Fundamentos de la Ciencia de Datos

Prueba de Laboratorio 2 (PL2)

Jorge Revenga Martín de Vidales
Ángel Salgado Aldao
Adrián García

Grado en Ingeniería Informática
Universidad de Alcalá

26 de diciembre de 2023

Índice

1. Ejercicios con ayuda del profesor

Realización de cuatro ejercicios con ayuda del profesor en los que se van a realizar, utilizando el entorno R, dos análisis de clasificación no supervisada y dos análisis de clasificación supervisada, aplicando todos los conceptos teóricos vistos en cada lección.

1.1. Análisis de clasificación no supervisada

1.1.1. k-Means

El primer conjunto de datos, que se empleará para realizar el análisis de clasificación no supervisada con k-Means, estará formado por las siguientes 8 calificaciones de estudiantes: 1.{4, 4}; 2.{3, 5}; 3.{1, 2}; 4.{5, 5}; 5.{0, 1}; 6.{2, 2}; 7.{4, 5}; 8.{2, 1}, donde las características de las calificaciones son: {Teoría, Laboratorio}.

Solución:

- `m<-matrix(c(4,4, 3,5, 1,2, 5,5, 0,1, 2,2, 4,5, 2,1),2,8)`: Introducción de los datos en forma de matriz y se transpone con `(m<-t(m))`

```
> m<-matrix(c(4,4, 3,5, 1,2, 5,5, 0,1, 2,2, 4,5, 2,1),2,8)
> (m<-t(m))

     [,1] [,2]
[1,]    4    4
[2,]    3    5
[3,]    1    2
[4,]    5    5
[5,]    0    1
[6,]    2    2
[7,]    4    5
[8,]    2    1
```

- `c<-matrix(c(0,1,2,2),2,2)`: La función de k-means viene en los paquetes por defecto, toma la matriz y los centroides iniciales, este código crea los centriodes

```
> c<-matrix(c(0,1,2,2),2,2)
> (c<-t(c))

     [,1] [,2]
[1,]    0    1
[2,]    2    2
```

- `(clasificacionns=(kmeans(m,c,4)))`: Almacena el resultado en la variable clasificacionns(no supervisada)

```
> (clasificacionns=(kmeans(m,c,4)))

K-means clustering with 2 clusters of sizes 4, 4

Cluster means:
```

```
  [,1] [,2]
1 1.25 1.50
2 4.00 4.75

Clustering vector:
[1] 2 2 1 2 1 1 2 1

Within cluster sum of squares by cluster:
[1] 3.75 2.75
 (between_SS / total_SS =  84.8 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

- : Extraemos el vector de la solución de kmeans (usando el dolar obtenemos la variable cluster) y se la añadimos a la matriz

```
> (m=cbind(clasificacionns$cluster,m))

     [,1] [,2] [,3]
[1,]    2    4    4
[2,]    2    3    5
[3,]    1    1    2
[4,]    2    5    5
[5,]    1    0    1
[6,]    1    2    2
[7,]    2    4    5
[8,]    1    2    1
```

- : Separa la matriz por clusters

```
> mc1=subset(m,m[,1]==1)
> mc2=subset(m,m[,1]==2)
> mc1

     [,1] [,2] [,3]
[1,]    1    1    2
[2,]    1    0    1
[3,]    1    2    2
[4,]    1    2    1

> mc2

     [,1] [,2] [,3]
[1,]    2    4    4
[2,]    2    3    5
[3,]    2    5    5
[4,]    2    4    5
```

- : Quitamos la primera columna de ambas ya que sólo indica el cluster

```
> (mc1=mc1[,-1])

     [,1] [,2]
[1,]    1    2
[2,]    0    1
[3,]    2    2
[4,]    2    1

> (mc2=mc2[,-1])

     [,1] [,2]
[1,]    4    4
[2,]    3    5
[3,]    5    5
[4,]    4    5
```

- : Finalmente, vemos los datos separados en tablas por el cluster al que pertenecen.

1.1.2. Clusterización Jerárquica Aglomerativa

El segundo conjunto de datos, que se empleará para realizar el análisis de clasificación no supervisada con Clusterización Jerárquica Aglomerativa, estará formado por 6 calificaciones de estudiantes: 1.{0.89, 2.94}; 2.{4.36, 5.21}; 3.{3.75, 1.12}; 4.{6.25, 3.14}; 5.{4.1, 1.8}; 6.{3.9, 4.27}.

Solución

- Cargamos el paquete `LearnClust`:

```
> library(LearnClust)
> search()

 [1] ".GlobalEnv"         "package:LearnClust" "package:magick"
 [4] "package:arules"     "package:Matrix"     "package:stats"
 [7] "package:graphics"   "package:grDevices"  "package:utils"
[10] "package:datasets"   "package:methods"    "Autoloads"
[13] "package:base"
```

- Se introduccen los datos en una matriz y se transpone:

```
> m<-matrix(c(0.89,2.94, 4.36,5.21, 3.75,1.12, 6.25,3.14, 4.1,1.8, 3.9,4.27),2,6)
> (m<-t(m))

     [,1] [,2]
[1,] 0.89 2.94
[2,] 4.36 5.21
[3,] 3.75 1.12
[4,] 6.25 3.14
[5,] 4.10 1.80
[6,] 3.90 4.27
```

- La función `agglomerativeHC` sirve para realizar un agrupamineto jerárquico algomerativo en la matriz `m`. `EUC`: se refiere a la métrica de distancia utilizada durante el agrupamiento, en este caso, es la distancia euclidiana. `MIN`: se refiere al criterio de enlace utilizado para calcular la distancia entre los grupos, en este caso, es MIN (single linkage). Explicacion

```
> agglomerativeHC(m,'EUC','MIN')

$dendrogram
Number of objects: 6


$clusters
$clusters[[1]]
    X1   X2
1 0.89 2.94

$clusters[[2]]
```

```
    X1   X2
1 4.36 5.21

$clusters[[3]]
    X1   X2
1 3.75 1.12

$clusters[[4]]
    X1   X2
1 6.25 3.14

$clusters[[5]]
   X1  X2
1 4.1 1.8

$clusters[[6]]
   X1   X2
1 3.9 4.27

$clusters[[7]]
    X1   X2
1 3.75 1.12
2 4.10 1.80

$clusters[[8]]
    X1   X2
1 4.36 5.21
2 3.90 4.27

$clusters[[9]]
    X1   X2
1 3.75 1.12
2 4.10 1.80
3 4.36 5.21
4 3.90 4.27

$clusters[[10]]
    X1   X2
1 6.25 3.14
2 3.75 1.12
3 4.10 1.80
4 4.36 5.21
5 3.90 4.27

$clusters[[11]]
    X1   X2
1 0.89 2.94
2 6.25 3.14
```

```
3 3.75 1.12
4 4.10 1.80
5 4.36 5.21
6 3.90 4.27


$groupedClusters
  cluster1 cluster2
1        3        5
2        2        6
3        7        8
4        4        9
5        1       10
```

- Para obtener más detalles sobre el proceso de agrupamiento jerárquico aglomerativo se usa `.details`:

```
> agglomerativeHC.details(m,'EUC','MIN')

[[1]]
     [,1] [,2] [,3]
[1,] 0.89 2.94    1

[[2]]
     [,1] [,2] [,3]
[1,] 4.36 5.21    1

[[3]]
     [,1] [,2] [,3]
[1,] 3.75 1.12    1

[[4]]
     [,1] [,2] [,3]
[1,] 6.25 3.14    1

[[5]]
     [,1] [,2] [,3]
[1,]  4.1  1.8    1

[[6]]
     [,1] [,2] [,3]
[1,]  3.9 4.27    1

          [,1]     [,2]      [,3]     [,4]      [,5]     [,6]
[1,] 0.000000 4.146541 3.3899853 5.363730 3.4064204 3.290745
[2,] 4.146541 0.000000 4.1352388 2.803034 3.4198977 1.046518
[3,] 3.389985 4.135239 0.0000000 3.214094 0.7647876 3.153569
[4,] 5.363730 2.803034 3.2140940 0.000000 2.5333969 2.607566
[5,] 3.406420 3.419898 0.7647876 2.533397 0.0000000 2.478084
```

```
[6,] 3.290745 1.046518 3.1535694 2.607566 2.4780839 0.000000
    X1   X2
1 3.75 1.12
2 4.10 1.80
         [,1]     [,2] [,3]     [,4] [,5]     [,6]     [,7]
[1,] 0.000000 4.146541    0 5.363730    0 3.290745 3.389985
[2,] 4.146541 0.000000    0 2.803034    0 1.046518 3.419898
[3,] 0.000000 0.000000    0 0.000000    0 0.000000 0.000000
[4,] 5.363730 2.803034    0 0.000000    0 2.607566 2.533397
[5,] 0.000000 0.000000    0 0.000000    0 0.000000 0.000000
[6,] 3.290745 1.046518    0 2.607566    0 0.000000 2.478084
[7,] 3.389985 3.419898    0 2.533397    0 2.478084 0.000000
    X1   X2
1 4.36 5.21
2 3.90 4.27
         [,1] [,2] [,3]     [,4] [,5] [,6]     [,7]     [,8]
[1,] 0.000000    0    0 5.363730    0    0 3.389985 3.290745
[2,] 0.000000    0    0 0.000000    0    0 0.000000 0.000000
[3,] 0.000000    0    0 0.000000    0    0 0.000000 0.000000
[4,] 5.363730    0    0 0.000000    0    0 2.533397 2.607566
[5,] 0.000000    0    0 0.000000    0    0 0.000000 0.000000
[6,] 0.000000    0    0 0.000000    0    0 0.000000 0.000000
[7,] 3.389985    0    0 2.533397    0    0 0.000000 2.478084
[8,] 3.290745    0    0 2.607566    0    0 2.478084 0.000000
    X1   X2
1 3.75 1.12
2 4.10 1.80
3 4.36 5.21
4 3.90 4.27
          [,1] [,2] [,3]     [,4] [,5] [,6] [,7] [,8]     [,9]
 [1,] 0.000000    0    0 5.363730    0    0    0    0 3.290745
 [2,] 0.000000    0    0 0.000000    0    0    0    0 0.000000
 [3,] 0.000000    0    0 0.000000    0    0    0    0 0.000000
 [4,] 5.363730    0    0 0.000000    0    0    0    0 2.533397
 [5,] 0.000000    0    0 0.000000    0    0    0    0 0.000000
 [6,] 0.000000    0    0 0.000000    0    0    0    0 0.000000
 [7,] 0.000000    0    0 0.000000    0    0    0    0 0.000000
 [8,] 0.000000    0    0 0.000000    0    0    0    0 0.000000
 [9,] 3.290745    0    0 2.533397    0    0    0    0 0.000000
    X1   X2
1 6.25 3.14
2 3.75 1.12
3 4.10 1.80
4 4.36 5.21
5 3.90 4.27
          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]    [,10]
 [1,] 0.000000    0    0    0    0    0    0    0    0 3.290745
 [2,] 0.000000    0    0    0    0    0    0    0    0 0.000000
```

