Universidad Internacional Menéndez Pelayo

Máster Universitario en Investigación en Inteligencia Artificial

TRABAJO SERIES TEMPORALES

Datos temporales y complejos

Laura Rodríguez Navas

Diciembre de 2020

rodrigueznavas@posgrado.uimp.es

Índice general

Capítulo 1

Análisis de la serie temporal

1.1. Introducción

Una empresa tecnológica cuya área de negocio es la inteligencia artificial es contratada por una empresa eléctrica para que diseñe un sistema de recomendación que haga ofertas personalizadas a sus clientes sobre los paquetes energéticos más adecuados a sus consumos. Para ello, la empresa primero debe llevar a cabo un análisis exhaustivo de los consumos energéticos y diseñar una técnica de predicción que sea capaz de predecir dichos consumos con un horizonte temporal dado. La empresa eléctrica suministra a la empresa tecnológica para dicho cometido los consumos eléctricos desde el 01 de enero de 2015 hasta el 31 de diciembre de 2015 medidos con una frecuencia temporal de 10 minutos.

Se pide:

1. Analizar la serie temporal de consumos eléctricos y describir brevemente las principales características de esta.
2. Realizar un estudio para determinar si la serie temporal presenta *outliers*. Describir brevemente el estudio realizado y las conclusiones alcanzadas.
3. Seleccionar un artículo publicado en una revista científica de prestigio internacional en el que se presente un método de predicción. Estudiar de manera detallada el método publicado y describirlo de forma resumida.
4. Aplicar el método seleccionado en el apartado anterior para obtener una predicción del consumo eléctrico con un horizonte temporal de 4 horas para los días de la semana desde el lunes 8 de junio hasta el domingo 14 de junio. Para ello, se debe implementar un método de predicción en el lenguaje de programación que se estime oportuno o bien usar software libre disponible como WEKA, R, KEEL, ... Una vez obtenidas las predicciones del periodo indicado, visualizar los resultados (predicciones, errores, ...).

El trabajo de investigación se realiza utilizando el lenguaje de programación R y todos los datos y el código necesarios para reproducir el estudio se proporcionan dentro de este documento para garantizar que éste sea completamente reproducible.

Se utilizan los siguientes paquetes para el análisis:

```
library(tidyverse)
library(lubridate)
library(xts)
library(forecast)
```

- tidyverse es un conjunto de librerías de R diseñadas para la *"Ciencia de datos"*.
- lubridate es una librería que permite manipular fechas e intervalos de tiempo.
- xts es una librería que permite convertir datos en series temporales.
- forecast es una librería que contiene métodos y herramientas para analizar series temporales

1.2. Datos de la serie temporal

Los datos están disponibles para descargar aquí. Una vez descargados, se cargan en el espacio de trabajo.

```
data <- read.csv("Demanda_2015.csv", header = FALSE, sep = ",")
```

Buscamos valores perdidos.

```
sum(is.na(data))
```

```
## [1] 0
```

Vemos que no existen valores perdidos, por lo tanto no tendremos que eliminarlos.

```
# Establecemos nombres a las columnas
colnames(data) <- c("date", "time", "demand")
head(data, 10)
```

```
##          date time demand
## 1  01/01/2015 0:00  25459
## 2  01/01/2015 0:10  25591
## 3  01/01/2015 0:20  25531
## 4  01/01/2015 0:30  25453
## 5  01/01/2015 0:40  25329
## 6  01/01/2015 0:50  25247
## 7  01/01/2015 1:00  25093
## 8  01/01/2015 1:10  24853
## 9  01/01/2015 1:20  24678
## 10 01/01/2015 1:30  24391
```

```
summary(data)
```

```
##      date               time               demand
##  Length:52560       Length:52560       Min.   :17985
##  Class :character   Class :character   1st Qu.:24392
##  Mode  :character   Mode  :character   Median :28566
##                                        Mean   :28349
##                                        3rd Qu.:31664
##                                        Max.   :40648
```

Tenemos la variable *date* como *character*, la variable *time* como *character* y la variable *demand* como *integer*. Para su análisis posterior, el marco de datos de fechas y horas se formatea. Para ello, primero crearemos una nueva columna *datetime*, donde juntaremos las fechas con las horas usando la función *cbind()*. Luego procederemos a la conversión de tipos *character* a *POSIXct* para la fecha-hora usando la función *parse_date_time()*.

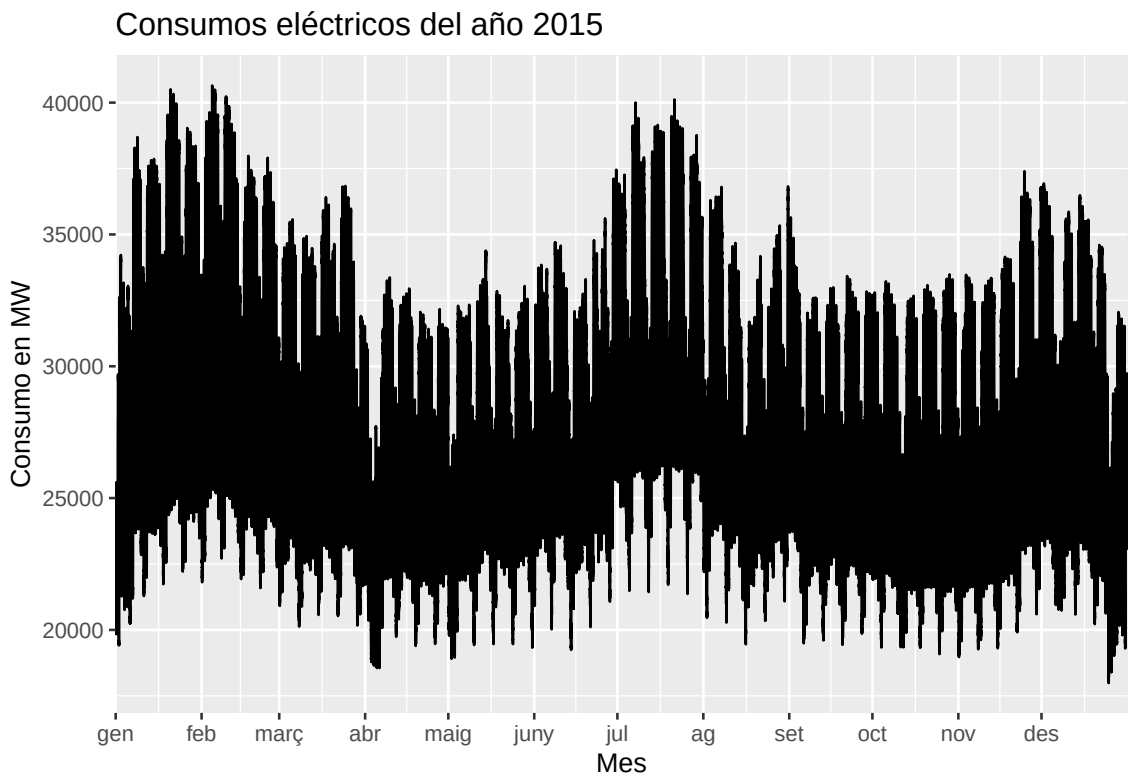
```
data <- cbind(datetime = paste(data$date, data$time), data)
data$datetime <- parse_date_time(data$datetime, "dmY HMS",
                                 truncated = 3, tz = "UTC")
head(data, 10)
```

```
##                datetime       date time demand
## 1  2015-01-01 00:00:00 01/01/2015 0:00  25459
## 2  2015-01-01 00:10:00 01/01/2015 0:10  25591
## 3  2015-01-01 00:20:00 01/01/2015 0:20  25531
## 4  2015-01-01 00:30:00 01/01/2015 0:30  25453
## 5  2015-01-01 00:40:00 01/01/2015 0:40  25329
## 6  2015-01-01 00:50:00 01/01/2015 0:50  25247
## 7  2015-01-01 01:00:00 01/01/2015 1:00  25093
## 8  2015-01-01 01:10:00 01/01/2015 1:10  24853
## 9  2015-01-01 01:20:00 01/01/2015 1:20  24678
## 10 2015-01-01 01:30:00 01/01/2015 1:30  24391
```

