UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

Máster en Software de Sistemas Distribuidos y Empotrados

Ciencia de Datos

Proyecto Final – Clustering y Series Temporales

Alejandro Casanova Martín

N.º de matrícula: bu0383

Madrid, 29 de abril 2024

Índice

Índice de Figuras

Índice de Tablas

1. Clustering

Tenemos los datos de vinos de España de distintas denominaciones de origen, se requiere hacer una segmentación de vinos españoles en función de precio, calidad y acidez del vino.

a) Instalar el paquete *Tidyverse*.

```
>#install.packages("tidyverse") # install if necessary
>library(tidyverse)
```

b) Cargar el fichero en un *dataframe*, considerando el tipo de separador adecuado.

```
> script_dir <- "C:/Users/alex/Desktop/Máster Software Embebido/2
   Segundo Semestre/2 Ciencia de Datos/Ejercicios" # Actualizar con el
   directorio correcto
> setwd(script_dir); getwd()
[1] "C:/Users/alex/Desktop/Máster Software Embebido/2 Segundo
Semestre/2 Ciencia de Datos/Ejercicios"
> Frame0 <- as.data.frame(read.csv("Ficheros/wines_SPA_1.csv",
   header=TRUE, sep=';', dec ='.'))
> head(Frame0)
              winery                                wine year rating ...
1        Felix Solis                 Mucho Mas Tinto N,V,    4.2 ...
2           Baluarte                             Muscat 2020    4.2 ...
3              Mocen      Seleccion Especial Verdejo 2021    4.3 ...
4     Familia Torres          San Valentin Parellada 2019    4.3 ...
5 Marques de Caceres Rioja Satinela Blanco Semidulce 2020    4.2 ...
6             Lustau       Candela Cream Dulce Sweet N,V,    4.2 ...
```

c) Identificar la estructura de los datos y realizar la coerción de los datos que no se encuentren en formato numérico y que sean necesarios para la segmentación.

Primero extraemos las variables que nos interesa segmentar: 'price', 'rating' y 'acidity'. También mantendremos el campo 'region' (la denominación de origen), dado que tendremos que hacer un filtrado más adelante, y también para realizar comparaciones con los clusters obtenidos.

```
> SubFrame0 <-subset.data.frame(Frame0, select=c('price', 'rating',
   'acidity', 'region'))
> head(SubFrame0)
  price rating acidity             region
1  4.99    4.2    2.33     Vino de Espana
2   5.5    4.2    <NA>            Navarra
3  6.26    4.3    2.76              Rueda
4  6.95    4.3    <NA>           Cataluna
5  6.99    4.2    2.92              Rioja
6   7.1    4.2    2,00 Jerez-Xeres-Sherry
> sapply(SubFrame0[1,], class) # Comprobar el tipo de dato de las
   variables
      price      rating     acidity      region
"character"   "numeric" "character" "character"
```

Algunas de las variables son de tipo 'character' por lo que tendrán que ser convertidas a tipo numérico. Primero, creamos un mapa que traduzca a números muy pequeños los valores de la variable 'region', de modo que no tenga peso en el algoritmo de segmentación, pero se mantenga en el dataframe para poder ser analizada a posteriori su relación con los clusters.

```
> region_names <- unique(SubFrame0$region)
> head(region_names)
```

```
[1] "Vino de Espana"     "Navarra"
[3] "Rueda"              "Cataluna"
[5] "Rioja"              "Jerez-Xeres-Sherry"
> region_to_num_mapping <- setNames(1:length(region_names)*1e-8,
   region_names)
```

A continuación, creamos un nuevo dataframe, y realizamos las conversiones necesarias para que todas las variables sean de tipo numérico.

```
> SubFrame1 <- SubFrame0
> SubFrame1$region <- sapply(SubFrame1$region, function(x)
   region_to_num_mapping[as.character(x)])
> SubFrame1$price <- as.numeric(SubFrame1$price)
Warning message:
NAs introduced by coercion
> SubFrame1$acidity <- as.numeric(SubFrame1$acidity)
Warning message:
NAs introduced by coercion
> sapply(SubFrame1[1,], class) # Comprobar el tipo de dato de las
   variables
    price    rating   acidity    region
"numeric" "numeric" "numeric" "numeric"
> apply(SubFrame1, 2, range) # Comprobar el rango de las variables
     price rating acidity  region
[1,]    NA    4.2      NA 1.0e-08
[2,]    NA    4.9      NA 7.6e-07
```

Todas las variables son ahora de tipo numérico, pero como nos indica el aviso generado por R, había algunos campos no válidos en las variables 'price' y 'acidity' que han sido convertidos a NA. Analizando el dataframe original se ha comprobado que había campos vacíos y campos con el string "NA".

d) Eliminar aquellas filas que no contengan datos o que no estén disponibles.

```
> SubFrame2 <- subset.data.frame(SubFrame1, !is.na(price)
   & !is.na(rating) & !is.na(acidity) & !is.na(region))
> apply(SubFrame2, 2, range) # Comprobar el rango de las variables
       price rating acidity  region
[1,]    4.99    4.2    0.15 1.0e-08
[2,] 3119.08    4.9    3.99 7.6e-07
```

Como hemos eliminado los campos con NA, los rangos de las variables ahora aparecen correctamente. Finalmente, convertimos el dataframe a una matriz numérica, apta para el algoritmo kmeans.

```
> kmdata_orig = as.matrix(SubFrame2[,1:4])
> mode(kmdata_orig) = "numeric"
> head(kmdata_orig)
   price rating acidity region
1   4.99    4.2    2.33  1e-08
3   6.26    4.3    2.76  3e-08
5   6.99    4.2    2.92  5e-08
8   7.10    4.2    2.01  6e-08
9   7.10    4.2    2.01  6e-08
10  7.10    4.2    2.02  6e-08
```

e) Valorar si es necesario realizar un escalado de algunos de los datos: *price*, *rating*, *acidity*.

Será especialmente crítico realizar un escalado de la variable precio ya que, debido a la gran amplitud de su rango de valores, el resto de las variables podrían volverse irrelevantes

durante el algoritmo de clustering. Adicionalmente, dado que las tres variables son de naturalezas distintas, y tienen unidades diferentes, haremos también un normalizado de las variables 'rating' y 'acidity'. Si hubiese una relación espacial entre varias de las variables, no sería recomendable hacer un normalizado por evitar posibles distorsiones del dataset; pero como este no es el caso, hemos optado por normalizar las tres variables entre 0 y 1.

```
> NUM_VAL <- 3
> kmdata <- kmdata_orig
> max_val = numeric(NUM_VAL)
> min_val = numeric(NUM_VAL)
> for (k in 1:NUM_VAL) max_val[k] <- max(as.numeric(kmdata[,k]))
> for (k in 1:NUM_VAL) min_val[k] <- min(as.numeric(kmdata[,k]))
> kmdata[,1:NUM_VAL] <- scale(kmdata[,1:NUM_VAL], center=min_val,
   scale=(max_val-min_val))
> head(kmdata)
          price    rating   acidity region
1  0.0000000000 0.0000000 0.5677083  1e-08
3  0.0004078238 0.1428571 0.6796875  3e-08
5  0.0006422422 0.0000000 0.7213542  5e-08
8  0.0006775655 0.0000000 0.4843750  6e-08
9  0.0006775655 0.0000000 0.4843750  6e-08
10 0.0006775655 0.0000000 0.4869792  6e-08
> apply(kmdata, 2, range)
     price rating acidity  region
[1,]     0      0       0 1.0e-08
[2,]     1      1       1 7.6e-07
```

Podemos comprobar que el normalizado se ha realizado correctamente, y que las tres variables relevantes están comprendidas entre 0 y 1.

f) Haz un *clustering* de todos los vinos. Considera si puedes ponderar o escalar los datos de alguna manera razonable. Usa el método *elbow* para determinar el valor más apropiado de k.

Mantendremos el escalado del apartado anterior, de modo que las variables 'price', 'rating' y 'acidity' tengan la misma relevancia. La variable 'region' no tendrá impacto en el algoritmo, debido a su ínfima magnitud.

```
> wss <- numeric(8)
> for (k in 1:8) wss[k] <- (sum(kmeans(kmdata, centers=k,
   nstart=25)$withinss))
> #wss <- log(wss)
> plot(1:8, wss, type="b", xlab="Number of Clusters", ylab="Within Sum
   of Squares")
```

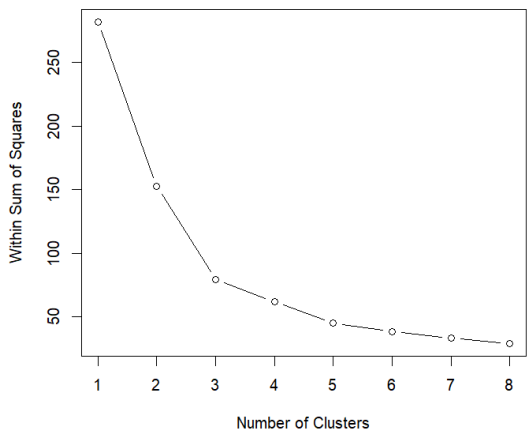


Figura 1. Representación del ajuste del algoritmo kmeans para diferente número de clusters.

Para k = 3 hay un cambio de pendiente significativo, por lo que nos decantaremos por este valor.

```
> km3 = kmeans(kmdata,3, nstart=25)
> km3
K-means clustering with 3 clusters of sizes 692, 1547, 3272

Cluster means:
        price     rating   acidity       region
1 0.064324194 0.44611891 0.8527667 1.712572e-07
2 0.016373878 0.02788808 0.8733234 1.288752e-07
3 0.005716464 0.01039120 0.6097188 1.615648e-07

Clustering vector:
   1    3    5    8    9   10   11   12   13   14   15
   3    3    3    3    3    3    3    3    3    3    3
  16   17   18   19   20   21   22   23   24   25   26
   3    3    3    3    3    3    3    3    3    3    3
...

Within cluster sum of squares by cluster:
[1] 37.14135 14.11527 27.98599
 (between_SS / total_SS =  71.9 %)
```

Parece que el ajuste es lo suficientemente bueno (71,9%) y los tres clusters tienen tamaños razonables (692, 1647 y 3272). A continuación, almacenamos los resultados del clustering en un dataframe, y devolvemos las variables a sus magnitudes originales (revertimos la normalización), para su mejor representación y análisis.

```
> NUM_CLUSTERS <- 3
> km_selected <- km3
> df <- as.data.frame(kmdata)
> df$cluster <- factor(km_selected$cluster)
> centers <- as.data.frame(km_selected$centers)
> # Devolvemos los datos a sus magnitudes originales
> df$region <- sapply(df$region, function(x)
   names(region_to_num_mapping[as.numeric(round(x*1e08))]))
> for (k in 1:NUM_VAL) df[,k] <- df[,k] * (max_val[k] - min_val[k]) +
   min_val[k] # Unscale
> for (k in 1:NUM_VAL) centers[,k] <- centers[,k] * (max_val[k] -
   min_val[k]) + min_val[k] # Unscale centers
> apply(df[,1:3], 2, range) # Comprobar el rango de las variables
       price rating acidity
[1,]    4.99    4.2    0.15
[2,] 3119.08    4.9    3.99
> centers
      price   rating  acidity       region
1 205.30133 4.512283 3.424624 1.712572e-07
2  55.97973 4.219522 3.503562 1.288752e-07
3  22.79158 4.207274 2.491320 1.615648e-07
```

También se ha aplicado la "desnormalización" a los centroides de los clusters. De esta manera, podemos apreciar, en sus unidades originales, los valores medios de cada variable para cada cluster. Finalmente, definimos y ejecutamos varias funciones para la representación de los resultados (estas funciones serán reutilizadas más adelante).

Se representarán las gráficas 'rating'-'price', 'acidity-rating' y 'acidity'-'price'. Adicionalmente, también se han generado las gráficas 'region'-'price' y 'region'-'rating'.

```
> library(ggplot2)
```

```
> library(grid) #gráficos en cuadrícula
> library(gridExtra)
> plotGraphs <- function(df, centers){
+
+   g1 <- ggplot(data=df, aes(x=rating, y=price, color=cluster )) +
+     geom_point() +
+     geom_point(data=centers, aes(x=rating, y=price,
   color=as.factor(1:NUM_CLUSTERS)),
+                size=10, alpha=.6, show.legend=FALSE)
+
+   g2 <- ggplot(data=df, aes(x=acidity, y=rating, color=cluster )) +
+     geom_point() +
+     geom_point(data=centers, aes(x=acidity, y=rating,
   color=as.factor(1:NUM_CLUSTERS)),
+                size=10, alpha=.6, show.legend=FALSE)
+
+   g3 <- ggplot(data=df, aes(x=acidity, y=price, color=cluster )) +
+     geom_point() +
+     geom_point(data=centers, aes(x=acidity, y=price,
   color=as.factor(1:NUM_CLUSTERS)),
+                size=10, alpha=.6, show.legend=FALSE)
+
+   g4 <- ggplot(data=df, aes(x=region, y=price, color=cluster )) +
+     geom_point() + theme(axis.text.x=element_blank())
+     #geom_point(data=centers, aes(x=region, y=price,
   color=as.factor(1:NUM_CLUSTERS)),
+     #           size=10, alpha=.6, show.legend=FALSE)
+
+   g5 <- ggplot(data=df, aes(x=region, y=rating, color=cluster )) +
+     geom_point() + theme(axis.text.x=element_blank())
+     #geom_point(data=centers, aes(x=region, y=rating,
   color=as.factor(1:NUM_CLUSTERS)),
+     #           size=10, alpha=.6, show.legend=FALSE)
+
+   grid.arrange(
+     arrangeGrob(g1 + theme(
+       legend.box.background = element_rect(),
+       legend.box.margin = margin(6, 6, 6, 6))),
+     arrangeGrob(g2 + theme(legend.position="none"),
+                 g3 + theme(legend.position="none"),
+                 ncol = 2)
+   )
+
+   return(list(g1,g2,g3,g4,g5))
+ }
> plotSingleGraph <- function(gg_element){
+   plot(gg_element + theme(
+     legend.box.background = element_rect(),
+     legend.box.margin = margin(6, 6, 6, 6)))
+ }
> g_i <- plotGraphs(df, centers)
```

