Universidad de Valladolid

Facultad de Ciencias

Trabajo de fin de grado

Grado en Estadística

Creación de un paquete de análisis de series temporales para RStudio

Autor: Daniel López Martínez

Tutora: María Pilar Rodríguez del Tío

Agradecimientos

Quiero agradecer a mi tutora, María Pilar Rodríguez del Tío por su paciencia y dedicación a lo largo del desarrollo de este trabajo.

A mi familia y a Miriam por el apoyo constante que me han brindado a lo largo de toda la carrera, y a mis amigos por los ánimos y ayuda que me han proporcionado en todo momento.

Índice general

Índice de figuras

Índice de tablas

Resumen

En este trabajo se presenta el paquete CBJTSA para el lenguaje de programación R, el cual implementa distintas funciones para el análisis de series temporales mediante modelos ARIMA, completando las salidas de otros paquetes de R. Este paquete está enfocado al ámbito educativo, por lo que el objetivo del mismo será facilitar el acceso a herramientas de análisis de series temporales, usando este paquete como sustituto de otras herramientas más complejas y costosas.

A lo largo de esta memoria se describen las distintas funciones que contiene el paquete con detalle, explicando su uso y las salidas que proporciona. Además, se incluye un ejemplo práctico de análisis de una serie temporal utilizando el paquete CBJTSA, para ilustrar su funcionamiento en un caso más específico.

Palabras claves: Paquetes, CBJTSA, R, Series Temporales, ARIMA.

Abstract

This paper presents the CBJTSA package for the R programming language, which implements different functions for the analysis of time series using ARIMA models, complementing the outputs of other R packages. This package is focused on the educational enviroment, so its objective is to facilitate access to time series analysis tools, using this package as a substitute for other more complex and expensive tools.

Throughout this report, the different functions contained in the package are described in depth, explaining its use and the outputs it provides. In addition, a practical example of a time series analysis using the CBJTSA package is included to illustrate its operation in a more specific case.

Keywords: Package, CBJTSA, R, Time Series, ARIMA.

Capítulo 1

Introducción

El análisis univariante de series temporales se compone de un conjunto de métodos y técnicas para el estudio de una variable que evoluciona a lo largo del tiempo. Este tipo de análisis tiene suficiente importancia como para dedicarle una asignatura en cualquier grado universitario de estadística o relacionado con la economía. Sin embargo, la mayoría de herramientas existentes que implementan este tipo de análisis suelen ser complicadas de utilizar para los estudiante y en algunos casos muy costosas económicamente hablando.

Es por esto que el objetivo principal de este trabajo es la creación de una herramienta para análisis de series temporales sencilla y gratuita. De tal manera que cualquier persona pueda utilizarla como iniciación al análisis de series temporales. Para esto se ha decidido crear un paquete para el lenguaje de programación R [1], el cual es gratuito, sencillo de utilizar y muy versátil. Este paquete se ha llamado *Complete Box Jenkins Time Series Analysis* o CBJTSA abreviado.

El objetivo del paquete CBJTSA es crear una herramienta sencilla de utilizar para el análisis de series temporales, mediante modelos ARIMA, para el ámbito educacional. Su desarrollo se ha basado en otras herramientas como el procedimiento `arima` de SAS, y en otros paquetes de análisis de series temporales para R ya existentes. De esta manera se quiere plantear su uso como sustituto de otro software, cuya licencia suele ser de un coste considerable, para facilitar el acceso a este tipo de análisis sin limitaciones tanto a los alumnos como a los profesores.

Si bien R ya cuenta con diversos paquetes para el análisis de series temporales, en algunos casos se ha considerado que las funciones que proporciona son poco intuitivas y en ocasiones poco completas. Por ello, el paquete CBJTSA se ha desarrollado tratando de unificar las distintas funcionalidades que proporcionan algunos paquetes de R, y completar la salida de estos con gráficos y estadísticos, que ayuden a entender todos los conceptos que son objeto de estudio en una asignatura como la ya mencionada.

El contenido de esta memoria se puede dividir en tres partes. Empezando por un contexto teórico en el que se explican los distintos métodos y técnicas que utiliza el paquete CBJTSA para el análisis de series temporales (Capítulo 2). A continuación, se describe el contenido

del paquete desarrollado, junto con una explicación de como se ha implementado cada una de sus funciones (Capítulo 3). Por último, se presentan unos datos de ejemplo sobre los que se realiza un análisis de series temporales utilizando el paquete CBJTSA, como aplicación práctica (Capítulo 4). Además, se incluye un capítulo de comparación del paquete con otras herramientas y paquetes de R (Capítulo 5).

Capítulo 2

Contexto

El objetivo principal de este trabajo es la creación de un paquete que permita realizar el análisis completo de una serie temporal mediante el lenguaje de programación R. Así pues, la estructura del paquete se llevará a cabo siguiendo la metodología Box-Jenkins, descrita en [2]. Esta metodología para el análisis de una serie temporal se explica en muchos de los grados relacionados con la estadística, actualmente en vigor, en particular en la asignatura de Análisis de Series Temporales del Grado en Estadística de la Universidad de Valladolid (en ella se explica tomando como software de referencia el SAS). En este capítulo se describirá brevemente dicha metodología, así como los contenidos de los paquetes ya existentes en R utilizadas para realizar este análisis y las limitaciones de las mismas.

En la Sección 2.1 se describe el proceso iterativo en el que se aplicará la metodología Box-Jenkins con las distintas fases que le componen. Posteriormente en la Sección 2.2 se muestran los paquetes existentes para realizar este tipo de análisis así como sus limitaciones.

2.1. Metodología Box-Jenkins. Modelos ARIMA

La metodología Box-Jenkins, descrita en [2], define modelos capaces de representar series temporales homogéneas y equilibradas estadísticamente en muchos aspectos. El modelo ARIMA de orden (p, d, q) para la serie z_t se define por la ecuación

$$\phi(B)(1-B)^d z_t = \theta_0 + \theta(B)a_t$$

donde $\phi(B)$ y $\theta(B)$ son polinomios en B, donde B es el operador de retardo, y a_t es un ruido blanco. Denotamos por p y q los grados de los polinomios, respectivamente.

El desarrollo de modelos de este tipo para la descripción de una serie temporal observada se puede alcanzar mediante un proceso iterativo que habitualmente consta de tres pasos:

1. Fase de **Identificación**, en la que se usan los datos y cualquier información sobre cómo se han generado para proponer una serie de modelos que podrían ser adecuados.

2. Fase de **Estimacion**, en la que se usan los datos para hacer inferencias sobre los parámetros de cada modelo propuesto en el paso anterior.

3. Fase de **Validación**, en la que se comprueba si los residuales del modelo estimado cumplen las hipótesis necesarias para un buen ajuste, o por el contrario, se detectan deficiencias en el modelo, que teniéndolas en cuenta pueden llevarnos a conseguir mejoras en el mismo.

A estos pasos se añade una cuarta fase de **Comparación** para los casos en los que tenemos varios modelos que se ajustan bien a los datos, en este se comparan dichos modelos para elegir aquel que mejor se ajuste a los datos observados y con el que se obtengan las mejores predicciones. Una vez elegido el modelo final se podría considerar como una última fase la **Predicción**, en la cual se calculan predicciones de los datos observados a partir del modelo ajustado.

A continuación se describen las fases de este proceso iterativo, detallando los procesos y métodos más adecuados para cada uno de los pasos.

2.1.1. Identificación

En esta fase de identificación se abusará más de métodos gráficos para examinar la serie y explicar su comportamiento, en este sentido los gráficos de autocorrelaciones serán una herramienta especialmente útil. La finalidad de este paso es proponer un buen abanico de modelos que podrían ser adecuados, para ello se tendrá que conseguir que la serie sea estacionaria (si no lo es ya) mediante diferenciaciones de la serie original y después identificar el modelo autoregresivo de media móvil al que corresponde la serie diferenciada.

Antes de esto hay algunos aspectos de la serie observada que deben tratarse:

- Si la serie presenta una varianza dependiente del tiempo, es posible que requiera una transformación para estabilizar la varianza. Esto se puede detectar en el gráfico de la serie si, por ejemplo, observamos que los datos presentan mayor variabilidad. Pero también es útil el gráfico rango-media, para decidir si una transformación es necesaria, e incluso qué transformación realizar.

- Respecto a la componente estacional de la serie, hay ocasiones en las que el periodo de la parte estacional no es tan claro de distinguir en el gráfico de la serie observada. Sin embargo, el periodograma nos ayuda en este aspecto, clarificando cual puede ser el periodo de la componente estacional.

Con estos aspectos ya centrados podemos empezar con el problema de la estacionaridad en la serie original. Es importante que la serie con la que trabajemos sea estacionaria, ya que los modelos que intentamos ajustar son en la teoría estacionarios. Dividiremos este problema en la componente regular y la componente estacional, si la serie no parece tener una parte estacional solo será necesario atender a la parte regular.

Componente regular

Para analizar la estacionaridad de una serie el método más común es calcular la función de autocorrelación, ya que en un proceso estacionario autoregresivo de media móvil de orden

$(p, 0, q)$, $\phi(B)\tilde{z}_t = \theta(B)a_t$, su función de autocorrelación cumple la ecuación en diferencias

$$\phi(B)\rho_k = 0, \qquad k > q.$$

Además, si $\phi(B) = \prod_{i=1}^{p}(1 - G_i B)$, la solución a la ecuación en diferencias para la autocorrelación en el retardo k es, asumiendo raíces reales y distintas, de la forma

$$\rho_k = A_1 G_1^k + A_2 G_2^k + ... + A_p G_p^k, \qquad k > q - p.$$

El requisito de estacionaridad de que los ceros de $\phi(B)$ deben estar fuera del círculo unidad implica que las raíces de la ecuación característica, $G_1, G_2, ..., G_p$ deben estar dentro del círculo unidad.

Esta expresión muestra que en el caso de un modelo estacionario en el que ninguna raíz este cerca de los límites del círculo unidad, la función de autocorrelación descenderá rápidamente para valores de k grandes y también moderados. Supongamos ahora que una raíz real G_1 se aproxima a uno

$$G_1 = 1 - \delta,$$

donde δ es positivo y de valor bajo. Entonces, dado que para valores altos de k

$$\rho_k \simeq A_1(1 - k\delta),$$

la función de autocorrelación no descenderá rápidamente, sino que decrecerá lentamente y de forma casi lineal. Este mismo argumento se puede aplicar si más de una raíz se aproxima a uno.

Se puede afirmar que una tendencia en la función de autocorrelación que no descienda rápidamente es un indicador de que puede existir una raíz próxima a uno. Y, dado que la función de autocorrelación estimada tiende a comportarse como la función teórica, se puede deducir que si no se observa un descenso rápido en la función de autocorrerlación estimada, la serie temporal estudiada puede no ser estacionaria. Esto último es fácilmente apreciable en la representación gráfica de la función de autocorrelación o correlograma.

Si estamos ante el caso de una serie no estacionaria es necesario diferenciar la serie hasta obtener una que sí lo sea, es decir, hasta que se distinga un decrecimiento rápido en la función de autocorrelación. En la práctica esto se consigue normalmente con 0, 1 o 2 diferenciaciones regulares, demasiadas diferenciaciones pueden complicar el modelo y hacer que la varianza de la serie aumente, por lo que se debe utilizar la primera diferenciación que haga la serie estacionaria.

Componente estacional

Partiendo de que conocemos el periodo p de la componente estacional, solo podremos comprobar si la serie es estacionaria por la parte estacional si nos hemos asegurado de que la serie es estacionaria por la parte regular, ya que una tendencia grande puede ocultar la

parte estacional de la serie. Por ello, es importante comprobar de nuevo la función de autocorrelación después de diferenciar la serie.

Para saber si la serie es estacionaria por la parte estacional seguiremos utilizando la función de autocorrelación, y observaremos algo similar al caso de la parte regular. En este caso el descenso rápido se debe observar en los retardos del correlograma que sean múltiplos del periodo p, implicando esto estacionaridad por la parte estacional. Igual que antes, si no se da el decrecimiento rápido, se resuelve aplicando diferenciaciones pero de orden p aplicadas una o dos veces. De nuevo debemos quedarnos con la primera diferenciación que convierta a la serie en estacionaria, ya que sobrediferenciar puede causar un aumento de la varianza.

Habiendo conseguido ya la serie estacionaria podemos pasar a encontrar el abanico de modelos que podrían ser adecuados.

Identificar un modelo ARMA

Suponiendo que tenemos ya un valor para el grado de diferenciación, tanto regular d, como estacional D, debemos examinar el patrón de función de autocorrelación y de autocorrelación parcial para elegir los órdenes p y q del modelo ARMA. Esto también incluye la posibilidad de un modelo multiplicativo del tipo SARMA$(p,q)(P,Q)_s$. Disponiendo de las funciones de autocorrelación y autocorrelación parcial estimadas, debemos compararlas con las funciones teóricas de modelos con distintos valores de p, q, P y Q.

Si bien esto no siempre es fácil, se puede obtener intervalos de confianza para saber si ρ_k es significativamente distinto de cero, simplificando así la comparación. En este apartado no vamos a entrar en detallar las principales características de las funciones de autocorrelación y autocorrelación parcial de modelos AR, MA, ARMA ó SARMA, para una descripción más detallada véase [2].

2.1.2. Estimación

En este paso se calculan estimaciones para los parámetros de los distintos modelos planteados en el apartado anterior, los cuales se asume que se han seleccionado adecuadamente. Los métodos más habituales de estimación son el método de mínimos cuadrados y el método de máxima verosimilitud. Implementar estos métodos no es sencillo y existen variantes en su implementación. Una de ellas para el método de estimación máximo verosímil del modelo ARIMA es utilizar el filtro de Kalman. Dado que el software utilizado en este trabajo implementa la estimación mediante este método, este apartado se centrará en su explicación. Se pueden encontrar explicaciones más detalladas en [3] y [4].

Si bien el filtro de Kalman nos permite evaluar la función de verosimilitud de cualquier modelo ARIMA, inicialmente fue diseñado para resolver el problema de estimación en modelos de espacio de estados, que se detallan a continuación.

Modelos de espacio de estados

Un modelo de espacio de estados para una serie temporal (que puede ser multivariante) $z_t, t = 1, 2, ...$, se representa mediante dos ecuaciones. La primera, conocida como ecuación de observación, es:

$$z_t = H_t\alpha_t + \epsilon_t, t = 1, 2, ...,$$

donde z_t es el vector de observaciones de tamaño k de la serie, H_t es la matriz de dimensiones $k \times p$ de valores conocidos para todo t, α_t, son las p variables de estado que no se observan y ϵ_t, un proceso de ruido blanco con $\epsilon_t \sim WN(0, V_t)$. La segunda ecuación, llamada ecuación de estado, describe las variables de estado α_t:

$$\alpha_t = \Omega_t\alpha_{t-1} + u_t, t = 1, 2, ...,$$

donde Ω_t es la matriz de dimensiones $p \times p$ de valores conocidos y u_t es otro proceso de ruido blanco, $u_t \sim WN(0, R_t)$.

Cualquier modelo ARMA(p, q) y ARIMA(p, d, q), puede escribirse en esta formulación y, si bien una representación en el espacio de estados no es única, en [3] y [4] se describe una posible representación para estos modelos.

El filtro de Kalman

En este apartado se explica a grandes rasgos el filtro de Kalman, algoritmo recursivo de predicción que se realiza en tres pasos. Para una explicación más detallada, véase [3].

El primer paso consiste en estimar el estado futuro utilizando una estimación del estado actual. Suponemos conocidas las observaciones $Z_{t-1} = \{z_1, ...z_{t-1}\}$, y el estimador del vector de estado $\hat{\alpha}_{t-1}$, y se desea estimar $\hat{\alpha}_{t|t-1}$, el siguiente valor del estado, mediante Z_{t-1}. La estimación se calcula mediante esperanzas condicionadas a Z_{t-1} obteniendo:

$$\hat{\alpha}_{t|t-1} = \Omega_t\hat{\alpha}_{t-1}, \tag{2.1}$$

y, junto con la estimación, calculamos la matriz de covarianzas de esta estimación $S_{t|t-1}$, cuya expresión final será:

$$S_{t|t-1} = \Omega_t S_{t-1}\Omega_t' + R_t. \tag{2.2}$$

El segundo paso consiste en predecir la nueva observación z_t usando la información hasta el momento $t - 1$. Podemos obtener esta predicción mediante la esperanza condicionada a Z_{t-1} de tal manera que:

$$\hat{z}_{t|t-1} = E(z_t|Z_{t-1}) = H_t\hat{\alpha}_{t|t-1}. \tag{2.3}$$

Podemos medir la incertidumbre de esta predicción utilizando la matriz de varianzas y covarianzas de los errores de predicción:

$$P_{t|t-1} = H_t S_{t|t-1}H_t' + V_t. \tag{2.4}$$

Por último, el tercer paso consistirá en revisar la estimación de estado a la vista de la nueva información. La estimación de $\hat{\alpha}_t = \hat{\alpha}_{t|t} = E(\alpha_t|Z_t)$ se puede calcular por regresión o con las propiedades de la normal multivariante, obteniendo:

$$\hat{\alpha_t} = \hat{\alpha}_{t|t-1} + K_t(z_t - \hat{z}_{t|t-1}), \tag{2.5}$$

y la matriz de covarianzas de la estimación S_t:

$$S_t = S_{t|t-1} - S_{t|t-1}H_t' P_{t|t-1}^{-1} H_t S_{t|t-1}. \tag{2.6}$$

Las ecuaciones (2.1) - (2.6) son las que forman el filtro de Kalman que, bajo hipótesis de normalidad, proporcionan estimaciones y predicciones óptimas.

