Facultad de Ciencias Sociales y Jurídicas

Grado en Estadística y Empresa

Universidad de Jaén

Facultad de Ciencias Sociales y Jurídicas

Trabajo Fin de Grado

ANÁLISIS DE SERIES TEMPORALES NO ESTACIONARIAS EN RSTUDIO

Alumno: Álvaro Hervás Martos

Mayo, 2020

RESUMEN

En análisis de series de tiempo es de vital importancia en diferentes campos de la investigación a la hora de elaborar predicciones. En el siguiente documento nos centramos en su análisis utilizando la metodología de Box & Jenkins a través del software estadístico “RStudio”, ya que, a pesar de tratarse de un sofware gratuito, se trata de una de las herramientas más importantes y potentes a la hora de realizar análisis estadísticos. Además, se revisan las principales funciones disponibles para el análisis de series temporales y se presenta un ejemplo práctico de su uso.

PALABRAS CLAVE: Estadística, Series de tiempo, Remuneraciones, software Rstudio.

ABSTRACT

In time series analysis it is of vital importance in different fields of research when making predictions. In the following document we focus on its analysis using the Box & Jenkins methodology through the statistical software "RStudio", since, despite being free software, it is one of the most important and powerful tools for time to perform statistical analysis. In addition, the main functions available for time series analysis are reviewed and a practical example of their use is presented.

Key Word: Statistics, Time Series, software RStudio, Remunerations

ÍNDICE

Capítulo 1. Introducción a las series temporales y metodología de Box-Jenkins

1.1. Introducción

En este capítulo abordaremos de forma teórica la definición de serie temporal y el conjunto de técnicas y procedimientos para llevar a cabo el análisis de una serie cronológica no estacionaria mediante la metodología de Box & Jenkins. De tal forma que, comenzaremos definiendo qué se entiende por serie temporal, conoceremos sus características específicas y explicaremos como llevar a cabo su análisis aplicando la metodología paso a paso.

1.2. Concepto de serie temporal

Cuando hablamos de serie temporal o cronológica, nos referimos a una secuencia de datos medidos en determinados momentos y ordenados en el tiempo. Para estudiar series, se utilizan diferentes métodos, que permiten modelizar el conjunto de datos para así poder interpretarlos y extraer conclusiones, tales como predecir el comportamiento futuro de la serie a base a los valores pasados de esta. Uno de los métodos más importantes y que será el que llevemos a cabo en este manual, es de Box-Jenkins, que explicaremos de forma más concreta en el siguiente apartado.

1.3. Metodología de Box-Jenkins

La metodología de Box & Jenkins se creó en 1970 por los estadísticos George Edward Pelham Box y Gwilym Meirion Jenkins, para intentar hacer más fácil el trabajo

a los analistas a la hora de crear un modelo para una serie de tiempo, ayudando a explicar su estructura y a realizar una predicción a cerca del comportamiento futuro de la serie. Esta metodología, que se explicó por primera vez en el libro "Time Series Analysis: Forecasting and Control" se trata de un procedimiento de análisis estadístico que sirve para ajustar un tipo especial de modelos autorregresivos ARMA, siglas en inglés de (Autorregressive Moving Average*)* además de los modelos autorregresivos integrados de medias móviles, también llamados ARIMA (Autorregresive Integrated Moving Average).

Esta metodología formula una teoría para crear un tipo general de modelos que sirvan para describir la serie y construye un procedimiento para encontrar el mejor modelo que se ajuste a una serie temporal real, esto es, mediante esta metodología podemos crear el mejor modelo que nos describa la serie para conseguir obtener unas predicciones fiables sobre el comportamiento futuro de una serie cronológica.

La metodología de Box-Jenkins se trata de un procedimiento con distintos pasos a seguir para crear el modelo, estos son: identificación, estimación, diagnosis, verificación y predicción. A continuación, vamos a proceder a explicar de manera más detallada las diferentes fases.

I. Identificación.

En este primer paso se trata de identificar si la serie es estacionaria, o lo que es lo mismo, si observamos que los datos parecen oscilar alrededor de un valor fijo. Una forma práctica de observar la estacionariedad es graficar la serie temporal y observar su gráfica y la función de autocorrelación. En caso de que la serie no sea estacionaria, debemos llevar a cabo los mecanismos necesarios para hacerla estacionaria. Nos podemos encontrar con diferentes situaciones:

- El caso en el que exista heterocedasticidad, es decir, que se observe que la varianza no presente un grado de dispersión similar durante todo momento, en este caso deberemos de

realizar transformaciones. Entre las transformaciones más usuales se encuentran la logarítmica o la familia de transformaciones de Box-Cox.

- Otro caso, es en el caso en que la serie presente algún tipo de tendencia y no sea estacionaria en media, entonces deberemos de diferenciar la serie objeto de estudio hasta conseguir eliminar la tendencia hasta observar que los puntos de la muestra oscilan entorno a la media.

Una vez hemos solucionado los problemas de estacionariedad, podemos observar una nueva grafica de la serie corregida, además de otras nuevas funciones de autocorrelación simple (f.a.s) y de autocorrelación parcial (f.a.p). Con la serie ya tratada y corregida de estacionariedad, ya es posible proceder a estimar un Modelo Autorregresivo-Integrado de Medias Móviles de orden p,d,q (ARIMA), esto es, un modelo ARMA(p,q) aplicado a una serie integrada de orden d, dicho de otro modo, una serie la cual ha sido necesario realizarle una diferenciación para eliminar su tendencia.

En la siguiente fase procederemos a hallar los órdenes p y q de nuestro modelo ARIMA; El método de Box & Jenkins nos posibilita hacerlo por medio de la autocorrelación parcial, es decir, de hacerlo es observando las gráficas de las f.a.s y f.a.p que hemos obtenido como resultado de corregir nuestra serie. También lo podemos hacer de manera más precisa mediante softwares de análisis estadístico, ya que estos nos proporcionaran el p-valor exacto para decidir sobre el modelo más apropiado.

- A la hora de buscar el mejor orden de p, lo hacemos con la gráfica de la función de autocorrelación parcial (f.a.p), ya observando el retardo en el que corta, este nos dice cual será nuestro orden de p. En un proceso AR(p), la f.a.s decrece exponencialmente mediante series sinusoidales amortiguadas y tendrá una función

de autocorrelación parcial distinta de cero para valores que se encuentren dentro del grado de libertad del modelo y los demás serán nulos.

- Para encontrar el orden de q más idóneo, lo haremos observando el retardo en el que corta la f.a.s. En un modelo MA(q) sucede al contrario que en modelos AR(p), ahora es la f.a.p la que decrece de forma exponencial y la f.a.s la que tiene los q primeros coeficientes significativos y los demás nulos.

Es así que, deberemos de plantear diferentes parámetros para p y q que consideremos aptos observando las autocorrelaciones y, dado que no existe un único modelo correcto, debemos plantear diferentes modelos ARIMA(p, d, q) aspirantes.

II. Estimación y Verificación

En esta etapa se procede a estimar los parámetros de los modelos seleccionados en la fase anterior. Se estiman los coeficientes de los términos autorregresivos y de medias móviles seleccionados en el modelo, cuyos términos p, d y q han sido identificados en la etapa anterior.

Hay momentos en que la estimación se realiza por el método de mínimos cuadrados lineales, pero en otras se recurre a la estimación no lineal de los parámetros. Este último procedimiento utiliza un algoritmo para minimizar la suma de los cuadrados de los residuos, empezando en un valor inicial de los parámetros del modelo. Por lo general el algoritmo busca si otro vector de parámetros mejora el valor de la función objetivo, produciendo iteraciones sucesivas hasta alcanzar la convergencia. Las librerías estadísticas realizan este procedimiento a través de rutinas de computación en las que se tienen definidos los parámetros iniciales, así como también los criterios de convergencia.

En teoría, el método de minimos cuadrados ordinarios cuanto mayor sean las muestras posee propiedades asintóticas, lo que quiere decir que se crean estimadores consistentes y que convergen a una distribución normal, en consecuencia las pruebas de hipótesis convencionales para los parámetros serán válidas.

Para realizar la estimación de un modelo ARIMA(p, d, q) se realiza para una serie la cual se ha refutado que estacionaria. En la práctica, los modelos más usuales son los autorregresivos pero, de acuerdo al teorema de Wold, el modelo ARMA siempre debería de ser la primera opción, ya que incluyendo términos adicionales MA puede mejorar las propiedades estadísticas de estimación.

Como en la práctica resulta difícil reconocer con exactitud los ordendes p y q del modelo ARMA, se plantean dos o más modelos posibles, que luego al estimarlos elegimos el más conveniente.

III. Diagnosis.

En este momento, puesto que debemos de quedarnos con el mejor modelo de los aspirantes que hemos seleccionado, debemos contrastar cuál de estos modelos elegidos es el más adecuado para modelar nuestra la serie temporal. Esto se hace mediante el análisis de los residuos, si la serie en estudio está bien identificada, y el modelo ajustado es el mejor, estos residuos no deben presentar ninguna estructura, ya que, una serie sin estructura es lo que denominamos el ruido blanco.

Cuando hablamos de un ruido blanco, nos referimos a una serie cronológica estacionaria en la cual las observaciones son independientes, es decir, su f.a.s y su f.a.p son cero, ninguna barra se sale de las barras de confianza.

- Los residuos deberán de cumplir la hipótesis de independencia. Para confirmarlo debemos observar si las gráficas de la f.a.s y f.a.p residuales tienen algún retardo significativo o cercano a los intervalos de confianza. Para cerciorarnos tenemos que realizar el "test de Box-Pierce" el cual, cuanto mayor nos indique el p-valor, podremos afirmar con más fuerza que los residuos son un ruido blanco.

- Otra importante hipótesis que tenemos que contrastar es la de homocedasticidad de los residuos. Esta se puede hacer realizando un “test de Bartlett” además de comprobándolo gráficamente.
- Los residuos también deben de cumplir la hipótesis de normalidad. Podremos contrastarla mediante el “test de Shapiro-Wilk” además de observando una gráfica “Q-Q plot” y comprando que estos residuos se ajusten a una normal.

IV. Verificación.

En esta fase se realiza un análisis detallado de los errores, estos estadísticos tienen en cuenta las diferencias que existen entre los valores históricos y los estimados. Hay que hacer una verificación exhaustiva para corroborar que no exista un comportamiento sistemático de estos, así como analizar la posibilidad de que existan errores especialmente significativos. Según estos resultados, decidiremos el modelo definitivo que adoptaremos para realizar nuestra predicción. Para realizar esta verificación nos fijaremos en los criterios que nombramos a continuación:

- ME: Error medio.
- RMSE: Raíz del error cuadrático medio.
- MAE: Error absoluto medio.
- MPE: Error medio porcentual.
- MAPE: Error absoluto medio porcentual.

Observado los criterios anteriores, elegiremos el modelo que, en su conjunto, minimice los valores de estos errores.

Otros criterios a tener en cuenta a la hora de la elección de nuestro modelo son el AIC y el SC, dado que, es importante que den el menor valor posible al comparar varios modelos con combinaciones diferentes de p y q.

V. Predicción.

Es el objetivo principal de la metodología, en esta última fase trataremos de conocer cual será el comportamiento futuro de nuestra serie, además, de saber si el modelo elegido es óptimo para realizar la predicción. Se conoce que los modelos ARIMA(p, d, q) proporcionan buenos resultados a corto plazo.

Capítulo 2. Qué es RStudio, instalación, y uso del software

2.1. Introducción

En el siguiente capitulo procederemos a explicar como descargar e instalar el software estadístico "R" y su entorno "RStudio" que utilizaremos para realizar nuestros análsis. Además, indicaremos de manera pormenorizada, como llevar a cabo, de diferente maneras, la instalación de los paquetes necesarios para el análisis de series cronológicas.

2.2. ¿En qué consisten "R" y "RStudio"?

"R" es un entorno y lenguaje de programación dedicado al análisis estadístico. Lo que queremos decir cuando hablamos de lenguaje programación, es que, "R" nos permite dar instrucciones mediante códigos para que el software realice diversas tareas tales como: realizar operaciones con multitud de tipos de datos, hacer cálculos y realizar gráficos para su interpretación. Es muy utilizado en el ámbito estádistico, debido a que se trata de un software de los denominados GNU, que son las siglas en inglés de *"General Public Licence",* dicho en Español, que estamos ante un software totalmente gratuito.

En cuanto a “RStudio”, es un entorno integrado de desarrollo (IDE, de sus siglas en ingles) que, aunque podemos usar R directamente, nos ofrece una forma mas cómoda e intuitiva de trabajar. Las principales ventajas que nos ofrece son:

- Respeta la filosofía de R.

- Expone los objetos de nuestro espacio de trabajo.

- Su forma de exponer gráficas es muy interesante al unificar entornos.

- Dispone de un visor de la librería de paquetes que tenemos instalados y carga dos.

- Tiene una función muy practica de autocompletado de códigos.

- Nos permite trabajar en varios proyectos al mismo tiempo.

2.2.1. Descarga e instalación de “R”

En primer lugar, debemos de descargar el programa, esto lo haremos desde la página oficial (FIGURA 1), accediendo desde la siguiente dirección: https://www.r-project.org/, de forma gratuita y sin coste alguno, ya que se trata de un software libre. Para poder instalar “RStudio”, primero deberemos de descargar “R”, ya que, por un lado, R es el lenguaje de programación, es decir, quien realiza los cálculos y por otra parte, RStudio es un IDE o entorno de desarrollo integrado, lo que quiere decir que, es como un subprograma que se utiliza para manejar R y utilizarlo de una forma más fácil y con más comodidad.

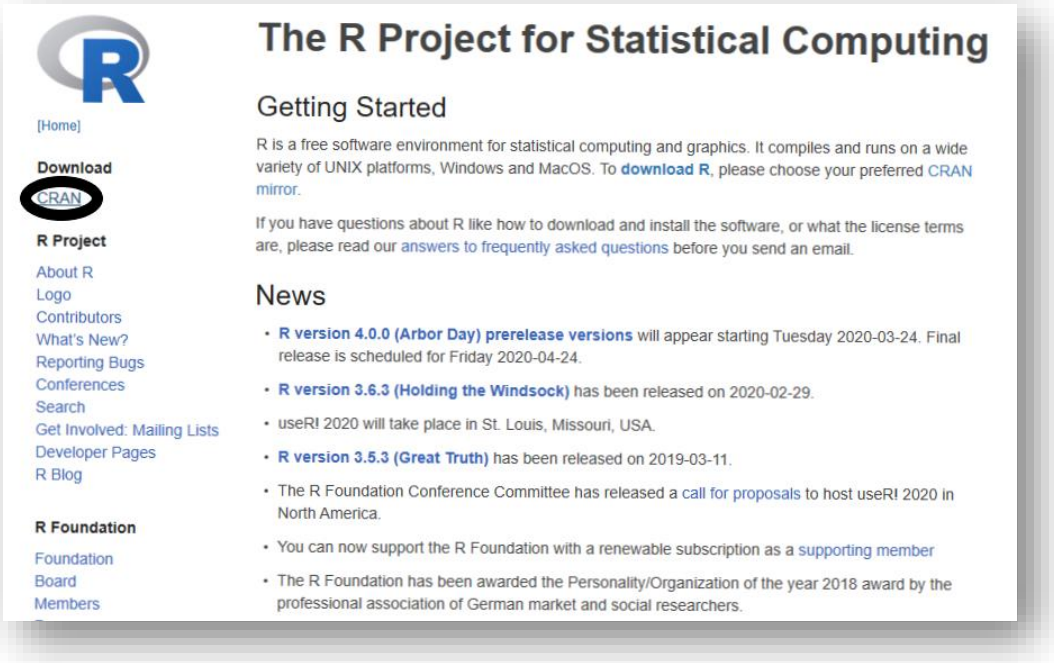


FIGURA 1

Una vez dentro de la página web de "R", nos iremos al extremo izquierdo de la imagen y dentro de la sección *"Download"* haremos clic en *"CRAN"*. Al pulsar no va a mandar a un nuevo enlace (FIGURA 2), en el cual, deberemos de buscar nuestro país y hacer clic en *"Spain"*. Lo siguiente que tenemos que hacer es seleccionar cualquiera de los en enlaces disponibles que nos salen y seguidamente elegir el que indica el sistema operativo de nuestro ordenador, en mi caso *"Download R for Windows"(FIGURA 3).*

FIGURA 2

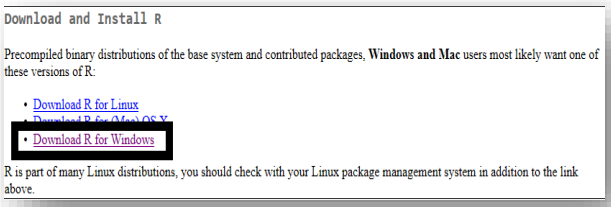


FIGURA 3

En la siguiente ventana a la que nos enviará (FIGURA 4), pulsaremos la opción *"Install R for the first time"*.