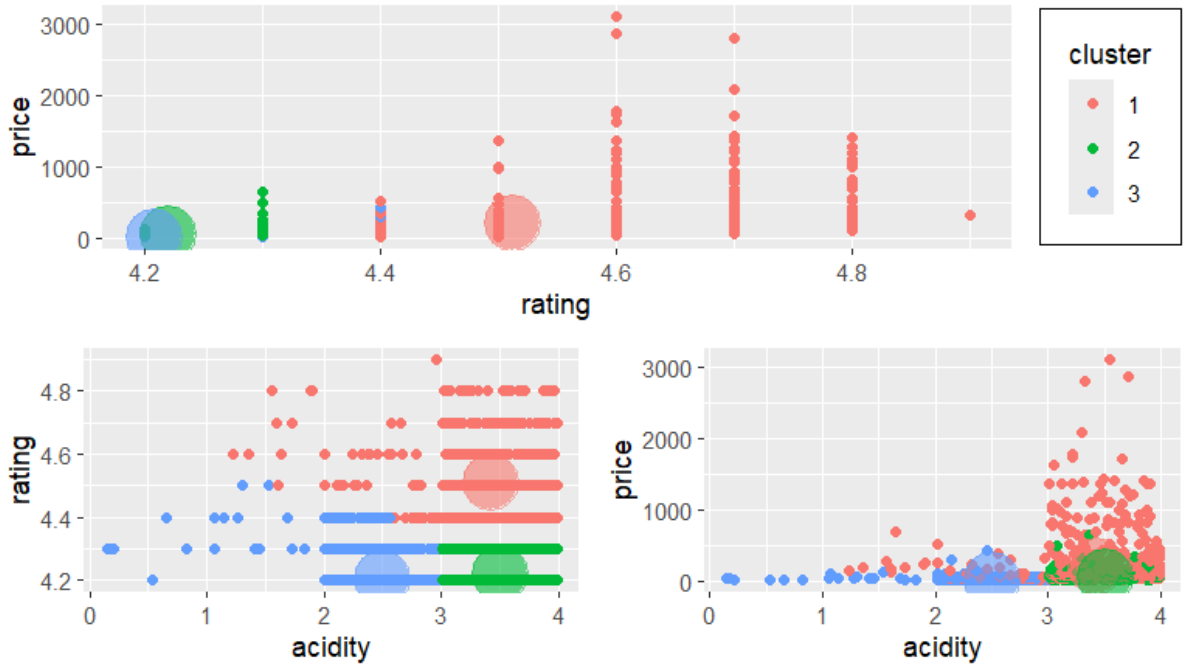


Figura 2. Visión global de los resultados del clustering.

```
> plotSingleGraph(g_i[[1]])
```

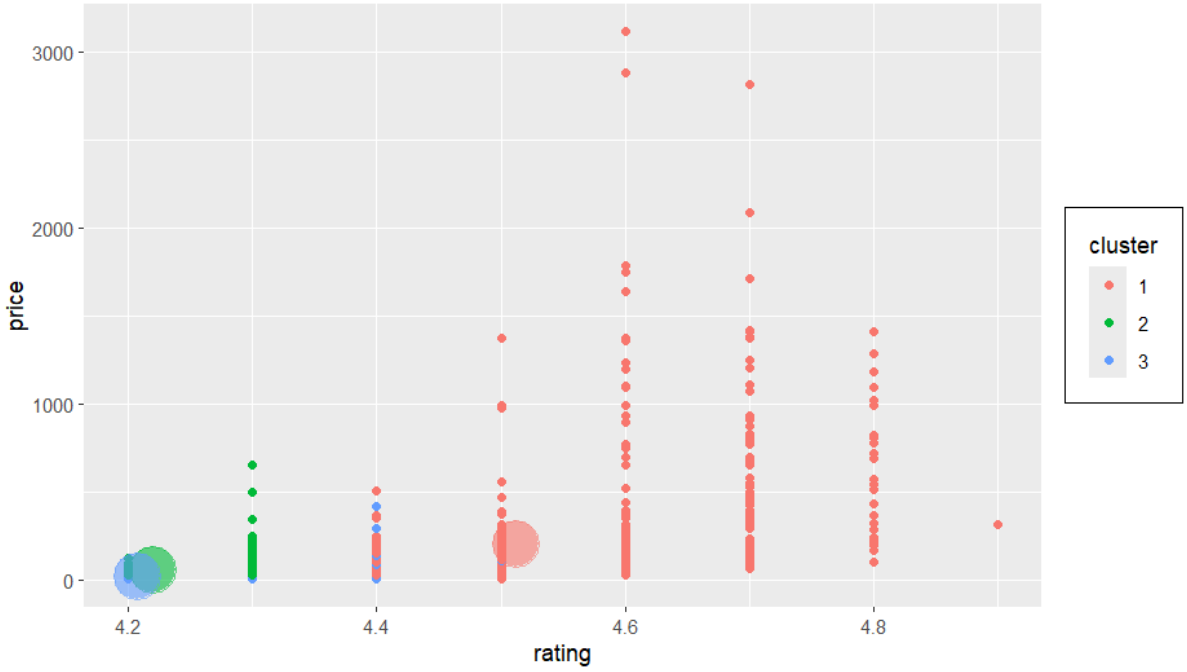


Figura 3. Resultados del clustering. Gráfica 'rating'-'price'.

```
> plotSingleGraph(g_i[[2]])
```

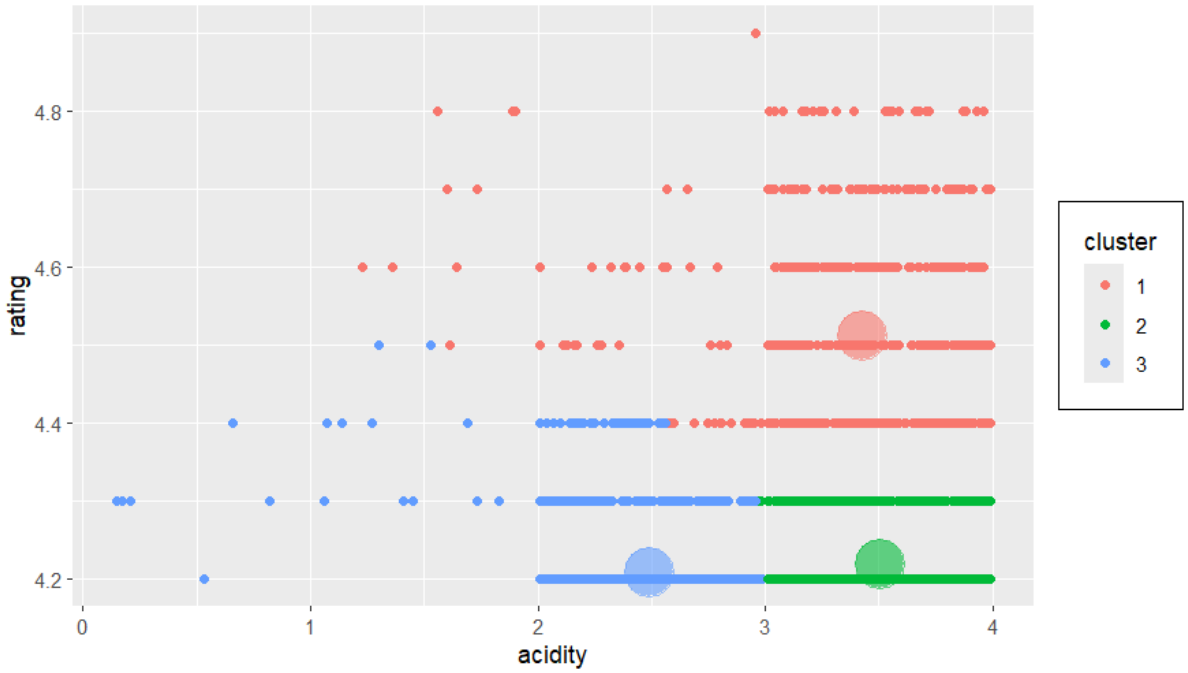


Figura 4. Resultados del clustering. Gráfica 'acidity'-'rating'.

```
> plotSingleGraph(g_i[[3]])
```

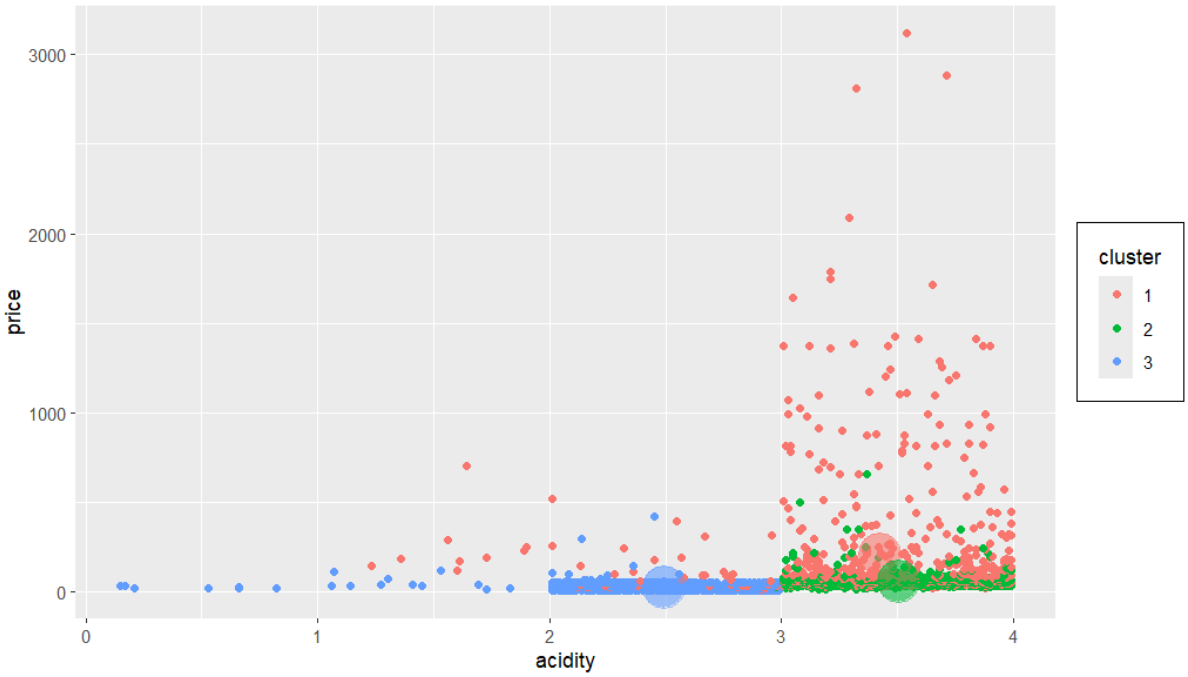


Figura 5. Resultados del clustering. Gráfica 'acidity'-'price'.

```
> plotSingleGraph(g_i[[4]])
```

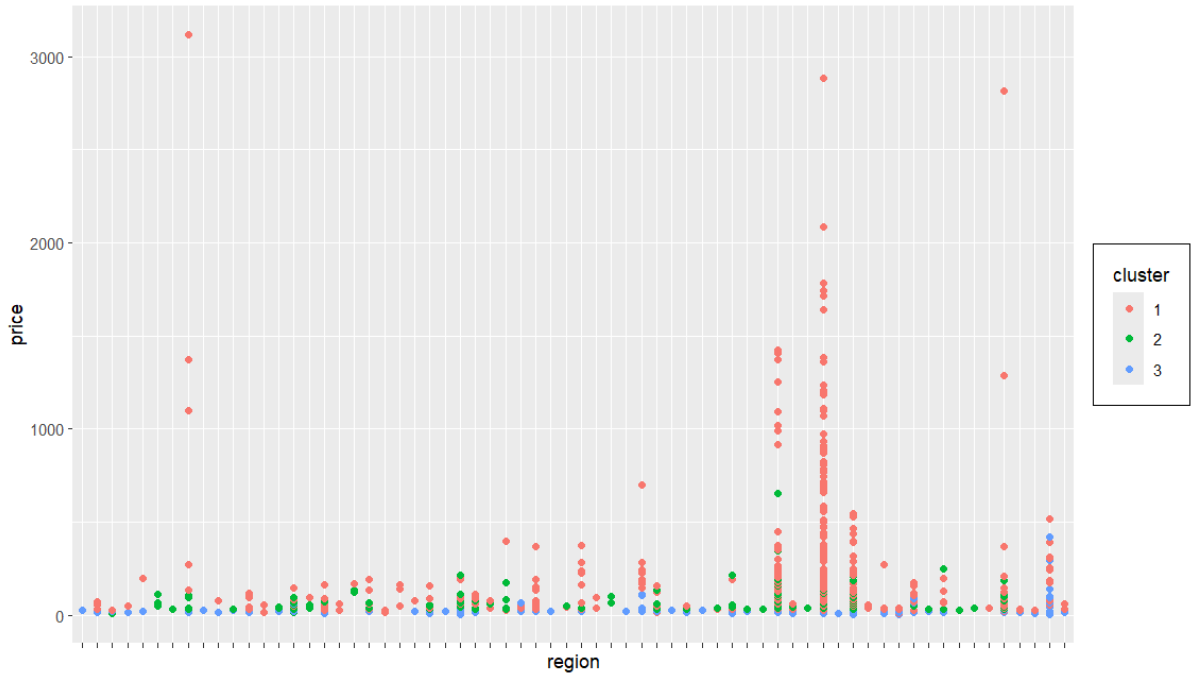


Figura 6. Resultados del clustering. Gráfica 'region'-'price'.

```
> plotSingleGraph(g_i[[5]])
```