Para poder aplicar el filtro de Kalman a la función de verosimilitud del modelo ARMA, se necesita escribir dicho modelo en el espacio de estados y, calcular los errores de predicción $e_t = z_t - \hat{z}_{t|t-1}$, y sus varianzas $P_{t|t-1}$. Para empezar el filtro, se necesita dar valores iniciales a las variables de estado α_0, y su matriz de covarianzas S_0.

2.1.3. Validación

En este paso se utilizan diversos gráficos y estadísticos para comprobar si el o los modelos estimados son adecuados o no para la serie observada. Si bien existen varios métodos para validar un modelo, nos centraremos en explicar el más importante, el análisis de autocorrelaciones de los residuos. Explicaciones mas detalladas se pueden encontrar en [2] y [4].

Primero debemos recordar las hipótesis que hacemos al construir un modelo de Box-Jenkins, que son (a parte de la estacionaridad e invertibilidad del modelo ARMA) que las variables a_t sean normales independientes con media cero y varianza constante, es decir

$$\{a_t\} \sim WN(0, \sigma^2).$$

Y por lo tanto se espera que los residuales del ajuste de cada modelo no contradigan estas hipótesis. A continuación se explican los distintos métodos que existen para comprobar que estas hipótesis se cumplen.

Representación de la función de autocorrelación residual

Es bien conocido que para tamaños de muestra n grandes, las distribuciones de las autocorrelaciones muestrales de una secuencia de variables i.i.d. (independientes igualmente distribuidas) $Y_1, ..., Y_n$ con varianza constante se aproximan a una distribución $N(0, 1/n)$. Por tanto, se puede comprobar si los residuales se comportan como un ruido de variables i.i.d., examinando la función de autocorrelación muestral de los residuales, representando la función junto con la banda de confianza del 95 % $\pm 1.96/\sqrt{n}$.

Test de Ljung-Box

Este test es uno de los llamados test de "Portmanteau" que estudia las autocorrelaciones de los residuales, y es un buen complemento al análisis gráfico de la función de autocorrelación residual. Este test tiene como objetivo contrastar que el conjunto de los K primeras autocorrelaciones de los residuos son iguales a cero,

$$H_0 : \rho_1(a) = \rho_2(a) = ... = \rho_k(a) = 0,$$

y con ello comprobar si los residuos son incorrelados o no.

Supongamos que tenemos las K primeras autocorrelaciones residuales $r_k(\hat{a})(k = 1, 2, ..., K)$ de cualquier modelo ARIMA(p, d, q), entonces podemos comprobar que si el modelo es apropiado,

$$Q = n \sum_{k=1}^{K} r_k^2(\hat{a}),$$

este estadístico de Box y Pierce sigue una distribución $\chi^2(K - p - q)$, donde $n = N - d$ es el número de observaciones de la serie diferenciada w_t usadas para estimar el modelo. Sin embargo, si el modelo es inapropiado, la media de los valores de Q se verán aumentados, por lo tanto se lleva a cabo un test de "Portmanteau" aproximado utilizando Q, con la distribución χ^2 mencionada, como estadístico del contraste.

En lugar de esa expresión del estadístico se usa otra desarrollada por Ljung y Box, que aproxima mejor la distribución del estadístico Q,

$$\tilde{Q} = n(n + 2) \sum_{k=1}^{K} (n - k)^{-1} r_k^2(\hat{a}),$$

el cual tiene aproximadamente una media $E[\tilde{Q}] \approx K - p - q$ y distribución $\chi^2(K - p - q)$.

Otros test

Si bien los anteriores métodos son los más habituales para validación de modelos, es bueno disponer de otros test como complemento a estos. A continuación se describen brevemente algunos de ellos:

- *Test de la media de los residuales.* Se puede realizar un contraste de hipótesis sobre la media de los residuales $H_0 : \mu_a = 0$ a partir de $\bar{a} = \frac{\sum a_t}{n}$ y su varianza muestral $s_a^2 = \frac{\sum (a_t - \bar{a})^2}{n-p-q}$. Es importante realizar este contraste después de comprobar que los residuos son incorrelados, para que s_a^2 sea un buen estimador de la varianza.
- *Gráfico rango-media de los residuales.* Como ya se explica en la Sección 2.1.1 el gráfico rango-media es una representación que nos ayuda a estudiar si la varianza es dependiente del tiempo o si por el contrario es constante, como se asume durante la estimación del modelo.

- *Test de normalidad de los residuales*. Para contrastar la hipótesis de normalidad de los residuos se puede utilizar tanto métodos gráficos como un test. En lo referente a gráficos, se puede usar un cuantil-cuantil o una representación de la densidad de los residuos. En cuanto a contrastes de hipótesis, el test de Shapiro-Wilk es un test adecuado para la comprobación de la normalidad en los residuos.

2.1.4. Comparación

Si nos encontramos en el caso en el que se dispone de dos o más modelos estimados y validados se nos plantea el problema de que modelo escoger para representar la serie observada. Una opción es ser más estrictos en la fase de validación y solo aceptar aquellos modelos que superen ampliamente los test descritos, sin embargo, eso puede no ser la mejor opción. Suele ser mejor optar por comparar los modelos ya validados según diversos criterios para elegir el que mejor se ajuste a los datos observados y además, consiga predicciones buenas.

Criterios de información

Existen diversos criterios para la comparación de modelos, pero solo trataremos dos de los más conocidos y usados. Para obtener información más detallada sobre estos y otros criterios, véase [4].

El criterio de información de Akaike o AIC abreviado, fue diseñado como un estimador insesgado aproximado para el índice de Kullback-Leibler de la relación entre el modelo estimado y el modelo real. Su expresión más habitual es

$$AIC = -2ln(L) + 2m,$$

siendo L la verosimilitud del modelo ARIMA(p, d, q) estimado y m el número de parámetros libres, es decir, $p + q + 1$ en el caso de que tenga constante significativa o $p + q$ en caso contrario. Se debe elegir aquel modelo con menor valor del AIC.

El criterio Bayesiano de Schwarz (SBC) o criterio de información Bayesiano (BIC), es otro criterio que pretende corregir la sobrestimación que comete el AIC. Usualmente se define como

$$SBC = -2ln(L) + mln(n),$$

donde n es el número de residuales que pueden calcularse para el modelo estimado, es decir, en el caso de $d > 0$ sería $n = N - d$ con N el número de observaciones de la serie original, y m el número de parámetros libres calculados como se ha explicado en el prólogo anterior. Igual que con el AIC, se debe elegir el modelo con menor SBC.

Capacidad predictiva

A la hora de conseguir un buen modelo lo que más suele interesar es que pueda realizar buenas predicciones fuera de la muestra observada. Sin embargo, al no tener observaciones reales fuera de la muestra no podemos comprobar cuando se equivoca el modelo ajustado al

calcular esas predicciones. Normalmente, solo disponemos de los residuales para las predicciones dentro de la muestra observada, y con ello podemos calcular la suma de cuadrados de los residuos o SSE abreviado. La comparación de los SSE de dos modelos ajustados nos da información de cómo predicen dentro de la muestra, pero no fuera de ella.

Una solución a esto es descartar las últimas observaciones de la muestra para el ajuste del modelo y construir predicciones para estos valores, de tal manera que se puede calcular los residuos de estas últimas observaciones y con ello la suma de cuadrados de los residuos. Este SSE calculado sí nos da información sobre la capacidad del modelo para realizar las predicciones fuera de la muestra utilizada para el ajuste.

2.1.5. Predicción

En la fase de predicción, al igual que en la de estimación, el software utilizado implementa el filtro de Kalman de manera similar para obtener predicciones fuera de la muestra.

También se pueden obtener bandas de confianza para las predicciones, pudiendo así analizar mejor el desarrollo futuro de la serie temporal. Más detalles sobre esto se pueden obtener en [3] y [4].

2.2. Paquetes y funciones de R

El lenguaje de programación R posee por defecto multitud de funciones para análisis estadístico de todo tipo, incluido análisis de series temporales, pero además también existen multitud de paquetes adicionales para complementar y mejorar los análisis que se quieran realizar. Sin embargo, algunas de estas funciones para el análisis de series temporales pueden ser poco eficaces o poco detalladas, y aunque podamos paliar esto con el uso de otras funciones de distintos paquetes, estas siguen teniendo algunas limitaciones. Una descripción de los diversos paquetes y funciones para análisis de series temporales se ha realizado ya previamente, véase [5], por lo que en este trabajo no se profundizará en ello.

Si bien R no es un lenguaje que destaque por su potencia y rendimiento, por lo menos da la posibilidad de usarse como un intermediario entre el usuario y otro lenguajes como C o C++, al permitir que los paquetes contengan código en otros lenguajes que pueden ser ejecutados desde código en R. De esta manera se logra optimizar más las operaciones de análisis estadístico que se quiera realizar. Esto debe tenerse en cuenta para el análisis de grandes cantidades de datos.

Volviendo al análisis de series temporales, se puede destacar que el principal paquete de R para este tema es el paquete ***forecast*** [6], el cual utiliza como base funciones que R tiene ya por defecto para el análisis de series temporales, mejorando y filtrando sus salidas. Como ya hemos comentado la mayoría de paquetes utilizan fragmentos de código en lenguajes como C o C++ para optimizar su ejecución, este también es el caso del paquete *forecast* que utiliza funciones básicas de R que contienen fragmentos de código en C. A pesar de ello tiene

limitaciones, que se notan sobre todo al usar series con gran cantidad de observaciones, o también en el limite de 350 como periodo máximo para ajustar un modelo ARIMA.

La mayoría de estas limitaciones no pueden solucionarse simplemente cambiando algunas líneas de código y pueden llegar a requerir una programación de la función entera desde cero, incluso en algunos casos puede no existir una solución factible.

Capítulo 3

Descripción de las funciones del paquete CBJTSA

En este capítulo se describe en profundidad las distintas funciones que se han diseñado para el análisis de series temporales, nombrando los distintos paquetes implicados en su desarrollo. También se muestran detalles sobre el lenguaje de programación R que pueden ser útiles para la comprensión de estas funciones.

Como ya se ha explicado, el paquete CBJTSA se ha diseñado agrupando las funciones según las distintas etapas de la metodología explicada en el Capítulo 2. Se ha añadido una sección sobre el formato de los datos, que proporciona información sobre la forma de almacenar series temporales en R y funcionalidades adicionales para facilitar esto. Se puede encontrar más información y ejemplos de su uso en la ayuda de R.

Posteriormente en el Capítulo 4 se realizará un ejemplo de análisis de una serie donde se podrán observar las salidas en detalle de cada función del paquete.

3.1. Formato y filtrado de los datos

En el lenguaje de programación R existen distintas formas de almacenaje de los datos. Habitualmente se utilizan objetos de tipo "DataFrame", en el que cada columna representa una variable y cada fila una observación, pero también existen matrices o vectores. Si bien el tipo de objeto de datos que se utilice parece poco relevante, algunas funciones solo aceptan un tipo de objeto como entrada de datos y pueden dar errores si no se les proporciona dicho tipo.

En el caso de las funciones habituales de análisis de series temporales, el objeto que se suele utilizar es de tipo `ts`, véase [7]. Este tipo de dato representa tanto series temporales univariantes como multivariantes junto con el conjunto de fechas o tiempos de cada observación. La forma de crear un objeto de ese tipo requiere de cuatro parámetros:

- `data`: los valores de la serie en cada unidad de tiempo en formato de vector, o matriz si la serie es multivariante. Este parámetro es obligatorio.

- `start`: la fecha de la primera observación de la serie. En este caso el formato es un poco libre, pero en general solo reconoce fechas en meses, trimestres o años, escritos en un vector. Si no se especifica pondrá como primera fecha 1.

- `end`: la fecha de la última observación de la serie. Funciona igual que el parámetro `start`. Si no se especifica puede calcularlo en función del número de observaciones proporcionadas.

- `frequency`: el número de observaciones por unidad de tiempo, que funcionará como periodo del modelo ARIMA cuando se use en funciones de estimación. Si no se especifica se considera 1 como frecuencia.

El principal problema de este tipo de objeto es el parámetro `frequency`, ya que especifica dos cosas totalmente distintas. Por un lado, indica qué tipo de unidad de tiempo usa la serie en conjunto con la fecha de inicio especificada. Por ejemplo, si la serie es mensual y la primera fecha es enero de 2022, la frecuencia debe ser 12 para que agrupe los datos de tal forma que cada doce observaciones cambia de año; si la serie es trimestral y la primera fecha es la misma que antes, la frecuencia debe ser 4 para indicar que cada cuatro observaciones tiene que cambiar de año.

Por otro lado, la frecuencia indica también el periodo de la parte estacional de la serie temporal, es decir que una serie mensual con periodo 12 tendría un periodo anual y una serie trimestral con periodo 4 tendría un periodo anual también.

Esto plantea una limitación, no solo por el tipo de fechas representables, sino también por los periodos de la parte estacional que se pueden especificar. Por ejemplo, una serie mensual se debe representar con frecuencia 12 en la función `ts`, pero si se quisiera plantear un modelo con parte estacional de periodo trimestral en lugar de periodo anual, habría que indicar frecuencia 3.

Es por esto que se ha decidido utilizar como formato de datos una matriz o vector para las observaciones y otro vector para las fechas, ya que es una forma mucho más sencilla y flexible. Sin embargo, algunas funciones seguirán permitiendo la entrada de series temporales en formato `ts`, aunque en algunos casos se requiere de un vector de fechas adicional.

Otro problema a solucionar tiene que ver con el formato de algunos conjuntos de datos, los cuales suelen prescindir de un conjunto de fechas bien escritas para cada observación. Si bien la fecha de cada observación no es un dato determinante durante la estimación de un modelo, es bastante útil en las gráficas que hagamos para entender mejor qué se está representando. Existen ya paquetes de R con algunas funciones que nos permiten crear secuencias de fechas, sin embargo, la mayoría de ellas son poco variadas y se limitan a representar secuencia anual, mensual o diaria.

Teniendo en cuenta todo esto se ha decidido crear una función que permita crear secuencias de fechas que incluya más unidades de tiempo llamada `dateSeq`. También se ha utilizado como tipo de dato para las fechas la cadena de caracteres, ya que es más cómodo y visual a la hora de usarlas en gráficos. A continuación se explicará en detalle la implementación y el uso de esta función.

3.1.1. Función dateSeq

Esta función es un generador de secuencias de fechas para análisis de series temporales. Para su implementación se ha utilizado la función `seq` del paquete base de R, de tal forma que se crea la secuencia de fechas en valor numérico y después se pasan los números a cadena de caracteres con el formato de fecha seleccionado. Los parámetros que utiliza esta función son los siguientes:

- `from`: fecha de inicio, que dependiendo de la unidad de tiempo debe tener un formato específico.
- `to`: fecha final. Puede darse este parámetro en lugar del parámetro `length.out`.
- `by`: la unidad de tiempo con la que se hará la secuencia. Puede ser "min", "hour", "day", "week", "month", "quarter" o "year".
- `length.out`: longitud de la secuencia. Puede darse este parámetro en lugar del parámetro `to`.

Dependiendo del valor del parámetro `by`, los parámetros `from` y `to`, deberán tener un formato reconocible de la fecha, es decir, que para cada unidad de tiempo se usa un tipo de dato de R distinto. Una sentencia de ejemplo de esta función sería la siguiente:

```
dateSeq(from = "2022-01", length.out = 24, by = "month")
```

El resultado será un vector que empiece en "ene. 2022" y termine con "dic. 2023".

3.2. Fase de identificación

La parte fundamental de esta fase son las representaciones gráficas de la serie, por lo que las funciones aquí planteadas utilizan los métodos de R para crear gráficos. En particular, se utilizan el paquete `ggplot2`, véase [8], que permite crear gráficos de forma flexible y muy variada.