```
 [3,] 0.000000    0    0    0    0    0    0    0    0 0.000000
 [4,] 0.000000    0    0    0    0    0    0    0    0 0.000000
 [5,] 0.000000    0    0    0    0    0    0    0    0 0.000000
 [6,] 0.000000    0    0    0    0    0    0    0    0 0.000000
 [7,] 0.000000    0    0    0    0    0    0    0    0 0.000000
 [8,] 0.000000    0    0    0    0    0    0    0    0 0.000000
 [9,] 0.000000    0    0    0    0    0    0    0    0 0.000000
[10,] 3.290745    0    0    0    0    0    0    0    0 0.000000
    X1   X2
1 0.89 2.94
2 6.25 3.14
3 3.75 1.12
4 4.10 1.80
5 4.36 5.21
6 3.90 4.27
```

- Para cambiar el tipo de criterio de enlace utilizado para calcular la distancia entre clusters se le pasa diferente argumento, en este caso, se utiliza MAX (también conocido como complete linkage):

```
> agglomerativeHC.details(m,'EUC','MAX')

[[1]]
     [,1] [,2] [,3]
[1,] 0.89 2.94    1

[[2]]
     [,1] [,2] [,3]
[1,] 4.36 5.21    1

[[3]]
     [,1] [,2] [,3]
[1,] 3.75 1.12    1

[[4]]
     [,1] [,2] [,3]
[1,] 6.25 3.14    1

[[5]]
     [,1] [,2] [,3]
[1,]  4.1  1.8    1

[[6]]
     [,1] [,2] [,3]
[1,]  3.9 4.27    1

          [,1]     [,2]      [,3]     [,4]      [,5]     [,6]
[1,] 0.000000 4.146541 3.3899853 5.363730 3.4064204 3.290745
[2,] 4.146541 0.000000 4.1352388 2.803034 3.4198977 1.046518
```

```
[3,] 3.389985 4.135239 0.0000000 3.214094 0.7647876 3.153569
[4,] 5.363730 2.803034 3.2140940 0.000000 2.5333969 2.607566
[5,] 3.406420 3.419898 0.7647876 2.533397 0.0000000 2.478084
[6,] 3.290745 1.046518 3.1535694 2.607566 2.4780839 0.000000
    X1   X2
1 3.75 1.12
2 4.10 1.80
         [,1]     [,2] [,3]     [,4] [,5]     [,6]     [,7]
[1,] 0.000000 4.146541    0 5.363730    0 3.290745 3.406420
[2,] 4.146541 0.000000    0 2.803034    0 1.046518 4.135239
[3,] 0.000000 0.000000    0 0.000000    0 0.000000 0.000000
[4,] 5.363730 2.803034    0 0.000000    0 2.607566 3.214094
[5,] 0.000000 0.000000    0 0.000000    0 0.000000 0.000000
[6,] 3.290745 1.046518    0 2.607566    0 0.000000 3.153569
[7,] 3.406420 4.135239    0 3.214094    0 3.153569 0.000000
    X1   X2
1 4.36 5.21
2 3.90 4.27
         [,1] [,2] [,3]     [,4] [,5] [,6]     [,7]     [,8]
[1,] 0.000000    0    0 5.363730    0    0 3.406420 4.146541
[2,] 0.000000    0    0 0.000000    0    0 0.000000 0.000000
[3,] 0.000000    0    0 0.000000    0    0 0.000000 0.000000
[4,] 5.363730    0    0 0.000000    0    0 3.214094 2.803034
[5,] 0.000000    0    0 0.000000    0    0 0.000000 0.000000
[6,] 0.000000    0    0 0.000000    0    0 0.000000 0.000000
[7,] 3.406420    0    0 3.214094    0    0 0.000000 4.135239
[8,] 4.146541    0    0 2.803034    0    0 4.135239 0.000000
    X1   X2
1 6.25 3.14
2 4.36 5.21
3 3.90 4.27
         [,1] [,2] [,3] [,4] [,5] [,6]     [,7] [,8]     [,9]
 [1,] 0.00000    0    0    0    0    0 3.406420    0 5.363730
 [2,] 0.00000    0    0    0    0    0 0.000000    0 0.000000
 [3,] 0.00000    0    0    0    0    0 0.000000    0 0.000000
 [4,] 0.00000    0    0    0    0    0 0.000000    0 0.000000
 [5,] 0.00000    0    0    0    0    0 0.000000    0 0.000000
 [6,] 0.00000    0    0    0    0    0 0.000000    0 0.000000
 [7,] 3.40642    0    0    0    0    0 0.000000    0 4.135239
 [8,] 0.00000    0    0    0    0    0 0.000000    0 0.000000
 [9,] 5.36373    0    0    0    0    0 4.135239    0 0.000000
    X1   X2
1 0.89 2.94
2 3.75 1.12
3 4.10 1.80
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]    [,9]   [,10]
 [1,]    0    0    0    0    0    0    0    0 0.00000 0.00000
 [2,]    0    0    0    0    0    0    0    0 0.00000 0.00000
```

```
 [3,]    0    0    0    0    0    0    0    0 0.00000 0.00000
 [4,]    0    0    0    0    0    0    0    0 0.00000 0.00000
 [5,]    0    0    0    0    0    0    0    0 0.00000 0.00000
 [6,]    0    0    0    0    0    0    0    0 0.00000 0.00000
 [7,]    0    0    0    0    0    0    0    0 0.00000 0.00000
 [8,]    0    0    0    0    0    0    0    0 0.00000 0.00000
 [9,]    0    0    0    0    0    0    0    0 0.00000 5.36373
[10,]    0    0    0    0    0    0    0    0 5.36373 0.00000
    X1   X2
1 6.25 3.14
2 4.36 5.21
3 3.90 4.27
4 0.89 2.94
5 3.75 1.12
6 4.10 1.80
```

1.2. Análisis de clasificación supervisada

1.2.1. Árboles de decisión

El tercer conjunto de datos, que se empleará para realizar el análisis de clasificación supervisada utilizando árboles de decisión, estará formado por las siguientes 9 calificaciones de estudiantes: 1.{A,A,B,Ap}; 2.{A,B,D,Ss}; 3.{D,D,C,Ss}; 4.{D,D,A,Ss}; 5.{B,C,B,Ss}; 6.{C,B,B,Ap}; 7.{B,B,A,Ap}; 8.{C,D,C,Ss}; 9.{B,A,C,Ss}, donde las características de las calificaciones son: {Teoría, Laboratorio, Prácticas, Calificación Global).

Solución:

- Primero creamos un txt con los datos del problema en cuestión, para después mediante el uso del paquete "rpart´´ realizar la clasificación.

```
> library("rpart")
> (muestra=read.table("calificaciones.txt"))
```

```
  T L P C.G
1 A A B  Ap
2 A B D  Ss
3 D D C  Ss
4 D D A  Ss
5 B C B  Ss
6 C B B  Ap
7 B B A  Ap
8 C D C  Ss
9 B A C  Ss
```

- Acto seguido hacemos uso de la función que viene incluida en el paquete llamada `rpart()`, la cual recibe la variable dependiente `C.G` y las variables predictoras `T,L,P`, el conjunto de datos sobre el que realiza la clasificación `data=muestra`, con `method=class` indica que se está ajustando un modelo de clasificación y el número mínimo de observaciones `minsplit=1`.

```
> clasificacion=rpart(C.G~T+L+P, data=muestra, method="class", minsplit=1)
```

- Esta función hace la clasificación pertinente y devuelve en forma de árbol la solución con los diferentes nodos, además indica que nodos representan un nodo final usando el símbolo "*´´.

1.2.2. Regresión

El cuarto conjunto de datos, que se empleará para realizar el análisis de clasificación supervisada utilizando regresión, estará formado por los siguientes 4 radios ecuatoriales y densidades de los planetas interiores: {Mercurio,2.4,5.4; Venus,6.1,5.2; Tierra,6.4,5.5; Marte,3.4,3.9}

Solución:

- El primer paso es crear un txt con los datos necesarios, para después hacer uso de él en el análisis de regresión

```
> (muestra=read.table("planetas.txt"))
```

```
           R   D
Mercurio 2.4 5.4
Venus    6.1 5.2
Tierra   6.4 5.5
Marte    3.4 3.9
```

- Después usando la función `lm()`, se calcula la regresión indicando la relación entre la variable dependiente `D` y la variable independienteR y el conjunto de datos en el que se encuentran dichas variables `data=muestra`.

```
> regresion=lm(D~R, data=muestra)
```

- Además de esto, gracias a la función `summary()`, podemos obtener el valor de R-squared y observar la bonanza de la regresión.