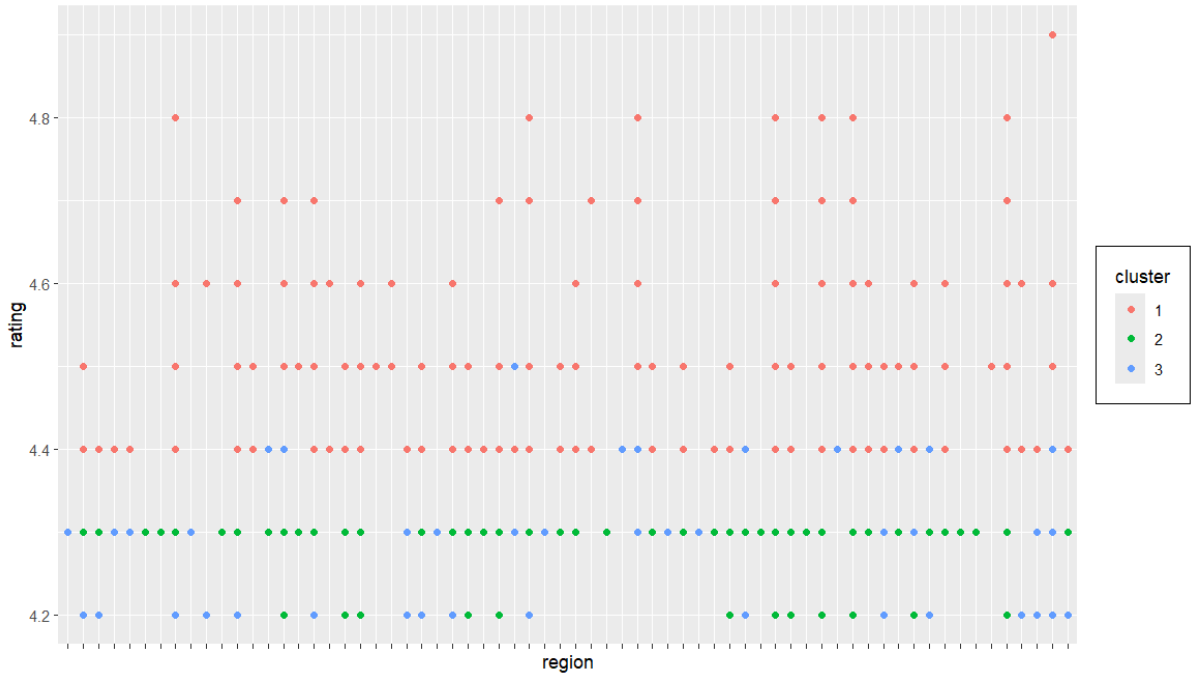


Figura 7. Resultados del clustering. Gráfica 'region'-'rating'.

g) Repite el proceso solo para los vinos de denominación "Rioja".

Filtramos los vinos de la denominación indicada y repetimos el proceso de preparación de los datos.

```
> kmdata_rioja = as.matrix(SubFrame_rioja[,1:4])
> mode(kmdata_rioja) = "numeric"
> head(kmdata_rioja)
    price rating acidity region
5    6.99    4.2    2.92  5e-08
234  9.88    4.3    2.77  5e-08
242 11.20    4.2    2.72  5e-08
480 12.90    4.3    2.13  5e-08
485 13.58    4.2    2.01  5e-08
486 13.58    4.2    2.01  5e-08
> apply(kmdata_rioja, 2, range)
      price rating acidity region
[1,]   6.99    4.2    1.83  5e-08
[2,] 544.50    4.8    3.99  5e-08
> # Scale the data
> NUM_VAL <- 3
> max_val_rioja = numeric(NUM_VAL)
> min_val_rioja = numeric(NUM_VAL)
> for (k in 1:NUM_VAL) max_val_rioja[k] <-
   max(as.numeric(kmdata_rioja[,k]))
> for (k in 1:NUM_VAL) min_val_rioja[k] <-
   min(as.numeric(kmdata_rioja[,k]))
> kmdata_rioja[,1:NUM_VAL] <- scale(kmdata_rioja[,1:NUM_VAL],
+                                   center=min_val_rioja,
+                                   scale=(max_val_rioja-
   min_val_rioja))
> head(kmdata_rioja)
          price    rating    acidity region
5   0.000000000 0.0000000 0.50462963  5e-08
234 0.005376644 0.1666667 0.43518519  5e-08
242 0.007832412 0.0000000 0.41203704  5e-08
480 0.010995144 0.1666667 0.13888889  5e-08
485 0.012260237 0.0000000 0.08333333  5e-08
486 0.012260237 0.0000000 0.08333333  5e-08
> apply(kmdata_rioja, 2, range)
     price rating acidity region
[1,]     0      0       0  5e-08
[2,]     1      1       1  5e-08
```

A continuación, aplicamos nuevamente el método elbow para escoger k.

```
> wss_r <- numeric(8)
> for (k in 1:8) wss_r[k] <- (sum(kmeans(kmdata_rioja, centers=k,
   nstart=25)$withinss))
> #wss_r <- log(wss_r)
> plot(1:8, wss_r, type="b", xlab="Number of Clusters", ylab="Within
   Sum of Squares")
```

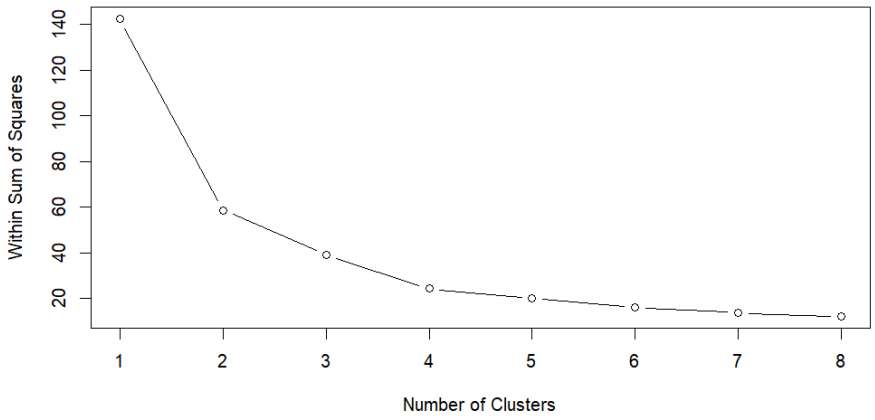


Figura 8. Representación del ajuste del algoritmo kmeans para diferente número de clusters (para sólo los vinos con denominación Rioja).

Nos quedaremos nuevamente con el valor k=3.

```
> km3_rioja = kmeans(kmdata_rioja, 3, nstart=25)
> km3_rioja
K-means clustering with 3 clusters of sizes 363, 665, 689

Cluster means:
       price      rating   acidity region
1 0.12645131 0.211662075 0.8124426  5e-08
2 0.03631152 0.005513784 0.4580688  5e-08
3 0.02816103 0.004354136 0.2051215  5e-08

Within cluster sum of squares by cluster:
[1] 29.345418  5.534277  4.316516
 (between_SS / total_SS =  72.5 %)
```

Se puede comprobar que el ajuste es bueno (72,5%) y que los tamaños de los clusters son razonables (363, 665 y 689).

Seguidamente, almacenamos los resultados en un dataframe, y revertimos la normalización para apreciar las verdaderas magnitudes de cada variable y representarlas apropiadamente.

```
> NUM_CLUSTERS <- 3
> km_selected_r <- km3_rioja
> df_rioja <- as.data.frame(kmdata_rioja)
> df_rioja$cluster <- factor(km_selected_r$cluster)
> centers_rioja <- as.data.frame(km_selected_r$centers)
> # Devolvemos los datos a sus magnitudes originales
> df_rioja$region <- sapply(df_rioja$region, function(x)
   names(region_to_num_mapping[as.numeric(round(x*1e08))]))
> for (k in 1:NUM_VAL) df_rioja[,k] <- df_rioja[,k] * (max_val_rioja[k]
   - min_val_rioja[k]) + min_val_rioja[k] # Unscale
> for (k in 1:NUM_VAL) centers_rioja[,k] <- centers_rioja[,k] *
   (max_val_rioja[k] - min_val_rioja[k]) + min_val_rioja[k] # Unscale
   centers
> apply(df_rioja[,1:3], 2, range) # Comprobar el rango de las variables
      price rating acidity
[1,]   6.99    4.2    1.83
[2,] 544.50    4.8    3.99
> centers_rioja
     price   rating  acidity region
```

```
1 74.95884 4.326997 3.584876  5e-08
2 26.50780 4.203308 2.819429  5e-08
3 22.12684 4.202612 2.273062  5e-08
```

Finalmente, representamos las diferentes gráficas de los resultados:

```
> g_i_rioja <- plotGraphs(df_rioja, centers_rioja)
```

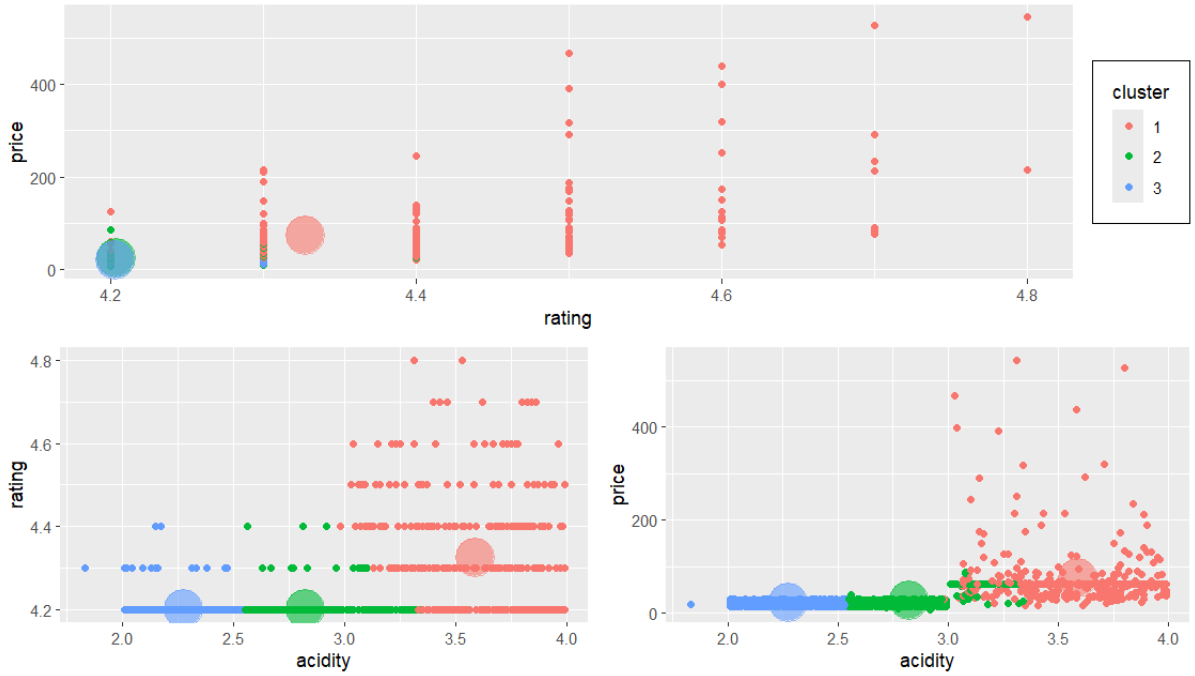


Figura 9. Visión global de los resultados del clustering (para sólo los vinos de denominación rioja).

```
> plotSingleGraph(g_i_rioja[[1]])
```

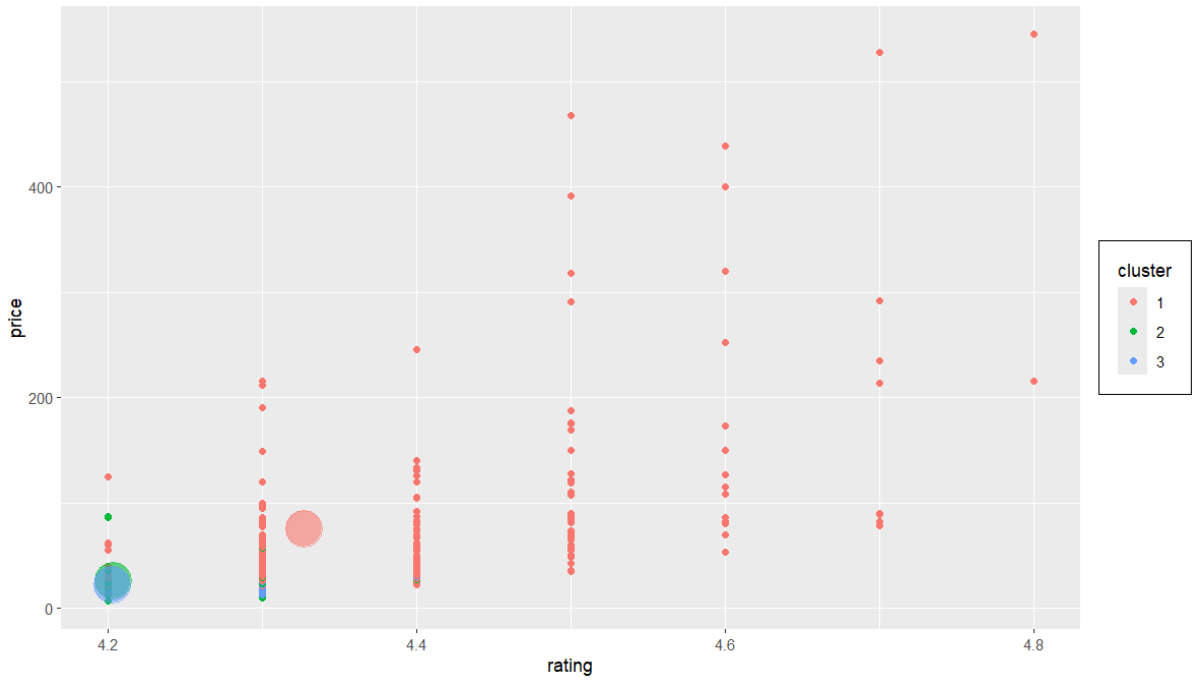


Figura 10. Resultados del clustering (para sólo los vinos de denominación 'Rioja'). Gráfica 'rating'-'price'.

```
> plotSingleGraph(g_i_rioja[[2]])
```