1.3. Visualización de la serie temporal

La siguiente figura muestra la distribución del consumo eléctrico a lo largo del 2015.

```
ggplot(data = data, aes(x = datetime, y = demand)) +
  geom_line() +
  scale_x_datetime(date_labels = "%b",
                   breaks = "1 month",
                   expand = c(0, 0)) +
  ggtitle("Consumos eléctricos del año 2015") +
  xlab("Mes") +
  ylab("Consumo en MW")
```

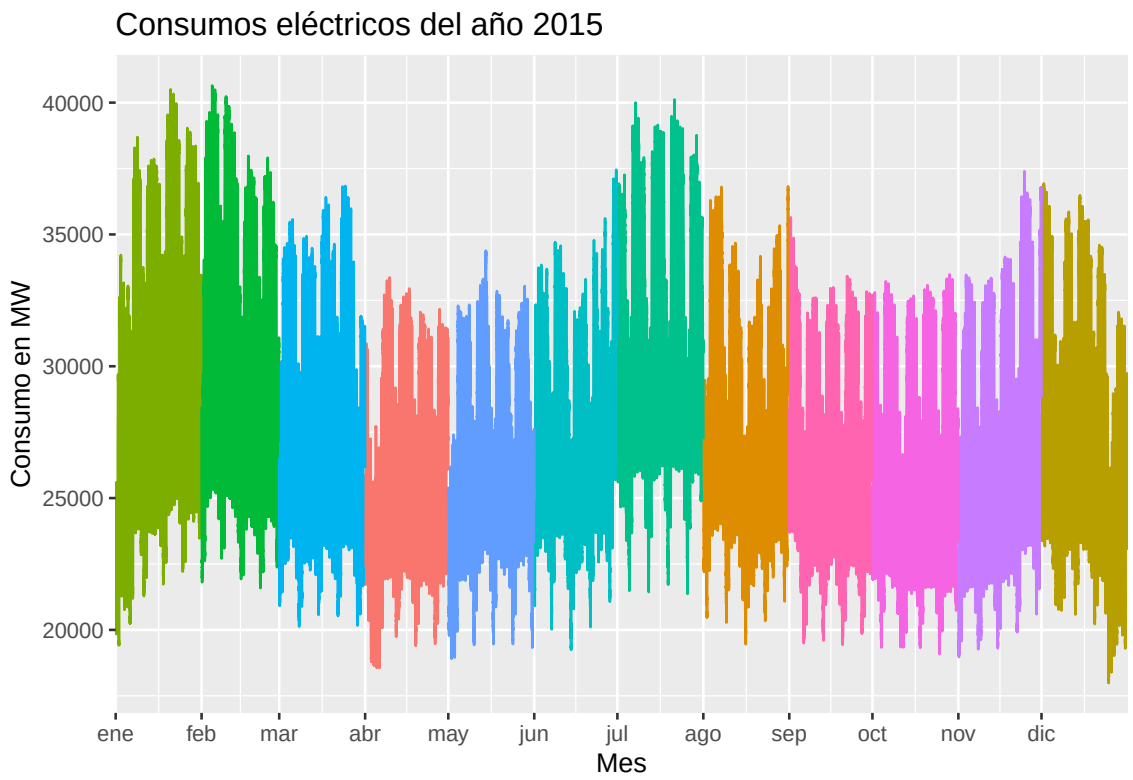


Observando la figura se puede detectar una dependencia estacional del consumo eléctrico, aunque pueden existir otros factores que pueden afectarla, como los días festivos, los fines de semana, ... Parece que el principal desafío introducido por la serie temporal podría ser su estacionalidad múltiple. En ese caso, las siguientes estacionalidades pueden estar presentes:

- Las personas usan la mayoría de sus electrodomésticos en determinadas horas del día, lo que resulta en una estacionalidad diaria.
- Las personas usan la electricidad de manera diferente los días entre semana y los fines de semana, lo que resulta en una estacionalidad semanal.
- Las personas usan la electricidad de manera diferente en diferentes épocas del año, lo que resulta en una estacionalidad anual.

Para mejorar la visualización de la distribución del consumo eléctrico a lo largo del 2015, la siguiente figura es muy útil.

```
ggplot(data, aes(x = datetime, y = demand)) +
  geom_line(aes(colour = month)) +
  scale_x_datetime(date_labels = "%b",
                   breaks = "1 month",
                   expand = c(0, 0)) +
  ggtitle("Consumos eléctricos del año 2015") +
  xlab("Mes") +
  ylab("Consumo en MW") +
  theme(legend.position = "none")
```



Es importante tener en cuenta cómo aumenta el consumo de la electricidad tanto en los meses de verano, de mayo a septiembre, como en los días de invierno, desde noviembre hasta febrero. Esto se debe a que las personas usan la electricidad para enfriar y calentar sus viviendas. Durante los períodos de invierno y de verano, el consumo eléctrico es de manera clara mayor salvo, probablemente, en los períodos vacacionales. Como se puede observar en el mes de agosto. En resumen, visualmente podemos deducir que los consumos eléctricos siguen un patrón de subidas y bajadas en una tendencia que varía de manera estacional.

1.4. Componentes de la serie temporal

Es frecuente analizar las series temporales des del punto de vista de sus componentes estructurales. Pero primero necesitamos que R trate los datos como una serie temporal, así que para ello tendremos que determinar apropiadamente sus características con la función *ts()*. Para definir la serie correctamente escribimos:

```
ts <- ts(data$demand, frequency = 24 * 60 / 10)
# El argumento frequency de la función *ts()* se utiliza para indicar
# la periodicidad de la serie (en este caso es de 10 minutos)
```