```
> summary(regresion)
```

```
Call:
lm(formula = D ~ R, data = muestra)

Residuals:
Mercurio    Venus   Tierra    Marte
 0.70312 -0.01253  0.24566 -0.93624

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.3624     1.2050   3.620   0.0685 .
R             0.1394     0.2466   0.565   0.6289
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.846 on 2 degrees of freedom
Multiple R-squared:  0.1377,        Adjusted R-squared:  -0.2935
F-statistic: 0.3193 on 1 and 2 DF,  p-value: 0.6289
```

- Otro uso que podemos hacer de la función `summary()`, es analizar los outliers de la siguiente forma. En la cual obtenemos los residuos, para después hacer la media de los mismos y con un bucle for comprobar cual de todos ellos es un outlier en caso de que los hubiese.

```
> (res=summary(regresion)$residuals)
```

```
   Mercurio       Venus      Tierra       Marte
 0.70312301 -0.01253452  0.24565541 -0.93624389
```

```
> sr=sqrt(sum(res^2)/4)
> for (i in 1:length(res)){if (res[i]>3*sr){print("el suceso"); print(res[i]); pr
```

2. Ejercicios de forma autónoma

Realización de cuatro ejercicios de forma autónoma por cada grupo de estudiantes en los que se van a realizar, utilizando el entorno R, dos análisis de clasificación no supervisada y dos análisis de clasificación supervisada, aplicando todos los conceptos teóricos vistos en cada lección.

2.1. Análisis de clasificación no supervisada

2.1.1. K-means

El primer conjunto de datos, que se empleará para realizar el análisis de clasificación no supervisada con K-means, estará formado por los siguientes 15 valores de velocidades de respuesta y temperaturas normalizadas de un microprocesador {Velocidad, Temperatura}: 1.{3.5, 4.5}; 2.{0.75, 3.25}; 3.{0, 3}; 4.{1.75, 0.75}; 5.{3, 3.75}; 6.{3.75, 4.5}; 7.{1.25, 0.75}; 8.{0.25, 3}; 9.{3.5, 4.25}; 10.{1.5, 0.5}; 11.{1, 1}; 12.{3, 4}; 13.{0.5, 3}; 14.{2, 0.25}; 15.{0, 2.5}. Del análisis visual de los datos se ha concluido que hay una alta probabilidad que sean tres clusters.

Solución:

- `datos <- read.csv("Datos2.1.csv")`: Se leen los datos del archivo .csv y se guardan en la variable "datos".
- `clasificacionns <- kmedias(datos, num_clusters = 3, mostrar_proceso = TRUE)`: Llamar a nuestra función que clasifica con K-means, seleccionando el número de clusters a generar (La función creada genera 2 en caso de no especificarse) y que nos muestre el proceso de clasificación de los clusters.

```
> #cargamos la función
> source("kmedias.R")
> datos <- read.csv("Datos2.1.csv")
> clasificacionns <- kmedias(datos, num_clusters = 3, mostrar_proceso = TRUE)

Iteración:  1
Matriz de Distancias:
           [,1]      [,2]      [,3]
 [1,] 0.0000000 3.0207615 3.8078866
 [2,] 3.0207615 0.0000000 0.7905694
 [3,] 3.8078866 0.7905694 0.0000000
 [4,] 4.1382363 2.6925824 2.8504386
 [5,] 0.9013878 2.3048861 3.0923292
 [6,] 0.2500000 3.2500000 4.0388736
 [7,] 4.3732139 2.5495098 2.5739075
 [8,] 3.5794553 0.5590170 0.2500000
 [9,] 0.2500000 2.9261750 3.7165172
[10,] 4.4721360 2.8504386 2.9154759
[11,] 4.3011626 2.2638463 2.2360680
[12,] 0.7071068 2.3717082 3.1622777
[13,] 3.3541020 0.3535534 0.5000000
```

```
[14,] 4.5069391 3.2500000 3.4003676
[15,] 4.0311289 1.0606602 0.5000000
Clusters Asignados:
 [1] 1 2 3 2 1 1 2 3 1 2 3 1 2 2 3
Nuevos centroides:
         [,1]     [,2]
[1,] 3.350000 4.200000
[2,] 1.291667 1.416667
[3,] 0.312500 2.375000
Iteración:  2
Matriz de Distancias:
           [,1]      [,2]      [,3]
 [1,] 0.3354102 3.7925823 3.8308982
 [2,] 2.7681221 1.9116783 0.9782797
 [3,] 3.5584407 2.0433666 0.6987712
 [4,] 3.8029594 0.8090203 2.1695694
 [5,] 0.5700877 2.8918588 3.0188212
 [6,] 0.5000000 3.9433929 4.0412908
 [7,] 4.0388736 0.6679675 1.8760414
 [8,] 3.3241540 1.8952609 0.6281172
 [9,] 0.1581139 3.5922853 3.6980780
[10,] 4.1367258 0.9400428 2.2194101
[11,] 3.9702015 0.5086065 1.5372967
[12,] 0.4031129 3.0970977 3.1405861
[13,] 3.0923292 1.7702205 0.6525192
[14,] 4.1743263 1.3648616 2.7135367
[15,] 3.7566608 1.6858274 0.3365728
Clusters Asignados:
 [1] 1 3 3 2 1 1 2 3 1 2 2 1 3 2 3
Nuevos centroides:
     [,1] [,2]
[1,] 3.35 4.20
[2,] 1.50 0.65
[3,] 0.30 2.95
Iteración:  3
Matriz de Distancias:
           [,1]      [,2]       [,3]
 [1,] 0.3354102 4.3384905 3.55562934
 [2,] 2.7681221 2.7060118 0.54083269
 [3,] 3.5584407 2.7879204 0.30413813
 [4,] 3.8029594 0.2692582 2.63486243
 [5,] 0.5700877 3.4438351 2.81602557
 [6,] 0.5000000 4.4592600 3.78219513
 [7,] 4.0388736 0.2692582 2.39635139
 [8,] 3.3241540 2.6617663 0.07071068
 [9,] 0.1581139 4.1182521 3.45398321
[10,] 4.1367258 0.1500000 2.72809457
[11,] 3.9702015 0.6103278 2.07183494
```

```
[12,] 0.4031129 3.6704904 2.89698119
[13,] 3.0923292 2.5539186 0.20615528
[14,] 4.1743263 0.6403124 3.19061123
[15,] 3.7566608 2.3817011 0.54083269
Clusters Asignados:
 [1] 1 3 3 2 1 1 2 3 1 2 2 1 3 2 3
Nuevos centroides:
     [,1] [,2]
[1,] 3.35 4.20
[2,] 1.50 0.65
[3,] 0.30 2.95
Convergencia alcanzada en la iteración: 3

> # Añadimos la columna de clusters a los datos originales
> m <- cbind(clasificacionns$cluster, datos)
> # Separamos la matriz por clusters
> mc1 <- subset(m, m[, 1] == 1)
> mc2 <- subset(m, m[, 1] == 2)
> mc3 <- subset(m, m[, 1] == 3)
> # Limpiamos los datos (eliminamos la columna de cluster)
> mc1 <- mc1[, -1]
> mc2 <- mc2[, -1]
> mc3 <- mc3[, -1]
> # Datos originales
> print(datos)

   Velocidad Temperatura
1       3.50        4.50
2       0.75        3.25
3       0.00        3.00
4       1.75        0.75
5       3.00        3.75
6       3.75        4.50
7       1.25        0.75
8       0.25        3.00
9       3.50        4.25
10      1.50        0.50
11      1.00        1.00
12      3.00        4.00
13      0.50        3.00
14      2.00        0.25
15      0.00        2.50

> # Resultado del K-Means
> print(m)

   clasificacionns$cluster Velocidad Temperatura
1                        1      3.50        4.50
2                        3      0.75        3.25
```

```
3                       3       0.00        3.00
4                       2       1.75        0.75
5                       1       3.00        3.75
6                       1       3.75        4.50
7                       2       1.25        0.75
8                       3       0.25        3.00
9                       1       3.50        4.25
10                      2       1.50        0.50
11                      2       1.00        1.00
12                      1       3.00        4.00
13                      3       0.50        3.00
14                      2       2.00        0.25
15                      3       0.00        2.50

> # Datos en el cluster 1
> print(mc1)

   Velocidad Temperatura
1       3.50        4.50
5       3.00        3.75
6       3.75        4.50
9       3.50        4.25
12      3.00        4.00

> #Datos en el cluster 2
> print(mc2)

   Velocidad Temperatura
4       1.75        0.75
7       1.25        0.75
10      1.50        0.50
11      1.00        1.00
14      2.00        0.25

> # Datos en el cluster 3
> print(mc3)

   Velocidad Temperatura
2       0.75        3.25
3       0.00        3.00
8       0.25        3.00
13      0.50        3.00
15      0.00        2.50

>
>
```

2.1.2. Clusterización Jerárquica Aglomerativa

El segundo conjunto de datos, que se empleará para realizar el análisis de clasificación no supervisada con Clusterización Jerárquica Aglomerativa, será el mismo que el utilizado en el ejercicio anterior, por lo tanto estará formado por los siguientes 15 valores de velocidades de respuesta y temperaturas normalizadas de un microprocesador {Velocidad, Temperatura}: 1.{3.5, 4.5}; 2.{0.75, 3.25}; 3.{0, 3}; 4.{1.75, 0.75}; 5.{3, 3.75}; 6.{3.75, 4.5}; 7.{1.25, 0.75}; 8.{0.25, 3}; 9.{3.5, 4.25}; 10.{1.5, 0.5}; 11.{1, 1}; 12.{3, 4}; 13.{0.5, 3}; 14.{2, 0.25}; 15.{0, 2.5}.

Se proporcionan dos soluciones, un paquete diferente al usado por el profesor y un algoritmo programado por nosotros.

Solución (paquete):

- `datos <- read.csv("Datos2.2.csv")`: Se leen los datos del archivo .csv y se guardan en la variable "datos".
- `(datos<-t(datos))`: Se hace la traspuesta de la matriz datos para un mejor tratamiento.
- `(clMin<-agglomerative_clustering(datos,''single´´,FALSE,FALSE))`: Clusterización usando como definición de proximidad, la distancia entre los puntos más cercanos de los clusters.
  - Parámetros:
    - `datos`: Variable que contiene los puntos a evaluar.
    - `"single"`: Algoritmo de clusterización, en este caso usando minimum linkage.
    - `details="FALSE"`: Booleano para determinar si se imprimen o no los logs con la explicación de todo el proceso.
    - `waiting="FALSE"`: Booleano para determinar si tiene que esperar input del usuario para ir imprimiendo los logs.
  - Salida: Matriz con los clusters y sus distancias.
  - Explicación: Realiza un análisis jerárquico aglomerativo de los datos introducidos por parámetro. Repite una serie de pasos hasta que solo quede un cluster. Inicialmente cada punto se asigna a su propio cluster y se calcula la matriz de distancias entre ellos. Después se calcula la proximidad entre dos clusters con la distancia entre los 2 puntos más cercanos de dichos clusters (falta formula)

