Análisis multivariante - PEC 2

Análisis de Conglomerados

Juan José Rodríguez Aldavero
June 12, 2020

Contents

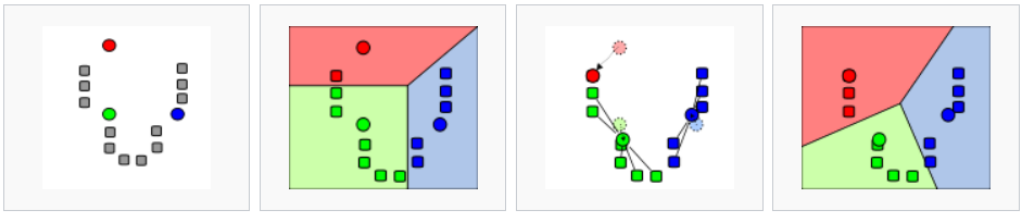
1 Aplicación de técnicas de clustering

Los procedimientos de *clustering* son un conjunto de algoritmos cuya finalidad es encontrar patrones o grupos denominados *clústers* dentro de un conjunto de datos determinado. Estas particiones se determinan de tal forma que todos los individuos que las componen comparten características en común. Dentro del lenguaje de *machine learning*, los algoritmos de clústering entrarían dentro de la categoría de aprendizaje no-supervisado, ya que no emplean en ningún momento el conjunto de categorías o etiquetas reales de los datos (si es que estas existen). En este apartado aplicaremos un conjunto de técnicas de clustering al conjunto de datos AM20 proporcionado por la asignatura. En particular, los métodos empleados serán:

A. **Algoritmo K-Means**
Este método determina un conjunto de G puntos en el conjunto de datos (normalmente aleatorios) y asigna cada uno de los elementos del conjunto de datos al punto más cercano empleando la distancia euclídea. Después, se optimiza de manera iterativa la distribución de los G puntos minimizando una métrica dada, normalmente la suma de los cuadrados dentro de cada grupo (WCSS):

$$\arg\min_{\mathrm{G}} \sum_{i=1}^{k} \sum_{\mathrm{x}_j \in G_i} \|\mathrm{x}_j - \boldsymbol{\mu}_i\|^2$$

donde $\boldsymbol{\mu}_i$ es la media de puntos en el grupo G.



1) *k* centroides iniciales (en este caso *k*=3) son generados aleatoriamente dentro de un conjunto de datos (mostrados en color).

2) *k* grupos son generados asociándole el punto con la media más cercana. La partición aquí representa el diagrama de Voronoi generado por los centroides.

3) EL centroide de cada uno de los *k* grupos se recalcula.

4) Pasos 2 y 3 se repiten hasta que se logre la convergencia.

Figure 1.1: *Esquema del algoritmo K-Means. Fuente: Wikipedia*

B. **Método jerárquico (librería *pvclust*)**
Los métodos jerárquicos intentan construir el conjunto de G grupos mediante un procedimiento iterativo que o bien une un conjunto de grupos pequeños para formar grupos más grandes (método aglomerativo), o bien realiza el procedimiento inverso y fragmenta el conjunto de datos inicial en los G grupos finales. Normalmente el criterio empleado es

la distancia euclídea entre los elementos del conjunto, esto es, la matriz de distancias. El procedimiento se suele representar gráficamente mediante un dendograma. Para aplicarlo al ejercicio, emplearemos el método *hclust* incluido por defecto en la librería *stats*.

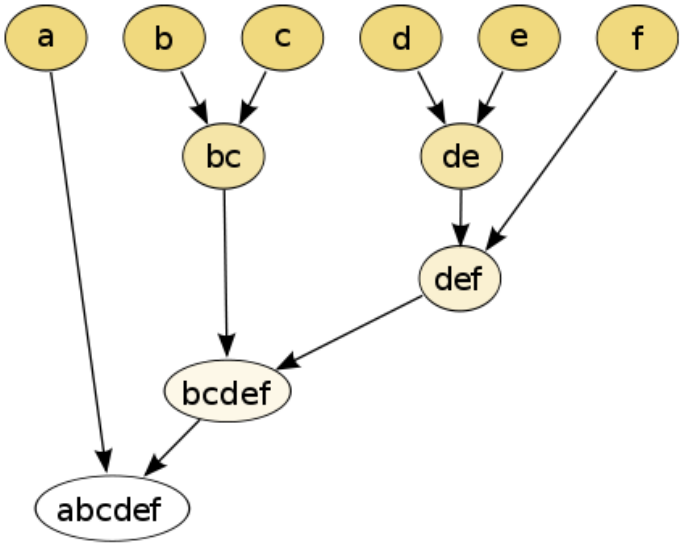


Figure 1.2: *Ejemplo esquemático de un dendograma para un algoritmo aglomerativo. Fuente: Wikipedia*

C. **Método basado en modelos (librería *Mclust*)**
Los procedimientos de clustering basados en modelos consisten en ajustar cada uno de los G clusters en términos de distribuciones de probabilidad hipotéticas, y posteriormente ajustar estas distribuciones minimizando una métrica que depende de los parámetros de las distribuciones de probabilidad. Dentro de esta categoría, los métodos más conocidos son los modelos de mezcla gaussiana (*Gaussian Mixture Models*), que son un subtipo dentro de los modelos de mezcla (*Mixture models*) eintentan modelar los datos a partir de distribuciones gaussianas. Para aplicarlo al ejercicio, emplearemos el método *Mclust* dentro de la librería *mclust*.