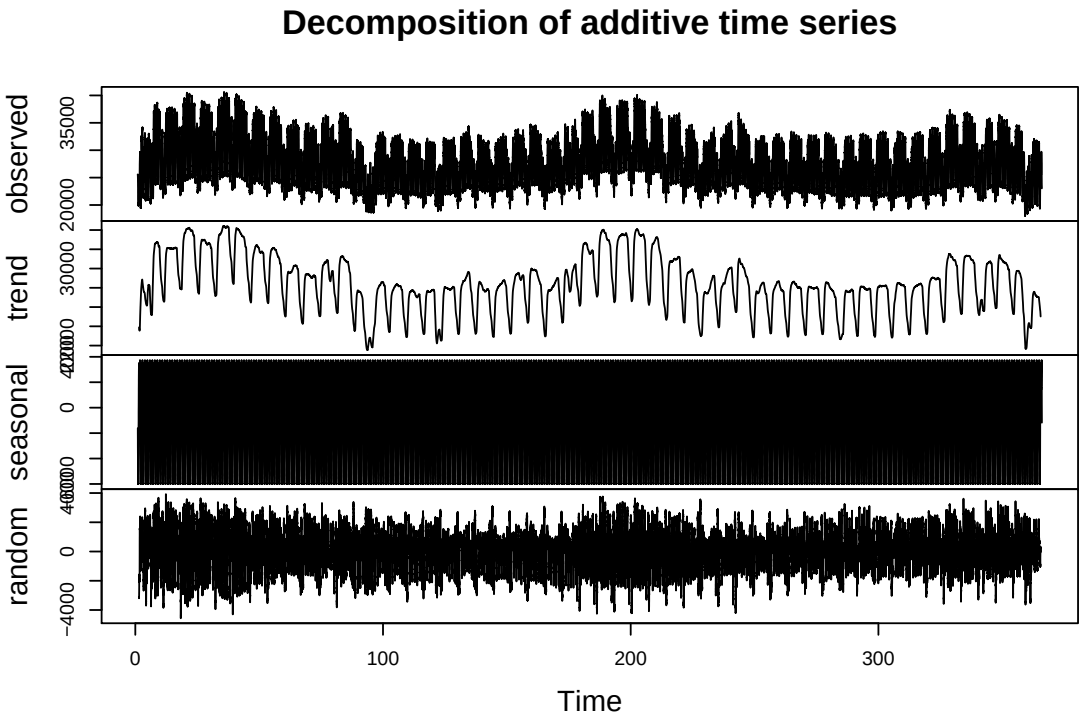
Los componentes de una serie temporal son:

- Tendencia: un aumento o disminución a largo plazo de los datos se denomina tendencia. No tiene por qué ser necesariamente lineal. Es el patrón subyacente en los datos a lo largo del tiempo.
- Estacional o Periódico: cuando una serie está influenciada por factores estacionales. Como en este caso y hemos podido visualizar en el apartado anterior.

- Cíclico: cuando los datos muestran subidas y bajadas que no son de un mismo período.

En R se pueden averiguar los componentes de la serie temporal con la siguiente función:

```
componentes.ts <- decompose(ts)
plot(componentes.ts)
```



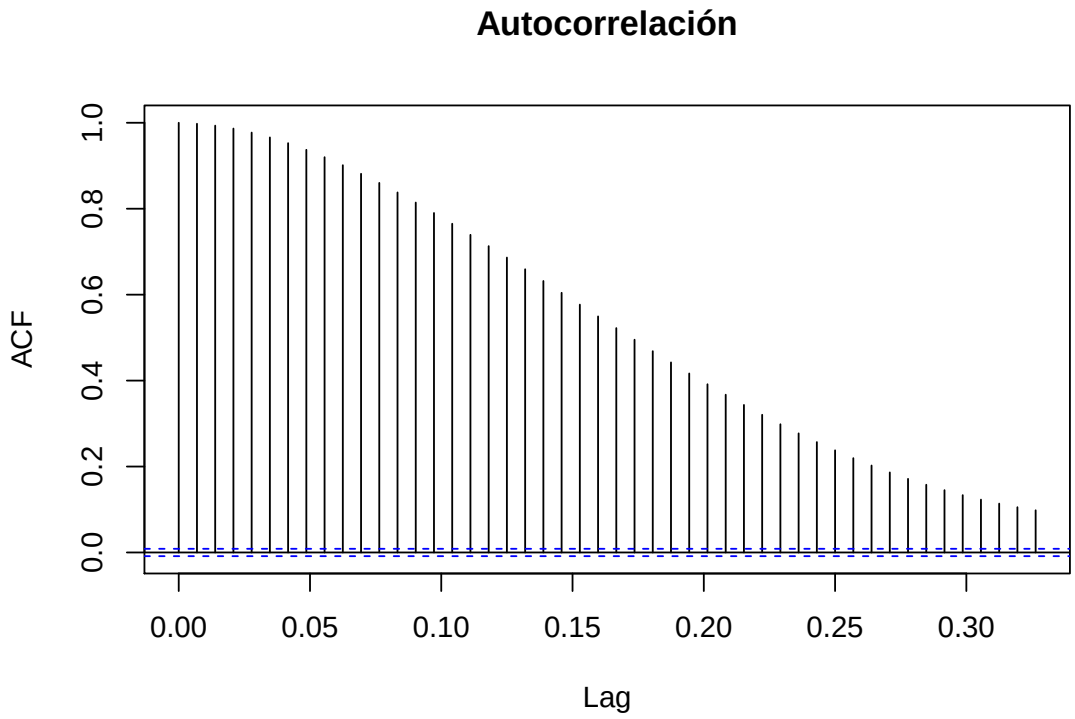
En la figura se observant cuatro sub-figuras:

- observed: los datos actuales.
- trend: el movimiento general hacia arriba o hacia abajo de los datos.
- seasonal: cualquier patrón mensual o anual de los os datos.
- random: parte inexplicable de los datos.

A partir de las sub-figuras se puede determinar que la serie temporal no presenta una tendencia clara. Por ese motivo podría ser estacionaria. También se puede observar que la serie podría ser estacionaria en cuando a la varianza, ya que no se aprecia gran variabilidad. Pero presenta una estacionalidad muy marcada (el consumo aumenta durante los meses de invierno y verano). Y por ese motivo la serie no es estacionaria. Según que modelo sea seleccionado y mejorar las futuras predicciones, se debe eliminar esa estacionalidad de la serie. Las series temporales no estacionarias son más difíciles de predecir.

Pero antes, para confirmar que la serie no es estacionaria trazaremos la gráfica de su función de autocorrelación, usando la función *acf()*. La función de autocorrelación *acf()* proporciona la autocorrelación en todos los retrasos posibles. La autocorrelación en el retraso 0 se incluye por defecto, lo que siempre toma el valor 1, ya que representa la correlación entre los datos y ellos mismos.

```r
acf(ts, main = "Autocorrelación")
```



Como podemos observar en la figura anterior, la autocorrelación disminuye a medida que aumenta el retraso, lo que confirma que no existe una asociación lineal entre las observaciones separadas por retrasos mayores. Es decir, la serie no es estacionaria, ya que el valor de la función de autocorrelación decae de manera exponencial a medida que aumentan los rezagos en el tiempo.

