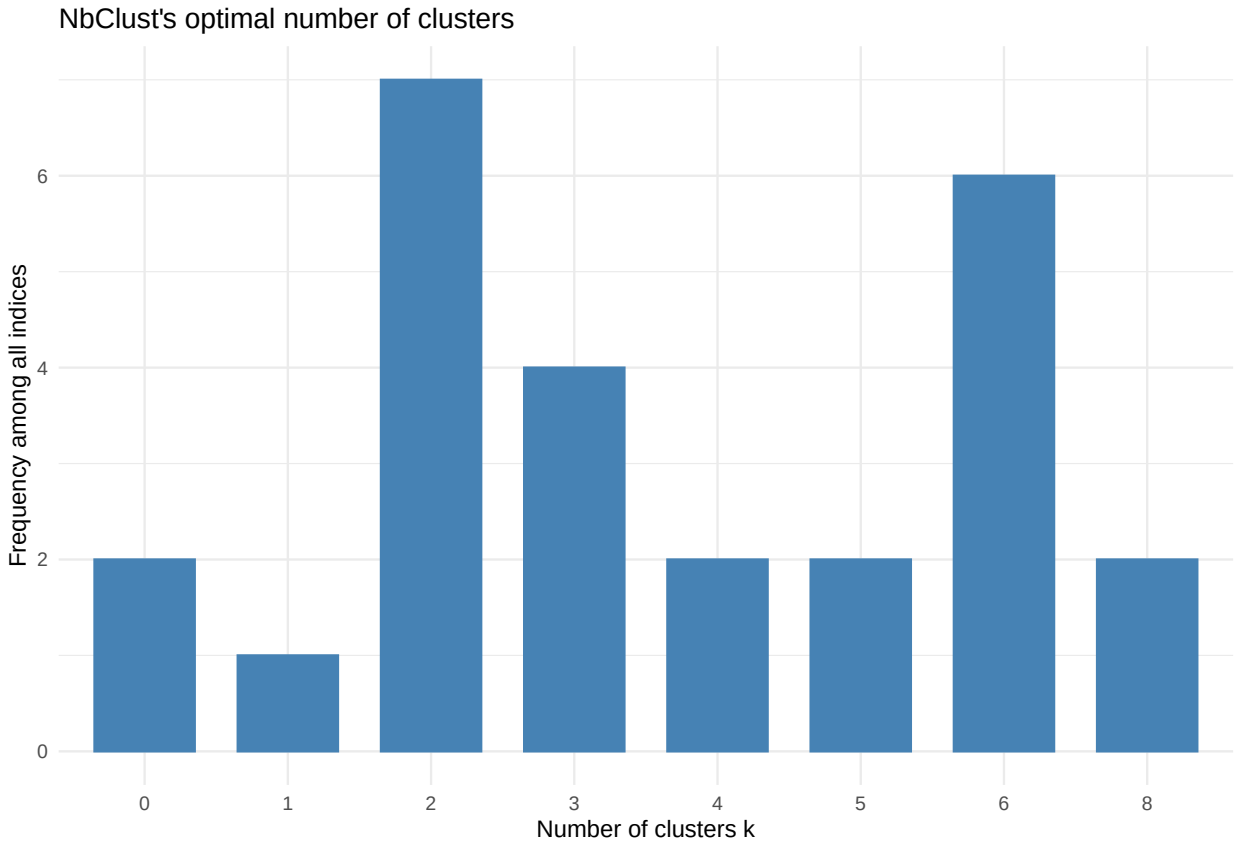

```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```




```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 7 proposed 2 as the best number of clusters
## * 4 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 2 proposed 5 as the best number of clusters
## * 6 proposed 6 as the best number of clusters
## * 2 proposed 8 as the best number of clusters
##
##                     ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************
```

```
factoextra::fviz_nbclust(nbclust_02) + theme_minimal() +
  ggtitle("NbClust's optimal number of clusters")
```

```
## Among all indices:
## ===================
## * 2 proposed  0 as the best number of clusters
## * 1 proposed  1 as the best number of clusters
## * 7 proposed  2 as the best number of clusters
## * 4 proposed  3 as the best number of clusters
## * 2 proposed  4 as the best number of clusters
## * 2 proposed  5 as the best number of clusters
## * 6 proposed  6 as the best number of clusters
## * 2 proposed  8 as the best number of clusters
##
## Conclusion
## =========================
## * According to the majority rule, the best number of clusters is  2 .
```



Los nº de clusters que obtienen los mayores números de indicadores son: 2 y 6. Estos números se repiten de nuevo respecto a la primera iteración y destacan considerablemente respecto al resto de opciones, por ello los tomaremos para hacer 2 segregaciones alternativas del dataset y comparar sus resultados.

Caso 1: 2 clusters (k=2)

Supongamos que se decide dividir todos los elementos en 2 clusters. Definimos el dataframe con una nueva columna que recoge el cluster al que pertenece cada observación.