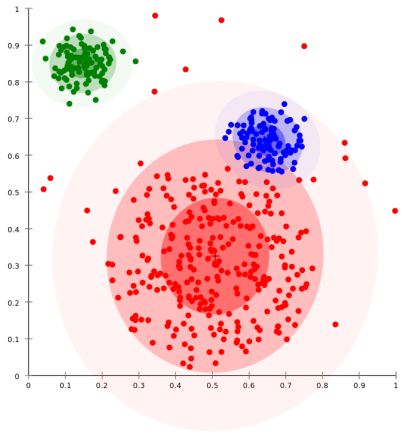


Figure 1.3: *Ejemplo de clústering mediante distribuciones gaussianas 2d. Fuente: Wikipedia*

1.a Aplicar técnicas de clustering con cardinalidad 10

Comenzamos el ejercicio preparando el conjunto de datos para asegurarnos de que los métodos de clustering funcionan correctamente. Para ello, procedemos a eliminar las variables linealmente dependientes y a estandarizar el conjunto de datos para que las diferencias en escala entre las variables no afecten de manera decisiva a la formación de los clústers. Como este conjunto de datos fue utilizado para la anterior práctica de la asignatura, conocemos de antemano que la variable E es una combinación lineal de las variables H y J por tanto debe ser eliminada del conjunto de datos. Además, también sabemos que las variables H e I presentan un grado muy elevado de kurtosis y asimetría. Esto hace que el procedimiento de estandarización de variables sea necesario para eliminar la distorsión introducida por estas variables.

```
/*Eliminamos la variable E y estandarizamos los datos*/
AM20 <- subset(AM20, select=-c(E))
AM20 <- scale(AM20)
```

Una vez hecho esto, aplicamos los distintos procedimientos. Lo que vamos a hacer en todos los algoritmos es similar: primero ajustamos un modelo al conjunto de datos para así extrar el conjunto de etiquetas que categorizan a cada elemento. Después, vamos a extraer las direcciones o coordenadas de los centroides de los grupos para así posteriormente poder construir una matriz de datos modificada donde las coordenadas de cada elemento se identifican con las coordenadas del centroide del grupo al que pertenecen. Esta última matriz nos permitirá calcular la pérdida de información a partir de la métrica

$$IL = \sum_{i=0}^{n} \sum_{j=0}^{p} \left(d_{ij} - \hat{d}_{ij} \right)^2$$

proporcionada por el ejercicio (diferencias al cuadrado entre las matrices original y modificada).

1.a.1 Algoritmo K-Means

Comenzamos con el algoritmo de K-Means para una cardinalidad de 10 elementos, empleando el método incluido por defecto en R (*kmeans*).

```
/*Ajustamos un modelo K-Means de 10 elementos*/
kmeans.modelo <- kmeans(AM20, 10)
/*Extraemos las etiquetas para cada elemento*/
kmeans.etiquetas <- data.frame(kmeans.modelo$cluster)
names(kmeans.etiquetas)[1] <- "etiquetas"
/*Extraemos las coordenadas de cada centroide junto con la etiqueta*/
kmeans.centroides <- data.frame(kmeans.modelo$centers) %>%
    mutate(etiquetas = row_number())
```

Para calcular la matriz de distancias del conjunto modificado sustituimos las coordenadas de cada punto por las coordenadas del centro del grupo al que pertenecen. Para ello, se pegan las coordenadas de los centroides usando como clave primaria la etiqueta que le corresponde a cada elemento. La librería *dplyr* incluye el procedimiento *left_join* que resulta especialmente útil para esto:

```
/*Creamos la matriz de distancias modificada*/
kmeans.AM20 <- subset(left_join(kmeans.etiquetas, kmeans.centroides,
    by="etiquetas"), select = -c(etiquetas))
```

Una vez disponemos de esta nueva matriz *kmeans.AM20*, calcular la pérdida de información es muy sencillo:

```
kmeans.IL <- sum((AM20 - kmeans.AM20)^2)
```

Utilizando este procedimiento, obtenemos una pérdida de información total de

$$IL_{kmeans} = 3.820$$

Podemos visualizar gráficamente el resultado para las primeras dos componentes principales (ya que hay que visualizar alguna proyección bidimensional del resultado) usando los métodos *clusplot* y *plotcluster* de las librerías *cluster* y *fpc* respectivamente.

```
library(cluster)
clusplot(AM20, kmeans.modelo$cluster, color=TRUE, shade=TRUE, labels=2,
    lines=0)
```