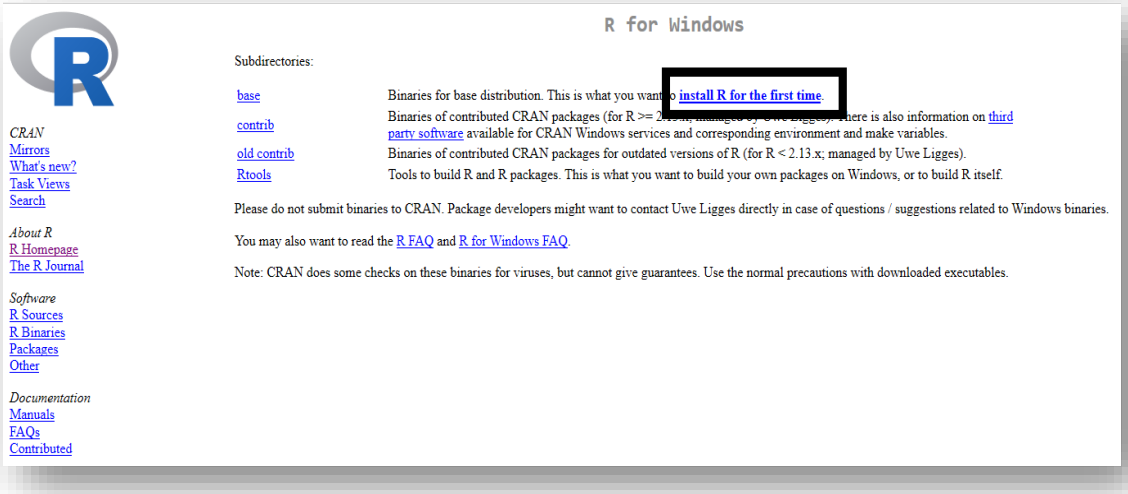


FIGURA 4

Y, a continuación, en la siguiente ventana que se muestra posteriormente (FIGURA 5), haremos clic en *"Download R 3.5.0 for Window"* para finalizar con la descarga del instalador. Una vez descargado, el siguiente paso será instalarlo, para ello debemos ejecutar el instalador, indicando el idioma, la carpeta de destino y dándole a siguiente hasta completar la instalación; El proceso es muy sencillo.

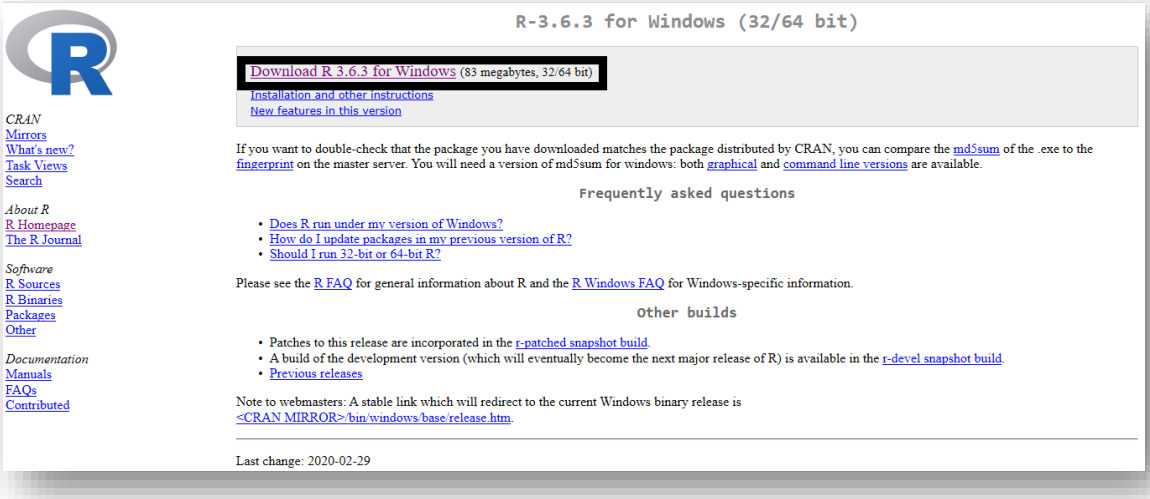


FIGURA 5

2.2.2. Descarga e instalación de RStudio

Una vez que hemos instalado "R", proseguimos con la instalación de su interfaz "RStudio". Para ello, primeramente, entramos en la página oficial: https://rstudio.com/ para proceder a la descarga del instalador.

Una vez dentro de la web (FIGURA 6), pulsaremos sobre *"DOWNLOAD"*. Seguidamente en la próxima ventana (FIGURA 7) volveremos a hacer clic en *"DOWNLOAD" "RStudio Desktop"* para descargar la versión gratuita.

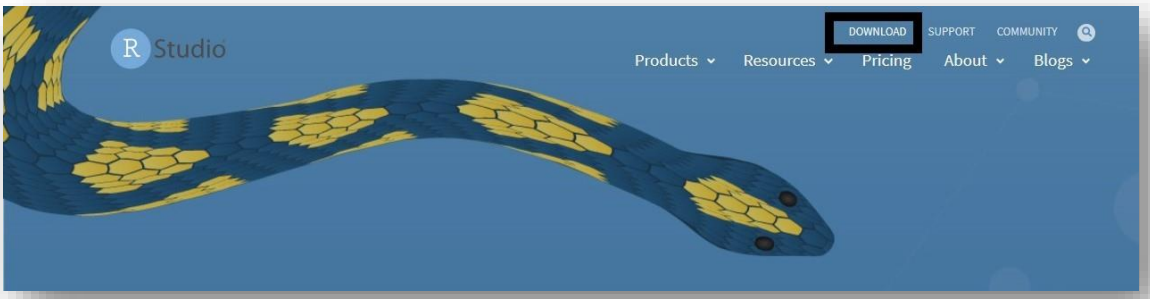


FIGURA 6

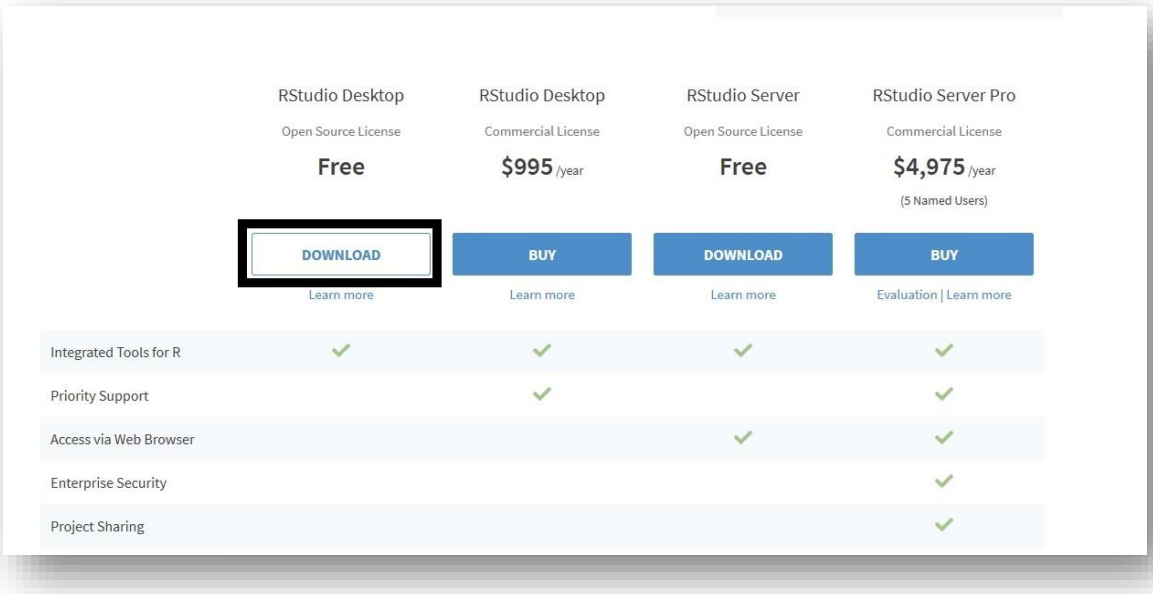


FIGURA 7

Seguidamente (FIGURA 8), la web detecta nuestro sistema operativo y bastará con clicar en *"DOWNLOAD RSTUDIO FOR WINDOW"* y comenzará la descarga. En la parte baja de esta página también nos da la posibilidad de descargarlo para diferentes sistemas operativos.

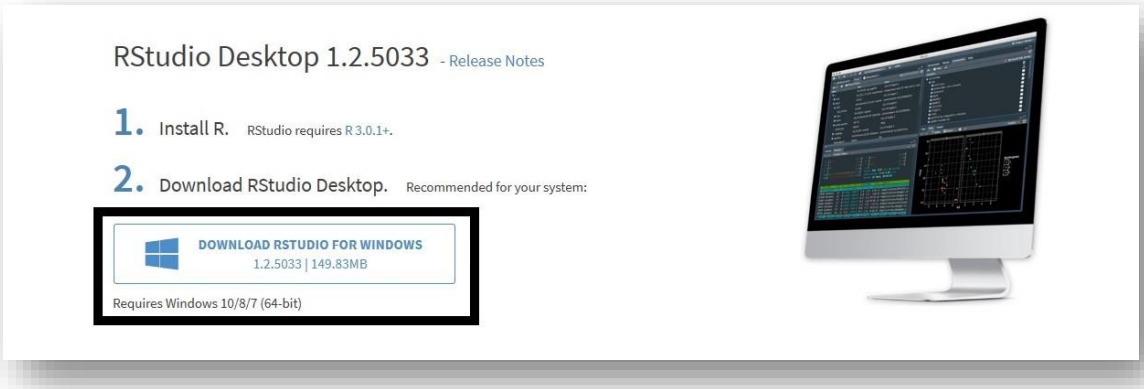


FIGURA 8

Una vez descargado, basta con ejecutar el "Asistente de instalación" y seguir los sencillos pasos para concluir la instalación.

2.3. Paquetes en R

Los paquetes o también llamados librerías son colecciones de distintas funciones, datos y estructuras que han sido diseñadas para obedecer a diferentes tareas. Estos paquetes hacen que R sea más potente y disponga de más funcionalidades. Por ejemplo, existen paquetes para realizar inferencia estadística, analizar series temporales, realizar regresión, análisis psicométricos y muchas más.

Podemos ver todos los paquetes que existen accediendo a la página oficial de "R", donde podemos encontrar casi 10.000 diferentes, ordenados por fecha de publicación y nombre. También se pueden encontrar ordenados según la disciplina que estemos estudiando, por ejemplo, en el caso de nuestro estudio encontraremos que existe una sección llamada *"TimeSeries",* donde encontraremos todo lo relacionado con el análisis de series temporales. Estos paquetes se localizan en el repositorio oficial *CRAN* y pasan una estricta verificación antes de estar dispuestos para usarlos.

2.3.1. Como instalar paquetes en RStudio

En RStudio tenemos varias técnicas distintas para instalar los paquetes o librerías. La primera de estas técnicas consiste en acceder a través de la consola haciendo clic en *"Tools"* y a continuación en *"Install Packages…"*. Este procedimiento nos abrirá la siguiente ventana (FIGURA 9):

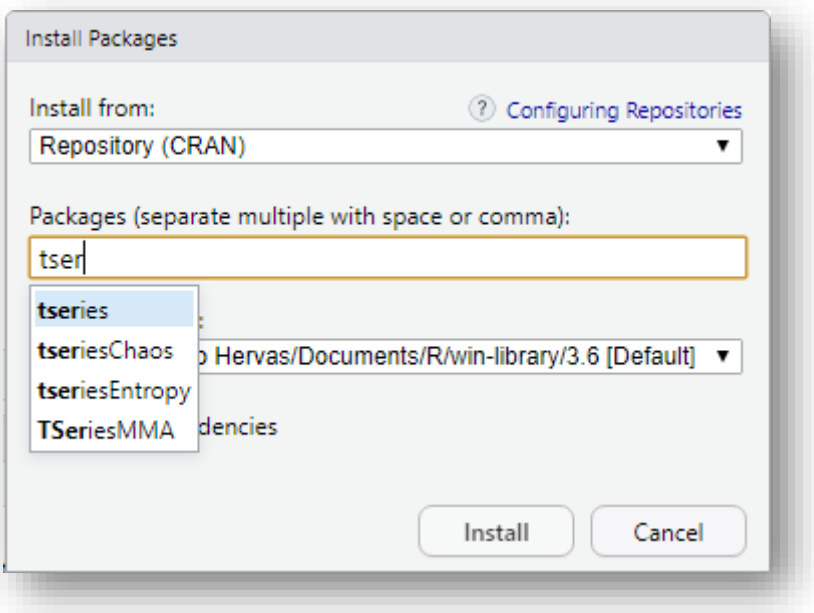


FIGURA 9

En el cuadro de dialogo que se nos abre, bastará con escribir el nombre de la librería que queremos y hacer clic en *"Install"* y el paquete quedará instalado. Como podemos ver en la FIGURA 9, RStudio no sugiere diferentes librerías al escribir las iniciales.

Otra forma de realizar la instalación de los paquetes es a través de nuestra consola y mediante la función *"install.packages()",* indicando como argumento el nombre del paquete que queremos instalar, entre comillas. Por ejemplo, para proceder a la instalación del paquete *"forecast"*, otro de los paquetes importantes que necesitaremos para nuestros análisis, escribiremos lo siguiente (FIGURA 10):

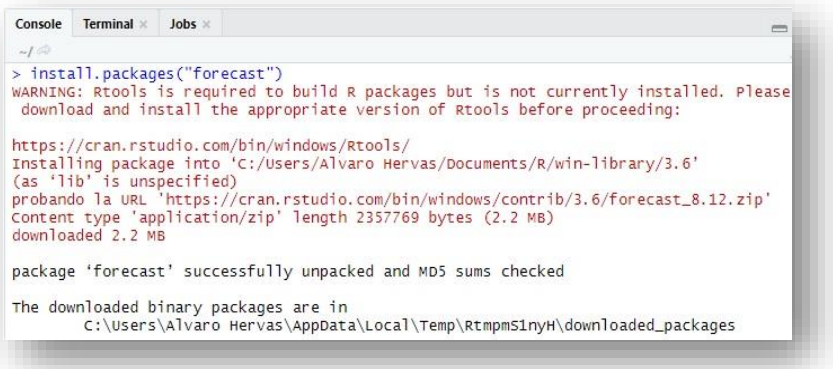


FIGURA 10

La siguiente forma de instalar paquetes en RStudio que presentamos se encuentra en la consola inferior derecha (FIGURA 11), accediendo en la pestaña *"Packages"* y clicando en *"Install"*; Esto nos llevará a una ventana como la de la primera opción en la que bastará con buscar el nombre del paquete y pulsar en instalar. Desde esta misma consola inferior izquierda también podemos cargar paquetes, como posteriormente indicaremos de forma más detallada.