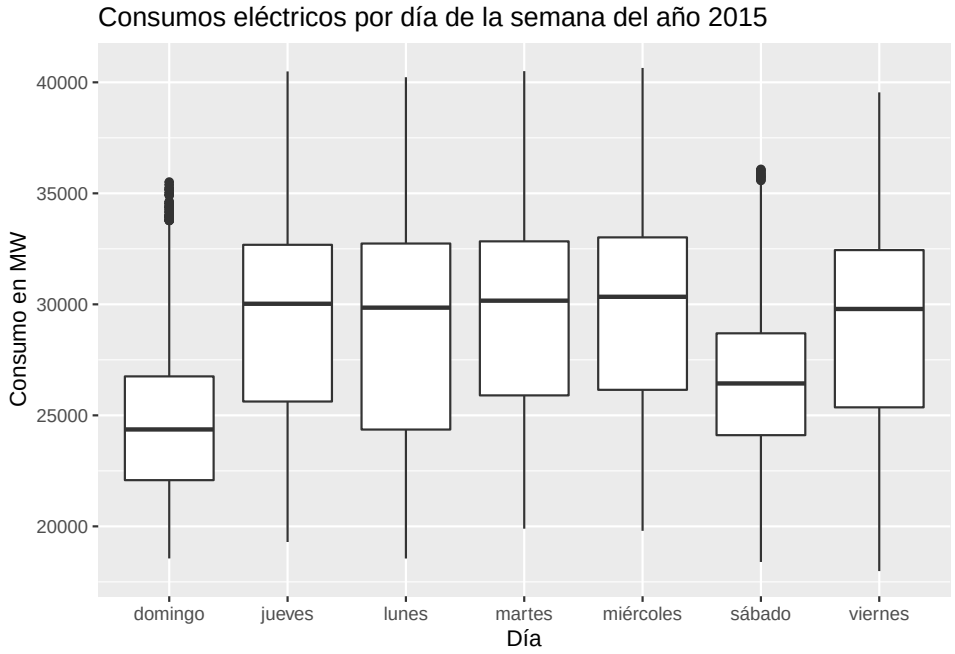
Capítulo 2

Outliers

2.1. Identificación de outliers

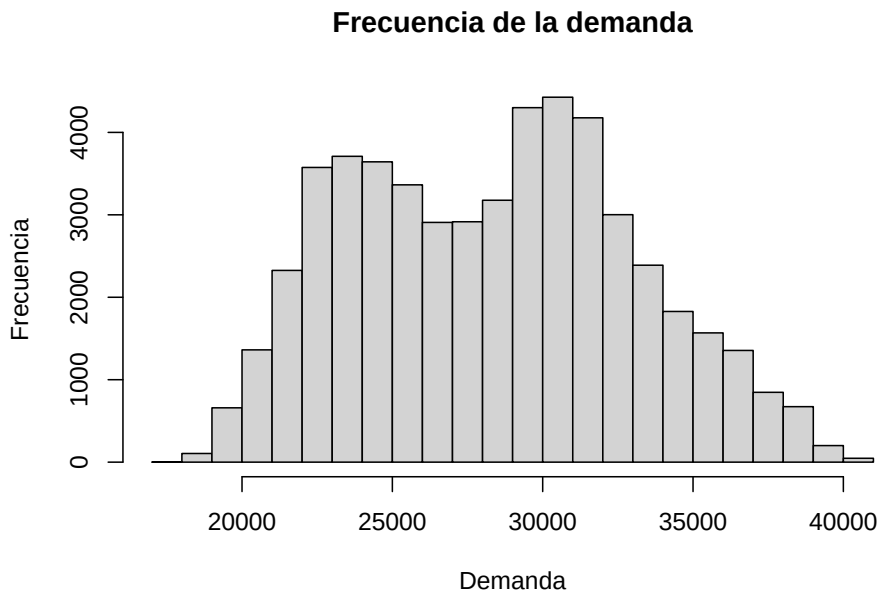
En el capítulo anterior, durante un breve análisis, se ha comentado que podrían existir factores que afecten a la serie temporal. Uno de esos factores podría ser que, durante los fines de semana, el consumo eléctrico disminuya considerablemente en comparación con el resto de los días de la semana, y eso provocaría la aparición de valores atípicos en la serie. Otro factor que puede hacer variar el consumo eléctrico considerablemente y provocar la aparición de valores atípicos en la serie, son los días festivos. Así, inicialmente se consideran estos dos factores para identificar valores atípicos en la serie, si es que existen. Primero vemos si el consumo eléctrico durante los fines de semana es más bajo que entre semana. Pero antes se añade a los datos una columna nueva *day*, para indicar los días de la semana de la serie.

```
data$day <- format(data$datetime, format = "%A")
ggplot(data, aes(day, demand)) +
  geom_boxplot() +
  ggtitle("Consumos eléctricos por día de la semana del año 2015") +
  xlab("Día") + ylab("Consumo en MW")
```



Como se observa en la figura durante los fines de semana (sábado y domingo), la demanda disminuye considerablemente en comparación con el resto de los días de la semana y se observan valores atípicos. Por otro lado, a continuación, nos fijaremos en qué rangos se mueve la demanda del consumo eléctrico a lo largo del 2015 para identificar más valores atípicos. Se usa un histograma para que aporte esa información con un simple vistazo.

```
hist(data$demand,
     main = "Frecuencia de la demanda",
     ylab = "Frecuencia",
     xlab = "Demanda")
```



Vemos que el histograma nos muestra una distribución extraña de la demanda.

- Hay poca frecuencia de la demanda, cuando la demanda es inferior a 22000 MW y superior a 34000 MW (aproximadamente).
- La frecuencia de la demanda es frecuente entre 22000 MW y 34000 MW.
- La baja demanda es más frecuente de lo normal dentro del conjunto de datos de la serie. Parece que predominan bastantes valores atípicos.

Creemos que es interesante estudiar cuantos valores atípicos hay en la demanda y porqué lo son. Para ello, el primer paso es convertir la variable demanda, una variable numérica, en categórica. Se define la siguiente categorización, basándonos en el histograma, que a priori parece razonable:

- Hasta 22000 MW la demanda es baja (*low*).
- De 22000 a 34000 MW la demanda no se consideraría un valor atípico (*medium*).
- Mayor de 34000 MW la demanda es alta (*high*).

```
breakPoints <- c(0, 22000, 34000, Inf)
categories <- c("low", "medium", "high")
data$demand.C <- cut(data$demand, breaks = breakPoints, labels = categories)
summary(data$demand.C)
```

```
##    low medium   high
##   4453  41588   6519
```

Con esta categorización, se puede decir que la serie tiene aproximadamente 10972 valores atípicos (4453 + 6519). Esta cantidad es bastante alta, y dado este caso sería recomendable aplicar un método de reemplazo de estos valores atípicos que se encuentran fuera del rango medio de la demanda. Además, como curiosidad, antes de reemplazar los *outliers* de la serie, analizaremos si muchos de estos valores atípicos pertenecen a días festivos, fines de semana o días en períodos de vacaciones, como parece que hemos detectado en nuestro primer análisis para identificar valores atípicos en la serie.

```
demand.high <- data[data$demand.C == "high", ]
table(demand.high$day)
```

```
##
##   domingo    jueves     lunes    martes miércoles    sábado   viernes
##        19      1272      1327      1378      1385        67      1071
```

Los días de más demanda eléctrica son los días entre semana.

```
demand.low <- data[data$demand.C == "low", ]
table(demand.low$day)
```

```
##
##   domingo    jueves     lunes    martes miércoles    sábado   viernes
##      1824       246       941       286       184       619       353
```

Los días de menos demanda eléctrica son los fines de semana. Pero se observa un dato extraño, hay muchos lunes donde la demanda es baja. Esto seguramente se debe a que bastantes días festivos del año 2015 fueron un lunes. Lo estudiamos a continuación.

```
demand.monday <- demand.low[demand.low$day == "lunes", ]
sort(table(demand.monday$date))
```

```
##
## 12/01/2015 22/06/2015 08/06/2015 14/12/2015 02/03/2015 10/08/2015 24/08/2015
##          1          1          6          9         14         15         15
## 30/11/2015 23/03/2015 16/03/2015 05/01/2015 09/03/2015 21/12/2015 07/12/2015
##         15         16         18         20         20         22         23
## 17/08/2015 18/05/2015 23/11/2015 01/06/2015 11/05/2015 30/03/2015 14/09/2015
##         24         24         24         25         25         26         27
## 15/06/2015 25/05/2015 05/10/2015 27/04/2015 07/09/2015 13/04/2015 21/09/2015
##         27         27         28         28         29         29         29
```

```
## 28/12/2015 16/11/2015 19/10/2015 20/04/2015 28/09/2015 04/05/2015 09/11/2015
##         30         31         31         31         31         32         32
## 02/11/2015 26/10/2015 06/04/2015 12/10/2015
##         33         34         40         49
```

Vemos que la afirmación anterior es verdad. Algunos ejemplos:

- 12/10/2015: Día de la Hispanidad.
- 06/04/2015: Lunes de Pascua.
- 26/10/2015: Esta fecha es muy interesante, ya que coincide que es el día después del cambio de hora (horario de invierno).
- 09/11/2015: Almudena (sólo en Madrid).
- 28/12/2015: En período vacacional de Navidades.
- 02/11/2015: Puente de la Fiesta de todos los Santos.