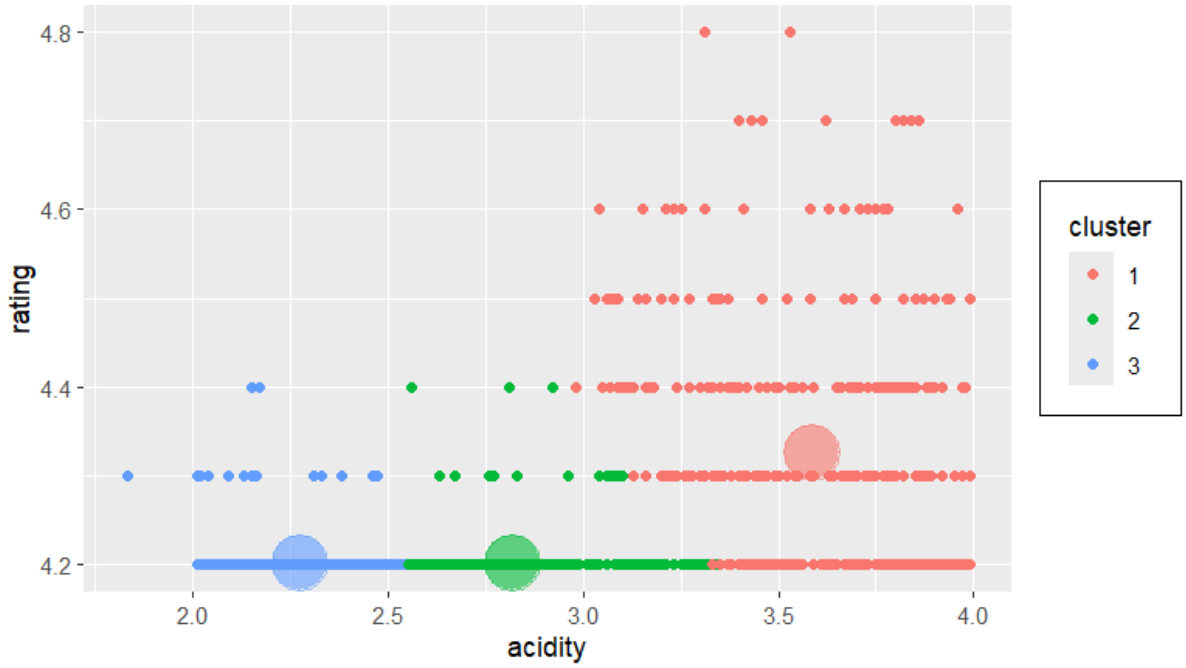


Figura 11. Resultados del clustering (para sólo los vinos de denominación 'Rioja'). Gráfica 'rating'-'price'.

```
> plotSingleGraph(g_i_rioja[[3]])
```

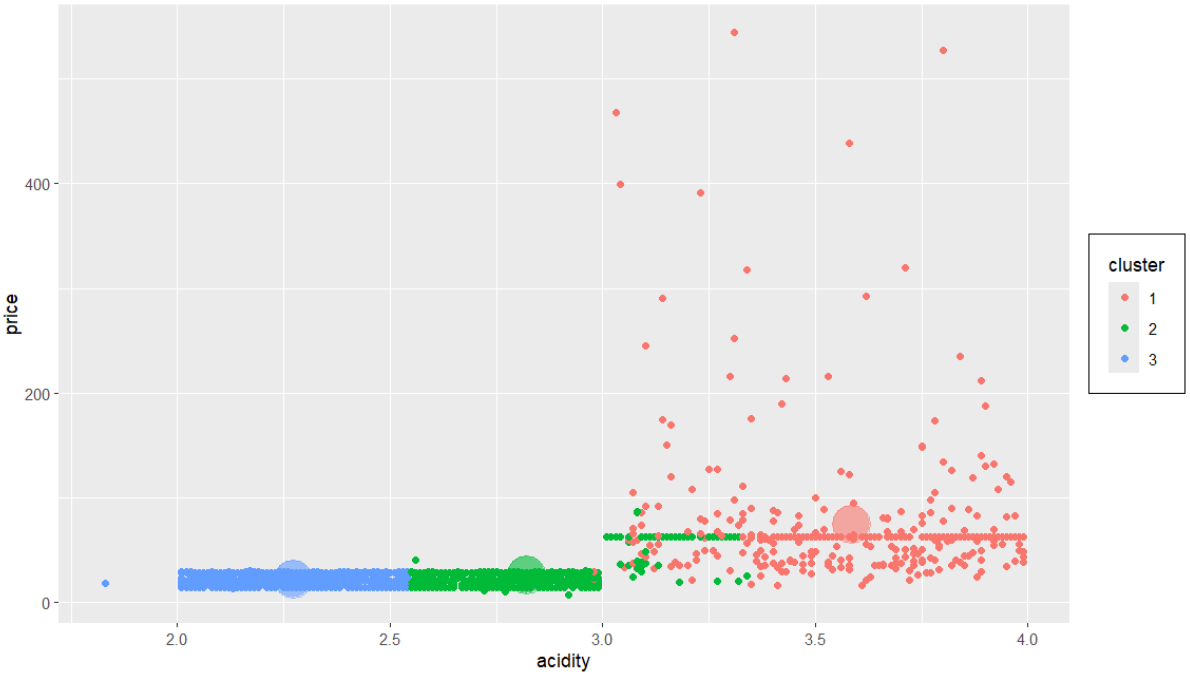


Figura 12. Resultados del clustering (para sólo los vinos de denominación 'Rioja'). Gráfica 'acidity'-'price'.

h) Estudia las características de los clústeres obtenidos y comenta las observaciones y conclusiones que se pueden extraer.

A continuación se muestra un resumen de las características de los clusters obtenidos, así como medidas estadísticas para cada variable del dataset completo y el dataset con únicamente los vinos de denominación 'Rioja':

Valores de los centroides (valores medios por cluster)			
cluster	***price***	***rating***	***acidity***
1	*205,30133*	*4,512283*	*3,424624*
2	*55,97973*	*4,219522*	*3,503562*
3	*22,79158*	*4,207274*	*2,491320*

Tabla 1. Valores de los centroides para el dataset completo.

Valores de los centroides (valores medios por cluster) para el dataset 'Rioja'			
cluster	***price***	***rating***	***acidity***
1	*74.95884*	*4.326997*	*3.584876*
2	*26.50780*	*4.203308*	*2.819429*
3	*22.12684*	*4.202612*	*2.273062*

Tabla 2. Valores de los centroides para el dataset con sólo los vinos de denominación 'Rioja'.

Medidas estadísticas del dataset completo			
	price	***rating***	***acidity***
Valor mínimo	*4,99*	*4,2*	*0,15*
Valor máximo	*3119,08*	*4,9*	*3,99*
Media	*55.025081*	*4.249011*	*2.892660*
Mediana	*28.30*	*4.20*	*2.83*

Tabla 3. Medidas estadísticas (máximo, mínimo, media y mediana) de las variables del dataset completo.

Medidas estadísticas del dataset 'Rioja'			
	price	***rating***	***acidity***
Valor mínimo	*6,99*	*4,2*	*1,83*
Valor máximo	*544,50*	*4,8*	*3,99*
Media	*34,993093*	*4,229179*	*2,762009*
Mediana	*28.30*	*4.20*	*2.68*

Tabla 4. Medidas estadísticas (máximo, mínimo, media y mediana) del dataset con sólo los vinos de denominación 'Rioja'.

Analizando los resultados del clustering del dataset completo (resumidos en la Tabla 1 y en la Figura 2), se pueden extraer las siguientes conclusiones sobre la naturaleza de cada cluster:

- **Cluster 1 (rojo):** agrupa los vinos con, generalmente, un valor mayor de las tres variables, especialmente del precio y la puntuación.
- **Cluster 2 (verde):** incluye vinos con generalmente un grado de acidez similar o incluso superior al cluster 1, pero precios y puntuaciones menores.
- **Cluster 3 (azul):** incluye vinos con puntuación similar al cluster 2, pero acidez y precio menores.

Para el caso del dataset de vinos 'Rioja' se extraen conclusiones similares, pero con algunas diferencias. Observando la Tabla 3 y la Tabla 4 se puede apreciar que el valor máximo de precio se ha reducido considerablemente, así como el valor medio, que ahora se encuentra más próximo a la mediana. También el valor mínimo de acidez ha aumentado. Podemos suponer que, al excluir del dataset los vinos de denominación diferente a 'Rioja', se han

eliminado también ciertos valores atípicos o "outliers" (valores de precio demasiado altos y valores de acidez demasiado bajos). Estos valores atípicos pueden apreciarse especialmente en la dimensión del precio (ver Figura 3). Como sabemos, el algoritmo k-means es especialmente sensible a la presencia de valores atípicos, dado que consiste en la minimización del cuadrado de las distancias a los centroides. Otros algoritmos como k-medians o k-medoids podrían ser más apropiados para realizar la segmentación de datasets como este, que tengan alta presencia de valores atípicos. También ponderando los precios mediante la aplicación de un logaritmo natural, se pudo apreciar que se reducía el impacto de los valores atípicos en la segmentación, aunque no en tanta medida como limitando el dataset a la denominación "Rioja". Se mantuvo por lo tanto la normalización original de los datos del dataset completo para realizar una mejor comparación con los resultados de la segmentación del dataset 'Rioja', dado que la aplicación del logaritmo natural al precio demostró no tener un impacto tan significativo.

Si calculamos la proporción de cada denominación de origen con respecto al dataset completo obtenemos que las primeras 5 denominaciones de origen más frecuentes suponen el 70,55% del total del dataset. Además, la denominación 'Ribera' es la más habitual, suponiendo el 31,16% del dataset (ver Figura 13), por lo que es la mejor opción para generar un sub-dataset, del mayor tamaño posible, pero con una única denominación. Al calcular el histograma de precios del dataset completo, comprobamos que el 94,34% de los vinos se encuentran por debajo de los 100€, el 97,02% se encuentran por debajo de los 200€, y el 98,62% se encuentran por debajo de los 500€ (ver Figura 14 y Figura 15). Tomando este 1,38% del total de vinos, que son los que se encuentran por encima de los 500€ (que es aproximadamente el límite superior del nuevo dataset 'Rioja'), calculamos las frecuencias de cada denominación de origen, y observamos que de los 76 vinos extraídos, 50 son 'Ribera del Duero', 17 son 'Priorato' y tan sólo 2 son 'Rioja' (ver Figura 16). Por lo tanto, hemos comprobado que la mayoría de los valores atípicos han quedado excluidos al tomar sólo los vinos 'Rioja'.

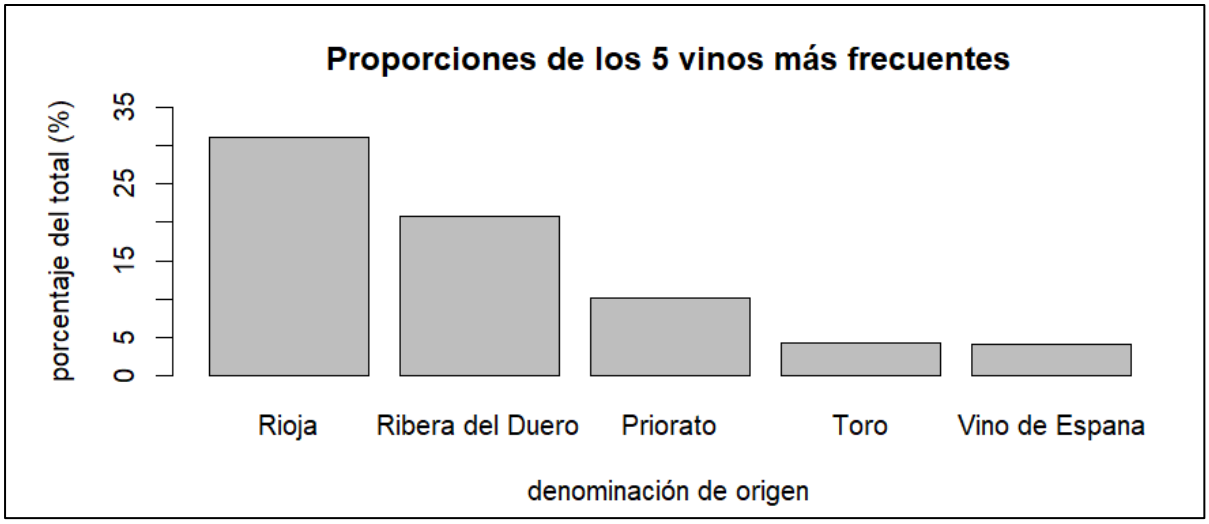


Figura 13. Gráfica de frecuencias de las denominaciones de origen (mostrando las cinco más habituales). El eje y mide el porcentaje con respecto al total.

Las principal diferencia en el clustering realizado para el dataset 'Rioja' es que, debido a la reducción de los valores atípicos de precio, el centroide del cluster 1 (rojo) se ha visto considerablemente desplazado hacia abajo en la dimensión precio, y también en cierta medida en la dimensión 'rating'. Como consecuencia, el centroide 2 (verde) se ha visto también desplazado hacia abajo en la dimensión 'price' y hacia la izquierda en la dimensión

'acidez'. Finalmente, el centroide 3 (azul) se ha visto ligeramente desplazado hacia la izquierda en la dimensión acidez. Pueden apreciarse estas diferencias comparando tanto las Tablas 1 y 2, como las Figuras 2 y 9.