El objetivo de esta fase es identificar las partes de la serie temporal y obtener información sobre su comportamiento, para relacionarla con algún modelo teórico ARIMA. Los gráficos típicos de una serie temporal dan información sobre su parte estacional y periodicidad, si la serie tiene tendencia o si los datos requieren de alguna transformación. Sin embargo, hay un tipo de gráfico que ayuda a determinar qué modelo ARIMA específico se ajustaría mejor, la función de autocorrelación y la función de autocorrelación parcial. Como ya se ha explicado encontrar similiaridades entre la función muestral y la teórica de un modelo ARIMA es indicativo de que la serie podría ajustarse correctamente con dicho modelo. Por ello se ha implementado una función que simula modelos ARIMA teóricos y representa gráficamente su ACF y PACF en comparación con las de los datos objeto del análisis.

Antes de explicar las distintas funciones creadas para esta fase es importante describir una función del paquete base de R que será de gran utilidad para esta etapa. Y posteriormente se describe el resto de funciones de este apartado.

3.2.1. Función diff

Función del paquete base de R que genera diferenciaciones de un vector dado, véase [9]. Los parámetros más relevantes que recibe esta función son:

- `x`: un vector o matriz de valores que serán diferenciados. También admite objetos de tipo `ts`.
- `lag`: valor numérico que indica el número de diferenciaciones a realizar. Este valor podemos considerarlo como el grado de diferenciación regular.
- `differences`: valor numérico que indica el orden de diferenciación. En este caso podemos considerarlo como el grado de diferenciación estacional.

Una sentencia adecuada de esta función sería:

```
diff(x, lag = 1, differences = 1)
```

3.2.2. Función tsplot

Crea el gráfico de una serie temporal con sus fechas. Para la implementación de esta función se ha utilizado la función básica del paquete ggplot2 para representar la serie temporal junto con un conjunto de fechas. Los parámetros que utiliza son los siguiente:

- `x`: un vector de fechas en formato de cadena de caracteres que representa el tiempo de cada observación. Puede ser creado con la función `dateSeq` explicada en la Sección 3.1.1.
- `y`:un vector o array con las observaciones de la serie temporal. También puede darse en formato `ts`, pero el vector de fechas sigue siendo necesario.
- `n.unlabel`: el número de observaciones sin etiquetar entre observaciones etiquetadas. Por defecto este parámetro vale 9, para que no se sobrecargue de fechas el eje x.

Con los argumentos elegidos, la sentencia de esta función debería ser de la forma:

```
tsplot(x, y, n.unlabel = 9)
```

3.2.3. Función tsspec

Genera el periodograma de una serie temporal junto con su representación gráfica. Para la implementación de esta función se ha utilizado la función `periodogram` del paquete TSA, véase [10]. Esta función calcula las estimaciones de las frecuencias del periodograma y utiliza estos datos para crear la representación gráfica consecuente. Además del gráfico se devuelve una tabla con los valores de cada frecuencia para poder analizar con facilidad la representación. El parámetro que recibe la función es:

- `y`: un vector con los valores de la serie temporal o un objeto del tipo `ts`.

En este caso la función devuelve una lista con dos elementos que se describen a continuación:

- `table`: un objeto de tipo DataFrame que contiene los valores de las frecuencias del periodograma.
- `plot`: un objeto de ggplot con la representación gráfica del periodograma.

La sentencia de esta función quedaría de la siguiente forma:

```
tsspec(y)
```

3.2.4. Función rmplot

Obtiene el gráfico rango-media de la serie observada. Para ello, separa el conjunto en grupos del tamaño elegido y calcula la media y varianza de cada uno, representándolas en un gráfico de puntos. Si bien se puede elegir el tamaño de los grupos, lo más aconsejado es utilizar el valor del periodo de la parte estacional. En caso de no poder separar los datos en grupos de manera que cada subconjunto tenga el mismo número de observaciones, los grupos estarán desequilibrados y el gráfico resultante puede ser menos fiable. Los parámetros que utiliza la función son:

- `y`: el vector de valores de la serie temporal o un objeto del tipo `ts`.
- `n`: el tamaño que deben tener los subconjuntos sobre los que se calculará la media y varianza. Por defecto tiene valor 5.

La sentencia de esta función quedaría de la siguiente manera:

```
rmplot(y, n = 12)
```

3.2.5. Funciones Acf y Pacf

Además de las funciones ya mencionadas se han importado directamente `Acf` y `Pacf` del paquete forecast, véase [12]. Generan una estimación de la función de autocorrelación y autocorrelación parcial, respectivamente, de la serie temporal especificada, junto con una representación de la misma. Estas funciones son bastante completas en su implementación, por lo que se han importado sin modificaciones al paquete, para facilitar el acceso a ellas. Los parámetros más relevantes de la función son:

- `x`: el vector o matriz de la serie temporal o un objeto del tipo `ts`.
- `lag.max`: el retardo máximo que se estimará de la función.
- `plot`: un objeto de tipo lógico. Si vale `TRUE`, se imprimirán los gráficos durante la ejecución y si vale `FALSE` no.

Esta función devuelve un objeto de tipo `acf`, el cual es una lista con distintos elementos. A continuación se explican los más relevantes:

- `lag`: un array con los valores de los retardos estimados de la función.

- `acf`: un array con los valores estimados de la función de autocorrelación o autocorrelación parcial.

Una sentencia adecuada de estas dos funciones sería:

```
Acf(x, lag.max = 30, plot = TRUE)
Pacf(x, lag.max = 30, plot = TRUE)
```

3.2.6. Función arimaSimComp

Simula un modelo ARIMA y genera representaciones gráficas de la función de autocorrelación y autocorrelación parcial para comparar con las de la serie objeto de estudio. Esta función permite identificar fácilmente qué modelo ARIMA se puede ajustar mejor a la muestra que se tenga, por medio de las gráficas ACF y PACF. Para su implementación se ha utilizado la función `arima.sim()` del paquete base, véase [11], que genera una muestra de un modelo ARIMA simulado dado el valor de los parámetros del mismo. Esta función está pensada para modelos ARIMA simples sin parte estacional, por lo que en la función `arimaSimComp` se ha tratado de generalizar de tal manera que acepte modelos estacionales e incluso modelos ARIMA multiplicativos. En el caso especial de los modelos multiplicativos se ha implementado de tal forma que solo sea necesario dar los valores de los parámetros principales y se calcule automáticamente los valores que son producto de los parámetros principales dados.

Para la estimación de la función de autocorrelación y autocorrelación parcial se han utilizado las funciones `Acf` y `Pacf` ya mencionadas, y se han utilizado distintos métodos del paquete ggplot2 para crear las representaciones gráficas necesarias. La función también da la opción de crear las representaciones del modelo simulado sin una muestra con la que comparar. Los parámetros que recibe esta función son:

- `dates`: vector de fechas en formato de cadena de caracteres que representa el tiempo de cada observación de la muestra. Solo será necesario si se quiere dar una muestra para comparar con la simulación.

- `sample`: vector o matriz con las observaciones de la serie temporal que se usará como muestra en la comparación. También puede darse en formato `ts`.

- `order`: un vector de tres elementos especificando la parte regular del modelo ARIMA a simular, de la forma (p, d, q), donde los componentes son: el orden de la parte autoregresiva (AR), el grado de diferenciación regular y el orden de la parte de media móvil (MA), respectivamente. El grado de diferenciación afectará únicamente a la diferenciación de la muestra que se proporcione.

- `seasonal`: vector especificando la parte estacional del modelo ARIMA a simular, de la forma (P, D, Q), donde los componentes son: el orden de la parte autoregresiva (AR), el grado de diferenciación estacional y el orden de la parte de media móvil (MA), respectivamente. El grado de diferenciación afectará únicamente a la diferenciación de la muestra que se proporcione.

- `n`: el tamaño de la simulación, por defecto es 10000, ya que con ese tamaño se observa en las gráficas del ACF y PACF que se aproxima bien al comportamiento del modelo teórico.
- `period`: el valor del periodo de la parte estacional del modelo ARIMA a simular.
- `ar`: el vector de coeficientes principales de la parte autoregresiva regular del modelo ARIMA a simular. Por ejemplo, `ar = c(0.5, 0.7)` indicaría que el coeficiente del primer retardo de la parte autoregresiva, es 0.5 y del segundo retardo es 0.7.
- `ma`: el vector de coeficientes principales de la parte de media móvil regular del modelo ARIMA a simular. Funciona igual que el parámetro `ar` solo que con la parte MA del modelo.
- `sar`: el vector de coeficientes principales de la parte autoregresiva estacional del modelo ARIMA a simular. Por ejemplo, `sar = c(-0.3, 0.6)` indcaría que el coeficiente del retardo correspondiente al valor del parámetro `period` de la parte autoregresiva, es -0.3 y del retardo correspondiente al doble del valor del parámetro `period`, es 0.6.
- `sma`: el vector de coeficientes principales de la parte de media móvil estacional del modelo ARIMA a simula. Funciona igual que el parámetro `sar` solo que con la parte MA del modelo.
- `lag.max`: el retardo máximo que se estimará en las funciones de autocorrelación y autocorrelación parcial.
- `plot`: un objeto de tipo lógico. Si vale TRUE, se imprimirán los gráficos durante la ejecución y si vale FALSE no.

Esta función devuelve un objeto de tipo `arimaSimComp` que contiene los siguientes elementos:

- `simplot`: una representación gráfica del modelo ARIMA simulado.
- `sampplot`: una gráfico del la serie temporal dada sin ninguna diferenciación realizada.
- `diffsampplot`: una representación gráfica de la serie diferenciada como se ha especificado en los parámetros.
- `acfPlot`: una representación de la función de autocorrelación del modelo simulado junto con el de la muestra en el mismo gráfico si se ha especificado.
- `pacfPlot`: una representación de la función de autocorrelación parcial del modelo simulado junto con el de la muestra en el mismo gráfico si se ha especificado.

Una sentencia correcta de esta función sería:

```
arimaSimComp(dates, sample, order = c(1, 0, 0), seasonal = c(0, 1, 1),
    period = 12, ar = c(0.5), sma = c(-0.6))
```

3.3. Fase de estimación y validación

La fase de estimación se compone de métodos estadísticos más precisos que estimarán los parámetros de los distintos modelos elegidos en la fase anterior, y en la fase de validación se realizarán los test de hipótesis para comprobar la adecuación de esos ajustes. En este paquete se encuentran estas dos fases en el mismo archivo, ya que es la misma función la que realizará ambas etapas al mismo tiempo, `CArima()`.

Como ya se explicó en la Sección 2.1.2, existen diversos métodos de estimación de los parámetros y en el caso de este paquete se utiliza la estimación máxima verosímil mediante el filtro de Kalman. Este método presenta valores de las estimaciones muy similares a los calculados por otro software como SAS, aunque este utilice un método distinto de implementación de la máxima verosímilitud. Sin embargo el método de estimación máximo verosímil mediante filtro de Kalman proporciona una estimación de matriz de varianzas-covarianzas poco precisa, como se indica en [3], por lo que hereda esta imprecisión la matriz de correlaciones entre parámetros.

Los métodos de validación que se implementan aquí son en su mayoría de análisis de los residuales, como se explicó en la Sección 2.1.3. Se proporciona un gran abanico de métodos, tanto contrastes de hipótesis como representaciones gráficas, para poder validar correctamente el ajuste realizado. Además de la función que realiza el ajuste del modelo y calcula todo lo necesario para la fase de validación, se han creado otras funciones ocultas al usuario para poder imprimir correctamente la salida de la función principal. De tal manera que el uso de las funciones `summary()` y `print()` con la salida de la función `CArima` proporcione una visualización adecuada de lo calculado en esta.

3.3.1. Función CArima

Función envoltorio de la función `Arima` del paquete forecast, véase [13], la cual ajusta modelos ARIMA para series univariantes. Esta última a su vez envuelve a la función `arima` del paquete Stats, véase [14], por lo que esta es la función original. Esta última función es más simple en su salida y menos flexible en cuanto al tipo de modelos que puede ajustar, sin embargo, proporciona una base sobre la que trabajar. Para la estimación máxima verosímil utiliza la función `KalmanLike` del paquete Stats, cuyos detalles se pueden consultar en [15].

Si bien la función `Arima` ya incluye algún estadístico y añade más diversidad a los modelos ARIMA que se pueden ajustar, en esta implementación se busca completar su salida para mejorar la toma de decisiones en esta etapa del análisis. En primer lugar, tenemos que hablar de la forma en la que la función `Arima` incluye constante en los modelos, ya que se puede distinguir tres casos, dependiendo de si el modelo tiene alguna diferenciación o no y de qué tipo es dicha diferenciación.

Si se está ajustando un modelo sin diferenciación, tanto en la parte regular como en la estacional, y se quiere incluir la constante en el modelo, se asume que la media de la serie es distinta de cero y se estima su valor en la ejecución de la función. La estimación de la media

se devuelve junto con la estimación del resto de parámetros, pero se puede calcular el valor de la constante a partir de esta si se quiere.

Por el contrario, si estamos ajustando un modelo con una única diferenciación, es decir, $d + D = 1$ donde d es el grado de diferenciación de la parte regular y D el grado de diferenciación de la parte estacional, la función `Arima` calcula lo que denomina "drift". Si esa única diferenciación es regular $d = 1$, significa que la serie original z_t, solo es estacionaria por la parte estacional y, por lo tanto, la serie diferenciada w_t será estacionaria por la parte regular y estacional. Si incluimos la constante en este modelo, la función `Arima` asume que la media de la serie original depende del tiempo t de la forma

$$\mu_t = a + bt, \tag{3.1}$$

donde $E(z_t) = \mu_t$ y, por lo tanto, la media de la serie diferenciada, $\mu_w = \mu_t - \mu_{t-1}$ es distinta de cero. El "drift" que calcula es el término b de la ecuación (3.1), que en este caso es también la media de la serie diferenciada, μ_w.

Por último si la diferenciación que se esta realizando es estacional $D = 1$, tenemos z_t estacionaria por la parte regular y w_t estacionaria por la parte regular y estacional. Al incluir la constante en el modelo, la función `Arima` supone que la media de la serie original depende del tiempo t pero esta vez de la forma

$$\mu_t = a + bt + S_t, \tag{3.2}$$

donde el nuevo término S_t es la componente estacional que cumple $S_t - S_{t-s} = 0$, con s el periodo de la parte estacional de la serie. Por lo tanto, la media de la serie diferenciada, $\mu_w = \mu_t - \mu_{t-s}$, es distinta de cero y además en este caso $\mu_w = sb$, donde de nuevo b es el "drift" estimado en la función.

Además de las funciones ya mencionadas, esta implementación hace uso de otros paquetes y de funciones de este mismo para obtener su salida. Para obtener contraste de hipótesis sobre la estimación de los parámetros del modelo ARIMA$(p, d, q)(P, D, Q)_s$ especificado, se usa la función `coeftest` del paquete lmtest, véase [16]. Esta función usa la t de Student como estadístico y se le proporciona como número de grados de libertad $n - p - q - P - Q - 1$ si el modelo tiene constante y $n - p - q - P - Q$ en caso contrario, donde n es el número de residuales que se pueden estimar.

También para los tests de Ljung-Box se utiliza la función `LB.test` del paquete TSA, véase [17]. Esta función realiza el test de Ljung-Box del retardo especificado para un objeto de la clase `arima`, es decir, la salida de la función `Arima`.

Esta función tiene distintos parámetros, pero a continuación describimos solo los más relevantes. Los detalles sobre el resto de parámetros pueden consultarse en la documentación de la función `Arima`.

- `y`: el vector o matriz de valores de la serie temporal o un objeto del tipo `ts`.

- `order`: un vector especificando la parte regular del modelo ARIMA a estimar: los tres componentes (p, d, q) del vector son el orden de la parte autoregresiva (AR), el grado de diferenciación y el orden de la parte de media móvil (MA), respectivamente.
- `seasonal`: un objeto de tipo lista que contenga:
  - `order`: un vector especificando la parte estacional del modelo ARIMA a estimar, donde los componentes (P, D, Q) del vector son el orden de la parte autoregresiva (AR), el grado de diferenciación y el orden de la parte de media móvil (MA), respectivamente.
  - `period`: el valor del periodo de la parte estacional del modelo ARIMA, en el caso de dar un objeto de tipo `ts` en el parámetro `y`, el periodo se tomará como el valor de `frequency` de dicho objeto y este parámetro se podría omitir.
- `plot`: un objeto de tipo lógico. Si vale `TRUE`, se imprimirán los gráficos durante la ejecución y si vale `FALSE` no.
- `include.constant`: un objeto de tipo lógico. Si vale `TRUE`, añade la constante al ajuste, de tal manera que para modelos sin diferenciar se estimará la media de la serie y para modelos diferenciados se estima la media de la serie diferenciada. En caso de que se elija un modelo con $d + D > 1$ no se añadirá la constante, por lo que no se estimará ninguna media. Si vale `FALSE` ajustará el modelo asumiendo que la constante es cero.

