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. Introducción - Consideraciones previas **2**
1.1. Uso de RStudio . . . 2
1.2. Introducción de datos de Excel/CSV en R . . . 2

2. Ejercicios con ayuda del profesor **4**
2.1. Análisis de clasificación no supervisada . . . 4
2.1.1. k-Means . . . 4
2.1.2. Clusterización Jerárquica Aglomerativa . . . 7
2.2. Análisis de clasificación supervisada . . . 14
2.2.1. Árboles de decisión . . . 14
2.2.2. Regresión . . . 14

3. Ejercicios de forma autónoma **17**
3.1. Análisis de clasificación no supervisada . . . 17
3.1.1. K-means . . . 17
3.1.2. Clusterización Jerárquica Aglomerativa . . . 21
3.2. Análisis de clasificación supervisada . . . 24
3.2.1. Árboles de decisión . . . 24
3.2.2. Regresión . . . 25

1. Introducción - Consideraciones previas

1.1. Uso de RStudio

Para utilizar una función en R se escribe el nombre de la función, seguido de los parámetros de entrada entre paréntesis e.g.: `función(parámetros)`

- Función `contributors()`: Muestra los creadores del programa (R)
- Función `help()`: Abre un HTML con información sobre la función `help()` o de la función entre paréntesis de haberla. Para todas las funciones que programemos (para todas las que existan) en R debe poder usarse la función `help()`.

  En el archivo HTML se distinguen varios elementos:

  - función {paquete}: la función de la que se obtiene información seguida del paquete al que pertenece.
  - Description: descripción de la función.
  - Usage: aparece la función y todos los argumentos que se le pueden introducir.
  - Arguments: Explicación de los argumentos o parámetros.
  - Details: Detalles adicionales de la función.
  - Offline help: Ayuda sin conexión.
  - Note: Nota del autor.
  - References: Referencias.
  - Examples: Ejemplos de uso de la función.
- Función `getwd()` se utiliza para obtener el directorio de trabajo actual (working directory).
- Función `setwd("C:/...")` permite cambiar el nuevo directorio de trabajo en el que queramos trabajar.
- `help.start()`: Manda a un compendio de todas las ayudas disponibles para trabajar con R.
- Función `list.files()`: Muestra todos los archivos en el directorio. `dir()` hace lo mismo.

1.2. Introducción de datos de Excel/CSV en R

Para poder leer archivos .xlsx y .csv seguimos un par de pasos:

- Archivos .csv:
  - Primero se crea el archivo .csv y se introducen los datos por filas (los datos de la fila i, se escriben en la celda (i,1)) y separando dichos datos con un delimitador (",´´ o ";´´).

  - Después se usa la función `read.csv(''ruta del archivo´´, sep = '';´´)`, la cual es parte del paquete base de R, para leer los datos del archivo. Recibe como parámetros la ruta del archivo y el delimitador que usa para separar los datos.

- Archivos .xlsx:

  - Primero se crea el archivo .xlsc y se introducen los datos en forma de matriz, escribiendo cada dato en una celda.
  - Después se carga la librería *openxlsx* y se usa la función `read_excel(''ruta del archivo´´)`, introduciendo la ruta del archivo por parámetro.

2. 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.

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 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

```
> 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.

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, 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

```
> 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"
```

- : Explicacion

```
> 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
```

- : 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
```

- : Explicacion

```
> 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
```

- : Explicacion

```
> 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
```

- : Explicacion

  >

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 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 "*´´.

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 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
```

3. 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.

3.1. Análisis de clasificación no supervisada

3.1.1. 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

>
>
```

3.1.2. 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}. Del análisis visual de los datos se ha concluido que hay una alta probabilidad que sean tres clusters.

Solución:

- `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
```

3.2. Análisis de clasificación supervisada

3.2.1. 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 ) *
```

3.2.2. 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.