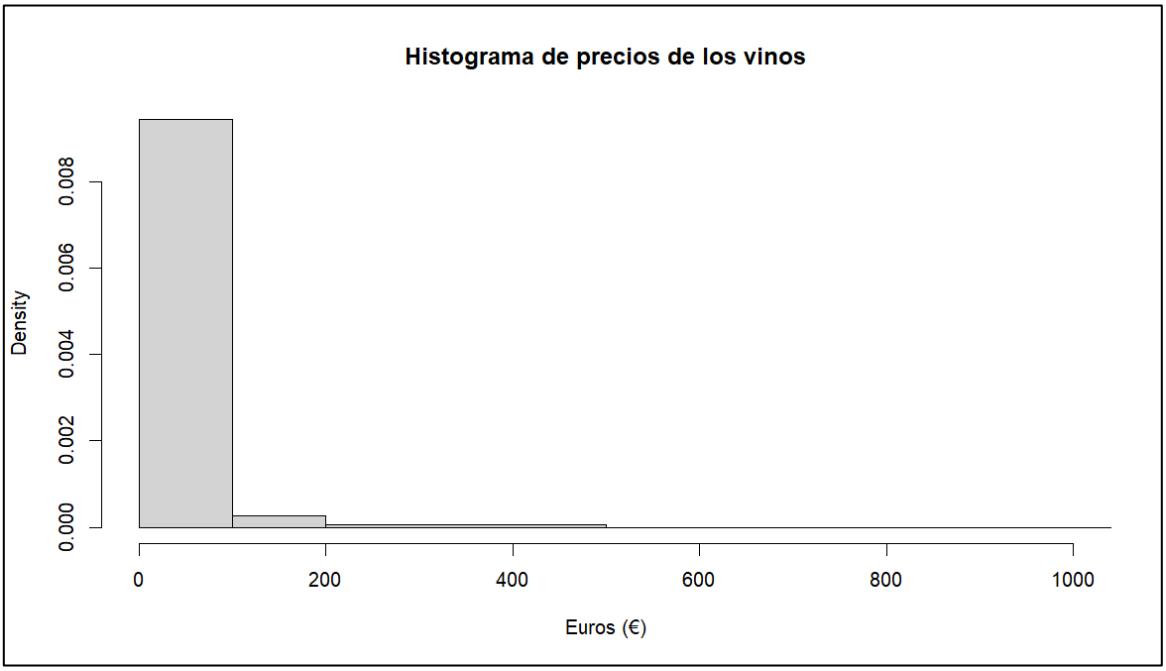


Figura 14. Histograma de precios de los vinos del dataset completo.

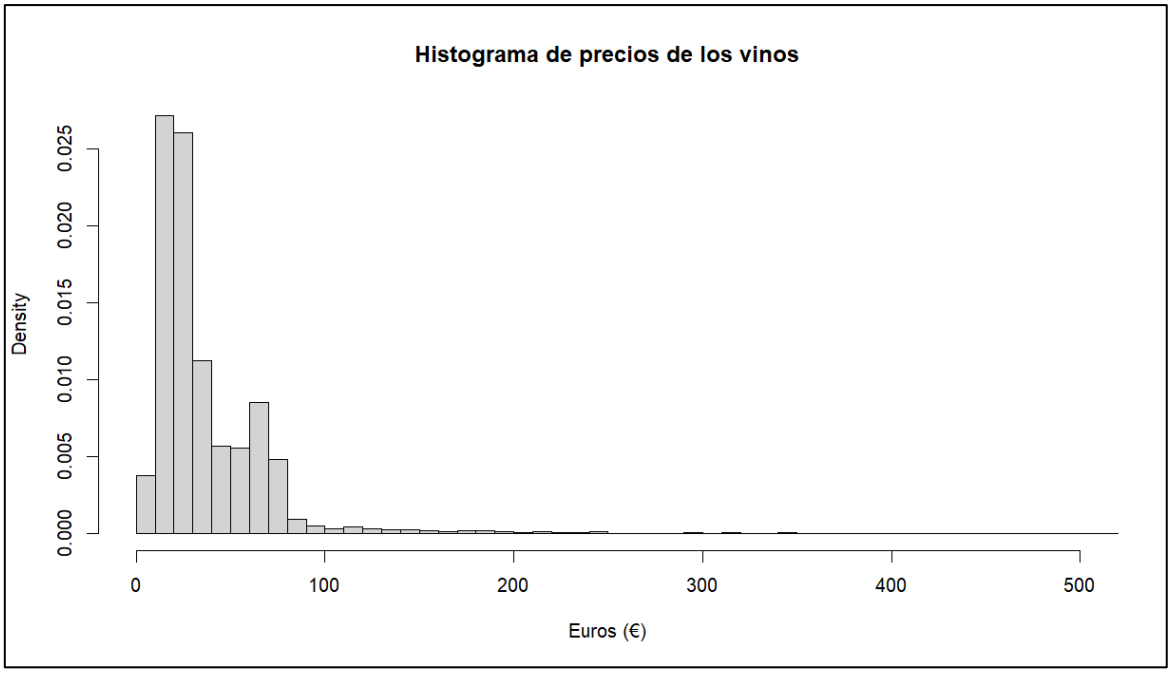


Figura 15. Histograma de precios de los vinos del dataset completo (ampliado).

Considerando que los centroides calculados para el dataset 'Rioja' estarán menos influenciados por valores atípicos, concluimos que la naturaleza de estos es la siguiente:

- **Cluster 1 (rojo):** agrupa los vinos con, generalmente, un valor mayor de las tres variables, especialmente del precio y puntuación..
- **Cluster 2 (verde):** incluye vinos con valores intermedios de precio y acidez, y una puntación menor que el cluster 1.

- **Cluster 3 (azul):** incluye vinos con menor acidez y precio, pero una puntuación similar a los del cluster 2.

La segmentación parece haber separado el dataset principalmente en tres regiones con respecto la acidez. Las dos regiones menos ácidas parecen tener puntuaciones similares, aunque la región más ácida presenta precios ligeramente superiores. A partir del grado de acidez 3, comienzan a aparecer puntuaciones mayores, y el valor de la variable 'price' se dispara y se vuelve especialmente volátil.

Hay una correlación positiva evidente entre las tres variables, pero no queda claro el motivo del comportamiento de la variable 'price' dentro del cluster 1. Podemos intuir que se debe a la influencia de otras variables que no se han tenido en cuanta en la segmentación. Por un lado, podría deberse a la calidad de la uva, y al prestigio de determinadas bodegas y denominaciones de origen. Como puede verse en la Figura 6, una gran parte de los valores atípicos de precio se agrupan en unas pocas denominaciones de origen. Sin embargo, en la Figura 7, se observa que las puntaciones más altas se encuentran mejor repartidas entre todas las denominaciones.

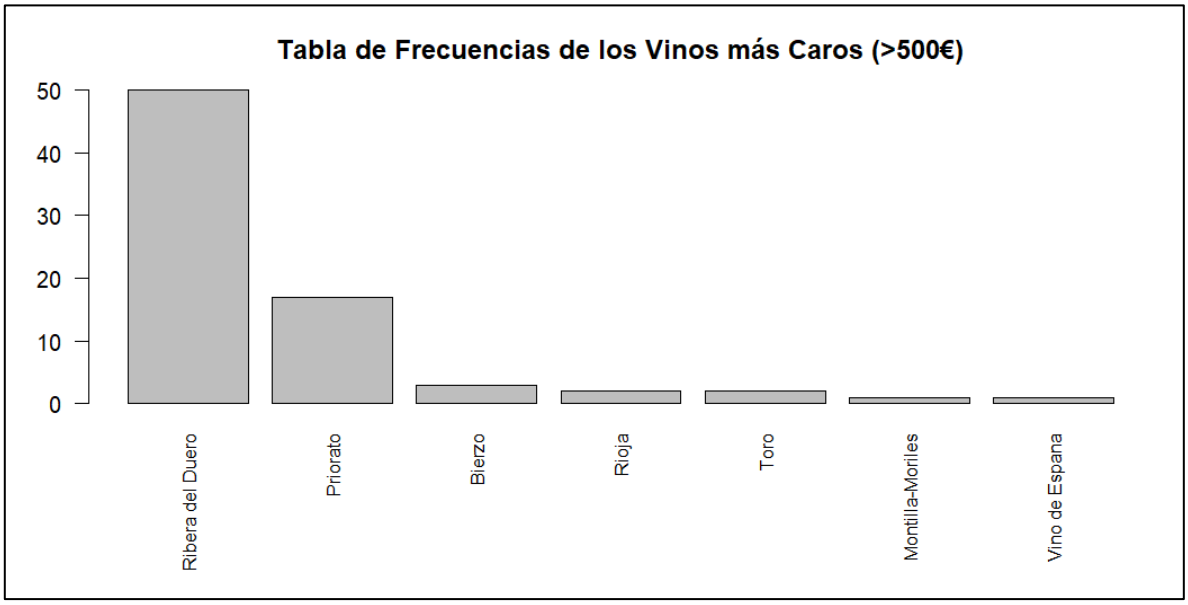


Figura 16. Tabla de frecuencias de los vinos más caros (>500€).

Finalmente, podemos teorizar que el precio desorbitado de algunos de los vinos podría deberse a otra variable que no se ha tenido en cuenta durante la segmentación: la edad/antigüedad del vino. Es bien sabida la relación existente entre la antigüedad de un vino y su precio. Además, también es conocida la fuerte influencia que pueden tener sobre la capacidad de envejecimiento y conservación de un vino, ciertas variables como el nivel de alcohol, la calidad del corcho del tapón, y el grado de acidez. Teniendo esto en cuenta, podemos especular que, tal vez, el precio desorbitado de algunos de los vinos más ácidos posiblemente se deba a que su acidez les haya permitido alcanzar un mayor grado de envejecimiento, o les dote del potencial de ello.

2. Series Temporales

Tenemos los datos del precio de venta del excedente de sistemas fotovoltaicos a la red eléctrica nacional, para distintos periodos de tiempo y con distinta resolución temporal.

Datos Semanales

a) Cargue el archivo de datos *'precios_excedente_peninsula_semana.xls'* y guárdelo en un dataframe.

```
> #install.packages("forecast") # install, if necessary
> library(forecast)
> #install.packages("readxl") # install, if necessary
> library("readxl")
> script_dir <- "C:/Users/alex/Desktop/Máster Software Embebido/2
   Segundo Semestre/2 Ciencia de Datos/Ejercicios" # Actualizar con el
   directorio correcto
> setwd(script_dir); getwd()
[1] "C:/Users/alex/Desktop/Máster Software Embebido/2 Segundo
Semestre/2 Ciencia de Datos/Ejercicios"
> precios_semana_input <-
   as.data.frame(read_excel("Ficheros/precios_excedente_peninsula_seman
   a.xlsx"))
> head(precios_semana_input)
             datetime  value
1 2024-01-08 00:00:00 130.53
2 2024-01-08 01:00:00 128.24
3 2024-01-08 01:59:59 126.22
4 2024-01-08 02:59:59 123.32
5 2024-01-08 03:59:59 122.81
6 2024-01-08 04:59:59 126.14
```

b) ¿Cuántos días de datos hay disponibles? ¿Con qué resolución? Guarde un 20% de los mismos para test, y el resto como datos de entrenamiento.

Hay un total de 5 días de datos en el dataset, con una resolución de una hora, es decir, 24 datos por día. Un 20% de los datos supone reservar para test el último día de los 5 de los que disponemos.

```
> precios_semana_input_train <- precios_semana_input[1:96,] # El 80% de
   los datos -> 96/120
> precios_semana_input_test <- precios_semana_input[97:120,] # El 20%
   restante
> precios_semana <- ts(as.numeric(precios_semana_input_train[,2]),
   start=8, frequency = 24)
> precios_semana_test <-
   ts(as.numeric(precios_semana_input_test[,2]),start=12, frequency =
   24)
```

c) Ajuste un modelo ARIMA posiblemente estacional (¿periodo?) de forma manual, justificando los parámetros escogidos. Realice también un ajuste con autoarima sin aproximación.

```
> plot(precios_semana, xlab = "Tiempo (días)",
+      ylab = "Precios Excedente Eléctrico (€)")
```

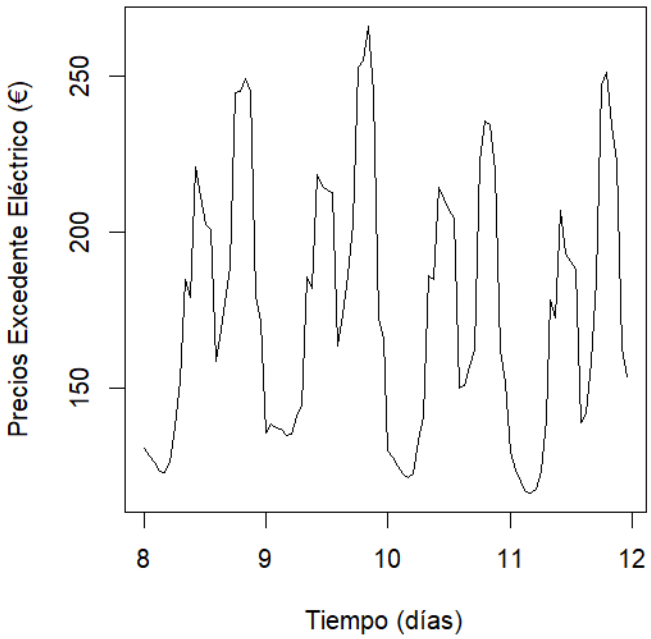


Figura 17. Representación del dataset de entrenamiento de los datos semanales del precio del excedente eléctrico.