Puesto que la función CArima pone el foco en la media de la serie diferenciada, los parámetros `include.mean` e `include.drift` de la función `Arima` se han eliminado en esta implementación para simplificar. Esta función devuelve un objeto de tipo `CArima` que, además de lo que contenía la salida de la función `Arima`, está formado por los siguientes elementos:

- `SBC`: el valor del estadístico SBC calculado como se explica en la Sección 2.1.4.
- `cor.coef`: matriz de correlaciones de los parámetros del modelo.
- `resid.acf`: objeto de tipo `acf` que contiene la función de autocorrelación de los residuales del modelo.
- `resid.pacf`: objeto de tipo `acf` que contiene la función de autocorrelación parcial de los residuales del modelo.
- `lbtests.df`: objeto de tipo DataFrame con los valore del test de Ljung-Box para los retardos múltiplos de seis hasta el retardo con valor $Ent\left(\frac{N}{4}\right) + s$, siendo N el número de observaciones de la serie, s el periodo de la parte estacional y el operador $Ent(x)$ el calculo de la parte entera de x.
- `lbtests.plot`: gráfico de los p-valores de los test de Ljung-Box calculados.
- `residRM`: gráfico rango-media de los residuales estimados del modelo.
- `residQQp`: gráfico cuantil-cuantil de los residuales estimados del modelo.

- `residDensity`: histograma de los residuales estimados junto con la representación de su función de densidad estimada y la función de densidad de una distribución Normal.
- `residMuTest`: contraste de hipótesis de la t de Student para la media de los residuales estimados.
- `residShapiro`: test de normalidad de Shapiro-Wilks para los residuales estimados.

Una sentencia adecuada de esta función para ajustar un modelo ARIMA$(1,0,0)(0,1,1)_{12}$ con constante sería:

```
CArima(y, order = c(1,0,0), seasonal = list(order = c(0,1,1), period = 12),
    include.constant = TRUE)
```

3.4. Fase de comparación y predicción

En este apartado tratamos las funciones implementadas para las fases de comparación y predicción. Estas se encuentran en el mismo fichero ya que todas ellas utilizan la función `forecast` del paquete forecast, véase [18], durante su ejecución.

Para la fase de comparación se ha creado una función que ayuda a analizar la capacidad predictiva de un modelo ARIMA ya estimado, como se explica en la Sección 2.1.4. Como ya se ha comentado este paquete hace uso del filtro de Kalman también para hacer predicciones fuera de la muestra, por lo que no es extraño que se obtengan distintos resultados en esta fase con respecto a otros softwares como SAS, que no utiliza este método de predicción. También se ha implementado una función que permite comparar distintos modelos de forma cómoda, creando tablas resumen de los valores más relevantes de cada modelo para la comparación de los mismos.

Por último, para la fase de predicción se ha creado una función que realiza predicciones de un modelo ARIMA ya estimado, junto con algunas representaciones gráficas para estas. Además, se han creado otras funciones ocultas al usuario para que el uso de las funciones `summary` y `print` con la salida de las funciones principales de este apartado proporcionen una visualización adecuada de sus salidas.

3.4.1. Función predCap

Realiza un análisis más completo de la capacidad predictiva de un modelo ARIMA estimado con la función `CArima`. Como ya se ha comentado, esta función realiza lo explicado en la Sección 2.1.4, de tal manera que se especifica el número de observaciones que se extraen del final de la serie temporal para usarlos como conjunto de prueba, y se ajusta de nuevo el modelo con las observaciones restantes como conjunto de entrenamiento. Sobre ese ajuste se estiman predicciones fuera de la muestra con la función `forecast` que devolverá las predicciones junto con bandas de confianzas a nivel 0.95 y 0.8 de cada observación.

Con el conjunto de prueba se calculan los residuales a partir de las predicciones estimadas y se calcula la suma de cuadrados de los residuales (SSE) de la predicción. Además, se representa gráficamente la estimación de las predicciones junto con las bandas de confianza y la evolución del tamaño de las bandas de confianza de nivel 0.95. Los parámetros que requiere esta función son:

- `object`: un vector o matriz con las observaciones de la serie temporal que se ha usado para el modelo. También puede darse en formato `ts`.
- `h`: el número de observaciones que se extraerán del final de la muestra para usarlos como conjunto de prueba. Si el modelo tiene parte estacional es adecuado usar un múltiplo del periodo como número de observaciones del conjunto de prueba.
- `model`: el modelo ARIMA estimado como salida de la función `CArima` de este paquete.
- `dates`: un vector de fechas en formato de cadena de caracteres que representa el tiempo de cada observación de la muestra.
- `plot`: un objeto de tipo lógico. Si vale TRUE, se imprimirán los gráficos durante la ejecución y si vale FALSE no.

Esta función devuelve una lista con los siguientes elementos:

- `forecasts`: un DataFrame con las `h` predicciones estimadas a partir del modelo ajustado con el conjunto de entrenamiento, junto con bandas de confianza para las predicciones a nivel 0.95 y 0.8.
- `SSEtable`: un DataFrame que contiene: las predicciones, el valor real del conjunto de prueba, los valores residuales calculados a partir del conjunto de prueba y la suma de cuadrados de los residuales en cada predicción.
- `forecPlot`: una representación gráfica de las predicciones junto con las bandas de confianza.
- `bandWidth`: un vector con el valor de la anchura de la banda de confianza a nivel 0.95 en cada predicción.
- `bandWidthPlot`: una representación gráfica del elemento `bandWidth`.

Una sentencia adecuada de esta función sería:

```
predCap(object, h = 12, model, dates, plot = T)
```

3.4.2. Función compareARIMA

Resume distintos estadísticos y gráficos para comparar hasta tres modelos ARIMA estimados con la función `CArima`. Esta función recoge la salida de distintas funciones de este paquete, ejecutadas sobre los distintos modelos de la misma serie temporal que se quiere comparar, para mostrarlos de forma que sea más cómodo realizar la valoración de qué modelo es

más adecuado. Utiliza la salida de la función `predCap` para obtener métricas sobre la capacidad predictiva de cada modelo y también extrae de los modelos estadísticas de comparación como el AIC y el SBC.

El interés principal de esta función es agrupar las distintas métricas calculadas durante el análisis de cada modelo para realizar la fase de comparación de manera adecuada. Cabe destacar que los modelos deberían tener el mismo grado de diferenciación, ya que comparar modelos con distintos grado de diferenciación no es adecuado. Los parámetros que requiere son los siguientes:

- `object`: un vector o matriz con las observaciones de la serie temporal que se ha usado para el modelo. También puede darse en formato `ts`.
- `dates`: un vector de fechas en formato de cadena de caracteres que representa el tiempo de cada observación de la muestra.
- `m1, m2 y m3`: los modelos ARIMA estimados como salida de la función `CArima` que se quiere comparar.
- `h`: el número de observaciones que se extraerán del final de la muestra para usarlos como conjunto de prueba. Este parámetro es el que se le proporcionará a la función `predCap` para obtener su salida.
- `plot`: un objeto de tipo lógico. Si vale TRUE, se imprimirán los gráficos durante la ejecución y si vale FALSE no.

Esta función devuelve como salida un objeto de tipo `compareARIMA` que contiene los siguientes elementos:

- `metrics`: DataFrame con algunas métricas de comparación de cada modelo, entre las que se incluyen: criterio de información de Akaike (AIC), varianza del model, criterio Bayesiano de Schwarz (SBC), suma de cuadrados de los residuales de la predicción (SSEpred) y dos retardos del test de Ljung-Box.
- `SSEtable`: un DataFrame resultante de combinar el elemento `SSEtable` de la salida de la función `predCap` de cada modelo especificado. Para poder comparar los residuales, predicciones y SSE de cada modelo calculado.
- `forecasts`: un DataFrame resultante de combinar el elemento `forecasts` de la salida de la función `predCap` de cada modelo especificado.
- `forecPlot1, forecPlot2, forecPlot3`: representaciones gráficas de las predicciones y bandas de confianza calculadas para cada modelo mediante la función `predCap`.
- `bandWidthComp`: un DataFrame con las anchuras de las bandas de confianza de predicción de cada modelo, resultado de la función `predCap`.
- `bandWidthPlot`: una representación gráfica de la evolución de las anchuras de las bandas de confianza de predicción de cada modelo.

Una sentencia correcta de esta función sería:

```
compareARIMA(object, dates, m1, m2, m3, h = 12, plot = T)
```

3.4.3. Función Cforecast

Genera predicciones fuera de la muestra de un modelo ARIMA estimado mediante la función `CArima`. `Cforecast` es una función envoltorio de la función `forecast` del paquete forecast, véase [18], la cual al recibir un modelo ARIMA ajustado llama a la función `predict.Arima` del paquete stats, véase [19]. Esta última usa la función `KalmanForecast` del paquete stats, véase [15], para estimar las predicciones usando el filtro de Kalman a partir del la estimación máxima verosímil que ya se ha hecho para estimar el modelo.

A partir de las predicciones que genera la función `forecast` se crea la salida de la función `Cforecast`, añadiendo distintos gráficos útiles para el análisis de las predicciones calculadas. Además, se han implementado funciones ocultas al usuario para que la salida de esta función sea compatible con las funciones `summary` y `print`, de tal manera que proporcionen una visualización adecuada de lo calculado. A continuación explicamos los parámetros más relevantes de esta función, los detalles sobre el resto de parámetros puede consultarse en la documentación de la función `forecast`.

- `object`: un vector o matriz con las observaciones de la serie temporal que se ha usado para el modelo. También puede darse en formato `ts`.
- `h`: el número de predicciones fuera de la muestra que se quieren obtener.
- `model`: un modelo ARIMA estimado como salida de la función `CArima` del que se quieren obtener predicciones.
- `plot`: un objeto de tipo lógico. Si vale TRUE, se imprimirán los gráficos durante la ejecución y si vale FALSE no.
- `dates`: un vector de fechas en formato de cadena de caracteres que representa el tiempo de cada observación de la muestra.

Esta función devuelve un objeto de tipo `Cforecast` que, además de lo que contenía la salida de la función `forecast`, está formado por los siguientes elementos:

- `forecasts`: un DataFrame con el valor de las predicciones estimadas y bandas de confianza a nivel 0.95 y 0.8 para cada una.
- `realForecPlot`: una representación gráfica de la serie original, extendiéndola por la derecha con el valor de las predicciones estimadas y las bandas de confianza.
- `realFitPlot`: un gráfico de la serie original y los valores ajustado junto con las predicciones fuera de la muestra en comparación.

Una sentencia adecuada de esta función sería:

```
Cforecast(object, h = 24, model, plot = T, dates)
```

Capítulo 4

Aplicación práctica

En este capítulo se analiza una serie temporal sobre el consumo del gas en España siguiendo las etapas explicadas en el Capítulo 2, utilizando el paquete de R CBJTSA, cuyas funciones se han explicado ya en el Capítulo 3. Para mayor comodidad, se ha creado un repositorio público en *GitHub* que contiene el código interno del paquete (véase [20]), de tal manera que se pueda acceder a ella e instalarla remotamente en R.

En las primeras Secciones se hablará del origen de los datos y la forma de instalar el paquete desde GitHub. Posteriormente se realizará el análisis siguiendo el proceso iterativo descrito en la Sección 2.1, mostrando en cada etapa las distintas salidas que aporta R.

4.1. Origen de los datos

Se ha tratado de buscar una serie temporal que se pueda ajustar mediante la metodología Box-Jenkins, para así poder mostrar la utilidad del paquete desarrollado. En este sentido, se podría haber elegido una serie sencilla que se use de ejemplo de forma habitual, pero se ha optado por buscar una serie temporal que tenga un interés actual.

La serie temporal elegida refleja el consumo de gas natural en España en los últimos dieciocho años. Estos datos han sido extraídos de la página web *EpData* [21], la cual proporciona estadísticas y conjuntos de datos muy variados. El consumo del gas viene medido de forma mensual en GWh (giga-vatios la hora) desde Enero de 2004 hasta Marzo del año 2022.

El objetivo es encontrar un patrón en el consumo del gas natural. Teniendo en cuenta que el conflicto bélico actual en Ucrania está dañando la economía de toda Europa y que Rusia ya ha cortado el suministro de gas en algunos países, este estudio puede ser interesante. Es por esto que se tratará de modelizar las necesidades de gas natural mensuales de España a partir de estos datos, como ejemplo del uso del paquete.

4.2. Instalación del paquete y preprocesado de los datos

Como ya se ha comentado, el paquete está disponible en un repositorio público de GitHub, por lo que es posible instalarla de forma remota. Instalar un paquete de esta manera es especialmente cómodo ya que todos los que tenga este como dependencias se instalarán también, si no lo estaban ya.

Para realizar la instalación se utilizará la función `install_github` del paquete remotes. Una vez importado este, la sentencia que se debe ejecutar es la siguiente:

```
install_github("danipequelangos/CBJTSA")
```

A partir de ese punto se podrán utilizar todas las funciones disponibles de CBJTSA.

Los datos descargados vienen en formato `.csv`, por lo que se puede utilizar la función `read.csv2` para leer el fichero desde R. Dado que los datos comienzan en Enero y acaban en Marzo se ha decidido prescindir de las tres primeras observaciones, correspondientes a los tres primeros meses del año 2004, para así tener exactamente dieciocho años completos en la muestra. Antes de empezar con el análisis es necesario crear un vector de fechas de la serie temporal. Para ello se puede hacer uso de la función `dateSeq` del paquete CBJTSA (explicada en la Sección 3.1.1). Se puede encontrar en el Anexo 7.1 el código en R utilizado para todo este capítulo.

4.3. Fase de Identificación

Como ya se ha explicado anteriormente, en esta fase se trata de extraer información general sobre el comportamiento de la serie. Se empieza analizando el gráfico de la serie temporal, creado mediante la función `tsplot` (explicada en la Sección 3.2.2).

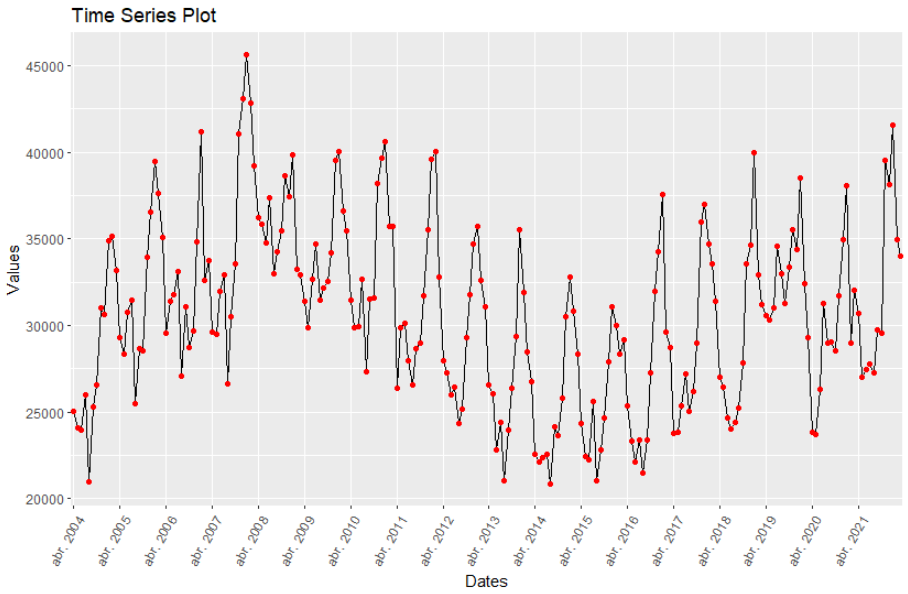


Figura 4.1: Gráfico de la serie temporal

En la Figura 4.1 se puede intuir un comportamiento estacional de periodo anual, principalmente en los primero años. Además, no se observa una tendencia clara en la serie a primera vista, pero sí se observa un comportamiento cíclico. Antes de seguir analizando el comportamiento de la serie conviene comprobar si la varianza puede ser dependiente del tiempo, ya que en ese caso la serie puede requerir algún tipo de transformación. Para comprobar esto se representa la media y la varianza de subconjuntos de la muestra en un gráfico mediante la función `rmplot` (explicada en la Sección 3.2.4).