En el apartado siguiente, se remplazarán los valores atípicos que se han podido identificar. Ya que si el conjunto de datos de la serie incluye bastantes valores atípicos la predicción será engañosa.

2.2. Reemplazo de outliers

Para reemplazar los valores atípicos, necesitamos dos umbrales. Como se ha comentado anteriormente, se opta por el reemplazo. No se opta por la eliminación de los valores atípicos para no perder datos representativos de la serie. La eliminación de bastantes datos representativos de la serie puede tener un impacto negativo en la predicción. Los umbrales que se definen indicarán qué valores de la demanda serán reemplazados.

Se usa la función *summary()*, que nos devuelve información sobre los cuartiles de la demanda. Los cuartiles son valores que dividen el conjunto de datos de la serie en diferentes partes. Los cuartiles se utilizarán para definir los umbrales.

```
summary(data$demand)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   17985   24392   28566   28349   31664   40648
```

Para el reemplazo que nos traemos entre manos, se observa que el primer cuartil (*Q1*) está en 24392 MW y que el tercer cuartil (*Q3*) está en 31664 MW. El uso de los cuartiles en el reemplazo de los valores atípicos se debe a que entre *Q1* y *Q3* sabemos que se encuentran el 50 % de los valores de la demanda de la serie. La distancia entre los valores de *Q1* y *Q3* se le llama rango intercuartílico (*IQR: InterQuantile Range*).

```
IQR <- 31664 - 24392
IQR
```

```
## [1] 7272
```

Según los cuartiles, se define como valor atípico leve aquel que dista 1.5 veces el rango intercuartílico por debajo de *Q1* o por encima de *Q3*

$$*outlier* < Q1 - 1.5 * IQR \text{ o bien } *outlier* > Q3 + 1.5 * IQR$$

y valor atípico extremo aquel que dista 3 veces el rango intercuartílico por debajo de *Q1* o por encima de *Q3*

$$*outlier* < Q1 - 3 * IQR \text{ o bien } *outlier* > Q2 + 3 * IQR$$

De hecho, con esta información, se calculan los umbrales *highLimit* y *lowLimit*:

```
highLimit <- 24392 + 1.5 * IQR
highLimit
```

```
## [1] 35300
```

```
lowLimit <- 31664 - 1.5 * IQR
lowLimit
```

```
## [1] 20756
```

Todos los valores de la demanda que superen los 35300 MW se consideran valores atípicos. Y todos los valores de la demanda que sean inferiores a los 20756 MW también se consideran valores atípicos.

Una vez identificados los valores atípicos en la demanda, se procede al reemplazo de estos. Para ello, se define una función propia que recibe como parámetros el conjunto de datos de la demanda, el umbral inferior y el umbral superior. En la función se decide reemplazar por la media aquellos valores que estén por debajo del umbral inferior, y por la mediana aquellos que estén por encima del umbral superior, procedimiento muy utilizado en el reemplazo de *outliers*. La función y su aplicación se muestran a continuación:

```
outliersReplace <- function(data, lowLimit, highLimit) {
  data[data < lowLimit] <- as.integer(mean(data))
  data[data > highLimit] <- as.integer(median(data))
  data
}

data$demand.WO <- outliersReplace(data$demand, lowLimit, highLimit)
```

Comprobamos con dos ejemplos si se ha realizado correctamente el reemplazo.

```
data[752, c("demand", "demand.WO")]
```

```
##     demand demand.WO
## 752  20243     28348
```

```r
data[1666, c("demand", "demand.WO")]
```

```
##      demand demand.WO
## 1666  35315     28566
```

Finalmente, se substituyen los valores de la demanda originales por los valores de la demanda sin valores atípicos*.

```r
data$demand <- data$demand.WO
summary(data$demand)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20756   24863   28566   27954   30720   35300
```

Es normal que la media, o la mediana o los cuartiles cambien después del reemplazo, en este caso, la media y los cuartiles, pues los valores atípicos tenían bastante peso en la serie y distorsionaban los datos.

```r
# Borrado de variables temporales para ahorrar espacio
data$date <- NULL
data$time <- NULL
data$month <- NULL
data$day <- NULL
data$demand.C <- NULL
data$demand.WO <- NULL
rm(
  list = c(
    "breakPoints",
    "categories",
    "demand.high",
    "demand.low",
    "demand.monday",
    "IQR",
    "highLimit",
    "lowLimit",
    "componentes.ts"
  )
)
```

Capítulo 3

Modelo ARIMA

3.1. Descripción del modelo

La metodología se puede aplicar solamente a procesos ARMA estacionarios (ARIMA antes de las correspondientes transformaciones para garantizar estacionalidad).

3.2. Metodologia del modelo

Box and Jenkins [1976] definieron una metodología de cuatro etapas para seleccionar el modelo ARIMA subyacente a una serie temporal concreta con el propósito de estimar, contrastar y predecir series temporales.

Las cuatro etapas son las siguientes:

- Identificación
- Estimación
- Contraste
- Predicción

Se pueden dividir las etapas en los pasos siguientes:

1) Representar la serie y calcular la función de autocorrelación (FA) y la función de autocorrelación parcial (FAP) y comprobar si la serie es estacionaria. Si lo es (sus correlaciones decrecen rápidamente) pasar al paso 3, si no lo son (lento decrecimiento) continuar con el paso 2.

2) Tomar logaritmos de la serie si parece que no es estacionaria en varianza (varianza no constante en el tiempo) y/o primeras diferencias si parece que no es estacionaria en media (tiene tendencia o medias distintas por tramos).

3) Examinar la función de autocorrelación (FA) y la función de autocorrelación parcial (FAP) de la nueva serie transformada (si siguiera sin ser estacionaria volver al paso 2 y aplicar una nueva diferencia) e intentar identificar el proceso ARMA teniendo en cuenta las correlaciones simples y parciales significativas (bandas de fluctuación).

4) Estimar el proceso que se ha especificado (máxima verosimilitud).

5) Contrastes de hipótesis:

   5.1) Contraste de significatividad individual (o conjunta) de los parámetros del modelo.

   5.2) Contrastes sobre los residuos del modelo: comprobar que la función de autocorrelación (FA) y la función de autocorrelación parcial (FAP) tienen un comportamiento de ruido aleatorio (ninguna correlación significativa), contraste de normalidad (test de Jarque-Bera) ...

   5.3) Usar los criterios de información de Akaike y Schwarz (AIC, BIC) además del R^2 ajustado para decidir sobre la bondad de los ajustes de posibles especificaciones alternativas (normalmente de la inspección de la función de autocorrelación (FA) y de la función de autocovarianza (FAC) se pueden identificar distintos modelos).

