Clustering en R – parte I

Estudio de los algoritmos de clustering k-Means y clustering jerárquico, de tipo aglomerativo, con varios ejemplos.

Ejercicio 1: datos de prueba en dos dimensiones.
Se le proporciona un fichero de nombre "toy_example.txt" que contiene dos columnas con valores numéricos. Representan puntos en el plano y queremos agruparlos usando la distancia euclidea (que es la distancia por defecto).

- Cargue los datos en una variable de nombre puntos que será un dataframe con dos variables o atributos, de nombres Xs e Ys, y tantas observaciones como puntos vienen codificados en el fichero.

```
puntos<-read.delim("toy_example.txt", sep = "\t", head = FALSE)
colnames(puntos)<-c("Xs","Ys")
```

  Inspeccione (funciones head, str, dim) y represente geométricamente (función plot) la variable datos.

- La función para aplicar el **algoritmo k-Means** se llama "**kmeans**". Teclee en la consola help("kmeans") y estudie como aplicarla.
  Aplique el k-Means al dataframe puntos usando tres centros de masa y tomando el parámetro nstar con valor 20, almacene el resultado en una nueva variable de nombre km_puntos.
  - Inspeccione el resultado con la función summary
  - ¿Qué devuelve km_puntos$cluster? ¿Y si hace print(km_puntos)?
  - Visualice el resultado usando la función plot

```
plot(puntos,col=km_puntos$cluster, main="Tres clusters")
```

- Estudiemos la aleatoriedad subyacente en el funcionamiento del k-Means.
  - Ejecute seis veces el algoritmo k-Means, y visualice el resultado, con el parámetro nstar con valor igual a 1.
  - Repita la operación con el valor de nstar igual a 20.
  - Estudie en la ayuda qué significan:
    - nstar
    - tot.withinss

- Estudiemos una posible forma de elegir el valor del parámetro k:
  - Aplique el algoritmo quince veces variando, incrementalmente, el valor de los centros de masa. Es decir, el valor de center varía de uno a quince. Construya un vector, de nombre *vector_compactacion*, que en cada paso almacene el valor del parámetro tot.withinss de cada resultado.
  - Represente el valor de k de cada iteración frente al valor de la compactación interna de cada cluster obtenida.

```
plot(1:15, vector_compactacion, type = "b", xlab = "Numero de clusters",
     ylab = "Compactacion")
```

- ¿Para qué valor de k se produce un cambio significativo en la gráfica obtenida?

- El **algoritmo de clustering jerárquico de tipo aglomerativo** se llama "**hclust**". Teclee en la consola help("hclust") y estudie su funcionamiento.
  - Teclee hclust(puntos) y verá que da un error, ¿por qué?
  - Calcule la matriz de distancia con la función dist(puntos) y pásele el resultado a la función hclust. Almacene el resultado en una variable d enombre hclust_aux
  - ¿Qué devuelve print(hclust_aux)?
  - ¿Qué devuelve hclust_aux$cluster? ¿Por qué?
  - Construcción de los clústeres a partir del dendograma:
    - Visualice el hclust_aux usando la función plot.
    - Si quiere cortar el dendograma según la altura: cutree(hclust_aux,h= …)
    - Si quiere cortarlo según el número de clústeres a obtener: cutree(hclust_aux,k= …)

- Estudio del método de linkage usado, es decir, cómo se calcula la distancia entre los distintos clústeres al realizar el algoritmo aglomerativo:
  - Complete: la distancia elegida es el máximo de las distancias entre los puntos
  - Average: la distancia es la distancia media.
  - Single: la distancia es la distancia mínima.
    Estudie el comportamiento de los tres métodos de linkage:

```
hclust.complete <- hclust(dist(puntos), method = "complete")
plot(hclust.complete, main = "Distancia maxima: complete", h=-2)
```

- El escalado de los datos, o si éstos están medidos en una misma magnitud, tiene una importancia vital a la hora de calcular las distancias entre los puntos. ¿Están escalados los datos del ejemplo usado?

```
colMeans(puntos)
puntos_escalado<-scale(puntos)
colMeans(puntos)
```

- Comparación entre los resultados de aplicar el k-Means y el clustering jerárquico

```
km_puntos # Resultado del k-Means

hclust_aux # No son directamente los clusteres sino el dendograma
corte_hclust_aux<-cutree(hclust_aux,k=3)

# Comparamos los metodos:
table(km_puntos$cluster,corte_hclust_aux)
```

  - ¿Cómo interpreta los resultados?
  - ¿Qué algoritmo funciona mejor?