```
asignJer_k2 <- cbind(crime_df3, cutree(clusterJer, k=2))

colnames(asignJer_k2)[6] <- "Cluster_k2"

str(asignJer_k2)
```

```
## 'data.frame':    50 obs. of  6 variables:
##  $ Murder    : num  3.67 2.79 2.46 2.28 2.97 ...
##  $ Robbery   : num  1.096 1.096 1.564 0.942 3.248 ...
##  $ Assault   : num  2.78 2.83 3.12 2.03 3.57 ...
##  $ Larceny   : num  2.59 4.64 6.15 2.57 4.82 ...
##  $ Auto_Theft: num  1.451 3.895 2.273 0.948 3.431 ...
##  $ Cluster_k2: int  1 1 1 2 1 1 1 1 1 1 ...
```

Calculamos los centroides asociados a cada uno de los grupos jerárquicos (clusters):

```
centroJer_k2 <- sqldf("SELECT Cluster_k2,
                count(*) AS tamano_Cluster,
                avg(Murder) AS Murder,
                avg(Robbery) AS Robbery,
                avg(Assault) AS Assault,
                avg(Larceny) AS Larceny,
                avg(Auto_Theft) AS Auto_Theft
                FROM asignJer_k2
                GROUP BY Cluster_k2")

print.data.frame(centroJer_k2)
```

```
##   Cluster_k2 tamano_Cluster   Murder   Robbery Assault  Larceny Auto_Theft
## 1          1             29 2.396191 1.9213863 2.67100 4.052937   2.403878
## 2          2             21 1.274597 0.6908782 1.32973 3.164808   1.328226
```

Calculamos la distribución porcentual de los elementos en cada grupo jerárquico (cluster):

```
tam_Cluster_k2 <- centroJer_k2[2]
tam_Cluster_k2
```

```
##   tamano_Cluster
## 1             29
## 2             21
```

```
perc_tamJer_Cluster_k2 <- centroJer_k2[2]/sum(centroJer_k2[2])
perc_tamJer_Cluster_k2
```

```
##   tamano_Cluster
## 1           0.58
## 2           0.42
```

Caso 2: 6 clusters (k=6)

Supongamos que se decide dividir todos los elementos en 6 clusters. Definimos el dataframe con una nueva columna que recoge el cluster al que pertenece cada observación.

```
asignJer_k6 <- cbind(crime_df3, cutree(clusterJer, k=6))

colnames(asignJer_k6)[6] <- "Cluster_k6"

str(asignJer_k6)
```

```
## 'data.frame':    50 obs. of  6 variables:
##  $ Murder    : num  3.67 2.79 2.46 2.28 2.97 ...
##  $ Robbery   : num  1.096 1.096 1.564 0.942 3.248 ...
##  $ Assault   : num  2.78 2.83 3.12 2.03 3.57 ...
##  $ Larceny   : num  2.59 4.64 6.15 2.57 4.82 ...
##  $ Auto_Theft: num  1.451 3.895 2.273 0.948 3.431 ...
##  $ Cluster_k6: int  1 2 3 4 3 2 2 2 3 1 ...
```

Calculamos los centroides asociados a cada uno de los grupos jerárquicos (clusters):

```
centroJer_k6 <- sqldf("SELECT Cluster_k6,
                count(*) AS tamano_Cluster,
                avg(Murder) AS Murder,
                avg(Robbery) AS Robbery,
                avg(Assault) AS Assault,
                avg(Larceny) AS Larceny,
                avg(Auto_Theft) AS Auto_Theft
                FROM asignJer_k6
                GROUP BY Cluster_k6")

print.data.frame(centroJer_k6)
```

```
##   Cluster_k6 tamano_Cluster    Murder   Robbery  Assault  Larceny Auto_Theft
## 1          1              9 3.1723299 1.2839295 2.902101 3.048424   1.418517
## 2          2             13 1.7804569 1.7691289 2.120020 4.356619   2.658974
## 3          3              6 2.8318216 3.2086919 3.578279 5.046548   2.747322
## 4          4              8 2.0074900 1.0280303 1.558930 2.837357   1.357653
## 5          5             13 0.8235856 0.4833999 1.188684 3.366317   1.310117
## 6          6              1 0.8017029 1.9140095 2.310154 3.184009   5.895206
```

Calculamos la distribución porcentual de los elementos en cada grupo jerárquico (cluster):

```
tam_Cluster_k6 <- centroJer_k6[2]
tam_Cluster_k6
```

```
##   tamano_Cluster
## 1              9
## 2             13
```

```
## 3              6
## 4              8
## 5             13
## 6              1
```