Puede observarse en la Figura 17 que existe un patrón estacional diario. Dado que la resolución es horaria, el valor de *s* será 24. A continuación se realizan varias diferenciaciones para determinar los valores de *d* y *D*.

```
> plot(diff(precios_semana,differences=1), main="d=1, D=0", ylab="")
> abline(a=0, b=0)
> var(diff(precios_semana,differences=1))
[1] 626.7807
> plot(diff(precios_semana,differences=2), main="d=2, D=0", ylab="")
> abline(a=0, b=0)
> var(diff(precios_semana,differences=2))
[1] 1025.812
> plot(diff(precios_semana,lag=24,differences=1), main="d=0, D=1,
   s=24", ylab="")
> abline(a=0, b=0)
> var(diff(precios_semana,lag=24,differences=1))
[1] 159.174
> plot(diff(precios_semana,lag=24,differences=2), main="d=0, D=2,
   s=24", ylab="")
> abline(a=0, b=0)
> var(diff(precios_semana,lag=24,differences=2))
[1] 561.3828
> plot(diff(diff(precios_semana,differences=1),lag=24,differences=1),
   main="d=1, D=1, s=24", ylab="")
> abline(a=0, b=0)
> var(diff(diff(precios_semana,differences=1),lag=24,differences=1))
[1] 49.95462
> plot(diff(diff(precios_semana,differences=2),lag=24,differences=1),
   main="d=2, D=1, s=24", ylab="")
> abline(a=0, b=0)
> var(diff(diff(precios_semana,differences=2),lag=24,differences=1))
[1] 97.80032
```

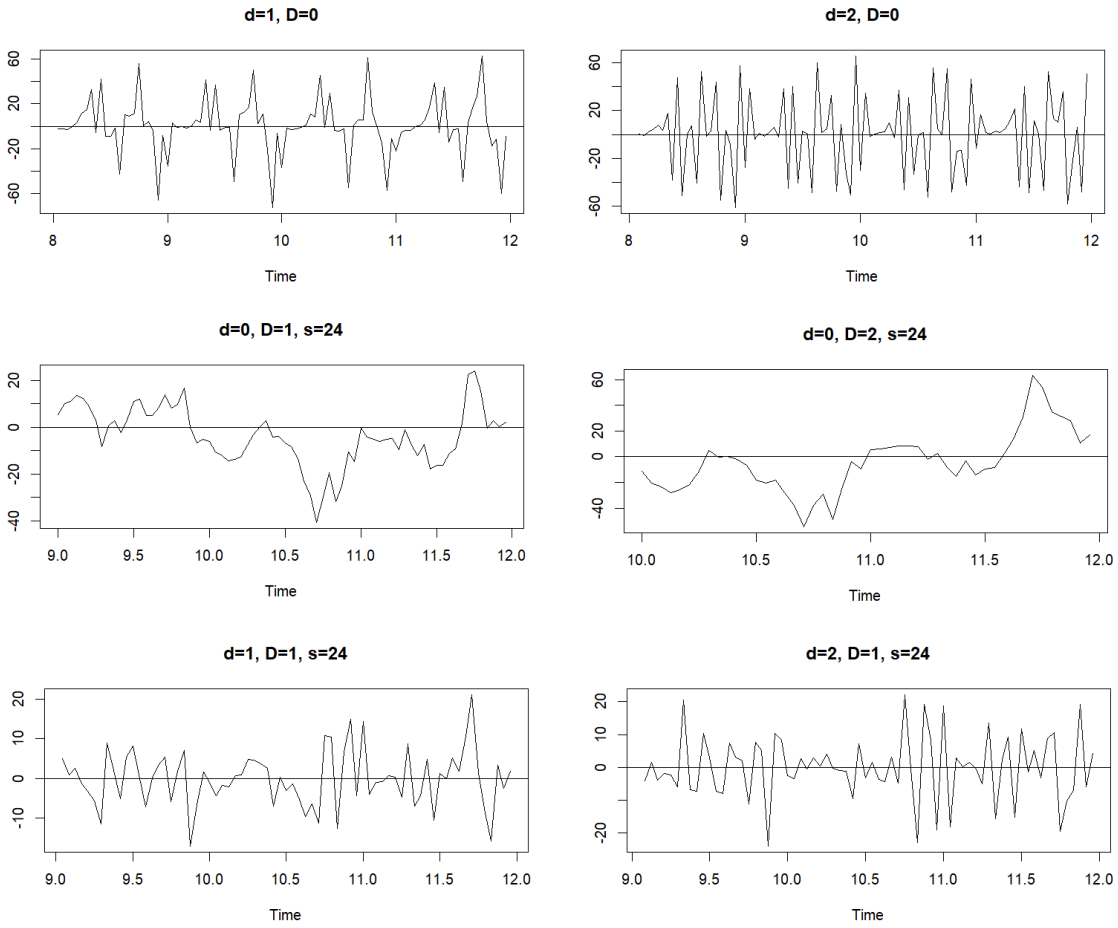
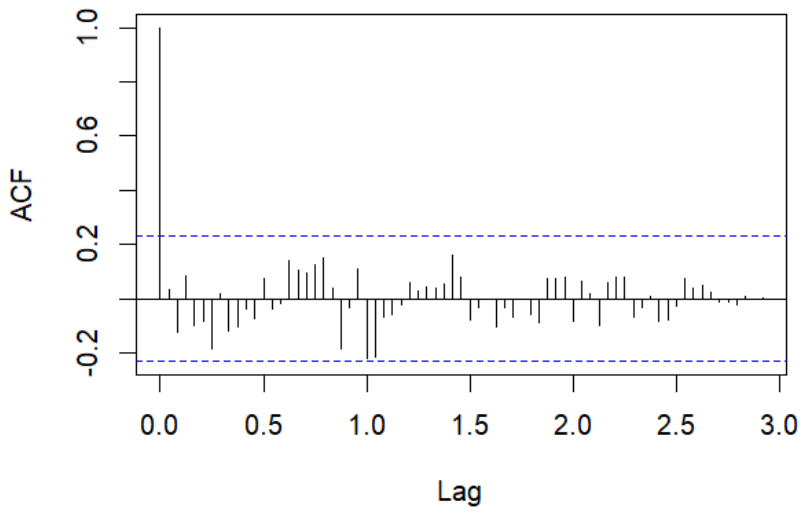


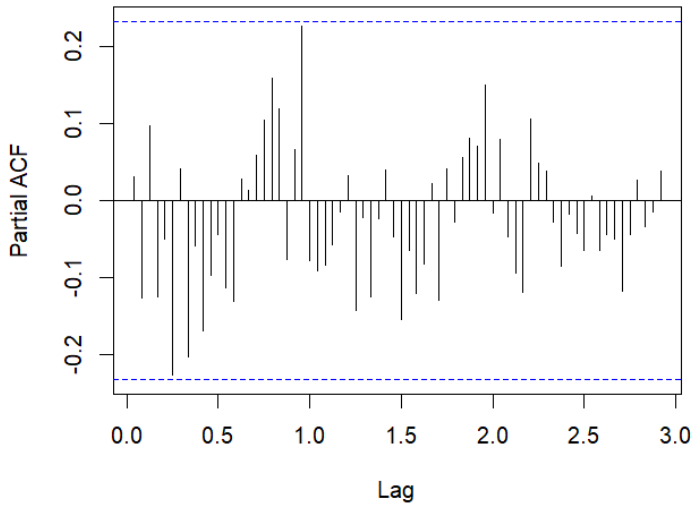
Figura 18. Representación de la serie de datos semanales, para distintas diferenciaciones.

A la vista de las gráficas y el valor de la varianza en cada caso, nos decantamos por los valores d = 1, D = 1 y s = 24, dado que la varianza en dicho caso es considerablemente menor, y la serie presenta un comportamiento que se acerca más a la estacionariedad. A continuación, analizamos las gráficas del ACF y PACF de la serie para determinar los valores de p, q, P, y Q.

```
> acf(diff(diff(precios_semana,differences=1),lag=24,differences=1),
    lag.max=96, main="")
```

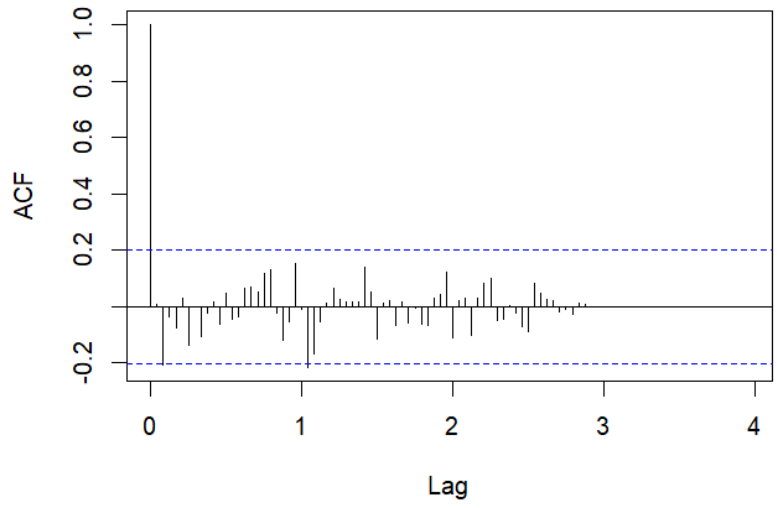


```
> pacf(diff(diff(precios_semana,differences=1),lag=24,differences=1),
   lag.max=96, main="")
```

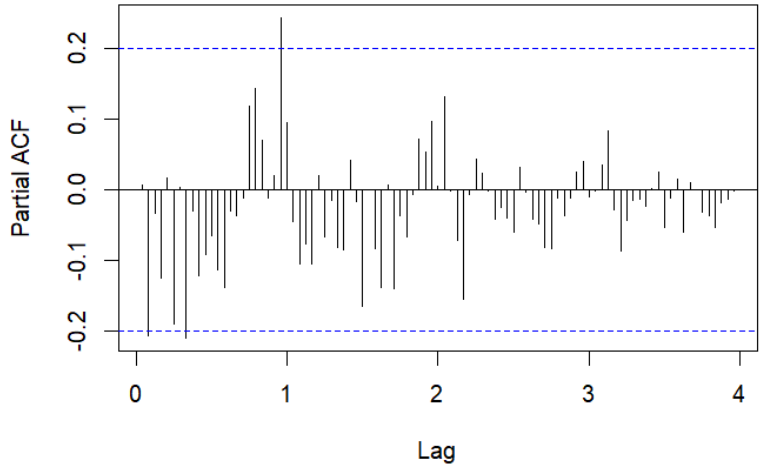


A la vista de las gráficas, el componente estacional parece corresponder a P = 0 y Q = 1. Entrenaremos un modelo ARIMA (0,1,0) x (0,1,1) [24] y estudiaremos sus residuos para determinar p y q.

```
> arima_semana_1 <- arima(precios_semana, order=c(0,1,0),
+                         seasonal = list(order=c(0,1,1), period=24))
> arima_semana_1

Call:
arima(x = precios_semana, order = c(0, 1, 0), seasonal = list(order =
c(0, 1,
    1), period = 24))

Coefficients:
         sma1
      -0.5667
s.e.   0.2492

sigma^2 estimated as 38.56:  log likelihood = -234.9,  aic = 473.81
> AIC(arima_semana_1, k = log(length(precios_semana))) # BIC
[1] 478.9385
> acf(arima_semana_1$residuals, lag.max=96, main="")
```



```
> pacf(arima_semana_1$residuals, lag.max=96, main="")
```



A la vista de las gráficas, que parecen mezclar características AR y MA, nos decantamos por un modelo con p = 3 y q = 2.

```
> arima_semana_2 <- arima (precios_semana, order=c(3,1,2),
+                          seasonal = list(order=c(0,1,1), period=24))
> arima_semana_2

Call:
arima(x = precios_semana, order = c(3, 1, 2), seasonal = list(order =
c(0, 1,
    1), period = 24))

Coefficients:
          ar1     ar2      ar3     ma1      ma2     sma1
      -0.0273  0.4632  -0.0392  0.0156  -0.8080  -0.7202
s.e.   0.1852  0.1827   0.1422  0.1511   0.1251   0.4445

sigma^2 estimated as 30.37:  log likelihood = -230.21,  aic = 474.43
> AIC(arima_semana_2, k = log(length(precios_semana))) # BIC
[1] 492.3791
```

El modelo obtenido tiene un AIC de 474.43, lo que supone un ligero empeoramiento. Probando con otros valores de p y q se llegó al modelo ARIMA (2,1,2) x (0,1,1) [24], que es el mismo que se obtuvo posteriormente con la herramienta autoarima. Este modelo cuenta con un AIC de 472.5, que sí que supone una ligera mejora con respecto al modelo con p = 0 y q = 0, aunque el BIC sea peor (486.08 frente al 478.94 original). Para el fin de realizar una comparación con el modelo generado con autoarima, tomaremos el modelo alcanzado en el primer intento de ajuste manual (p = 3 y q = 2).

```
> arima_semana_3 = auto.arima(precios_semana, d=1, D=1, max.order=5,
+                             trace=TRUE, approx=FALSE,
+                             allowdrift=FALSE, stepwise=FALSE)
> arima_semana_3
Series: precios_semana
ARIMA(2,1,2)(0,1,1)[24]

Coefficients:
          ar1     ar2      ma1      ma2     sma1
      -0.0176  0.4654  -0.0067  -0.8079  -0.6927
s.e.   0.1838  0.1842   0.1301   0.1276   0.3880

sigma^2 = 33.35:  log likelihood = -230.25
```

```
AIC=472.5   AICc=473.82   BIC=486.08
```

d) Analice el ACF, el PACF y la normalidad de los residuos para ambos modelos. Comente si hay diferencias significativas.

```
> acf(arima_semana_2$residuals, lag.max=96, main="Residuos (Modelo
   Manual)")
> pacf(arima_semana_2$residuals, lag.max=96, main="Residuos (Modelo
   Manual)")
> acf(arima_semana_3$residuals, lag.max=96, main="Residuos (Modelo
   Autoarima)")
> pacf(arima_semana_3$residuals, lag.max=96, main="Residuos (Modelo
   Autoarima)")
```