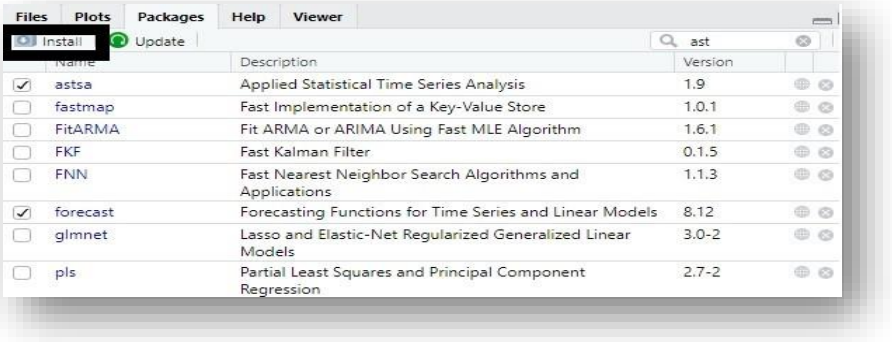


FIGURA 11

2.3.2. Como realizar la carga de paquetes en RStudio

Una vez que hemos instalado los paquetes, si queremos hacer uso de un paquete, la manera mejor de utilizarlos para trabajar es cargarlo en la memoria y desde ese momento podremos hacer uso de las funciones, comandos y datos. En "R" y "RStudio" existen distintas formas para realizar las cargas de las librerías o paquetes. A continuación, explicaremos cada procedimiento que existe.

La primera forma de realizar la carga se realiza desde la consola inferior derecha (FIGURA 11), esta, además de poder realizar la instalación de los paquetes, también nos

da la posibilidad de cargarlos. Para hacerlo únicamente debemos de marcar la pestaña *"Packages"*, buscar el paquete que nos interesa en el buscador y con marcar el recuadro del paquete de interés bastará para realizar la instalación como mostramos en la FIGURA anterior. Además, pulsando el nombre del paquete nos dará información acerca del mismo, como por ejemplo sus funciones, fecha de creación, etc.

Otra forma de proceder a la carga de paquetes es a través de la función *"library()"*, indicando entre paréntesis el nombre del paquete que queremos cargar (FIGURA 12).

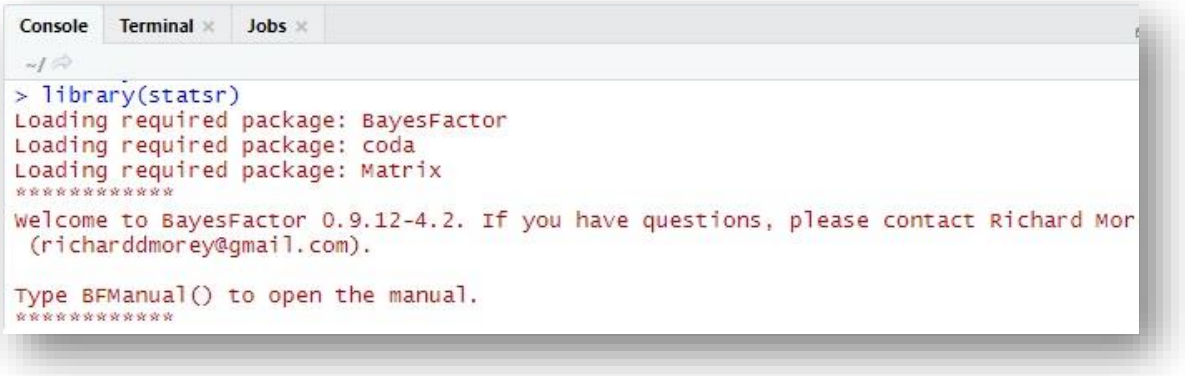


FIGURA 12

2.4. Scripts

Los *Scripts* no son más que documentos de texto que R puede leer y ejecutar su código. Si bien es cierto que R permite que se utilicen interactivamente, es recomendable que guardemos nuestro código en un archivo .R, ya que, así se puede utilizar después además de compartirlo con otras personas. Cuando utilizamos y abrimos un script en RStudio, este nos abre una consola en la cual podemos ver todo el contenido y así podemos ejecutar todos los códigos o las partes que nos interesen.

2.5. Cómo cargar e importar datos en RStudio

Para importar datos en RStudio, lo primero es disponer de ellos. Los datos los podemos encontrar en diferentes formatos como pueden ser Excell, SPSS o algún otro tipo de archivo. No obstante, la mayoría de los datos que utilicemos nos los encontraremos por internet o los obtendremos desde otras fuentes diferentes. Esta carga de datos se puede efectuar de diferentes maneras, es por eso que nosotros nos centraremos en la más fácil y rápida para no tener que complicarnos utilizando las distintas funciones que existen.

Por tanto, a la hora de importar nuestro conjunto de datos a RStudio, resulta interesante constatar algunas cuestiones:

- Si nuestros datos provienen de una hoja de cálculo, debemos saber que la primera fila se reserva para la cabecera y la primera columna se utiliza para identificar el dato.
- Tenemos que obviar nombres o campos con espacios en blanco, debido a que cada palabra será interpretada como una variable y nos dará errores con respecto al número de elementos por fila del conjunto de observaciones.
- Si queremos concatenar palabras, se recomienda poner un guion (-) entre cada palabra en vez de un espacio en blanco.
- Es aconsejable escoger nombres cortos en lugar de más extensos.
- Borrar algún comentario que haya en el archivo Excell para evitar que nos haga columnas extra y que nos introduzca valores “NA”.
- Verificar que si hay valores desconocidos en nuestro conjunto sea indicado como “NA”.

De modo que, una vez que hemos llevado a cabo la verificación anterior en nuestros datos, para comenzar a importar o cargar los datos, debemos hacer clic en la pestaña *"File",* que la podemos encontrar en la consola superior de RStudio, cuando hagamos clic se nos abrirá otra ventana en la que escogeremos la opción *"Import Dataset"* que nos lanzará otra ventana donde se muestran todos los formatos de datos que RStudio acepta (FIGURA 13). En nuestro caso seleccionaremos *"From Excell"*, puesto que el conjunto de datos objeto de estudio en este documento procede de un archivo del tipo *.xls.*

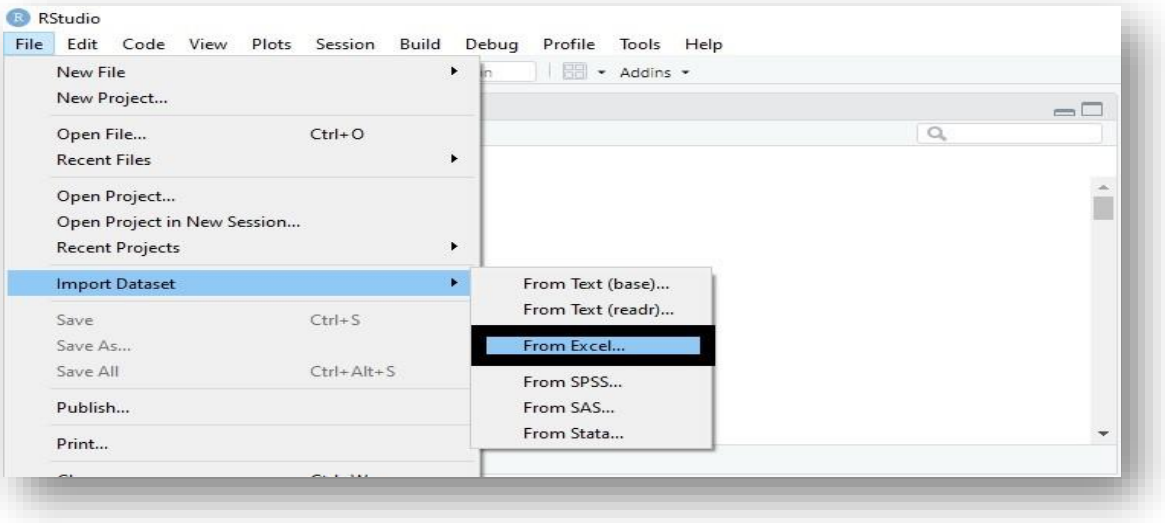


FIGURA 13

Después de implementar los pasos anteriores se abrirá otra ventana (FIGURA 14), en la cual pulsaremos sobre la opción *"Browse"* para localizar nuestros datos. Inmediatamente a continuación pulsaremos sobre la opción *"Import"* y tendremos nuestro conjunto de datos importado correctamente para comenzar a trabajar.

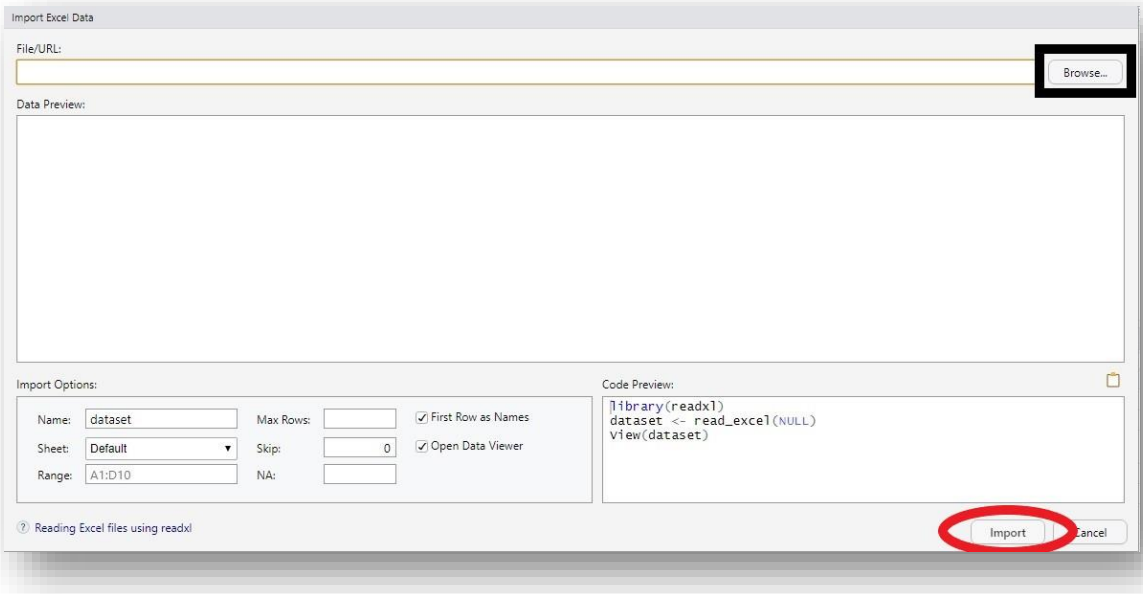

Import Excel Data
File/URL:
Browse...
Data Preview:
Import Options:
Name:
dataset
Sheet:
Default
Range:
A1:D10
Max Rows:
Skip:
0
NA:
First Row as Names
Open Data Viewer
Code Preview:
library(readxl)
dataset <- read_excel(NULL)
View(dataset)
Reading Excel files using readxl
Import
Cancel

FIGURA 14

Capítulo 3. Preparación del entorno de trabajo y análisis en “RStudio”

3.1. Introducción

Una vez que conocemos la metodología de Box & Jenkins y comprendemos la base del funcionamiento de RStudio, en este último capítulo llevaremos a cabo el análisis de nuestro conjunto de datos. En primer lugar, explicaremos en que consiste nuestro conjunto de datos y su procedencia. Además, expondremos los paquetes necesarios que utilizaremos para realizar el análisis en RStudio así como las funciones necesarias para llevar a la práctica el análisis de nuestra serie. El objetivo principal será encontrar el mejor modelo que consiga explicar nuestra serie utilizando la metodología estudiada y sirviéndonos del software estadístico RStudio.

3.2. Preparación del entorno de trabajo

Resulta importante antes de proceder a la carga o importación del conjunto de datos en RStudio preparar el espacio de trabajo. Esto es debido a que nos podemos encontrar ante un entorno con otros datos, valores o funciones que se pueden borrar utilizando la función *“rm(list = ls())”*. La función *“rm()”* nos da la posibilidad de hacer un borrado de todos los objetos del entorno y si argumentamos con el la función “ls()” esta nos proporciona un vector que nos indica los nombres de los objetos del espacio de trabajo.

3.2.1. Paquetes necesarios para análisis de series temporales

Una vez que nuestro entorno se encuentra limpio y preparado, el siguiente paso necesario para llevar a cabo el estudio de nuestra serie es, como explicamos con anterioridad, instalar y cargar los paquetes necesarios para llevarlo a cabo. Existen varias librerías para realizar por lo que nosotros nos centraremos en la siguientes:

- Forecast: Creado por Rob J. Hyndman, nos proporciona diferentes métodos y funciones para visualizar y realizar pronósticos de seres cronológicas además de proporcionarnos una función muy practica para realizar una selección rápida y automática del mejor modelo ARIMA.

  Algunas de sus funciones más importantes de este paquete que utilizaremos son:

  - *auto.arima()* este comando nos encuentra de manera rápida las (p, d, q) más adecuadas del modelo ARIMA en base a criterios como el *BIC, AIC y AICc*.
  - *tsdisplay()* nos realiza un gráfico de la serie, además de su f.a.s y f.a.p.
  - *forecast()* nos proporciona la predicción estimada para nuestro modelo ARIMA.
  - *checkresiduals()* es una de las funciones nuevas de las que dispone tras sus últimas actualizaciones y que nos sirve para realizar una evaluación de los residuos de nuestro modelo.

- Stats: Se trata de un paquete que RStudio instala automáticamente y que contiene funciones estadisticas básicas importantes como pueden ser:

  - *qqnorm()* con la que podemos realizar una Q-Q plot
  - *acf()* es una función que nos sirve para calcular la función de autocorrelación simple y además, nos realiza su gráfica.
  - *pacf()* nos realiza exactamente lo mismo que el comando anterior pero en este caso para la función de autocorrelación parcial.
  - *arima()* ajusta el modelo arima que indiquemos a la serie de tiempo.

- *Box.test()* otra importante función que nos permite realizar contrantes de Box-Pierce y Ljung-Box para refutar si los residuos son un ruido blanco.

- Tseries: este paquete es otro de los que podemos encontrar en el repositorio CRAN y que sirve para el análisis de series de tiempo y finanzas computacionales. Una de sus funciones más importantes que utilizaremos será:
  - *adf.test(),* función a través de la cual podremos realizar el Test de Dickey-Fuller aumentado para poder comprobar si nuestra serie es estacionaria en media.
  - *pp.test()* nos sirve para realizar la prueba de raíz unitaria de Phillips-Perron.
  - *Jarque.bera.test()* se utiliza para realizar la prueba de normalidad de Jarque-Bera en los residuos de un modelo.
- Uroot*:* Contiene funciones para realizar contrastes de raíces unitarias en series de tiempo que contiene componente periódica.
- Locfit*:* Este paquete se utiliza para estimar densidades, y realizar pruebas y estudios de regresión y probabilidad.
- Astsa: Paquete de análisis estadístico aplicado a series de tiempo. Contiene funciones interesantes y muy útiles como:

  -*sarima()* que nos sirve para realizar gráficas de diagnostico del modelo y realizar estimaciones del ajuste.