```
> library("clustlearn")
> datos <- read.csv("Datos2.2.csv")
> (datos<-t(datos))
```

```
       [,1]
X3.5   4.50
X0.75  3.25
X0     3.00
X1.75  0.75
X3     3.75
X3.75  4.50
```

```
X1.25  0.75
X0.25  3.00
X3.5.1 4.25
X1.5   0.50
X1     1.00
X3.1   4.00
X0.5   3.00
X2     0.25
X0.1   2.50
```

```
> (clMin<-agglomerative_clustering(datos,"single",FALSE,FALSE))
```

```
Cluster method   : single
Distance         : Euclidean
Number of objects: 15
```

- `(clMax<-agglomerative_clustering(datos,''complete´´,FALSE,FALSE))`: Clusterización usando como definición de proximidad, la distancia que haya entre los dos puntos más lejanos de los clusters.

  - Parámetros:
    - `datos`: Variable que contiene los puntos a evaluar.
    - `"single"`: Algoritmo de clusterización, en este caso usando maximum linkage.
    - `details="FALSE"`: Booleano para determinar si se imprimen o no los logs con la explicación de todo el proceso.
    - `waiting="FALSE"`: Booleano para determinar si tiene que esperar input del usuario para ir imprimiendo los logs.
  - Salida: Matriz con los clusters y sus distancias.
  - Explicación: Realiza un análisis jerárquico aglomerativo de los datos introducidos por parámetro. Repite una serie de pasos hasta que solo quede un cluster. Inicialmente cada punto se asigna a su propio cluster y se calcula la matriz de distancias entre ellos. Después se calcula la proximidad entre dos clusters con la distancia entre los puntos más lejanos de dichos clusters (falta formula)

```
> install.packages("clustlearn")
> library("clustlearn")
> datos <- read.csv("Datos2.2.csv")
> (datos<-t(datos))
```

```
       [,1]
X3.5   4.50
X0.75  3.25
X0     3.00
X1.75  0.75
X3     3.75
X3.75  4.50
X1.25  0.75
X0.25  3.00
X3.5.1 4.25
```

```
X1.5   0.50
X1     1.00
X3.1   4.00
X0.5   3.00
X2     0.25
X0.1   2.50
```

```
> (clMin<-agglomerative_clustering(datos,"complete",FALSE,FALSE))
```

```
Cluster method   : complete
Distance         : Euclidean
Number of objects: 15
```

- `(clAvg<-agglomerative_clustering(datos,"average´´,FALSE,FALSE))`: Clusterización usando como definición de proximidad, la media de distancias entre todas las parejas de puntos de los dos clusters.

  - Parámetros:
    - `datos`: Variable que contiene los puntos a evaluar.
    - `."verage"`: Algoritmo de clusterización, en este caso usando average linkage.
    - `details="FALSE"`: Booleano para determinar si se imprimen o no los logs con la explicación de todo el proceso.
    - `waiting="FALSE"`: Booleano para determinar si tiene que esperar input del usuario para ir imprimiendo los logs.
  - Salida: Matriz con los clusters y sus distancias.
  - Explicación: Realiza un análisis jerárquico aglomerativo de los datos introducidos por parámetro. Repite una serie de pasos hasta que solo quede un cluster. Inicialmente cada punto se asigna a su propio cluster y se calcula la matriz de distancias entre ellos. Después se calcula la proximidad entre dos clusters con la media de todos los puntos de ambos (falta formula)

```
> library("clustlearn")
> datos <- read.csv("Datos2.2.csv")
> (datos<-t(datos))
```

```
       [,1]
X3.5   4.50
X0.75  3.25
X0     3.00
X1.75  0.75
X3     3.75
X3.75  4.50
X1.25  0.75
X0.25  3.00
X3.5.1 4.25
X1.5   0.50
X1     1.00
X3.1   4.00
X0.5   3.00
```

```
X2     0.25
X0.1   2.50
```

```
> (clMin<-agglomerative_clustering(datos,"average",FALSE,FALSE))
```

```
Cluster method   : average
Distance         : Euclidean
Number of objects: 15
```

Solución (programado):

- Explicación: Este código implementa un algoritmo jerárquico aglomerativo para la agrupación de datos. El algoritmo utiliza matrices de distancia entre puntos y fusiona iterativamente los clusters más cercanos hasta que queda un único cluster que contiene todos los puntos. Los datos de entrada están representados en la matriz 'datos'. El código proporciona tres funciones para calcular la distancia entre clusters: MIN (single link), MAX (complete link), y Group Average. El algoritmo utiliza una de estas funciones según el método de distancia especificado. Durante cada iteración, se calcula la matriz de distancias entre clusters, se encuentra el par de clusters más cercano, se fusionan en un nuevo cluster, y se repite el proceso hasta que solo queda un cluster. El código imprime la matriz de distancias en cada iteración y muestra los clusters que se fusionan. Al final, se presenta un resumen de las iteraciones, mostrando qué clusters se fusionaron en cada paso y la distancia correspondiente. En el propio codigo hay comentarios con una breve explicación de cada parte.

```
> # Datos proporcionados
> datos <-
+   matrix(c(3.5, 4.5, 0.75, 3.25, 0, 3, 1.75, 0.75, 3, 3.75,
+                   3.75, 4.5, 1.25, 0.75, 0.25, 3, 3.5, 4.25, 1.5, 0.5,
+                   1, 1, 3, 4, 0.5, 3, 2, 0.25, 0, 2.5), ncol = 2, byrow = TRUE)
> # Función para calcular la distancia con MIN (single link)
> calcular_distancia_minima <-
+   function(cluster1, cluster2, matriz_distancias) {
+   min_dist <-
+     min(matriz_distancias[cluster1$elementos, cluster2$elementos])
+   return(min_dist)
+ }
> # Función para calcular la distancia con MAX (complete link)
> calcular_distancia_maxima <-
+   function(cluster1, cluster2, matriz_distancias) {
+   max_dist <-
+     max(matriz_distancias[cluster1$elementos, cluster2$elementos])
+   return(max_dist)
+ }
> # Función para calcular la distancia con Group Average
> calcular_distancia_grupo_promedio <-
+   function(cluster1, cluster2, matriz_distancias) {
+   distancias <-
+     matriz_distancias[cluster1$elementos, cluster2$elementos]
+   if (length(distancias) == 0) {
```

```
+     return(NA)
+     # Evitar divisiones por cero si no hay elementos en los clusters
+   }
+   return(mean(distancias, na.rm = TRUE))
+ }
> # Algoritmo jerárquico aglomerativo
> algoritmo_aglomerativo <- function(datos, metodo_distancia) {
+   # Número de puntos
+   n_puntos <- nrow(datos)
+
+   # Crear una copia de la matriz original de distancias
+   matriz_distancias_original <- as.matrix(dist(datos))
+
+   # Crear una lista de clusters
+   # cada punto es un cluster en la primera iteración
+   clusters <-
+     lapply(1:n_puntos, function(i)
+       list(etiqueta = as.character(i), elementos = i))
+
+   # Seleccionar la función de distancia
+   if (metodo_distancia == "MIN") {
+     calcular_distancia <- calcular_distancia_minima
+   } else if (metodo_distancia == "MAX") {
+     calcular_distancia <- calcular_distancia_maxima
+   } else if (metodo_distancia == "GROUP_AVERAGE") {
+     calcular_distancia <- calcular_distancia_grupo_promedio
+   } else {
+     stop("Método de distancia no válido.")
+   }
+
+   iteraciones <- list()
+   etiqueta <- 1
+   while (length(clusters) > 1) {
+     # Inicializar la matriz de distancias entre clusters con Inf
+     distancias_clusters <-
+       matrix("", nrow = length(clusters), ncol = length(clusters),
+              dimnames = list(sapply(clusters, function(x) x$etiqueta),
+                              sapply(clusters, function(x) x$etiqueta)))
+
+     # Calcular la distancia entre cada par de clusters
+     for (i in 1:(length(clusters) - 1)) {
+       for (j in (i + 1):length(clusters)) {
+         # Evitar imprimir Inf en la matriz
+         dist <- calcular_distancia(clusters[[i]], clusters[[j]],
+                                    matriz_distancias_original)
+         distancias_clusters[j, i] <-
+           if (is.finite(dist)) round(dist, 2) else ""
+       }
```