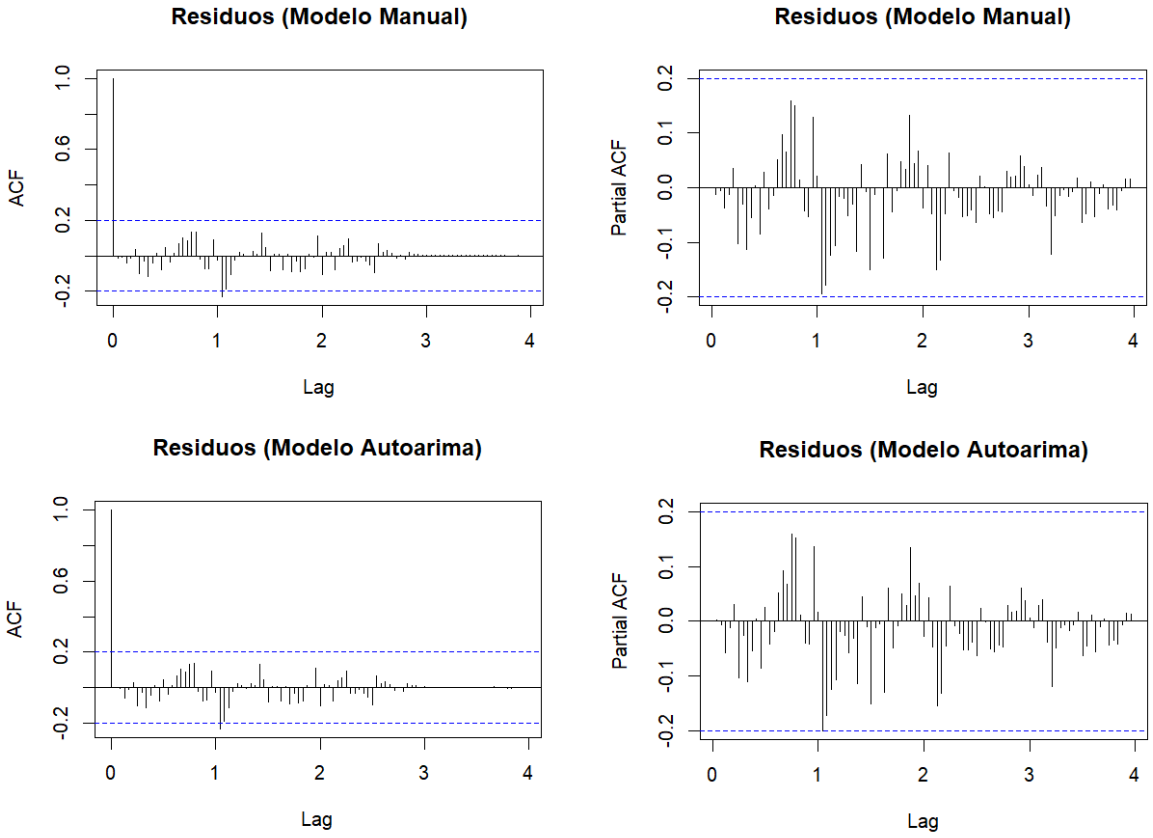


Figura 19. ACF y PACF de los residuos de los modelos obtenidos mediante ajuste manual y autoarima.

Las gráficas del ACF y PACF de los residuos, correspondientes a ambos modelos, son similares entre sí, y presentan un buen comportamiento. A continuación, analizamos la normalidad de dichos residuos.

```
> plot(arima_semana_2$residuals, ylab = "Residuals", main="Ajuste
   Manual")
> abline(a=0, b=0)
> hist(arima_semana_2$residuals, xlab="Residuals", xlim=c(-80,80),
   main="Ajuste Manual")
> qqnorm(arima_semana_2$residuals, main="Ajuste Manual")
> qqline(arima_semana_2$residuals)
> # Modelo autoarima
> plot(arima_semana_3$residuals, ylab = "Residuals", main="Autoarima")
> abline(a=0, b=0)
> hist(arima_semana_3$residuals, xlab="Residuals", xlim=c(-80,80),
   main="Autoarima")
> qqnorm(arima_semana_3$residuals, main="Autoarima")
```

```
> qqline(arima_semana_3$residuals)
```

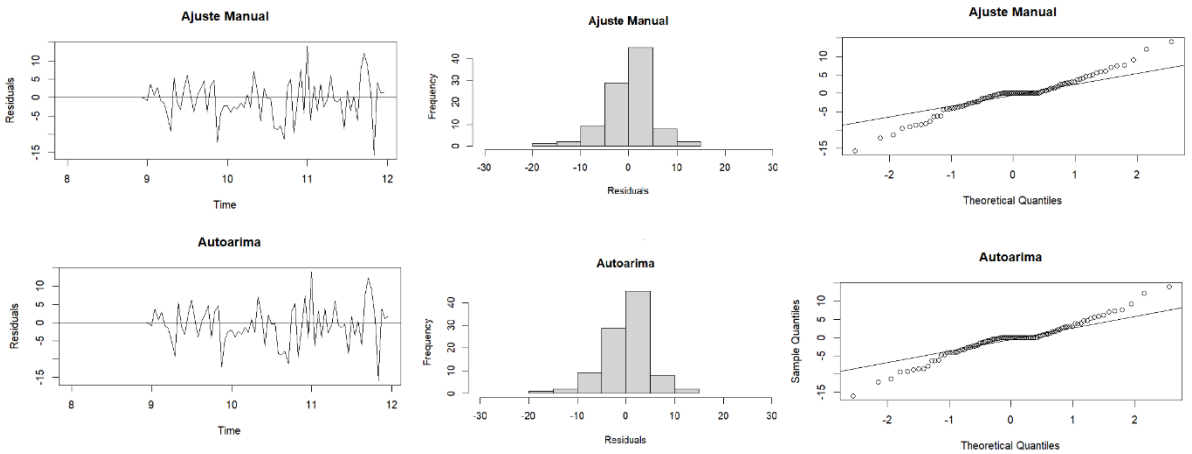


Figura 20. Estudio de la normalidad de los residuos de los modelos ajustados de forma manual, y con autoarima, para el dataset de datos semanales.

En el caso de la normalidad de los residuos, tampoco hay diferencias significativas entre ambos modelos. Se puede observar que la distribución de los residuos se aleja de lo normal, lo que indica que los datos originales no se corresponden con una serie temporal estacionaria, y que las predicciones de estos modelos podrían no ser buenas.

e) Utilice ambos modelos para predecir un día de datos y compárelos con los datos guardados para test. ¿Qué modelo es más preciso? ¿Cuál tiene más incertidumbre en las predicciones?

```
> arima_semana_2.predict <- predict(arima_semana_2, n.ahead=24)
> lines(arima_semana_2.predict$pred, col=2)
> plot(precios_semana, xlab = "Tiempo (días)",
+      ylab = "Precios Excedente Eléctrico (€)",
+      xlim = c(8, 13),
+      main="Predicción del Modelo Ajustado Manualmente")
> lines(arima_semana_2.predict$pred, col=2)
> lines(arima_semana_2.predict$pred+1.96*arima_semana_2.predict$se,
   col=3, lty=2)
> lines(arima_semana_2.predict$pred-1.96*arima_semana_2.predict$se,
   col=3, lty=2)
> lines(precios_semana_test, col=4)
```

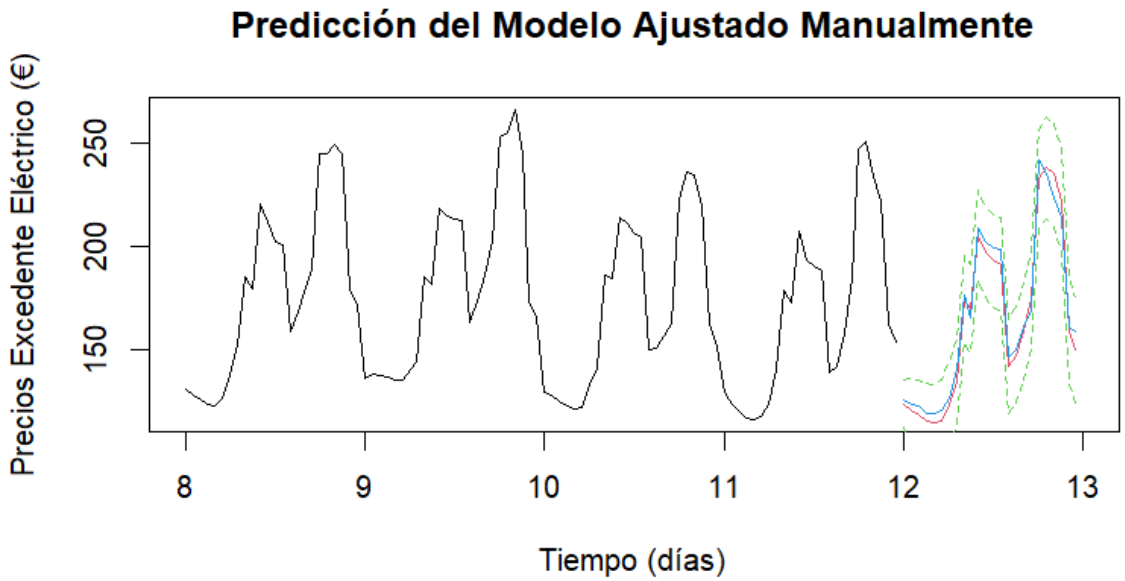


Figura 21. Predicción del modelo ajustado manualmente para el dataset semanal.

```
> arima_semana_3.predict <- predict(arima_semana_3, n.ahead=24)
> plot(precios_semana, xlab = "Tiempo (días)",
+      ylab = "Precios Excedente Eléctrico (€)",
+      #ylim = c(1, 50),
+      xlim = c(8, 13),
+      main="Predicción del Modelo Ajustado con Autoarima")
> lines(arima_semana_3.predict$pred, col=2)
> lines(arima_semana_3.predict$pred+1.96*arima_semana_3.predict$se,
   col=3, lty=2)
> lines(arima_semana_3.predict$pred-1.96*arima_semana_3.predict$se,
   col=3, lty=2)
> lines(precios_semana_test, col=4)
```

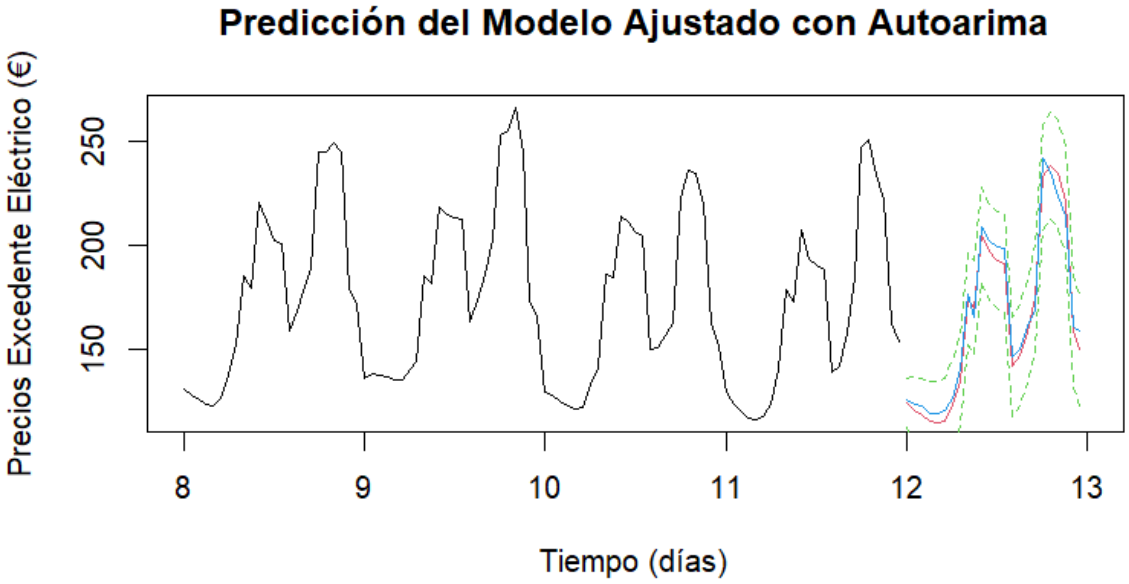


Figura 22. Predicción del modelo ajustado con autoarima para los datos semanales.

Las respuestas de ambos modelos son muy similares. En ambos casos el intervalo de confianza es bastante estrecho y la predicción se ajusta considerablemente bien a los valores reales. Tanto la predicción como los valores reales se encuentran dentro de los intervalos de confianza.

Datos Mensuales

a) Cargue el archivo de datos *'precios_excedente_peninsula_mes.xls'* y guárdelo en un dataframe.

```
> precios_mes_input <-
   as.data.frame(read_excel("Ficheros/precios_excedente_peninsula_mes.x
   lsx"))
> head(precios_mes_input)
             datetime  value
1 2024-01-01 00:00:00 123.83
2 2024-01-01 01:00:00 111.11
3 2024-01-01 01:59:59 110.77
4 2024-01-01 02:59:59 109.28
5 2024-01-01 03:59:59 110.16
6 2024-01-01 04:59:59 111.47
```

b) ¿Cuántos días de datos hay disponibles? ¿Con qué resolución? Guarde cuatro semanas de datos para test, y el resto como datos de entrenamiento.