```
perc_tamJer_Cluster_k6 <- centroJer_k6[2]/sum(centroJer_k6[2])
perc_tamJer_Cluster_k6
```

```
##   tamano_Cluster
## 1           0.18
## 2           0.26
## 3           0.12
## 4           0.16
## 5           0.26
## 6           0.02
```

3.4. MÉTODO DE OPTIMIZACIÓN

El método de optimización a utilizar va a ser el método "Kmeans". Los métodos de optimización tienen menor coste computacional y son más rápidos pero presuponen el número de clusters a generar. Además son métodos muy sensibles a los centroides de partida.

Para evitar estos inconvenientes el método bietápico utiliza el número de clusters y los centroides de los mismos, calculados en el método jerárquico anterior, como información de partida para el método Kmeans.

Caso 1: 2 clusters (k=2)

Se ejecuta el método "Kmeans" con los centroides obtenidos con el jerárquico

```
kmeans_k2 <- kmeans(crime_df3,centers=centroJer_k2[,3:7])
kmeans_k2$centers
```

```
##     Murder   Robbery  Assault  Larceny Auto_Theft
## 1 2.289295 2.1078563 2.770207 4.407837   2.621293
## 2 1.614900 0.8054795 1.543280 3.059848   1.382054
```

Calculamos la distribución porcentual de los elementos en cada grupo del método de optimización (cluster):

```
perc_tamKms_Cluster_k2 <- kmeans_k2$size/sum(kmeans_k2$size)
perc_tamKms_Cluster_k2
```

```
## [1] 0.46 0.54
```

Caso 2: 6 clusters (k=6)

Se ejecuta el método "Kmeans" con los centroides obtenidos con el jerárquico.

```
kmeans_k6 <- kmeans(crime_df3,centers=centroJer_k6[,3:7])
kmeans_k6$centers
```

```
##      Murder   Robbery  Assault  Larceny Auto_Theft
## 1 3.0904355 1.4125866 3.065992 3.327502   1.544124
## 2 1.7985418 1.6800190 2.145661 4.652814   2.542708
## 3 2.8232012 3.4420479 3.515604 4.746143   2.838672
## 4 2.2959163 1.0531517 1.621231 2.564510   1.329810
## 5 0.8866761 0.5302859 1.232381 3.395409   1.306767
## 6 0.8663564 1.4465430 2.157540 3.550997   4.993681
```

Calculamos la distribución porcentual de los elementos en cada grupo del método de optimización (cluster):

```
perc_tamKms_Cluster_k6 <- kmeans_k6$size/sum(kmeans_k6$size)
perc_tamKms_Cluster_k6
```

```
## [1] 0.16 0.22 0.12 0.18 0.28 0.04
```

3.5. VISUALIZACIÓN DE LOS RESULTADOS

Caso 1: 2 clusters (k=2)

Centroides de los clusters definidos por ambos métodos.

- Método jerárquico:

```
print.data.frame(centroJer_k2)
```

```
##   Cluster_k2 tamano_Cluster   Murder   Robbery Assault  Larceny Auto_Theft
## 1          1             29 2.396191 1.9213863 2.67100 4.052937   2.403878
## 2          2             21 1.274597 0.6908782 1.32973 3.164808   1.328226
```

- Método Kmeans:

```
cbind(Cluster_k2=c(1:2),tamano_Cluster=kmeans_k2$size,kmeans_k2$centers)
```

```
##   Cluster_k2 tamano_Cluster   Murder   Robbery  Assault  Larceny Auto_Theft
## 1          1             23 2.289295 2.1078563 2.770207 4.407837   2.621293
## 2          2             27 1.614900 0.8054795 1.543280 3.059848   1.382054
```

Distribución porcentual de los elementos en cada grupo de ambos métodos.

- Método jerárquico:

```
perc_tamJer_Cluster_k2
```

```
##   tamano_Cluster
## 1           0.58
## 2           0.42
```

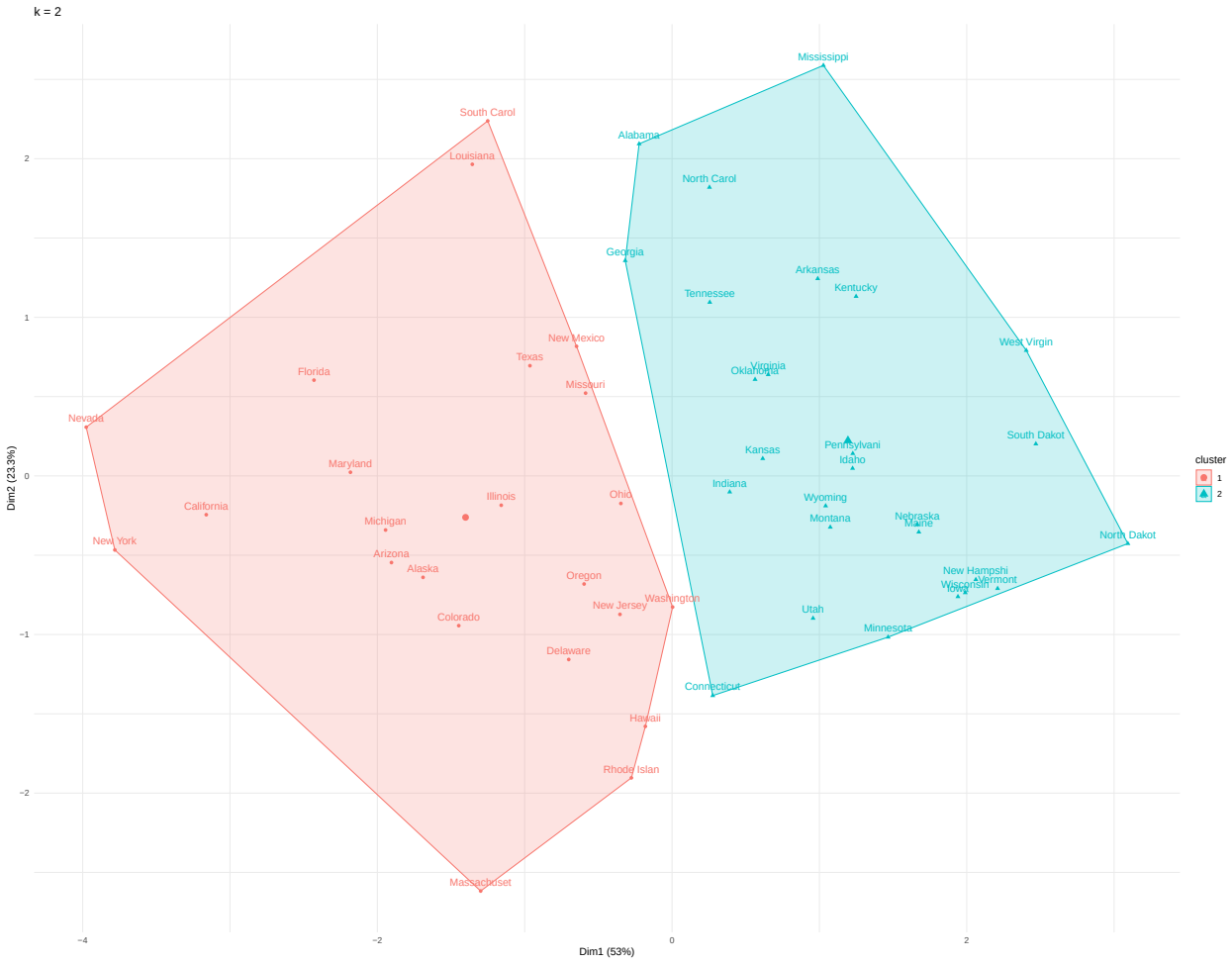
- Método Kmeans:

```
perc_tamKms_Cluster_k2
```

```
## [1] 0.46 0.54
```

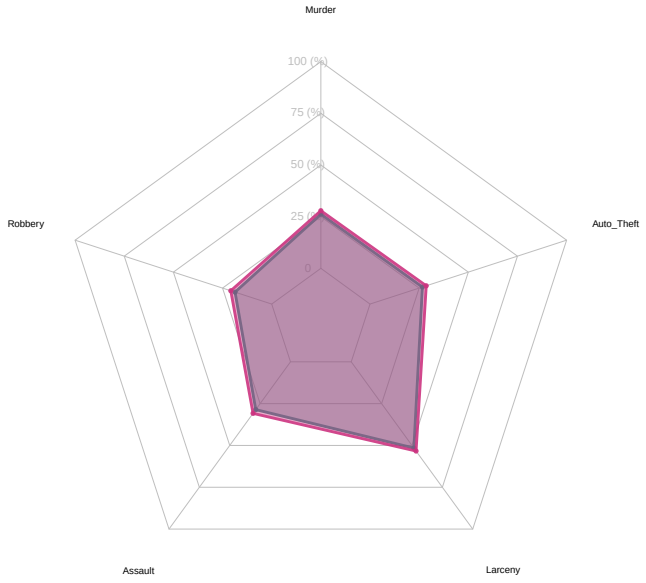
Obtenemos una primera visualización gráfica de los clusters definidos (K=2):