6) Si se deciden cambios en el modelo original volver estimar los nuevos modelos en el paso 4.

7) Predicción bajo normalidad y varianza constante:

8) Evaluación de las predicciones:

http://campus.usal.es/~ehe/perote/documentos/TEMA%202%20MODELIZACION%20ECONOMICA%20II_2.pdf

Capítulo 4

Aplicación del modelo ARIMA

Puesto que el entrenamiento del modelo de una serie temporal muy larga puede ser computacionalmente costoso, se filtran los datos de la serie para contemplar sólo desde el lunes 8 de junio hasta el domingo 14 de junio.

```
ts <- data %>% filter(
  datetime >= as.POSIXct("2015-06-08 00:00:00", tz = "UTC"),
  datetime <= as.POSIXct("2015-06-14 23:50:00", tz = "UTC")
  ) %>%
  select(demand) %>%
  ts(frequency = 24 * 60 / 10)
```

Se utilizan los siguientes paquetes para la aplicación del modelo ARIMA:

```
library(tidyverse)
library(xts)
library(forecast)
library(tseries)
```

Las librerías *tidyverse*, *xts* y *forecast* se han usado anteriormente. En este capítulo se añade la librería *tseries.*

- tseries es una librería para analizar series temporales.

4.1. Transformación de la serie

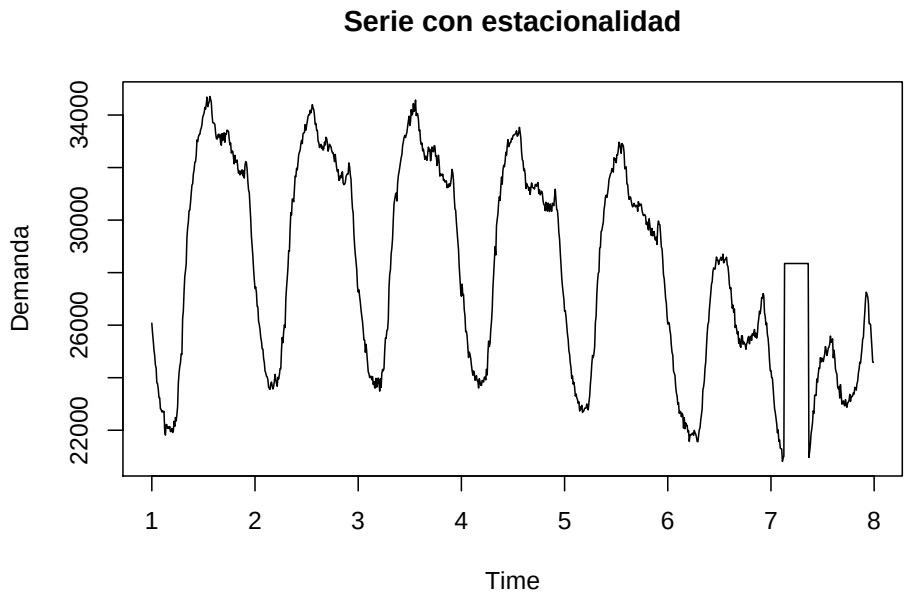
El análisis previo del primer capítulo nos ha revelado que la serie temporal no es estacionaria por su estacionalidad, y como el modelo que vamos a utilizar para la predicción es un modelo ARIMA, necesitamos que la serie sea estacionaria. Para ello, tendremos que transformar la serie eliminando la estacionalidad. Diferenciando la serie lograremos que se convierta en estacionaria. Para empezar con la transformación de la serie, primero se observará el resultado de la función *ndiffs()*, que calcula el número de diferenciaciones estacionales que se necesitan llevar a cabo para que la serie sea estacionaria.

```r
ndiffs(ts)
```

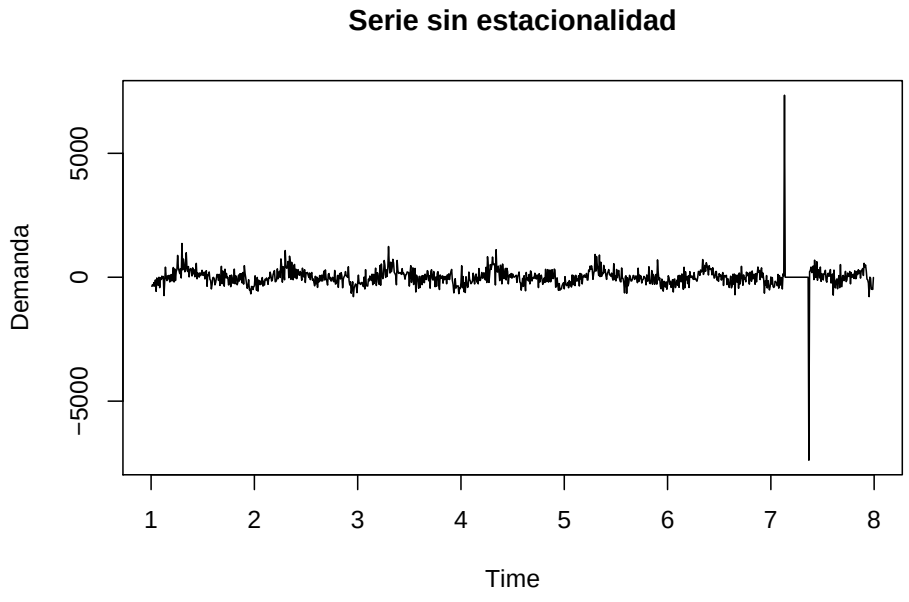
```
## [1] 1
```

El respectivo cálculo muestra que la serie necesita de una diferenciación estacional. Para eliminar la estacionalidad se usa la función *diff()* que resta el componente estacional de la serie original y luego lo diferencia para que sea estacionario (varianza constante e independiente).

```r
tsstationary <- diff(ts, differences = 1)
plot(ts,  main = "Serie con estacionalidad", ylab = "Demanda")
```

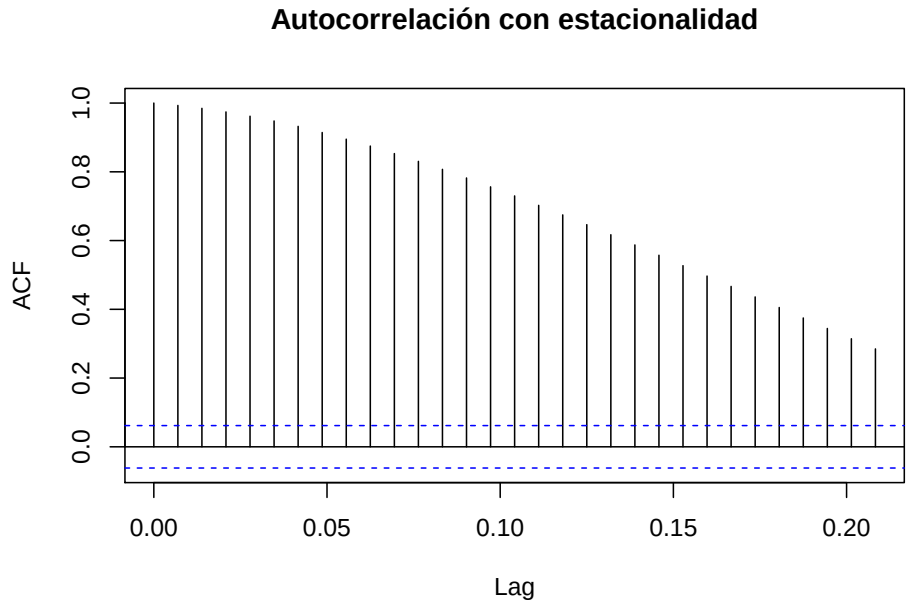



```r
plot(tsstationary, main = "Serie sin estacionalidad", ylab = "Demanda")
```