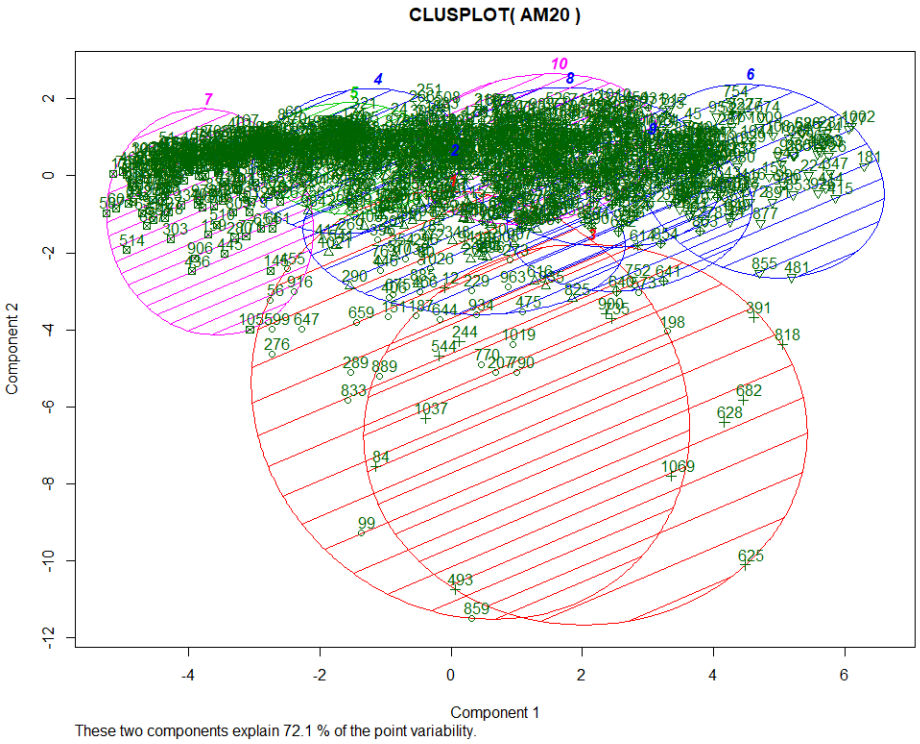


Figure 1.4: *Distribución de las regiones de clasificación en las dos primeras componentes principales*

```
library(fpc)
plotcluster(AM20, kmeans.modelo$cluster)
```