3.2.2. Conjunto de datos objeto de nuestro estudio

En cuanto a nuestro conjunto de datos de estudio, hemos obtenido nuestra serie temporal a través del INE (Instituto Nacional de Estadística). El INE, en su página www.ine.es, dispone de una base con multitud de datos y series, además, posee una aplicación de consulta de series temporales para su banco de datos, la cual, permite con facilidad la búsqueda de estas y añadirlas a una cesta para su posterior descarga. En la siguiente imagen (FIGURA 15), se muestra el panel de selección con las pestañas que hemos seleccionado para encontrar nuestra serie de tiempo.

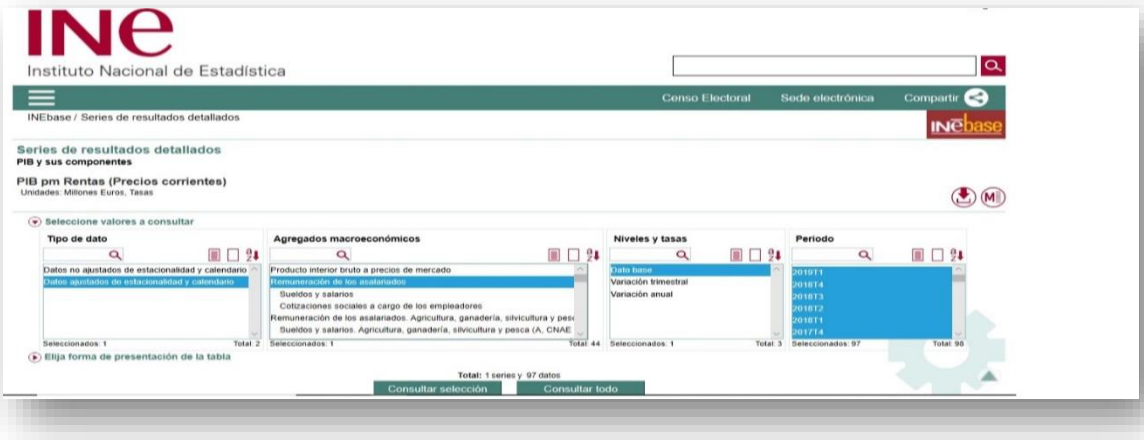


FIGURA 15

Una vez seleccionados, pulsaremos en *"Consultar selección"* y podremos visualizar la serie (FIGURA 16) y solo bastará con pulsar descargar y seleccionar en el tipo de archivo que la queremos (FIGURA 17). En nuestro caso hemos descargado un archivo Excell del tipo .xlm.

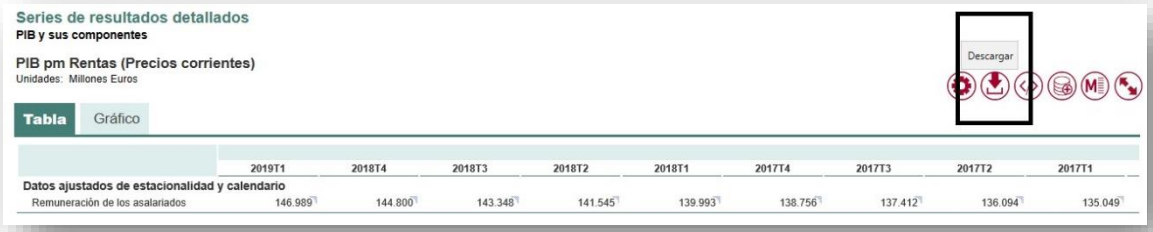


FIGURA 16

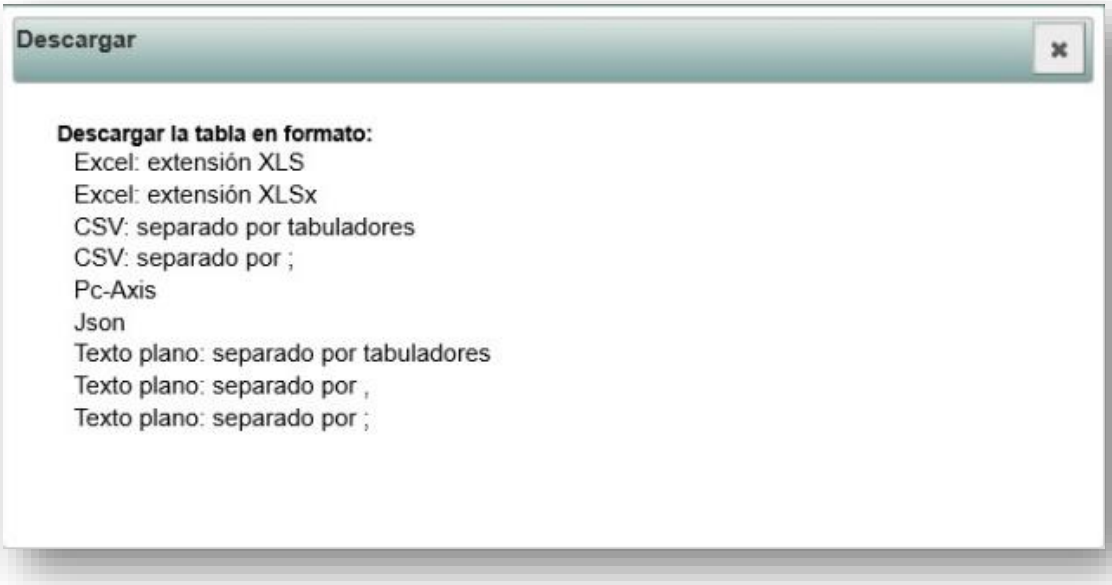


FIGURA 17

Hemos de indicar que, al descargar la serie, las observaciones vienen ordenadas de forma horizontal y empezando por el último año, por lo que, antes de proceder a realizar la importación en RStudio debemos de poner los datos en forma de columna y empezando desde el primer año de la serie, al contrario. También es importante saber que (como hemos explicado en el apartado 2.5) la primera fila se reserva para la cabecera y la primera columna se utiliza para identificar el dato.

Observando nuestra serie, podemos afirmar que se trata de una serie cronológica discreta que consta de 97 observaciones, debido a que las observaciones están tomadas en momentos de tiempo puntuales, en nuestro caso trimestralmente. Asimismo, y en vista

de que únicamente vamos a estudiar una variable concreta "Remuneración de los asalariados", podemos afirmar que estamos ante una variable univariante.

En cuanto a la serie de datos, supone la evaluación trimestral de la "Remuneración de los asalariados", esta información se utiliza en economía para el cálculo del PIB (Producto Interior Bruto). La Remuneración de los asalariados supone las cantidades que se destinan a pagar a los trabajadores a cambio de la prestación de sus servicios: sueldos y salarios, y contribuciones a la Seguridad Social. Esta variable sumada al *Excedente bruto de explotación* más *los Impuestos sobre la producción y las importaciones* suponen el Producto Interior Bruto calculado desde el **enfoque de las rentas**.

3.3. Ejemplo práctico: Aplicación de la metodología de Box & Jenkins a una serie temporal no estacionaria en RStudio

3.3.1. Convertir datos en objeto tipo "Serie temporal"

Desde el momento en que ya tenemos nuestros paquetes instalados y nuestro conjunto de datos importado, el primer paso antes de comenzar a graficar y realizar nuestro análisis, será convertir nuestros datos en un objeto del tipo serie temporal (ts). Esto se realiza debido a que esta no va a venir con este tipo de objeto y tendremos que indicárselo a RStudio a través de la función *"ts()"*. Este comando crea un objeto del tipo serie temporal por medio de un vector o una matriz, dependiendo de la naturaleza de la serie, es decir, si se trata de una serie univariante como la nuestra, este nos creará un vector y en el caso multivariante originará una matriz o archivo de datos según le especifiquemos en la función.

El comando "ts()" se compone de la siguiente configuración:

ts(data = NA , start = , end = , frequency = , deltat = , ts.eps. = getOption(“ts.eps”, class, names) con la siguiente argumentación:

data: En el siguiente argumento debemos apuntar el nombre del archivo de datos que queremos convertir en el objeto serie temporal.

start: En este argumento escribiremos el comienzo de la sere, por ejemplo si esta es trimestral, indicamos el año de inicio y el trimestre que es.

end: Aquí escribimos el final de nuestra serie temporal, de la misma forma que hemos explicado en el argumento *“start”*.

frequency: En este lugar se deben de escribir el numero de observaciones por unidad de tiempo, en otros términos, si nuestra serie de tiempo es trimestral, le pondremos el numero 4.

deltat: En este argumento se escribe una fracción del tipo de muestreo entre las diferentes observaciones de la serie, o lo que es lo mismo, si elegimos una fracción 1/12 estaremos indicando que la serie temporal objeto de estudio es mensual.

Cabe decir que, si utilizamos el argumento *“frecuency”* sobrará utilizar *“deltat”* debido a que ambos indican lo mismo, pero de diferentes maneras.

ts.eps: Se utiliza únicamente en el caso en que necesitemos tolerancia para comparar series temporales. Las frecuencias son tomadas todas de igual manera, si esta diferencia es inferior que *“ts.eps”*.

class: El siguiente argumento sirve para indicar la clase que le vamos a asignar a la serie. Si lo obviamos, se asignará la clase “ts” para una serie univariante. Si indicamos c(“mts”, “ts”) se asignara el objeto serie temporal multivariante.

names: El siguiente argumento se utiliza para escribir el nombre de las columnas en el caso en que estemos ante una serie multivariante. Si no escribimos nada, se utilizará el nombre de origen en nuestro archivo del conjunto de datos.

Después de especificar en esta función, vamos a proceder a aplicarla a nuestro conjunto de datos importado para convertirlo en objeto *“ts”* y comenzar a trabajar con dicho objeto durante todo el análisis. En nuestro caso, la función quedaría de la siguiente manera (FIGURA 18):

> ts(data = remuxlscolumna, start = c(1995,1), end = c(2019,1), frequency = 4,deltat = NA ,ts.eps = getOption("ts.eps"),class = ,names =)

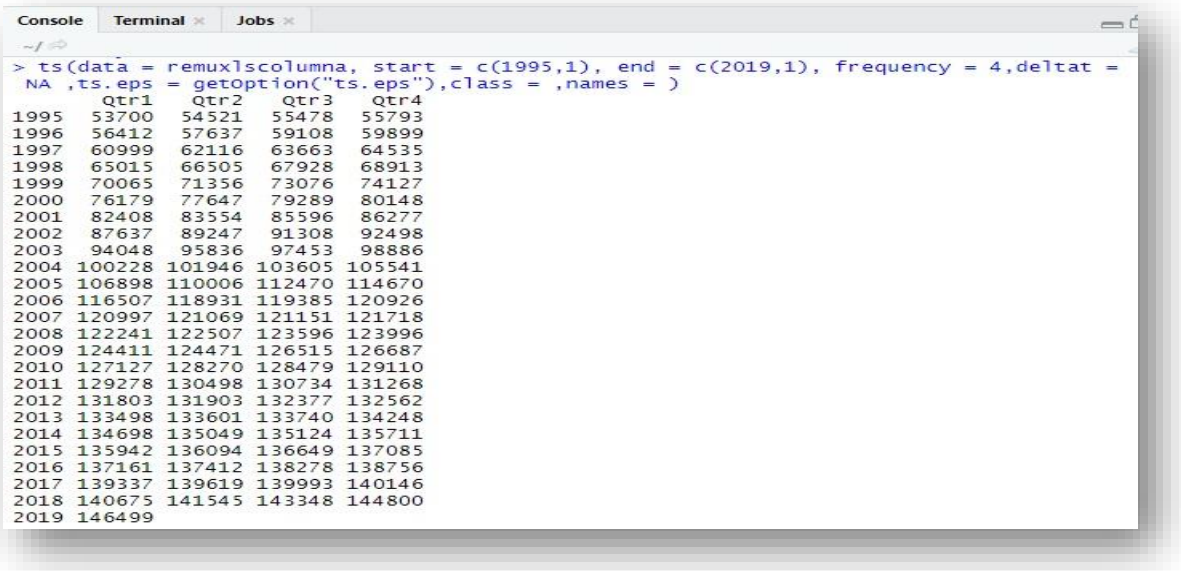


FIGURA 18

Como podemos observar en la FIGURA 18, en la función hemos indicado el nombre de nuestro conjunto de datos, el año y trimestre en que comienza, el año y trimestre en que termina y hemos indicado que las observaciones están tomadas en trimestres por lo que aparecen cuatro columnas.

En vista de que ya tenemos nuestro archivo de datos convertido en objeto del tipo serie temporal, vamos a proceder a otorgarle un nombre, ya que así nos facilitará el posterior uso de la misma. Para nombrar un vector o matriz, hay que hacerlo indicando el nombre que queremos darle acompañado por el carácter "<-" seguido de la función que tenemos creada para el objeto *"ts"*. En mi caso otorgaré el nombre *"salarios"* y la función quedaría de la siguiente forma (FIGURA 19):

>salarios<-ts(data = remuxlscolumna, start = c(1995,1), end = c(2019,1), frequency = 4,deltat = NA ,ts.eps = getOption("ts.eps"),class = ,names =

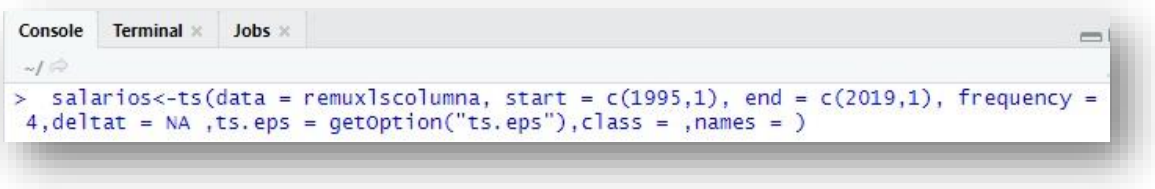


FIGURA 19

Una vez asignado el nombre, con solo escribirlo será suficiente para poder visualizar nuestro conjunto de datos y, además, podremos utilizarlo en cualquier otra función únicamente señalando el nombre **"salarios"**.

3.3.2. Realización de gráficas para el estudio de la serie e identificación.

Después de que hayamos pormenorizado nuestra serie, el primer paso para comenzar a analizar la serie y ajustar un modelo es plotear nuestra serie para observar cómo se comporta. Así podremos ver si esta presenta algún tipo de tendencia o componente estacional para así corregirlos con los diferentes métodos según la metodología de Box & Jenkins.

RStudio muestra un abanico bastante grande e interesante de gráficas, estas se mostrarán en la ventana inferior derecha de nuestro entorno, pulsando en el apartado *"Plots"* de esta, podremos observarlas, además de ampliarlas con el botón *"Zoom"* o guardarlas a través de *"Export"*. Si utilizamos la función *"demo(graphics)"*, esta función nos mostrará algunas de las gráficas que RStudio puede realizar.

Después de especificar las distintas acciones que ofrece el apartado de gráficas *"Plots"* de RStudio, vamos a proceder a describir el funcionamiento de la función que utilizaremos para plotear nuestra serie temporal.

En nuestro análisis vamos a realizar la gráfica a través de la función *"plot()*, ya que se trata de una función genérica y sencilla de RStudio que nos servirá para exponer nuestra serie temporal y poder interpretar su comportamiento. La función *"plot()"* se descompone en las siguientes partes:

plot (x, type, main, sub, xlab, xlim, ylim, col, asp) y cada argumento significa lo siguiente:

x: En el siguiente argumento debemos de escribir el nombre de la serie que queremos graficar en los ejes de abscisas y ordenadas.

type: el comando "type" sirve para determinar la apariencia de nuestro gráfico y dependiendo de la letra que pongamos nos lo hará de una forma u otra. Existen distintas formas que nombraremos a continuación:

- Si ponemos *"l"* en nuestro gráfico, los datos se encontrarán unidos por una única línea.