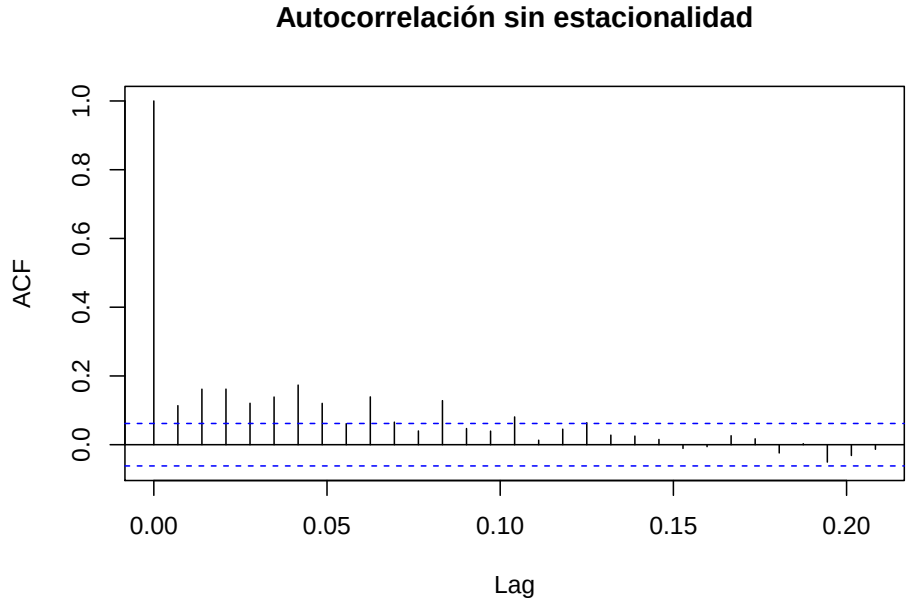


Como podemos ver ahora en la serie se ha eliminado la componente de estacionalidad. Aunque la serie parece tener ruido, que trataremos más adelante, se parece bastante a una serie estacionaria, ya que parece ser constante en media y varianza. Volvemos a visualizar la función de autocorrelación.

```
acf(ts, main = "Autocorrelación con estacionalidad")
```




```
acf(tsstationary, main = "Autocorrelación sin estacionalidad")
```



Podemos apreciar que la serie ya es estacionaria una vez se ha eliminado la componente estacional. Para asegurarnos se aplica la prueba de estacionalidad como la prueba de Dickey-Fuller Aumentado *(Augmented Dickey-Fuller Test (ADF), en inglés)* y la prueba de Phillips–Perron (PP), una modificación del test de Dickey-Fuller. En estas pruebas, se considera que

en la hipótesis nula H_0 se observan raíces unitarias en la serie, por tanto, no es estacionaria. Al contrario, la serie es estacionaria (H_1). Con un *p-value* inferior a 0.05 (significancia α), la hipótesis nula H_0 se rechaza, confirmando que la serie es estacionaria.

El motivo de basarse en la observación de las raíces unitarias es porqué una raíz unitaria, es una tendencia estocástica en la serie temporal. Algunas veces se le llama *"paseo aleatorio con deriva"*. Y, por tanto, si la serie temporal tiene raíces unitarias, ésta presenta un patrón sistemático que es impredecible. Entonces, la serie temporal es estacionaria si un cambio en el tiempo no cambia la forma de la distribución, y las raíces unitarias son una causa de no estacionalidad.

Se aplica la prueba de Dickey-Fuller Aumentado:

```
adf.test(tsstationary)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  tsstationary
## Dickey-Fuller = -6.8215, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

El valor del *p-value* de 0.01, indica que rechazamos la hipótesis nulaH_0, indicando que la serie es estacionaria. A continuación se aplica la prueba de PP:

```
pp.test(tsstationary)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  tsstationary
## Dickey-Fuller Z(alpha) = -1191.1, Truncation lag parameter = 7, p-value
## = 0.01
## alternative hypothesis: stationary
```

El valor del *p-value* de 0.01, indica que también rechazamos la hipótesis nulaH_0, indicando que la serie es estacionaria.

4.2. Ajuste del modelo

Después de eliminar la estacionalidad y hacer que los datos sean estacionarios, se dividen los datos en un conjunto de entrenamiento y en un conjunto de prueba.

```
trainset <- subset(tsstationary,
                   end = length(tsstationary) - (24 * 60 / 10))
testset <- subset(tsstationary,
                  start = length(tsstationary) - (24 * 60 / 10) + 1)
```

Una vez que tenemos los datos listos y han satisfecho todas las suposiciones del modelo ARIMA, para determinar el orden del modelo que se ajustará a los datos, necesitamos tres variables: p, d y q que son enteros positivos que se refieren al orden de las partes medias autorregresivas, integradas y móviles del modelo, respectivamente.

Para identificar que valores de p, d y q serán apropiados, se usa la función *auto.arima()*. La función calcula automáticamente cuál es la mejor combinación de órdenes para el modelo. Para ello, utiliza una combinación de pruebas de raíz unitaria, minimización de los valores AIC y MLE para obtener el mejor modelo. Básicamente, los valores p, d y q son elegidos minimizando el AIC. La función utiliza el algoritmo Hyndman and Khandakar [2008], que usa una búsqueda por pasos para recorrer el espacio del modelo para seleccionar el mejor modelo con el AIC más pequeño.

```
fitARIMA <- auto.arima(trainset, trace = TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2)(1,1,1)[144] with drift         : Inf
##  ARIMA(0,0,0)(0,1,0)[144] with drift         : 8444.229
##  ARIMA(1,0,0)(1,1,0)[144] with drift         : Inf
##  ARIMA(0,0,1)(0,1,1)[144] with drift         : Inf
##  ARIMA(0,0,0)(0,1,0)[144]                    : 8442.454
##  ARIMA(0,0,0)(1,1,0)[144] with drift         : Inf
##  ARIMA(0,0,0)(0,1,1)[144] with drift         : Inf
##  ARIMA(0,0,0)(1,1,1)[144] with drift         : Inf
##  ARIMA(1,0,0)(0,1,0)[144] with drift         : 8336.283
##  ARIMA(1,0,0)(0,1,1)[144] with drift         : Inf
##  ARIMA(1,0,0)(1,1,1)[144] with drift         : Inf
##  ARIMA(2,0,0)(0,1,0)[144] with drift         : 8321.729
##  ARIMA(2,0,0)(1,1,0)[144] with drift         : Inf
##  ARIMA(2,0,0)(0,1,1)[144] with drift         : Inf
##  ARIMA(2,0,0)(1,1,1)[144] with drift         : Inf
##  ARIMA(3,0,0)(0,1,0)[144] with drift         : 8324.007
##  ARIMA(2,0,1)(0,1,0)[144] with drift         : 8322.7
##  ARIMA(1,0,1)(0,1,0)[144] with drift         : 8324.375
##  ARIMA(3,0,1)(0,1,0)[144] with drift         : 8325.721
##  ARIMA(2,0,0)(0,1,0)[144]                    : 8320.609
##  ARIMA(2,0,0)(1,1,0)[144]                    : Inf
##  ARIMA(2,0,0)(0,1,1)[144]                    : Inf
##  ARIMA(2,0,0)(1,1,1)[144]                    : Inf
##  ARIMA(1,0,0)(0,1,0)[144]                    : 8334.894
##  ARIMA(3,0,0)(0,1,0)[144]                    : 8322.812
##  ARIMA(2,0,1)(0,1,0)[144]                    : 8321.518
##  ARIMA(1,0,1)(0,1,0)[144]                    : 8323.262
##  ARIMA(3,0,1)(0,1,0)[144]                    : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(2,0,0)(0,1,0)[144]                    : 9925.185
##
##  Best model: ARIMA(2,0,0)(0,1,0)[144]
```