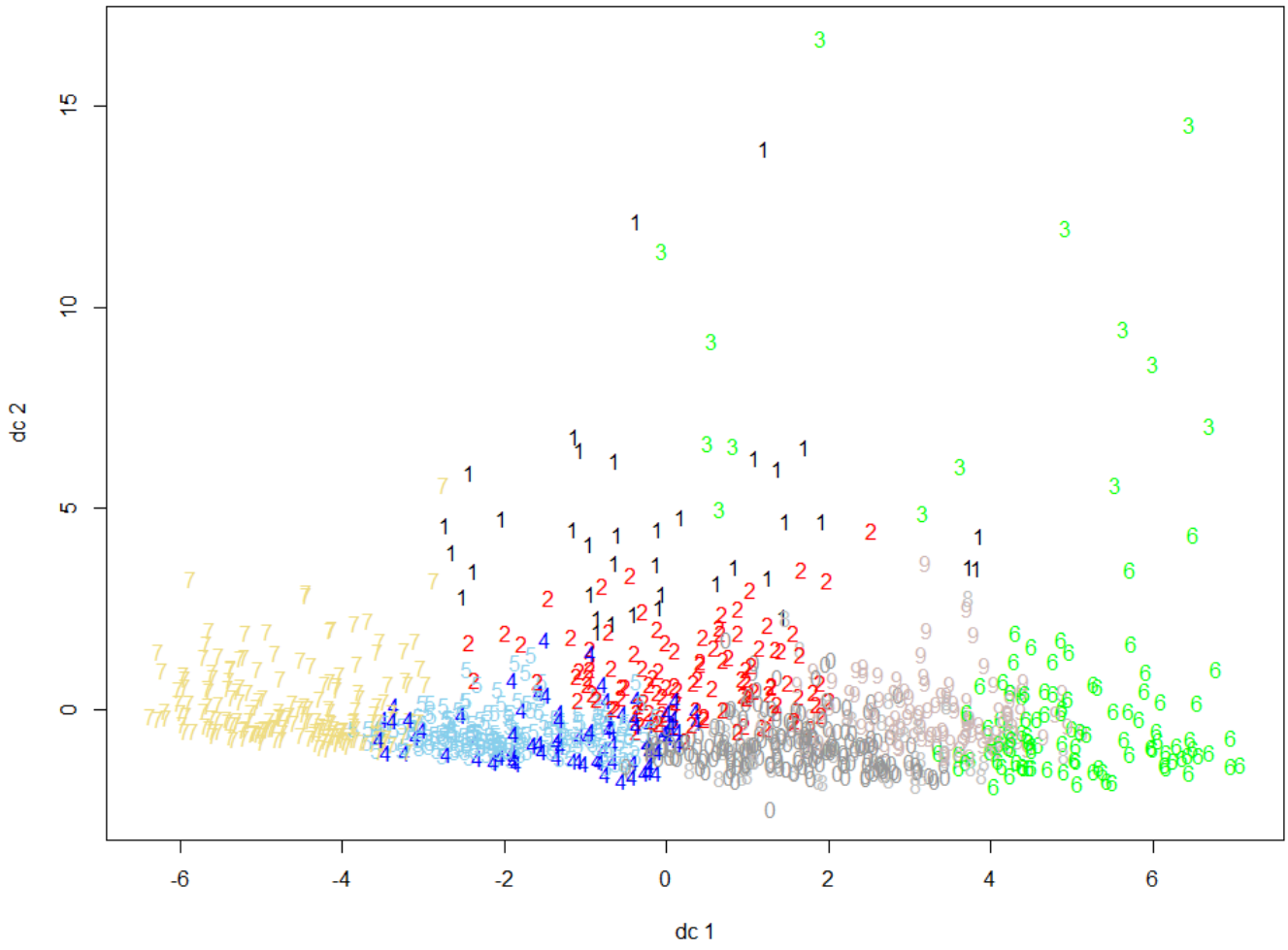


Figure 1.5: *Resultado de la clasificación para KMeans*

1.a.2 Método jerárquico

Continuamos este procedimiento para el método jerárquico. Para ello, empleamos el método *hclust* incluido por defecto en las librerías de R. Este método funciona sobre matrices de distancia en lugar de conjuntos de datos, pero el resto del procedimiento será similar.

```
library(hclust)
/*Creamos la matriz de distancias*/
```

```
D <- dist(AM20, method="euclidean")
/*Repetimos el procedimiento anterior*/
jerarq.modelo <- hclust(D, method = "ward")
jerarq.etiquetas <- data.frame(cutree(jerarq.modelo, k=10))
names(jerarq.etiquetas)[1] <- "etiquetas"
jerarq.centroides <- data.frame(apply(AM20, 2, function (x) tapply (x,
    aux, mean)))%>% mutate(etiquetas = row_number())
```

Una vez hechos todos estos cálculos auxiliares podemos volver a calcular la matriz de distancias modificada:

```
/*Creamos la matriz de distancias modificada*/
jerarq.AM20 <- subset(left_join(jerarq.etiquetas, jerarq.centroides,
    by="etiquetas"), select = -c(etiquetas))
```

Y el valor de la pérdida de información:

```
jerarq.IL <- sum((AM20 - jerarq.AM20)^2)
```

Con un valor de:

$$IL_{jerarq} = 4.391$$

En este caso, podemos visualizar gráficamente el resultado en forma de dendograma. La parte inferior del dendograma mostrará los 1080 elementos y a medida que se asciende se verá el resultado del algoritmo uniéndolos de manera progresiva hasta alcanzar un único elemento.

```
plot(jerarq.modelo)
rect.hclust(jerarq.modelo, k=10, border="red")
```