```
+     }
+
+     # Si la matriz está completamente vacía, no imprimir
+     if (any(distancias_clusters != "")) {
+       # Guardar la matriz de distancias actualizada
+       cat("Matriz de distancias", etiqueta, ":\n")
+       print(distancias_clusters, quote = FALSE)
+     }
+
+     # Encontrar el par de clusters más cercano
+     min_dist <- min(as.numeric(distancias_clusters), na.rm = TRUE)
+     min_index <- which(distancias_clusters == min_dist, arr.ind = TRUE)
+
+     # Unir los dos clusters más cercanos en uno nuevo
+     new_cluster <- list(etiqueta = paste("C", etiqueta, sep = ""),
+                         elementos = c(clusters[[min_index[1, 1]]]$elementos,
+                                       clusters[[min_index[1, 2]]]$elementos))
+
+     # Guardar la iteración actual
+     iteraciones[[etiqueta]] <- list(cluster1 = clusters[[min_index[1, 1]]],
+                                     cluster2 = clusters[[min_index[1, 2]]],
+                                     nuevo_cluster = new_cluster,
+                                     distancia = min_dist)
+
+     # Mostrar los clusters que se unen y forman
+     cat("Se unen los clusters", iteraciones[[etiqueta]]$cluster1$etiqueta,
+         "y", iteraciones[[etiqueta]]$cluster2$etiqueta, "con distancia de",
+         round(iteraciones[[etiqueta]]$distancia, 2), "para formar",
+         iteraciones[[etiqueta]]$nuevo_cluster$etiqueta, "\n\n")
+
+     # Incrementar la etiqueta para la próxima iteración
+     etiqueta <- etiqueta + 1
+
+     # Eliminar los clusters antiguos
+     clusters <- clusters[-c(min_index[1, 1], min_index[1, 2])]
+
+     # Agregar el nuevo cluster
+     clusters <- append(clusters, list(new_cluster))
+   }
+
+   # Imprimir el resultado final
+   cat("Resumen:\n")
+   for (i in seq_len(length(iteraciones))) {
+     cat("Iteración", i, ": Se unen",
+         iteraciones[[i]]$cluster1$etiqueta,
+         "y", iteraciones[[i]]$cluster2$etiqueta, "con distancia",
+         round(iteraciones[[i]]$distancia, 2), "para formar",
+         iteraciones[[i]]$nuevo_cluster$etiqueta, "\n")
```

```
+   }
+ }
```

- Ejemplo de uso con el método de distancia "MIN"

```
> algoritmo_aglomerativo(datos, "MIN")

Matriz de distancias 1 :
   1    2    3    4    5    6    7    8    9    10   11   12   13   14   15
1
2  3.02
3  3.81 0.79
4  4.14 2.69 2.85
5  0.9  2.3  3.09 3.25
6  0.25 3.25 4.04 4.25 1.06
7  4.37 2.55 2.57 0.5  3.47 4.51
8  3.58 0.56 0.25 2.7  2.85 3.81 2.46
9  0.25 2.93 3.72 3.91 0.71 0.35 4.16 3.48
10 4.47 2.85 2.92 0.35 3.58 4.59 0.35 2.8  4.25
11 4.3  2.26 2.24 0.79 3.4  4.45 0.35 2.14 4.1  0.71
12 0.71 2.37 3.16 3.48 0.25 0.9  3.69 2.93 0.56 3.81 3.61
13 3.35 0.35 0.5  2.57 2.61 3.58 2.37 0.25 3.25 2.69 2.06 2.69
14 4.51 3.25 3.4  0.56 3.64 4.6  0.9  3.26 4.27 0.56 1.25 3.88 3.13
15 4.03 1.06 0.5  2.47 3.25 4.25 2.15 0.56 3.91 2.5  1.8  3.35 0.71 3.01
Se unen los clusters 6 y 1 con distancia de 0.25 para formar C1

Matriz de distancias 2 :
   2    3    4    5    7    8    9    10   11   12   13   14   15   C1
2
3  0.79
4  2.69 2.85
5  2.3  3.09 3.25
7  2.55 2.57 0.5  3.47
8  0.56 0.25 2.7  2.85 2.46
9  2.93 3.72 3.91 0.71 4.16 3.48
10 2.85 2.92 0.35 3.58 0.35 2.8  4.25
11 2.26 2.24 0.79 3.4  0.35 2.14 4.1  0.71
12 2.37 3.16 3.48 0.25 3.69 2.93 0.56 3.81 3.61
13 0.35 0.5  2.57 2.61 2.37 0.25 3.25 2.69 2.06 2.69
14 3.25 3.4  0.56 3.64 0.9  3.26 4.27 0.56 1.25 3.88 3.13
15 1.06 0.5  2.47 3.25 2.15 0.56 3.91 2.5  1.8  3.35 0.71 3.01
C1 3.02 3.81 4.14 0.9  4.37 3.58 0.25 4.47 4.3  0.71 3.35 4.51 4.03
Se unen los clusters 8 y 3 con distancia de 0.25 para formar C2

Matriz de distancias 3 :
   2    4    5    7    9    10   11   12   13   14   15   C1   C2
2
4  2.69
5  2.3  3.25
```

```
7  2.55 0.5  3.47
9  2.93 3.91 0.71 4.16
10 2.85 0.35 3.58 0.35 4.25
11 2.26 0.79 3.4  0.35 4.1  0.71
12 2.37 3.48 0.25 3.69 0.56 3.81 3.61
13 0.35 2.57 2.61 2.37 3.25 2.69 2.06 2.69
14 3.25 0.56 3.64 0.9  4.27 0.56 1.25 3.88 3.13
15 1.06 2.47 3.25 2.15 3.91 2.5  1.8  3.35 0.71 3.01
C1 3.02 4.14 0.9  4.37 0.25 4.47 4.3  0.71 3.35 4.51 4.03
C2 0.56 2.7  2.85 2.46 3.48 2.8  2.14 2.93 0.25 3.26 0.5  3.58
Se unen los clusters 12 y 5 con distancia de 0.25 para formar C3

Matriz de distancias 4 :
   2    4    7    9    10   11   13   14   15   C1   C2   C3
2
4  2.69
7  2.55 0.5
9  2.93 3.91 4.16
10 2.85 0.35 0.35 4.25
11 2.26 0.79 0.35 4.1  0.71
13 0.35 2.57 2.37 3.25 2.69 2.06
14 3.25 0.56 0.9  4.27 0.56 1.25 3.13
15 1.06 2.47 2.15 3.91 2.5  1.8  0.71 3.01
C1 3.02 4.14 4.37 0.25 4.47 4.3  3.35 4.51 4.03
C2 0.56 2.7  2.46 3.48 2.8  2.14 0.25 3.26 0.5  3.58
C3 2.3  3.25 3.47 0.56 3.58 3.4  2.61 3.64 3.25 0.71 2.85
Se unen los clusters C1 y 9 con distancia de 0.25 para formar C4

Matriz de distancias 5 :
   2    4    7    10   11   13   14   15   C2   C3   C4
2
4  2.69
7  2.55 0.5
10 2.85 0.35 0.35
11 2.26 0.79 0.35 0.71
13 0.35 2.57 2.37 2.69 2.06
14 3.25 0.56 0.9  0.56 1.25 3.13
15 1.06 2.47 2.15 2.5  1.8  0.71 3.01
C2 0.56 2.7  2.46 2.8  2.14 0.25 3.26 0.5
C3 2.3  3.25 3.47 3.58 3.4  2.61 3.64 3.25 2.85
C4 2.93 3.91 4.16 4.25 4.1  3.25 4.27 3.91 3.48 0.56
Se unen los clusters C2 y 13 con distancia de 0.25 para formar C5

Matriz de distancias 6 :
   2    4    7    10   11   14   15   C3   C4   C5
2
4  2.69
7  2.55 0.5
```

```
10 2.85 0.35 0.35
11 2.26 0.79 0.35 0.71
14 3.25 0.56 0.9  0.56 1.25
15 1.06 2.47 2.15 2.5  1.8  3.01
C3 2.3  3.25 3.47 3.58 3.4  3.64 3.25
C4 2.93 3.91 4.16 4.25 4.1  4.27 3.91 0.56
C5 0.35 2.57 2.37 2.69 2.06 3.13 0.5  2.61 3.25
Se unen los clusters C5 y 2 con distancia de 0.35 para formar C6

Matriz de distancias 7 :
   4    7    10   11   14   15   C3   C4   C6
4
7  0.5
10 0.35 0.35
11 0.79 0.35 0.71
14 0.56 0.9  0.56 1.25
15 2.47 2.15 2.5  1.8  3.01
C3 3.25 3.47 3.58 3.4  3.64 3.25
C4 3.91 4.16 4.25 4.1  4.27 3.91 0.56
C6 2.57 2.37 2.69 2.06 3.13 0.5  2.3  2.93
Se unen los clusters 10 y 4 con distancia de 0.35 para formar C7

Matriz de distancias 8 :
   7    11   14   15   C3   C4   C6   C7
7
11 0.35
14 0.9  1.25
15 2.15 1.8  3.01
C3 3.47 3.4  3.64 3.25
C4 4.16 4.1  4.27 3.91 0.56
C6 2.37 2.06 3.13 0.5  2.3  2.93
C7 0.35 0.71 0.56 2.47 3.25 3.91 2.57
Se unen los clusters 11 y 7 con distancia de 0.35 para formar C8

Matriz de distancias 9 :
   14   15   C3   C4   C6   C7   C8
14
15 3.01
C3 3.64 3.25
C4 4.27 3.91 0.56
C6 3.13 0.5  2.3  2.93
C7 0.56 2.47 3.25 3.91 2.57
C8 0.9  1.8  3.4  4.1  2.06 0.35
Se unen los clusters C8 y C7 con distancia de 0.35 para formar C9

Matriz de distancias 10 :
   14   15   C3   C4   C6   C9
14
```

```
15 3.01
C3 3.64 3.25
C4 4.27 3.91 0.56
C6 3.13 0.5  2.3  2.93
C9 0.56 1.8  3.25 3.91 2.06
Se unen los clusters C6 y 15 con distancia de 0.5 para formar C10

Matriz de distancias 11 :
    14   C3   C4   C9  C10
14
C3  3.64
C4  4.27 0.56
C9  0.56 3.25 3.91
C10 3.01 2.3  2.93 1.8
Se unen los clusters C9 y 14 con distancia de 0.56 para formar C11