- Utilizando “p”, los datos del gráfico serán representados mediante círculos.

- “b” creará un grafico con datos representados por círculos que se encuentran conectados entre si por líneas.

- El argumento “o” nos indicará un gráfico en el cual los datos se cuentran representados mediante círculos que a su vez se encuentran ensablados por líneas, pero en este caso, las líneas se verán por encima de estos círculos.

- “h” se trata de un gráfico el cual los datos se encuentran representados por líneas verticales.

- “s” realiza un plot en el cual los datos se representan por líneas horizontales a modo de peldaños de escalera y en este caso, muestra la parte superior de cada peldaño.

- “S” mayúscula se tratará de un gráficos como el anterior, es decir, con los datos representados con peldaños pero en este caso, nos muestra la parte inferior de ellos.

main: sirve para indicar el título de la serie en el gráfico que estamos ploteando.

sub: se utiliza para escribir un subtitulo en nuestro gráfico que aparecerá en una letra más pequeña, debajo del título principal.

xlab: En el siguiente argumento se escribe el nombre que queremos darle al eje X.

ylab: En este caso sirve para introducir un nombre para nuestro eje Y.

xlim: En este argumento podemos indicar el límite de nuestro eje X.

ylim: Este argumento sirve para escribir el límite del eje Y.

col: Sirve para indicar el color que queremos darle a las diferentes partes del gráfico, un ejemplo seria “col.main”, “col.sub” o “col.lab” etcétera. A la hora de elegir el color que deseamos, “RStudio” dispone de una lista donde están todos los colores que se puedene elegir, esta podemos observarla mediante la función “colors()”. Después de implementarla podremos ver una lista con más de 600 colores para seleccionar el que deseamos para nuestros gráficos.

asp: Este comando sirve para indicar la relación o ratio de aspecto entre las variables *x* e *y*.

Después de ahondar en la función de forma detallada, vamos a proceder a plotear nuestro conjunto de datos, ya convertidos en objeto serie temporal y la función quedaría de la siguiente manera:

>plot (x = salarios, type="b", main= "Salario de los trabajadores", sub=NA, xlab = "Años", ylab= "Salario", col= "steelblue1")

Como podemos observar en la función que hemos introducido, hemos introducido el nombre de nuestra serie, le hemos indicado que queremos nuestro gráfico con los datos unidos por círculos y hemos indicado el nombre de los ejes de abscisas y ordenadas además del color que queremos darles a los círculos.

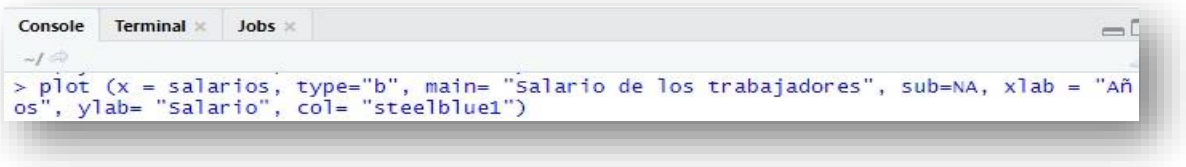


FIGURA 20

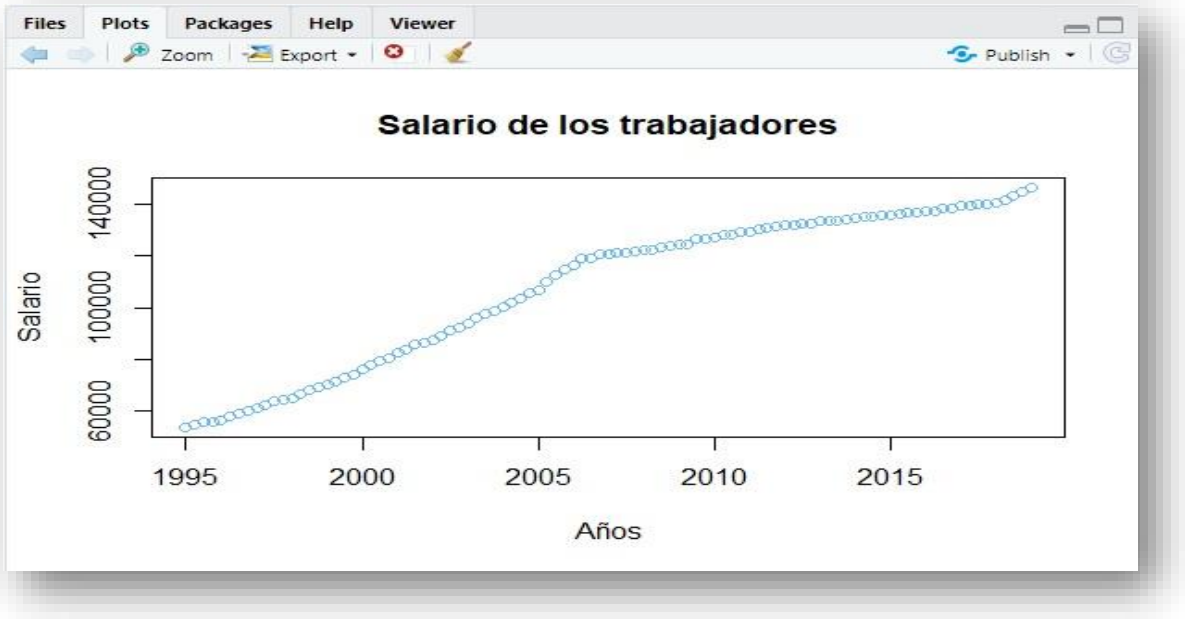


FIGURA 21

Observando nuestra serie graficada (FIGURA 21), podemos comprobar que presenta una tendencia creciente, esto se debe a que los salarios ven incrementándose gradualmente cada año. También se contempla que los datos no oscilan en torno a un valor constante, es decir, la serie no es estacionaria en media y deberemos aplicar diferencias para hacerla estacionaria y eliminar la tendencia que presenta.

Una vez representada nuestra serie, el siguiente paso es figurar la función de autocorrelación simple (f.a.s) y la función de autocorrelación parcial (f.a.p) a través de las funciones *"acf()"* y *"pacf()"* que sen encuentran en el paquete *stats*, ya cargado anteriormente.

La primera de estas dos funciones en la que ahondaremos será la *"acf()"*, que se eucuentra constituida de la siguiente forma:

acf(x, y, lag.max = NULL, type = c("correlation" , "covariance", "partial"), main, xlab, col) y se descompone de la siguiente forma:

x, y: El siguiente argumento indicaremos nuestro objeto de tipo serie temporal, este puede ser univariante, como en nuestro caso, o multivariante, además puede ser un objeto del tipo "acf", vector o matriz.

lag.max: El siguiente argumento sirve para indicar el retardo hasta el que queremos que nos calcule la función. Si no completamos el argumento, RStudio nos aplica por defecto la fórmula **10*log10 (M/m),** en la cual *"M"* significa el numero total de observaciones de la serie y *"m"* el número de series.

type: Sirve para indicar el tipo de función que calculará. Por defecto el valor es *"correlation"* aunque también podemos indicar *"covariance"* o *"partial"*.

main: Vale para indicar el enunciado del gráfico.

xlab: Argumento que sirve para escribir el nombre del eje X.

col: Como hemos indicado en funciones anteriores, servirá para indicar el color que queremos dar a las partes del gráfico.

Para representar la función de autocorrelación parcial (f.a.p) utilizaremos el comando *"pacf()"* que se estructura de la siguiente forma:

pacf(x, y, lag.max = NULL, type = c("correlation" , "covariance", "partial"), main, xlab, col) y sus argumentos se descomponen de igual forma que la función "acf()" explicada en el apartado anterior.

A continuación, ya que conocemos el funcionamiento de las funciones, vamos a proceder a plotear las funciones de autocorrelación simple y parcial.

Para el cálculo de la función de autocorrelación simple:

>acf(x=salarios, main="Función de Autocorrelación Simple", lag.max = 25, col="steelblue1",xlab =,ylab=)

```
Console   Terminal ×   Jobs ×
~/
> salarios<-ts(data = remuxlscolumna, start = c(1995,1), end = c(2019,1), frequency =
 4,deltat = NA ,ts.eps = getOption("ts.eps"),class = ,names = )
```

FIGURA 22

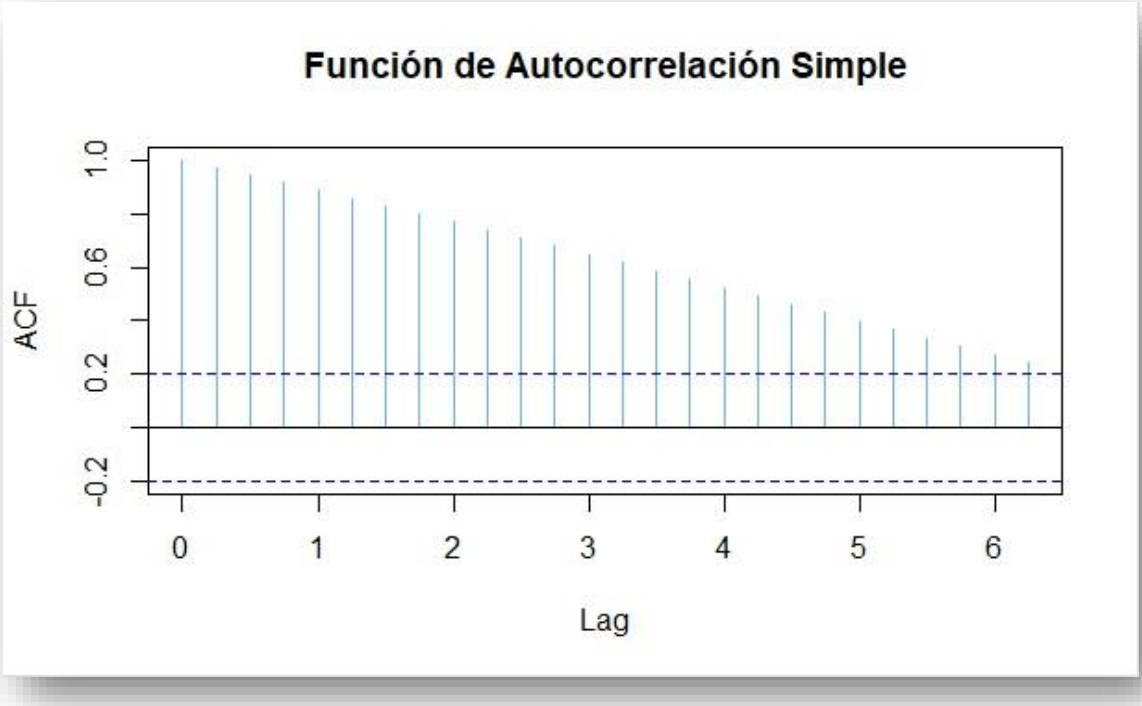


FIGURA 23

Para calcular la función de autocorrelación parcial;

>pacf(x=salarios, main="Función de Autocorrelación Parcial",lag.max = 25, col="steelblue1",xlab =,ylab=)

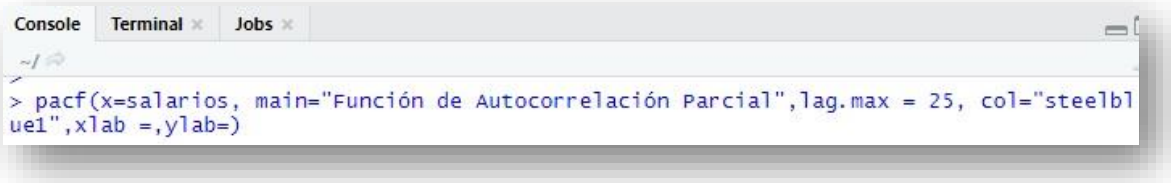


FIGURA 24

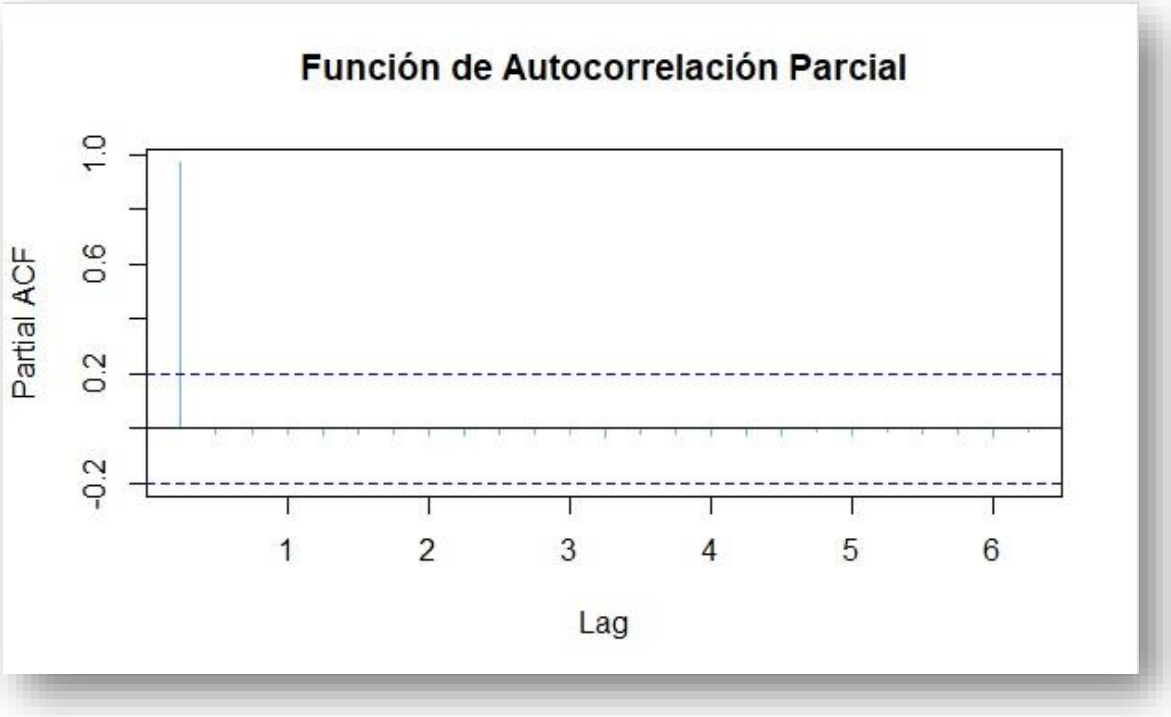


FIGURA 25

La FIGURA 23 nos vuestra la gráfica de nuestra Función de Autocorrelación Simple en la que podemos observar un patrón que se observa en los casos en que la serie

presenta tendencia y no es estacionaria: la barras disminuyen lentamente. Esto nos indica que tendremos que aplicar diferencias para corregir la serie y hacerla estacionaria.

3.3.3. Corrección de la serie: aplicación de diferencias.

Para corregir el problema de no estacionariedad en media detectado, primeramente, realizaremos un contraste para cerciorarnos de forma analítica si la serie es o no estacionaria, el **"Test de Dickey-Fuller aumentado"** nos indicará las veces que tenemos que diferenciar según el número de raíces unitarias que tenga la serie en estudio. En este contraste, la hipótesis nula es que el coeficiente de autocorrelación es igual a cero y la hipótesis alternativa de que dicho coeficiente es estrictamente menor que uno.