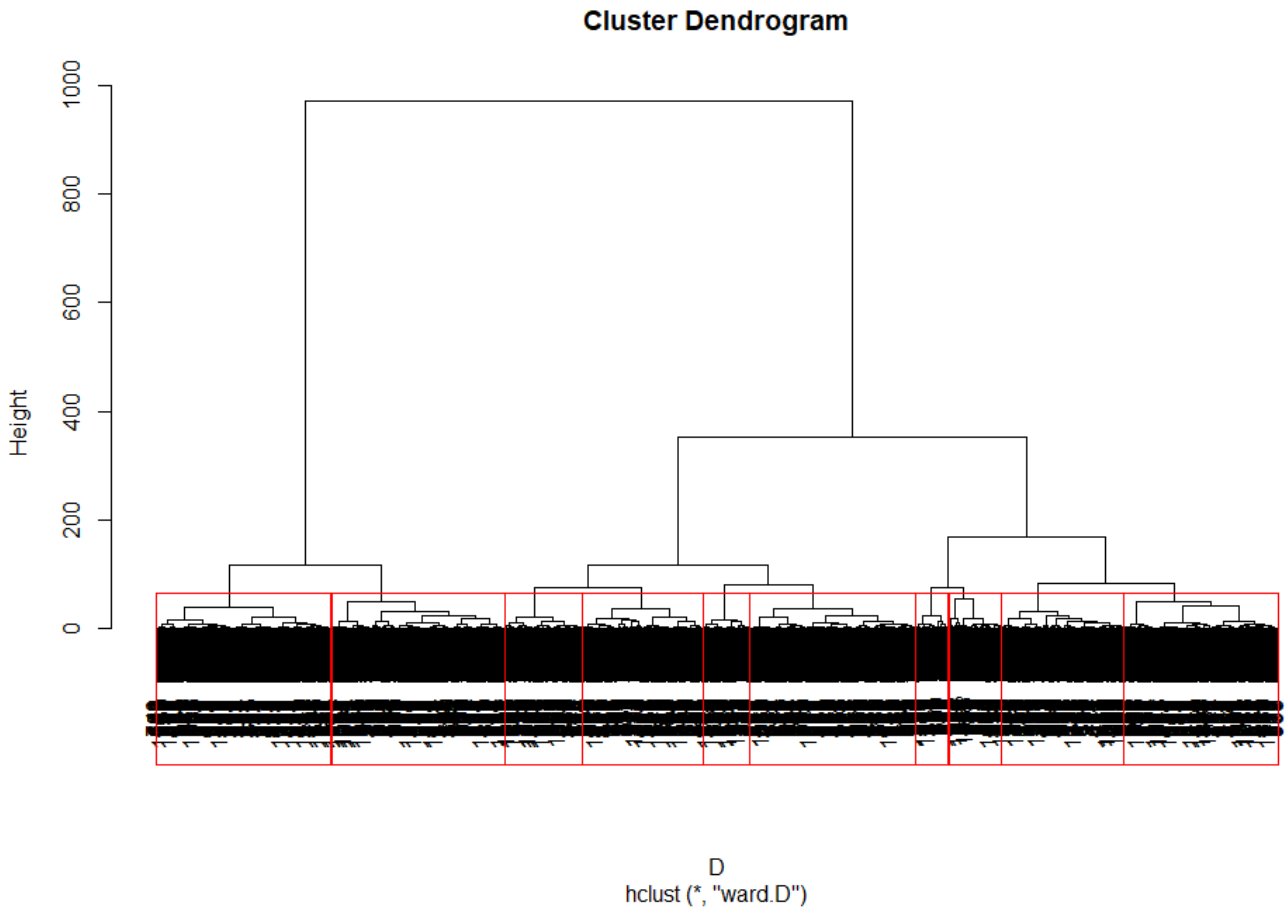


Figure 1.6: *Dendograma del algoritmo de clustering jerárquico*

De nuevo, representamos las regiones de clasificación y el resultado de los algoritmos:

```
etiquetas <- cutree(jerarq.modelo, k=10)
clusplot(AM20, etiquetas, color=TRUE, shade=TRUE, labels=2, lines=0)
```

```
plotcluster(AM20, etiquetas)
```

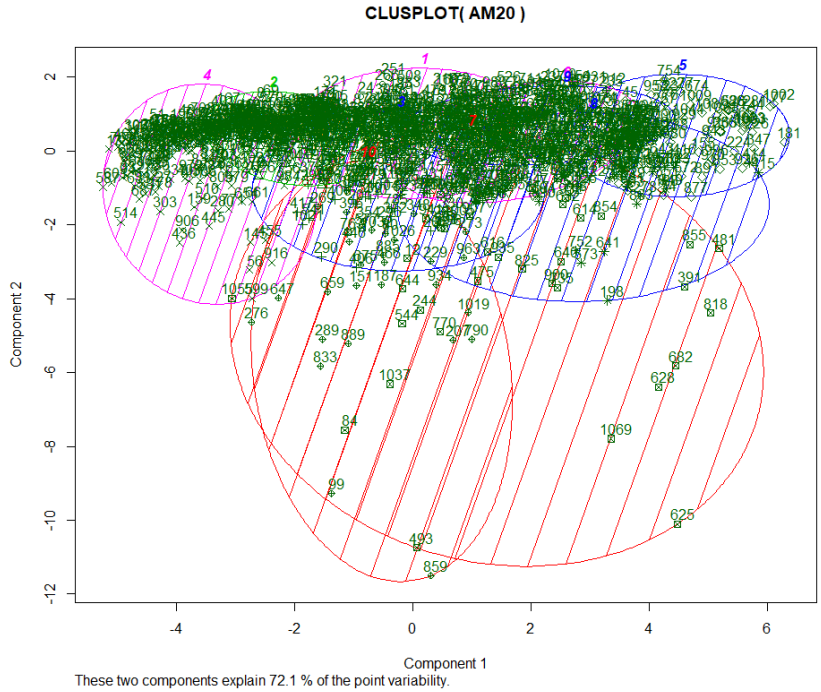


Figure 1.7: *Distribución de las regiones de clasificación en las dos primeras componentes principales para el agrupamiento jerárquico*

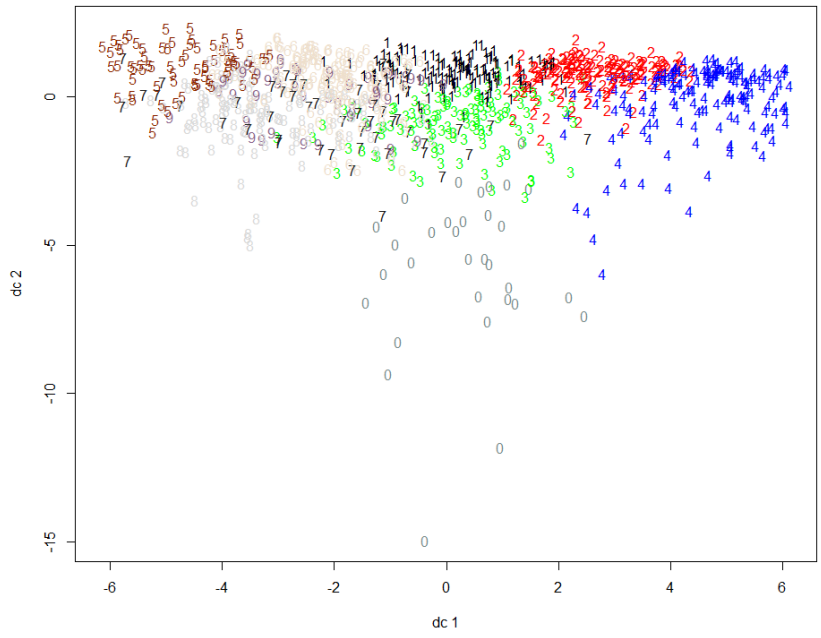


Figure 1.8: *Resultado de la clasificación para el agrupamiento jerárquico*

1.a.3 Método basado en modelos (Gaussian Mixture Model)