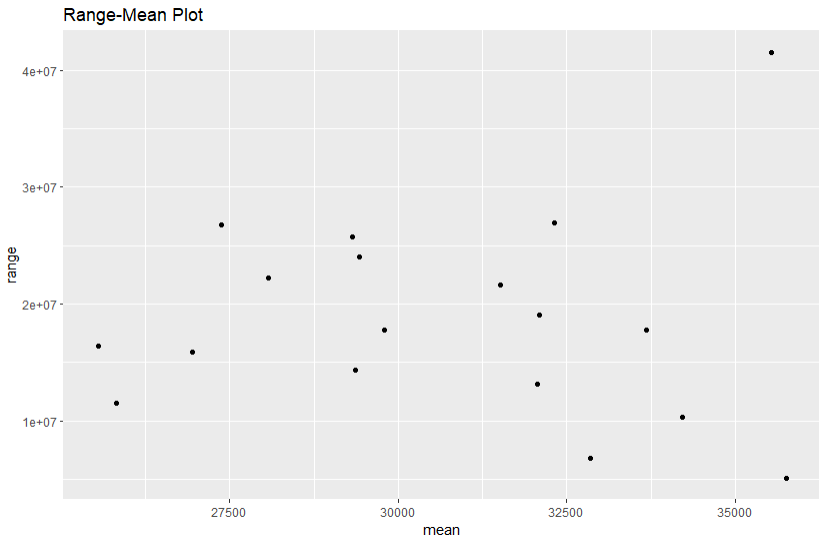


Figura 4.2: Gráfico rango-media

A juzgar por el gráfico de la Figura 4.2, la varianza de la serie no parece ser dependiente del tiempo, ya que en la nube de puntos no se observa una relación entre la media y la varianza. Por lo tanto, la muestra no parece necesitar ningún tipo de transformación. Continuando con el análisis, se puede representar la función de autocorrelación de la muestra para obtener más información. Para ello se utiliza la función `Acf` (explicada en la Sección 3.2.5), importada al paquete CBJTSA desde el forecast.

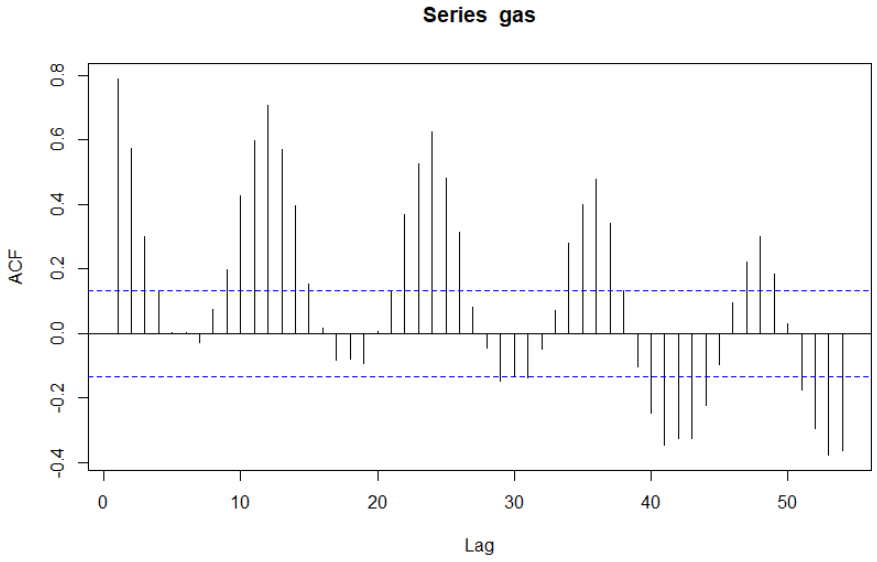


Figura 4.3: Gráfico de la función de autocorrelación muestral

El gráfico del ACF muestral (Figura 4.3) muestra un descenso lento de la función en los retardos 12, 24, 36 y 48; esto indica que la serie no es estacionaria por la parte estacional, y también que el periodo de la serie es 12, es decir, periodo anual. Sin embargo, por la parte regular, la función parece descender más rápidamente en los primeros retardos de la serie, pero no está claro si la serie es estacionaria por la parte regular.

Como comprobación adicional se puede representar el periodograma de la serie para contrastar la información obtenida hasta ahora. Para ello se puede utilizar la función `tsspec` (explicada en la Sección 3.2.3), la cual proporciona una tabla con los valores del periodograma, además de su representación.

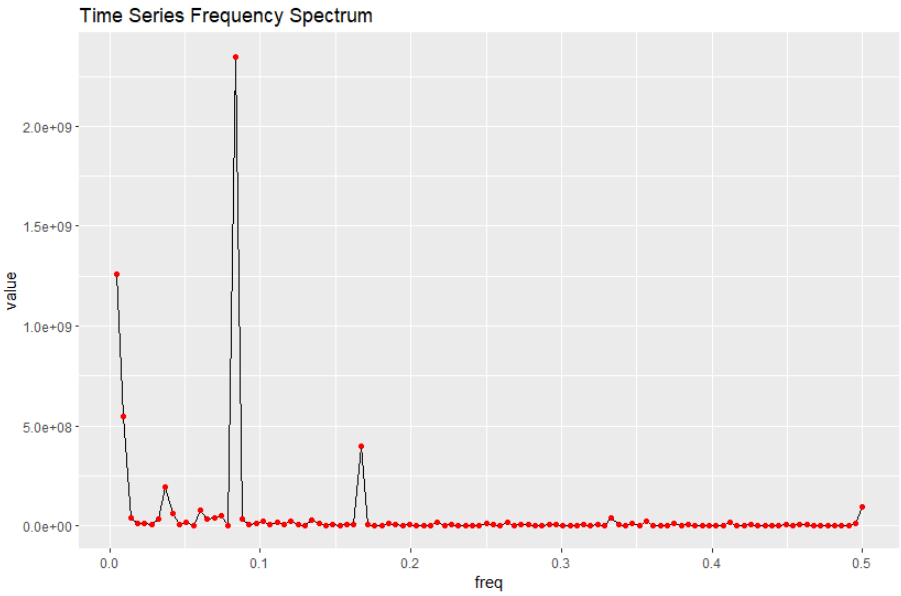


Figura 4.4: Periodograma de la muestra

Fila	Frecuencia	Valor
1	0.004629630	1.258609e+09
2	0.009259259	5.451198e+08
3	0.013888889	4.079945e+07
	...	...
18	0.083333333	2.349179e+09
	...	...
36	0.166666667	3.958746e+08

Tabla 4.1: Valores del periodograma de la muestra

Como se observa en la Figura 4.4, es posible que la serie tenga algo de tendencia ya que los dos primeros valores del periodograma destacan un poco sobre el resto. En la Tabla 4.1 se indican los valores del periodograma, donde Fila hace referencia al índice de la observación, Frecuencia hace referencia al valor del eje x y Valor hace referencia al valor del eje y. En esta tabla se observa que destacan los valores correspondientes a las frecuencias $\frac{1}{12}$ y $\frac{2}{12}$, lo que indica que el periodo de la parte estacional es 12, como ya se había asumido.

Concluimos entonces que la serie no parece ser estacionaria por la parte regular, ni tampoco por la parte estacional, por lo tanto, antes de poder ajustar un modelo se debe convertir la serie en estacionaria. Para ello, se analizarán las diferenciaciones de la serie original, para comprobar cuántas diferenciaciones son necesarias para obtener una serie estacionaria. La diferenciación de una serie se puede obtener mediante la función `diff` del paquete base de R (explicada en la Sección 3.2.1).

4.3.1. Serie diferenciada regularmente

Inicialmente se analiza la primera diferenciación regular de la serie original, para ello se muestra el gráfico de la función de autocorrelación.

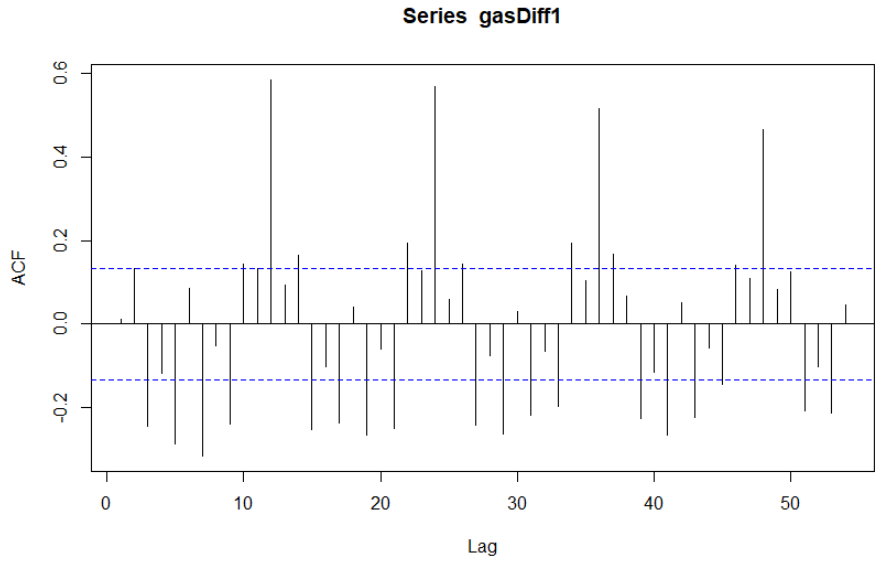


Figura 4.5: Gráfico de la función de autocorrelación de la diferenciación regular

En la Figura 4.5 se observa que la diferenciación ha eliminado en gran parte la tendencia de la serie original, por lo que se puede considerar estacionaria por la parte regular. Sin embargo, por la parte estacional se sigue viendo un descenso lento de la función, lo que indica que no es estacionaria por la parte estacional. Esto es indicativo suficiente para considerar una diferenciación estacional.

4.3.2. Serie diferenciada regular y estacionalmente

Se realiza una diferenciación estacional de orden 12, ya que es el valor del periodo que se ha asumido durante el análisis anterior. Además, esta diferenciación se realiza sobre la anterior, ya que una sola diferenciación estacional tendría mayor varianza (ver Tabla 4.2) y la función de autocorrelación de esta diferenciada decrecería lentamente en los primeros retardos. A continuación, se representa la función de autocorrelación de la serie diferenciada regular y estacionalmente.

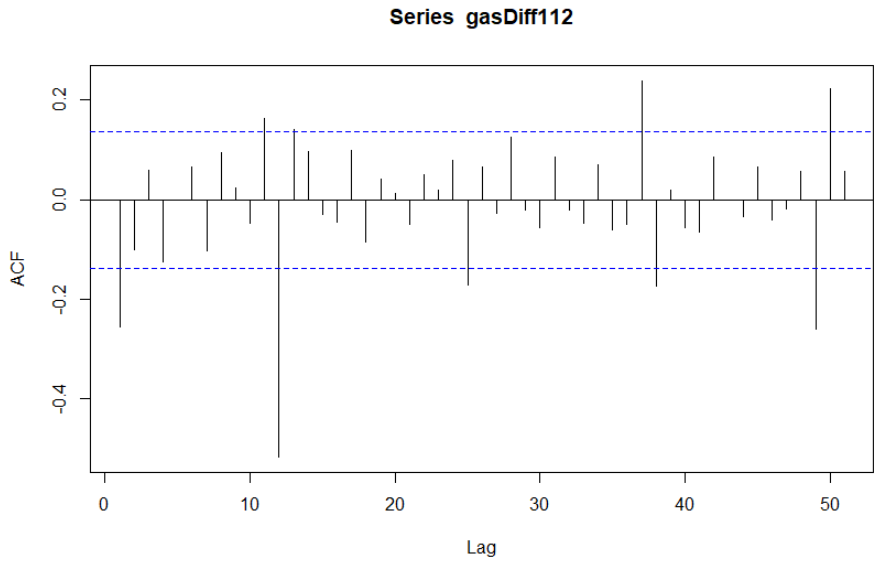


Figura 4.6: Gráfico de la función de autocorrelación de la diferenciación regular y estacional

La representación del ACF (Figura 4.6) muestra un comportamiento que puede considerarse estacionario, tanto en la parte regular como en la parte estacional, por lo que ahora sí se puede asegurar que la serie es estacionaria por ambas partes.

Habiendo obtenido ya una serie estacionaria, se debe comprobar si las diferenciaciones realizadas no han aumentado la varianza de la muestra, ya que en ese caso puede que alguna de las dos diferenciaciones realizada no sea necesaria.

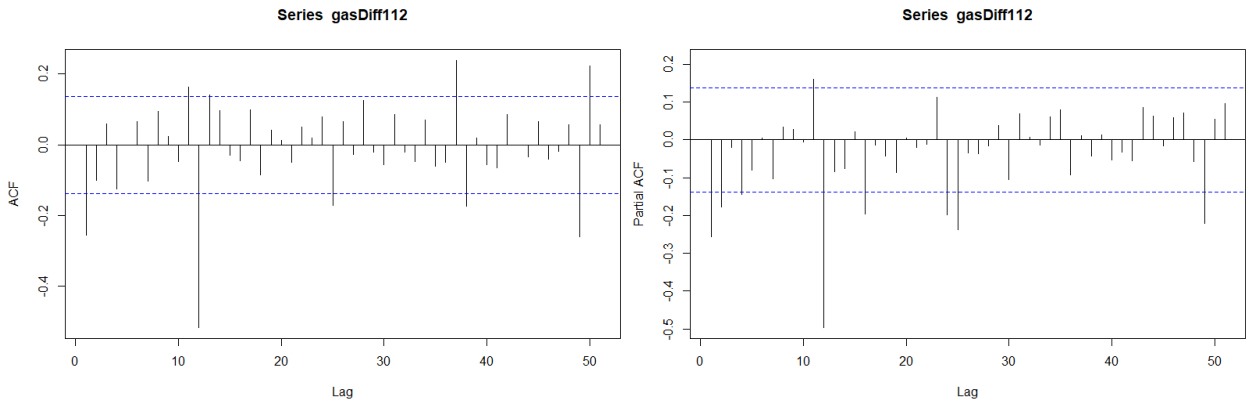
Serie original	26767090
Diferenciación regular	11087627
Diferenciación estacional	13336921
Diferenciación regular y estacional	8319175

Tabla 4.2: Varianza de las series

Como se observa en la Tabla 4.2, la varianza ha disminuido con cada diferenciación realizada, por lo que se podría decir que sí eran necesarias.

4.3.3. Modelos candidatos

Finalmente, se utiliza la diferenciación regular y estacional para ajustar el modelo ARIMA a la serie, por lo que se representa a continuación la función de autocorrelación y la función de autocorrelación parcial de la diferenciación elegida (Figura 4.7).



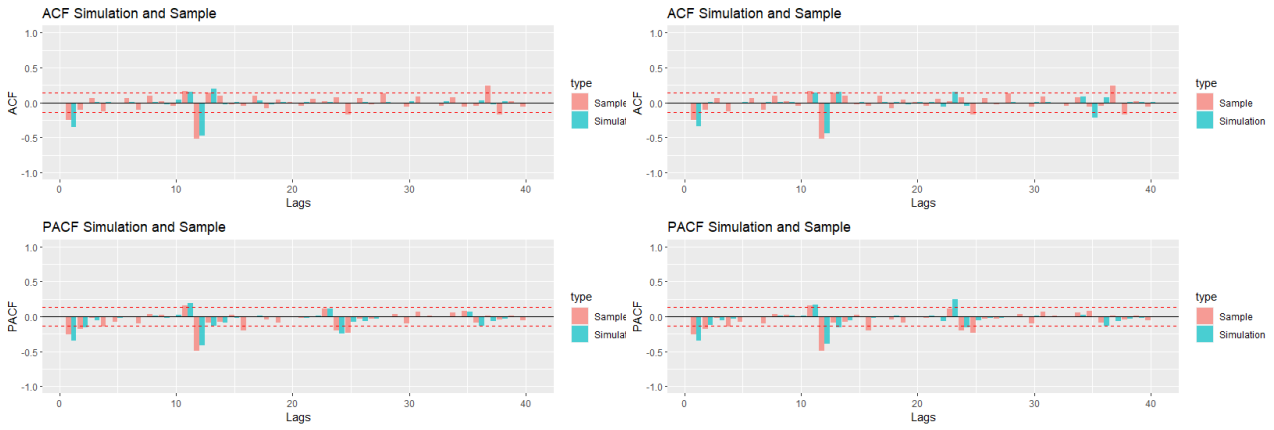
(a) ACF (b) PACF

Figura 4.7: ACF y PACF de la diferenciación regular y estacional

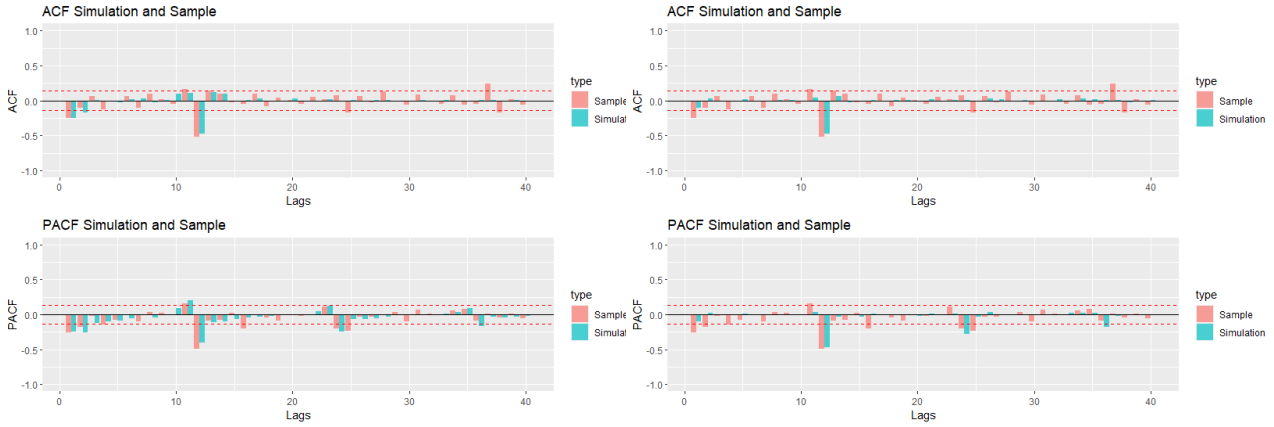
A la vista del ACF y PACF de la serie diferenciada, se plantean algunos modelos ARIMA:

- ARIMA$(0,1,1)(0,1,1)_{12}$
- ARIMA$(0,1,1)(0,1,2)_{12}$
- ARIMA$(0,1,2)(0,1,1)_{12}$
- ARIMA$(1,1,1)(0,1,1)_{12}$
- ARIMA$(0,1,1)(1,1,0)_{12}$
- ARIMA$(1,1,0)(0,1,1)_{12}$

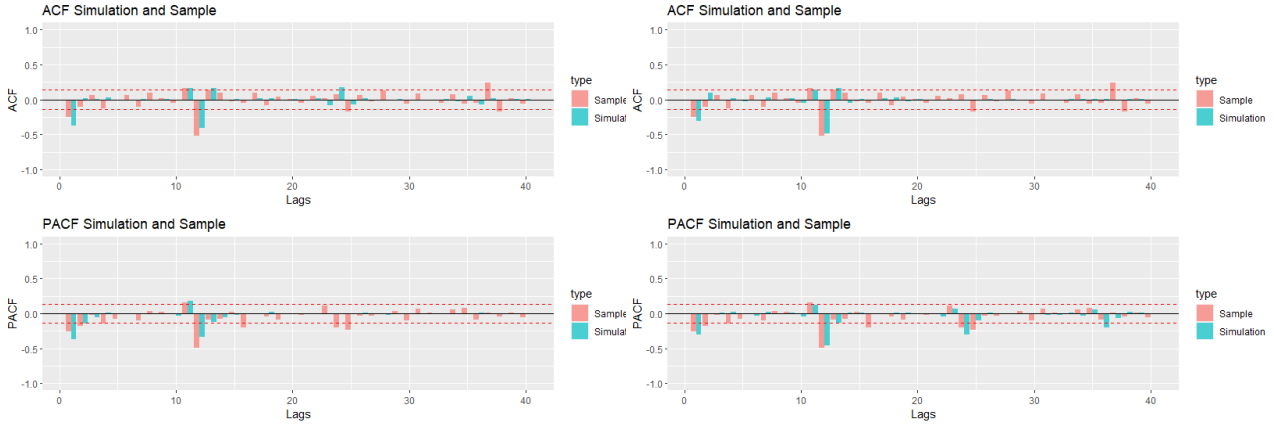
Para comprobar si los modelos planteados son candidatos adecuados, se utiliza la función `arimaSimComp` (explicada en la Sección 3.2.6), la cual compara las funciones de autocorrelación y autocorrelación parcial de la muestra diferenciada con las funciones de una simulación de 10000 observaciones del modelo indicado. En la Figura 4.8 se muestra la salida para cada modelo planteado. Se observa que el ACF y PACF de la serie diferenciada se parecen al de los modelos simulados. Por lo tanto, se continuará con la fase de estimación de estos modelos planteados.



(a) $\text{ARMA}(0,1)(0,1)_{12}$ (b) $\text{ARMA}(0,1)(0,2)_{12}$



(c) $\text{ARMA}(0,2)(0,1)_{12}$ (d) $\text{ARMA}(1,1)(0,1)_{12}$



(e) $\text{ARMA}(0,1)(1,0)_{12}$ (f) $\text{ARMA}(1,0)(0,1)_{12}$

Figura 4.8: Comparación del ACF y PACF de la simulación con el de la serie diferenciada

4.4. Fase de Estimación

En esta sección se muestra la estimación de los parámetros de los distintos modelos planteados, y se comprueba si sobran parámetros, a partir de los resultados obtenidos mediante la función `CArima` (explicada en la Sección 3.3.1). Para ello, se comprueba si los parámetros son significativos, y si la matriz de correlaciones de los mismos no contiene valores demasiado altos.El resto de elementos de la salida se tendrán en cuenta en las siguientes etapas del análisis.

En cada uno de los modelos se hace uso de la función `print` junto con la salida de la función `CArima` para mostrar únicamente la información necesaria para esta fase.

$\mathbf{ARIMA}(0,1,1)(0,1,1)_{12}$

```
Series: gas
ARIMA(0,1,1)(0,1,1)[12]

t test of coefficients:

      Estimate Std. Error  t value  Pr(>|t|)
ma1  -0.378135   0.074542  -5.0728  8.49e-07 ***
sma1 -0.818607   0.058218 -14.0610 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

sigma^2 = 4275860:  log likelihood = -1843.52
AIC=3693.04   BIC=3702.97   SBC=3730.04

Correlations of the estimated parameters:
            ma1       sma1
ma1  1.00000000 0.04830412
sma1 0.04830412 1.00000000
```

Como se observa en la salida de texto obtenida con R, todos los parámetros de este modelo son significativos a nivel 0.001. Además, la matriz de correlaciones de los parámetros no muestra ninguna correlación especialmente alta, por lo que el modelo puede pasar a la siguiente fase del análisis.

ARIMA$(0,1,1)(0,1,2)$

```
Series: gas
ARIMA(0,1,1)(0,1,2)[12]

t test of coefficients:

      Estimate Std. Error  t value  Pr(>|t|)
ma1  -0.376777   0.074737  -5.0414 9.865e-07 ***
sma1 -0.880369   0.074260 -11.8552 < 2.2e-16 ***
sma2  0.111043   0.082410   1.3475    0.1793
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

sigma^2 = 4283162:  log likelihood = -1842.61
AIC=3693.23   BIC=3706.48   SBC=3749.73

Correlations of the estimated parameters:
            ma1        sma1        sma2
ma1  1.00000000  0.03405778  0.01478772
sma1 0.03405778  1.00000000 -0.62951669
sma2 0.01478772 -0.62951669  1.00000000
```

A diferencia del modelo anterior se observa en este caso un parámetro no significativo, el parámetro al que R llama `sma2` es el correspondiente al segundo retardo de la parte estacional de media móvil, es decir el retardo de orden 24. Además, se observa una correlación más alta de lo habitual entre este parámetro y el parámetro `sma1` (correspondiente al primer retardo de la parte estacional de media móvil). Por ello descartamos el modelo.

ARIMA$(0,1,2)(0,1,1)$

```
Series: gas
ARIMA(0,1,2)(0,1,1)[12]

t test of coefficients:

      Estimate Std. Error  t value  Pr(>|t|)
ma1  -0.353254   0.070529  -5.0086 1.149e-06 ***
ma2  -0.121627   0.082620  -1.4721    0.1425
sma1 -0.808431   0.060276 -13.4120 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

sigma^2 = 4263369:  log likelihood = -1842.45
AIC=3692.9   BIC=3706.15   SBC=3749.4
```

```
Correlations of the estimated parameters:
             ma1        ma2        sma1
ma1   1.00000000 -0.2667871  0.09225081
ma2  -0.26678713  1.0000000 -0.11773356
sma1  0.09225081 -0.1177336  1.00000000
```

En la salida de R se puede observar que el parámetro `ma2`, correspondiente al segundo retardo de la parte de media móvil regular, tiene un p-valor mayor que 0.05, por lo que no es significativo para el ajuste. Dado que este parámetro añadido no es significativo para el ajuste descartamos el modelo en esta fase.

$\mathbf{ARIMA}(1,1,1)(0,1,1)_{12}$

```
Series: gas
ARIMA(1,1,1)(0,1,1)[12]

t test of coefficients:

      Estimate Std. Error  t value  Pr(>|t|)
ar1   0.443332   0.164530   2.6945  0.007611 **
ma1  -0.776940   0.125292  -6.2010 2.872e-09 ***
sma1 -0.806430   0.061292 -13.1573 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

sigma^2 = 4231051:  log likelihood = -1841.71
AIC=3691.41   BIC=3704.67   SBC=3747.92

Correlations of the estimated parameters:
            ar1         ma1        sma1
ar1   1.0000000 -0.92373933  0.11201301
ma1  -0.9237393  1.00000000 -0.09930768
sma1  0.1120130 -0.09930768  1.00000000
```

A juzgar por la salida de R, todos los parámetros del modelo ajustado parecen ser significativos a nivel 0.01. Sin embargo, una de las correlaciones entre los parámetros es especialmente alta, la correlación entre el parámetro `ma1` (correspondiente al primer retardo de la parte regular de media móvil) y el parámetro `ar1` (correspondiente al primer retardo de la parte regular autoregresiva). Esto nos indica que al modelo le sobran parámetros, por lo que se descarta en esta fase.

$\mathbf{ARIMA}(0,1,1)(1,1,0)$

```
Series: gas
ARIMA(0,1,1)(1,1,0)[12]

t test of coefficients:

      Estimate Std. Error t value  Pr(>|t|)
ma1  -0.344399   0.081468 -4.2274 3.501e-05 ***
sar1 -0.538971   0.059748 -9.0207 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

sigma^2 = 5396264:  log likelihood = -1862.53
AIC=3731.07   BIC=3741.01   SBC=3768.07

Correlations of the estimated parameters:
             ma1        sar1
ma1   1.00000000 -0.01993018
sar1 -0.01993018  1.00000000
```

En este caso se observa un modelo con todos los parámetros significativo a nivel 0.001, y ninguna de las correlaciones entre los parámetros tiene un valor alto. Por lo tanto, este modelo pasará a la siguiente fase del análisis.

$\mathbf{ARIMA}(1,1,0)(0,1,1)_{12}$

```
Series: gas
ARIMA(1,1,0)(0,1,1)[12]

t test of coefficients:

      Estimate Std. Error  t value  Pr(>|t|)
ar1  -0.283793   0.067431  -4.2086 3.781e-05 ***
sma1 -0.820131   0.057640 -14.2286 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

sigma^2 = 4396329:  log likelihood = -1846.35
AIC=3698.7   BIC=3708.64   SBC=3735.7

Correlations of the estimated parameters:
          ar1     sma1
ar1  1.000000 0.096354
sma1 0.096354 1.000000
```

Se puede observar en la salida de R que todos los parámetros de este ajuste son significativos a nivel 0.001. Además la correlación entre ambos parámetros del modelo es bastante baja, por lo que este modelo pasará a la siguiente fase del análisis.

4.5. Fase de Validación

En la fase anterior se descartaron tres modelos por sobreparametrización, de tal manera que los modelos que se analizarán en este fase son:

- Modelo 1: ARIMA$(0,1,1)(0,1,1)_{12}$
- Modelo 2: ARIMA$(0,1,1)(1,1,0)_{12}$
- Modelo 3: ARIMA$(1,1,0)(0,1,1)_{12}$

En esta fase se hace uso del resto de elementos de la salida de la función `CArima`, para comprobar si los modelos ajustados son válidos. Igual que en la sección anterior se pasa por los tres modelos seleccionados uno a uno, y se hace uso de la función `summary` junto con la salida de la función `CArima` para obtener la información necesaria para esta etapa del análisis.

Modelo 1

Lo primero que se analiza en este modelo son las funciones de autocorrelación y autocorrelación parcial de los residuales, para comprobar si se comportan como variables aleatorias incorreladas (Figura 4.9).

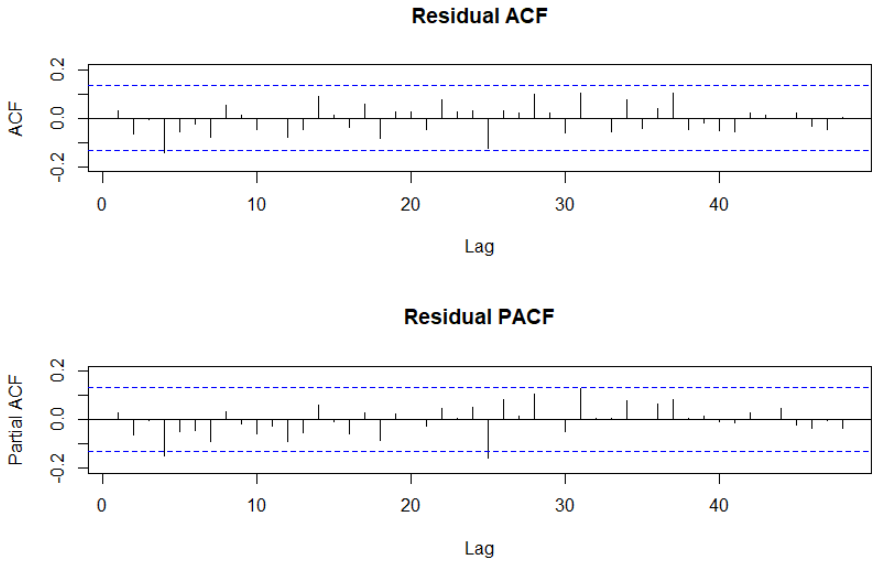


Figura 4.9: Gráfico del ACF y PACF de los residuos del modelo 1

En la gráfica se puede observar que muy pocos valores de la función de autocorrelación y de autocorrelación parcial se salen de las bandas de confianza, lo que indica que los residuales

se comportan como variables aleatorias incorreladas. Como complemento a esto se pueden comprobar los p-valores de los test de Ljung-Box, tanto gráficamente como en la tabla que proporciona la función.

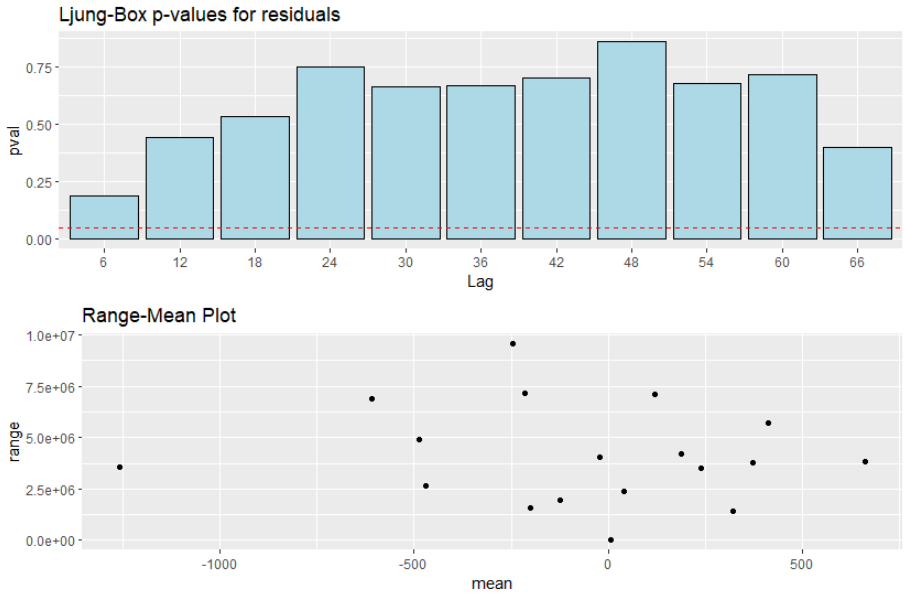


Figura 4.10: P-valores de Ljung-Box para los residuales y plot rango-media del Modelo 1

En la Figura 4.10 se observa que ninguno de los p-valores de los test de Ljung-Box es significativo, lo que indica que no se puede rechazar que las autocorrelaciones de los residuales sean nulas. En conjunto se puede concluir que el modelo es válido.

Modelo 2

En la Figura 4.11 se observa que varios de los valores de la función de autocorrelación y autocorrelación parcial de los residuales se salen de las bandas de confianza.

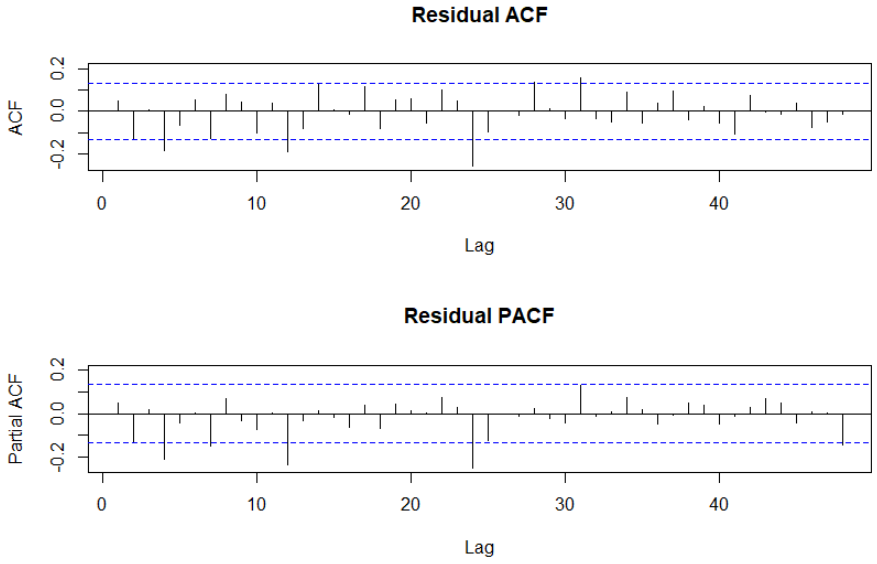


Figura 4.11: Gráfico del ACF y PACF de los residuos del modelo 2

No obstante, se comprueban los p-valores de los test de Ljung-Box para ver si se rechaza la incorrelación de los residuales.