El siguiente test lo realizaremos a través de "RStudio" mediante la función *"adf.test()"* que pertence al paquete *Tseries* y se ejecuta de la siguiente forma:

adf.test(x, alternative=c("stationary", "explosive"), k) y, cada argumento nos indica lo siguiente:

x: Sirve para definir la serie temporal objeto de estudio.

alternative: Sirve para eligir la hipótesis alternativa en el test, por defecto quedará asignada como estacionaria.

k: El siguiente argumento se utiliza para definir el orden del retardo que queremos que calcule en la prueba, por defecto tiene asignada la función trunc((lenght(x)-1)^(1/3))).

A continuación, vamos a proceder a realizar el test de Dickey-Fuller para nuestra serie:

> adf.test(salarios)

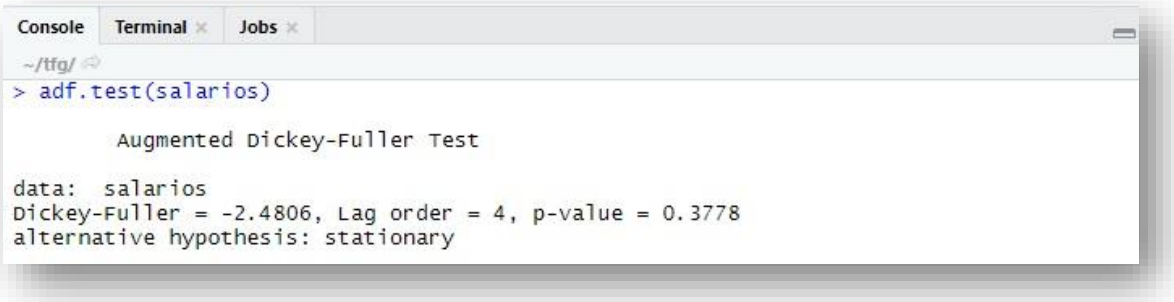


FIGURA 26

Si observamos el p-valor (0,3778) en la FIGURA 26, este nos dice que no se puede rechazar la hipótesis nula, por lo que la serie tiene raíz unitaria y no es estacionaria como hemos observado en las gráficas anteriores. Esto nos hace indicar que debemos de llevar a cabo diferencias para eliminar la tendencia y hacer estacionaria la serie para su posterior ajuste.

Otro contraste que podemos utilizar para refutar esta hipótesis es el **"Test de Phillips-Perron"**, este test consiste en una modificación del Test de Dickey-Fuller que corrige la autocorrelación y heterocedasticidad en los errores. Se implementa a través de la función *"pp.test()"* y para nuestra serie nos proporciona los siguientes resultados:

> pp.test(salarios)

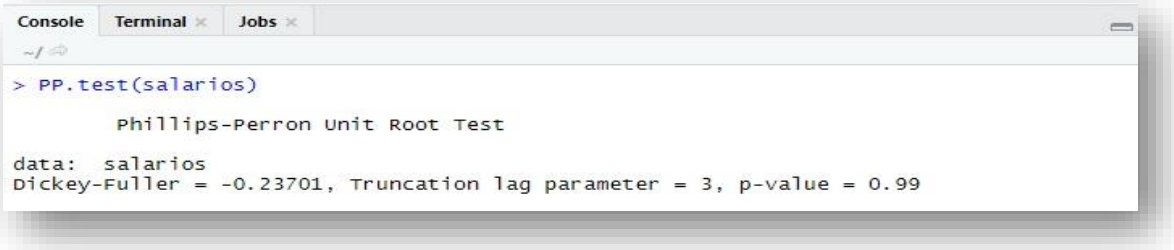


FIGURA 27

Con este test (FIGURA 27), se obtuvo un p-valor de 0.99 y, por tanto, no rechazamos la hipótesis nula y tenemos evidencias para afirmar que la serie tiene una raíz unitaria y no es estacionaria.

Como hemos podido observar en los gráficos anteriores (FIGURAS 23 Y 25), y hemos contrastado posteriormente (FIGURAS 26 y 27), nos encontramos ante una serie temporal no estacionaria y debemos aplicar diferencias para hacerla estacionaria. Esto es que, cuando una serie presenta tendencia, es preciso quitarla mediante una diferencia regular. Una diferencia regular consiste en restar a cada elemento de la serie la observación anterior.

Si además de la tendencia, la serie presentara estacionalidad, sería necesario realizar transformaciones del tipo logarítmica o familia de Box-Cox, pero en nuestro caso únicamente necesitaremos aplicar diferencias para corregir la estacionariedad en media.

La función que se utiliza para llevar a cabo las diferenciaciones en series temporales es *"diff()"*. Esta función se encuentra en el paquete base de análisis de R y consta de las siguientes partes:

diff(x, diff=1) y se descompone en los siguiente argumentos:

x: En el siguiente argumento hay que escribir el nombre de nuestro archivo de la serie con objeto "ts", vector o matriz sobre la que queremos aplicar las diferencias.

diff: Aquí debemos escribir un numero entero, este argumento indicará el orden de diferenciación que queremos aplicar a la serie.

A continuación, procedemos a aplicar una diferenciación de **orden 1** a nuestra serie y a graficarla para observar si conseguimos corregir el problema de no estacionariedad, en el caso de no corregirse, deberemos realizar una segunda diferenciaciación.

>salariosdif1<-diff(salarios)

> plot (x = salariosdif1, type="l", main= "Salario de los trabajadores", sub=NA, xlab = "Años", ylab= "Salario", col= "steelblue1")

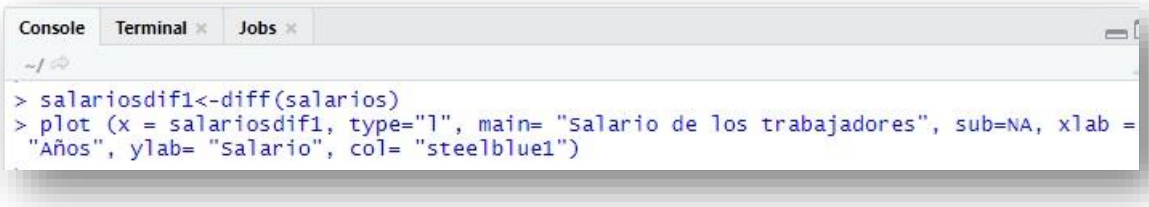


FIGURA 28

FIGURA 29

Las gráficas de la función de autocorrelación simple y parcial tras la **diferenciación de orden 1** quedarían de la siguiente forma (FIGURAS 31 Y 33):

> acf(x=salariosdif1, main="Función de Autocorrelación Simple", lag.max = 25, col="steelblue1",xlab =,ylab=)

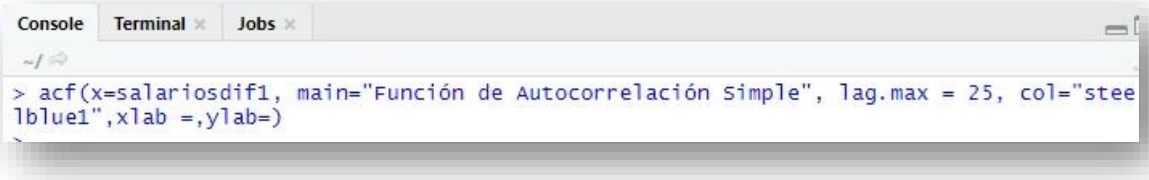


FIGURA 30

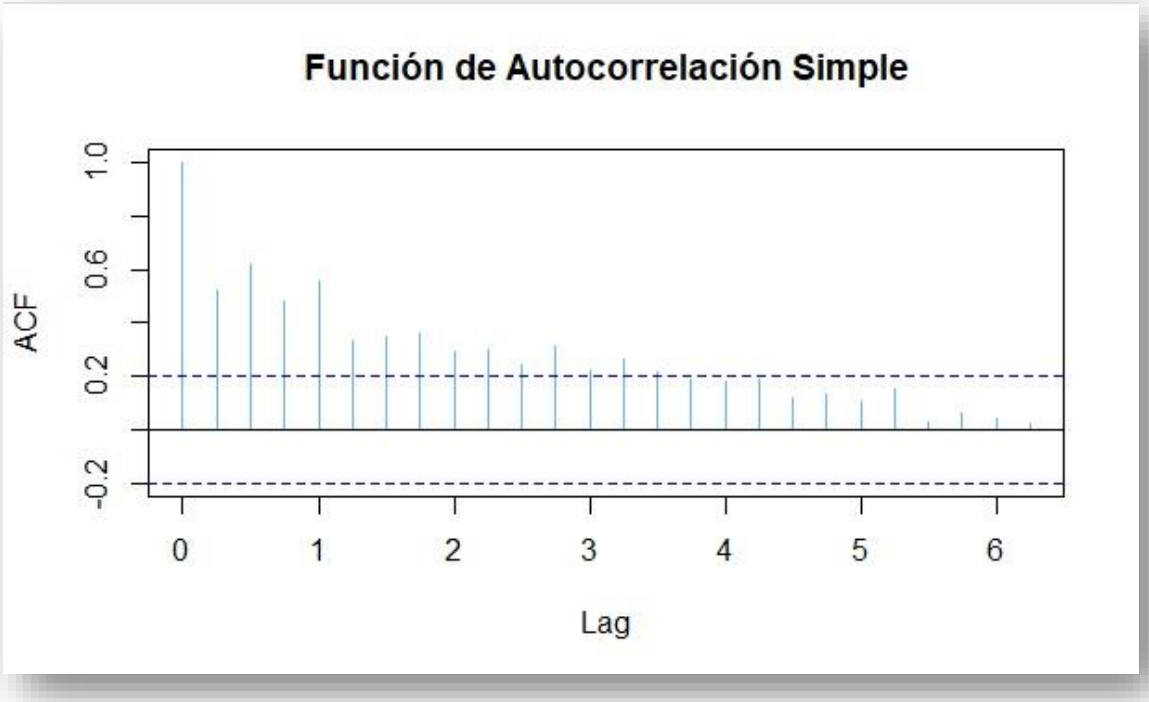


FIGURA 31

> pacf(x=salariosdif1, main="Función de Autocorrelación Parcial",lag.max = 25, col="steelblue1",xlab =,ylab=)

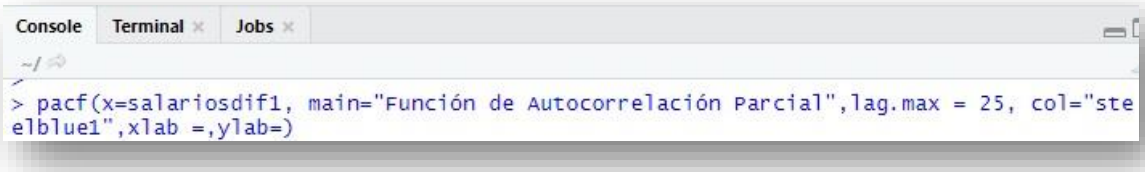


FIGURA 32

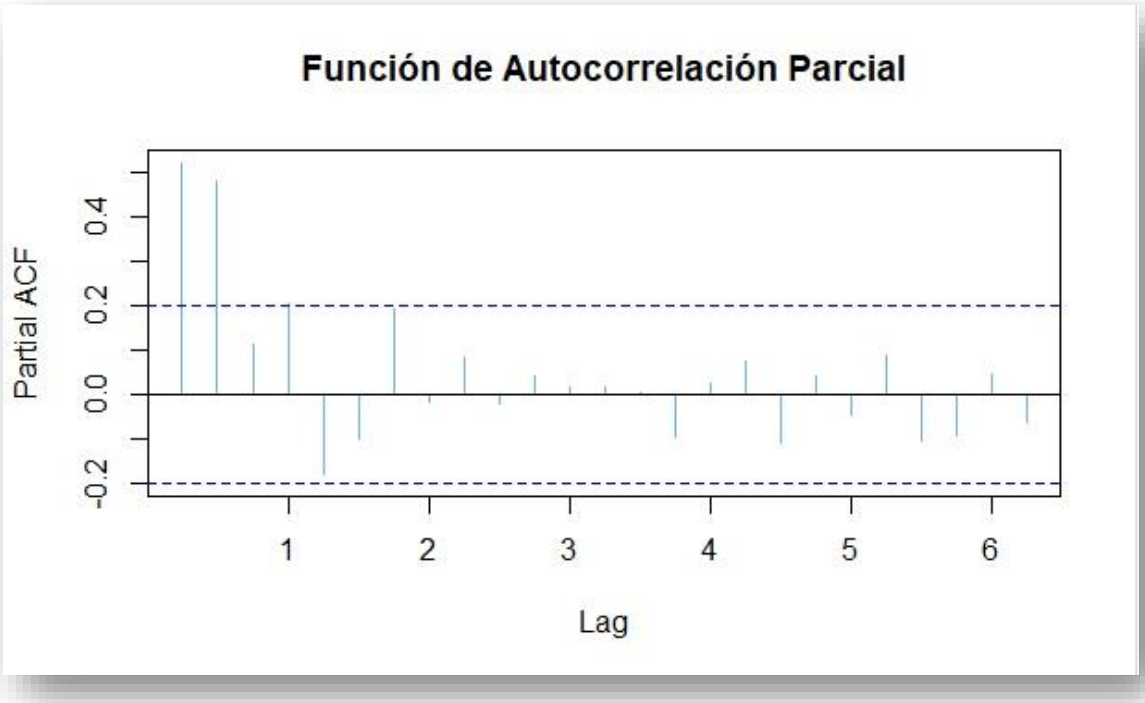


FIGURA 33

Si observamos las gráficas para la serie con una diferenciación de orden 1, podemos decir que el problema de no estacionariedad en media sigue sin resolverse. Observando la gráfica para la serie con la diferenciación de orden 1 (FIGURA 29), podemos ver que los datos siguen sin oscilar en torno a un valor constante por lo que debemos de realizar una segunda diferenciación. No obstante, y para cerciorarnos mejor,

realizaremos la prueba de *"Dickey-Fuller"* para contrastar de forma más analítica la hipótesis de estacionariedad.

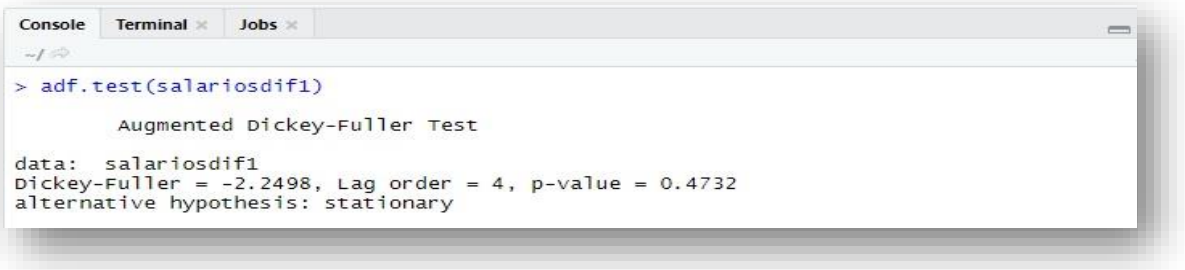


FIGURA 34

Como se puede observar en la FIGURA 34, el test nos proporciona un p-valor de **0,4732**, por lo que no podemos rechazar la hipótesis nula y, como apuntábamos observando las gráficas de la serie con una diferenciación de orden 1, no podemos afirmar que se corrija el problema de no estacionariedad en media y deberemos diferenciar la serie una segunda vez para intentar solventar el problema.

En vista de lo anterior, vamos a pasar a realizar una diferenciación de **orden 2** en nuestra serie temporal para comprobar si así conseguimos corregir el problema de no estacionariedad en media.