Por último, continuamos con la práctica aplicando un método de aglomeración basado en modelos. Estos métodos se caracterizan como se ha comentado anteriormente por ajustar un conjunto de G distribuciones de probabilidad teóricas, normalmente gaussianas, sobre el conjunto de datos. En este caso, se empleará el método más habitual basado en distribuciones gaussianas. Emplearemos la librería *mclust* que incluye por defecto estos procedimientos.

```
library(mclust)
/*Repetimos el procedimiento anterior*/
gauss.modelo <- Mclust(AM20, G=10)
gauss.etiquetas <- data.frame(gauss.modelo$classification)
names(gauss.etiquetas)[1] <- "etiquetas"
gauss.centroides <- data.frame(apply(AM20, 2, function (x) tapply (x,
    aux, mean)))%>% mutate(etiquetas = row_number())
```

Aplicando este código y calculando la matriz de coordenadas de los 10 centros podemos calcular la pérdida de información del mismo modo que en los casos anteriores.

```
/*Creamos la matriz de distancias modificada*/
gauss.AM20 <- subset(left_join(gauss.etiquetas, gauss.centroides,
    by="etiquetas"), select = -c(etiquetas))
/*Calculamos la perdida de informacion*/
gauss.IL <- sum((AM20 - gauss.AM20)^2)
```

El resultado es:

$$IL_{gauss} = 7.752$$

De nuevo, representamos las regiones de clasificación y el resultado de los algoritmos:

```
clusplot(AM20, gauss.modelo$classification, color=TRUE, shade=TRUE,
    labels=2, lines=0)
```

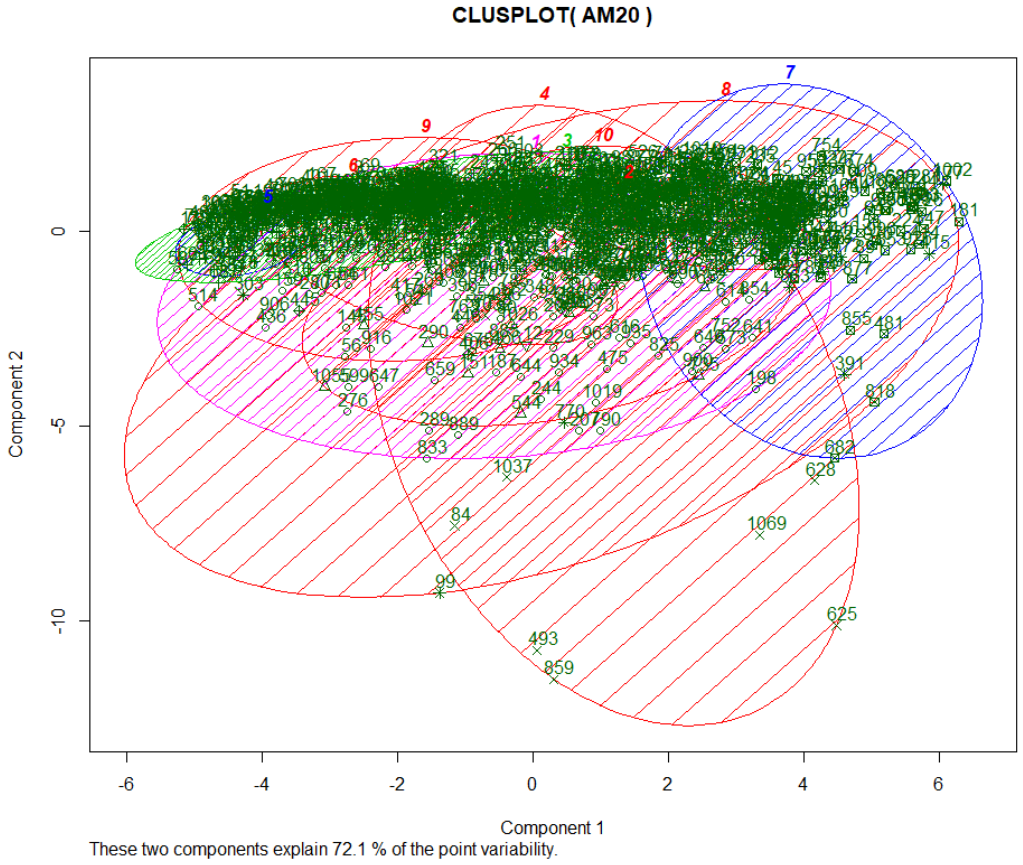


Figure 1.9: *Distribución de las regiones de clasificación en las dos primeras componentes principales para Gaussian Mixture Model*

Vemos como en esta ocasión las regiones de clasificación son elipses perfectas (distribuciones gaussianas multivariantes).

```
plotcluster(AM20, gauss.modelo$classification)
```