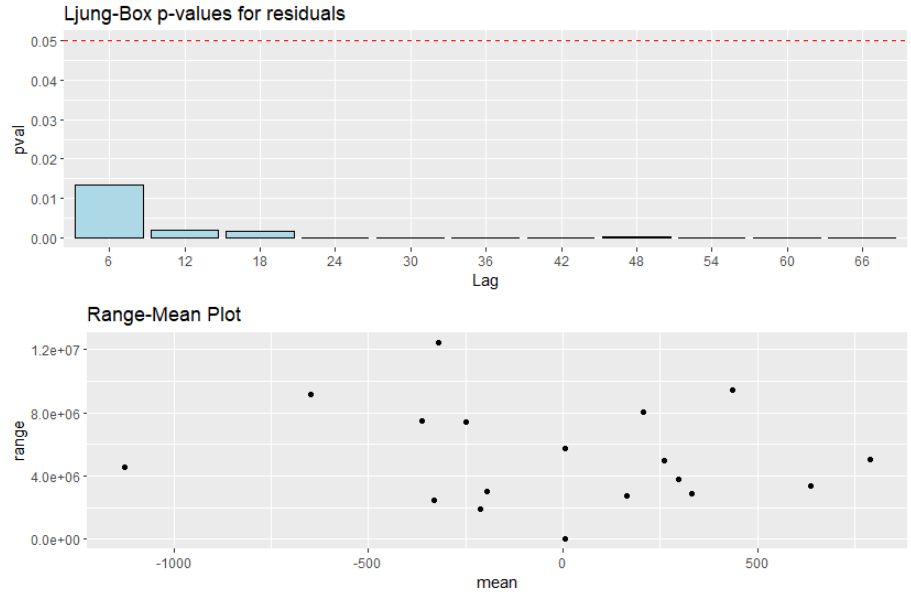


Figura 4.12: P-valores de Ljung-Box para los residuales y plot rango-media del modelo 2

En la Figura 4.12 se observa que ningún p-valor de Ljung-Box sobrepasa el valor 0.05, por lo que se rechaza la hipótesis nula para todos los retardos calculados. Teniendo en cuenta esto no se puede asegurar que el Modelo 2 sea válido, por lo que no pasa a fase de comparación.

Modelo 3

Al comprobar las funciones de autocorrelación y autocorrelación parcial de los residuales del tercer modelo (Figura 4.13), se observa que casi ningún valor de estas funciones se encuentra fuera de las bandas de confianza.

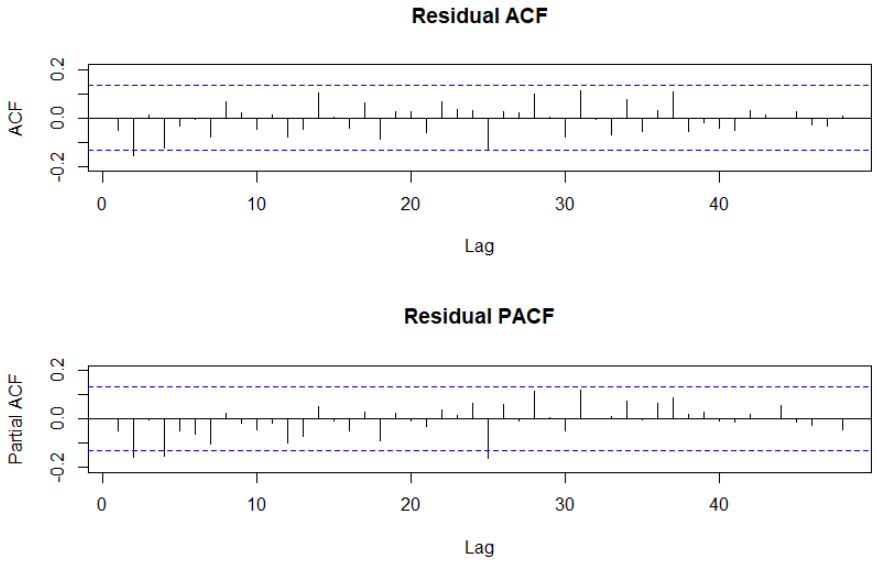


Figura 4.13: Gráfico del ACF y PACF de los residuos del Modelo 3

Al igual que en el primer modelo los residuales parecen comportarse como variables aleatorias incorreladas, pero de nuevo se comprueban los p-valores de los test de Ljung-Box (Figura 4.14).

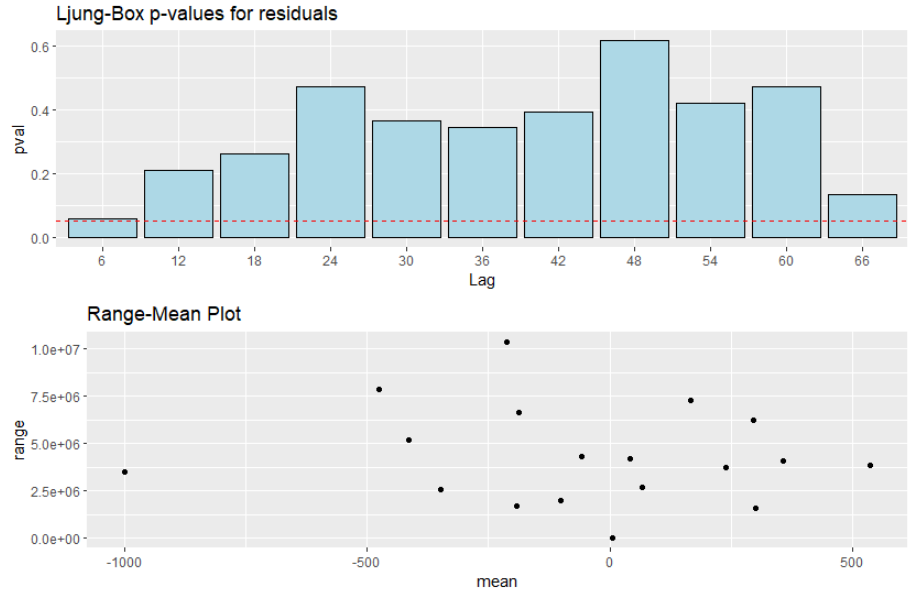


Figura 4.14: P-valores de Ljung-Box para los residuales y plot rango-media del modelo 3

Se observa que todos los p-valores están por encima de 0.05 lo que indica que no se puede rechazar la hipótesis de que las autocorrelaciones de los residuos sean nulas. Se concluye entonces que el Modelo 3 es válido.

4.6. Fase de Comparación

En esta sección se trata de elegir un modelo entre los dos que han quedado tras el resto de fases, para ello se usará la función `compareARIMA` (explicada en la Sección 3.4.2). Esta función muestra distintos estadísticos y salidas gráficas de cada modelo, de tal manera que se pueda elegir el modelo que mejor se ajuste a la muestra. Los modelos que se compararán son:

- Modelo 1: ARIMA$(0,1,1)(0,1,1)_{12}$
- Modelo 3: ARIMA$(1,1,0)(0,1,1)_{12}$

Antes de realizar la comparación es adecuado analizar primero el test de normalidad de los residuales que también se ha obtenido durante la fase de validación.

```
Model1:
    Shapiro-Wilk normality test

data:  tmp$residuals
W = 0.98804, p-value = 0.06789

Model3:
    Shapiro-Wilk normality test

data:  tmp$residuals
W = 0.99061, p-value = 0.1755
```

Se observa que en ambos modelos los residuales pasan el test de normalidad, aunque el Modelo 1 menos holgadamente que el Modelo 3. Además del test de normalidad, con la ejecución de la función `summary` se puede obtener gráficos informativos sobre la normalidad de los residuos.

La primera salida que proporciona la función `compareARIMA` al ejecutar la función es la siguiente:

```
Comparison of metrics:

                           m1           m2
Sigma2           4.275860e+06 4.396329e+06
AIC              3.693035e+03 3.698696e+03
SBC              3.730037e+03 3.735698e+03
SSEpred          2.622087e+08 3.254248e+08
Ljung-Box Lag 6  1.878353e-01 5.888636e-02
Ljung-Box Lag 60 7.169305e-01 4.722298e-01
```

Aunque en la salida aparezca m2 se refiere al Modelo 3. En esta tabla se observan diferentes estadísticos de cada modelo a comparar, entre los cuales se puede destacar el valor de la suma de cuadrados de predicción. Este valor es mucho mayor para el Modelo 3 que para el Modelo 1, lo que indica que el primer modelo tiene mejor capacidad de predicción. Esta función ofrece más tablas como salida de texto, pero debido a su tamaño se ha decidido no mostrarlas aquí. Se puede ejecutar el código utilizado para esta sección para observar el resto de la salida (Anexo 7.1).

En la Figura 4.15, se muestran los gráficos de predicción que proporciona la función.

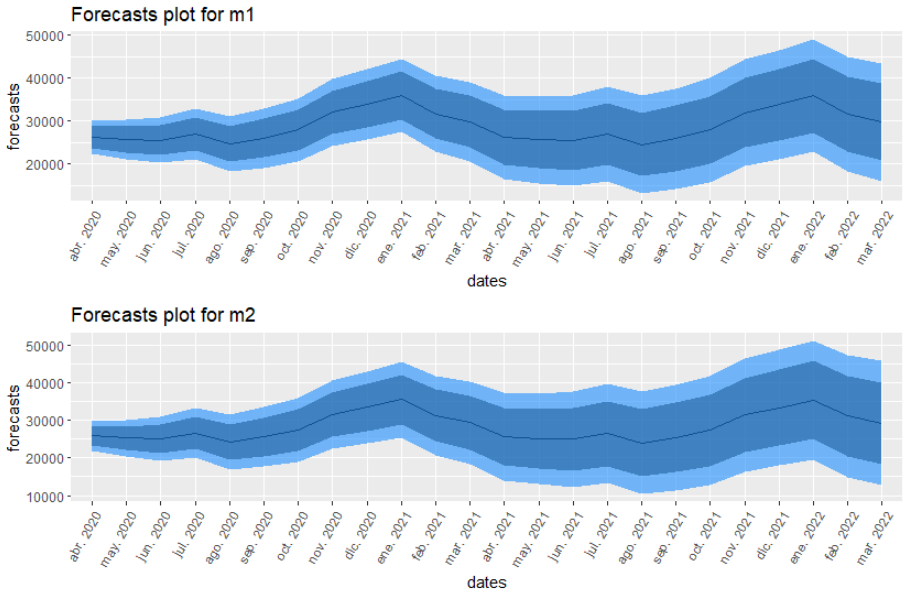


Figura 4.15: Gráfica de las predicciones de la función `compareARIMA`

En este gráfico no se percibe especialmente la diferencia en las bandas de confianza entre ambos modelos. Sin embargo, esta función proporciona otro gráfico que muestra como la anchura de las bandas de predicción para las últimas predicciones es mayor en el Modelo 3 que en el Modelo 1 (Figura 4.16).

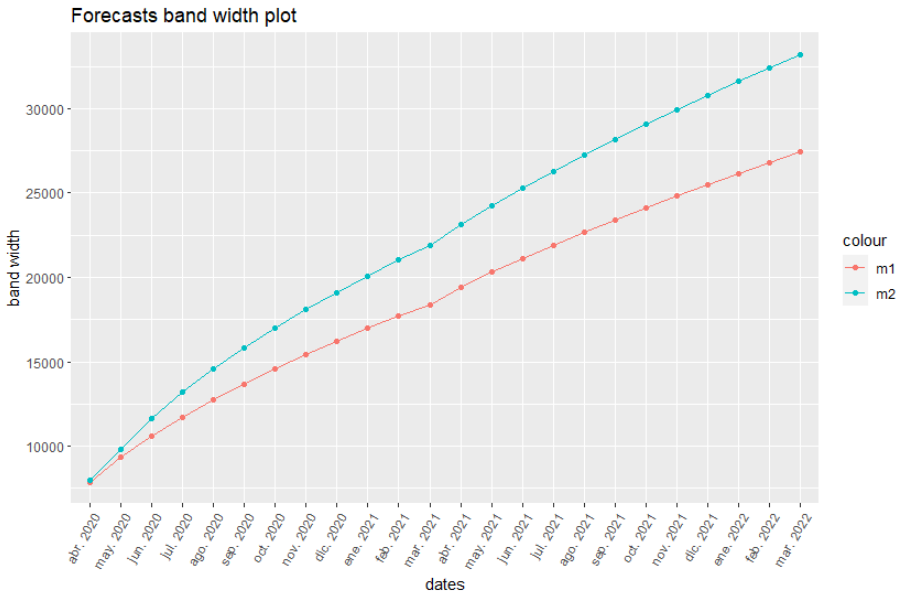


Figura 4.16: Gráfica de la anchura de las bandas de confianza

Elegimos el Modelo 1 frente al Modelo 3, ya que tiene mejor capacidad de predicción.

4.7. Fase de Predicción

Una vez seleccionado el modelo, puede interesar realizar predicciones sobre el comportamiento de la serie en los siguientes años. Para ello se puede utilizar la función `Cforecast` (explicada en la Sección 3.4.3), la cual proporciona representaciones gráficas para las predicciones que se calculen.

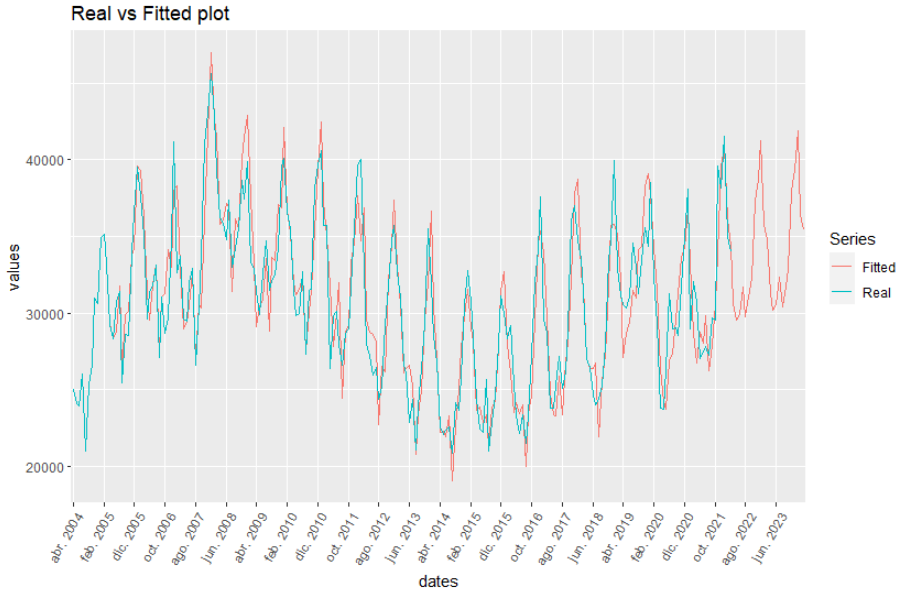


Figura 4.17: Gráfica de la serie original junto con la predicción realizada con el modelo 1

En la Figura 5.1 se observa la serie original junto con los valores estimados del modelo ARIMA$(0,1,1)(0,1,1)_{12}$, y las predicciones de los siguientes dos años. Las predicciones sugieren un comportamiento estacional en el que los picos de consumo del gas natural se dan durante la estación de invierno y otoño, mientras que durante primavera y verano es mucho menor.

Capítulo 5

Comparación con otras herramientas

En este capítulo se tratan algunas de las diferencias y limitaciones del paquete CBJTSA frente a otras herramientas existentes. En primer lugar, se habla de como los métodos utilizados por CBJTSA pueden obtener distintos resultados que los proporcionados por el procedimiento ARIMA de la herramienta SAS. A continuación se mencionan otros paquetes aún en desarrollo y otras herramientas que solucionan o tienen intención de solucionar algunas limitaciones de anteriores paquetes de R que estiman los parámetros de un modelo ARIMA por máxima verosimilitud.