En la FIGURA 36 observamos la gráfica de nuestra serie tras aplicarle una diferenciación de orden 2:

```
> salariosdif2<-diff((salarios),differences = 2)
> plot (x = salariosdif2, type="l", main= "Salario de los trabajadores", sub=NA, xlab = "Años", ylab= "Salario", col= "steelblue1")
```

```
> salariosdif2<-diff((salarios),differences = 2)
> plot (x = salariosdif2, type="l", main= "Salario de los trabajadores", sub=NA, xlab =
 "Años", ylab= "Salario", col= "steelblue1")
```

FIGURA 35

FIGURA 36

La función de autocorrelación simple quedaría de la siguiente forma (FIGURA 38):

> acf(x=salariosdif2, main="Función de Autocorrelación Simple", lag.max = 25, col="steelblue1",xlab =,ylab=)

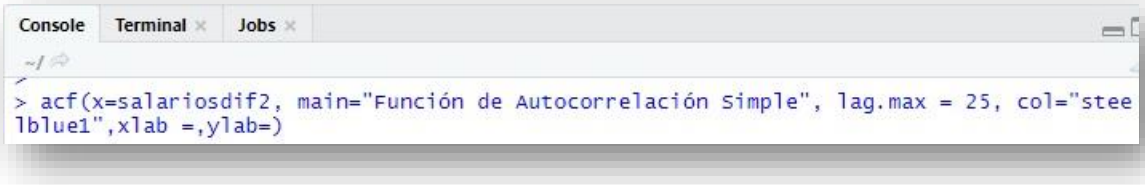


FIGURA 37

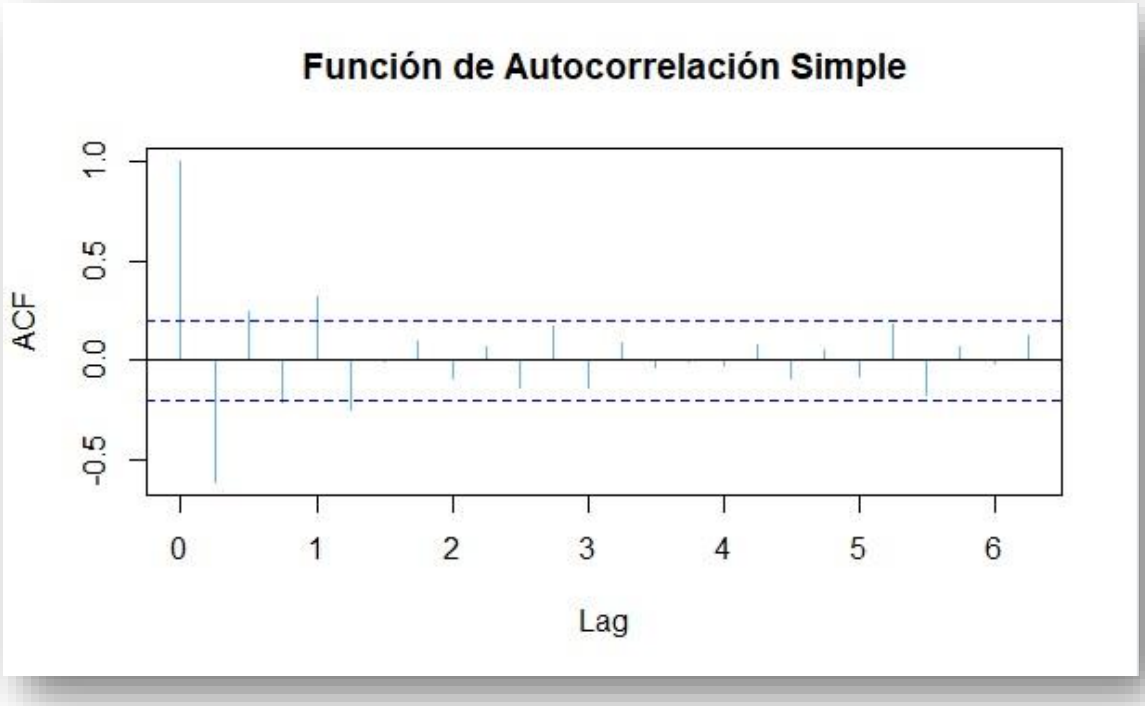


FIGURA 38

Y, por último, la función de autocorrelación parcial para la serie con una diferenciación de **orden 2** aplicada quedaría así (FIGURAS 39 Y 40):

> pacf(x=salariosdif2, main="Función de Autocorrelación Parcial",lag.max = 25, col="steelblue1",xlab =,ylab=)

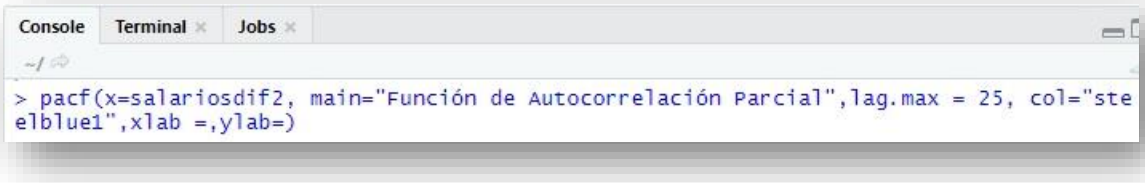


FIGURA 39

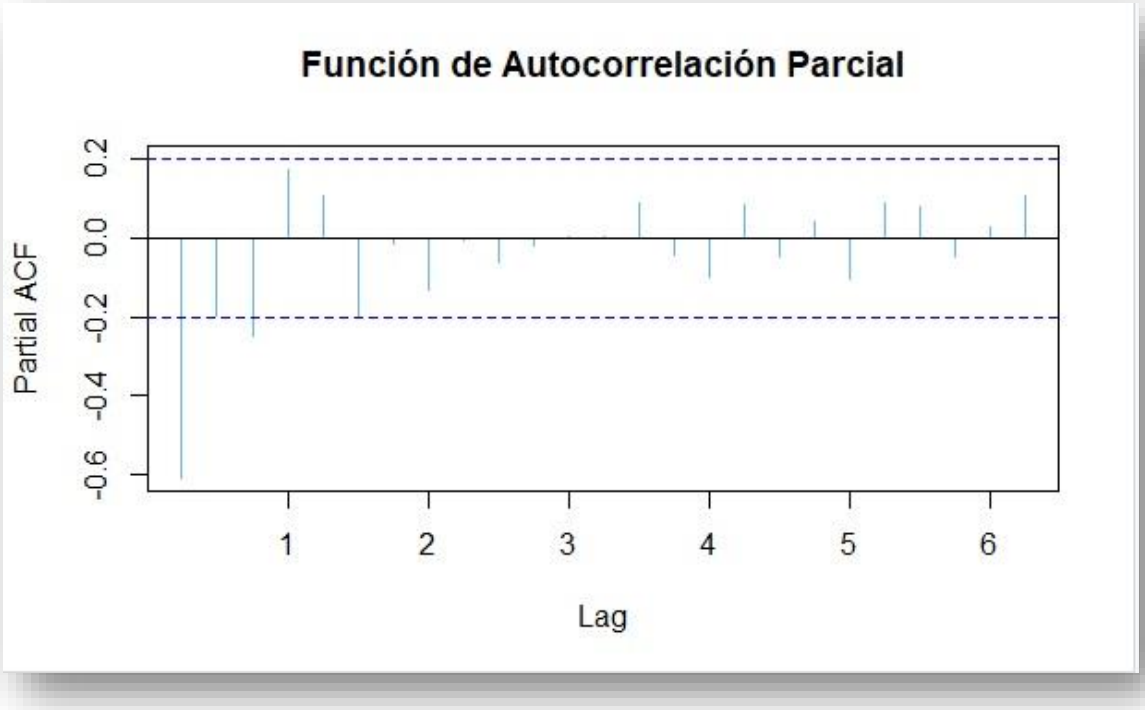


FIGURA 40

Finalmente, tras realizar una segunda diferenciación, ya podemos observar que los datos oscilan en torno a un punto medio en el gráfico (FIGURA 36) y se puede certificar que el problema de no estacionariedad en media se ha corregido. No obstante, a pesar de afirmarlo observando la gráfica, vamos a volver a realizar el *"Test de Dickey-Fuller aumentado"* que hemos venido utilizando para contrastar la hipótesis de estacionariedad.

> adf.test(salariosdif2)

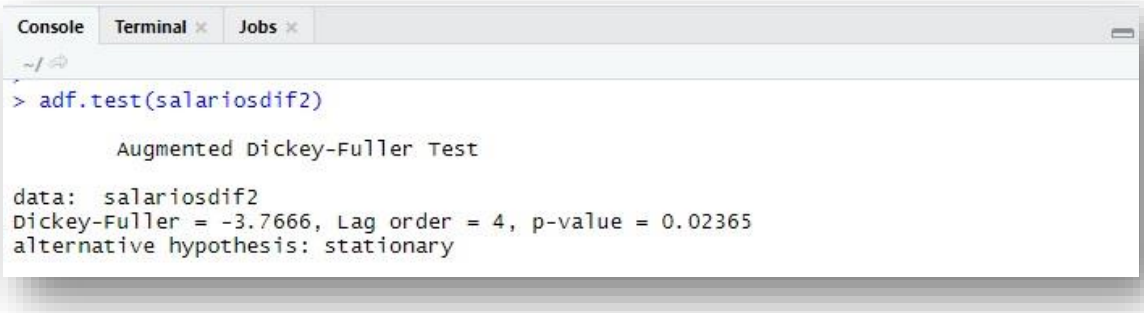


FIGURA 41

Como observamos en la FIGURA 41, el p-valor obtenido es **0,0236** y, en consecuencia, existen evidencias para rechazar la hipótesis nula, la serie se ha corregido y ya es estacionaria en media. Por lo que, de acuerdo al enfoque moderno del análisis de series cronológicas, implementado por Box & Jenkins, que nos indica que la serie tiene que ser estacionaria al menos en media, ya podemos proceder a ajustar un Modelo ARIMA.

3.3.4. Estimación

En el siguiente punto, una vez hemos corregido los problemas de estacionariedad en media que presentaba la serie, entre los todos los modelos, observando las gráficas de la f.a.s y f.a.p, hemos seleccionado un modelo ARIMA (0,2,2). Para comprobar si este modelo es el que mejor ajuste nos proporciona, llevaremos a cabo la estimación a traves de la función *"sarima()"* del paquete *astsa*.

La función "sarima()" presenta la siguiente estructura:

sarima(x, order=c(0,0,0)) con los siguientes argumentos:

x: En esta parte debemos de escribir le nombre de nuestros conjunto de datos objeto de estudio, que puede ser una serie univariante o multivariante.

order: En este argumento debemos de escribir los términos (p, d, q) del modelo ARIMA que hemos elegido, estas representan a los términos de la parte autorregresiva (p), grado de la diferenciación de I (d) y el orden de las medias móviles (q).

La función consta de algunos más argumentos para casos en que nuestra serie presente componente estacional, pero nosotros no los trataremos, ya que este documento se realiza para series no estacionales.

> sarima(salarios, 0,2,2)

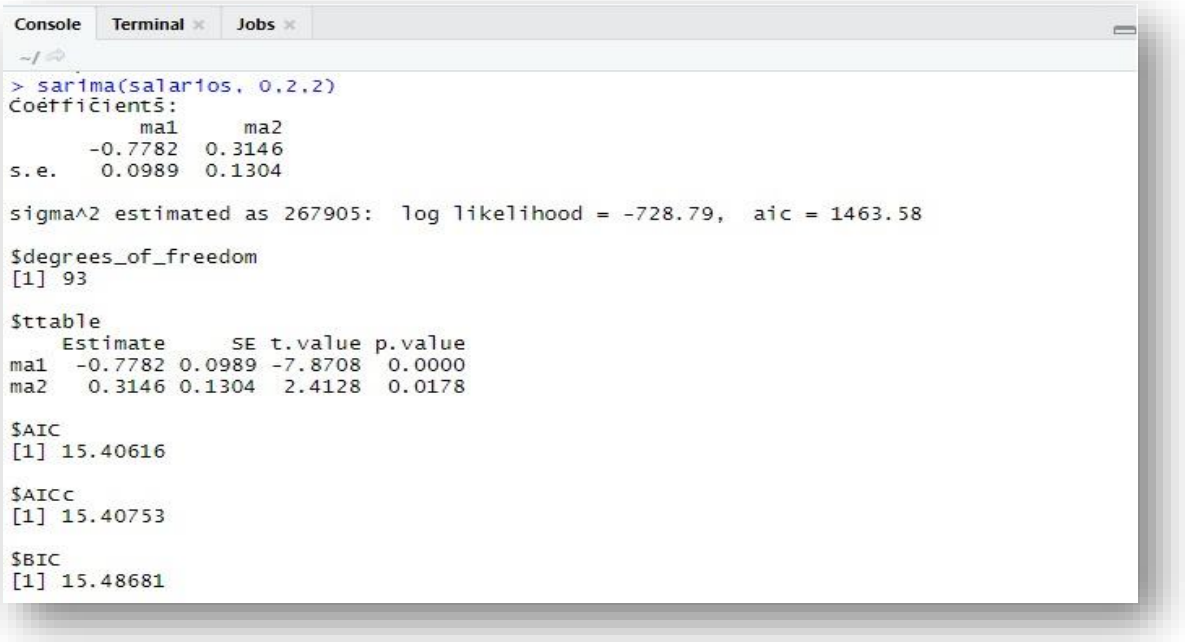


FIGURA 42

Si observamos los resultados de la estimación (FIGURA 42), podemos ver los valores estimados para nuestro modelo, los errores estándar y, además, podemos comprobar que

todos los términos son apropiados, ya que los p-valor son significativos. Esta estimación también nos proporciona los criterios los AIC, AICc y BIC.

3.3.5. Diagnosis

Para saber si nuestro modelo ARIMA (p, d, q) es adecuado, este debe de cumplir de manera apropiada la fase de identificación. Si hemos identificado bien nuestra serie, y el modelo ajustado es correcto, los residuos no pueden tener ninguna estructura, es decir, deben de ser completamente aleatorios, cuando esto se cumple, podemos hablar de que son un ruido blanco.

Un ruido blanco no es más que una serie estacionaria en la que todas las observaciones son independientes y sus f.a.s y f.a.p son cero, es decir, ninguna barra se sale de las bandas de confianza.

La función *"sarima()",* explicada que en apartado 3.3.4, además de realizarnos la estimación de nuestro modelo, nos realiza una serie de gráficos con los cuales podemos refutar estas hipótesis de residuos de manera visual.