```
fviz_cluster(kmeans_k2, data=crime_df3, ellipse.type="convex") + theme_minimal() + ggtitle("k = 2")
```



Para comparar y revisar las diferencias de los centroides obtenidos mediante ambos métodos (jerárquico y Kmeans) vamos a representar los centroides medios (media de todos los centroides) de cada método.

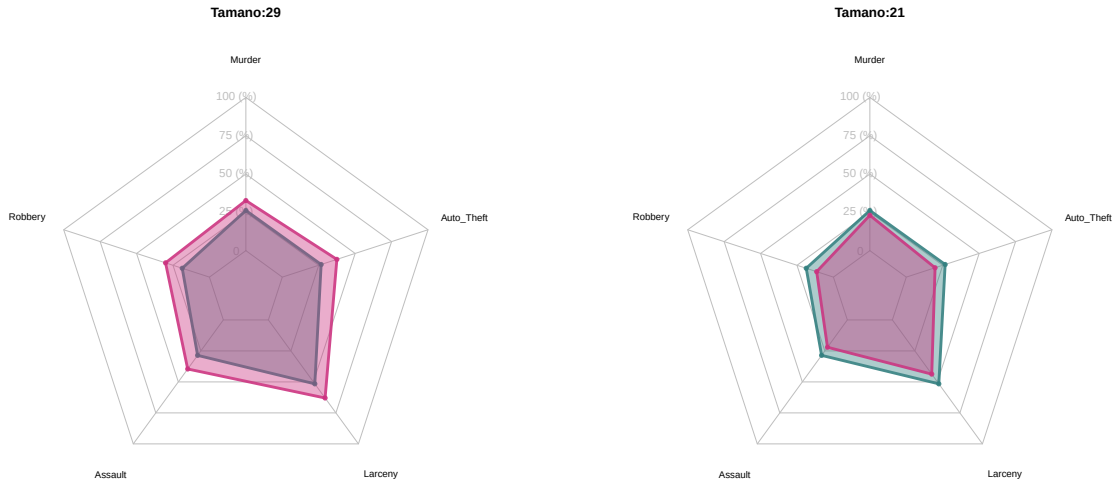
```
centrOpt_k2 <- kmeans_k2$centers

tamanoClusters <- sqldf("SELECT cluster_k2,
                        count(*) AS tamano_Cluster from asignJer_k2
                        GROUP BY cluster_k2")

centrOptRadar_k2 <- rbind(
  rep(7,5) ,
  rep(0,5) ,
  apply(centroJer_k2[,3:7], 2, mean),
  apply(centrOpt_k2, 2, mean),
  centrOpt_k2)

colors_border = c( rgb(0.2,0.5,0.5,0.9), rgb(0.8,0.2,0.5,0.9) , rgb(0.7,0.5,0.1,0.9) )
colors_in = c( rgb(0.2,0.5,0.5,0.4), rgb(0.8,0.2,0.5,0.4) , rgb(0.7,0.5,0.1,0.4) )

radarchart( as.data.frame(centrOptRadar_k2[c(1:4),])  , axistype=1 ,
            #custom polygon
            pcol=colors_border , pfcol=colors_in , plwd=4 , plty=1,
            #custom the grid
            cglcol="grey", cglty=1, axislabcol="grey", caxislabels=seq(0,1,5), cglwd=0.8,
            #custom labels
            vlcex=0.8,
          )
```



Como se puede apreciar no hay prácticamente diferencias entre los mismos.

Por último, vamos a representamos los centroides de los clusters obtenidos con el método Kmeans. Ésta es una forma interesante para analizar y describir los grupos de elementos obtenidos.

```
par(mfrow=c(1,2))

for (i in 5:nrow(centrOptRadar_k2))
{
  tamano <- tamanoClusters[i-4,2]

  radarchart( as.data.frame(centrOptRadar_k2[c(1:3,i),])  , axistype=1 ,
              #custom polygon
              pcol=colors_border , pfcol=colors_in , plwd=4 , plty=1,
              #custom the grid
              cglcol="grey", cglty=1, axislabcol="grey", caxislabels=seq(0,1,5), cglwd=0.8,
              #custom labels
              vlcex=0.8,
              title=paste0("Tamano:",tamano)
  )
}
```



Caso 2: 6 clusters (k=6)

Centroides de los clusters definidos por ambos métodos:

- Método jerárquico:

```
print.data.frame(centroJer_k6)
```

```
##   Cluster_k6 tamano_Cluster    Murder   Robbery  Assault  Larceny Auto_Theft
## 1          1              9 3.1723299 1.2839295 2.902101 3.048424   1.418517
## 2          2             13 1.7804569 1.7691289 2.120020 4.356619   2.658974
## 3          3              6 2.8318216 3.2086919 3.578279 5.046548   2.747322
## 4          4              8 2.0074900 1.0280303 1.558930 2.837357   1.357653
## 5          5             13 0.8235856 0.4833999 1.188684 3.366317   1.310117
## 6          6              1 0.8017029 1.9140095 2.310154 3.184009   5.895206
```

- Método Kmeans:

```
cbind(Cluster_k6=c(1:6),tamano_Cluster=kmeans_k6$size,kmeans_k6$centers)
```

```
##   Cluster_k6 tamano_Cluster    Murder   Robbery  Assault  Larceny Auto_Theft
## 1          1              8 3.0904355 1.4125866 3.065992 3.327502   1.544124
## 2          2             11 1.7985418 1.6800190 2.145661 4.652814   2.542708
## 3          3              6 2.8232012 3.4420479 3.515604 4.746143   2.838672
## 4          4              9 2.2959163 1.0531517 1.621231 2.564510   1.329810
## 5          5             14 0.8866761 0.5302859 1.232381 3.395409   1.306767
## 6          6              2 0.8663564 1.4465430 2.157540 3.550997   4.993681
```

Distribución porcentual de los elementos en cada grupo de ambos métodos:

- Método jerárquico:

```
perc_tamJer_Cluster_k6
```

```
##   tamano_Cluster
## 1           0.18
## 2           0.26
## 3           0.12
## 4           0.16
## 5           0.26
## 6           0.02
```

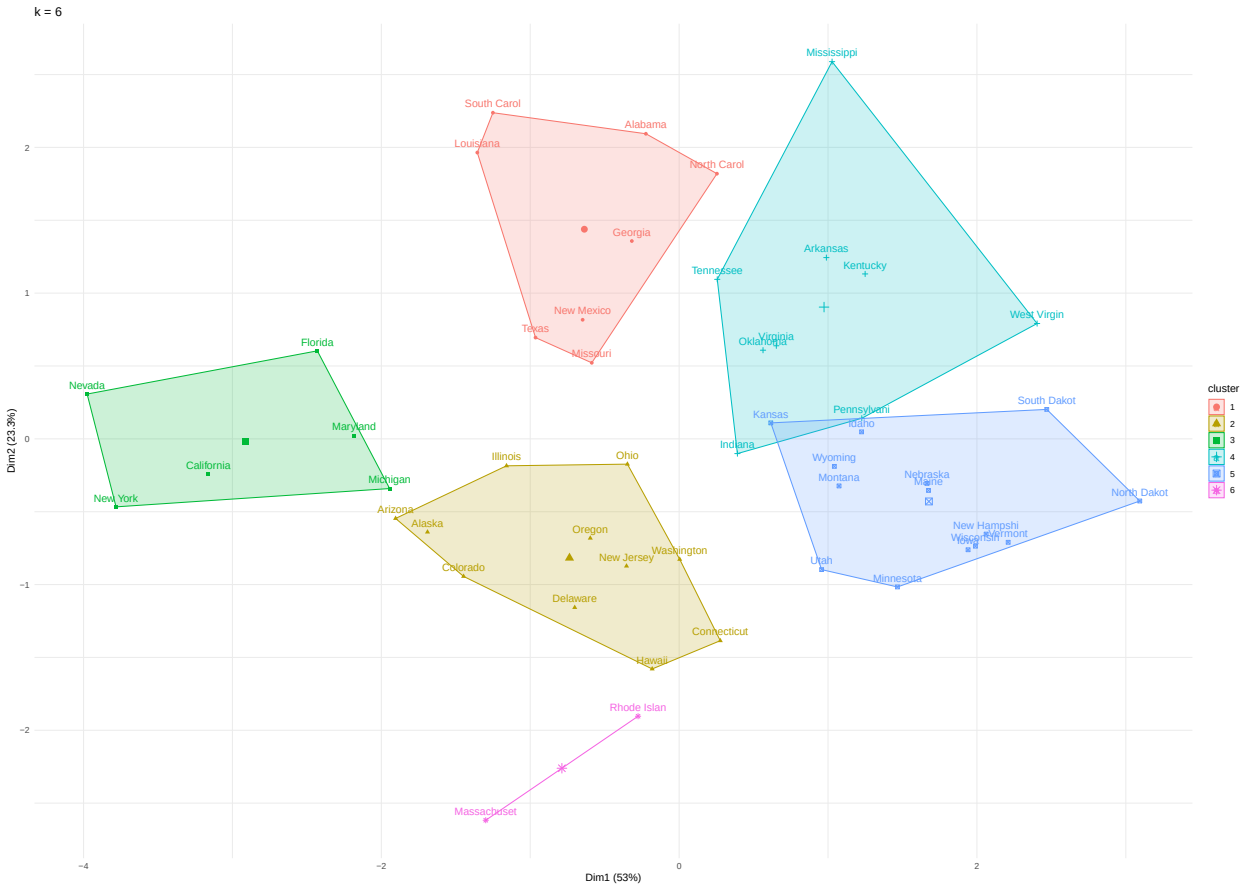
- Método Kmeans:

```
perc_tamKms_Cluster_k6
```

```
## [1] 0.16 0.22 0.12 0.18 0.28 0.04
```

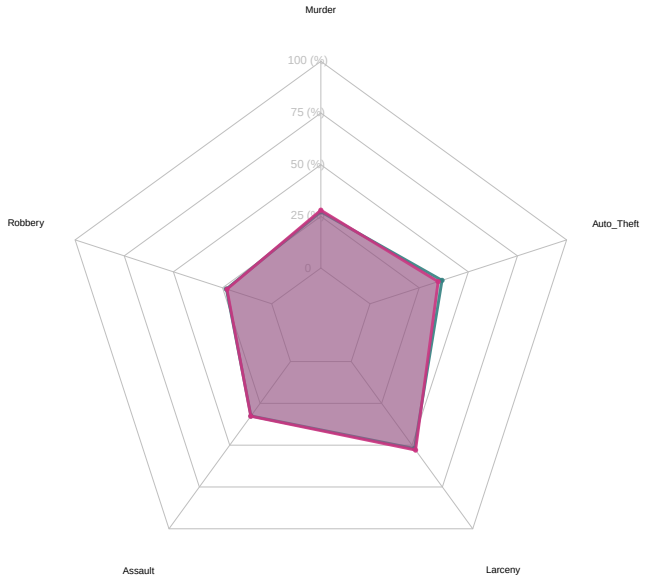
Obtenemos una primera visualización gráfica de los clusters definidos (K=6):

```
fviz_cluster(kmeans_k6, data=crime_df3, ellipse.type="convex") + theme_minimal() + ggtitle("k = 6")
```



Para comparar y revisar las diferencias de los centroides obtenidos mediante ambos métodos (jerárquico y Kmeans) vamos a representar los centroides medios (media de todos los centroides) de cada método.

```
centrOpt_k6 <- kmeans_k6$centers
tamanoClusters <- sqldf("SELECT cluster_k6,
                        count(*) AS tamano_Cluster from asignJer_k6
                        GROUP BY cluster_k6")

centrOptRadar_k6 <- rbind(
  rep(7,5), rep(0,5) ,
  apply(centroJer_k6[,3:7], 2, mean),
  apply(centrOpt_k6, 2, mean), centrOpt_k6)

colors_border = c( rgb(0.2,0.5,0.5,0.9), rgb(0.8,0.2,0.5,0.9) , rgb(0.7,0.5,0.1,0.9) )
colors_in = c( rgb(0.2,0.5,0.5,0.4), rgb(0.8,0.2,0.5,0.4) , rgb(0.7,0.5,0.1,0.4) )

radarchart( as.data.frame(centrOptRadar_k6[c(1:4),])  , axistype=1 ,
            #custom polygon
            pcol=colors_border , pfcol=colors_in , plwd=4 , plty=1,
            #custom the grid
            cglcol="grey", cglty=1, axislabcol="grey", caxislabels=seq(0,1,5), cglwd=0.8,
            #custom labels
            vlcex=0.8
    )
```

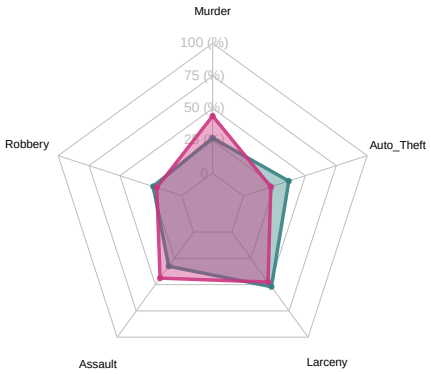


Como se puede apreciar no hay prácticamente diferencias entre los mismos.

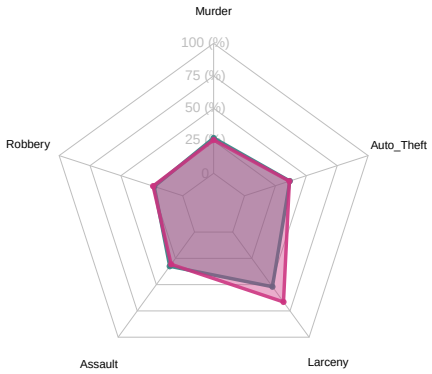
Por último, vamos a representamos los centroides de los clusters obtenidos con el método Kmeans. Ésta es una forma interesante para analizar y describir los grupos de elementos obtenidos.