5.1. Comparativa con SAS

Como ya se ha comentado en otros capítulos, el paquete CBJTSA utiliza métodos de estimación distintos a los que usa SAS. Esto implica que ante un análisis de la misma serie temporal, el modelo final elegido puede no ser el mismo si se usa una herramienta u otra. Para mostrar esto no es necesario buscar ningún ejemplo nuevo, ya que durante el análisis realizado en el Capítulo 4 se encontró ya un caso en el que SAS y R no coincidían pudiendo cambiar todo el análisis.

Durante la fase de estimación realizada en la Sección 4.4 se observó que el resultado obtenido en el ajuste del modelo ARIMA$(0,1,2)(0,1,1)$ era distinto al que obtenía SAS.

Salida del paquete CBJTSA de R:

```
ARIMA(0,1,2)(0,1,1)[12]

t test of coefficients:

      Estimate Std. Error  t value  Pr(>|t|)
ma1  -0.353254   0.070529  -5.0086 1.149e-06 ***
ma2  -0.121627   0.082620  -1.4721    0.1425
sma1 -0.808431   0.060276 -13.4120 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Estimación de probabilidad máxima					
Parámetro	Estimación	Error estándar	t valor	Approx Pr > |t|	Retardo
MA1,1	0.33802	0.06762	5.00	<.0001	1
MA1,2	0.13648	0.06780	2.01	0.0441	2
MA2,1	0.78960	0.05902	13.38	<.0001	12

Figura 5.1: Salida de SAS para el modelo ARIMA$(0, 1, 2)(0, 1, 1)$

Como se observa el valor de la estimación de los parámetros difiere por centésimas. Sin embargo, como se comentó en el apartado 3.3, la estimación poco precisa de la matriz de varianzas-covarianzas cuando se utiliza filtro de Kalman lleva a que en el caso del parámetro `ma2`, correspondiente al segundo retardo de la parte de media móvil regular, su p-valor sea mayor que 0.05 en el caso de R, y menor en el caso de SAS. Teniendo esto en cuenta el modelo se podría haber considerado en futuras etapas del análisis, e incluso podría haberse considerado un modelo mejor que el resto de modelos propuestos.

Si bien ambos métodos de estimación son igual de válidos, es importante tener en cuenta que el análisis puede cambiar dependiendo de los criterios utilizados para la elección del modelo usando una herramienta u otra.

5.2. Otros paquetes y herramientas

En el Capítulo 2 ya se ha hablado de que R es una herramienta muy versátil, propiedad que permite la implementación de multitud de métodos de análisis de datos, y esto también incluye al análisis de serie temporales.

Al comienzo del desarrollo del paquete CBJTSA se realizaron diversas búsquedas, para determinar los paquetes existentes más utilizados para el análisis de series temporales. Sin embargo, en posteriores búsquedas, realizadas en las últimas fases de desarrollo del paquete, se encontraron otros paquetes de R aún en desarrollo para el análisis de series temporales. Estos paquetes son: sarima [22] y fable [23].

La última versión del paquete sarima es bastante reciente, finales de febrero de este año, y está en las últimas fases de desarrollo, ya que como indican en la página de GitHub de este paquete (véase [24]) se encuentran en el 93 % de finalización del paquete. Este paquete tiene varias funciones para análisis de series temporales, específicamente su función `sarima` permite el ajuste de un modelo ARIMA de distintas formas, incluso permitiendo múltiples estacionalidades al mismo tiempo, lo cual es una de las limitaciones del paquete CBJTSA. En general, añade algunas características que otros paquetes de análisis de series temporales no tenían, abriendo aún más el abanico de posibilidades ya existente.

Por otro lado el paquete fable es un poco más antiguo, siendo su última versión de hace casi un año. Además parece que aún le faltan varias fases de desarrollo, ya que como

se observa en la página de GitHub de este paquete (véase [25]) se encuentra en el 71 % de finalización del paquete. Este paquete no se limita al ajuste de modelos ARIMA, sino que también implementa el ajuste de modelos mediante suavizado exponencial. En general este paquete agrupa distintos métodos y funcionalidades para el análisis de series temporales.

Además de los paquetes ya comentados de R, existen otras herramientas de libre uso que también pueden ser útiles para el análisis de series temporales. Específicamente el lenguaje de programación Python tiene a su disposición multitud de bibliotecas para el análisis de datos. Gracias a la potencia y versatilidad de Python, algunas de sus bibliotecas son capaces de lidiar con las limitaciones de algunos paquetes de R. Por ejemplo la biblioteca PyCaret contiene un módulo con funciones para el análisis de series temporales, capaz de ajustar modelos para series diarias con periodo anual de 365, cuando en el paquete CBJTSA el periodo máximo aceptado es de 355.

Capítulo 6

Conclusiones y líneas futuras

En este trabajo se ha conseguido cumplir los objetivos planteados, habiendo desarrollado un paquete para R que ordena y completa algunas de las salidas de distintos paquetes ya existentes, y se ha mostrado su utilidad con un ejemplo práctico, comprobando que es una herramienta apta para el ámbito educacional. Aunque el paquete tenga algunas limitaciones, añade bastante información útil al análisis de series temporales que realizan otros paquetes de R. Parte de la motivación de este trabajo era facilitar el acceso a herramientas de análisis de series temporales a alumnos y profesores, es por esto que se ha incluido de forma libre el código del paquete a la plataforma de *GitHub*. De esta manera cualquier persona puede utilizar el paquete y mejorar sus salidas.

Como ya se ha comentado, existen otros paquete aún en desarrollo cuyas funcionalidades podrán lidiar con algunas de las limitaciones del paquete CBJTSA. Dado que este paquete es ampliable y editable, puede ser mejorado con las funciones que estos nuevos paquetes planteen. Como líneas futuras sería interesante analizar el contenido de los paquetes que se vayan desarrollando para R y completar el contenido del paquete CBJTSA con ellos.

También se ha hablado sobre el lenguaje de programación Python y la multitud de bibliotecas que trabajan con análisis de series temporales. Puede ser interesante comparar estas bibliotecas con los paquetes de R y otra herramientas que realizan este análisis, incluso tratar de utilizar estas bibliotecas para completar las salidas de R, ya que, como ya se ha explicado, es posible ejecutar código de otros lenguajes de programación con R.

Capítulo 7

Anexo

7.1. Código en R para la Aplicación Práctica

```
####INSTALACION####
library(remotes)
install_github("danipequelangos/CBJTSA")

library(CBJTSA)

####CARGADO DE LOS DATOS Y PREPROCESADO####
gas <- read.csv2("consumogas.csv", sep = ";", dec = ",", header = T)
#Le quito los dos primeros meses porque la muestra acaba en marzo
gas <- as.numeric(gas[4:219,3])*1000
head(gas)

dates <- dateSeq(from = "2004-04", length.out = NROW(gas), by = "month")

####IDENTIFICACION####
#Grafico de la serie
tsplot(dates, gas, n.unlabel = 11)
#Grafico rango-media
rmplot(gas, n = 12)
#ACF de la serie
Acf(gas, lag.max = 12*round((NROW(gas))/4)/12)
#Periodograma de la serie
tsspec(gas)

##Diferenciacion regular##
gasDiff1 <- diff(gas, difference=1)
datesDiff1 <- dates[2:NROW(gas)]
Acf(gasDiff1, lag.max = 12*round((NROW(gasDiff1))/4)/12)

##Diferenciacion regular y estacional##
```

```
gasDiff112 <- diff(gasDiff1, lag = 12)
datesDiff112 <- datesDiff1[13:NROW(gasDiff1)]
Acf(gasDiff112, lag.max = 12*round((NROW(gasDiff112))/4)/12)
Pacf(gasDiff112, lag.max = 12*round((NROW(gasDiff112))/4)/12)

##Diferenciación estacional##
gasDiff12 <- diff(gas, lag = 12)

cat("Varianza de la serie original:",var(gas), "\n",
    "Varianza de la serie diferenciada regularmente:", var(gasDiff1), "\n",
    "Varianza de la serie diferenciada estacionalmente:", var(gasDiff12), "\n",
    "Varianza de la serie diferenciada regular y estacionalmente:",
    var(gasDiff112), "\n")

##Candidatos para modelo
#ARIMA(0,1,1)(0,1,1)
sim <- arimaSimComp(dates, gas, order = c(0,1,1), seasonal = c(0,1,1),
                    ma = c(-0.4), sma = c(-0.7), period = 12, lag.max = 40,
                    plot = F)
sim
#ARIMA(0,1,1)(0,1,2)
sim <- arimaSimComp(dates, gas, order = c(0,1,1), seasonal = c(0,1,2),
                    ma = c(-0.4), sma = c(-0.7, -0.3), period = 12,
                    lag.max = 40, plot = F)
sim
#ARIMA(0,1,2)(0,1,1)
sim <- arimaSimComp(dates, gas, order = c(0,1,2), seasonal = c(0,1,1),
                    ma = c(-0.4, -0.2), sma = c(-0.75), period = 12,
                    lag.max = 40, plot = F)
sim
#ARIMA(1,1,1)(0,1,1)
sim <- arimaSimComp(dates, gas, order = c(1,1,1), seasonal = c(0,1,1),
                    ma = c(0.4), sma = c(-0.7), ar = c(-0.5), period = 12,
                    lag.max = 40, plot = F)
sim
#ARIMA(0,1,1)(1,1,0)
sim <- arimaSimComp(dates, gas, order = c(0,1,1), seasonal = c(1,1,0),
                    ma = c(-0.4), sar = c(-0.4), period = 12,
                    lag.max = 40, plot = F)
sim
#ARIMA(1,1,0)(0,1,1)
sim <- arimaSimComp(dates, gas, order = c(1,1,0), seasonal = c(0,1,1),
                    ar = c(-0.3), sma = c(-0.8), period = 12,
                    lag.max = 40, plot = F)
sim
```

```
####ESTIMACION####
#ARIMA(0,1,1)(0,1,1)
model1 <- CArima(gas, order = c(0, 1, 1),
                 seasonal = list(order = c(0,1,1), period = 12), plot = F)
print(model1)

#ARIMA(0,1,1)(0,1,2)
model2 <- CArima(gas, order = c(0, 1, 1),
                 seasonal = list(order = c(0,1,2), period = 12), plot = F)
print(model2)

#ARIMA(0,1,2)(0,1,1)
model2 <- CArima(gas, order = c(0, 1, 2),
                 seasonal = list(order = c(0,1,1), period = 12), plot = F)
print(model2)

#ARIMA(1,1,1)(0,1,1)
model2 <- CArima(gas, order = c(1, 1, 1),
                 seasonal = list(order = c(0,1,1), period = 12), plot = F)
print(model2)

#ARIMA(0,1,1)(1,1,0)
model2 <- CArima(gas, order = c(0, 1, 1),
                 seasonal = list(order = c(1,1,0), period = 12), plot = F)
print(model2)

#ARIMA(1,1,0)(0,1,1)
model3 <- CArima(gas, order = c(1, 1, 0),
                 seasonal = list(order = c(0,1,1), period = 12), plot = F)
print(model3)

####VALIDACION####
#ARIMA(0,1,1)(0,1,1)
summary(model1)

#ARIMA(0,1,1)(1,1,0)
summary(model2)

#ARIMA(1,1,0)(0,1,1)
summary(model3)

####COMPARACION####
#Comprobamos los test de normalidad
model1$residShapiro
```

```
model3$residShapiro
#Comparamos los dos modelos
comp <- compareARIMA(gas, dates, model1, model3, h=24, plot = F)
comp

####PREDICCION####
pred <- Cforecast(gas, h = 24, model = model1, plot = F, dates = dates)
summary(pred)
```

Bibliografía

[1] The R Foundation *The R Project for Statistical Computing*. URL: `https://www.r-project.org/`, último acceso Junio 2022.

[2] George E.P. Box, Gwylim M. Jenkins. *Time Series Analysis Forecasting and Control, Fifth edition* Wiley Series in Probability and Statistics, 2015.

[3] Peña Sánchez de Rivera, D. *Análisis de series temporales*. Alianza Editorial, 2010.

[4] Brockwell Peter J., Davis Richard A. *Introduction to Time Series and Forecasting*. Springer texts in Statistics, 2002.

[5] Martín Mateos, M. *Clasificación de paquetes de series temporales en R. Ejemplo de modelización ARIMA con datos de contaminación de Madrid*. Trabajo fin de grado, Grado en Estadística, Universidad de Valladolid, 2020.

[6] Hyndman, R. J. *RDocumentation forecast library: Forecasting Functions for Time Series and Linear Models*. forecast version 8.16. URL:`https://www.rdocumentation.org/packages/forecast/versions/8.16`, último acceso Junio 2022.

[7] R Core Team. *RDocumentation ts: Time-Series Objects*. stats version 3.6.2. URL: `https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/ts`, último acceso Junio 2022.

[8] Lind Pedersen, T. *RDocumentation ggplot2 library: Create Elegant Data Visualisations Using the Grammar of Graphics*. ggplot2 version 3.3.6. URL: `https://www.rdocumentation.org/packages/ggplot2/versions/3.3.6`, último acceso Junio 2022.

[9] R Core Team. *RDocumentation diff: Lagged Differences*. base version 3.6.2. URL: `https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/diff`, último acceso Junio 2022.

[10] Chan, K. *RDocumentation periodogram: Computing the periodogram*. TSA version 1.3. URL: `https://www.rdocumentation.org/packages/TSA/versions/1.3/topics/periodogram`, último acceso Junio 2022.

[11] R Core Team. *RDocumentation arima.sim: Simulate from an ARIMA Model*. base version 3.6.2. URL: `https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/arima.sim`, último acceso Junio 2022.

[12] Hyndman, R. J. *RDocumentation Acf: (Partial) Autocorrelation and Cross-Correlation Function Estimation*. forecast version 8.16. URL: https://www.rdocumentation.org/packages/forecast/versions/8.16/topics/Acf, último acceso Junio 2022.

[13] Hyndman, R. J. *RDocumentation Arima: Fit ARIMA model to univariate time series*. forecast version 8.16. URL:https://www.rdocumentation.org/packages/forecast/versions/8.16/topics/Arima, último acceso Junio 2022.

[14] R Core Team. *RDocumentation arima: ARIMA Modelling of Time Series*. stats version 3.6.2. URL:https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/arima, último acceso Junio 2022.

[15] R Core Team. *RDocumentation KalmanLike: Kalman Filtering*. stats version 3.6.2. URL:https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/KalmanLike, último acceso Junio 2022.

[16] Zeilis, A. *RDocumentation coeftest: Inference for Estimated Coefficients*. lmtest version 0.9-40. URL:https://www.rdocumentation.org/packages/lmtest/versions/0.9-40/topics/coeftest, último acceso Junio 2022.

[17] Chan, K. *RDocumentation LB.test: Portmanteau Tests for Fitted ARIMA models*. TSA version 1.3. URL:https://www.rdocumentation.org/packages/TSA/versions/1.3/topics/LB.test, último acceso Junio 2022.

[18] Hyndman, R. J.. *RDocumentation forecast: Forecasting time series*. forecast version 8.16. URL:https://www.rdocumentation.org/packages/forecast/versions/8.16/topics/forecast, último acceso Junio 2022.

[19] R Core Team. *RDocumentation predict.Arima: Forecast from ARIMA fits*. stats version 3.6.2. URL:https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/predict.Arima, último acceso Junio 2022.

[20] López Martínez, D. *Código del paquete CBJTSA en GitHub*. URL: https://github.com/danipequelangos/CBJTSA, último acceso Junio 2022.

[21] EpData. *Evolución del consumo de gas en España*. URL: https://www.epdata.es/evolucion-consumo-gas-espana/80082729-fa15-420f-95b6-fea8b00ebd89, último acceso Junio 2022.

[22] Boshnakov, G. N. *Package sarima: Simulation and Prediction with Seasonal ARIMA Models* sarima version 0.9. URL: https://cran.r-project.org/web/packages/sarima/sarima.pdf, último acceso Junio 2022.

[23] O'Hara-Wild, M. *Package fable: Forecasting Models for Tidy Time Series* fable version 0.3.1. URL: https://cran.r-project.org/web/packages/fable/fable.pdf, último acceso Junio 2022.

[24] N. Boshnakov, G. *Código del paquete sarima en GitHub*. URL: https://github.com/GeoBosh/sarima, último acceso Junio 2022.

[25] O'Hara-Wild, M. *Código del paquete fable en GitHub*. URL: `https://github.com/tidyverts/fable/`, último acceso Junio 2022.

[26] Alonso Losa, L. *Aplicación del análisis de series temporales a variables atmosféricas*. Trabajo fin de grado, Grado en Estadística, Universidad de Valladolid, 2021.

[27] Vaquero Martínez, V. *Análisis de series multiestacionales mediante modelos de Espacio de Estados*. Trabajo fin de grado, Grado en Estadística, Universidad de Valladolid, 2020.

[28] Gómez Nuño, R. *Nuevo paquete para el análisis de datos de juego y jugadores de fútbol: GASB*. Trabajo fin de grado, Grado en Estadística, Universidad de Valladolid, 2020.