```
fitARIMA
```

```
## Series: trainset
## ARIMA(2,0,0)(0,1,0)[144]
##
## Coefficients:
##           ar1      ar2
##       -0.4298  -0.1501
## s.e.   0.0370   0.0370
##
## sigma^2 estimated as 57549:  log likelihood=-4959.58
## AIC=9925.15   AICc=9925.18   BIC=9938.88
```

En este caso la función sugiere que el mejor modelo que representa a la serie sería un ARIMA(2,0,0)(0,1,0) sin tendencia, y que arroja un AIC de 9925.15. Lo validaremos con la prueba *Ljung-Box* (una versión simplificada de la prueba de Box y Pierce) que evalúa si los residuos de la serie son ruido aleatorio.

La prueba de *Ljung-Box* es una prueba para comprobar si una serie de observaciones, en un período de tiempo específico, son aleatorias o independientes en todos los retardos hasta el retardo especificado. En lugar de probar la aleatoriedad en cada retardo distinto, se prueba la aleatoriedad *"general"* basada en un número de retardos y, por lo tanto, también es una prueba de comparación.

La prueba se aplica a los residuos de un modelo ARIMA ajustado y no a la serie original, y en tales aplicaciones, la hipótesis que se prueba es que los residuos del modelo no tengan autocorrelación. Si los retardos no son independientes, un retardo puede estar relacionado con otros retardos de k unidades de tiempo, por lo que la autocorrelación puede reducir la exactitud del modelo predictivo y conducir a una interpretación errónea de los datos. La hipótesis nula H_0 indicará que los datos no son diferentes del ruido aleatorio (H_0: los residuos se distribuyen aleatoriamente).

A continuación, se aplica la prueba de *Ljung-Box* sobre el modelo ARIMA(2,0,0)(0,1,0), si el modelo rechaza la hipótesis nula H_0, debemos considerar el modelo como problemático.

```
Box.test(fitARIMA$residuals, type = "Ljung-Box")
```

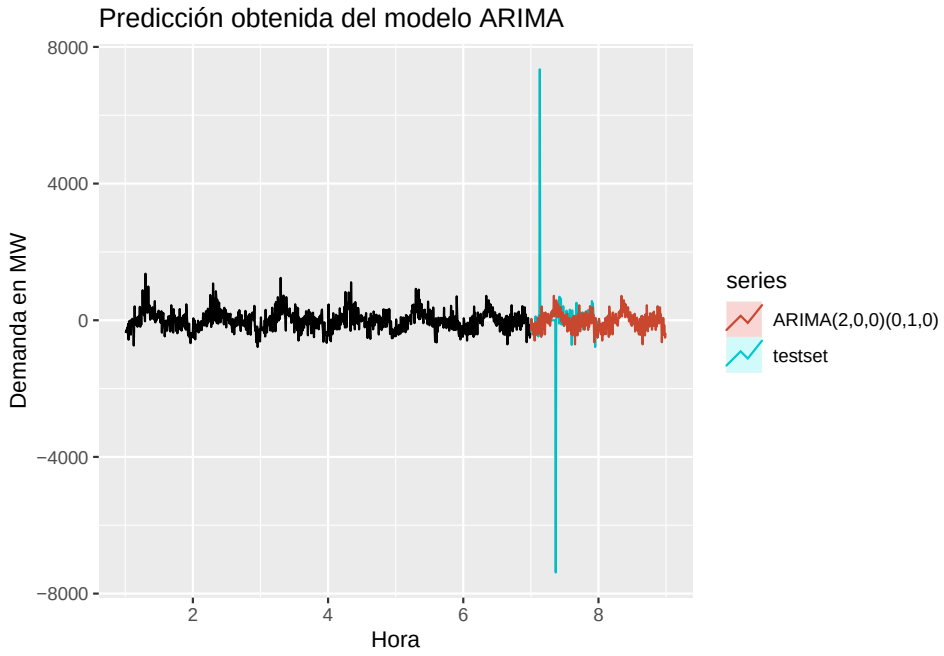
```
##
##  Box-Ljung test
##
## data:  fitARIMA$residuals
## X-squared = 0.023686, df = 1, p-value = 0.8777
```

En este caso, el valor de p (*p-value = 0.8777*) está lejos de ser significativo (inferior a 0.05). Por lo tanto, no podemos rechazar la hipótesis nula H_0de que los residuos se distribuyen aleatoriamente y se concluye que el modelo, como se especifica, se ha ajustado correctamente.

4.3. Predicción

Los parámetros del modelo ARIMA se pueden usar como un modelo predictivo para hacer pronósticos de valores futuros de las series temporales, una vez que se selecciona el modelo más adecuado para los datos de la serie temporal. Se usará la función *forecast()* para la predicción de la serie.

```
forecastARIMA <- forecast(fitARIMA)
autoplot(trainset, xlab = "Hora", ylab = "Demanda en MW") +
  autolayer(testset) +
  autolayer(forecastARIMA, series = "ARIMA(2,0,0)(0,1,0)", PI = FALSE) +
  ggtitle("Predicción obtenida del modelo ARIMA")
```



La figura muestra que el modelo proporciona una buena predicción. Finalmente, se guarda el resultado de la predicción en un fichero Excel. Para ello, necesitaremos el paquete openxlsx.

```
library(openxlsx)

resultado <- createWorkbook()
addWorksheet(resultado, "resultado")
writeData(resultado, "resultado", forecastARIMA, rowNames = TRUE)
saveWorkbook(resultado, "resultado.xlsx", overwrite = TRUE)
```

Bibliografía

G. E. P. Box and G. M. Jenkins. Some comments on a paper by chatfield and prothero and on a review by kendall. *Journal of the Royal Statistical Society*, 136(3):337–352, 1976.

Fernando Campos. Análisis de series temporales en r. arima. Disponible en https://www.diegocalvo.es/analisis-de-series-temporales-en-r-arima/, 2018.

R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2nd edition edition, 2018.

Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: the forecast package for r. *Journal of Statistical Software*, 27(3), 2008.

Pedro L. Luque-Calvo. Escribir un trabajo fin de estudios con r markdown. Disponible en http://destio.us.es/calvo/post/escribir-un-trabajo-fin-de-estudios-con-rmarkdow/, 2017.

Juan Antonio Jiménez Torres. Detección y reemplazo de outliers con r. Disponible en https://www.adictosaltrabajo.com/2019/11/28/deteccion-y-reemplazo-de-outliers-con-r/, 2019.