Matriz de distancias 12 :
    C3   C4   C10 C11
C3
C4  0.56
C10 2.3  2.93
C11 3.25 3.91 1.8
Se unen los clusters C4 y C3 con distancia de 0.56 para formar C12

Matriz de distancias 13 :
    C10 C11  C12
C10
C11 1.8
C12 2.3 3.25
Se unen los clusters C11 y C10 con distancia de 1.8 para formar C13

Matriz de distancias 14 :
    C12 C13
C12
C13 2.3
Se unen los clusters C13 y C12 con distancia de 2.3 para formar C14

Resumen:
Iteración 1 : Se unen 6 y 1 con distancia 0.25 para formar C1
Iteración 2 : Se unen 8 y 3 con distancia 0.25 para formar C2
Iteración 3 : Se unen 12 y 5 con distancia 0.25 para formar C3
Iteración 4 : Se unen C1 y 9 con distancia 0.25 para formar C4
Iteración 5 : Se unen C2 y 13 con distancia 0.25 para formar C5
Iteración 6 : Se unen C5 y 2 con distancia 0.35 para formar C6
Iteración 7 : Se unen 10 y 4 con distancia 0.35 para formar C7
Iteración 8 : Se unen 11 y 7 con distancia 0.35 para formar C8
Iteración 9 : Se unen C8 y C7 con distancia 0.35 para formar C9
Iteración 10 : Se unen C6 y 15 con distancia 0.5 para formar C10
```

```
Iteración 11 : Se unen C9 y 14 con distancia 0.56 para formar C11
Iteración 12 : Se unen C4 y C3 con distancia 0.56 para formar C12
Iteración 13 : Se unen C11 y C10 con distancia 1.8 para formar C13
Iteración 14 : Se unen C13 y C12 con distancia 2.3 para formar C14
```

- Ejemplo de uso con el método de distancia "MAX"

```
> algoritmo_aglomerativo(datos, "MAX")

Matriz de distancias 1 :
   1    2    3    4    5    6    7    8    9    10   11   12   13   14   15
1
2  3.02
3  3.81 0.79
4  4.14 2.69 2.85
5  0.9  2.3  3.09 3.25
6  0.25 3.25 4.04 4.25 1.06
7  4.37 2.55 2.57 0.5  3.47 4.51
8  3.58 0.56 0.25 2.7  2.85 3.81 2.46
9  0.25 2.93 3.72 3.91 0.71 0.35 4.16 3.48
10 4.47 2.85 2.92 0.35 3.58 4.59 0.35 2.8  4.25
11 4.3  2.26 2.24 0.79 3.4  4.45 0.35 2.14 4.1  0.71
12 0.71 2.37 3.16 3.48 0.25 0.9  3.69 2.93 0.56 3.81 3.61
13 3.35 0.35 0.5  2.57 2.61 3.58 2.37 0.25 3.25 2.69 2.06 2.69
14 4.51 3.25 3.4  0.56 3.64 4.6  0.9  3.26 4.27 0.56 1.25 3.88 3.13
15 4.03 1.06 0.5  2.47 3.25 4.25 2.15 0.56 3.91 2.5  1.8  3.35 0.71 3.01
Se unen los clusters 6 y 1 con distancia de 0.25 para formar C1

Matriz de distancias 2 :
   2    3    4    5    7    8    9    10   11   12   13   14   15   C1
2
3  0.79
4  2.69 2.85
5  2.3  3.09 3.25
7  2.55 2.57 0.5  3.47
8  0.56 0.25 2.7  2.85 2.46
9  2.93 3.72 3.91 0.71 4.16 3.48
10 2.85 2.92 0.35 3.58 0.35 2.8  4.25
11 2.26 2.24 0.79 3.4  0.35 2.14 4.1  0.71
12 2.37 3.16 3.48 0.25 3.69 2.93 0.56 3.81 3.61
13 0.35 0.5  2.57 2.61 2.37 0.25 3.25 2.69 2.06 2.69
14 3.25 3.4  0.56 3.64 0.9  3.26 4.27 0.56 1.25 3.88 3.13
15 1.06 0.5  2.47 3.25 2.15 0.56 3.91 2.5  1.8  3.35 0.71 3.01
C1 3.25 4.04 4.25 1.06 4.51 3.81 0.35 4.59 4.45 0.9  3.58 4.6  4.25
Se unen los clusters 8 y 3 con distancia de 0.25 para formar C2

Matriz de distancias 3 :
   2    4    5    7    9    10   11   12   13   14   15   C1   C2
2
```

```
4  2.69
5  2.3  3.25
7  2.55 0.5  3.47
9  2.93 3.91 0.71 4.16
10 2.85 0.35 3.58 0.35 4.25
11 2.26 0.79 3.4  0.35 4.1  0.71
12 2.37 3.48 0.25 3.69 0.56 3.81 3.61
13 0.35 2.57 2.61 2.37 3.25 2.69 2.06 2.69
14 3.25 0.56 3.64 0.9  4.27 0.56 1.25 3.88 3.13
15 1.06 2.47 3.25 2.15 3.91 2.5  1.8  3.35 0.71 3.01
C1 3.25 4.25 1.06 4.51 0.35 4.59 4.45 0.9  3.58 4.6  4.25
C2 0.79 2.85 3.09 2.57 3.72 2.92 2.24 3.16 0.5  3.4  0.56 4.04
Se unen los clusters 12 y 5 con distancia de 0.25 para formar C3

Matriz de distancias 4 :
   2    4    7    9    10   11   13   14   15   C1   C2   C3
2
4  2.69
7  2.55 0.5
9  2.93 3.91 4.16
10 2.85 0.35 0.35 4.25
11 2.26 0.79 0.35 4.1  0.71
13 0.35 2.57 2.37 3.25 2.69 2.06
14 3.25 0.56 0.9  4.27 0.56 1.25 3.13
15 1.06 2.47 2.15 3.91 2.5  1.8  0.71 3.01
C1 3.25 4.25 4.51 0.35 4.59 4.45 3.58 4.6  4.25
C2 0.79 2.85 2.57 3.72 2.92 2.24 0.5  3.4  0.56 4.04
C3 2.37 3.48 3.69 0.71 3.81 3.61 2.69 3.88 3.35 1.06 3.16
Se unen los clusters 13 y 2 con distancia de 0.35 para formar C4

Matriz de distancias 5 :
   4    7    9    10   11   14   15   C1   C2   C3   C4
4
7  0.5
9  3.91 4.16
10 0.35 0.35 4.25
11 0.79 0.35 4.1  0.71
14 0.56 0.9  4.27 0.56 1.25
15 2.47 2.15 3.91 2.5  1.8  3.01
C1 4.25 4.51 0.35 4.59 4.45 4.6  4.25
C2 2.85 2.57 3.72 2.92 2.24 3.4  0.56 4.04
C3 3.48 3.69 0.71 3.81 3.61 3.88 3.35 1.06 3.16
C4 2.69 2.55 3.25 2.85 2.26 3.25 1.06 3.58 0.79 2.69
Se unen los clusters 10 y 4 con distancia de 0.35 para formar C5

Matriz de distancias 6 :
   7    9    11   14   15   C1   C2   C3   C4   C5
7
```

```
9  4.16
11 0.35 4.1
14 0.9  4.27 1.25
15 2.15 3.91 1.8  3.01
C1 4.51 0.35 4.45 4.6  4.25
C2 2.57 3.72 2.24 3.4  0.56 4.04
C3 3.69 0.71 3.61 3.88 3.35 1.06 3.16
C4 2.55 3.25 2.26 3.25 1.06 3.58 0.79 2.69
C5 0.5  4.25 0.79 0.56 2.5  4.59 2.92 3.81 2.85
Se unen los clusters 11 y 7 con distancia de 0.35 para formar C6

Matriz de distancias 7 :
   9    14   15   C1   C2   C3   C4   C5   C6
9
14 4.27
15 3.91 3.01
C1 0.35 4.6  4.25
C2 3.72 3.4  0.56 4.04
C3 0.71 3.88 3.35 1.06 3.16
C4 3.25 3.25 1.06 3.58 0.79 2.69
C5 4.25 0.56 2.5  4.59 2.92 3.81 2.85
C6 4.16 1.25 2.15 4.51 2.57 3.69 2.55 0.79
Se unen los clusters C1 y 9 con distancia de 0.35 para formar C7

Matriz de distancias 8 :
   14   15   C2   C3   C4   C5   C6   C7
14
15 3.01
C2 3.4  0.56
C3 3.88 3.35 3.16
C4 3.25 1.06 0.79 2.69
C5 0.56 2.5  2.92 3.81 2.85
C6 1.25 2.15 2.57 3.69 2.55 0.79
C7 4.6  4.25 4.04 1.06 3.58 4.59 4.51
Se unen los clusters C5 y 14 con distancia de 0.56 para formar C8

Matriz de distancias 9 :
   15   C2   C3   C4   C6   C7  C8
15
C2 0.56
C3 3.35 3.16
C4 1.06 0.79 2.69
C6 2.15 2.57 3.69 2.55
C7 4.25 4.04 1.06 3.58 4.51
C8 3.01 3.4  3.88 3.25 1.25 4.6
Se unen los clusters C2 y 15 con distancia de 0.56 para formar C9

Matriz de distancias 10 :
```

```
   C3   C4   C6   C7   C8  C9
C3
C4 2.69
C6 3.69 2.55
C7 1.06 3.58 4.51
C8 3.88 3.25 1.25 4.6
C9 3.35 1.06 2.57 4.25 3.4
Se unen los clusters C7 y C3 con distancia de 1.06 para formar C10

Matriz de distancias 11 :
    C4   C6   C8  C9   C10
C4
C6  2.55
C8  3.25 1.25
C9  1.06 2.57 3.4
C10 3.58 4.51 4.6 4.25
Se unen los clusters C9 y C4 con distancia de 1.06 para formar C11