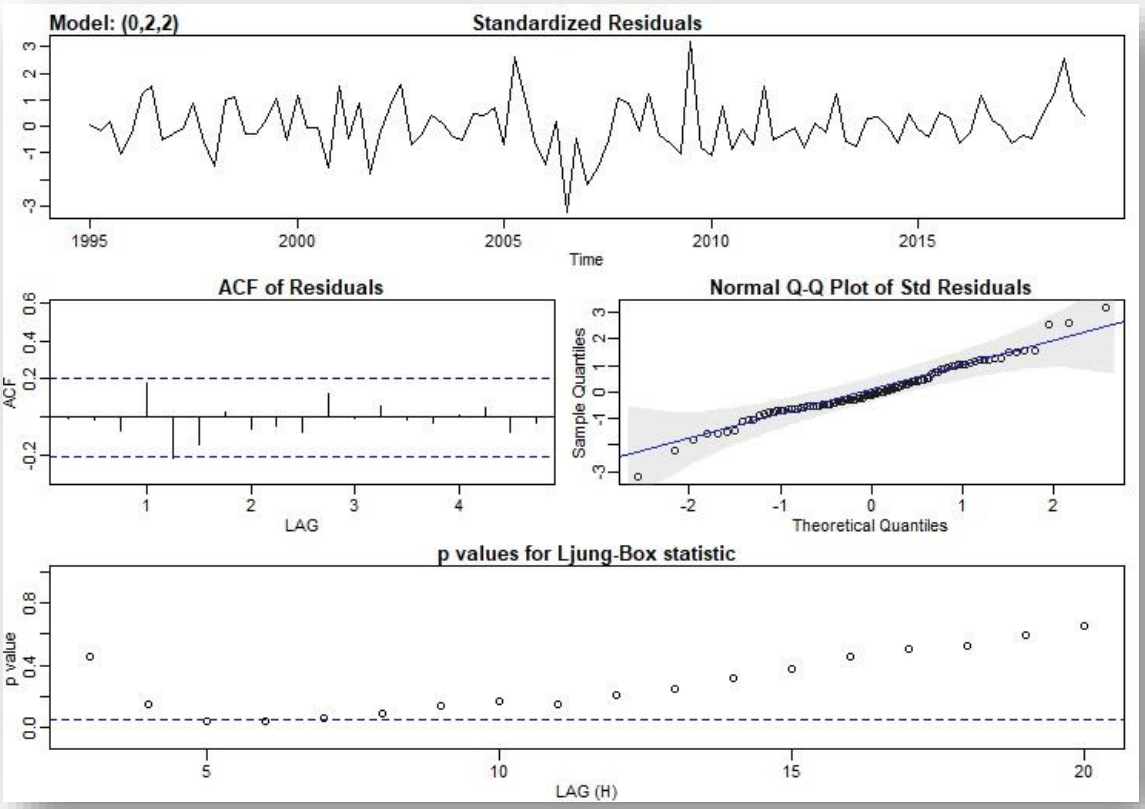


FIGURA 43

Si atendemos a las gráficas de residuos para nuestro modelo ARIMA(0,2,2) (FIGURA 43), podemos deducir lo siguiente:

- ✓ En la primera grafica observamos que los residuos estandarizados parecen ser totalmente independientes, ya que se comportan aleatoriamente alrededor del cero.
- ✓ En la segunda gráfica, podemos ver que la función de autocorrelación simple es un ruido blanco, ya que ninguna barra se sale de las bandas de significación y es nula.
- ✓ En la tercera gráfica tenemos un Q-Q Plot para contrastar la hipótesis de normalidad.
- ✓ En la cuarta gráfica podemos ver que para el "test de ruido blanco para los residuos de Ljung-Box" podemos aceptar la hipótesis nula de la independencia de los residuales. Esto es debido a que el valor del p-valor es mayor que 0,5.

No obstante, además de cerciorarnos visualmente de que todos los ajustes están bien hechos, realizaremos un test de ruido blanco a través de la función *"Box.test()"* del paquete *stats*. Para realizar el test debemos de indicarle que utilice los residuos del ajuste de la serie y especificarle que el tipo de prueba sea la de "Ljung-Box".

>ajuste2<-sarima(salarios, 0,2,2)

>Box.test(residuals(ajuste2$fit),type = "Ljung-Box")

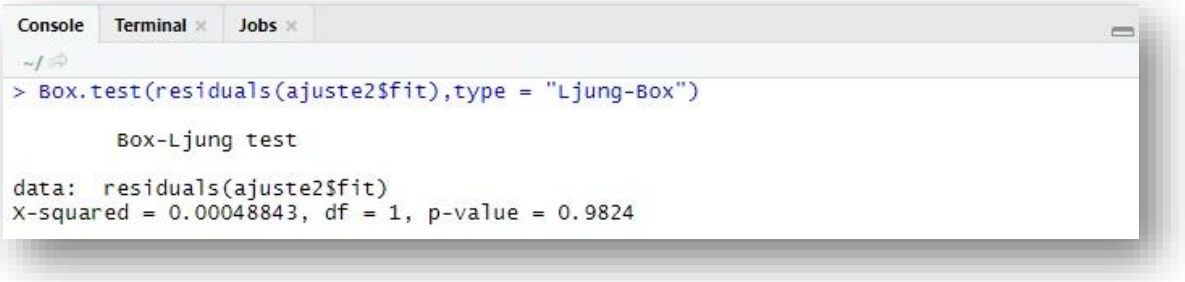


FIGURA 44

El test de "Ljung-Box para los residuos nos da un p-valor de 0.9824 (FIGURA 44), por lo que se puede interpretar que no existen evidencias para rechazar que los residuos sean un ruido blanco. Este test nos indica la existencia de problemas cuando el p-valor es menor que 0.05. Al nuestro p-valor ser muy alto, la evidencia de que los residuos son un ruido blanco es muy alta por lo que podemos asumir que el modelo ARIMA(0,2,2) es adecuado para modelizar la serie temporal "Remuneración de los asalariados".

3.3.6. Verificación

En esta siguiente fase del proceso, mostraremos un resumen de nuestro modelo en el que podremos ver diferentes estadísticos de los errores que se utilizan como criterios a la hora de seleccionar un modelo. Los estadísticos que RStudio nos mostrará a través de

la función *"summary()"* serán **ME, RMSE, MAE, MPE, MAPE y MASE**. Es interesante elegir siempre el modelo que conjunta mente tenga menores valores de estos.

Para realizarlo, primero debemos dar un nombre a nuestro modelo y posteriormente ejecutar la función "summary()" como se muestra a continuación:

>model1<-arima(x=salarios,order=c(0,2,2))

> summary(model1)

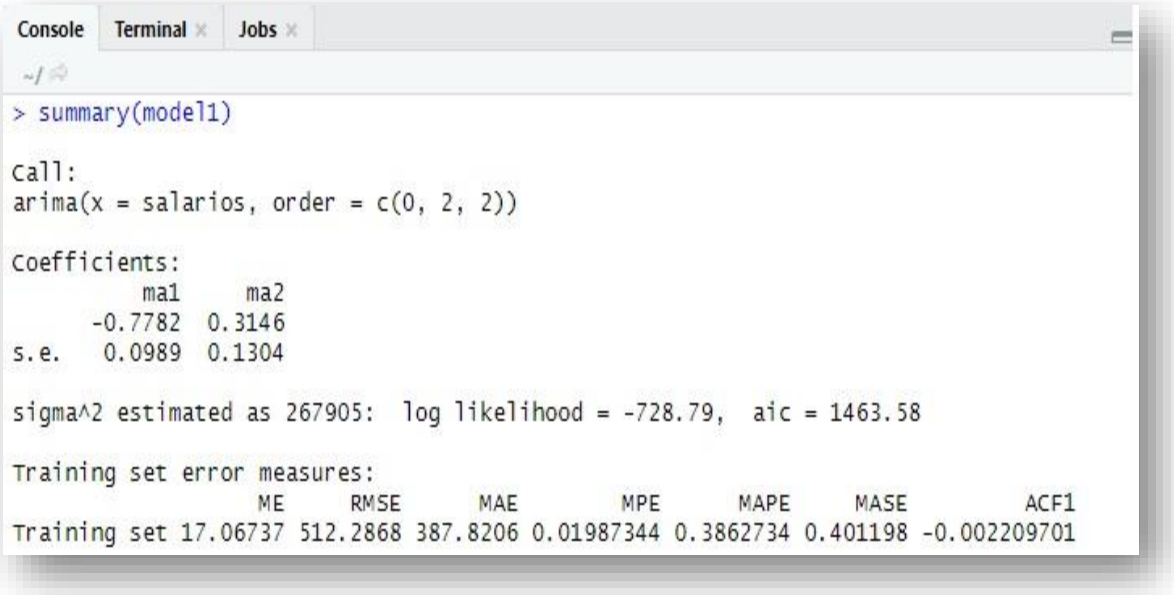


FIGURA 45

3.3.7. Predicción

La predicción es la última fase y el principal objetivo de la metodología de Box & Jenkins, esto consiste en, una vez que hemos corroborado que el modelo ajustado es correcto, lo vamos a utilizar para predecir los valores futuros de la serie temporal objeto de estudio.

Para realizar predicciones en modelos ARIMA(p,d,q) a través de RStudio existen diferentes funciones, en nuestro caso utilizaremos la función *"forecast()",* que se encuentra en el paquete *"Atsa".* Esta función consta forma:

forecast(object, h= 5, …) y en cada argumento debemos informar lo siguiente:

object: Sirve para escribir el nombre del modelo que hemos ajustado o también podemos escribir el nombre de la serie la cual queremos predecir.

h: Aquí indicaremos el número de periodos sobre los que queremos que nos realice la predicción.

La función quedaría de la siguiente forma:

```
> forecast::forecast(model1,h=13)
```

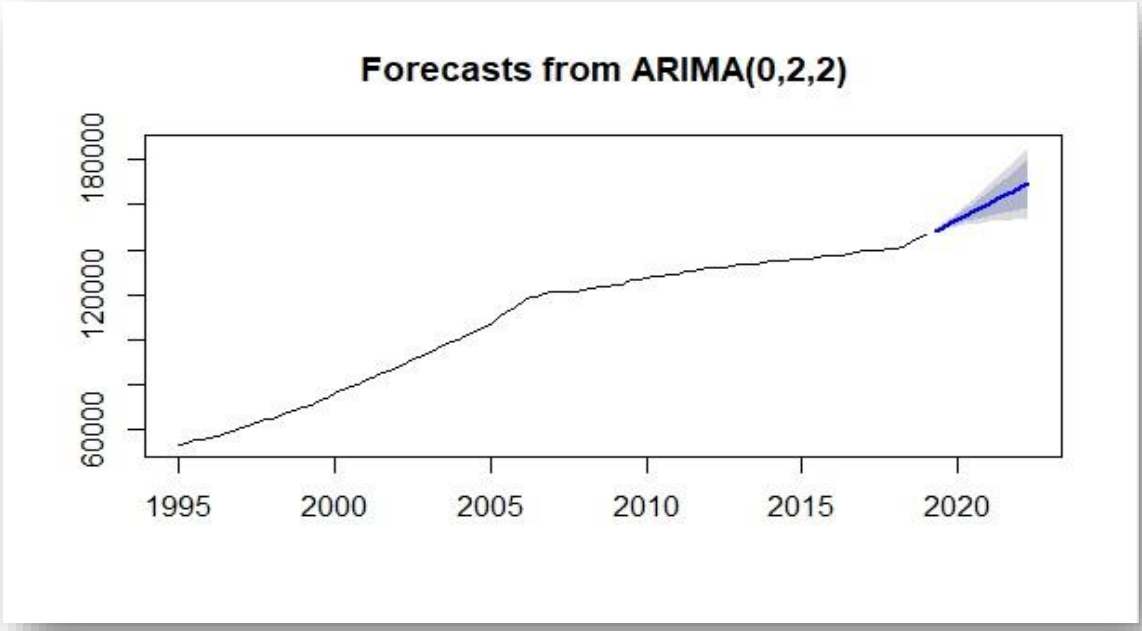
```
Console   Terminal ×   Jobs ×
~/
> forecast::forecast(model1,h=13)
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2019 Q2       148187.4 147524.1 148850.8 147173.0 149201.9
2019 Q3       149942.8 148895.5 150990.1 148341.1 151544.5
2019 Q4       151698.2 150130.7 153265.7 149301.0 154095.4
2020 Q1       153453.6 151268.7 155638.4 150112.1 156795.0
2020 Q2       155208.9 152328.0 158089.8 150803.0 159614.9
2020 Q3       156964.3 153318.9 160609.7 151389.2 162539.4
2020 Q4       158719.7 154248.3 163191.1 151881.2 165558.1
2021 Q1       160475.0 155120.8 165829.3 152286.5 168663.6
2021 Q2       162230.4 155940.4 168520.4 152610.7 171850.1
2021 Q3       163985.8 156710.1 171261.5 152858.6 175113.0
2021 Q4       165741.2 157432.4 174050.0 153034.0 178448.4
2022 Q1       167496.5 158109.4 176883.7 153140.1 181852.9
2022 Q2       169251.9 158743.1 179760.7 153180.0 185323.8
```

FIGURA 46

Como podemos observar en la FIGURA 46, la función nos devuelve los valores predichos para la serie y los intervalos de confianza para los 13 periodos.

A continuación, procederemos a realizar una representación gráfica de los valores predichos:

>plot(forecast::forecast(model1,h=13))



Sobre la línea azul oscura se representan los valores predichos, que se encuentran dentro de los límites de confianza al 80% y 95%, también podemos observar que, cuanto más en el tiempo hacemos las predicciones, las bandas de confianza se van ampliando.

3.4. Conclusión

Como hemos observado a lo largo del documento, la metodología de Box & Jenkins nos otorga una manera sencilla y eficaz de ajustar series de tiempo de manera rápida y relativamente sencilla para realizar predicciones fiables a cerca de su evolución futura.

Rstudio es un paquete gratuito que sirve para realizar análisis estadístico y dota de multitud de herramientas necesarias para realizar el análisis de series y otros muchos relacionados con la estadística. Este documento trata de realizar una introducción a dicho ambiente de programación, haciendo hincapié en los principales paquetes y funciones necesarias para la realización de un análisis completo por etapas de una serie temporal.

Bibliografía

Casado de Lucas, D. (27 de Enero de 2006). *Protocolo para la identificación de modelos ARIMAen series temporales*. Obtenido de (segun los pasos de Box-Jenkins: http://www.est.uc3m.es/esp/nueva_docencia/leganes/ing_industrial/estadistica_industrial/doc_grupo2/archivos/protocoloARIMA.pdf

Casares, F. (18 de Mayo de 2017). *Paquete Forecast: una revisión a la última actualización*. Obtenido de https://casaresfelix.com/econometria/forecast-con-r/

Coghlan, A. (2015). *A Little Book of R For Time Series.* Cambridge, United Kingdom: Parasite Genomics Group.

García Martos, D., & Espasa, A. (s.f.). *Metodología de Box-Jenkins*. Obtenido de http://www.est.uc3m.es/esp/nueva_docencia/comp_col_get/lade/Econometria_II_NOdocencia/Documentación%20y%20apuntes/TEMA%206_Metodología%20Box-Jenkins.pdf

Gómez, B. (26 de Enero de 2020). *Metodología de Box-Jenkins*. Obtenido de https://rpubs.com/brianz0r/574656

Meca, I., & Agulló, O. (4 de Mayo de 2018). *MODELOS ARIMA*. Obtenido de https://rpubs.com/Meca/386432

Mendoza Vega, J. B. (6 de Noviembre de 2018). *R para principiantes*. Obtenido de https://bookdown.org/jboscomendoza/r-principiantes4/

Pulido San Román, A., & López García, A. M. (1999). *Predicción y simulación aplicada a la economía y gestión de empresas.* Madrid: Ediciones Pirámide.

Uriel Jiménez, E., & Peiro Jimenez, A. (2000). *Introducción al análisis de series temporales.* Valencia: Alfa Centaruro.

Uriel, E. (1985). *Análisis de series temporales modelos ARIMA.* Madrid: Paraninfo S.A.

Velásquez Henao, J. D., Olaya Morales, Y., & Franco Cardona, C. J. (Febrero de 2011). *Análisis y predicción de series de tiempo en mercados de energía usando el lenguaje R*. Obtenido de https://www.redalyc.org/pdf/496/49622372030.pdf

Villavicencio, J. (s.f.). *Indtroducción a Series de Tiempo*. Obtenido de http://www.estadisticas.gobierno.pr/iepr/LinkClick.aspx?fileticket=4_BxecUaZmg%3D