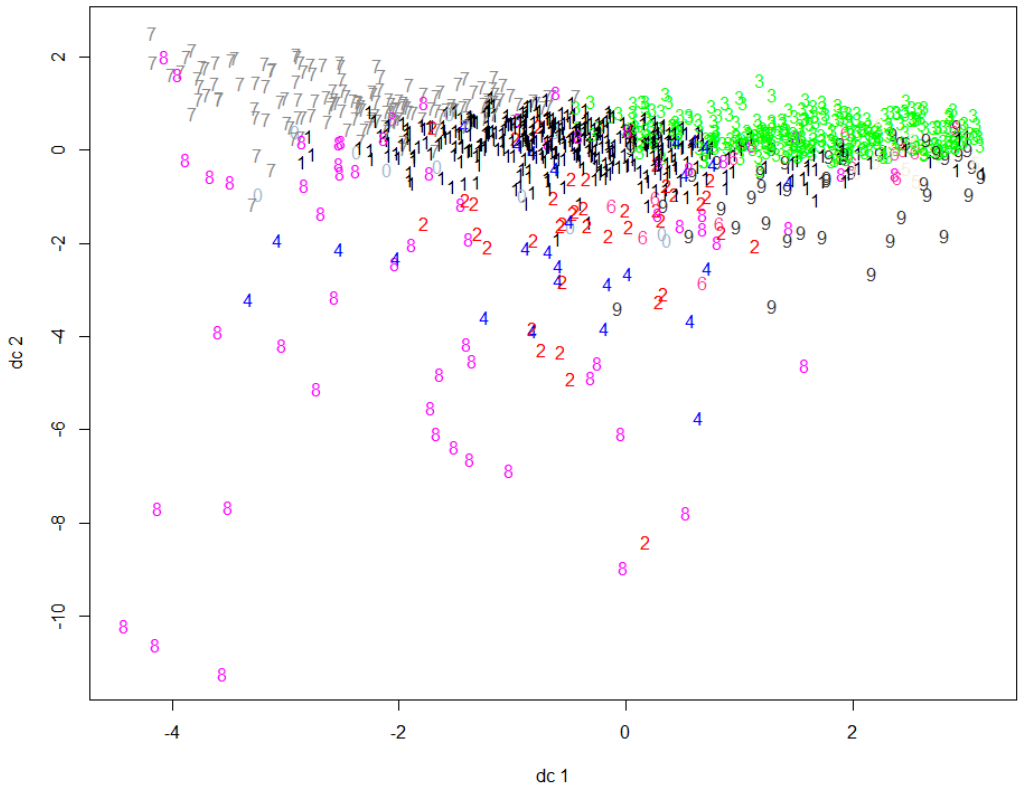


Figure 1.10: Resultado de la clasificación para Gaussian Mixture Model

1.b Aplicar técnicas de clustering con cardinalidad variable entre 10 y 30

Repetimos los tres procedimientos anteriores pero en esta ocasión variando el número de clústers entre 10 y 30. En particular, calcularemos la pérdida de información para $G = \{10, 15, 20, 25, 30\}$ para que de esta manera podamos estudiar con precisión la relación entre pérdida de información y cardinalidad en la sección siguiente. Esto se hará creando un script a partir del código elaborado para los ejercicios anteriores, y ejecutándolo para los distintos valores de G. Dicho script se incluye en el anexo ejercicio. Una vez hecho esto, los resultados son los siguientes:

	10	**15**	**20**	**25**	**30**
K-Means	3.820	3.260	2.907	2.623	2.417
Jerárquico	4.391	3.665	3.238	2.951	2.714
Gaussian Mixture Model	7.752	5.963	5.557	5.185	4.786

2 Análisis de los resultados

2.a Analizar la relación entre la pérdida de información y la cardinalidad

A partir de la tabla de datos anterior podemos ver una fuerte relación entre la pérdida de información causada por los métodos de aglomeración y la cardinalidad del método. Si representamos los datos en un gráfico de columnas,