```
par(mfrow=c(1,2))

for (i in 5:nrow(centrOptRadar_k6))
{
  tamano <- tamanoClusters[i-4,2]

  radarchart( as.data.frame(centrOptRadar_k6[c(1:3,i),])  , axistype=1 ,
              #custom polygon
              pcol=colors_border , pfcol=colors_in , plwd=4 , plty=1,
              #custom the grid
              cglcol="grey", cglty=1, axislabcol="grey", caxislabels=seq(0,1,5), cglwd=0.8,
              #custom labels
              vlcex=0.8,
              title=paste0("Tamano:",tamano)
  )
}
```

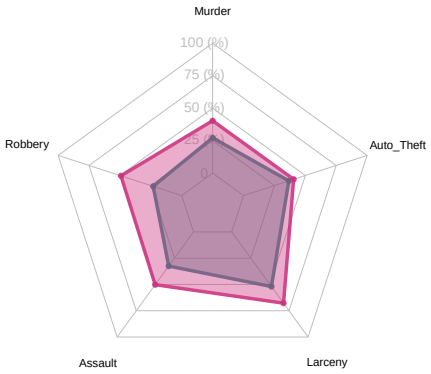
Tamano:9



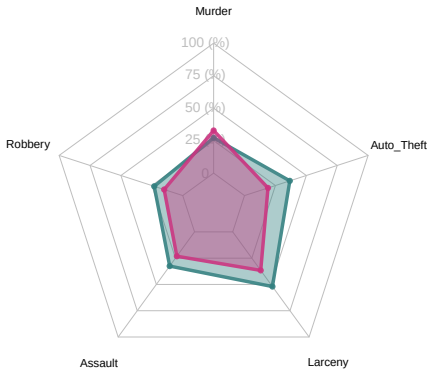
Tamano:13



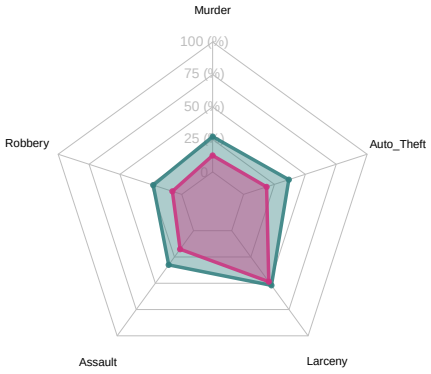
Tamano:6



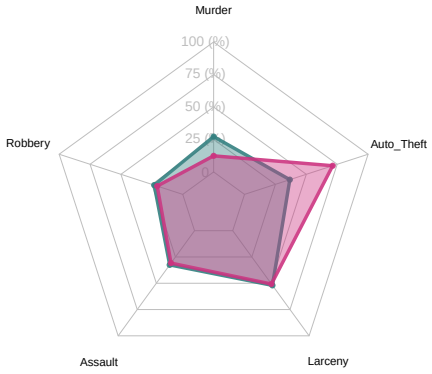
Tamano:8



Tamano:13



Tamano:1



3.6. ANÁLISIS DE LOS RESULTADOS

Analizando los resultados obtenidos de las 2 segregaciones propuestas, ambas son correctas pero se considera que quizás la mejor solución sea la de dividir los datos en 6 clusters (K=6).

Se toma esta decisión final, teniendo en cuenta el orden de magnitud del número de elementos del dataset (dividir los 50 elementos en sólo 2 grupos, quizás sea demasiado simple) y tras el análisis y visualización de los distintos clusters obtenidos en ambos casos, ya que los obtenidos en el segundo caso (k=6) se considera que permiten una diferenciación/clasificación mayor y bastante óptima de los elementos del dataset.

---

4. CONCLUSIONES

Es importante ser concientes de que para realizar el "clustering" se van a calcular y utilizar las distancias entre elementos por lo que será necesario un análisis y tratamiento preliminar de los datos (missing values, outliers, variables continuas y/o discretas, categorías, distancia entre categorías, transformaciones, estandarización...) teniendo en cuenta el caso en estudio. Posterior a ese análisis también será importante definir la forma de calcular esas distancias entre elementos.

Por último, para la determinación del tipo de algoritmo influyen muchos factores pero quizás los que se puedan considerar más influyentes son: el tamaño de los datos y el conocimiento sobre ellos y el área relacionada con los mismos. Ésto nos va a condicionar en muchos casos a tomar un método de un tipo u otro:

- Metodos jerárquicos - se tienen pocos datos datos.
- Métodos de optimización - se tienen muchos datos, se cuenta con orden aproximado para el número de clusters.

Un método que combina las ventajas de ambos tipos es el método bietápico y es una muy buena estrategia siempre y cuando no se tengan condicionantes específicos de los datos o externos del cliente. Consiste en aplicar primero un método jerárquico para conseguir información sobre el posible número de clusters óptimos para posteriormente, empleando esa información, aplicar un método de optimización para resolver el problema de clustering.

Todo lo mencionado anteriormente puede servir como guía para recordar elementos importantes del análisis, pero a fin de cuentas el "clustering" es un análisis de tipo no supervisado (no existe una variable "target" que nos permita definir un error de forma exacta) y por tanto es subjetivo. Depende de muchos factores, consideraciones particulares, conocimiento del área/sector del caso y a veces incluso condicionantes externos, por ello lo más importante es realizar el análisis teniendo en cuenta toda esa información, centrándose en el caso concreto de estudio.