Matriz de distancias 12 :
    C6   C8  C10  C11
C6
C8  1.25
C10 4.51 4.6
C11 2.57 3.4 4.25
Se unen los clusters C8 y C6 con distancia de 1.25 para formar C12

Matriz de distancias 13 :
    C10  C11 C12
C10
C11 4.25
C12 4.6  3.4
Se unen los clusters C12 y C11 con distancia de 3.4 para formar C13

Matriz de distancias 14 :
    C10 C13
C10
C13 4.6
Se unen los clusters C13 y C10 con distancia de 4.6 para formar C14

Resumen:
Iteración 1 : Se unen 6 y 1 con distancia 0.25 para formar C1
Iteración 2 : Se unen 8 y 3 con distancia 0.25 para formar C2
Iteración 3 : Se unen 12 y 5 con distancia 0.25 para formar C3
Iteración 4 : Se unen 13 y 2 con distancia 0.35 para formar C4
Iteración 5 : Se unen 10 y 4 con distancia 0.35 para formar C5
Iteración 6 : Se unen 11 y 7 con distancia 0.35 para formar C6
Iteración 7 : Se unen C1 y 9 con distancia 0.35 para formar C7
Iteración 8 : Se unen C5 y 14 con distancia 0.56 para formar C8
```

```
Iteración 9 : Se unen C2 y 15 con distancia 0.56 para formar C9
Iteración 10 : Se unen C7 y C3 con distancia 1.06 para formar C10
Iteración 11 : Se unen C9 y C4 con distancia 1.06 para formar C11
Iteración 12 : Se unen C8 y C6 con distancia 1.25 para formar C12
Iteración 13 : Se unen C12 y C11 con distancia 3.4 para formar C13
Iteración 14 : Se unen C13 y C10 con distancia 4.6 para formar C14
```

- Ejemplo de uso con el método de distancia "GROUP_AVERAGE"

```
> algoritmo_aglomerativo(datos, "GROUP_AVERAGE")

Matriz de distancias 1 :
   1    2    3    4    5    6    7    8    9    10   11   12   13   14   15
1
2  3.02
3  3.81 0.79
4  4.14 2.69 2.85
5  0.9  2.3  3.09 3.25
6  0.25 3.25 4.04 4.25 1.06
7  4.37 2.55 2.57 0.5  3.47 4.51
8  3.58 0.56 0.25 2.7  2.85 3.81 2.46
9  0.25 2.93 3.72 3.91 0.71 0.35 4.16 3.48
10 4.47 2.85 2.92 0.35 3.58 4.59 0.35 2.8  4.25
11 4.3  2.26 2.24 0.79 3.4  4.45 0.35 2.14 4.1  0.71
12 0.71 2.37 3.16 3.48 0.25 0.9  3.69 2.93 0.56 3.81 3.61
13 3.35 0.35 0.5  2.57 2.61 3.58 2.37 0.25 3.25 2.69 2.06 2.69
14 4.51 3.25 3.4  0.56 3.64 4.6  0.9  3.26 4.27 0.56 1.25 3.88 3.13
15 4.03 1.06 0.5  2.47 3.25 4.25 2.15 0.56 3.91 2.5  1.8  3.35 0.71 3.01
Se unen los clusters 6 y 1 con distancia de 0.25 para formar C1

Matriz de distancias 2 :
   2    3    4    5    7    8    9    10   11   12   13   14   15   C1
2
3  0.79
4  2.69 2.85
5  2.3  3.09 3.25
7  2.55 2.57 0.5  3.47
8  0.56 0.25 2.7  2.85 2.46
9  2.93 3.72 3.91 0.71 4.16 3.48
10 2.85 2.92 0.35 3.58 0.35 2.8  4.25
11 2.26 2.24 0.79 3.4  0.35 2.14 4.1  0.71
12 2.37 3.16 3.48 0.25 3.69 2.93 0.56 3.81 3.61
13 0.35 0.5  2.57 2.61 2.37 0.25 3.25 2.69 2.06 2.69
14 3.25 3.4  0.56 3.64 0.9  3.26 4.27 0.56 1.25 3.88 3.13
15 1.06 0.5  2.47 3.25 2.15 0.56 3.91 2.5  1.8  3.35 0.71 3.01
C1 3.14 3.92 4.19 0.98 4.44 3.69 0.3  4.53 4.38 0.8  3.47 4.55 4.14
Se unen los clusters 8 y 3 con distancia de 0.25 para formar C2

Matriz de distancias 3 :
```

```
   2    4    5    7    9    10   11   12   13   14   15   C1   C2
2
4  2.69
5  2.3  3.25
7  2.55 0.5  3.47
9  2.93 3.91 0.71 4.16
10 2.85 0.35 3.58 0.35 4.25
11 2.26 0.79 3.4  0.35 4.1  0.71
12 2.37 3.48 0.25 3.69 0.56 3.81 3.61
13 0.35 2.57 2.61 2.37 3.25 2.69 2.06 2.69
14 3.25 0.56 3.64 0.9  4.27 0.56 1.25 3.88 3.13
15 1.06 2.47 3.25 2.15 3.91 2.5  1.8  3.35 0.71 3.01
C1 3.14 4.19 0.98 4.44 0.3  4.53 4.38 0.8  3.47 4.55 4.14
C2 0.67 2.78 2.97 2.52 3.6  2.86 2.19 3.04 0.38 3.33 0.53 3.81
Se unen los clusters 12 y 5 con distancia de 0.25 para formar C3

Matriz de distancias 4 :
   2    4    7    9    10   11   13   14   15   C1   C2   C3
2
4  2.69
7  2.55 0.5
9  2.93 3.91 4.16
10 2.85 0.35 0.35 4.25
11 2.26 0.79 0.35 4.1  0.71
13 0.35 2.57 2.37 3.25 2.69 2.06
14 3.25 0.56 0.9  4.27 0.56 1.25 3.13
15 1.06 2.47 2.15 3.91 2.5  1.8  0.71 3.01
C1 3.14 4.19 4.44 0.3  4.53 4.38 3.47 4.55 4.14
C2 0.67 2.78 2.52 3.6  2.86 2.19 0.38 3.33 0.53 3.81
C3 2.34 3.37 3.58 0.63 3.69 3.5  2.65 3.76 3.3  0.89 3.01
Se unen los clusters C1 y 9 con distancia de 0.3 para formar C4

Matriz de distancias 5 :
   2    4    7    10   11   13   14   15   C2   C3   C4
2
4  2.69
7  2.55 0.5
10 2.85 0.35 0.35
11 2.26 0.79 0.35 0.71
13 0.35 2.57 2.37 2.69 2.06
14 3.25 0.56 0.9  0.56 1.25 3.13
15 1.06 2.47 2.15 2.5  1.8  0.71 3.01
C2 0.67 2.78 2.52 2.86 2.19 0.38 3.33 0.53
C3 2.34 3.37 3.58 3.69 3.5  2.65 3.76 3.3  3.01
C4 3.07 4.1  4.35 4.44 4.28 3.39 4.46 4.06 3.74 0.81
Se unen los clusters 13 y 2 con distancia de 0.35 para formar C5

Matriz de distancias 6 :
```

```
   4    7    10   11   14   15   C2   C3   C4   C5
4
7  0.5
10 0.35 0.35
11 0.79 0.35 0.71
14 0.56 0.9  0.56 1.25
15 2.47 2.15 2.5  1.8  3.01
C2 2.78 2.52 2.86 2.19 3.33 0.53
C3 3.37 3.58 3.69 3.5  3.76 3.3  3.01
C4 4.1  4.35 4.44 4.28 4.46 4.06 3.74 0.81
C5 2.63 2.46 2.77 2.16 3.19 0.88 0.52 2.49 3.23
Se unen los clusters 10 y 4 con distancia de 0.35 para formar C6

Matriz de distancias 7 :
   7    11   14   15   C2   C3   C4   C5  C6
7
11 0.35
14 0.9  1.25
15 2.15 1.8  3.01
C2 2.52 2.19 3.33 0.53
C3 3.58 3.5  3.76 3.3  3.01
C4 4.35 4.28 4.46 4.06 3.74 0.81
C5 2.46 2.16 3.19 0.88 0.52 2.49 3.23
C6 0.43 0.75 0.56 2.49 2.82 3.53 4.27 2.7
Se unen los clusters 11 y 7 con distancia de 0.35 para formar C7

Matriz de distancias 8 :
   14   15   C2   C3   C4   C5   C6   C7
14
15 3.01
C2 3.33 0.53
C3 3.76 3.3  3.01
C4 4.46 4.06 3.74 0.81
C5 3.19 0.88 0.52 2.49 3.23
C6 0.56 2.49 2.82 3.53 4.27 2.7
C7 1.08 1.98 2.35 3.54 4.32 2.31 0.59
Se unen los clusters C5 y C2 con distancia de 0.52 para formar C8

Matriz de distancias 9 :
   14   15   C3   C4   C6   C7   C8
14
15 3.01
C3 3.76 3.3
C4 4.46 4.06 0.81
C6 0.56 2.49 3.53 4.27
C7 1.08 1.98 3.54 4.32 0.59
C8 3.26 0.71 2.75 3.48 2.76 2.33
Se unen los clusters C6 y 14 con distancia de 0.56 para formar C9
```

```
Matriz de distancias 10 :
   15   C3   C4   C7   C8   C9
15
C3 3.3
C4 4.06 0.81
C7 1.98 3.54 4.32
C8 0.71 2.75 3.48 2.33
C9 2.66 3.61 4.33 0.75 2.93
Se unen los clusters C8 y 15 con distancia de 0.71 para formar C10

Matriz de distancias 11 :
    C3   C4   C7   C9   C10
C3
C4  0.81
C7  3.54 4.32
C9  3.61 4.33 0.75
C10 2.86 3.6  2.26 2.87
Se unen los clusters C9 y C7 con distancia de 0.75 para formar C11

Matriz de distancias 12 :
    C3   C4   C10  C11
C3
C4  0.81
C10 2.86 3.6
C11 3.58 4.33 2.63
Se unen los clusters C4 y C3 con distancia de 0.81 para formar C12