Ejercicio 2: datos de iris.
El dataset de iris es un famoso dataset que viene por defecto con la librería de datos de R. Mide los sépalos y pétalos de una flor llamada iris. Suele ser usado frecuentemente en estudios de Data Science. Aunque están disponibles poniendo tan sólo iris en la consola, se les proporciona como un fichero de texto para que tenga que cargarlos usted mismo (…`sep="\t", header=T, row.names = "Filas"`).

- Como son unos datos etiquetados, con tres clases, construya una variable datos, que no tenga esas clases, y otra que se llame datosAux que tenga el dataset original.

```
datos #Observo los datos
datosAux<-datos
datos$species <-NULL #Quitamos la variable species (nominal)
datos #Observo los datos
```

- Aplique el **algoritmo k-Means** con tres centros de masa y nstar=20
- Estudie los resultados y represente los centros de masa.
- Estudie los resultados en según la clase a la que pertenece cada uno:
  `table(datosAux$species,kc$cluster)`
- Cada punto tiene cuatro coordenadas pero queremos representarlos en el plano, ¿qué coordenadas son las más adecuadas?
  Pruebe de esta manera:
  `plot(datosAux[c("sepal_length","sepal_width")], col=kc$cluster)`

- Aplique el algoritmo de clustering jerárquico aglomerativo usando la distancia euclídea
  - Cálcule la matriz de distancia: dist()
  - Aplique el algoritmo: hclust()
  - Visualice el dendograma: plot() sobre el resultado anterior
  - Corte según la altura o el número de clústeres, por ejemplo:
    `id.grupos1<-cutree(dendrograma, k=4)`
  - Represente los resultados:
    `plot(datosAux[c("sepal_length","sepal_width")], col=id.grupos1)`

Ejercicio 3: datos de tipo geográficos.
Usando la librería ggmap, que permite averiguar la posición geográfica de una ciudad dada, entre otras cosas, vamos a agrupar las ciudades europeas según su posición en el mapa. Se construirá un vector con todas las ciudades y se calculará su latitud y longitud a través de la librería citada. Una vez construido el dataset se estudiará mediante un algoritmo de clustering.

```
#install.packages("ggmap") #instala librería
library(ggmap) #carga librería

#Vector con todas las capitales de europa

capitales <- c("Albania, Tirana", "Andorra, Andorra la Vella", "Armenia, Yerevan",
               "Austria, Vienna", "Azerbaijan, Baku", "Belarus, Minsk",
               "Belgium, Brussels", "Bosnia and Herzegovina, Sarajevo",
               "Bulgaria, Sofia", "Croatia, Zagreb", "Cyprus, Nicosia",
               "Czech Republic, Prague", "Denmark, Copenhagen", "Estonia, Tallinn",
               "Finland, Helsinki", "France, Paris", "Germany, Berlin",
```

```
                "Greece, Athens", "Georgia, Tbilisi", "Hungary, Budapest",
                "Iceland, Reykjavik", "Italy, Rome", "Latvia, Riga",
                "Kazakhstan, Astana", "Liechtenstein, Vaduz", "Lithuania, Vilnius",
                "Luxembourg, Luxembourg", "Macedonia, Skopje", "Malta, Valletta",
                "Moldova, Chisinau", "Monaco, Monaco-Ville", "Montenegro, Podgorica",
                "Netherlands, Amsterdam", "Norway, Oslo", "Poland, Warsaw",
                "Portugal, Lisbon", "Republic of Ireland, Dublin",
                "Romania, Bucharest", "Russia, Moscow", "San Marino, San Marino",
                "Serbia, Belgrade", "Slovakia, Bratislava", "Slovenia, Ljubljana",
                "Spain, Madrid", "Sweden, Stockholm", "Switzerland, Bern",
                "Turkey, Ankara", "Ukraine, Kiev", "United Kingdom, London",
                "Vatican City, Vatican City"
)

capitales #muestra contenido

#Calcula la longitud y latidud de cada capital (usa "ggmap")
#datos <- geocode(capitales) # Consulta con: "google" for Google
datos <- geocode(capitales, source = "dsk") #Consulta con: "dsk" for Data Science
Toolkit

datos #Observa dataset
rownames(datos) <- capitales #Incluye la columna con los nombres
datos #Observa dataset
…
```

Realice el correspondiente estudio de clustering.
Para representar los resultados puede, entre otras opciones, usar la siguiente llamada:

```
plot(datos, cex=0, xlim=c(-30,75)) # (mejorado)
text(datos, rownames(datos), cex=0.6, col=groups) # anyade el texto
```

Siendo datos los datos originales y groups un resultado de un algoritmo de clustering-