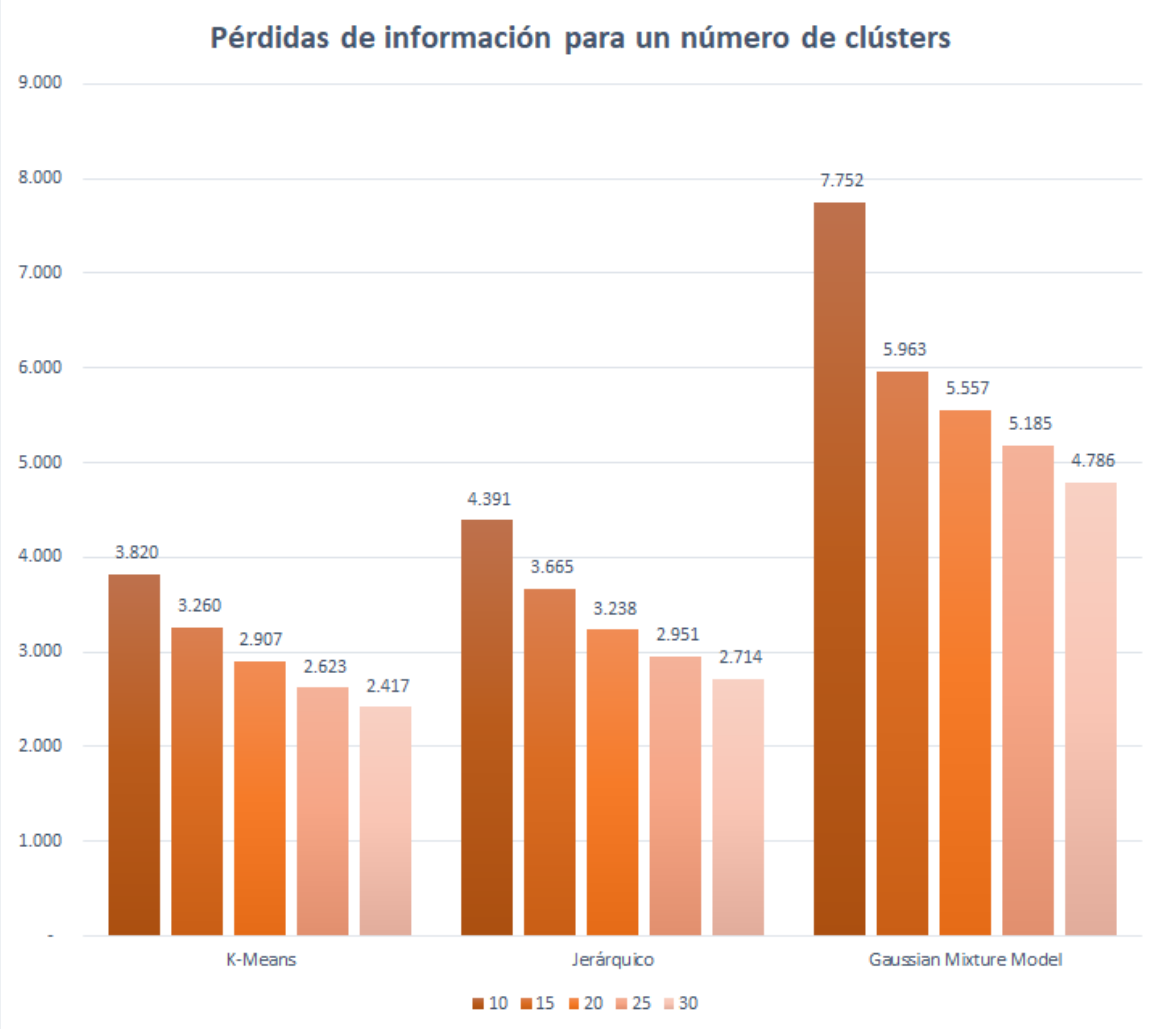


Figure 2.1: *Matriz de dispersión de las primeras 6 variables*

vemos como a medida que aumenta la cardinalidad disminuye la pérdida de información con una relación aparentemente no-lineal. Esto tiene mucho sentido ya que al aumentar el número de clústers, aumenta el número de puntos del conjunto de datos final y por tanto la cantidad de información que este almacena. La evolución de la pérdida de información a medida que aumenta el número de clústeres tenderá a cero, aunque sería interesante comprobar si se ajusta a alguna expresión matemática en particular.

Sería interesante estudiar la bondad de cada método de clustering utilizando métricas alternativas en lugar de la proporcionada por el ejercicio. Existen múltiples métricas de este tipo en la literatura, que se pueden dividir entre evaluación interna (evalúa la bondad de los contenidos internos de los clústers) y evaluación externa (se emplean datos externos al algoritmo, como por ejemplo etiquetas de clases conocidas o marcadores externos.) entre las que podemos destacar:

A. **Índice de Davies-Bouldin:**

$$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{j\neq i}\left(\frac{\sigma_i + \sigma_j}{d\left(c_i, c_j\right)}\right)$$

El algoritmo de agrupamiento que produce la colección de grupos con menores índices de Davies–Bouldin está considerado el algoritmo mejor basado en este criterio.

B. **Índice de Rand:**
El índice de Rand computa la proximidad del resultado del algoritmo de clústering con respecto a las clasificaciones de los marcadores externos.

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

Dónde TP es el número de positivos verdaderos, TN es el número de negativos verdaderos, FP es el número de falsos positivos, y FN es el número de falsos negativos.

3 Anexo

Incluímos el script empleado para calcular la pérdida de información para distintos valores de G.

```
n_grupos <- 10

kmeans.modelo <- kmeans(AM20, n_grupos)
kmeans.etiquetas <- data.frame(kmeans.modelo$cluster)
names(kmeans.etiquetas)[1] <- "etiquetas"
kmeans.centroides <- data.frame(kmeans.modelo$centers) %>%
    mutate(etiquetas = row_number())
kmeans.AM20 <- subset(left_join(kmeans.etiquetas, kmeans.centroides,
    by="etiquetas"), select = -c(etiquetas))
kmeans.IL<- sum((AM20 - kmeans.AM20)^2)

library(hclust)
D <- dist(AM20, method="euclidean")
jerarq.modelo <- hclust(D, method = "ward.D")
jerarq.etiquetas <- data.frame(cutree(jerarq.modelo, k=n_grupos))
names(jerarq.etiquetas)[1] <- "etiquetas"
jerarq.centroides <- data.frame(apply(AM20, 2, function (x) tapply (x,
    jerarq.etiquetas, mean)))%>% mutate(etiquetas = row_number())
jerarq.AM20 <- subset(left_join(jerarq.etiquetas, jerarq.centroides,
    by="etiquetas"), select = -c(etiquetas))
jerarq.IL <- sum((AM20 - jerarq.AM20)^2)


library(mclust)
gauss.modelo <- Mclust(AM20, G=n_grupos)
gauss.etiquetas <- data.frame(gauss.modelo$classification)
names(gauss.etiquetas)[1] <- "etiquetas"
gauss.centroides <- data.frame(apply(AM20, 2, function (x) tapply (x,
    gauss.etiquetas, mean)))%>% mutate(etiquetas = row_number())
gauss.AM20 <- subset(left_join(gauss.etiquetas, gauss.centroides,
    by="etiquetas"), select = -c(etiquetas))
gauss.IL <- sum((AM20 - gauss.AM20)^2)
```