Matriz de distancias 13 :
    C10  C11  C12
C10
C11 2.63
C12 3.3  4.03
Se unen los clusters C11 y C10 con distancia de 2.63 para formar C13

Matriz de distancias 14 :
    C12  C13
C12
C13 3.67
Se unen los clusters C13 y C12 con distancia de 3.67 para formar C14

Resumen:
Iteración 1 : Se unen 6 y 1 con distancia 0.25 para formar C1
Iteración 2 : Se unen 8 y 3 con distancia 0.25 para formar C2
Iteración 3 : Se unen 12 y 5 con distancia 0.25 para formar C3
Iteración 4 : Se unen C1 y 9 con distancia 0.3 para formar C4
Iteración 5 : Se unen 13 y 2 con distancia 0.35 para formar C5
Iteración 6 : Se unen 10 y 4 con distancia 0.35 para formar C6
```

```
Iteración 7 : Se unen 11 y 7 con distancia 0.35 para formar C7
Iteración 8 : Se unen C5 y C2 con distancia 0.52 para formar C8
Iteración 9 : Se unen C6 y 14 con distancia 0.56 para formar C9
Iteración 10 : Se unen C8 y 15 con distancia 0.71 para formar C10
Iteración 11 : Se unen C9 y C7 con distancia 0.75 para formar C11
Iteración 12 : Se unen C4 y C3 con distancia 0.81 para formar C12
Iteración 13 : Se unen C11 y C10 con distancia 2.63 para formar C13
Iteración 14 : Se unen C13 y C12 con distancia 3.67 para formar C14
```

2.2. Análisis de clasificación supervisada

2.2.1. Árboles de decisión

El tercer conjunto de datos, que se empleará para realizar el análisis de clasificación supervisada utilizando árboles de decisión, estará formado por el siguiente conjunto de 10 sucesos constituidos por los valores de cuatro características de vehículos: 1.{B,4,5,Coche}; 2.{A,2,2,Moto}; 3.{N,2,1,Bicicleta}; 4.{B,6,4,Camión}; 5.{B,4,6,Coche}; 6.{B,4,4,Coche}; 7.{N,2,2,Bicicleta}; 8.{B,2,1,Moto}; 9.{B,6,2,Camión}; 10.{N,2,1,Bicicleta}, donde las características de cada suceso son: {TipoCarnet, NúmeroRuedas, NúmeroPasajeros, TipoVehículo}. Se debe clasificar el tipo de vehículo en función del resto de características. TipoCarnet, es el tipo de carnet necesario para conducir el vehículo.

Solución:

- Primero instalamos la libreria "readr´´ y se leen los datos del archivo .csv para guardarlos en la variable "datos´´.

```
> library(readr)
> (datos=read.csv('Datos2.3.csv'))
```

```
   TipoCarnet NumeroRuedas NumeroPasajeros TipoVehiculo
1           B            4               5        Coche
2           A            2               2         Moto
3           N            2               1    Bicicleta
4           B            6               4       Camion
5           B            4               6        Coche
6           B            4               4        Coche
7           N            2               2    Bicicleta
8           B            2               1         Moto
9           B            6               2       Camion
10          N            2               1    Bicicleta
```

- Después de importar los datos pasamos los datos que no son tipo Integer, a tipo factor, ya que si no no podremos realizar el análisis correctamente.

```
> TipoCarnet=factor(datos$TipoCarnet)
> TipoVehiculo=factor(datos$TipoVehiculo)
```

- La variable tipo factor es de tipo categórica usada asiduamente para clasificaciones de datos. Sus valores se almacenan internamente en R como números enteros. Tras al conversión de tipos, creamos un dataframe con las nuevas variables y con las que no han sido cambiadas.

```
> muestra=data.frame(TipoCarnet=TipoCarnet, NumeroRuedas=datos$NumeroRuedas, Nume
```

- Una vez hecho esto procedemos a instalar el paquete "tree´´, el cual construye árboles de decisión y haciendo uso de la función `tree()`. La función tree recibe la variable sobre la que queremos predecir, en este caso (TipoVehiculo), además de usar ".´´ para que prediga respecto al resto de variables. También recibe el dataset que hemos creado previamente y del cual obtiene todos los datos. Por último recibe

la función para que el árbol se ajuste a los datos en cuestión, `tree.control()`, que recibe los parámetros el número de observaciones (10), `minsize=2` y `mindev=0`.

```
> library(tree)
> arbol <- tree(TipoVehiculo ~ ., data=muestra, control=tree.control(10,minsize=2
```

- Para finalizar visualizamos el árbol resultante con `print()`, muestra las variables de los nodos internos y los nodos terminales.

```
> print(arbol)

node), split, n, deviance, yval, (yprob)
      * denotes terminal node

1) root 10 27.32 Bicicleta ( 0.3 0.2 0.3 0.2 )
  2) NumeroRuedas < 3 5  6.73 Bicicleta ( 0.6 0.0 0.0 0.4 )
    4) TipoCarnet: A,B 2  0.00 Moto ( 0.0 0.0 0.0 1.0 ) *
    5) TipoCarnet: N 3  0.00 Bicicleta ( 1.0 0.0 0.0 0.0 ) *
  3) NumeroRuedas > 3 5  6.73 Coche ( 0.0 0.4 0.6 0.0 )
    6) NumeroRuedas < 5 3  0.00 Coche ( 0.0 0.0 1.0 0.0 ) *
    7) NumeroRuedas > 5 2  0.00 Camion ( 0.0 1.0 0.0 0.0 ) *
```

2.2.2. Regresión

El cuarto conjunto de datos, que se empleará para realizar el análisis de clasificación supervisada utilizando regresión, estará formado por los siguientes 4 subconjuntos de datos: 1.{10, 8.04; 8, 6.95; 13, 7.58; 9, 8.81; 11, 8.33; 14, 9.96; 6, 7.24; 4, 4.26; 12, 10.84; 7, 4.82; 5, 5.68}; 2.{10, 9.14; 8, 8.14; 13, 8.74; 9, 8.77; 11, 9.26; 14, 8.1; 6, 6.13; 4, 3.1; 12, 9.13; 7, 7.26; 5, 4.74}; 3.{10, 7.46; 8, 6.77; 13, 12.74; 9, 7.11; 11, 7.81; 14, 8.84; 6, 6.08; 4, 5.39; 12, 8.15; 7, 6.42; 5, 5.73}; 4.{8, 6.58; 8, 5.76; 8, 7.71; 8, 8.84; 8, 8.47; 8, 7.04; 8, 5.25; 19, 12.5; 8, 5.56; 8, 7.91; 8, 6.89}. Se deben calcular las rectas de regresión de los cuatro subconjuntos y sus parámetros de ajuste.

Solución: Algoritmo de minería de reglas de asociación (apriori): Su objetivo principal es descubrir patrones de asociación entre diferentes conjuntos de datos.

- Para empezar tenemos que introducir los datos en un archivo con extensión .txt

```
> source("regresion_lineal.R")
> source("R-squared.R")
> # Leer el archivo del primer subconjunto de datos
> muestra <- read.table("datos2.4.1.txt")
> muestra2 <- read.table("datos2.4.2.txt")
> muestra3 <- read.table("datos2.4.3.txt")
> muestra4 <- read.table("datos2.4.4.txt")
> # Llamar a la función de regresión lineal
> resultado <- regresion_lineal(muestra$x,muestra$y)
> resultado2 <- regresion_lineal(muestra2$x,muestra2$y)
> resultado3 <- regresion_lineal(muestra3$x,muestra3$y)
> resultado4 <- regresion_lineal(muestra4$x,muestra4$y)
```

```
> # Mostrar los coeficientes de las rectas
> resultado

$alpha
[1] 3.000091

$beta
[1] 0.5000909

> resultado2

$alpha
[1] 3.000909

$beta
[1] 0.5

> resultado3

$alpha
[1] 3.002455

$beta
[1] 0.4997273

> resultado4

$alpha
[1] 3.001727

$beta
[1] 0.4999091

> # Llamar a la función de R-squared
> Rsquared <- calculo_R_cuadrado(muestra$x,muestra$y,resultado$alpha,resultado$be
> Rsquared2 <- calculo_R_cuadrado(muestra2$x,muestra2$y,resultado2$alpha,resultac
> Rsquared3 <- calculo_R_cuadrado(muestra3$x,muestra3$y,resultado3$alpha,resultac
> Rsquared4 <- calculo_R_cuadrado(muestra4$x,muestra4$y,resultado4$alpha,resultac
> #Mostrar bonanza regresión
> print(paste("R^2:", Rsquared))

[1] "R^2: 0.666542459508774"

> print(paste("R^2:", Rsquared2))

[1] "R^2: 0.666242033727482"

> print(paste("R^2:", Rsquared3))

[1] "R^2: 0.666324041066559"
```

```
> print(paste("R^2:", Rsquared4))

[1] "R^2: 0.666707256898466"
```

- Retorno:
  - La función `regresion_lineal()` retorna una lista con los coeficientes alpha y beta.
  - La función `calculo_R_cuadrado()` retorna un double con el valor del ajuste (R-squared).
  - La función `calculoSSR()` retorna un double con la dispersión de y calculados.
  - La función `calculoSSy()` retorna un double con la dispersión de y observados.
- Explicación: La función `regresion_lineal()`, calcula los coeficientes alpha y beta de la recta que forman los datos introducidos por parámetros. Después la función `calculo_R_cuadrado()` recibe como parámetros dichos coeficientes más los datos, con ello hace uso de las funciones `calculoSSR()` (que calcula la dispersión de los valores de y calculados a través de la recta creada con alpha y beta) y `calculoSSy()` (que calcula la dispersión de los valores y observados de los datos iniciales) para calcular R-squared, que se mide entre 0 y 1 y muestra el ajuste de la recta sobre los datos iniciales.