Hay 31 días de datos en el dataset, con una resolución de una hora, es decir, 24 datos por día. Para el entrenamiento guardaremos 28 días, y para testing sólo 3.

```
> precios_mes_input_train <- precios_mes_input[1:672,] # 4 semanas para
   entrenamiento -> 4*7*24=672
> precios_mes_input_test <- precios_mes_input[673:744,] # Los 3 días
   restantes -> 3*24=72
> precios_mes <- ts(as.numeric(precios_mes_input_train[,2]), start=1,
   frequency = 24)
> precios_mes_test <-
   ts(as.numeric(precios_mes_input_test[,2]),start=29, frequency = 24)
> precios_mes_por_semanas <-
   ts(as.numeric(precios_mes_input_train[,2]), start=1, frequency =
   168)
> precios_mes_test_por_semanas <-
   ts(as.numeric(precios_mes_input_test[,2]),start=5, frequency = 168)
```

Para realizar comparaciones posteriormente, hemos generado dos series temporales, con distintas frecuencias de datos (24 y 168), que se corresponden al patrón diario y semanal de estacionalidad, respectivamente. Lo hacemos así porque la función autoarima extrae de la propia serie el valor del periodo estacional.

c) Tras una primera diferenciación sin estacionalidad, compare la serie resultante de aplicar una segunda diferenciación con estacionalidad si se considera un patrón diario o un patrón semanal. ¿Cuál es más adecuado? Comente qué ocurre con los precios el fin de semana, y por qué esto hace más conveniente un patrón u otro.

```
> plot(precios_mes, xlab = "Tiempo (días)",
+      ylab = "Precios Excedente Eléctrico (€)")
```

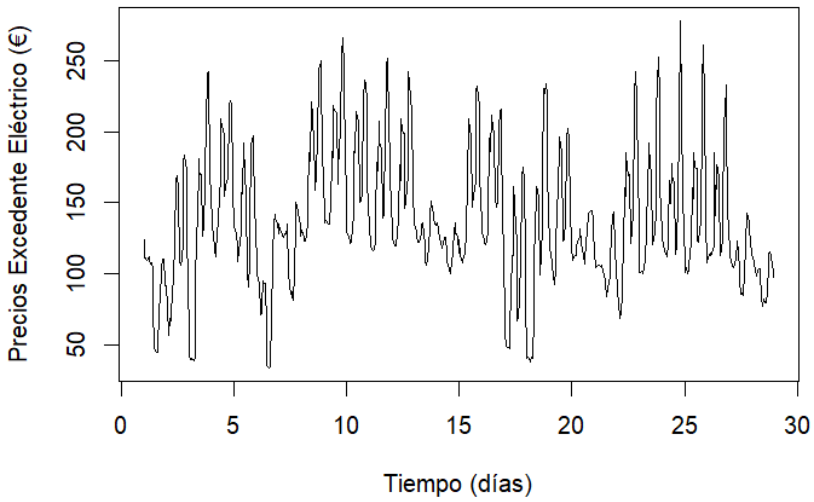


Figura 23. Representación de la serie de datos mensuales, sin diferenciar.

```
> plot(diff(precios_mes,differences=1), main="d=1, D=0", ylab="")
> abline(a=0, b=0)
> var(diff(precios_mes,differences=1))
[1] 504.7393
> plot(diff(diff(precios_mes,differences=1),lag=24,differences=1),
   main="d=1, D=1, s=24", ylab="")
> abline(a=0, b=0)
> var(diff(diff(precios_mes,differences=1),lag=24,differences=1))
[1] 259.1211
> plot(diff(diff(precios_mes,differences=1),lag=168,differences=1),
   main="d=1, D=1, s=128", ylab="")
> abline(a=0, b=0)
> var(diff(diff(precios_mes,differences=1),lag=168,differences=1))
[1] 138.8557
```

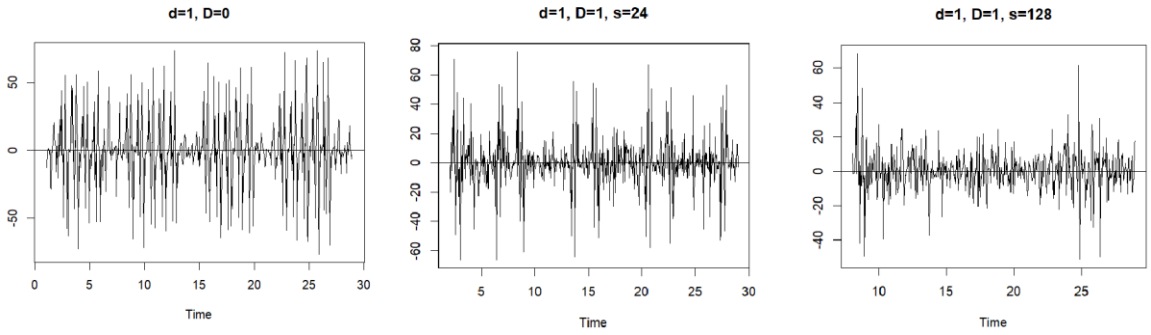


Figura 24. Representación de la serie de datos mensuales, tras aplicar diferentes diferenciaciones estacionales y no estacionales. De izquierda a derecha: tras una derivación no estacional, tras una derivación estacional y una no estacional con patrón diario, y tras una derivación no estacional y una estacional con patrón semanal.

A la vista de las gráficas de las tres diferenciaciones, y considerando el valor de la varianza para cada caso, nos decantaremos por el patrón semanal de estacionalidad, dado que la varianza es la menor y la respuesta presenta un comportamiento más estacionario. Analizando la Figura 23 se puede apreciar cómo los precios se reducen considerablemente cada 5 días (los fines de semana). También por este motivo, es más apropiado considerar el patrón de estacionalidad semanal, que incluye dentro de su periodo tanto los picos de entre semana, como los valles de los fines de semana. Un modelo con periodo diario de estacionalidad tendrá más dificultad para predecir los cambios bruscos correspondientes al paso de viernes a sábado, y de domingo a lunes.

d) Realice un ajuste con autoarima con aproximación, y utilice el modelo resultante para predecir 3 días de datos. Comente los resultados.

Para fines comparativos, se han ajustado dos modelos: uno con patrón diario de estacionalidad, y otro con patrón semanal.

```
> arima_mes_1 = auto.arima(precios_mes, d=1, D=1, max.order=4,
+                              trace=TRUE, approx=TRUE,
+                              allowdrift=FALSE, stepwise=FALSE)
Best model: ARIMA(0,1,2)(2,1,0)[24]
> arima_mes_1
Series: precios_mes
ARIMA(0,1,2)(2,1,0)[24]

Coefficients:
         ma1     ma2     sar1     sar2
      0.0404  0.3111  -0.2383  -0.3178
s.e.  0.0362  0.0467   0.0407   0.0418

sigma^2 = 217.1:  log likelihood = -2659.66
AIC=5329.32   AICc=5329.41   BIC=5351.68
> arima_mes_2 = auto.arima(precios_mes_por_semanas, d=1, D=1,
   max.order=4,
+                           trace=TRUE, approx=TRUE,
+                           allowdrift=FALSE, stepwise=FALSE)

Best model: ARIMA(1,1,2)(0,1,0)[168]
> arima_mes_2
Series: precios_mes_por_semanas
ARIMA(1,1,2)(0,1,0)[168]

Coefficients:
         ar1      ma1      ma2
      0.9081  -0.8256  -0.1446
s.e.  0.0457   0.0648   0.0506

sigma^2 = 135.1:  log likelihood = -1950.1
AIC=3908.19   AICc=3908.27   BIC=3925.08
```

Se han obtenido los siguientes modelos: ARIMA (0,1,2) x (2,1,0) [24], y ARIMA (1,1,2) x (0,1,0) [168]. A continuación, se predecirán con cada modelo los últimos tres días del dataset de datos mensuales.

Para el modelo con periodo estacional diario:

```
> arima_mes_1.predict <- predict(arima_mes_1, n.ahead=72)
> plot(precios_mes, xlab = "Tiempo (días)",
+      ylab = "Precios Excedente Eléctrico (€)",
+      ylim = c(0, 370),
+      xlim = c(1, 31),
+      main = "Predicción del Modelo con Estacionalidad Diaria")
> lines(arima_mes_1.predict$pred, col=2)
> lines(arima_mes_1.predict$pred+1.96*arima_mes_1.predict$se, col=3,
   lty=2)
> lines(arima_mes_1.predict$pred-1.96*arima_mes_1.predict$se, col=3,
   lty=2)
> lines(precios_mes_test, col=4)
```

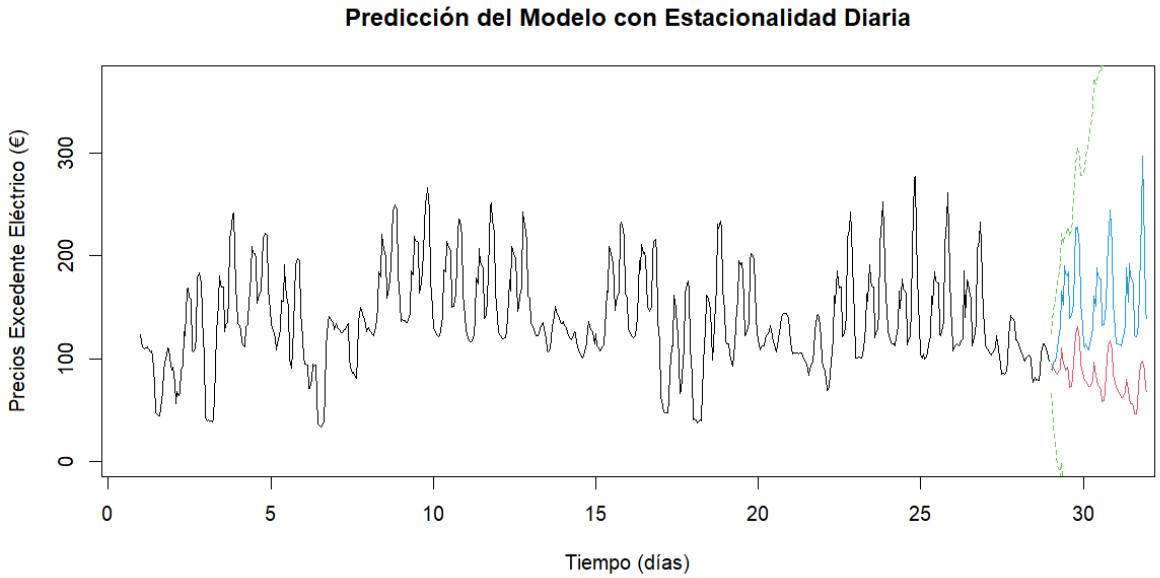


Figura 25. Predicción del modelo ajustado con estacionalidad diaria, para el dataset de datos mensuales.

Para el modelo con periodo estacional semanal:

```
> arima_mes_2.predict <- predict(arima_mes_2, n.ahead=72)
> plot(precios_mes_por_semanas, xlab = "Tiempo (días)",
+      ylab = "Precios Excedente Eléctrico (€)",
+      ylim = c(20, 340),
+      xlim = c(1, 5.4),
+      main = "Predicción del Modelo con Estacionalidad Semanal")
> lines(arima_mes_2.predict$pred, col=2)
> lines(arima_mes_2.predict$pred+1.96*arima_mes_2.predict$se, col=3,
   lty=2)
> lines(arima_mes_2.predict$pred-1.96*arima_mes_2.predict$se, col=3,
   lty=2)
> lines(precios_mes_test_por_semanas, col=4)
```

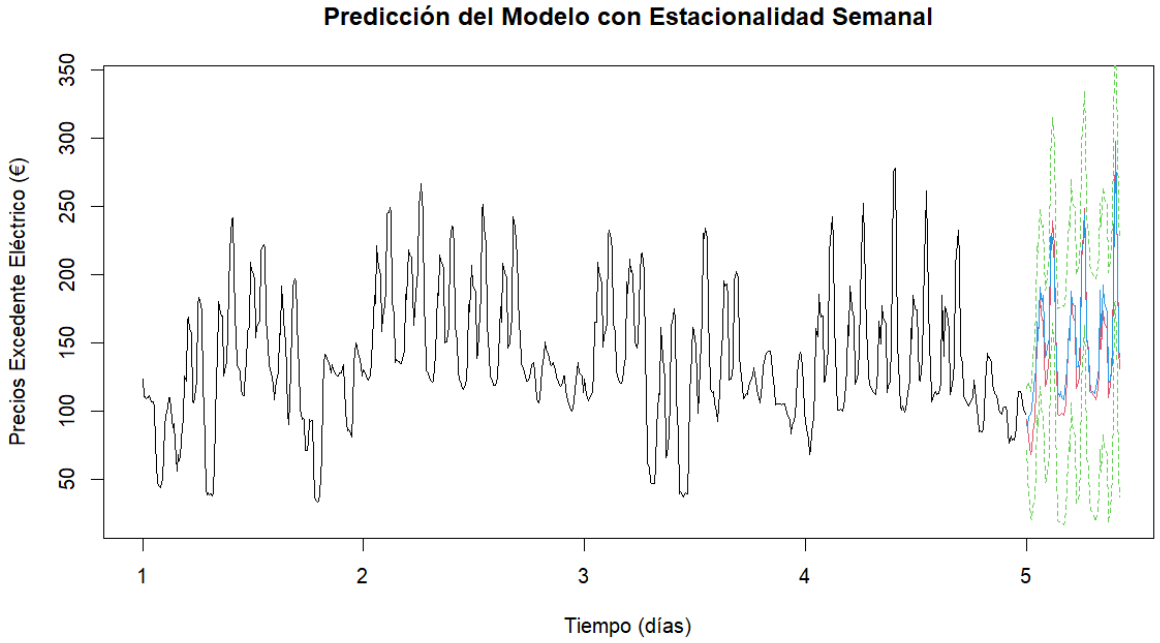


Figura 26. Predicción realizada por el modelo con estacionalidad semanal, para el dataset de datos mensuales.

Con autoarima se generó un modelo ARIMA (1,1,2) x (0,1,0) [168], con un AIC y un BIC de 3908.19 y 3925.08, respectivamente. En la Figura 26 se observa que el modelo es capaz de predecir las ventas de la semana entrante con considerable precisión, siendo capaz de anticiparse a los altos valores propios del inicio de la semana, viniendo de valores más bajos en el fin de semana. Además, los intervalos de confianza son relativamente estrechos, y tanto la predicción como la respuesta real están contenidos dentro de ellos. Por otro lado, el modelo generado para un patrón estacional diario ha sido incapaz de predecir la subida de los precios del inicio de semana, y mantiene la baja tendencia propia del fin de semana (ver Figura 25). Tanto el AIC como el BIC de este modelo son significativamente mayores, y los intervalos de confianza de la predicción son desproporcionados. Aunque la respuesta real y la predicción se mantienen dentro de los intervalos, la discrepancia entre ambas es cualitativamente importante.